

Manuel d'utilisation du logiciel de calcul par éléments finis Herezh++

Gérard Rio

IDDN.FR.010.0106078.000.R.P.2006.035.20600

2 février 2013

Table des matières

I	Introduction	19
0.1	Présentation	20
0.2	Historique de la conception	20
0.3	Du côté du développeur	21
0.4	Du côté de l'utilisateur	22
II	Généralités et entête	23
1	Entrée des données	24
1.1	Choix de la langue	24
1.2	Commentaires	25
1.3	Ordre sur plusieurs lignes	25
1.4	Inclusion de fichier	26
2	Liste exhaustive des différentes informations que l'on peut trouver dans un fichier .info	26
3	Dimension du problème	27
4	Niveau de commentaire	28
III	Type de problème traité	29

5	Introduction des Algorithmes et listes exhaustives	30
5.1	Liste exhaustive des algorithmes : Galerkin continu et utilitaires	30
5.1.1	Paramètres génériques pour tous les algorithmes	32
5.2	Accélération de convergence	34
5.3	Méthode de Tchamwa-Wielgoz	34
5.4	Méthode classique des différences finies centrées (DFC)	36
5.5	Famille de Newmark et méthode HHT (Hilbert-Hughes-Taylor)	38
5.6	Méthode proposée par Chung-Lee	39
5.7	Méthode proposée par Zhai	39
5.8	Méthode De Runge-Kutta	39
5.9	Utilisation d'Herezh++ comme Umat	40
5.10	Remarques concernant les conditions limites et initiales	46
6	Relaxation dynamique	46
6.1	Utilisation d'algorithmes classiques avec de l'amortissement cinétique	47
6.2	Utilisation d'algorithmes classiques avec de l'amortissement visqueux critique	48
6.3	Algorithme de Relaxation dynamique	49
6.3.1	Cas d'une relaxation avec amortissement cinétique	50
6.3.2	Cas d'une relaxation avec amortissement visqueux	50
6.3.3	Paramètres de contrôle de la masse	51
6.3.4	Extension de la méthode de Barnes	52
6.3.5	Pseudo-masse fonction de la raideur réelle	56
6.3.6	Contrôle du re-calcul de la masse	57
6.3.7	Contrôle matrice viscosité critique	57
6.3.8	Contrôle mode debug	58
6.4	Critère d'arrêt en résidu en dynamique explicite	58
7	"Utilitaires"	59
7.1	Transformation d'un maillage quadratique incomplet en quadratique complet	60
7.2	Relocalisation des noeuds intermédiaires pour les arrêtes des éléments qua- dratiques	61
7.3	Sauvegarde des maillages en cours aux formats .her et .lis	61
7.4	Suppression des noeuds non référencés	61
7.5	Renumérotation des noeuds	62
7.6	Orientation des éléments	62
7.6.1	Cas des éléments membranes, plaques et coques	63
7.7	Création d'un maillage SFE	65
7.8	Fusion de noeuds très voisins	66
7.9	Fusion d'éléments superposés	66
7.10	Fusion de maillages	67
8	"Information"	68
8.1	Définition de références pour un maillage existant	68
9	Galerkin discontinu	71
9.1	Algorithme explicite temporelle de Bonelli	71

IV	Maillages	74
10	Outils de création de maillage	75
11	Liste des informations contenues dans un maillage et syntaxe	76
12	Mouvements solides	78
13	Suppression des noeuds non référencés	79
14	Renumérotation des noeuds	80
15	Fusion de noeuds très voisins	81
16	Fusion d'éléments superposés	81
17	Fusion de maillages	82
18	Références de noeuds, de faces, d'arêtes et d'éléments	82
18.1	Création automatique de références de frontières	84
19	Eléments disponibles	84
19.1	POUT BIE1 : élément barre	85
19.2	TRIANGLE LINEAIRE	86
19.3	TRIANGLE quadratique (QUADRACOMPL)	86
19.4	TRIANGLE CUBIQUE	88
19.5	Triangle axisymétrique linéaire : TRIA_AXI LINEAIRE	89
19.6	Triangle axisymétrique quadratique : TRIA_AXI QUADRACOMPL	90
19.7	SFE1 : élément coque	91
19.8	SFE2 : élément coque	93
19.9	SFE3 et SFE3C : élément coque	94
19.10	SFE3 _cmjpti	96
19.11	Quadrangle linéaire : QUADRANGLE LINEAIRE	96
19.12	Quadrangle quadratique incomplet : QUADRANGLE QUADRATIQUE	97
19.13	Quadrangle quadratique complet : QUADRANGLE QUADRACOMP	98
19.14	Quadrangle cubique complet : QUADRANGLE CUBIQUE	99
19.15	Quadrangle axisymétrique linéaire : QUAD_AXI LINEAIRE	100
19.16	Quadrangle axisymétrique quadratique incomplet : QUAD_AXI QUADRATIQUE	101
19.17	Quadrangle axisymétrique quadratique complet : QUAD_AXI QUADRACOMP	103
19.18	Quadrangle axisymétrique cubique complet : QUAD_AXI CUBIQUE	104
19.19	Hexaédres linéaires : HEXAEDRE LINEAIRE	105
19.20	Hexaédres quadratique incomplet : HEXAEDRE QUADRATIQUE	106
19.21	Hexaédres quadratique complet : HEXAEDRE QUADRACOMPL	108
19.22	Pentaèdre linéaires : PENTAEDRE LINEAIRE	109
19.23	Pentaèdre quadratique incomplet : PENTAEDRE QUADRATIQUE	111

19.24	Pentaèdre quadratique complet : PENTAEDRE QUADRACOMPL	113
19.25	Tétraèdre linéaire : TETRAEDRE LINEAIRE	115
19.26	Tétraèdre quadratique : TETRAEDRE QUADRATIQUE	116
19.27	Eléments de référence s'appuyant sur une géométrie 1D	118
19.28	Position des points d'intégrations pour les éléments de géométrie 1D	118
19.29	Eléments de référence à géométrie surfacique 2D triangulaire	119
19.30	Position des points d'intégrations pour les éléments de géométrie 2D triangulaires	119
19.31	Eléments de référence à géométrie surfacique 2D quadrangulaire	120
19.32	Position des points d'intégrations pour les éléments de géométrie 2D quadrangulaires	122
19.33	Eléments de référence à géométrie 3D hexaédrique	122
19.34	Eléments de référence à géométrie 3D pentaédrique	126
19.35	Eléments de référence à géométrie 3D tétraédrique	129
20	Positions des points d'intégrations	131
20.1	Position des points d'intégrations pour les éléments hexaédriques	131
20.2	Position des points d'intégrations pour les éléments pentaédriques	131
20.3	Position des points d'intégrations pour les éléments tétraédriques	132
21	Remarques sur le nombres de points d'intégration	133
22	Gestion des modes d'hourglass	133
23	Remarques sur la prise en compte de la variation d'épaisseur pour les éléments 2D	134
24	Remarques sur la prise en compte de la variation de section pour les éléments 1D	135
V	Courbes	136
25	Introduction et utilisation de courbe 1D : "liste de courbes 1D"	137
VI	Lois de comportement	138
26	Lois de comportement : généralités	139
27	Type de déformation	140
27.1	Exemple de déclaration pour intégrer la dilatation thermique dans un calcul mécanique	141
28	Liste de lois mécaniques et thermo-mécaniques disponibles	143
28.1	Lois iso-élastiques linéaires	146
28.1.1	ISOELAS1D	146

28.1.2	ISOELAS2D_D	147
28.1.3	ISOELAS2D_C	148
28.1.4	ISOELAS	148
28.2	Lois isotropes élastiques non-linéaires	150
28.2.1	ISO_ELAS_ESPO1D	150
28.2.2	ISO_ELAS_ESPO3D	151
28.2.3	ISO_ELAS_SE1D	151
28.3	Lois Hyper-élastiques	153
28.3.1	ISOHYPER3DFAVIER3	153
28.3.2	ISOHYPERBULK3	154
28.3.3	MOONEY_RIVLIN_1D	155
28.3.4	MOONEY_RIVLIN_3D	157
28.3.5	ISOHYPER3DORGEAS1	158
28.3.6	ISOHYPER3DORGEAS2	162
28.3.7	POLY_HYPER3D	162
28.3.8	HART_SMITH3D	164
28.4	Lois élasto-plastiques.	168
28.4.1	PRANDTL_REUSS1D	168
28.4.2	PRANDTL_REUSS2D_D	169
28.4.3	PRANDTL_REUSS	169
28.5	Lois visco-élastiques isotropes.	170
28.5.1	NEWTON1D	170
28.5.2	NEWTON2D_D	171
28.5.3	NEWTON3D	171
28.5.4	MAXWELL1D	172
28.5.5	MAXWELL2D_C	173
28.5.6	MAXWELL3D	173
28.6	Composition additive de différentes lois de base (ex : loi des mélanges).	179
28.6.1	LOI_ADDITIVE_EN_SIGMA	179
28.6.2	LOI_DES_MELANGES_EN_SIGMA	180
28.7	Passage en déformations et contraintes planes pour une loi 3D quelconque	186
28.7.1	LOI_DEFORMATIONS_PLANES	186
28.7.2	LOI_CONTRAINTES_PLANES	186
28.8	Lois d'hystérésis.	189
28.8.1	HYSTERESIS_1D	189
28.8.2	HYSTERESIS_3D	190
28.9	Lois hypo-élastiques	199
28.9.1	HYPO_ELAS3D	199
28.9.2	HYPO_ELAS2D_C	199
28.10	Lois qui ne font rien!	203
28.10.1	LOI_RIEN1D	203
28.10.2	LOI_RIEN2D	203
28.10.3	LOI_RIEN3D	203
28.11	Loi externe de type Umat	205

29	Liste de lois thermo-physiques disponibles	213
29.1	Lois isotropes thermiques	213
29.2	Loi simple isotrope 1D 2D 3D	213
29.3	Loi de Tait 1D 2D 3D	214
29.4	Loi d'Hoffman d'évolution de la cristalinité	219
29.5	Loi d'Hoffman2 d'évolution de la cristalinité	219
30	Liste de lois de frottement disponibles	220
30.1	Loi de Coulomb	220
VII	Informations particulières liées au contexte	225
31	Stockages divers	226
31.1	Définition de l'épaisseur d'un groupe d'éléments	227
31.2	Définition de la section d'un groupe d'éléments	227
31.3	Définition de la largeur d'un groupe d'éléments	227
31.4	Définition de la masse volumique d'un groupe d'éléments	228
31.5	Définition des masses additionnelles	228
VIII	Conditions de contact	229
32	Introduction	230
32.1	Introduction des notions de maillages esclaves et de maillages maîtres	230
32.2	Auto-contact	231
32.3	Introduction explicite des zones présumées de contact	231
32.4	Paramètres de gestion de la méthode de contact	232
IX	Types d'efforts	233
33	Conditions de chargement imposé	234
33.1	Mot clé et exemple du chargement ponctuel	234
33.1.1	Chargement particulier suivant une courbe de charge	235
33.1.2	Mise en place d'un temps mini et/ou d'un temps maxi	235
33.1.3	Particularité de la mise en place d'un chargement actif pour $t=0$	235
33.2	Différent type de chargement	236
33.2.1	Chargement volumique	236
33.2.2	Chargement linéique	236
33.2.3	Chargement : densité linéique qui suit l'évolution de la frontière	237
33.2.4	Chargement : densité d'effort dont la direction reste fixe	237
33.2.5	Chargement de type pression	238
33.2.6	Chargement : densité d'effort dont la direction suit l'évolution des frontières	238
33.2.7	Chargement hydrostatique	239

33.2.8	Chargement aéro-hydrodynamique	239
33.2.9	Cas particulier de chargement avec des éléments axisymétriques . . .	241

X Conditions limites en déplacements 243

34 Conditions limites pour les degrés de liberté (déplacements, positions . . .) 244

34.1	Chargement particulier suivant une courbe de charge	245
34.2	Mise en place d'un temps mini et d'un temps maxi pour l'application des conditions	246
34.3	Notion de données et de variables	246
34.4	Prise en compte d'un champ de valeurs	247
34.5	Cas d'un ddl imposé en initialisation et au cours du temps : exemple de la température	247
34.6	Cas de la mise en place d'un mouvement solide	248
34.7	Condition de symétrie et encastrement pour les SFE	249
34.7.1	Direction de tangente imposée	249
34.7.2	Encastrement imposé	250
34.8	Cas particulier de géométries et chargements axisymétrique	250

35 Conditions limites linéaires (CLL) entre degrés de liberté (déplacements, positions . . .) 251

35.1	Déplacement ou positionnement dans un plan (3D) ou sur une droite (2D)	251
35.2	Condition linéaire générale	253
35.2.1	Particularités liées aux noms de maillage	256
35.3	Conséquences des CLL sur le stockage matriciel (largeur de bande)	257

36 Conditions initiales 258

36.1	Particularité de l'initialisation des positions par rapport aux autres degrés de libertés	258
36.2	Conditions de symétrie ou d'encastrement initiale	259

XI Chargement global 260

37 Algorithme de chargement 261

XII Paramètres de contrôles et de pilotage 265

38 Résolution : introduction 266

39 Contrôle général 266

40 Dynamique 271

41	Calculs des énergies	274
42	Résolution des systèmes linéaires	276
43	Pilotage de la résolution globale	279
44	Pilotage du contact	282
45	Affichage des résultats	285
46	Calculs géométriques	287
XIII	Sortie des résultats	289
47	Sortie des résultats	290
47.1	Introduction	290
47.2	Sortie des résultats sur fichiers	290
47.2.1	Aucune sortie demandée	292
47.3	Sortie des résultats pour une visualisation graphique directe.	292
47.3.1	Utilisation des fichiers de commande	293
47.3.2	Formats vrml	300
47.3.3	Formats Maple : introduction	302
47.3.4	Cas des grandeurs aux noeuds pour le format maple	304
47.3.5	Cas des grandeurs de type degré de liberté, aux éléments pour le format maple	306
47.3.6	Cas des grandeurs de type spécifique, aux éléments pour le format maple	307
47.3.7	Cas des grandeurs globales	308
47.3.8	Cas des torseurs de réactions	308
47.3.9	Remarques concernant l'ordre de sortie des composantes des grandeurs tensorielles dans le cas du format Maple	309
47.3.10	Formats geomview	310
47.3.11	Formats Gid	310
47.3.12	Formats gmsh et comparaison avec le format Gid	311
47.4	Sortie des résultats au fil du calcul	313
47.5	Exportation des grandeurs des points d'intégrations aux noeuds	314
47.6	Significations des grandeurs disponibles	315
47.6.1	Grandeurs liés à la cinématique	315
47.6.2	Grandeurs liés aux contraintes	316
47.6.3	Grandeurs liées aux énergies	316
47.7	Remarque concernant les contraintes et déformations pour les membranes, plaques et coques	316
47.8	Remarque concernant le volume délimité par des membranes, plaques et coques	317

XIV Utilitaires 318

48 Utilitaires 319

48.1 Fonctions 1D 319

48.1.1 Fonction poly-linéaire 319

48.1.2 Fonction polylineaire-simple 320

48.1.3 Fonction $f(x) = \gamma + \alpha(|x|^n)$ 321

48.1.4 Fonction $f(x) = \frac{c}{2}(1 - \cos(\frac{(x-a)\Pi}{(b-a)}))$ 321

48.1.5 Fonction polynomial $f(x) = \sum_{i=1}^n a_i x^i$ 321

48.1.6 Fonctions composées : $f(x) = f1 \circ f2(x)$ 321

48.1.7 Fonctions composées : $f(x) = f1(x) + f2(x)$ 322

48.1.8 Fonctions cycliques : amplification multiplicative 323

48.1.9 Fonctions cycliques : amplification additive 323

48.1.10 Fonctions réunion de domaine 325

48.1.11 Fonctions $f(x) = (1. + \gamma \cos(3 x))^{(-n)}$ 325

48.1.12 Fonctions $f(x) = (1. + \gamma (\cos(3 x))^2)^{(-n)}$ 325

48.1.13 Fonctions $f(x) = (\gamma + \alpha x)^n$ 325

48.1.14 Fonctions $f(x) = (\gamma + \alpha x^2)^n$ 326

48.1.15 Fonctions $f(x) = (a - b) \exp(-c x) + b$ 328

48.1.16 Fonctions $f(x) = e \cos(\alpha x + \beta)$ 328

48.1.17 Fonctions $f(x) = e \sin(\alpha x + \beta)$ 329

48.1.18 Fonction $f(x) = a + b \tanh((x - c)/d)$ 330

XV Chronologie et historique des mises à jour 331

49 Introduction 332

50 Liste exhaustive 332

Bibliographie 338

Liste des tableaux

1	liste de mots clés	25
2	liste des type de problèmes	31
3	liste des type de paramètres génériques pouvant être associés aux paramètres particuliers de l’algorithme	32
4	Exemple d’utilisation du mode ”debug”, de l’amortissement critique, et d’un arrêt sur l’équilibre statique, ceci avec un algorithme de relaxation dynamique	33
5	Exemple de paramètres pour mettre en oeuvre et conduire la méthode d’accélération de convergence	34
6	Exemple de de l’algorithme de Tchamwa avec une fonction de modération sur φ	35
7	Exemple de de l’algorithme de Tchamwa avec une fonction de modération sur φ et la prise en compte d’une moyenne de l’accélération	37
8	Exemple de déclaration des paramètres de pilotage du Runge-Kutta	40
9	déclaration de la fonction Umat fortran : partie fortran	42
10	déclaration de la fonction Umat fortran : partie en langage C	43
11	entête des routines C appelées par la subroutine fortran	44
12	exemple de l’algorithme permettant d’utiliser Herezh++ en tant qu’Umat	45
13	Exemple de déclaration d’un amortissement cinétique avec l’algorithme de Tchamwa	48
14	Exemple de déclaration d’un amortissement visqueux avec l’algorithme de Tchamwa	49
15	Exemple de déclaration de paramètres de l’algorithme de relaxation dynamique avec amortissement cinétique	51
16	Exemple de déclaration de paramètres de l’algorithme de relaxation dynamique avec amortissement visqueux	52
17	Exemple de déclaration pour l’algorithme de relaxation dynamique avec un contrôle sur le résidu	54
18	Exemple de déclaration pour l’algorithme de relaxation dynamique avec un contrôle sur le déplacement	55
19	Exemple de déclaration pour l’algorithme de relaxation dynamique avec utilisation du mode debug	59
20	Exemple d’utilisation de l’algorithme ”informations” pour définir de nouvelles références.	69
21	Exemple d’utilisation de l’algorithme de Bonelli.	73
22	Exemple de déclaration de deux mouvements solides : une translations suivant z et une rotation autour de l’axe y	79
23	Exemple de déclaration d’un mouvement solide de rotation autour d’un noeud	79
24	Exemple de d’utilisation des méthodes de suppression des noeuds non référencés et de renumérotation	80
25	Exemple de d’utilisation de la méthode de suppression des noeuds voisins d’une distance inférieure à 0.001	81

26	Exemple de d'utilisation de la méthode de suppression des éléments superposés	82
27	Exemple de l'utilisation successive de deux opérations de fusion de maillage, intégrant des déplacements solides pendant et après les opérations.	83
28	liste d'élément finis mécaniques	85
29	Exemple de déclaration d'un élément biellette	86
30	Exemple de déclaration d'un élément triangle avec interpolation linéaire	87
31	Exemple de déclaration d'un élément triangle avec interpolation quadratique	87
32	Exemple de déclaration d'un élément triangle avec interpolation quadratique, et avec 3 points d'intégrations strictement interne à l'élément	88
33	Exemple de déclaration d'un élément triangle avec interpolation quadratique et plusieurs type d'intégration	88
34	Exemple de déclaration de deux éléments triangles axisymétriques avec interpolation linéaire	89
35	Exemple de déclaration d'un élément triangle axisymétriques avec interpolation quadratique et différents nombres de points d'intégration	90
36	numérotation des éléments SFE1	92
37	Exemple de déclaration d'éléments SFE1	93
38	Exemple de déclaration d'éléments SFE2	94
39	Exemple de déclaration d'éléments SFE3 (pour l'élément SFE3C, il suffit de changer SFE3 par SFE3C)	95
40	Exemple de déclaration de deux éléments quadrangles avec interpolation linéaire et plusieurs types de nombres de points d'intégration	97
41	Exemple de déclaration d'un élément quadrangle avec interpolation quadratique incomplete	98
42	Exemple de déclaration d'un élément quadrangle avec interpolation quadratique complete et 2 types de nombres de points d'intégration	99
43	Exemple de déclaration d'un élément quadrangle avec interpolation quadratique complete et plusieurs types de nombres de points d'intégration	100
44	Exemple de déclaration de deux éléments quadrangles axisymétriques avec interpolation linéaire et plusieurs types de nombres de points d'intégration	101
45	Exemple de déclaration d'un élément quadrangle axisymétriques avec interpolation quadratique incomplete	102
46	Exemple de déclaration d'un élément quadrangle axisymétriques avec interpolation quadratique complete et plusieurs types de nombres de points d'intégration	103
47	Exemple de déclaration d'un élément quadrangle axisymétriques avec interpolation cubique et plusieurs types de nombres de points d'intégration	104
48	Exemple de déclaration d'éléments hexaédrique avec interpolation linéaire	105
49	Exemple de déclaration d'éléments hexaédriques linéaire avec différents nombres de points d'intégration	106
50	Exemple de déclaration d'éléments hexaédrique avec interpolation quadratique incomplète	106
51	Exemple de déclaration d'éléments hexaédriques quadratique incomplet avec différents nombres de points d'intégration	107

52	Exemple de déclaration d'éléments hexaédrique avec interpolation quadratique complète	108
53	Exemple de déclaration d'éléments hexaédriques quadratique avec différents nombres de points d'intégration	109
54	Exemple de déclaration d'éléments pentaédriques avec interpolation linéaire	110
55	Exemple de déclaration d'éléments pentaédriques linéaire avec 6 points d'intégration	110
56	Exemple de déclaration d'éléments pentaédriques avec interpolation quadratique incomplet	111
57	Exemple de déclaration d'éléments pentaédriques quadratique avec différents nombres de points d'intégration	112
58	Exemple de déclaration d'éléments pentaédriques avec interpolation quadratique complet	114
59	Exemple de déclaration d'éléments pentaédriques quadratique complet avec différents nombres de points d'intégration	114
60	Exemple de déclaration d'éléments tétraèdre avec interpolation linéaire . .	116
61	Exemple de déclaration d'éléments tétraédriques avec interpolation quadratique	117
62	numérotation des éléments 1D de référence	118
63	numérotation des éléments triangulaires de référence	119
64	numérotations des éléments de référence quadrangulaires	121
65	numérotation de l'élément de référence hexaédrique linéaire	123
66	numérotation de l'élément de référence pour les hexaèdres quadratiques incomplet	124
67	numérotation de l'élément de référence pour les hexaèdres quadratiques complets	125
68	numérotation de l'élément pentaédrique linéaire de référence.	126
69	Numérotation de l'élément pentaédrique quadratique incomplet de référence.	127
70	Numérotation de l'élément pentaédrique quadratique complet de référence.	128
71	Numérotation de l'élément tétraédrique linéaire de référence.	129
72	Numérotation de l'élément tétraédrique quadratique de référence.	130
73	exemple de correspondance de numéro de point d'intégration pour un pentaèdre, entre la numérotation globale et la numérotation du triangle et du segment associé	131
74	Positions de points d'intégration pour les éléments tétraédriques.	132
75	Exemple de déclaration de contrôle d'hourglass à l'aide d'un comportement matériel simple, et d'un élément interne à intégration complète	134
76	Exemple de déclaration d'une liste de courbes 1D.	137
77	Exemple de déclaration d'un nom de loi de comportement associée à une référence, ceci dans le cas d'un maillage.	139
78	Exemple de déclaration d'un nom de loi de comportement associée à une référence, ceci dans le cas de plusieurs maillages.	139
79	Exemple de déclaration de loi de comportement associée à deux matériaux.	140
80	Exemple de déclaration d'une loi de comportement élastique associée à une déformation logarithmique.	141

81	Exemple de déclaration d'une loi de comportement mécanique associée à une loi de comportement thermo-physique en vue d'intégrer la dilatation thermique dans le calcul mécanique.	142
82	liste des différents lois	144
83	suite de la liste des différents lois	145
84	liste des différents lois isotropes élastiques disponibles	146
85	Exemple de déclaration de la loi élastique 1D dont le module d'Young dépend de la température selon une courbe indiquée explicitement.	147
86	Exemple de déclaration de la loi élastique 1D dont le module d'Young dépend de la température selon une courbe repérée par un nom de référence.	147
87	Exemple de déclaration de la loi élastique 3D dont le module d'Young dépend de la température selon une courbe repérée par un nom de référence.	149
88	liste des différents lois isotropes élastiques non linéaires disponibles	150
89	Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_ESPO1D	150
90	Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_ESPO3D	151
91	Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_SE1D, ceci dans le cas d'un comportement symétrique et de l'utilisation d'une courbe déjà existante.	152
92	Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_SE1D, cas d'un comportement non symétrique et définition directe de la courbe d'évolution	152
93	liste des différents lois isotropes hyper-élastiques disponibles	153
94	Exemple de déclaration de la loi hyperélastique ISOHYPER3DFAVIER3 sans phase.	154
95	Exemple de déclaration de la loi hyperélastique ISOHYPER3DFAVIER3 avec phase.	154
96	Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3	155
97	Exemple de déclaration de la loi de Mooney Rivlin en 1D.	155
98	Exemple de déclaration de la loi de Mooney Rivlin en 1D, avec les deux paramètres thermo-dépendants.	156
99	Exemple de déclaration de la loi de Mooney Rivlin en 1D, avec le second paramètre thermo-dépendant, le premier étant fixe.	156
100	Exemple de déclaration de la loi de Mooney Rivlin en 3D.	157
101	Exemple de déclaration de la loi de Mooney Rivlin en 3D, dans le cas où les trois coefficients matériaux sont thermo-dépendants.	158
102	Exemple de déclaration de la loi d'hyper-élasticité Orgéas 1.	159
103	Exemple de déclaration de la loi d'hyper-élasticité Orgéas 1 avec dépendance à la phase.	160
104	Exemple de déclaration de la loi d'hyper-élasticité Orgéas avec dépendance à la phase et une dépendance à la température du paramètre Q_{0s} selon l'évolution proposée par Laurent Orgéas dans sa thèse.	161
105	Exemple de déclaration de la loi d'hyper-élasticité Orgéas avec dépendance à la phase et une dépendance à la température du paramètre Q_{0s} selon une évolution donnée par la courbe courbe_evol_Q0s.	161

106	Exemple de déclaration de la loi d'hyper-élasticité Orgéas avec dépendance à la phase et une dépendance à la température des paramètre Q_{0s} selon une évolution donnée par la courbe <code>courbe_evol_Q0s.</code> , et Q_{0e} selon la loi interne	162
107	Exemple de déclaration de la loi d'hyper-élasticité Orgéas avec dépendance à la phase selon des fonctions courbes	163
108	Exemple de déclaration de la loi polynomiale en 3D.	164
109	Exemple de déclaration de la loi polynomiale en 3D, dans le cas où les trois coefficients matériaux sont thermo-dépendants.	165
110	Exemple de déclaration de la loi de hart-smith3D en 3D.	165
111	Exemple de déclaration de la loi de Hart Smith en 3D, dans le cas où les quatre coefficients matériaux sont thermo-dépendants.	166
112	Exemple de déclaration de la loi de Hart Smith en 3D avec un terme additionnelle dans le potentiel, permettant de décrire un raidissement en fin de chargement.	167
113	Exemple de déclaration de la loi de Hart Smith en 3D avec un terme additionnelle dans le potentiel, permettant de décrire un raidissement en fin de chargement : ici les paramètres a et r sont thermo-dépendants	167
114	liste des différents lois isotropes élasto-plastiques disponibles	168
115	Exemple de déclaration de la loi élasto-plastique de Prandtl Reuss en 1D .	168
116	liste des différents lois visco-élastiques isotropes disponibles	170
117	Exemple de déclaration d'une loi de Newton en 1D	170
118	Exemple de déclaration d'une loi de Newton en 1D dont le coefficient μ dépend de la température	171
119	Exemple de déclaration d'une loi de Maxwell en 1D	172
120	Exemple de déclaration d'une loi de Maxwell en 1D dont les coefficients dépendent de la température au travers de courbes définies explicitement .	172
121	Exemple de déclaration d'une loi de Maxwell en 1D dont les coefficients dépendent de la température au travers de courbes définies explicitement .	173
122	Exemple de déclaration d'une loi de Maxwell en 2D contraintes planes . . .	174
123	Exemple de déclaration d'une loi de Maxwell en 3D, dont les paramètres E et μ dépende de la déformation au sens de Mises	175
124	Exemple de déclaration d'une loi de Maxwell en 3D	176
125	Exemple de déclaration d'une loi de Maxwell en 3D avec une viscosité non linéaire	176
126	Exemple de déclaration d'une loi de Maxwell en 3D avec les deux viscosités dépendantes d'une courbe de température "mu1dT" défini par ailleurs dans le fichier <code>.info</code>	177
127	Exemple de déclaration d'une loi de Maxwell en 3D avec une viscosité dépendante de la cristallinité	178
128	liste des différentes compositions additives disponibles de lois élémentaires	179
129	Exemple de déclaration d'une loi additive en contrainte	179
130	Exemple de déclaration d'une loi additive en contrainte	182
131	Exemple de déclaration d'une loi additive en contrainte avec une proportion sur les accroissements des contraintes (et non sur la contrainte totale) . .	183

132	Exemple de déclaration d'une loi additive en contrainte, dans le cas ou on utilise une proportion directement accessible au point d'intégration (donc non interpolée à partir de grandeurs aux noeuds)	184
133	Liste exhaustive des différentes grandeurs susceptibles d'être disponible aux noeuds	184
134	exemple de loi des mélanges pilotée par la trace des contraintes, la loi de mélange appelle elle même deux lois additives	185
135	exemple de loi de déformations planes intégrant un comportement 3D et des conditions de déformations planes	187
136	Exemple de loi de contraintes planes intégrant un comportement 3D et des conditions de contraintes planes. Dans cette exemple, la prise en compte de la condition de contraintes planes s'effectue par perturbation.	188
137	liste des différents lois d'élasto-hystérésis	189
138	Exemple de déclaration d'une loi d'hystérésis 1D	190
139	Exemple de déclaration d'une loi d'hystérésis 1D avec paramètres de réglage pour une méthode de résolution de l'équation constitutive par linéarisation	191
140	Exemple de déclaration d'une loi d'hystérésis 1D avec paramètres de réglage pour une méthode de résolution Runge-Kutta	191
141	valeur par défaut des paramètres de réglage du calcul de l'hystérésis 3D	194
142	Exemple de déclaration d'une loi d'hystérésis 3D	195
143	Exemple de déclaration d'une loi d'hystérésis 3D avec des paramètres de contrôle de l'algorithme de résolution	195
144	Exemple de déclaration d'une loi d'hystérésis 3D avec des paramètres matériau thermo-dépendants	196
145	Exemple de déclaration d'une loi d'hystérésis 3D avec un premier paramètre matériau fixe puis les autres thermo-dépendants	196
146	Exemple de déclaration d'une loi d'hystérésis 3D avec paramètres de réglage pour une méthode de résolution Runge-Kutta	196
147	liste des différents lois isotropes élastiques disponibles	199
148	Exemple de déclaration de la loi hypo-élastique linéaire 3D.	200
149	Exemple de déclaration d'une loi hypo-élastique 3D dont les coefficients dépendent de la température selon une courbe repérée par un nom de référence.	201
150	Exemple de déclaration d'une loi hypo-élastique 3D dont la compressibilité provient d'une loi thermo-physique	202
151	Exemple de déclaration d'une loi hypo-élastique 2D en contraintes planes	202
152	liste des différents lois qui ne font rien, disponibles	203
153	Exemple de déclaration d'une loi 1D ne faisant rien mécaniquement.	203
154	Exemple de déclaration d'une loi 2D ne faisant rien mécaniquement.	203
155	Exemple de déclaration d'une loi 3D ne faisant rien mécaniquement.	204
156	Exemple de déclaration d'une loi Umat utilisable par Abaqus.	205
157	Exemple de déclaration d'une loi Umat utilisé en interne, pour vérification.	206
158	Exemple de déclaration d'une loi Umat avec redéfinition des pipes.	206
159	Définition du stockage des informations d'entrée-sortie pour la création d'une Umat externe au programme Herezh	207

160	rappel des parametres de passage de la subroutine fortran utilisable par Abaqus pour	209
161	Procédure de lecture des informations sur le pipes	210
162	procédure d'écriture sur le pipe, dans le cas de la création d'une Umat externe à Herezh	211
163	liste des lois de comportement thermophysique	213
164	Exemple de déclaration d'une loi thermo physique simple dont les coefficients sont constants.	213
165	Exemple de déclaration d'une loi thermo physique simple dont les coefficients sont thermo-dépendants.	214
166	Exemple de déclaration d'une loi thermo physique de Tait.	215
167	Exemple de déclaration d'une loi thermo physique de Tait avec préparation pour le postraitement des résultats.	216
168	Exemple de déclaration d'une loi thermo physique de Tait avec calcul du taux de cristallinité par le modèle d'Hoffman	218
169	Exemple de déclaration d'une loi d'Hoffman permettant le calcul de la cristallinité en fonction de la pression et de la température.	220
170	Exemple de déclaration d'une loi d'Hoffman permettant le calcul de la cristallinité en fonction de la pression et de la température.	221
171	liste des lois disponibles de frottement lié au contact	221
172	Exemple de déclaration d'une loi de frottement classique de Coulomb.	222
173	Exemple de déclaration d'une loi de frottement de Coulomb, avec un coefficient de frottement qui dépend de la vitesse tangentielle.	222
174	Exemple de déclaration d'une loi de frottement de Coulomb régularisée.	224
175	exemple de définition d'épaisseur et de section.	226
176	exemple de définition d'épaisseur et de section dans le cas d'un seul maillage.	226
177	exemple de définition d'épaisseur et de section dans le cas de plusieurs maillages.	227
178	exemple de définition de la largeur d'un groupe d'éléments.	227
179	exemple de définition de la masse volumique d'un groupe d'éléments.	228
180	exemple de définition de masse additionnelle sur un groupe de noeud.	228
181	Exemple de mise en place d'un chargement ponctuel dans le cas d'un seul maillage	234
182	Exemple le mise en place d'un chargement ponctuel dans le cas où il existe plusieurs maillages	234
183	Exemple le mise en place d'un chargement ponctuel avec une courbe de charge, un temps mini et un temps maxi	235
184	liste des différents types de chargement	236
185	Exemple de déclaration d'un chargement volumique	237
186	Exemple de déclaration d'un chargement linéique	237
187	Exemple de déclaration d'un chargement linéique suiveur	237
188	Exemple de déclaration d'un chargement uniforme surfacique	238
189	Exemple de déclaration d'un chargement pression	238
190	Exemple de déclaration d'un chargement de type surfacique directionnelle	239
191	Exemple de déclaration d'un chargement hydrostatique	239

192	Exemple de déclaration d'un chargement hydrodynamique	240
193	Exemple de déclaration des bloquages des degrés de liberté dans le cas d'un seul maillage	244
194	Exemple de déclaration des bloquages des degrés de liberté dans le cas de plusieurs maillages	245
195	Exemple de déclaration de degrés de liberté imposé suivant une courbe de charge, selon une procédure relative $X(t + \Delta t) = X(t) + U(t + \Delta t) - U(t)$	245
196	Exemple de déclaration des bloquages des degrés de liberté dans le cas d'un champ de valeurs fixes	247
197	Exemple de déclaration des bloquages des degrés de liberté dans le cas d'un champ de valeurs déterminées par des courbes de charge	247
198	Exemple de déclaration d'un déplacement imposé par mouvements solides .	248
199	Exemple de déclaration d'un déplacement imposé par mouvements solides .	249
200	Exemple de déclaration d'une direction de tangente imposée à une surface moyenne d'une coque SFE	250
201	Exemple de mise en place de conditions linéaire par projection sur un plan, avec utilisation de fonction de charge et centre fixe	251
202	Exemple de conditions linéaires par projection sur un plan, avec un centre noeud initial et à t	253
203	Exemple de conditions linéaires par projection sur un plan, avec un temps mini et un temps maxi	253
204	Exemple de conditions linéaires entre ddl de plusieurs noeuds	254
205	Exemple de conditions linéaires avec l'utilisation de fonctions de charge . .	255
206	Exemple de conditions linéaires entre noeuds de maillages différents	257
207	Exemple de mise en place de conditions initiales dans le cas d'un seul maillage	258
208	Exemple de mise en place de conditions initiales dans le cas de plusieurs maillages	259
209	liste des algorithmes de chargement	261
210	liste des sous mots clés associés aux paramètres de contrôle	267
211	liste des sous mots clés associés aux paramètres de contrôle liés à la dynamique	272
212	liste des sous mots clés associés aux paramètres de calcul d'énergie	275
213	liste des sous mots clés associés aux paramètres de contrôle liés à la dynamique	277
214	liste des sous mots clés associés aux paramètres de contrôle du pilotage . .	280
215	liste des sous mots clés associés aux paramètres de contrôle liés à la dynamique	283
216	liste des sous mots clés associés aux paramètres de contrôle du pilotage . .	286
217	liste des sous mots clés associés aux paramètres de contrôle du pilotage . .	287
218	liste des différentes options de post-traitement graphique, à indiquer après le type de calcul	293
219	Exemple d'en-tête de fichier .maple	305
220	liste des fonctions 1D	319
221	Exemple de déclaration d'une fonction d'écroissage plastique poly-linéaire	320
222	Exemple de déclaration d'une courbe poly-linéaire avec translations initiales	320

223	Exemple de déclaration d'une courbe poly-linéaire simplifiée	321
224	Exemple de déclaration d'une fonction de type $\sigma = \gamma - \alpha x ^n$	321
225	Exemple de déclaration d'une fonction de type $f(x) = \frac{c}{2}(1 - \cos(\frac{(x-a)\Pi}{(b-a)}))$	321
226	Exemple de déclaration d'une fonction polynomiale $f(x) = \sum_{i=1}^n a_i x^i$	322
227	Exemple de déclaration d'une fonction composée $f(x) = f1of2(x)$. Les 2 fonctions sont définis par un nom qui est une référence à des fonctions définis par ailleurs dans la liste des courbes déjà définis.	322
228	Exemple de déclaration d'une fonction composée $f(x) = f1of2(x)$. Les 2 fonctions sont définis ici définis explicitement.	322
229	Exemple de déclaration d'une fonction composée $f(x) = f1(x) + f2(x)$. Les 2 fonctions sont définis par un nom qui est une référence à des fonctions définis auparavant.	323
230	Exemple de déclaration d'une fonction cyclique avec un facteur d'amplification multiplicatif.	324
231	Exemple de déclaration d'une fonction cyclique avec un facteur d'amplification additif.	326
232	Exemple de déclaration d'une fonction union permettant de globaliser plusieurs fonctions déjà définis.	327
233	Exemple de déclaration d'une fonction de type $f(x) = (1. + \gamma \cos(3 x))^{(-n)}$	327
234	Exemple de déclaration d'une fonction de type $f(x) = (1. + \gamma (\cos(3 x))^2)^{(-n)}$	327
235	Exemple de déclaration d'une fonction de type $f(x) = (\gamma + \alpha x)^n$	328
236	Exemple de déclaration d'une fonction de type $f(x) = (\gamma + \alpha x^2)^n$	328
237	Exemple de déclaration d'une fonction de type $f(x) = (a - b) \exp(-c x) + b$	329
238	Exemple de déclaration d'une fonction de type $f(x) = e \cos(\alpha x + \beta)$	329
239	Exemple de déclaration d'une fonction de type $f(x) = e \sin(\alpha x + \beta)$	330
240	Exemple de déclaration d'une fonction de type $f(x) = a + b \tanh((x - c)/d)$	330

Table des figures

Première partie
Introduction

0.1 Présentation

Herezh++ est un code de calcul éléments finis développé comme son nom l'indique en C++, par l'auteur de ce document. L'objectif est principalement de résoudre des problèmes divers de mécanique du solide déformable.

Le nom breton "Herezh" signifie en français héritage, hérédité. Ce nom rappelle une loi de comportement particulière et originale implantée dans Herezh et dédiée aux Alliages à Mémoire de Forme : la loi d'élasto-hystérésis. C'est une loi à structure "héréditaire" qui utilise un concept de "mémoire discrète" ! Herezh a donc été adopté, vu l'importance de cette loi pour l'auteur (et d'autres collègues !) et également pour rappeler le terroir Breton !

Beaucoup d'étudiants, utilisant Herezh++ on pris la liberté de l'appeler Thérèse !! je ne sais pas pourquoi...

Quelques mots clés :

- * Dédié à la simulation du comportement mécanique des solides déformables 1D, 2D, 3D. De nombreux éléments classiques sont disponibles.
- * Eléments coques SFE (sans degré de liberté de rotation) : ces éléments sont originaux,
- * Calcul par éléments finis en transformations finies : grands déplacements, grandes déformations.
- * Calcul statique, transitoire, dynamique (rapide). De nombreux algorithmes sont disponibles, certains classiques (DFC, Newmark ...) d'autres moins (Tchamwa, Tsai, Chung-Lee, Galerkin-Discontinu ...).
- * Lois de comportements diverses en 1D, 2D, 3D : hypo-élastique, élastique, hyper-élastique, visco-élastique, élastoplastique ...
- * Prise en compte de la dépendance thermique (loi de comportement, dilatation, couplage).
- * Couplage avec le code industriel Abaqus.
- * Interfaçage avec de nombreux outils en pré et post-traitement :
 - **Stamm** : un mailleur élémentaire, développé avec herezh++, permettant de générer très simplement et rapidement des maillages de tests en 1D, 2D et 3D,
 - **GID** : un pré et post-processeur éléments finis commercial permettant de construire les maillages et d'exploiter les résultats obtenus par Herezh++
 - **GMSH** : un pré et post-processeur éléments finis universitaire, permettant comme GID de construire les maillages.
 - **Gnuplot** : Herezh++ peut générer facilement des tableaux de valeurs de grandeurs particulières, qui peuvent ensuite être exploitées par des "grappeurs" tels que Gnuplot, Xmgrace, Excel ...
 - **navigateur Web** : Herezh++ peut générer des résultats graphiques aux formats vrml, qui peuvent ensuite être exploités sous un navigateur Web quelconque munis d'un Plugin vrml
 - **Geomview** : une sortie graphique exploitable par le logiciel libre Geomview est également disponible.

0.2 Historique de la conception

Le développement du code de calcul élément fini `HEREZH++` a débuté principalement durant 1996-97, à Berkeley, Université de Californie, au cours d'un séjour sabbatique. L'expérience du développement d'Herezh (1990-1998), effectué en Fortran77, où de nombreux collègues ont été impliqués : montre de nombreuses difficultés à mesure que le code intègre de nouvelles possibilités c'est à dire à mesure que l'on s'éloigne des données initiales du programme. L'objectif du projet Herezh++ a été alors la conception d'une structure informatique souple qui permet de dépasser ces limites traditionnelles du Fortran tout en tentant de concerver une bonne efficacité à l'exécution.

Le choix du C++ s'inscrit dans cette logique. La structure du code actuelle, Herezh++, doit permet d'intégrer les objectifs suivants : . multi-domaines fluide - solides déformables ou non, . loi de comportement attachées au domaines, quelconques : mécanique, thermique .. . prise en compte d'interactions interdomaines : en contact ou à distance, . interaction avec des logiciels externes, . intégration d'algorithmes quelconques, statique, transitoire, dynamique implicite ou explicite ... La le codage est entièrement nouveau, en particulier aucune routine Fortran n'a été reprise, et la structuration a été entièrement pensée et réalisé en "objets".

Le code fonctionne, et les nombreuses mises à jour montrent la bonne flexibilité des structures informatiques retenues.

0.3 Du coté du développeur

Nous pouvons relever différent points particuliers.

- Création d'une classe spécifique aux tenseurs d'ordre 2 et 4 : les tenseurs en dimensions 1, 2 ou 3 sont dérivés d'une classe virtuelle générique. Un typage différent est utilisé pour les 3 types de coordonnées et il y a une vérification des dimensions en phase de mise au point grâce à des directives de compilation. Toutes les opérations classiques sont surchargées : + += - -= * == != / /= &&, on notera le produit contracté une fois et doublement contracté, ainsi que le calcul des invariants, du tenseur transposé, de l'inverse par rapport au produit contracté Le stockage et l'allocation, pour les tenseurs est également optimisé au travers de l'utilisation des conteneurs classique STL (Standard Template Library) . Cette classe constitue une base appréciable pour les calculs relatifs aux métriques pour les déformations, et aux lois constitutifs, particulièrement dans le cas de coordonnées matériels entrainées.
- Création d'une technique permettant l'introduction de nouveaux éléments avec une modification minimal du code actuel. Au moment de l'édition de lien, l'introduction des nouveaux fichiers correspondant au nouvel élément, met à jour le gestionnaire général d'élément, ceci au travers du mécanisme d'initialisation des variables statiques. Ensuite, l'élément qui doit décliner d'une classe virtuelle générique déjà existante, mécanique thermique ..., est utilisé au travers de la définition particulière des fonctions virtuelle génériques.
- Utilisation systématique de directives de compilation pour différencier les phases de mise au point où l'efficacité du programme n'est pas recherché et les phases d'utilisation où la rapidité est un facteur important.
- Utilisation systématique de classe patron (Template) pour gérer les structures classiques telles que : tableaux et liste comportant des surcharge d'opérateurs. Certaines

- classes templates sont propres à l'étude, d'autres proviennent de la bibliothèque STL.
- Surcharge systématique des opération d'algèbre linéaire.

0.4 Du coté de l'utilisateur

Les éléments disponibles sont :

- un manuel d'utilisation a priori à jour, avec un systèmes d'hyperliens. Ce manuel contient de manière exhaustive, toutes les possibilités offertes par le code. Le 14 sep 2007, le manuel contenait 239 pages. Il est évidemment fortement conseillé de parcourir l'ensemble du document...
- un manuel théorique : qui ne contient actuellement que quelques particularités du logiciel. Les informations classiques sur le fonctionnement d'un code éléments finis sont disponibles dans de nombreux ouvrages. On peut également télécharger les cours d'éléments finis dispensés par l'auteur, sur son site Web. Pour les parties recherches il faut se référer aux publications.
- des informations en ligne : que l'on peut obtenir inter-activement directement avec Herezh++ pendant son fonctionnement.
- Quelques fichiers de tests simple, existants sur le site Web, permettant de commencer un premier calcul. Ensuite, ensuite on trouve dans la documentation un exemple qui fonctionne pour chaque possibilité offerte par Herezh++

Deuxième partie

Généralités et entête

1 Entrée des données

Le programme activé fonctionne à partir d'un fichier de commande. Il demande donc un nom de fichier. Ce fichier qui contient les informations d'entrée nécessaires pour le calcul, doit posséder l'extension .info, exemple : biell.info . On se reportera au fichier biell.info pour avoir un exemple concret de fichier d'entrée.

Il est possible sur le système UNIX de transmettre le nom du fichier au moment de l'appel du programme. Ainsi la commande :

```
HZpp -f biell
```

indique au programme herezh "HZPP" que le fichier de commande a pour nom "biell.info". Notez qu'il ne faut pas indiquer le suffixe ".info". La chaîne "-f" signale qu'un nom fichier suit, sa présence est obligatoire.

Plusieurs version d'hereh++ sont disponibles, "HZpp" est une version avec vérification interne d'erreurs, par exemple de dépassement de tableau. L'exécution est de ce fait ralentie, par contre les erreurs d'exécution sont en principe plus précises. La version "HZppfast" est la plus rapide, elle inclus peut de vérification au cours de l'exécution.

La lecture des informations s'effectue de manière séquentielle c'est-à-dire les unes après les autres. Il est nécessaire de suivre une chronologie. Par exemple il n'est pas possible de définir des conditions limites avant d'avoir défini l'entité (maillage, élément, noeud ...) sur laquelle vont s'appliquer ces conditions limites. Les informations sont donc ordonnées. Cette procédure permet de vérifier la cohérence des informations lues. Schématiquement, on trouvera :

- la dimension du problème : 1 2 ou 3,
- la définition du type de calcul,
- la définition du maillage, c'est-à-dire les noeuds et les éléments, ainsi que la définition de groupe ou liste de noeuds et éléments, ces grandeurs étant utilisées par la suite.
- les lois de comportement,
- les conditions de chargement,
- les conditions de bloquages cinématiques,
- et enfin les indications pour la sortie des résultats.

NB : En fait la liste précédente n'est pas exhaustive, on se reportera au chapitre (2) pour consulter la liste complète. Il est préférable de consulter cette liste avant toute préparation de fichier.

Pour permettre au programme de repérer ces différentes étapes, les informations sont séparées par des mots clés. Chaque mot clé est suivi des informations particulières au mot clé. Il est nécessaire d'utiliser l'orthographe exacte du mot clé, sinon la lecture s'arrête ou n'est pas effectuée correctement. Dans certain cas, les mots clés principaux sont suivis de sous-mots clés qui permettent un découpage plus fin de l'information.

La liste exhaustives des mots clés est donnée par le tableau (1). Dans le cas de l'existence de sous mots clés, la liste en sera donnée lors de l'examen du mot clé principal.

1.1 Choix de la langue

TABLE 1 – liste de mots clés

TYPE_DE_CALCUL	PARA_TYPE_DE_CALCUL	domaine_esclave	_fin_point_info_
mvt_maitre	les_courbes_1D	inerties	orthotropie
reperes_locaux	masse_addi initialisation	controle_contact	
para_affichage	dilatation_thermique	para_contact	zone_contact
para_energie	para_calculs_geometriques	condition_limite_lineaire_	
choix_materiaux	materiaux	epaisseurs	largeurs
sections	charges	blocages	typecharge
controle	noeuds	elements	nom_maillage
resultats	nom_maillage	flotExterne	masse_volumique
para_syteme_lineaire	para_pilotage_equi_global	para_dedies_dynamique	dimension

Le français est la langue utilisée par défaut. Il est possible d'utiliser l'anglais. Pour cela, après un numéro de version éventuelle, on indique le choix de la langue anglaise pour toutes les entrées sorties. L'emploi d'un numéro de version n'est pas actuellement décrit dans ce manuel, donc le choix de la langue doit-être la première instruction du fichier de donné selon la syntaxe suivante : mot clé "lang" suivi de "ENGLISH" ou "FRANCAIS". Pour utiliser l'anglais on indique donc :

```
lang ENGLISH
```

Sinon, par défaut, la langue est française. Mais on peut néanmoins l'indiquer explicitement par la syntaxe suivante :

```
lang FRANCAIS
```

1.2 Commentaires

Il est souvent intéressant de commenter les différentes informations utiles pour le programme, ceci dans le fichier même de lecture. Pour cela on utilise un caractère spéciale, le caractère #, pour indiquer que l'on a affaire à un commentaire. Ainsi dès que le programme rencontre le caractère#, il ignore la fin de la ligne qui suit ce caractère. Lorsque le caractère # est en début de ligne, toute la ligne est considérée comme une ligne de commentaire.

1.3 Ordre sur plusieurs lignes

Lorsque les informations a fournir sont importantes, il peut être préférable, pour des raisons de lisibilité, de pouvoir séparer une ligne d'entrée de données sur plusieurs lignes de fichiers. Par exemple supposons la ligne d'entrée suivante :

```
nQs= 0.1 gammaQs= 0.9 nQe= 0.2 gammaQe= 0.5 nMu2= 1 gammaMu2= 0.7 nMu2= 1 gammaMu2= 0.7 nMu3= 1 gammaMu3= 0.6
```

Vue la longueur de la ligne, il est préférable de la séparer en deux, pour cela on peut indiquer dans le fichier d'entrée :

```
nQs= 0.1 gammaQs= 0.9  nQe= 0.2 gammaQe= 0.5  \  
nMu2= 1 gammaMu2= 0.7   nMu2= 1 gammaMu2= 0.7  nMu3= 1 gammaMu3= 0.6
```

On remarque le caractère

qui est un caractère de continuation (style unix), qui indique qu'il faut également inclure la ligne qui suit. On peut ainsi utiliser un nombre quelconque de ligne, la seule limitation est que la ligne finale (qui globalise toutes les lignes qui se suivent) doit contenir un nombre de caractère inférieur à 1500 (par défaut). Par exemple les lignes suivantes sont également licites :

```
nQs= 0.1 gammaQs= 0.9  \  
nQe= 0.2 gammaQe= 0.5  \  
nMu2= 1 gammaMu2= 0.7  \  
nMu2= 1 gammaMu2= 0.7  \  
nMu3= 1 gammaMu3= 0.6
```

1.4 Inclusion de fichier

Il est possible d'inclure le contenu d'un fichier déjà existant directement dans le fichier .info, simplement en indiquant son nom. Pour cela on utilise la syntaxe suivante :
< nom_du_fichier

Le signe < permet au programme de reconnaître le nom suivant comme nom de fichier. Dans ce cas la lecture se poursuit dans le nouveau fichier jusqu'à sa fin, et ensuite il revient au fichier original. Il est possible d'utiliser un fichier inclus dans un fichier déjà inclus, en fait le procédé est récursif.

Nous allons maintenant examiner successivement les principales étapes de la lecture.

2 Liste exhaustive des différentes informations que l'on peut trouver dans un fichier .info

L'ordre des informations doit être respecté. Dans le cas où certaines libertés sont possibles c'est indiqué, de même lorsque l'information est optionnelle. On trouve successivement :

- dimension du problème (cf. 3). Cette information est optionnelle, par défaut c'est 3.
- le niveau de commentaire (cf. 4). Cette information est optionnelle, par défaut c'est 0.
- le type de problème (cf. III). Cette information est obligatoire.
- la description des maillages (cf. 11). Cette information est obligatoire. On y décrit les maillages et les références.
- le nombre éventuelle de maillages esclaves (cf.32.1). Cette information est optionnelle, par défaut c'est 0.
- la description des courbes1D (cf. 25). Cette information est optionnelle. Elle permet ensuite par exemple, de définir les chargements aux travers de courbes de charges.

- la description des lois de comportement (cf. 26). La définition d’une loi de comportement pour chaque élément est obligatoire.
- divers stockages (cf. 31). Ces informations sont obligatoires, mais varient en fonction des informations précédentes. Il s’agit :
 - de la masse volumique a définir pour tous les éléments,
 - par exemple de l’épaisseur des éléments coques ou plaque. S’il n’y pas d’éléments plaques ou coques, l’information est inutile sinon elle est obligatoire.
 - par exemple la section des éléments poutres s’il y a des poutres,
 - ...
- Le nombre éventuel de maillages en auto-contact (cf.32.2). Cette information est optionnelle, par défaut c’est 0.
- Les zones éventuelles, présumées en contact (cf.32.3). Cette information est optionnelle, par défaut aucune zone particulière n’est précisée, l’ensemble des noeuds esclaves et des maillages maîtres est considéré pour l’étude du contact.
- les différents efforts (cf. 33). Cette information est optionnelle. Par contre même s’il n’y pas de chargement en effort il faut indiquer le mot clé de chargement.
- les conditions limites cinématiques (cf. 34). Comme pour le chargement en effort, cette information est optionnelle, cependant même s’il n’y a aucune condition cinématique il est cependant nécessaire d’indiquer le mot clé de départ.
- les conditions initiales (cf. 36). Comme pour le chargement en effort, cette information est optionnelle, cependant même s’il n’y a aucune condition initiale il est cependant nécessaire d’indiquer le mot clé de départ.
- divers stockages (cf. 31). Il s’agit ici de la seconde lecture des stockage divers. Cela concerne la lecture éventuelle des masses additionnelles. Cette information est optionnelle.
- l’algorithme de chargement (cf. 37). Information optionnelle. Elle décrit comment le chargement global s’effectue. Cela ne concerne pas les conditions limites qui utilise directement une fonction de charge. Cela concerne donc que les conditions cinématiques ou en efforts fixes.
- les paramètres généraux du contrôle de la résolution (cf. 38). Cette information comprend en fait un ensemble de paramètre qui sont pour la plupart optionnelle. Par contre le mot clé de départ est obligatoire.
- la sortie des resultats (cf. 47). Information obligatoire. Par contre comprend en interne un ensemble de paramètre qui sont eux optionnels, et qui permettent de spécifier le type de sortie de l’information que l’on désire.
- le mot clé ”_fin_point_info_” suivi de plusieurs lignes vides.

3 Dimension du problème

La première information que le programme doit connaître est la dimension du problème à traiter. Celle-ci peut-être 1 2 ou 3. La syntaxe est la suivante :

```
dimension n
```

où n la dimension demandé. Un exemple de début de fichier .info est le suivant :

```

#
#   etude de la traction simple d'une biellette
#
#-----
# definition de de la dimension      |
# elle peut etre 1 ou 2 ou 3        |
#-----

dimension 1

```

On remarque les commentaires puis la définition de la dimension.

La dimension influe sur le type d'élément possible. Remarquons que la dimension intrinsèque des éléments c'est-à-dire la dimension de l'élément de référence, n'est pas obligatoirement identique à celle de l'espace dans lequel on effectue le calcul. Cependant, la dimension intrinsèque doit toujours être inférieure ou égale à celle de l'espace de travail. Par exemple l'élément biellette de dimension intrinsèque 1 convient pour toute les dimensions d'espace. A l'opposé les éléments volumiques : hexaèdre, tétraèdre, pentaèdre ..., ne conviennent que pour la dimension 3. Au moment de la lecture des éléments, s'il y a une incohérence de dimension, la lecture est stoppée.

Le mot clé dimension est optionnel. Par défaut la dimension est 3.

4 Niveau de commentaire

Le niveau de commentaire est un paramètre qui permet de régler le degré d'information que le programme affiche sur l'écran pendant l'exécution. Le niveau doit être un nombre entier positif en général compris entre 0 et 10. Un exemple de définition du niveau est :

```

#-----
# definition facultative du niveau d'impression
#-----

niveau_commentaire 0

```

On remarque les commentaires puis la définition du niveau après le mot clé : niveau_commentaire.

Le niveau est optionnel, sa valeur par défaut est 0.

Dans le cas où l'on utilise le niveau maximum, on a accès à des informations spécifiques qui dépendent du contexte. Par exemple il est possible de visualiser le contenu de la matrice masse, ou de la matrice de raideur. Ce niveau de commentaire n'est recommandé que pour la recherche d'erreur éventuelle, mais est inadaptée à un calcul réel.

Troisième partie

Type de problème traité

5 Introduction des Algorithmes et listes exhaustives

On entend par type de problème le fait d'étudier de la mécanique en statique, ou dynamique avec ou sans contact, en explicite ou implicite, ou encore de faire un calcul d'erreur etc.

La syntaxe est la suivante, le mot clé "TYPE_DE_CALCUL" puis sur la ligne suivante par exemple le sous mot clé : "non_dynamique" qui indique que le calcul concerne un problème d'équilibre non dynamique c'est-à-dire statique. Un exemple de définition dans un fichier .info est :

```
TYPE_DE_CALCUL
#-----
# problème d'équilibre non dynamique
#-----
non_dynamique
```

La liste exhaustive des problèmes que l'on peut traiter est donnée par le tableau (2).

5.1 Liste exhaustive des algorithmes : Galerkin continu et utilitaires

Chaque nom de type de calcul peut être suivi d'un sous-mot clé indiquant des calculs supplémentaires effectués. Par exemple :

```
TYPE_DE_CALCUL
#-----
# TYPE DE      |
# CALCULS     |
#-----
dynamique_explicite avec plus visualisation
```

indique que le calcul explicite est suivi d'une séquence interactive permettant de définir des fichiers pour la visualisation avec des outils spécifiques : Maple ou Gnuplot pour les courbes d'évolution et des plugins VRML pour la visualisation de déformées.

Enfin notons que la définition du type de calcul est obligatoire. Plusieurs autres types existent mais ils sont pour l'instant du domaine de la recherche, c'est-à-dire peu validés, c'est pourquoi leurs emplois n'est pas pour l'instant précisés dans ce document.

Les différentes méthodes proposées possèdent des paramètres de réglage. Dans le cas où aucune valeur particulière pour ces paramètres n'est proposée, le calcul utilise des paramètres par défaut. Dans le cas contraire l'utilisation du mot clé : "PARAM_TYPE_DE_CALCUL" permet de donner une valeur particulière aux paramètres. Ce mot clé doit être défini juste après la définition du type de calcul.

Par exemple le texte suivant :

TABLE 2 – liste des type de problèmes

mot clé	commentaire
non_dynamique	non dynamique sans contact (c'est-à-dire statique), paramètres éventuels d'accélération de convergence (5.2)
non_dyna_contact	non dynamique avec contact (c'est-à-dire statique en intégrant des conditions éventuelles de contact) pas de paramètre associé
dynamique_explicite	dynamique explicite avec ou sans contact, la méthode utilisée est les différences finis centrées cf.(5.4) un paramètre facultatif associé cf.(5.4)
dynamique_explicite_tchamwa	dynamique explicite sans contact, il s'agit de la méthode de Tchamwa-Wielgoz cf.(5.3) 1 paramètres associé cf. (5.3)
dynamique_implicit	dynamique implicite sans contact, correspond à la méthode de Newmark cf.(5.5) ou à la méthode HHT (Hilbert-Hughes-Taylor) deux ou 1 paramètres associés (cf. 5.5) ou cf. (5.5)
dynamique_explicite_chung_lee	dynamique explicite sans contact correspond à la méthode proposée par Chung-Lee cf.(5.6) 1 paramètre associé cf. (5.6)
dynamique_explicite_zhai	dynamique explicite sans contact correspond à la méthode proposée par Zhai. cf.(5.7) 2 paramètres associés cf. (5.7)
dynamique_Runge_Kutta	dynamique : avancement temporel résolu par la méthode de Runge-Kutta. cf.(5.8) plusieurs paramètres associés cf. (5.8)
dynamique_relaxation_dynam	relaxation dynamique recherche de la solution statique via la relaxation dynamique et la masse optimisée cf. (6.3)
umat_abaqus	utilisation d'herezh++ comme umat, permet l'utilisation de toutes les lois d'herezh++ par abaqus cf. (5.9)
utilitaires	différents utilitaires en pré ou post calcul par exemple de modification de maillage (cf. 7)
informations	recuperation d'informations par exemple récupération et création interactives de references (cf. 8)

PARA_TYPE_DE_CALCUL

```
#-----
# type de parametre   |          |
# associe au calcul   | valeur   |
```

```

#      phie          |          |
#-----
#      phi=          1.05

```

indique que le paramètre phi à la valeur 1.05. Notons que ce paramètre n'est utilisable qu'avec la méthode de Thamwa-Wielgoz, aussi dans le cas de l'utilisation d'une autre méthode une erreur à l'exécution sera générée.

Remarque

- Le pas de temps proposé par l'utilisateur, est comparé au pas de temps donné par la condition de stabilité du type Courant. Cette vérification n'est effectuée que pour les méthodes de Tchamwa-Wielgoz et Zhai. Dans le cas où le pas critique est plus faible que celui proposé, c'est le pas critique qui est retenue pour le calcul! Cependant cette stabilité peut néanmoins être mise en défaut dans des cas complexe de comportement matériel ou de contact par exemple. Il faut donc la prendre en compte avec précautions, en particulier vérifier qu'il n'y a pas de divergence flagrante, sinon il faut diminuer le pas.
- Dans le cas des méthodes explicites d'avancement temporel, le fait d'utiliser le pas critique, n'implique pas que le résultat obtenu soit précis!! Ce pas critique fournis uniquement une bonne maxi.

5.1.1 Paramètres génériques pour tous les algorithmes

A la suite des paramètres particuliers de l'algorithme, il est possible d'indiquer des paramètres génériques qui sont utilisés (ou non) par l'algorithme particulier. Actuellement ces paramètres sont les suivants :

TABLE 3 – liste des type de paramètres génériques pouvant être associés aux paramètres particuliers de l'algorithme

objectif des paramètres	référence	types de calcul associé
amortissement cinétique	(6.3.7)	algorithmes de dynamique et de relaxation dynamique
amortissement visqueux	(6.1)	algorithmes de dynamique et de relaxation dynamique
critère d'arrêt sur le résidu statique hors viscosité numérique	(6.4)	algorithmes de dynamique et de relaxation dynamique
contrôle mode debug	(6.3.8)	tous les algorithmes de calcul

La table 4 donne un exemple d'utilisation de paramètres génériques.

TABLE 4 – Exemple d'utilisation du mode "debug", de l'amortissement critique, et d'un arrêt sur l'équilibre statique, ceci avec un algorithme de relaxation dynamique

```

### il s'agit d'une relaxation dynamique avec amortissement visqueux
### le calcul de la masse utilise la matrice de raideur
### on a un re-calcul de la masse tous les 100 itérations et on utilise le casMass_relax= 5
### calcul de la viscosité -> pajand
### le critère d'arrêt est mixte résidu - déplacement

dynamique_relaxation_dynam #avec plus visualisation

PARA_TYPE_DE_CALCUL
# .....
# / type d'algorithme /
#.....
typeCalRelaxation= 2  lambda= 0.9  type_calcul_mass= 2  option_recalcul_mass= 4
parametre_calcul_de_la_masse_  casMass_relax= 5
parametre_recalcul_de_la_masse_  fac_epsilon= 100.
parametre_calcul_de_la_viscosite_  type_calcul_visqu_critique= 2  opt_cal_C_critique= 1  f_= 0.9

mode_debug_= 3

ARRET_A_EQUILIBRE_STATIQUE_ 2

```

5.2 Accélération de convergence

Cette partie est exploratoire (partie recherche en développement). La méthode provient d’une proposition de Jean-Marc Cadou, et est réservée à un calcul non-dynamique. Elle consiste à partir d’une suite de vecteurs solutions correspondants à n itérés, d’extrapoler ces vecteurs par la méthode “MPEM” (Minimal Polynomial Extrapolation Method) de manière à obtenir une solution optimisée. Pour utiliser cette méthode, on définit des paramètres associés à l’algorithme. La table (5) donne un exemple d’utilisation.

TABLE 5 – Exemple de paramètres pour mettre en oeuvre et conduire la méthode d’accélération de convergence

```
#-----
#| parametres (facultatifs ) associes au calcul implicite statique |
#-----

  PARA_TYPE_DE_CALCUL
# .....
# / acceleration de convergence par extrapolation methode MMPE /
#.....
acceleration_convergence_= 1 cas_acceleration_convergence_= 1 nb_vec_cycle_= 8
```

Le paramètre “acceleration_convergence=” indique si oui (=1) ou non (=0) on met en oeuvre l’accélération de convergence. Le paramètre “cas_acceleration_convergence=” permet de différencier le cas de projection, =1 : on projette sur les vecteurs solutions $\Delta X_n^{t+\Delta t} = X_n^{t+\Delta t} - X^t$, =2 : on projette sur les vecteurs résidus de l’itéré “n”, =3 : on projette sur les accroissements du vecteur solution $S_n = \Delta X_{n+1}^{t+\Delta t} - \Delta X_n^{t+\Delta t}$. Le paramètre “nb_vec_cycle=” indique le nombre de vecteur solutions maximum que l’on considère dans l’extrapolation. Au dessus de cette valeur, on recommence un nouveau cycle d’extrapolation.

5.3 Méthode de Tchamwa-Wielgoz

Il s’agit d’une méthode explicite, pour la résolution de l’équation d’équilibre instantané spatial et temporel, correspondant au principe des puissances virtuelles. L’avancement temporel est résolu par une technique analogue à la méthode classique des différences finies centrées. Par rapport à cette dernière méthode, la méthode de Tchamwa-Wielgoz introduit un terme d’amortissement des hautes fréquences. Cet amortissement est particulièrement efficace lorsque le pas de temps est proche de pas de temps critique. En revanche, l’efficacité diminue à mesure que l’on s’éloigne du pas critique.

L’amortissement est piloté par un paramètre ϕ qui peut-être compris entre 1 et l’infini. Plus ϕ est élevé, plus l’amortissement numérique des hautes fréquences est élevé. Une valeur de 1.1 conduit à un amortissement très élevé. Une valeur de 1.03 est un maxima, si l’on veut conserver une précision correcte.

Il est également possible d'utiliser un algorithme dérivé (pour la recherche) qui module la valeur de φ en fonction du niveau de l'accélération à chaque ddl. Pour cela on définit une fonction "f" et le coefficient résultant devient pour chaque ddl : $(1 + f(|\gamma(i)|)) \cdot (\varphi - 1)$ (au lieu de φ initialement), "i" étant le numéro du ddl. La table (6) donne un exemple de ce type de déclaration. On remarque le mot clé optionnel : "typeCalEqui=" suivi d'un nombre entier : 1 dans le cas d'un calcul normal, 2 pour le calcul avec modulation. Dans ce dernier cas il est nécessaire de faire figurer ensuite le nom d'une courbe après le mot clé "CGamma_pourVarPhi=", courbe qui devra ensuite être défini après le maillage.

TABLE 6 – Exemple de de l'algorithme de Tchamwa avec une fonction de modération sur φ .

```

TYPE_DE_CALCUL
#-----
# TYPE DE      |coefficients      |
# CALCULS     | phie de la loi  |
#-----
dynamique_explicite_tchamwa      avec      plus      visualisation

PARA_TYPE_DE_CALCUL
#-----
# type de parametre |          |
# associe au calcul | valeur  |
#      phie        |          |
#-----
      phi=                1.03  typeCalEqui= 2  CGamma_pourVarPhi= cgamma

# def du maillage
< hz090_200.her

les_courbes_1D -----
cgamma COURBEPOLYLINEAIRE_1_D
      Debut_des_coordonnees_des_points
Coordonnee dim= 2      0          0.
Coordonnee dim= 2      1.e3       0.
Coordonnee dim= 2      1.e7       1.0
Coordonnee dim= 2      1.e10      1.0
      Fin_des_coordonnees_des_points

```

Un second raffinement est possible en se referent à une moyenne des accelerations sur n pas.

$$moyenne(i) = \frac{\sum_{r=1}^n (\gamma(i)_{t_r} + \gamma(i)_{t_r+\Delta t}) \cdot \Delta t}{2 \cdot n} \quad (1)$$

Le coefficient phi est ensuite pondere suivant la valeur de cette moyenne, entre 1 et la valeur demandée selon la méthode suivante en fonction de deux paramètres *valmax* et *valmin* fournis par l'utilisateur :

- si $|moyenne(i)| > valmax$ alors cela signifie qu'il y a réellement une augmentation de l'accélération, il ne faut pas filtrer, on retient $\varphi = 1$;
- si $|moy(i)| < valmin$ alors il s'agit d'oscillations numérique, il faut filtrer, on retient φ donné par l'utilisateur,
- si $|moy(i)|$ est entre *valmin* et *valmax* on utilise une progression.

En résumé, appelant $\varphi(i)_{modif}$, la valeur de ce φ modifié, le coefficient φ_{util} retenue a pour valeur au noeud "i" :

$$\varphi_{util} = (1. + (\varphi(i)_{modif} - 1) \cdot f(|\gamma(i)|)) \quad (2)$$

avec f la fonction de modération. **Remarque :** $\varphi(i)_{modif}$ est mis a jour uniquement tous les n pas de temps, et par défaut n= 0 et $\varphi(i)_{modif} = \varphi$

La table (7) donne un exemple d'utilisation.

5.4 Méthode classique des différences finies centrées (DFC)

Il s'agit d'une méthode explicite, pour la résolution de l'équation d'équilibre instantané en spatial et temporel, correspondant au principe des puissances virtuelles. L'avancement temporel est résolu par la méthode classique des différences finies centrées. Il n'y a pas d'amortissement des hautes fréquences numériques introduites par la méthode. Par contre il est possible d'en introduire, via un amortissement de Rayleigh [3], ou via la méthode du "bulk viscosity" [7] par exemple.

La méthode peut être présentée de deux manières équivalentes qui ont donné lieu à trois implantations différentes dans Herezh++.

La première méthode consiste à appliquer deux fois la méthode des différences finies centrées pour obtenir l'accélération. On a :

$$\dot{\mathbf{X}}_{n-1/2} = \frac{\mathbf{X}_n - \mathbf{X}_{n-1}}{\Delta t} \quad (3)$$

$$\dot{\mathbf{X}}_{n+1/2} = \frac{\mathbf{X}_{n+1} - \mathbf{X}_n}{\Delta t} \quad (4)$$

d'où

$$\begin{aligned} \ddot{\mathbf{X}}_n &= \frac{(\dot{\mathbf{X}}_{n+1/2} - \dot{\mathbf{X}}_{n-1/2})}{\Delta t} \\ &= \frac{\mathbf{X}_{n+1} - 2\mathbf{X}_n + \mathbf{X}_{n-1}}{\Delta t^2} \end{aligned} \quad (5)$$

TABLE 7 – Exemple de de l’algorithme de Tchamwa avec une fonction de modération sur φ et la prise en compte d’une moyenne de l’accélération

```

TYPE_DE_CALCUL
#-----
# TYPE DE CALCULS | sous type |
#-----
# ----- schema de tchamwa -----
dynamique_explicite_tchamwa

PARA_TYPE_DE_CALCUL
#-----
# type de parametre |          |
# associe au calcul | valeur   |
# phie              |          |
#-----
phi= 1.03  typeCalEqui= 2  CGamma_pourVarPhi= fonc n_= 4  valmin_= 1.e0 valmax_= 2.e1

# ----- fin du schema de tchamwa -----

# definition du maillage et des references: via stamm01
< barre100.her

les_courbes_1D -----
#-----
# definition d’une fonction constante
#-----
fonc COURBEPOLYLINEAIRE_1_D
      Debut_des_coordonnees_des_points
Coordonnee dim= 2      0      0.
Coordonnee dim= 2      1.e3    0.
Coordonnee dim= 2      1.e7    1.0
Coordonnee dim= 2      1.e10   1.0
      Fin_des_coordonnees_des_points

```

D’une manière pratique, le calcul s’effectue dans Herezh++ en un pas. Dans la version initiale (voir 5.4 avec `type_cal_equilibre = 1` ou `2`), on définit le vecteur des inconnues généralisées au temps “n” suivant :

$$\left\{ \begin{array}{c} \mathbf{X}_n \\ \dot{\mathbf{X}}_n \\ \dot{\mathbf{X}}_{n-1/2} \\ \ddot{\mathbf{X}}_n \end{array} \right\} \quad (6)$$

Pour calculer le vecteur au temps suivant “n+1” on adopte la méthodologie suivante :

- calcul de $\dot{\mathbf{X}}_{n+1/2}$ avec (5), première expression,
- calcul de \mathbf{X}_{n+1} avec (4),
- calcul d’une version explicite de la vitesse au temps “n+1” :

$$\dot{\mathbf{X}}_{n+1} = \dot{\mathbf{X}}_{n+1/2} + \frac{\Delta t}{2} \ddot{\mathbf{X}}_n \quad (7)$$

- calcul de $\ddot{\mathbf{X}}_{n+1}$ à l’aide de l’équation d’équilibre

$$\ddot{\mathbf{X}}_{n+1} = [M]^{-1} \left(R_{(int+ext)}(\mathbf{X}_{n+1}, \dot{\mathbf{X}}_{n+1}) \right) \quad (8)$$

- calcul de la vitesse réelle (au sens des DFC) au temps “n+1”

$$\dot{\mathbf{X}}_{n+1} = \dot{\mathbf{X}}_{n+1/2} + \frac{\Delta t}{2} \ddot{\mathbf{X}}_{n+1} \quad (9)$$

Les relations (3) (4) et (5) sont utilisées pour les conditions limites. Seules les conditions en vitesse ou accélération permettent d’avoir une avancée strictement explicite. La condition de déplacement imposé ne le permet pas. En effet, la connaissance de \mathbf{X}_{n+1} ne permet pas de connaître les autres éléments du vecteur généralisé.

Dans le cas où `type_cal_equilibre = 1`, le calcul utilise une procédure d’initialisation, pour laquelle lors du premier pas de temps on considère que l’accélération est déduite des conditions limites où est initialisée à 0. Dans le cas où `type_cal_equilibre = 2`, un premier pas de temps nul est effectué pour utiliser l’équation d’équilibre à $t=0$ d’où la valeur de l’accélération à $t=0$.

Il est également possible d’éviter d’utiliser les vitesses intermédiaires $\dot{\mathbf{X}}_{n-1/2}$ et $\dot{\mathbf{X}}_{n+1/2}$ en utilisant la séquence suivante :

- 1) $\mathbf{X}_n = \mathbf{X}_{n-1} + \Delta t \dot{\mathbf{X}}_{n-1} + \frac{(\Delta t)^2}{2} \ddot{\mathbf{X}}_{n-1}$
- 2) approximation explicite des vitesses : $\dot{\mathbf{X}}_n = \dot{\mathbf{X}}_{n-1} + \Delta t \ddot{\mathbf{X}}_{n-1}$
- 3) $\ddot{\mathbf{X}}_n = [M]^{-1} \left(R_{(int+ext)}(\mathbf{X}_n, \dot{\mathbf{X}}_n) \right)$
- 4) correction des vitesses : $\dot{\mathbf{X}}_n = \dot{\mathbf{X}}_{n-1} + \frac{\Delta t}{2} (\ddot{\mathbf{X}}_{n-1} + \ddot{\mathbf{X}}_n)$ (10)

Cette séquence constitue maintenant l’algorithme par défaut, correspondant en fait à `type_cal_equilibre = 3`. Le calcul de la vitesse exacte après résolution, permet d’obtenir une énergie cinétique plus précise. Comme pour `type_cal_equilibre = 2`, on utilise l’équation d’équilibre à $t=0$ d’où la valeur de l’accélération, pour initier le calcul.

5.5 Famille de Newmark et méthode HHT (Hilbert-Hughes-Taylor)

Il s’agit de la famille classique des méthodes de Newmark, méthodes de préférence implicites, pour la résolution de l’équation d’équilibre instantané spatial et temporel,

correspondant au principe des puissances virtuelles. L'avancement temporel est résolu par une méthode de quadrature, qui introduit deux coefficients de pilotage. Suivant la valeur de ces coefficients, la précision peut-être soit du second ordre soit dans un cas particulier du troisième ordre, la méthode est alors explicite.

D'une manière plus précise, lorsqu'il y a deux paramètres, ceux-ci correspondent aux coefficients β et γ de la méthode classique de Newmark. La méthode est ici "a priori" inconditionnellement stable. Le choix de $\beta = 1/12$ conduit à une précision d'ordre 3 sur la fréquence, ce qui est l'optimum en terme de précision comparé aux autres valeurs de β qui conduisent à une précision de 2, par contre la stabilité est alors conditionnelle.

Lorsqu'il y a un seul paramètre, celui-ci correspond à la méthode HHT. Ainsi c'est la présence du mot clé "hht=" ou "beta_et_gamma=" qui permet de distinguer entre la méthode de Newmark classique et la méthode HHT qui correspond en fait à la méthode de Newmark modifiée. Dans le cas de la méthode HHT, les paramètres β et γ sont imposés. En appelant θ le paramètre HHT on a : $\beta = (1 - \theta)(1 - \theta)/4$ et $\gamma = 1/2 - \theta$.

5.6 Méthode proposée par Chung-Lee

Il s'agit d'une méthode explicite, pour la résolution de l'équation d'équilibre instantané spatial et temporel, correspondant au principe des puissances virtuelles. L'avancement temporel est résolu par une technique analogue à la méthode classique des différences finies centrées. Par rapport à cette dernière méthode, la méthode de Chung-Lee introduit un terme d'amortissement des hautes fréquences. Cet amortissement est particulièrement efficace lorsque le pas de temps est proche de pas de temps critique. En revanche, l'efficacité diminue à mesure que l'on s'éloigne du pas critique, d'une manière analogue à la méthode de Tchamwa. Par contre, ici la plage de réglage du paramètre d'atténuation des hautes fréquences, est faible.

De manière plus précise, Le paramètre de réglage correspond au paramètre β de la méthode de Chung-Lee.

5.7 Méthode proposée par Zhai

Il s'agit d'une méthode explicite, pour la résolution de l'équation d'équilibre instantané spatial et temporel, correspondant au principe des puissances virtuelles. L'avancement temporel est résolu par une technique analogue à la méthode classique des différences finies centrées. Par rapport à cette dernière méthode, la méthode de Zhai introduit un terme d'amortissement des hautes fréquences.

Les paramètres de contrôle correspondent aux paramètres φ et ϕ de la méthode de Zhai.

5.8 Méthode De Runge-Kutta

Il s'agit de la famille classique des méthodes de Runge-Kutta, méthodes de préférence explicites, pour la résolution de l'équation d'équilibre instantané spatial et temporel, correspondant au principe des puissances virtuelles.

L'équation d'avancement temporel est résolu par une méthode de la famille de Runge-Kutta. Plusieurs paramètres sont disponibles pour piloter l'algorithme. La précision sur

le pas de temps est garantie en fonction d'une erreur estimée calculée à l'aide de deux méthodes de Runge-Kutta imbriquées de niveau de troncature (en Δt) différents. Ici, 3 méthodes explicites sont actuellement disponibles : ordre 2 et 3 (c'est-à-dire de niveau de troncature au deuxième ordre et au troisième ordre), ordre 3 et 4, ordre 4 et 5. L'ordre est indiqué à l'aide du paramètre "algo_kutta_" suivi du chiffre 3 pour les ordres 2-3, ou 4 pour les ordres 3-4, ou 5 pour les ordres 4-5. La précision demandée est spécifiée à l'aide de deux paramètres : algoErrAbs_ qui permet d'indiquer la précision absolue voulue, et algoErrRel_ qui permet d'indiquer la précision relative désirée. Le calcul est accepté si la précision estimée est inférieure à "algoErrAbs_+algoErrRel_ * max(max(|X_i|), Δt * max(|Ẋ_i|))". Dans cette formule, la vitesse est multipliée par le pas de temps pour être d'un même ordre de grandeur que la position.

Dans le cas où le critère de précision n'est pas respecté, le pas de temps est subdivisé selon un algorithme particulier et le calcul est reconduit. De manière à éviter une suite infinie de subdivision, il est possible d'indiquer un nombre maxi d'appel à la fonction dérivée (ici l'équation d'équilibre dynamique globale), à l'aide du paramètre nbMaxiCall_ suivi du nombre maxi d'appels. La table (8) donne un exemple de déclaration de paramètres.

Il est à remarquer que l'algorithme proposé fait appel un stockage informatique volumineux, environ 10 fois celui nécessaire avec DFC, de plus les calculs sont beaucoup plus long (par exemple un facteur 10) qu'avec un calcul classique. En fait l'objectif est d'être capable d'obtenir des solutions de références pour une précision fiable donnée, ce qui n'est pas possible aisément avec les méthodes classiques.

TABLE 8 – Exemple de déclaration des paramètres de pilotage du Runge-Kutta

```
#PARA_TYPE_DE_CALCUL
#-----
# definition des parametres de calcul |
#-----
algo_kutta_ 5  algoErrAbs_  1.e-4  algoErrRel_ 1.e-4  nbMaxiCall_ 1000
```

5.9 Utilisation d'Herezh++ comme Umat

L'objectif est de permettre un couplage entre le logiciel commercial Abaqus et Herezh++ au travers d'une Umat. On se reportera à l'article [Rio et al., 2008] pour plus d'information sur les fondements scientifiques.

Au niveau d'Abaqus il est nécessaire de définir la fonction utilisateur Umat selon le format donné par la table 9. Cette fonction appelle une routine C définie par les tables 10 et 11. Les routines de lecture et écriture sont données au chapitre (28.11).

Concernant Herezh, l'appel d'un comportement particulier s'effectue à l'aide d'un fichier .info classique qui contient l'algorithme "umat_abaqus", sans autre paramètre particulier. La table 12 donne un exemple d'un fichier de commande permettant de définir l'utilisation d'une loi hyper-élastique de type Mooney-Rivlin. On remarque que le maillage contient aucun noeud, et un seul élément qui est d'un type particulier : "POINT CONSTANT"

qui n'est relié à aucun noeud. Il ne faut utiliser que ce type de maillage qui est spécifique et ne peut-être utilisé pour autre chose.

Ensuite il est nécessaire de définir deux "fichiers nommés" situés en mémoire centrale (named pipes) : "Umat_envoi_Hz" pour l'envoi et "Umat_reception_Hz" pour la réception. La création des pipes nommés, ouverts en lecture-écriture, s'effectue à l'aide de l'utilitaire "mkfifo" de la manière suivante (dans un terminal, à l'endroit où on veut les utiliser) :

```
mkfifo -m+wr Umat_reception_Hz
mkfifo -m+wr Umat_envoi_Hz
```

Une fois le fichier de commande réalisé, et les deux fichiers pipe créé, on lance Herezh avec en argument le fichier de commande .info, puis on lance Abaqus avec la subroutine Umat. Les deux programmes dialogues et se synchronise grâce aux opérations d'écriture-lecture sur les pipes.

Il est également possible de faire jouer le rôle d'Abaqus à un second processus Herezh dont le fichier de commande comprendra comme loi de comportement, un appel à une routine externe Umat. On se reportera au chapitre (28.11) pour plus d'information.

TABLE 9 – déclaration de la fonction Umat fortran : partie fortran

```

C -----
C   UMAT_Herezh
c   appel a subroutine c
C   pour execution de HZ++
C -----
C
C   SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
1  RPL,DDSDDT,DRPLDE,DRPLDT,STRAN,DSTRAN,
2  TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,MATERL,NDI,NSHR,NTENS,
3  NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,CELENT,
4  DFGRDO,DFGRD1,NOEL,NPT,KSLAY,KSPT,KSTEP,KINC)
C
C   INCLUDE 'ABA_PARAM.INC'
C
C   CHARACTER(LEN=80) MATERL
C   DIMENSION STRESS(NTENS),STATEV(NSTATV),
1  DDSDDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
2  STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),
3  PROPS(NPROPS),COORDS(3),DROT(3,3),
4  DFGRDO(3,3),DFGRD1(3,3)
C
C   DIMENSION EELAS(6),EPLAS(6),FLOW(6)
C   PARAMETER (ONE=1.0D0,TWO=2.0D0,THREE=3.0D0,SIX=6.0D0)
C   DATA NEWTON,TOLER/10,1.D-6/
C
cc  Appel fonction C pour lancement de Herezh
C   EXTERNAL appelc
C
C   IF (NDI.NE.3) THEN
C     WRITE(6,1)
1    FORMAT(//,30X,'***ERROR - THIS UMAT MAY ONLY BE USED FOR ',
1    'ELEMENTS WITH THREE DIRECT STRESS COMPONENTS')
C     STOP
C   ENDIF

```

TABLE 10 – déclaration de la fonction Umat fortran : partie en langage C

```
call appelc(  
1 %ref(STRESS),  
2 %ref(DDSDDE),  
3 %ref(SSE),  
4 %ref(SPD),  
5 %ref(SCD),  
6 %ref(RPL),  
7 %ref(DDSDDT),  
8 %ref(DRPLDE),  
9 %ref(DRPLDT),  
1 %ref(STRAN),  
2 %ref(DSTRAN),  
3 %ref(TIME),  
4 %ref(DTIME),  
5 %ref(TEMP),  
6 %ref(DTEMP),  
7 %ref(MATERL // char(0)),  
8 %ref(NDI),  
9 %ref(NSHR),  
1 %ref(NTENS),  
2 %ref(NSTATV),  
3 %ref(PROPS),  
4 %ref(NPROPS),  
5 %ref(COORDS),  
6 %ref(DROT),  
7 %ref(PNEWDT),  
8 %ref(CELENT),  
9 %ref(DFGRD0),  
1 %ref(DFGRD1),  
2 %ref(NOEL),  
3 %ref(NPT),  
4 %ref(KSLAY),  
5 %ref(KSPT),  
6 %ref(KSTEP),  
7 %ref(KINC))  
  
RETURN  
END
```

TABLE 11 – entête des routines C appelées par la subroutine fortran

```

/*include of the header's*/
#include <hsys/types.h>
#include <hsys/stat.h>
#include <hctype.h>
#include <hstdio.h>
#include <hsys/fcntl.h>
#include <hunistd.h>
extern "C"
/*-----*/
void appelc_( double* STRESS,double* DDSDDDE,double* SSE,double* SPD,double* SCD
              , double* RPL,double* DDSDDT,double* DRPLDE,double* DRPLDT,double* STRAN
              , double* DSTRAN,double* TIME,double* DTIME,double* TEMP,double* DTEMP
              , char* CMNAME,int* NDI,int* NSHR,int* NTENS,int* NSTATV,double* PROPS
              , int* NPROPS,double* COORDS,double* DROT,double* PNEWDT,double* CELENT
              , double* DFGRDO,double* DFGRD1,int* NOEL,int* NPT,int* LAYER,int* KSPT
              , int* KSTEP,int* KINC)
{ /*-----*/
  /* C function for sending data in the pipe */
  /*-----*/
  EcritureDonneesUma(STRESS,SSE,SPD,SCD,STRAN,DSTRAN,TIME,DTIME,TEMP ,DTEMP
                    ,CMNAME,NDI,NSHR,NTENS,NSTATV,COORDS,DROT,PNEWDT,CELENT ,DFGRDO
                    ,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC);
  /*-----*/
  /* C function for reading data in the pipe */
  /*-----*/
  LectureDonneesUmat (STRESS,DDSDDDE,SSE,SPD,SCD,NDI,NSHR,NTENS,PNEWDT ,RPL
                    ,DDSDDT,DRPLDE,DRPLDT);
}

```

TABLE 12 – exemple de l’algorithme permettant d’utiliser Herezh++ en tant qu’Umat

```

# -----
#  exemple Umat                               #
# -----
dimension 3
niveau_commentaire 4
TYPE_DE_CALCUL
umat_abaqus      # definition de l'algorithme
#-----
#  maillage
#-----
  noeuds -----
    0 NOEUDS
  elements -----
    1 ELEMENTS
    1 POINT CONSTANT
    E_to 1
#-----
      choix_materiaux  #-----
E_to      hyper
      materiaux        #-----
hyper    MOONEY_RIVLIN_3D
      C01= 80.  C10= 40.  K= 900. type_potvol_ 4
      masse_volumique #-----
E_to    1.00
charges          #-----
blocages        #-----
controle        #-----
SAUVEGARDE 1
resultats       #-----
COPIE           0
POINTS_INTEGRATION E_to
Green-Lagrange
      _fin_point_info_ #-----

```

5.10 Remarques concernant les conditions limites et initiales

Concernant les conditions limites, leur mise en place demande quelques précisions. Tout d'abord il faut séparer la notion de conditions initiales et celles de conditions limites. Les conditions initiales n'agissent qu'à l'instant initiale. Les conditions limites agissent tout au long du calcul, quand elles sont actives. On se reportera au chapitre des conditions limites pour plus de précisions et en particulier (36), cependant on indique ici quelques particularités.

Au niveau du programme, l'idée est que les positions, vitesses et accélérations appartiennent à un groupe de ddl liés, c'est-à-dire que si par exemple la position est imposée, alors le calcul de la vitesse ou de l'accélération en découle. Ainsi, le fait d'imposer une position et une vitesse est considéré comme surabondant. Il y a alors émission d'un message signalant la surcharge, mais le calcul peut s'effectuer. En fait dans la pratique, une seule des conditions est utilisée, la dernière ! Ce fonctionnement est cohérent avec la réalité.

Dans le cas des conditions initiales, il est possible d'indiquer une vitesse initiale (de même qu'une accélération initiale). Par contre la donnée d'un déplacement initial (conduisant à une position initiale différente de la position du maillage initiale) ne sera prise en compte qu'au temps $0 + \Delta t$ (cf 36.1). Or suivant le type d'algorithme, ce déplacement sera surchargé par celui obtenu par l'application de l'algorithme. Il ne faut donc pas utiliser en dynamique, de déplacement initiale différent de la position du maillage, mais directement intégrer cette position initiale dans la position du maillage ! (ce qui paraît logique)

6 Relaxation dynamique

Il est possible d'introduire de la relaxation dans un calcul dynamique, avec pour objectif d'obtenir une solution figée (c'est-à-dire celle correspondante au calcul statique). Il est à noter que cette technique n'est a priori valide que pour un comportement global indépendant du chemin (dans l'espace des déformations) effectué entre la situation initiale et la situation finale, ceci dans le cas où le chemin est conséquent. Ainsi dans ce dernier cas, il faut réserver cette technique aux comportements matériels réversibles.

Cependant, un trajet important peut-être subdivisé en petites parties, et l'algorithme appliqué successivement sur chaque partie. Dans l'hypothèse où les parties de trajet sont suffisamment petites, il est alors possible d'utiliser un comportement quelconque.

Enfin, s'il y a des instabilités géométriques, il faut bien noter que la solution obtenue est une des solutions possibles, pas forcément celle qui est la plus physique, en tout cas rien ne nous permet de l'affirmer.

Actuellement deux possibilités sont implantées dans Herezh++ concernant la relaxation dynamique :

1. soit on utilise un algorithme dynamique classique (DFC, Tchamwa, Newmark ...) auquel on ajoute une fonction supplémentaire (cf.6.1) : de l'amortissement cinétique ou de l'amortissement visqueux de préférence proche de l'amortissement critique (cf. 6.3.7) . Dans ce cas, le calcul suit l'algorithme initial sauf :

- dans le cas de l’amortissement cinétique, à l’apparition de pics de l’énergie cinétique, les vitesses sont remises à 0. L’idée est que la situation correspondant à un pic d’énergie cinétique, est proche d’un point d’équilibre. A la suite de chaque pic, la vitesse va de nouveau croître, due aux efforts qui sont en jeux. Cette technique à pour conséquence de réduire petit à petit les vitesses mises en jeux, elle agit de manière analogue à de la viscosité artificielle. Dans la pratique, la technique s’avère particulièrement efficace pour des structures présentant de grandes variations de position (par exemple des structures souples, soumises à une pression de gonflage)
 - dans le cas de l’amortissement visqueux critique, à chaque pas de temps, l’équilibre prend en compte un amortissement qui tend à terme à réduire au mieux les oscillations de manière à tendre vers un équilibre statique. Comme pour l’amortissement cinétique, cette méthode s’avère très performante pour les situations d’équilibres instables.
2. soit on utilise un algorithme particulier. Dans ce dernier cas, d’une part on agit sur les vitesses, mais d’autre part la masse est également modifiée et optimisée(cf.6.3). Enfin là également il y a le choix entre l’amortissement cinétique et l’amortissement visqueux critique approché.

6.1 Utilisation d’algorithmes classiques avec de l’amortissement cinétique

Après avoir défini l’algorithme classique, on introduit à la fin des paramètres de l’algorithme un certain nombre de mots clé. La table (13) donne un exemple de déclaration de relaxation cinétique. On trouve successivement :

- “avec_amortissement_cinétique_” : mot clé obligatoire pour déclencher la prise en compte de la relaxation,
- ”nb_deb_test_amort_cinétique_” : mot clé facultatif, le nombre minimal d’itération, à partir duquel on démarre l’algorithme d’amortissement cinétique, par défaut = 1 ;
- “max_nb_decroit_pourRelaxDyn_” : mot clé facultatif, indique à partir de combien de diminution de l’énergie cinétique on peut appliquer la relaxation, par défaut = 1,
- “coef_arret_pourRelaxDyn_” : mot clé facultatif, permet d’arrêter l’application de la relaxation lorsque l’énergie cinétique est inférieure à ce coef * le dernier pic d’énergie cinétique, par défaut = 0.5, (affichage : ”relaxation_gelee”)
- “coef_redemarrage_pourRelaxDyn_” : mot clé facultatif, permet de redémarrer l’application de la relaxation lorsque l’énergie cinétique devient supérieur à ce coef * le maximum des pic d’énergie enregistré, par défaut = 0.05, (affichage : ”relaxation_re_active”)
- “max_deltaX_pourRelaxDyn_” : mot clé facultatif qui en fait dépend du cas traité, doit donc de préférence être réajusté. Il indique la limite inférieure de $\|X^{t+\Delta t} - X^t\|_\infty$ à partir de laquelle on arrête le calcul, par défaut = 0.1.
- “nb_max_dX_OK_pourRelaxDyn_” : mot clé facultatif, correspond au nombre de fois que le critère précédent doit être satisfait pour que le calcul s’arrête, par défaut = 6 ;
- “nb_deb_testfin_pourRelaxDyn_” : mot clé facultatif, donne le nombre de fois que

l’algorithme de relaxation doit-être activé, avant que l’on commence a appliquer le test d’arrêt du calcul.

- “fi_parametre_amortissement_cinetique_” : mot clé obligatoire indiquant la fin des paramètres

NB : Pour les paramètre ”mode_debug_=” et ”ARRET_A_EQUILIBRE_STATIQUE_” voir ??

Chaque paramètre doit-être sur une ligne différente !

TABLE 13 – Exemple de déclaration d’un amortissement cinétique avec l’algorithme de Tchamwa

```
TYPE_DE_CALCUL
#-----
# definition du type primaire de calcul avec des sous types |
# specifiant les traitements annexes: ici la visualisation |
#-----
dynamique_explicite_tchamwa avec plus visualisation

PARA_TYPE_DE_CALCUL
#-----
# definition des parametres de calcul |
#-----
phi= 1.03
avec_amortissement_cinetique_
    max_nb_decroit_pourRelaxDyn_      1
    coef_arret_pourRelaxDyn_          0.
    coef_redemarrage_pourRelaxDyn_    0.01
    max_deltaX_pourRelaxDyn_          0.4
    nb_max_dX_OK_pourRelaxDyn_       5
    nb_deb_testfin_pourRelaxDyn_     100
fi_parametre_amortissement_cinetique_

mode_debug_= 500
ARRET_A_EQUILIBRE_STATIQUE_ 2
```

6.2 Utilisation d’algorithmes classiques avec de l’amortissement visqueux critique

Après avoir défini l’algorithme classique, on introduit à la fin des paramètres de l’algorithme les mots clés permettant de contrôler le calcul de l’approximation de la viscosité critique numérique. La table (14) donne un exemple de déclaration. On se reportera à 6.3.7.

NB : Pour les paramètre "mode_debug=" et "ARRET_A_EQUILIBRE_STATIQUE_" voir ??

Chaque paramètre doit-être sur une ligne différente !

TABLE 14 – Exemple de déclaration d’un amortissement visqueux avec l’algorithme de Tchamwa

```
dynamique_explicite_tchamwa #avec plus visualisation

PARA_TYPE_DE_CALCUL
# .....
# / type d'algorithme /
#.....
phi= 1.03
#parametre_calcul_de_la_viscosite_ opt_cal_C_critique= 1 f_= 2.9 #ampli_visco_= 1.
parametre_calcul_de_la_viscosite_ type_calcul_visqu_critique= 1 \
opt_cal_C_critique= 1 f_= 0.9

mode_debug_= 500
ARRET_A_EQUILIBRE_STATIQUE_ 2
```

6.3 Algorithme de Relaxation dynamique

L’algorithme se décline en plusieurs versions, en particulier deux types d’amortissement peuvent être invoqués : cinétique ou visqueux critique. Les différents choix sont les suivants :

- Le type de relaxation : cinétique ou visqueux critique,
 - le type de calcul de la pseudo-masse et les mises à jour éventuelles pendant le calcul,
- A noter que "tous" les paramètres sont facultatifs et ont une valeur par défaut. Pour éviter un fonctionnement aléatoire, il est préférable d’indiquer une valeur pour tous les paramètres utilisés. Dans le cas de doute, on peut consulter le début du fichier .BI qui contient les valeurs qui ont été réellement utilisées pour le calcul.

Sur la première ligne des paramètres on trouve :

- "typeCalRelaxation=" : indique le type de relaxation.

1. typeCalRelaxation= 1 : (cf. 6.3.1) Il s’agit d’une relaxation avec amortissement cinétique, qui utilise différentes techniques pour le calcul de la pseudo-masse. Ces techniques sont différenciées à l’aide de paramètres particuliers explicités par la suite. L’amortissement étant cinétique on peut donc modifier également les paramètres de l’amortissement cinétique.
2. typeCalRelaxation= 2 : (cf.6.3.2) Il s’agit d’une relaxation avec amortissement visqueux critique, qui utilise les mêmes techniques de calcul de la pseudo-masse

que le cas `typeCalRelaxation= 1` par contre l'amortissement est différent car visqueux critique.

– suivi des paramètres principaux détaillés en [6.3.1](#) et [6.3.2](#)

Une fois "typeCalRelaxation" et les paramètres généraux définis on trouve sur les lignes qui suivent (une ligne par paquet de paramètres) :

1. éventuellement les paramètres permettant de contrôler le calcul de la pseudo-masse (cf.[6.3.3](#)),
2. éventuellement les paramètres de contrôle du re-calcul de la matrice masse (cf. [6.3.6](#)),
3. éventuellement les paramètres permettant de contrôler le calcul de la matrice visqueuse (cf. [6.3.7](#)),
4. éventuellement le paramètre de contrôle du mode "debug" (cf. [6.3.8](#)),
5. éventuellement les paramètres de contrôle de l'amortissement cinétique (voir [6.1](#) pour le détail des paramètres de contrôle)
6. éventuellement un paramètre indiquant que l'on veut une convergence sur le résidu et /ou sur le déplacement (voir [6.4](#) pour plus d'informations).

Le paramètre "typeCalRelaxation=" est par défaut : 1

6.3.1 Cas d'une relaxation avec amortissement cinétique

Exemple de declaration :

```
typeCalRelaxation= 1 lambda= 0.7 type_calcul_masse= 1 option_recalcul_masse= 1
```

1. `typeCalRelaxation= 1`, indique qu'il s'agit d'une relaxation avec amortissement cinétique,
2. `lambda= 0.7` : le paramètre `lambda` permet de pondérer le calcul de la masse
3. `type_calcul_masse= 1` : permet de choisir le type de calcul de la matrice de pseudo-masse
4. `option_recalcul_masse= 1` : permet de choisir le test de re-calcul de la matrice masse
5. à la fin des paramètres de l'algorithme on peut indiquer des paramètres particuliers d'amortissement cinétique (ex : [15](#))

Tous les paramètres sont facultatifs : les valeurs par défaut sont :

- `typeCalRelaxation= 2`;
- `lambda= 0.605`;
- `type_calcul_masse= 2`;
- `option_recalcul_masse= 0`;

La table [15](#) donne un exemple de déclaration.

6.3.2 Cas d'une relaxation avec amortissement visqueux

Exemple de declaration :

```
typeCalRelaxation= 2 lambda= 0.7 type_calcul_masse= 2 option_recalcul_masse= 4
```

TABLE 15 – Exemple de déclaration de paramètres de l’algorithme de relaxation dynamique avec amortissement cinétique

```

dynamique_relaxation_dynam    #avec plus visualisation

PARA_TYPE_DE_CALCUL
# .....
# / type d'algorithme /
#.....
typeCalRelaxation= 1    lambda= 0.6    type_calcul_mass= 2    option_recalcul_mass= 0
parametre_calcul_de_la_masse_    casMass_relax= 3

avec_amortissement_cinetique_
    max_nb_decroit_pourRelaxDyn_ 1
    coef_arret_pourRelaxDyn_ 0.
    coef_redemarrage_pourRelaxDyn_ 0.01
    max_deltaX_pourRelaxDyn_ 0.02
    nb_max_dX_OK_pourRelaxDyn_ 5
    nb_deb_testfin_pourRelaxDyn_ 100
fi_parametre_amortissement_cinetique_

ARRET_A_EQUILIBRE_STATIQUE_ 2

```

1. typeCalRelaxation= 2, indique qu’il s’agit d’une relaxation avec amortissement visqueux. On se reportera à [16](#) pour le choix du type de calcul de la matrice visqueuse.
2. lambda= 0.7 : le paramètre lambda permet de pondérer le calcul de la masse
3. type_calcul_mass= 2 : permet de choisir le type de calcul de la matrice de pseudo-masse
4. option_recalcul_mass= 4 : permet de choisir le test de re-calcul de la matrice masse

Tous les paramètres sont facultatifs : les valeurs par défaut sont :

- type_calcul_visqu_critique= 1 ;
- typeCalRelaxation= 2 ;
- lambda= 0.605 ;
- type_calcul_mass= 2 ;
- option_recalcul_mass= 0 ;

La table [15](#) donne un exemple de déclaration.

6.3.3 Paramètres de contrôle de la masse

Dans le cas où ”type_calcul_mass= 1” le calcul utilise la méthode proposée par Barnes et Han et Lee, puis amélioré par Julien Troufflard (cf. travaux de thèse) et enfin ici étendue à des éléments quelconques cf. travaux de thèse de Javier Rodriguez Garcia. Pour les détails

TABLE 16 – Exemple de déclaration de paramètres de l’algorithme de relaxation dynamique avec amortissement visqueux

```
dynamique_relaxation_dynam #avec plus visualisation

PARA_TYPE_DE_CALCUL
# .....
# / type d'algorithme /
#.....
typeCalRelaxation= 2   lambda= 0.7   type_calcul_mass= 2   option_recalcul_mass= 4
parametre_calcul_de_la_masse_   casMass_relax= 5
parametre_recalcul_de_la_masse_   fac_epsilon= 100.
parametre_calcul_de_la_viscosite_   type_calcul_visqu_critique= 1   opt_cal_C_critique= 1   f_ = 0.9

ARRET_A_EQUILIBRE_STATIQUE_ 2
```

de la bibliographie, on se reportera aux travaux de thèse de Julien et de Javier puis aux références indiquées.

Les différents paramètres a utiliser sont présentés en 6.3.4.

Dans le cas où "type_calcul_mass= 2" le calcul utilise la matrice de raideur tangente. L'idée est ici de calculer cette matrice seulement à des moments particuliers. Le paramètre "option_recalcul_mass=" permet de choisir ces moments particuliers.

Les différents paramètres a utiliser sont présentés en 6.3.5 ; ;

6.3.4 Extension de la méthode de Barnes

Dans la première version, la masse est adaptée de manière à converger le plus rapidement possible vers la solution statique. L'algorithme a tout d'abord été proposé par Barnes puis une amélioration a été introduite par Julien Troufflard pour des éléments triangulaires et un comportement élastique (travaux de thèse) enfin dans la version d'He-rezh++, l'algorithme est étendu à des éléments quelconque et à des lois quelconques à l'aide d'une formule générale.

Exemple de déclaration :

```
parametre_calcul.de_la_masse_ alpha= 1. beta= 1. gamma= 1. theta= 1. casMass_relax= 1
```

Dans le cas d'éléments 2D, la masse est calculée selon :

$$k_{imax} = \sum_e \frac{ep}{4} \left(\alpha K + \beta \mu + \gamma \frac{\mathbf{I}_\sigma}{3} + \frac{\theta}{2} \sigma_{mises} \right) \quad (11)$$

Les paramètres α , β , γ , θ permettent ainsi de contrôler l'influence de chaque entité. L'algorithme comporte un autre paramètre de contrôle :

1. “casMass_relax” : qui permet de choisir entre plusieurs mode de calcul des termes de la matrice masse. Considérons tous les éléments “ N ” entourant un noeud :

(a) =1 : la formule (11) est cumulé au noeud :

$$k_{noeud} = \sum_{ne=1}^N k_{imax}$$

la valeur finale dépend donc du nombre d’éléments.

(b) =2 : on retient la valeur maximum de (11)

$$k_{noeud} = Max_N(k_{imax})$$

(c) =3 : on retient la valeur moyenne de (11) :

$$k_{noeud} = \frac{1}{N} \sum_{ne=1}^N k_{imax}$$

(d) =4 : idem le cas 3, et de plus on divise par la surface moyenne entourant le noeud, calculée de la manière suivante :

$$k_{noeud} = \frac{1}{N} \sum_{ne=1}^N \left(k_{imax} \right) / \left(\frac{S_{ne}}{NBN_{ne}} \right)$$

où S_{ne} est la surface de l’élément et NBN_{ne} est le nombre de noeuds de l’élément.

(e) =5 : idem le cas 1, et de plus on divise par la surface moyenne entourant le noeud, calculée de la manière suivante :

$$k_{noeud} = \sum_{ne=1}^N (k_{imax}) / \left(\frac{S_{ne}}{NBN_{ne}} \right)$$

où S_{ne} est la surface de l’élément et NBN_{ne} est le nombre de noeuds de l’élément.

Dans le cas d’éléments volumiques, et linéique, la masse est calculée selon une formule équivalente :

$$k_{imax} = \sum_e \frac{l_e}{4} \left(\alpha K + \beta \mu + \gamma \frac{\mathbf{I}_\sigma}{3} + \frac{\theta}{2} \sigma_{mises} \right) \quad (12)$$

avec “ l_e ” une longueur caractéristique telle que : $l_e = (volume_e)^{1/3}$ pour les éléments volumiques et $l_e = volume_e / (section\ moyenne)$ pour les éléments 1D.

Dans les formules précédentes ((a) à (e)) les termes $\frac{S_{ne}}{NBN_{ne}}$ sont remplacés par :

- $Volume_{ne} / NBN_{ne}$ pour les volumes
- $volume_{ne} / (section\ moyenne)_{ne} / NBN_{ne}$ pour les éléments 1D

Il est possible d'utiliser une convergence sur le résidu soit seul soit combinée avec la convergence de l'algorithme de relaxation cinétique, voir cf. 6.4 pour plus d'informations.

Bien que le pas de temps peut être quelconque, et donc n'intervient pas explicitement dans l'algorithme, il faut noter qu'il intervient pour piloter le chargement, ainsi que dans les paramètres généraux du calcul (temps maxi autorisé par exemple). Il faut donc choisir ce temps en fonction de ces éléments.

La table (17) donne un exemple d'utilisation de l'algorithme dans le cas d'un critère sur le résidu, tandis que la table (18) donne un exemple d'utilisation pour le cas d'un critère sur le déplacement.

Tous les paramètres sont facultatifs : les valeurs par défaut sont :

- alpha= 1;
- beta= 1.;
- gamma= 1.;
- theta= 1.;
- casMass_relax= 3;

TABLE 17 – Exemple de déclaration pour l'algorithme de relaxation dynamique avec un contrôle sur le résidu

```
#-----
# relaxation dynamique
#-----

dynamique_relaxation_dynam #avec plus visualisation

PARA_TYPE_DE_CALCUL
# .....
# / type d'algorithme /
#.....
typeCalRelaxation= 1  lambda= 20  type_calcul_mass= 1  option_recalcul_mass= -1
parametre_calcul_de_la_masse_  alpha= 1. beta= 1. gamma= 1. theta= 1. casMass_relax= 1

avec_amortissement_cinetique_
  max_nb_decroit_pourRelaxDyn_ 1
  coef_arret_pourRelaxDyn_ 0.
  coef_redemarrage_pourRelaxDyn_ 0.01
  max_deltaX_pourRelaxDyn_ 0.02
  nb_max_dX_OK_pourRelaxDyn_ 5
  nb_deb_testfin_pourRelaxDyn_ 100
fi_parametre_amortissement_cinetique_

ARRET_A_EQUILIBRE_STATIQUE_ 2
```

TABLE 18 – Exemple de déclaration pour l’algorithme de relaxation dynamique avec un contrôle sur le déplacement

```

TYPE_DE_CALCUL
#-----
# relaxation dynamique
#-----
#-----
# relaxation dynamique
#-----

dynamique_relaxation_dynam #avec plus visualisation

PARA_TYPE_DE_CALCUL
# .....
# / type d'algorithme /
#.....
typeCalRelaxation= 1   lambda= 20   type_calcul_mass= 1   option_recalcul_mass= -1
parametre_calcul_de_la_masse_   alpha= 1. beta= 1. gamma= 1. theta= 1. casMass_relax= 1

avec_amortissement_cinetique_
  max_nb_decroit_pourRelaxDyn_ 1
  coef_arret_pourRelaxDyn_ 0.
  coef_redemarrage_pourRelaxDyn_ 0.01
  max_deltaX_pourRelaxDyn_ 0.02
  nb_max_dX_OK_pourRelaxDyn_ 5
  nb_deb_testfin_pourRelaxDyn_ 100
fi_parametre_amortissement_cinetique_

```

6.3.5 Pseudo-masse fonction de la raideur réelle

Exemple de déclaration :

parametre_calcul_de_la_masse_ casMass_relax= 3

La technique proposée consiste à un calcul de la masse qui s'appuie sur la matrice réelle de raideur. Soit $[K_{ij}]$ la matrice de raideur réelle, donc calculée à partir du maillage réel, et du comportement réel. La masse élémentaire du noeud i m_i , utilisée pour construire la matrice masse diagonale est donnée par :

$$m_i = \frac{\lambda(\Delta t)^2}{2} S_i \quad (13)$$

avec λ un paramètre global d'ajustement, Δt arbitrairement mis à 1 et S_I calculé à l'aide de $[K]$. Différentes options sont disponibles, les différences se situent au niveau de la formule du calcul de la pseudo-masse.

- "casMass_relax= 0" : $S(a_i) = |K(a_i, a_i)|$: a=1,dim,
- "casMass_relax= 1" : $S(b_i) = MAX_{a=1}^{dim} |K(a_i, a_i)|$: b=1,dim ; on retient donc le terme diagonal dominant de la raideur suivant les dim axes,
- "casMass_relax= 2" : $S(a_i) = \sum_{j=1, b=1}^{n, dim} |K(a_i, b_j)|$ (théorème de Gershgorin)
- "casMass_relax= 3" : $S(c_i) = MAX_{a=1}^3 \sum_{j=1, b=1}^{n, dim} |K(a_i, b_j)|$ avec c=1,dim ; on retient le maxi de la somme des valeurs absolues de la raideur suivant les dim axes, et on applique ce maxi pour les dim directions. (théorème de Gershgorin également)
- "casMass_relax= 4" : $S(a_i) = MAX \left(2. |K(a_i, a_i)|, \sum_{j=1, b=1}^{n, dim} |K(a_i, b_j)| \right)$;
- "casMass_relax= 5" : $S(c_i) = MAX_a \left(MAX \left(2. |K(a_i, a_i)|, \sum_{j=1, b=1}^{n, dim} |K(a_i, b_j)| \right) \right)$,
 $c = 1, dim$;

Tous les paramètres sont facultatifs : les valeurs par défaut sont :

- casMass_relax= 3

La table (15) donne un exemple d'utilisation.

6.3.6 Contrôle du re-calcul de la masse

Exemple de déclaration :

```
parametre_recalcul_de_la_masse_fac_epsilon= 1.
```

L'idée est de minimiser autant que possible le calcul en continu de la masse. Concernant la syntaxe, on indique tout d'abord le mot clé : "parametre_recalcul_de_la_masse_" suivi des paramètres associés éventuelles. Les différentes options possible sont les suivantes :

1. option_recalcul_mass= -1 : la matrice masse est re-calculée a chaque itération. Il n'y a pas de paramètre associé supplémentaire.
2. option_recalcul_mass= 0 : la matrice masse est calculée au début du calcul de l'incrément et ensuite elle reste fixe. Il n'y a pas de paramètre associé supplémentaire.
3. option_recalcul_mass= 1 : après un calcul en début d'incrément, mise à jour à chaque maxi de l'énergie cinétique. Il n'y a pas de paramètre associé supplémentaire.
4. option_recalcul_mass= 2 : idem le cas 1, mais on garde la valeur maxi de la raideur entre la nouvelle et l'ancienne. Il n'y a pas de paramètre associé supplémentaire.
5. option_recalcul_mass= 3 : re-calcul après n cycles, valeur qui doit être indiquée apres le mot clé "ncycle_calcul=" de la manière suivante par exemple pour un calcul tous les 100 itérations :

```
parametre_recalcul_de_la_masse_ncycle_calcul= 100
```

par défaut ncycle_calcul= 100

6. option_recalcul_mass= 4 : re-calcul de la matrice masse lorsque l'indicateur suivant

$$\epsilon = MAX_{i=1}^{nbdll}(\epsilon_i) \text{ avec } \epsilon_i = \frac{\lambda(\Delta t)^2 |\Delta \ddot{X}_i|}{2 |\Delta X_i|} \quad (14)$$

est supérieur a "1*fac_epsilon ". Par défaut fac_epsilon = 1, on peut le modifier à l'aide du paramètre "fac_epsilon= une valeur " (ce type de controle provient de l'amortissement visqueux critique)

Exemple de déclaration : parametre_recalcul_de_la_masse_fac_epsilon= 0.8

par défaut fac_epsilon= 0.9

6.3.7 Contrôle matrice viscosité critique

On indique tout d'abord le mot clé : "parametre_calcul_de_la_viscosite_" suivi des paramètres associés éventuelles. Exemple de déclaration :

```
parametre_calcul_de_la_viscosite_type_calcul_visqu_critique= 1 opt_cal_C_critique= 0 f_= 1.3
```

Différentes techniques sont implantées pour approcher un amortissement critique, via un calcul de préférence peu coûteux en temps de calcul.

L'amortissement visqueux est introduit à l'aide d'une matrice diagonale.

$$[C] = c[M] \quad (15)$$

ω_0 est supposé être la fréquence la plus basse du système, approchée à l'aide du quotient de Rayleigh's.

$$\omega_0^2 \approx \frac{\Delta X^T K^n \Delta X}{\Delta X^T M \Delta X} \quad (16)$$

La matrice K^n n'étant pas directement accessible, on utilise une approximation dans le cadre d'un algorithme d'avancement de type différences finies :

$$\omega_0^2 \approx \frac{\Delta X^T {}^l K^n \Delta X}{\Delta X^T M \Delta X} \text{ avec } {}^l K_{ii}^n = \frac{\Delta R_{i(statique)}^n}{\Delta t \dot{X}_i^{n-1/2}} \quad (17)$$

Les différentes options pour le calcul de c sont fonction de ω_0 :

1. si "type_calcul_visqu_critique= 1 " : $c = 2 * \omega_0$
2. si "type_calcul_visqu_critique= 2 " : $c = \sqrt{(4 * \omega_0^2 - \omega_0^4)}$
3. si "type_calcul_visqu_critique= 3 " : $c = 2 * \sqrt{(\omega_0 / (1 + \omega_0))}$

Puis on indique la méthode retenue pour le calcul de ω_0 : deux méthodes possibles :

1. opt_cal_C_critique= 0 : $(\omega_0)^2 \approx (\Delta X^T \Delta R_{i(statique)}^n) / (\Delta X^T M \Delta X)$
2. opt_cal_C_critique= 1 : $(\omega_0)^2 \approx (\dot{X}^T \Delta R_{i(statique)}^n) / (\dot{X}^T M \dot{X})$

Il est prévu également un encadrement pour le paramètre c .

- si $(\omega_0)^2$ est négatif on pose $\omega_0^2 = 0$
- si $(\omega_0)^2 > f^2 * 4$, on pose $(\omega_0) = f^2 * 4$, par défaut : $f = 0.9$

Il est possible de changer la valeur de f avec le paramètre : f_- = suivi de la valeur (voir l'exemple de déclaration en début de paragraphe).

Tous les paramètres sont facultatifs : les valeurs par défaut sont :

opt_cal_C_critique= 1 ; f_- = 0.9 ;

6.3.8 Contrôle mode debug

Le mode debug permet de spécifier un nombre d'itération "n" tel que tous les "n" itérations, il y a une sortie de résultat. L'intérêt est de pouvoir suivre ainsi l'évolution de la déformée en fonction des itérations. La syntaxe est la suivante :

mode_debug_ = < un nombre "n" >

Ce paramètre doit-être sur une ligne séparée. La table 19 donne un exemple d'utilisation du mode debug. Si le nombre "n" = 0, on n'en tiens pas compte, c'est la valeur par défaut.

6.4 Critère d'arrêt en résidu en dynamique explicite

Dans le cas de la recherche d'une solution statique en dynamique, comme le processus est itératif, un critère d'arrêt est nécessaire. D'une manière générale, il est possible d'introduire un critère d'arrêt sur l'équilibre statique. L'idée est de mesurer la valeur du résidu interne plus le résidu externe, en dehors du résidu des forces d'accélération. A priori, pour tous les algorithmes, le résidu global est nul, ce qui implique que le résidu des forces d'accélération est égale à l'opposé du résidu statique. Pour mettre en route cette possibilité, on indique dans les paramètres de l'algorithme, après les paramètres spécifiques, également après les paramètres d'un amortissement cinétique éventuel, le mot clé : "ARRET_A_EQUILIBRE_STATIQUE_" suivi d'un entier tel que :

TABLE 19 – Exemple de déclaration pour l’algorithme de relaxation dynamique avec utilisation du mode debug

```

dynamique_relaxation_dynam #avec plus visualisation

PARA_TYPE_DE_CALCUL
# .....
# / type d'algorithme /
#.....
typeCalRelaxation= 1   lambda= 20   type_calcul_mass= 1   option_recalcul_mass= -1
parametre_calcul_de_la_masse_   alpha= 1. beta= 1. gamma= 1. theta= 1. casMass_relax= 1

mode_debug_= 2

avec_amortissement_cinetique_
    max_nb_decroit_pourRelaxDyn_ 1
    coef_arret_pourRelaxDyn_ 0.
    coef_redemarrage_pourRelaxDyn_ 0.01
    max_deltaX_pourRelaxDyn_ 0.02
    nb_max_dX_OK_pourRelaxDyn_ 5
    nb_deb_testfin_pourRelaxDyn_ 100
fi_parametre_amortissement_cinetique_

ARRET_A_EQUILIBRE_STATIQUE_ 2

```

1. ARRET_A_EQUILIBRE_STATIQUE_ 0 : valeur par défaut, indique que ce critère n’est pas pris en compte,
2. ARRET_A_EQUILIBRE_STATIQUE_ 1 : signifie que le résidu statique est effectivement utilisé comme critère,
3. ARRET_A_EQUILIBRE_STATIQUE_ 2 : signifie que le résidu statique et le critère de relaxation cinétique sont utilisés conjointement comme critères d’arrêt, le plus défavorable des deux est retenu.

Voir (19) pour un exemple de déclaration.

L’utilisation de ce paramètre est dédiée aux algorithmes dynamiques explicites classiques (sauf Runge Kutta cf.5.8 et les algorithmes de type Galerkin-discontinu) . Dans le cas des algorithmes non concerné il est a priori ignoré!!

7 “Utilitaires”

En fait ce type permet d’effectuer des traitements qui ne sont pas directement assimilables à un calcul éléments finis. Il peut s’agir par exemple d’action de pré ou post-traitement. Deux possibilités pour utiliser les “utilitaires”, soit avec un mot clé qui suit

le type de calcul selon la syntaxe : “avec plus <un mot clé>” ou alors au travers des paramètres de l’algorithme. La première méthode s’adresse au cas d’une action simple, en général sans paramètre de contrôle, la seconde concerne les actions plus complexes.

7.1 Transformation d’un maillage quadratique incomplet en quadratique complet

Par exemple le texte suivant :

```

          TYPE_DE_CALCUL
#-----
#  TYPE DE      |
#  CALCULS      |
#-----
      utilitaires   avec plus QuadIncVersQuadComp

```

indique que l’on désire transformer les éléments quadratiques incomplet du maillage en éléments quadratiques complets. Le ou les maillages initiaux ne sont pas modifiés, par contre il y a création de nouveaux fichiers contenant les maillages modifiés. L’utilitaire peut s’appliquer au cas :

- du quadrangle incomplet à 8 noeuds qui est transformé en un quadrangle complet à 9 noeuds,
- de l’hexaèdre incomplet à 20 noeuds qui est transformé en un hexaèdre complet à 27 noeuds,
- du pentaèdre incomplet à 15 noeuds qui est transformé en un pentaèdre complet à 18 noeuds.

A la suite de l’exécution du programme, il y a création de deux fichiers, le premier contenant la discrétisation, le second contenant les références. Le nom des fichiers est construit à l’aide du nom des maillages, suivi de ”_nevez.her” pour le premier fichier et suivi de ”_nevez.lis” pour le fichier des références.

Par exemple supposons que le maillage a pour nom ”tube”, les deux fichiers créés auront alors pour nom : tube_nevez.her et tube_nevez.lis.

Par rapport au maillage contenant des éléments incomplets, la numérotation est modifiée, aussi toutes les références initiales sont modifiées en conséquence de même évidemment que les numéros de connection des éléments.

Pour utiliser cet utilitaire, il est nécessaire d’indiquer un fichier .info valide, c’est-à-dire qui comporte la définition d’au moins une loi de comportement, des différents mots clés obligatoires ... même si la loi de comportement est farfelu!, qu’il n’y aucune condition limites après le mot clé ... Ceci est due au fait que le programme commence par créer complètement les différents maillages et les différentes grandeurs théoriquement nécessaires pour un calcul valide, avant toute action. Cependant les informations autres que celles directement liées aux maillages, ne sont pas utilisées.

7.2 Relocalisation des noeuds intermédiaires pour les arrêtes des éléments quadratiques

L'exemple de type de calcul suivant :

```
          TYPE_DE_CALCUL
#-----
#  TYPE DE      |
#  CALCULS     |
#-----
    utilitaires   avec plus relocPtMilieuQuad
```

indique que tous les points milieux des arrêtes quadratiques seront re positionnés aux milieux de l'arrête curviligne actuellement défini. Dans les cas où il n'y a pas d'arrête curviligne quadratique pour les éléments du maillage, la commande est ignoré. En sortie, il y a création d'un fichier .her et .lis correspondant au maillage relocalisé.

7.3 Sauvegarde des maillages en cours aux formats .her et .lis

Cette option permet de sauvegarder les maillages en cours par exemple après des translations ou rotations solides indiquées à la lecture.

L'exemple de ce type de calcul est le suivant :

```
          TYPE_DE_CALCUL
#-----
#  TYPE DE      |
#  CALCULS     |
#-----
    utilitaires   avec plus sauveMaillagesEnCours
```

7.4 Suppression des noeuds non référencés

Cette option permet de supprimer tous les noeuds d'un maillage (ou de plusieurs) qui ne sont pas référencés par des éléments. A priori, seules les noeuds référencés sont susceptible de participer au calcul, aussi le fait de supprimer les noeuds non référencés peut alléger la lisibilité du calcul. Cependant, cela ne pose pas de pb de conserver des noeuds non référencés "sauf" si l'on utilise l'algorithme de renumérotation. Ce dernier ne fonctionne qu'avec un maillage constitué de noeud, tous référencés.

Après que les noeuds non référencés aient été supprimés, les références de noeuds sont mises à jour en fonction de la nouvelle numérotation des noeuds restants. Les numéros de noeuds non référencés, sont supprimés.

L'exemple de ce type de calcul est le suivant :

```

        TYPE_DE_CALCUL
#-----
#  TYPE DE      |
#  CALCULS     |
#-----
    utilitaires   avec plus suppression_noeud_non_references

```

Remarque Il est également possible d’effectuer cette opération, juste après la lecture du maillage (cf ??), ce qui permet d’utiliser directement le nouveau maillage dans un autre calcul. Dans ce dernier cas, il n’y a pas de sauvegarde du nouveau maillage, contrairement à l’utilisation de l’algorithme “utilitaires”.

7.5 Renumérotation des noeuds

Cette option permet d’optimiser la numérotation des noeuds dans l’objectif de minimiser la largeur de bande de la matrice de raideur. L’algorithme actuellement implanté est celui de Cuthill Mac Kee, avec un choix de premier noeud (partie importante de l’algorithme) qui suit le début de l’algorithme de Gibbs (cf. Pironneau de Paris VI)

L’exemple de ce type de calcul est le suivant :

```

        TYPE_DE_CALCUL
#-----
#  TYPE DE      |
#  CALCULS     |
#-----
    utilitaires   avec plus renumerotation_des_noeuds

```

Remarque

- Il est également possible d’effectuer cette opération, juste après la lecture du maillage (cf 14), ce qui permet d’utiliser directement le nouveau maillage dans un autre calcul. Dans ce dernier cas, il n’y a pas de sauvegarde du nouveau maillage, contrairement à l’utilisation de l’algorithme “utilitaires”.
- L’utilisation de l’utilitaire, permet également d’optimiser la numérotation en tenant compte des conditions linéaires éventuelles.

7.6 Orientation des éléments

L’objectif est de vérifier et corriger éventuellement l’orientation des éléments du maillage. L’exemple suivant permet de mettre en oeuvre l’utilitaire :

```

        TYPE_DE_CALCUL
#-----

```

```

# TYPE DE      |
# CALCULS     |
#-----
      utilitaires   avec plus modif_orientation_element

```

On obtient alors un dialogue en mode texte, qui permet de choisir entre différents choix. Exemple de dialogue d'Herezh :

```

===== choix du module d'orientation      =====
vérification de l'orientation   ?           (rep veor)
orientation automatique pour un jacobien positif ? (rep oajp)
orientation des faces d'élément 2D ?       (rep faor)
fin                                 (rep   f)
réponse ?

```

1. vérification de l'orientation des maillages (sans modification)
2. vérification et construction de nouveaux maillages orientées, c'est-à-dire avec un jacobien toujours positif dans le cas des points d'intégrations mécaniques. Il y a une modification du maillage, mais pas d'action sur les références qui sont gardées telles quelles.
3. orientation des faces d'éléments 2D. Là il s'agit de modifier l'orientation des faces d'éléments coques-plaques ou membranes uniquement, de manière à récupérer des ensembles de faces ayant la même orientation. Il y a création de nouvelles références. Le fonctionnement de l'algorithme est explicité dans le paragraphe suivant.

7.6.1 Cas des éléments membranes, plaques et coques

Le programme demande de fournir un élément de départ. Deux formes de réponse sont possibles :

Affichage d'Herezh :

```

--- choix d'un element dont l'orientation servira de reference -----
le reperage peut se faire de differentes manieres :
par un numero d'element dans le maillage                -> (choix : 1)
(rep -1 si l'on veut tenter une orientation inverse pour les elements)
par des coordonnees d'un point (meme approximatives),
l'element choisit sera celui le plus proche de ce point -> (choix : 2)
(rep -2 si l'on veut tenter une orientation inverse pour les elements)
fin                                                       -> (choix : f ou 0)

```

Si l'on répond par un chiffre négatif, cela signifie que l'on désire une orientation inverse pour les éléments qui seront choisis (voir la suite de l'explication pour plus de précision).

Deux solutions possibles pour le premier élément qui servira de référence, soit via un numéro soit via un lieu géométrique.

Ensuite on fournit un angle maxi, au delà duquel on considère que la continuité d'orientation entre deux éléments mitoyens n'est pas à imposer. Exemple de dialogue d'Herezh :

```
angle maxi (en degré) au dessus duquel on considérera
qu'il n'y a plus de continuité : 50
```

```
angle= 50 (0.872665 rd)
```

Enfin on peut soit laisser le programme donner un nom à la référence générée, qui sera associée automatiquement au groupe d'éléments ayant une même orientation, soit indiquer une nom particulier.

A partir du premier élément qui sert de référence, le programme repère tout d'abord les éléments mitoyens, c'est-à-dire les éléments qui ont une arête en commun (il peut y en avoir plusieurs pour une arête). Seule les éléments dont la normale fait un angle en valeur absolue $< 50^\circ$ avec celle de l'élément de référence, sont conservés. Puis parmi ces derniers, on regarde si l'orientation (le sens de la normale) est la même que celle de l'élément, de référence, si non on l'inverse. Puis ces éléments mitoyens sont à leur tour choisis comme élément de référence et le même algorithme leur est appliqué. Ainsi de proche en proche tout un groupe d'éléments sont définis ayant une même orientation. Le programme génère alors deux nouvelles références : une qui groupe l'ensemble des éléments, et une qui groupe l'ensemble des faces de ces éléments (en fait une seule face par élément étant donné qu'il s'agit d'éléments plaque-coque ou membrane).

Exemple de dialogue d'Herezh :

```
nombre d'élément inversés 9031
création de la référence : E_ref_de_meme_orientation_que_ele_1- (18242 elements )
création de la référence : F_ref_de_meme_orientation_que_ele_1 (18242 surfaces )
des groupes d'éléments mitoyens ont été détectés comme ayant des orientations
différentes
voulez-vous les orienter (arbitrairement) par groupes mitoyens ? (rep o ou n )
```

Dans le cas où le groupe ainsi formé, comprend tous les éléments du maillage, l'opération est terminée. Sinon, le programme indique (comme sur l'exemple précédent) qu'il reste des éléments non orientés. Il est alors possible de créer de nouveaux groupes suivant la même méthodologie, en choisissant un élément restant arbitrairement comme référence, puis après orientation et création de nouvelles références, est répété jusqu'à ce que tous les éléments du maillage est été analysés.

Exemple de dialogue d'Herezh :

...


```

nombre d'élément inversés 0
création de la référence : E_ref_de_meme_orientation_que_ele_57884- (136 elements )
création de la référence : F_ref_de_meme_orientation_que_ele_57884 (136 surfaces )
nombre d'élément inversés 0
création de la référence : E_ref_de_meme_orientation_que_ele_58293- (25 elements )
création de la référence : F_ref_de_meme_orientation_que_ele_58293 (25 surfaces )
===== choix du module d'orientation =====
vérification de l'orientation ? (rep veor)
orientation automatique pour un jacobien positif ? (rep oajp)
orientation des faces d'élément 2D ? (rep faor)
fin (rep f)
réponse ?

```

7.7 Création d'un maillage SFE

Cette option permet de créer un maillage SFE à partir d'un maillage triangulaire linéaire.

La syntaxe de ce type de calcul est la suivante :

```

          TYPE_DE_CALCUL
#-----
#  TYPE DE      |
#  CALCULS     |
#-----
          utilitaires   avec plus creationMaillageSFE

```

Ensuite, dans le fichier .info, on indique un maillage triangulaire linéaire (ou plusieurs). Ces maillages seront transformés en maillage SFE triangulaire. Chaque éléments SFE a un tableau de connexion à 6 noeuds. Le (ou les) nouveau(x) maillage(s) sont sauvegardés dans des fichiers .her et .lis, dont le nom est celui du maillage triangulaire post-fixé de la chaîne “_nevez”.

Remarque

1. Seuls les maillages entièrement triangulaires sont traités, sinon le maillage est ignoré.
2. Il est nécessaire que le reste du fichier .info soit cohérent, car, même si le reste des informations lues n'est pas utilisées, il y a vérification de la cohérence. Par contre, il est nécessaire d'utiliser un fichier .info minimal, sans conditions limites ou tous les déplacements bloqués par exemple, et avec les paramètres de contrôle par défaut.
3. a priori il ne faut pas indiquer de sortie d'information, en particulier pas de mot clé “POINTS_INTEGRATION” car d'une part cela ne sert à rien, car aucune information n'a été calculée, et d'autre part cela génère actuellement une erreur provenant du fait qu'il existe à la sortie du programme deux maillages et qu'en générale un seule maillage à été lue au démarrage. Les références initiales constituées ne sont

donc pas en général préfixées par un nom de maillage, d'où une erreur au moment de la sortie aux points d'intégrations pour une référence non préfixée (à moins de les avoir préfixées explicitement dans le .info, mais ce n'est pas habituelle!).

4. Dans le cas où il n'y a qu'un seul maillage, et qu'il n'a pas de nom, par défaut il reçoit le nom "premier_maillage". Ainsi le maillage SFE créé devient "premier_maillage_nevez.her" et ".list".

7.8 Fusion de noeuds très voisins

Cette option permet de fusionner des noeuds qui sont très proches géométriquement. La syntaxe de ce type de calcul est la suivante :

```

                TYPE_DE_CALCUL
#-----
#  TYPE DE      |
#  CALCULS      |
#-----
    utilitaires avec plus fusion_de_noeuds

```

Ensuite le logiciel demande de manière interactive, la distance en dessous de laquelle on fusionne les noeuds. Les contraintes suivantes sont prises en compte :

- l'algorithme de fusion utilise les coordonnées initiales des noeuds,
- deux noeuds sont fusionables s'ils appartiennent à des éléments différents,
- et à un même maillage.

Les noeuds supprimés du maillage initiale, sont également supprimés des listes de références de noeuds. En sortie, un nouveau maillage et les références associées sont créés suivant la syntaxe suivante où "nom" représente le nom du maillage initiale :

- nom_nevez.her : contient le nouveau maillage
- nom_nevez.lis : contient les nouvelles références.

7.9 Fusion d'éléments superposés

Cette option permet de fusionner des éléments qui sont superposés. La fusion de deux éléments s'effectue si les conditions suivantes sont réunies :

- les deux éléments appartiennent à un même maillage,
- ils possèdent les mêmes noeuds dans leur connexion,
- ils possèdent le même type de géométrie, d'interpolation, et sont relatifs au même type de calcul (ex : mécanique, ou thermique ou ...)

La syntaxe de ce type de calcul est la suivante :

```

                TYPE_DE_CALCUL
#-----

```

```

# TYPE DE      |
# CALCULS     |
#-----
    utilitaires avec plus fusion_elements

```

Les éléments supprimés du maillage initial sont également supprimés des listes de références d'éléments, et également des références de faces, d'arêtes et de points d'intégration. Le résultat dépend de l'ordre de numérotation des éléments. Pour un lot d'éléments identiques, c'est celui de numéro le plus faible qui est retenu, et seules les références associées à cet élément sont conservées.

En sortie, un nouveau maillage et les références associées sont créés suivant la syntaxe suivante où "nom" représente le nom du maillage initiale :

- nom_nevez.her : contient le nouveau maillage
- nom_nevez.lis : contient les nouvelles références.

7.10 Fusion de maillages

Cette option permet de créer un nouveau maillage globalisant un ou plusieurs maillages déjà existants. Dans ce nouveau maillage :

- L'ensemble des noeuds comporte une seule numérotation, qui débute à 1 et se termine au nombre total cumulé de noeuds,
- idem pour les éléments. Une nouvelle connexion cohérente est alors créée.
- toutes les références initiales sont reportées avec la méthode suivante concernant le maillage de position "i" (dans la liste des maillages à fusionner) : le nom des références attachées au maillage "i" comporte le suffixe : "i"

La syntaxe de ce type de calcul est la suivante :

```

                TYPE_DE_CALCUL
#-----
# TYPE DE      |
# CALCULS     |
#-----
    utilitaires   avec plus   fusion_maillages

```

Après lecture des maillages, le programme propose différents choix :

```

=====
|                fusion de maillages                |
|                                                    |
=====

===== choix des maillages à fusionner             =====

```

liste des maillages disponibles	(rep list)
vider la liste a fusionner	(rep vlis)
nom du maillage issu de la fusion	(rep n fus)
fusionner tous les maillages	(rep to)
ajout d'un nom de maillage a fusionner	(rep nafu)
fin (appel du programme de fusion)	(rep f)
reponse ?	

Une fois que l'on a défini les paramètres de la fusion, un nouveau maillage est créé, et est sauvegardé dans deux fichiers de même nom, "nom.her" pour les noeuds et éléments, et "nom.lis" pour les références, "nom" étant le nom de maillage fourni durant la phase de dialogue interactive.

La fusion n'intervient qu'après la lecture complète des maillages. Ainsi, toutes les opérations indiquées dans le .info, du type : "déplacements solides" , "renumérotation", "suppression de noeuds non référencés" etc. , sont effectués au préalable avant l'appel à l'algorithme de fusion.

Il est possible d'utiliser plusieurs fois les mêmes informations de maillage en entrée, c'est-à-dire la même liste de noeuds, éléments et références, par contre il faut que ces informations soient précédées d'un nom de maillage différent pour chaque maillage ainsi défini !

Remarque : En combinant les opérations de fusion et de suppression des noeuds et éléments superposés, on peut ainsi réaliser des opérations simples d'additions booléennes de maillages élémentaires, obtenues par exemple avec stamm. L'objectif n'est cependant pas de remplacer un mailleur, qui comporte évidemment toujours beaucoup plus de possibilités. L'opération de fusion est en revanche particulièrement intéressante lorsque l'on dispose déjà de plusieurs maillages que l'on ne veut (ou peut) pas modifier aisément.

8 "Information"

8.1 Définition de références pour un maillage existant

De nombreux éléments du calcul font appel à la notion de références, qui sont en fait des listes d'entités groupée autour d'un nom. Par exemple, les conditions limites nécessitent l'utilisation de références de noeuds, de face, d'arêtes, d'éléments (cf.18).

Alors qu'il est relativement facile de déterminer une liste de noeuds ou d'éléments à l'aide d'un mailleur standart, il n'est pas du tout évident d'obtenir les mêmes informations pour les faces, les arêtes et les points d'intégrations. La raison, est que ces informations sont spécifiques au types de numérotations utilisées dans herezh++ (connections : noeuds, faces, arêtes, points d'intégration). Il y a donc une possibilité de définir interactivement de nouvelles références à l'aides de contraintes géométriques simples. Le mode d'emploi de ces outils est l'objet de ce chapitre.

Tout d'abord il est nécessaire de définir un fichier .info de départ, qui contient le maillage et un ensemble cohérent de données, mais seules les données de maillage seront utilisées, aussi les autres informations peuvent-être quelconques, pourvu qu'elles soient cohérentes, de manière à ne pas générer des erreurs de lectures, car elles seront lues en même temps

que le maillage. La table (20) donne un exemple de début de fichier .info permettant de mettre en route les utilitaires de recherche de référence. Le nom de l’algorithme est “informations” et le mot clé associé à la définition de référence est “creation_reference”. Le reste du fichier suit les règles habituelles.

TABLE 20 – Exemple d’utilisation de l’algorithme “informations” pour définir de nouvelles références.

```
dimension 3

#-----
# definition facultative du niveau d impression
#-----
niveau_commentaire 3

TYPE_DE_CALCUL
#-----

informations avec plus  creation_reference

# ---- importation du mailllage
< p100x50x10_10_4_3.her

etc..
```

Une fois Herezh++ démarré, on obtient un menu du type de celui qui suit :

```
=====
|                creation interactive de references                |
=====

il y a 1 maillages a considerer

ref de noeuds                (rep no)
ref d'elements               (rep el)
ref de faces                 (rep fa)
ref d'arretes                (rep ar)
ref de pt d'integration d'element (rep pt)
effacer la liste actuelle    (rep ef)
fin def des types de ref voulu (rep f)
reponses : (il peut y en avoir plusieurs differentes)
```

On choisit les types de références que l'on veut créer (on peut soit en choisir certaines soit toutes les indiquer). Pour le cas particulier des références de points d'intégration, il est nécessaire de préciser de plus le type de ddl qui est associé au point d'intégration. En effet, il est possible d'avoir plusieurs jeux de point d'intégration différents dans un même calcul. Une fois validé un deuxième menu apparaît, qui permet d'indiquer le nom que l'on veut donner aux nouvelles références. Il y a vérification que l'on n'utilise pas des noms déjà existants.

Ensuite, dans le cas où l'on veut définir des références de point d'intégration, un petit menu permet d'afficher quelques informations concernant les points d'intégrations existants.

```

---- information concernant les pt d'integ existant -----
les coor d'un pt d'integ d'un ele choisit par numero      (rep chnbe )
les coor d'un pt d'integ d'un ele choisit le plus pres de (rep npres )
les coor des pt d'integ d'un elem choisit par numero     (rep lespti )
fin                                                         (f )

```

On peut ainsi avoir les coordonnées d'un point d'intégration d'un numéro donné et pour un élément donné. On peut également obtenir ces deux numéros à partir d'un point (dans la matière!!) proche. Enfin, on peut obtenir la liste des coordonnées des points d'intégration d'un élément.

Vient ensuite la définition des contraintes. Cette partie est la plus intéressante. L'idée est de définir un ensemble de contraintes géométriques simples, qui conduiront à la liste d'item que l'on veut retenir. Par exemple en 3D on a :

```

pres d'un plan                (rep pres_plan)
d'un cote du plan            (rep cote_plan)
entre deux plans //          (rep entre_plans)
pres d'un cylindre           (rep pres_cylindre)
a l'interieur d'un cylindre  (rep in_cylindre)
a l'exterieur cylindre       (rep ex_cylindre)
entre deux cylindres //      (rep entre_cylindres)
pres d'une sphere            (rep pres_sphere)
dans une sphere              (rep in_sphere)
a l'exterieur d'une sphere   (rep ex_sphere)
entre deux spheres concentriques (rep entre_spheres)
pres d'un point              (rep pres_point)
pres d'une droite            (rep pres_droite)
pres d'un cercle             (rep pres_cercle)
effacer la liste des methode (rep ef)
fin def methodes             (rep f)

```

1. “pres_plan” : condition qui restreint la recherche aux points distant d’une grandeur donné à un plan,
2. “cote_plan” : condition qui restreint la recherche aux points d’un certain coté d’un plan,
3. “entre_plan” : condition qui restreint la recherche aux points entre deux plans,
4. “pres_cylindre” : condition qui restreint la recherche aux points distant d’une grandeur donné d’un cylindre circulaire,
5. “in_cylindre” : condition qui restreint la recherche aux points intérieur à un cylindre,
6. “ex_cylindre” : condition qui restreint la recherche aux points extérieur à un cylindre,
7. “entre_cylindres” : condition qui restreint la recherche aux points compris entre deux cylindres concentriques,
8. “pres_sphere” : condition qui restreint la recherche aux points distant d’une grandeur donné d’une sphère ,
9. “in_sphere” : condition qui restreint la recherche aux points intérieur à une sphère,
10. “ex_sphere” : condition qui restreint la recherche aux points extérieur à une sphère,
11. “entre_spheres” : condition qui restreint la recherche aux points compris entre deux sphères concentriques,
12. “pres_point” : condition qui restreint la recherche aux points distant d’une grandeur donné à un point de référence,
13. “pres_droite” : condition qui restreint la recherche aux points distant d’une grandeur donné à une droite de référence,
14. “pres_cercle” : condition qui restreint la recherche aux points distant d’une grandeur donné à un cercle de référence,

Dans le cas de conditions “près de”, et pour les éléments, il suffit qu’un des noeuds de l’élément ou de la face ou de l’arête satisfasse la condition pour que l’élément, la face ou l’arête soit recevable. Par contre pour les autres conditions il faut que tous les noeuds de l’éléments (ou de la face de l’élément, ou de l’arête de l’élément) satisfasse la condition, pour que l’élément (la face, l’arête) soit recevable.

Il est possible d’enchaîner un nombre quelconque de condition. Le résultat finale = l’intersection du résultat de chaque condition.

9 Galerkin discontinu

Les algorithmes de la famille “Galerkin discontinu” intègrent dans leur formulation, la possibilité intrinsèque d’avoir des discontinuités. Celles-ci peuvent être spatiales ou temporelles. Actuellement, seules les discontinuités temporelles sont prises en comptes.

9.1 Algorithme explicite temporelle de Bonelli

Il s’agit d’intégrer l’équation d’équilibre spatial et temporel, sur un pas de temps. La méthode est purement explicite, avec une précision de convergence d’ordre 3. Pour chaque

pas de temps, les inconnues du problème sont les positions et les quantités de mouvement à t^+ et à $t + \Delta t^-$. Ainsi en théorie, la taille du problème est multipliée par 4 par rapport à une méthode classique DFC par exemple. De manière à optimiser la résolution, d'une part les quantités de mouvement sont exprimées explicitement en fonction des autres ddl, et d'autre part une résolution itérative est mise en place. Trois cas sont disponibles : 1, 2 ou 3 itérations. A priori l'optimum est le cas 2 qui est le paramètre par défaut. Il est également possible de régler le rayon spectral de la matrice d'amplification, à la bifurcation. Plus on est proche de 1, moins il y a d'atténuation des hautes fréquences numériques. Par défaut le paramètre vaut 0.6.

L'intégration temporelle de la puissance interne et externe est effectuée via une quadrature de Gauss. Par défaut on utilise 3 points d'intégration.

Ainsi 3 paramètres sont disponibles pour le paramétrage de l'algorithme : "k_max_", "rho_b_" et "nb_pt_int_t_".

La table (21) donne un exemple de l'utilisation de l'algorithme.

TABLE 21 – Exemple d’utilisation de l’algorithme de Bonelli.

```

TYPE_DE_CALCUL
#-----
# TYPE DE CALCULS | sous type |
#-----
dynamique_explicite_bonelli avec plus visualisation

#-----
#| parametres (facultatifs ) associes au calcul de dynamique explicite DG |
#| definition des parametres de controle pour Bonelli (Galerkin Discontinu) |
#-----

PARA_TYPE_DE_CALCUL
#-----
# definition des parametres de calcul |
#-----

# .....
# / k_max : 2 nombre de boucle de correction (1, 2 ou 3), l’optimun est a priori 2 /
# / rho_b : 0.6 le rayon spectral de la matrice d’amplification, a la bifurcation /
# / pour k_max=1 (E-1C) -> rho_b peut appartenir a [0.34,1.] /
# / pour k_max=2 (E-2C) -> rho_b peut appartenir a [0.0,1.] /
# / pour k_max=3 (E-3C) -> rho_b = 0.4 (de maniere arbitraire) /
# / nb_pt_int_t : 3 le nombre de points d’integration pour la quadrature temporelle /
# / forces internes et externes /
# / ** parametres facultatifs : (mettre les parametres sur une meme ligne et /
# / dans l’ordre (meme si certain manque) *** /
#.....

k_max_ 2 rho_b_ 0.4 nb_pt_int_t_ 2

# / a titre de test il est egalement possible d’indiquer directement la valeur de b /
# / au lieu de celle de rho_b, selon la syntaxe b_b_ et une valeur par ex: b_b_ 0.5 /
# / dans ce cas le pas critique est celui de la dfc /

```

Quatrième partie
Maillages

10 Outils de création de maillage

Herezh++ lit un seul type de format de maillage. De manière pratique les maillages sont contenus dans des fichiers avec généralement l'extension .her (mais cette extension n'est pas obligatoire). Il existe 4 possibilités pour créer un maillage lisible par Herezh++.

1. **Utilisation du logiciel libre Stamm** : Stamm (veut dire tricot en breton) permet de créer très rapidement, c'est son objectif principal, à l'aide d'un jeux de questions réponses, des géométries simples telles que : barres, plaques rectangulaires, rondes ou parallélogrammes, cylindres creux ou pleins, entiers ou une portion, dômes ou portion de dômes, parallélépipède, anneaux ... Ces géométries sont en particulier, souvent utilisées pour des calculs rapides ou des tests. Outre la rapidité de création, Stamm fourni automatiquement un ensemble complet de références de noeuds, d'éléments, d'arêtes et de face. Ces références sont toujours repérées de manière identique sur la géométrie. Ainsi il est possible de garder exactement les mêmes conditions limites dans le .info, pour un jeux de géométries qui diffèrent uniquement par les dimensions. Stamm génère des interpolations classiques : linéaires, quadratiques (complètes ou non), cubique dans certains cas. Stamm intègre régulièrement de nouvelles options, cependant il n'est pas prévu de l'étendre à des structures complexes, d'autres outils existent remplissant parfaitement cette objectif.
2. **Utilisation du logiciel Castem** : Castem (développé par le CEA) est un logiciel d'éléments finis complets, intégrant un pré et post processeur. Le pré processeur possède plusieurs avantages :
 - Il possède une large gamme de méthodes différentes pour générer des maillages structurés ou non.
 - La méthodologie de construction d'un maillage est simple et "naturelle" (relativement à la méthode des éléments finis).
 - Il est possible de créer un maillage à partir d'un fichier de commande, qui sert alors de "script" de contrôle. Il est alors très simple et rapide de modifier une caractéristique du maillage, qui est alors un maillage paramétrique.

Une fois le maillage construit, il faut utiliser l'utilitaire CAST3M-HEREZH développé par Hervé Laurent, qui produit un fichier de maillage .her et éventuellement un fichier .lis qui contient des références de noeuds et d'éléments. On se reportera aux pages Web personnelles d'Hervé Laurent pour plus d'information sur l'utilitaire. Le seul point un peu négatif de Castem est son interface graphique, assez frustré.

3. **Utilisation du logiciel libre Gmsh** : Gmsh est un pré et post processeur d'origine Universitaire. Le fonctionnement est assez proche du logiciel Castem avec des avantages et inconvénients par rapport à ce dernier.

Avantages

- L'interface graphique de Gmsh est très performantes et conviviale.
- La génération de maillages non structurés est performantes,

Principale défaut :

- La génération de maillages structurés est limitée.

Gmsh crée un maillage dans un format propre. Ensuite on utilise le script perl (développé par Gérard Rio) : gmsh2her, qui permet de traduire le format Gmsh en format her. On trouvera le script perl sur la page perso de Gérard Rio.

4. **Utilisation d'un mailleur quelconque** : Il est en général très simple, de transformer un maillage d'un format quelconque (mais évidemment connu) au format .her. La méthode la plus rudimentaire, est d'utiliser un simple éditeur de texte. La méthode est souvent efficace mais peut-être fastidieuse. Le principal problème provient souvent des conditions limites, qui nécessite des références (de noeuds, d'éléments, d'arêtes de faces...). Or il n'est pas toujours simples de récupérer les références à partir d'un mailleur quelconque. Une solution, une fois le maillage transformé en .her, est d'utiliser directement Herezh++ pour reconstruire ces références. On se référera au chapitre (8.1) pour la méthodologie à adopter pour créer des références.

11 Liste des informations contenues dans un maillage et syntaxe

Un maillage est constitué normalement :

- d'un nom : mais ce paramètre n'est pas obligatoire dans le cas ou il n'y a qu'un seul maillage. Dans ce cas le nom du maillage est par défaut : "premier_maillage". Dans le cas où l'on veut indiquer un nom de maillage différent on utilise le mot clé : "nom_maillage" suivi du nom que l'on veut donner au maillage qui va suivre. Dans le cas où il y a plusieurs maillage définit l'utilisation d'un nom de maillage pour chacun est obligatoire pour pouvoir les repérer. En particulier lorsque l'on utilise des références de noeuds, d'éléments, de faces ou d'arrêtes il est nécessaire d'indiquer préalablement le nom de maillage auquel la référence se rapporte, par exemple lors de la définition des conditions limites, chargement ...
- des noeuds : La définition des noeuds est précédée du mot clé obligatoire "noeuds" suivi sur la ligne suivant du nombre de noeud, puis sur les lignes suivantes pour chacune des coordonnées des noeuds.
- de références de liste de noeud (cf.18) : Ces listes sont optionnelles. On verra par la suite que ces listes permettent de manipuler globalement un ensemble de noeuds par exemple pour imposer des conditions limites.
- des éléments : La définition des éléments est précédée du mot clé obligatoire "elements" suivi sur la ligne suivante, comme pour les noeuds, du nombre d'éléments qui constituent le maillage, puis de la définitions de chaque élément. Chaque élément est ainsi défini par un numéro, un ou plusieurs noms, et le tableau de connection qui indique les numéros des noeuds sur lesquels l'élément s'appui.
- de référence de liste d'éléments, d'arrêtes, de faces ou de noeuds (cf.18). Ces listes fonctionnent comme celles des noeuds. Elles sont également optionnelles.

Voici un exemple d'un maillage constitué d'un seul élément hexaédrique.

```
# un paralelepipedo rectangle:
# - longueur      1.00000000000000
# - largeur       1.00000000000000
# - hauteur       1.00000000000000
# - maillage LINEAIRE      1 X 1 X 1
```

noeuds

8 NOEUDS

```
#-----  
#NO DU|          X          |          Y          |          Z          |  
#NOEUD|          |          |          |          |  
#-----  
  
      1          0.          0.          0.  
      2          0.          0.          4.00000000000  
      3          0.          4.00000000000          0.  
      4          0.          4.00000000000          4.00000000000  
      5 4.00000000000          0.          0.  
      6 4.00000000000          0.          4.00000000000  
      7 4.00000000000          4.00000000000          0.  
      8 4.00000000000          4.00000000000          4.00000000000
```

definition des listes de références de noeuds

```
N_1      2 4 6 8  
N_3      1 5  
N_2      1 3 5 7  
N_tout   1 2 3 4 5 6 7 8
```

elements -----

1 ELEMENTS

```
#-----  
# NO      Type          Noeuds          |  
#-----  
  
      1 HEXAEDRE LINEAIRE          1 5 7 3 2 6 8 4
```

definition des listes de references d'elements

```
ELEMENTS 1  
F_1 1 4
```

En général, la définition des maillages est obtenue automatiquement à l'aide de mailleurs automatiques, à la suite desquels on utilise des interfaces de transcription automatique dans le format d'herezh ou très proche de ce format. Dans ce dernier cas il est nécessaire de terminer la transcription manuellement à l'aide d'un éditeur de texte.

Les informations générées par les mailleurs sont en générales de taille importante d'où l'utilisation courante de fichier inclus dans le fichier principal .info. Certaine fois, pour améliorer la clarté des informations, les listes de références sont placés dans un fichier à part. Par exemple on aura pour le fichier .info :

```
dimension 1
TYPE_DE_CALCUL
non_dynamique

# premier maillage

< cube.her # ce nom indique un fichier qui contient effectivement le maillage
< cube.lis # ce nom indique un fichier qui contient les listes de references

# etc.
```

Remarque 1) Dans le cas où certaine noeuds ne sont pas utilisées, normalement cela ne perturbe pas le déroulement du calcul, les noeuds sont simplement ignorés.

2) La numérotation indiqué pour chaque noeud est vérifiée. Cependant on peut utiliser des numéros farfelus, dans ce cas un message de Warning indique l'incohérence, par contre en interne c'est l'ordre de lecture qui induit la numérotation prise en compte par exemple pour les tableaux de connection et pour les références !

12 Mouvements solides

A la suite de la définition de chaque maillage, il est possible d'effectuer une suites de mouvements solides qui affecte la position initiale du maillage. Ces mouvements solides sont ainsi effectués avant toute considération mécanique. La position finale obtenue est alors "la condition initiale neutre du calcul" à ne pas confondre avec l'initialisation, qui elle affecte les positions à "t" du solide (cf.36).

La définition de mouvements solides s'effectue en indiquant à la suite des référence d'éléments, de face et d'arêtes :

1. tout d'abord le mot clé : "def_mouvement_solide_initiaux_"
2. ensuite la définition éventuelle de mouvements solides, qui commencent par le mot clé : "mouvement_solide_" puis sur les lignes qui suivent une succession de translations, nouvelle définition d'un centre de rotation, et rotations effectuées dans l'ordre où elles apparaissent, terminées par le mot clé "fin_mouvement_solide_".

Une translation est définit par un mot clé "translation_=" suivi par les composantes d'un vecteur déplacement.

Un nouveau centre de rotation est définit par le mot clé "centre_=" suivi par les composantes du nouveau centre de rotation, qui sera pris en compte pour les rotations qui suivent.

Une rotation est défini par un mot clé “rotation_=” suivi par 3 composantes d’un vecteur rotation. Dans le cas d’un espace de dimension 1, il n’y a pas de rotation possible. Dans le cas d’un espace de dimension 2, la seule rotation possible est celle autour la composante 3 (z), c’est-à-dire seule la troisième composantes du vecteur rotation est utilisée (valeur supposée en radian par défaut, on peut également utiliser des degrés, voir explications qui suivent). En 3D, on considère que la première composante est une rotation finie autour de x puis la seconde composante est une rotation autour de y et enfin la troisième composante est une rotation autour de z. Les 3 rotations sont toujours effectuées dans l’ordre x puis y puis z. Si l’on veut par exemple l’ordre inverse il faut définir 3 rotations simples dans l’ordre voulue.

Il est possible d’enchaîner autant de mouvements solides que l’on désire.

La table (22) donne un exemple de syntaxe pour définir deux mouvements solides.

TABLE 22 – Exemple de déclaration de deux mouvements solides : une translations suivant z et une rotation autour de l’axe y

```
def_mouvement_solide_initiaux_
  mouvement_solide_ # def de mouvements solides
    translation_= 0. 0. 0.2
    centre_=      10. 0. 0.
    rotation_=    0. 0.2 0.
  fin_mouvement_solide_
```

Par défaut les rotations sont en radians, dans le cas où l’on veut indiquer des valeurs de rotation en degré, on indique après les coordonnées, le mot clé “en_degre_” (cf. la table 23 pour un exemple).

Il est également possible d’utiliser la position initial d’un noeud pour définir le centre de rotation. Dans ce cas on indique le mot clé “centre_noeud_=” suivi du numéro du noeud, en lieu et place du mot clé “centre_=". La table 23 donne un exemple de déclaration.

TABLE 23 – Exemple de déclaration d’un mouvement solide de rotation autour d’un noeud

```
def_mouvement_solide_initiaux_
  mouvement_solide_      # def de mouvements solides
    centre_noeud_=       3 # c’est le noeud "3" qui sera le centre de rotation
    rotation_=           0. 0.2 0. en_degre_
  fin_mouvement_solide_
```

13 Suppression des noeuds non référencés

Il est possible de supprimer les noeuds non référencés par les éléments. A priori ces noeuds ne prennent pas part au calcul et ils ne gênent pas le calcul. Cependant, dans le cas où on veut les supprimer il est possible de le faire via le mot clé “suppression_noeud_non_references_”. La table (24) donne un exemple d’utilisation de la suppression des noeuds non référencés. Il faut noter que ce mot clé doit apparaître après l’application de mouvement solide (si c’est demandé cf 12).

Il est possible également d’utiliser un algorithme utilitaire, pour modifier définitivement le maillage (cf 7.4).

14 Renumérotation des noeuds

Il est possible d’effectuer une optimisation de la numérotation des noeuds. La méthode implantée est actuellement la méthode de Cuthill Mac Kee. Après exécution, la numérotation est modifiée, ainsi que les références des noeuds sont reconstruites en cohérence avec cette nouvelle numérotation. Pour mettre en oeuvre l’optimisation il suffit d’indiquer le mot clé “renumerotation_des_noeuds_” à la suite de la déclaration des maillages (ou du maillage). Cependant il faut noter que ce mot clé doit apparaître après la suppression des noeuds non référencés (si cette méthode est appliquée cf. 13) et après l’application de mouvement solide (si c’est demandé cf 12), et après la création automatique de références de frontières (si c’est demandé cf. 18.1). La table (24) donne un exemple d’utilisation de renumérotation.

Il est possible également d’utiliser un algorithme utilitaire, pour modifier définitivement le maillage (cf 7.5).

TABLE 24 – Exemple de d’utilisation des méthodes de suppression des noeuds non référencés et de renumérotation

```
# importation du maillage

< eprou1.her
< eprou1.lis

# --- demande de suppression des noeuds non references ----

    suppression_noeud_non_references_

# demande de renumeration

    renumeration_des_noeuds_
```

La demande de suppression de noeuds et/ou de renumérotation est à indiquer après la déclaration de chaque maillage. Concernant la numérotation, celle-ci n’est optimisée que pour le maillage considéré.

Dans le cas où l'on veut effectuer une renumérotation pour tous les maillages, avec une optimisation globale, il faut indiquer à la suite de tous les maillages, le mot clé `”renumerotation_tous_maillages_”`, sur une ligne seule.

15 Fusion de noeuds très voisins

De manière similaire à l'utilitaire de fusion de noeud (cf. 7.8), il est possible de demander de fusionner des noeuds proches, ceci durant la phase de lecture. Par contre ici, il n'y a pas de création d'un nouveau maillage, l'opération de fusion est donc réalisée à chaque lecture. La suite du calcul s'effectuera alors sur le maillage dont les noeuds voisins auront été exclus. Les références sont alors mises à jours, en tenant compte des fusions. La table (25) donne un exemple de syntaxe pour supprimer des noeuds voisins d'une distance inférieure à 0.001

On indique à la suite du mot clé `”fusion_noeuds_proches_”` la distance minimale en dessous de laquelle il y a fusion.

TABLE 25 – Exemple de d'utilisation de la méthode de suppression des noeuds voisins d'une distance inférieure à 0.001

```
#-----  
#| definition du (ou des) maillage(s) |  
#-----  
  
# un tube  
< tube_10x4.her  
  
# demande de fusion des noeuds de distances inférieures a 0.001  
fusion_noeuds_proches_ 0.001
```

16 Fusion d'éléments superposés

De manière similaire à l'utilitaire de fusion d'éléments superposés (cf. 7.9), il est possible de demander de fusionner des éléments superposés, ceci durant la phase de lecture. Par contre ici, il n'y a pas de création d'un nouveau maillage, l'opération de fusion est donc réalisée à chaque lecture. La suite du calcul s'effectuera alors sur le maillage dont les éléments superposés auront été exclus. Les références sont alors mises à jour, en tenant compte des fusions.

Pour activer la méthode, il faut indiquer après le maillage le mot clé : `”fusion_elements_superposes_”`. La table (26) donne un exemple de syntaxe pour supprimer des éléments superposés.

TABLE 26 – Exemple de d’utilisation de la méthode de suppression des éléments superposés

```
#-----  
#| definition du (ou des) maillage(s) |  
#-----  
  
# un tube  
< tube_10x4.her  
  
# demande de fusion des éléments superposes  
fusion_elements_superposes_
```

17 Fusion de maillages

De manière similaire avec l’utilitaire de fusion de maillage (7.10), il est possible de réaliser des fusions de maillages pendant la lecture. Par exemple, après la lecture du second maillage, on indique le mot clé : ”fusion_avec_le_maillage_precedent_”, dans ce cas le second maillage est fusionné avec le maillage précédent selon la technique indiquée dans (7.10). Cependant ici, il n’y a pas de nouveau maillage de créé, c’est seulement le second maillage avec ses références, qui est englobé dans le premier. L’opération peut-être répétée plusieurs fois. Enfin, après c’est différentes opérations de fusion, il est toujours possible d’exécuter des opérations d’affinage :

- déplacements solides
- collapsus de noeuds et d’éléments
- renumérotation
- ...

La table (27) donne un exemple assez complexe de successions de fusion et de mouvements solides, puis une fusion des noeuds proches sur le résultat global suivi d’une renumérotation.

18 Références de noeuds, de faces, d’arêtes et d’éléments

Les références de noeuds peuvent être déclarées soient après la définition des noeuds, soit après la définition des éléments. Elles sont constituées chacune d’un nom qui doit commencer par N, suivi d’une liste de numéros de noeuds. Cette liste peut continuée sur plusieurs lignes, en faite la lecture se termine lorque l’on rencontre soit un autre nom de liste ou soit un autre mot clé.

D’une manière systématique, même si l’utilisateur ne l’a pas définit, il y a création d’une référence contenant tous les noeuds. Elle a pour nom : ”N_tout”. Par contre si cette référence est déjà définit, elle n’est pas remplacée.

Les Références d’arêtes, de faces et d’éléments se définissent après la définition des éléments de la manière suivante :

TABLE 27 – Exemple de l'utilisation successive de deux opérations de fusion de maillage, intégrant des déplacements solides pendant et après les opérations.

```
#-----
#| definition du (ou des) maillage(s) |
#-----

# un tube
< tube_10x4.her

#-- un opercule au départ
    nom_maillage disque_depart
< disque.her
def_mouvement_solide_initiaux_
    mouvement_solide_ # def de mouvements solides
    translation_= 0. 0. 0.
    centre_=      0. 0. 0.
    rotation_=    0. 90. 0. en_degre_
    fin_mouvement_solide_

fusion_avec_le_maillage_precedent_

#-- un opercule à la fin
    nom_maillage disque_fin
< disque.her
def_mouvement_solide_initiaux_
    mouvement_solide_ # def de mouvements solides
    centre_=         0. 0. 0.
    rotation_=       0. 90. 0. en_degre_
    translation_=   0.9 0. 0.
    fin_mouvement_solide_

fusion_avec_le_maillage_precedent_
fusion_noeuds_proches_ 0.001
renumerotation_des_noeuds_
```

- Référence d'éléments : fonctionnent comme pour les références de noeuds, un nom qui doit commencer par la lettre E, suivi d'une liste de numéros d'éléments.
- Référence d'arêtes : un nom qui doit commencer par A, suivi d'un ensemble de couple de nombres entiers, qui représente un numéro d'élément suivi d'un numéro local d'arête. Exemple : " A_avant 1 2 4 1 ", ce qui représente l'arête 2 de l'élément 1 et l'arête 1 de l'élément 4.
- Référence de faces : un nom qui doit commencer par F, suivi d'un ensemble de couple de nombres entiers, qui représente un numéro d'élément suivi d'un numéro local de face. Exemple : " F_04 1 4 2 3 ", ce qui représente la face 4 de l'élément 1 et la face 3 de l'élément 2.

D'une manière systématique, même si l'utilisateur ne l'a pas défini, il y a création d'une référence contenant tous les éléments. Elle a pour nom : "E_tout". Par contre si cette référence est déjà défini, elle n'est pas remplacée.

Certaine fois, pour améliorer la clarté des informations, les listes de références sont placés dans un fichier à part.

Remarque : Du fait que l'on indique le nom de maillage, il est possible d'utiliser un même nom de référence pour plusieurs maillage.

18.1 Création automatique de références de frontières

Il est possible de créer automatiquement des références de frontières, il s'agit des noeuds, des éléments, des faces et des arêtes de la frontière. Pour activer cette option, il faut indiquer :

- après le maillage et
- après les mouvements solides s'il y en a et
- après la suppression des noeuds non référencés si cette option est activée
- **le mot clé : "def_auto_ref_frontiere"**, ce mot clé doit être
- "avant" la renumérotation, si on veut utiliser l'option de renumérotation.

Donc, une fois l'option activée, il y a calcul automatique des références de frontières qui ont pour noms :

- "N_tout_front" pour tous les noeuds de la frontière. Sont considérées comme Noeuds frontières, les noeuds qui sont sur la frontière externe.
- "E_front" pour tous les éléments de la frontière. Sont considérées comme éléments frontières, un élément qui contient au moins une face ou une arête frontière.
- "F_front" pour toutes les faces frontières. Sont considérées comme faces frontières, les faces qui n'appartiennent qu'à un seul élément,
- "A_front" pour toutes les arêtes frontières. Sont considérées comme arêtes frontières une arête qui n'appartient pas à plus d'un élément.
- "N_A_front" pour toutes les noeuds uniquement des arêtes frontières. Sont considérées comme arêtes frontières une arête qui n'appartient pas à plus d'un élément.

Ensuite, on peut utiliser ces différentes références.

Remarque : Dans le cas où l'on utilise un niveau de commentaire supérieur à 8 il y a affichage du contenu des nouvelles références calculées.

19 Eléments disponibles

On dispose d'élément :

- 1D : des bielles linéaire, c'est-à-dire à deux noeuds
- 2D : des triangles et des quadrangles linéaires et quadratiques, complets ou incomplets.
- 3D : des hexaèdres c'est-à-dire des briques, des pentaèdres c'est-à-dire la moitié d'une briques, des tétraèdres c'est-à-dire des pyramides à trois cotés ceci en linéaire et en quadratique.

Chaque élément est repéré par un type de géométrie et un type d'interpolation. La liste exhaustive des éléments ainsi définit est donnée par la table(28).

TABLE 28 – liste d'élément finis mécaniques

géométrie	commentaire	interpolation	commentaire	ref
POUT	poutre	BIE1	lineaire 2 noeuds	19.1
TRIANGLE	triangle	LINEAIRE	linéaire 3 noeuds	19.2
TRIANGLE	triangle	QUADRACOMPL	quadratique 6 noeuds	19.3
TRIANGLE	triangle	CUBIQUE	quadratique 10 noeuds	19.4
TRIA_AXI	triangle	LINEAIRE	axisymétrique linéaire 3 noeuds	19.5
TRIA_AXI	triangle	QUADRACOMPL	axisymétrique quadratique 6 noeuds	19.6
TRIANGLE	triangle	SFE1	élément coques à 6 noeuds	19.7
TRIANGLE	triangle	SFE2	élément coques à 6 noeuds	19.8
TRIANGLE	triangle	SFE3	élément coques à 6 noeuds	19.9
QUADRANGLE	quadrangle	LINEAIRE	linéaire 4 noeuds	19.11
QUADRANGLE	quadrangle	QUADRATIQUE	quadratique incomplet 8 noeuds	19.12
QUADRANGLE	quadrangle	QUADRACOMPL	quadratique complet 9 noeuds	19.13
QUADRANGLE	quadrangle	CUBIQUE	cubique complet 16 noeuds	19.14
QUAD_AXI	quadrangle	LINEAIRE	axisymétrique linéaire 4 noeuds	19.15
QUAD_AXI	quadrangle	QUADRATIQUE	axisymétrique quadratique incomplet 8 noeuds	19.16
QUAD_AXI	quadrangle	QUADRACOMPL	axisymétrique quadratique complet 9 noeuds	19.17
HEXAEDRE	brique, 6 faces	LINEAIRE	linéaire 8 noeuds	19.19
HEXAEDRE	brique, 6 faces	QUADRATIQUE	quadratique incomplet 20 noeuds	19.20
HEXAEDRE	brique, 6 faces	QUADRACOMPL	quadratique complet 27 noeuds	19.21
PENTAEDRE	1/2 brique 5 faces	LINEAIRE	6 noeuds	19.22
PENTAEDRE	1/2 brique 5 faces	QUADRATIQUE	incomplets 15 noeuds	19.23
PENTAEDRE	1/2 brique 5 faces	QUADRACOMPL	complets 18 noeuds	19.24
TETRAEDRE	tétraèdre à 4 faces	LINEAIRE	4 noeuds	19.25
TETRAEDRE	tétraèdre à 4 faces	QUADRACOMPL	10 noeuds	19.26

L'enregistrement classique d'un élément sera : un numéro d'élément suivi d'un identificateur de géométrie suivi d'un identificateur d'interpolation puis une liste de numéros de noeuds donnée dans l'ordre de la numérotation locale de l'élément de référence.

Il est également possible pour certains éléments de définir un nombre de points d'intégration différents.

19.1 POUT BIE1 : élément barre

L'élément s'appuie sur un élément linéaire 1D de référence (cf. 62).

Le nombre de point d'intégration par défaut est 1 (cf.19.28), et le nombre de noeuds est 2.

L'élément réagit comme une billette mécanique. Le champ de contrainte-déformation est de traction-compression, aucun effet de flexion et/ou de torsion n'est ici pris en compte.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 1. Il est donc nécessaire d'utiliser une loi de comportement associé 1D.

Il est également nécessaire de déclarer la section de l'élément, ce qui s'effectue après la définition du maillage, par référence à des groupes d'éléments de même section (cf. 31.2).

Lorsque la biellette est sollicitée en traction - compression, la section varie en fonction de la loi de comportement. L'équilibre mécanique s'effectue en tenant compte de la section déformée (cf. 24).

L'élément peut être utilisé dans un espace de travail à 1, 2 ou 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (29) donne un exemple de syntaxe pour définir un élément biellette.

TABLE 29 – Exemple de déclaration d'un élément biellette

```

elements -----
  1 ELEMENTS
#-----
#| NO | | |
#|ELTS | type element | Noeuds |
#-----
      1      POUT BIE1          1      2

```

19.2 TRIANGLE LINEAIRE

L'élément s'appuie sur un élément de référence 2D linéaire (cf. 63).

Le nombre de point d'intégration par défaut est 1 (cf.19.30), et le nombre de noeuds est 3.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci de manière uniforme dans l'épaisseur. Ainsi les effets de flexion ne sont pas pris en compte. C'est donc un élément de membrane ayant une épaisseur non nulle.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 2. Il est donc nécessaire d'utiliser une loi de comportement associé 2D : type contraintes planes ou type déformations planes.

Il est également nécessaire de déclarer l'épaisseur de l'élément, ce qui s'effectue après la définition du maillage, par référence à des groupes d'éléments de même section (cf. 31.1). L'épaisseur peut varier pendant le calcul (voir 23 pour plus d'information sur la méthode de calcul).

L'élément peut être utilisé dans un espace de travail à 2 ou 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (30) donne un exemple de syntaxe pour définir un élément triangle.

19.3 TRIANGLE quadratique (QUADRACOMPL)

L'élément s'appuie sur un élément de référence 2D quadratique (6 noeuds) (cf. 63).

Le nombre de point d'intégration par défaut est 3 (cf.19.30), sur les arêtes de 2 (cf.19.28), et le nombre de noeuds est 6.

TABLE 30 – Exemple de déclaration d’un élément triangle avec interpolation linéaire

```

elements -----
  1 ELEMENTS
#-----
#| NO  |          |          |
#|ELTS |  type element  |          |
#-----
      1      TRIANGLE LINEAIRE      1      2      3

```

L’élément réagit aux sollicitations de traction, compression cisaillement, ceci de manière uniforme dans l’épaisseur. Ainsi les effets de flexion ne sont pas pris en compte. C’est donc un élément de membrane ayant une épaisseur non nulle.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 2. Il est donc nécessaire d’utiliser une loi de comportement associé 2D : type contraintes planes ou type déformations planes. L’épaisseur peut varier pendant le calcul (voir 23 pour plus d’information sur la méthode de calcul).

Il est également nécessaire de déclarer l’épaisseur de l’élément, ce qui s’effectue après la définition du maillage, par référence à des groupes d’éléments de même section (cf. 31.1).

L’élément peut être utilisé dans un espace de travail à 2 ou 3 dimensions, en association avec n’importe quel autre type d’élément.

La table (31) donne un exemple de syntaxe pour définir un élément triangle quadratique.

TABLE 31 – Exemple de déclaration d’un élément triangle avec interpolation quadratique

```

elements -----
  1 ELEMENTS
#-----
#| NO  |          |          |
#|ELTS |  type element  |          |
#-----
      1      TRIANGLE QUADRACOMPL      1  2  3  4  5  6

```

Par défaut les 3 points d’intégrations sont situées sur les arrêtes. Il est possible d’utiliser un autre jeux de points d’intégrations, situés tous strictement à l’intérieur de l’élément (cf.19.30). La table (32) donne un exemple de syntaxe pour définir ce type d’élément.

Remarquons que dans le cas de 1 points d’intégration, ce nombre n’est pas suffisant pour obtenir une matrice locale de singularité 3 en 2D (les 3 mouvements solides, en 3D s’ajoute les mouvements suivants la 3ième direction). On peut néanmoins choisir un seul point d’intégration, dans ce cas l’élément est sous-intégré et on risque l’apparition de mode d’hourglass par exemple en dynamique, une solution pour y remédier est bloquer les modes d’hourglass, ceci est possible avec l’élément à 1 point d’intégration. On se reportera à 22 pour plus d’information.

TABLE 32 – Exemple de déclaration d’un élément triangle avec interpolation quadratique, et avec 3 points d’intégrations strictement interne à l’élément

```

elements -----
  2 ELEMENTS
#-----
#| NO  |          |          |
#|ELTS |  type element |          Noeuds |
#-----
  1 TRIANGLE QUADRACOMPL _cm3pti  1 2 3 4 5 6
  2 TRIANGLE QUADRACOMPL _cm1pti  1 2 3 4 5 6

```

19.4 TRIANGLE CUBIQUE

L’élément s’appuie sur un élément de référence 2D cubique (10 noeuds) (cf. 63).

Le nombre de point d’intégration par défaut est 6 (cf.19.30), sur les arrêtes de 3 (cf.19.28), et le nombre de noeuds est 10.

L’élément réagit aux sollicitations de traction, compression cisaillement, ceci de manière uniforme dans l’épaisseur. Ainsi les effets de flexion ne sont pas pris en compte. C’est donc un élément de membrane ayant une épaisseur non nulle.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 2. Il est donc nécessaire d’utiliser une loi de comportement associé 2D : type contraintes planes ou type déformations planes. L’épaisseur peut varier pendant le calcul (voir 23 pour plus d’information sur la méthode de calcul).

Il est également nécessaire de déclarer l’épaisseur de l’élément, ce qui s’effectue après la définition du maillage, par référence à des groupes d’éléments de même section (cf. 31.1).

L’élément peut être utilisé dans un espace de travail à 2 ou 3 dimensions, en association avec n’importe quel autre type d’élément.

La table (33) donne un exemple de syntaxe pour définir un élément triangle quadratique.

TABLE 33 – Exemple de déclaration d’un élément triangle avec interpolation quadratique et plusieurs type d’intégration

```

elements -----
  2 ELEMENTS
#-----
#| NO  |          |          |
#|ELTS |  type element |          Noeuds |
#-----
  1 TRIANGLE CUBIQUE  1 2 3 4 5 6 7 8 9 10
  2 TRIANGLE CUBIQUE _cm4pti  1 2 3 4 5 6 7 8 9 10

```

Il est possible d’utiliser une sous intégration avec 4 points d’intégration (cf. exemple de déclaration dans la table 33). Remarquons que dans ce cas on risque l’apparition de modes

d'hourglass , une solution pour y remédier est de bloquer ces modes ce qui est possible avec l'élément à 4 points d'intégration. On se reportera à 22 pour plus d'information.

19.5 Triangle axisymétrique linéaire : TRIA_AXI LINEAIRE

L'élément s'appuie sur un élément de référence 2D linéaire (cf. 63).

Le nombre de point d'intégration par défaut est 1 (cf.19.30), et le nombre de noeuds est 3.

L'élément réagit aux sollicitations axisymétriques de traction, compression cisaillement. La discrétisation élément finis est de dimension 2, mais l'élément est relatif à un volume engendré par la rotation de l'élément réel autour de l'axe de rotation qui est y.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D : type contraintes volumique ou type déformations volumique.

L'espace de travail est de dimension 3.

La table (34) donne un exemple de syntaxe pour définir un élément triangle axisymétrique linéaire.

TABLE 34 – Exemple de déclaration de deux éléments triangles axisymétriques avec interpolation linéaire

```

elements -----
  2 ELEMENTS
#-----
#| NO  |          |          |
#|ELTS |  type element  |          |
#-----
      1      TRIA_AXI LINEAIRE      1      3      2
      2      TRIA_AXI LINEAIRE      3      4      2

```

Concernant les efforts externes, du au fait que l'élément est analytique suivant l'axe de rotation et discrétisé suivant les deux autres axes, on se reportera au §(33.2.9) et §(34.8) pour la description des particularités du chargement.

En résumé pour les différents nombres on a :

```

nombre.nbne = 3; // le nombre de noeud de l'élément
nombre.nbneA = 2; // le nombre de noeud des aretes
nombre.nbi = 1; // le nombre de point d'intégration
// pour le calcul mécanique
nombre.nbiEr = 3; // le nombre de point d'intégration
// pour le calcul d'erreur
nombre.nbiS = 1; // le nombre de point d'intégration pour
// le calcul de second membre surfacique
nombre.nbiA = 1; // le nombre de point d'intégration pour
// le calcul de second membre linéique
nombre.nbiMas = 3; // le nombre de point d'intégration pour

```

```
// le calcul de la matrice masse
```

19.6 Triangle axisymétrique quadratique : TRIA_AXI QUADRACOMPL

L'élément s'appuie sur un élément de référence 2D quadratique (6 noeuds) (cf. 63).

Le nombre de point d'intégration par défaut est 3 (cf.19.30), et le nombre de noeuds est 6.

L'élément réagit aux sollicitations axisymétriques de traction, compression cisaillement. La discrétisation élément finis est de dimension 2, mais l'élément est relatif à un volume engendré par la rotation de l'élément réel autour de l'axe de rotation qui est y.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D : type contraintes volumique ou type déformations volumique.

L'espace de travail est de dimension 3.

La table (35) donne un exemple de syntaxe pour définir un élément triangle axisymétrique linéaire.

TABLE 35 – Exemple de déclaration d'un élément triangle axisymétriques avec interpolation quadratique et différents nombres de points d'intégration

```
elements -----
  2 ELEMENTS
#-----
#| NO | | | | | | | | | |
#|ELTS | type element | Noeuds |
#-----
  1 TRIA_AXI QUADRACOMPL 1 2 3 4 5 6
  2 TRIA_AXI QUADRACOMPL _cm1pti 1 2 3 4 5 6
```

Concernant les efforts externes, du au fait que l'élément est analytique suivant l'axe de rotation et discrétisé suivant les deux autres axes, on se reportera au §(33.2.9) et §(34.8) pour la description des particularités du chargement.

En résumé pour les différents nombres on a :

```
nombre.nbne = 6; // le nombre de noeud de l'élément
nombre.nbneA = 3; // le nombre de noeud des aretes
nombre.nbi = 3; // le nombre de point d'intégration
// pour le calcul mécanique
nombre.nbiEr = 6; // le nombre de point d'intégration
// pour le calcul d'erreur
nombre.nbiS = 3; // le nombre de point d'intégration
// pour le calcul de second membre surfacique
nombre.nbiA = 2; // le nombre de point d'intégration pour
// le calcul de second membre linéique
```

```

nombre.nbiMas = 6; // le nombre de point d'intégration
                //pour le calcul de la matrice masse

```

Il est possible d'utiliser une sous intégration avec 1 points d'intégration (cf. exemple de déclaration dans la table 35). Remarquons que dans ce cas on risque l'apparition de modes d'hourglass , une solution pour y remédier est de bloquer ces modes ce qui est possible avec l'élément à 1 points d'intégration. On se reportera à 22 pour plus d'information.

19.7 SFE1 : élément coque

Introduction L'élément SFE1 est un élément SFE (Semi Finite Element) coque qui a la particularité de ne pas utiliser de degré de liberté de rotation, mais uniquement les degrés de libertés classiques de translation. La cinématique locale s'appuie sur les hypothèses de Kirchhoff en transformations finies.

$$\vec{OM} = \vec{OP}(\theta^1, \theta^2) + z.\vec{N}(\theta^1, \theta^2) \quad (18)$$

Où M est un point courant dans l'épaisseur de la coque, P est le point correspondant de la surface médiane de référence, θ^α représentent les 2 coordonnées permettant de décrire la surface médiane, z est la coordonnée d'épaisseur qui est supposée évoluer entre $-h/2$ et $h/2$ avec h l'épaisseur de la coque, et \vec{N} est le vecteur normal à la surface.

A partir de cette cinématique on obtient le tenseur métrique sous la forme suivante :

$$\begin{aligned} g_{\alpha\beta} &= a_{\alpha\beta} - 2z.b_{\alpha\beta} + z^2 b_{\alpha\gamma}b_{\beta}^\gamma \\ g_{\alpha 0} &= 0 \\ g_{33} &= 1 \end{aligned} \quad (19)$$

Où $g_{\alpha\beta}$ représentent les composantes ($\alpha, \beta, \gamma = 1, 2$) de la métrique au point M qui varient, $a_{\alpha\beta}$ représentent les composantes de la métrique de la surface médiane au point P , $b_{\alpha\beta}$ les composantes du tenseur de courbure dans le repère naturel. On remarque que la métrique comporte un terme linéaire en z et un terme quadratique. Pour un faible courbure et une faible épaisseur, le terme quadratique peut-être négligé. Dans le cas des éléments Sfe1, la métrique complète est prise en compte.

A l'aide de cette métrique on obtient naturellement un tenseur vitesse de déformation quadratique en z . Pour la déformation, par exemple dans le cas d'une mesure de déformation d'Almansi on obtient les composantes :

$$\varepsilon_{\alpha\beta} = 0.5(g_{\alpha\beta}(t) - g_{\alpha\beta}(0)) \quad (20)$$

qui comportent donc également des termes linéaires et quadratiques en z .

Le calcul du tenseur courbure est un élément clé des éléments SFE. Un premier modèle a été présenté par G. Rio [], puis amélioré par G.Rio et B. Thati [], et également par la suite par H. Laurent []. Actuellement, S. Couedo [] a effectué une partie de ses travaux de thèse sur la recherche d'une mesure de courbure optimum en fonction de la position des noeuds.

Cette première implantation s'appuie sur les développements originaux de l'élément. De prochaines implantations sont prévu pour intégrer les différentes versions proposées comme mesure de la courbure. L'idée est ensuite de pouvoir les valider et les comparer.

L'idée est d'utiliser la position des éléments mitoyens à l'élément central, pour reconstruire une courbure non nulle.

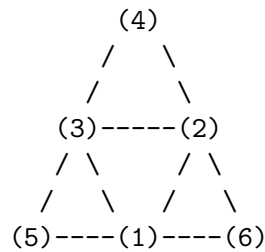
Les conditions limites de type "encastrement" ou "symétrie" sont particulières pour ces éléments. On se reportera à 34.7 pour plus d'informations.

Mode d'emploi de l'élément L'élément s'appuie sur un élément de référence 2D triangulaire linéaire (3 noeuds) (cf. 63).

Le nombre de point d'intégration par défaut est 2, 1 en surface (cf.19.30) et 2 en épaisseur (cf.19.28).

Cependant le nombre de noeuds de l'élément est 6. Les 3 premiers noeuds sont relatif au triangle central. Les 3 derniers noeuds correspondent aux noeuds externes (cf.36). On voit que le noeud 4 est opposé au noeud 1 et ainsi de suite. Dans le cas où le noeud 4 n'existe pas, dans la table de connection, on répète le noeud 1 au lieu du noeud 4. Par exemple supposons que les numéros indiqués dans la table (cf.36) sont les numéros des noeuds, et que le noeud 4 n'existe pas on aurait la connection : 1 2 3 1 5 6

TABLE 36 – numérotation des éléments SFE1



concernant les aretes: elles sont lineaire
numérotation des arrêtes :

1 : 1 2, 2 : 2 3, 3 : 3 1

concernant la triangulation linéaire de l'élément : l'élément est décomposé en lui même

Le comportement de membrane est celui de l'élément central, donc linéaire en espace.

L'élément réagit aux sollicitations de membrane de traction, compression cisaillement. L'élément réagit aux sollicitations de flexion.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 2. Il est donc nécessaire d'utiliser une loi de comportement associé 2D.

L'espace de travail est de dimension 3.

La table (37) donne un exemple de syntaxe pour définir un élément sfe1.

Il est également nécessaire de déclarer l'épaisseur de l'élément, ce qui s'effectue après la définition du maillage, par référence à des groupes d'éléments de même section (cf. 31.1).

En résumé pour les différents nombres on a :

```
nbnce = 3; // le nombre de noeuds de l'element central
nbnte = 6; // le nombre total de noeuds
```

TABLE 37 – Exemple de déclaration d’éléments SFE1

```

elements -----
  32 ELEMENTS      # definition du nombre d'elements
#-----
#| NO  |                |
#|ELTS |      type element |          Noeuds |
#-----

      1 TRIANGLE SFE1   1 6 2 7 6 2
      2 TRIANGLE SFE1   6 7 2 3 1 11
      3 TRIANGLE SFE1   2 7 3 8 7 6
.....

nbneA = 2; // le nombre de noeuds des aretes de l'élément central
nbis   = 1; // le nombre de points d'intégration de surface pour le calcul mécanique
nbie   = 2; // le nombre de points d'integ d'épaisseur pour le calcul mécanique
nbisEr = 3; // le nombre de points d'intégration de surface pour le calcul d'erreur
nbieEr = 2; // le nombre de points d'intégration d'épaisseur pour le calcul d'erreur
nbiSur = 3; // le nombre de points d'intégration pour le calcul de second membre surfacique
nbiA   = 1; // le nombre de points d'intégration pour le calcul de second membre linéique
nbisMas = 3; // le nombre de points d'intégration de surface pour le calcul de la matrice m
nbieMas = 2; // le nombre de points d'intégration d'épaisseur pour le calcul de la matrice m

```

Construction d’un maillage SFE Pour définir un maillage SFE, la démarche peut-être la suivante :

- Construire un maillage triangulaire linéaire, par exemple toto.her
- Utiliser l’utilitaire d’Herezh++ (cf.7.7) permettant de transformer un maillage triangulaire linéaire en maillage sfe.

19.8 SFE2 : élément coque

Introduction L’élément SFE2 est un élément SFE (Semi Finite Element) coque qui a la particularité de ne pas utiliser de degré de liberté de rotation, mais uniquement les degrés de libertés classiques de translation. La cinématique locale s’appuie sur les hypothèses de Kirchhoff en transformations finies. On se reportera à l’introduction de l’élément SFE1, pour les généralités sur les éléments SFE

Dans le cas de l’élément SFE2, la courbure est calculée à partir de l’interpolation des normales calculées sur chaque arrête, mais à la différence de l’élément SFE1, on cherche ici à tenir compte de l’influence de la non régularité du maillage. Il s’agit du modèle développé par H. Laurent et G. Rio. Le modèle est présenté dans la thèse de Hervé Laurent [] et dans []. L’implantation est ici légèrement différente, mais globalement elle devrait conduire aux mêmes résultats. En particulier on ne se sert pas de la position des centres de gravité pour calculer les normales sur les arrêtes.

La normale $\vec{N}^{(i)}$ sur chaque arête “i” est obtenue par une moyenne pondérée :

$$\vec{N}^{(i)} = \frac{(\vec{N} H_a A + \vec{N}_e DH_d)}{(H_a A + DH_d)} \quad (21)$$

avec \vec{N} la normale de la facette centrale, $H_a A$ la longueur de la hauteur sur la facette centrale, abaissée sur l’arête, \vec{N}_e la normale à la facette extérieure, DH_d la longueur de la hauteur sur la facette externe.

La position de la normale sur l’arête est également optimisée de manière à satisfaire la relation :

$$G\vec{H}_i \cdot \vec{C}B = H_i \vec{G}_i \cdot \vec{C}B \quad (22)$$

où “G” est le centre de gravité du triangle interne, G_i le centre de gravité du triangle externe, $\vec{C}B$ est l’arête, et H_i est la position cherchée de la normale.

Mode d’emploi de l’élément Le mode d’emploi est identique au cas de l’élément SFE1, la seule différence est qu’il faut mettre l’identificateur SFE2 à la place de SFE1, dans le fichier maillage. On se référera donc au mode d’emploi de l’élément SFE1 pour plus de détails.

La table (38) donne un exemple de syntaxe pour définir un élément sfe2.

TABLE 38 – Exemple de déclaration d’éléments SFE2

```
elements -----
  32 ELEMENTS      # definition du nombre d'elements
#-----
#| NO |                |
#|ELTS |      type element |                Noeuds |
#-----
      1 TRIANGLE SFE2   1 6 2 7 6 2
      2 TRIANGLE SFE2   6 7 2 3 1 11
      3 TRIANGLE SFE2   2 7 3 8 7 6
.....
```

Actuellement les différents nombres sont identique au cas sfe1.

Construction d’un maillage SFE Identique au cas SFE1 (19.7).

19.9 SFE3 et SFE3C : élément coque

Introduction L’élément SFE3 est un élément SFE (Semi Finite Element appelé également RF pour “Rotation Free”) coque qui a la particularité de ne pas utiliser de degré de liberté de rotation, mais uniquement les degrés de libertés classiques de translation. La cinématique locale s’appuie sur les hypothèses de Kirchhoff en transformations finies. On se reportera à l’introduction de l’élément SFE1, pour les généralités sur les éléments SFE

Dans le cas de l'élément SFE3, la courbure est calculée à partir d'une fonction qui donne la cote locale des points externes par rapport à la facette centrale. Ici cette fonction est un polynôme d'ordre 2 en θ^α

$$\theta^3(\theta^1, \theta^2) = a \theta^1 (\theta^1 - 1.) + b \theta^2 (\theta^2 - 1.) + c \theta^1 \theta^2 \quad (23)$$

θ^3 représentant la cote selon la normale à la facette centrale, a, b, c étant les coefficients de la fonction polynomiale. Ensuite la courbure est calculée à partir de ce polynôme sachant que tout point "M" a pour coordonnées dans le repère locale $\langle \theta^1, \theta^2, \theta^3 \rangle$. On a :

$$b_{11} = 2 a , \quad b_{22} = 2 b , \quad b_{12} = c \quad (24)$$

Dans le cas où le noeud extérieur n'existe pas, on considère que la courbure dans la direction de se noeud est nulle.

La différence entre les éléments SFE3 et SFE3C est relative au calcul du jacobien. Dans le premier cas, c'est le jacobien de la facette plane qui est pris en compte alors que dans le second cas c'est le jacobien de la facette courbe.

Mode d'emploi de l'élément Le mode d'emploi est identique au cas de l'élément SFE1, la seule différence est qu'il faut mettre l'identificateur SFE3 à la place de SFE1, dans le fichier maillage. On se référera donc au mode d'emploi de l'élément SFE1 pour plus de détails.

La table (39) donne un exemple de syntaxe pour définir des éléments sfe3 et sfe3C.

TABLE 39 – Exemple de déclaration d'éléments SFE3 (pour l'élément SFE3C, il suffit de changer SFE3 par SFE3C)

```

elements -----
  32 ELEMENTS      # definition du nombre d'elements
#-----
#| NO  |                |
#|ELTS |      type element |          Noeuds |
#-----
      1 TRIANGLE SFE3   1 6 2 7 6 2
      2 TRIANGLE SFE3   6 7 2 3 1 11
      3 TRIANGLE SFE3   2 7 3 8 7 6
.....

```

Actuellement les différents nombres sont identique au cas sfe1.

Construction d'un maillage SFE3 Identique au cas SFE1 (19.7).

19.10 SFE3 _cmjpti

Il s'agit des mêmes éléments que les SFE3 classiques, avec comme différences uniquement le nombre de points d'intégration. Deux grandes familles sont proposées :

1. "SFE3 _cmjpti" : "j" pair, l'intégration s'effectue selon "j" points d'intégration de Gauss dans l'épaisseur et 1 point d'intégration dans la surface. La position de chaque point de Gauss est telle qu'aucun point ne se situe exactement sur la couche supérieure ou inférieure, tous les points sont à l'intérieur de la matière. Le premier point et le dernier, sont les plus proches des peaux inf et sup respectivement.
2. "SFE3 _cmjpti" : "j" impair, l'intégration s'effectue selon "j" points d'intégration de Gauss-Lobatto dans l'épaisseur et 1 point d'intégration dans la surface. Ce système de points d'intégration est tel que le premier point se situe sur la peau inférieure de la coque et le dernier point sur la peau supérieure. Comme le nombre de points d'intégration est impair, le point de numéro : $j/2+1$ (division entière) , est situé au niveau de la couche médiane du matériau.

Actuellement, les cas suivants sont disponibles : **j= 3, 4, 5, 6, 7, 12, 13** . Le cas "j=2" correspond à l'élément par défaut, c'est-à-dire sans l'indication "_cmjpti"

Pour utiliser ces éléments il suffit d'indiquer à la suite du mot clé "SFE3" la précision "_cm4pti" (par exemple pour 4 points de Gauss) (avec un espace entre SFE3 et la précision).

19.11 Quadrangle linéaire : QUADRANGLE LINEAIRE

L'élément s'appuie sur un élément de référence 2D linéaire (cf. 64).

Le nombre de point d'intégration par défaut est 4 (cf.20.1), et le nombre de noeuds est 4.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci de manière uniforme dans l'épaisseur. Ainsi les effets de flexion ne sont pas pris en compte. C'est donc un élément de membrane ayant une épaisseur non nulle.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 2. Il est donc nécessaire d'utiliser une loi de comportement associé 2D : type contraintes planes ou type déformations planes.

Il est également nécessaire de déclarer l'épaisseur de l'élément, ce qui s'effectue après la définition du maillage, par référence à des groupes d'éléments de même section (cf. 31.1).

L'élément peut être utilisé dans un espace de travail à 2 ou 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (40) donne un exemple de syntaxe pour définir un élément quadrangle linéaire. En résumé pour les différents nombres on a :

```
nombre.nbne = 4; // le nombre de noeud de l'élément
nombre.nbneA = 2; // le nombre de noeud des aretes
nombre.nbi = 4; // le nombre de point d'intégration
// pour le calcul mécanique
nombre.nbiEr = 4; // le nombre de point d'intégration
// pour le calcul d'erreur
nombre.nbiS = 4; // le nombre de point d'intégration pour
// le calcul de second membre surfacique
```


TABLE 40 – Exemple de déclaration de deux éléments quadrangles avec interpolation linéaire et plusieurs types de nombres de points d’intégration

```

elements -----
  2 ELEMENTS
#-----
#| NO  |          |          |
#|ELTS |  type element  |          |
#-----
      1   QUADRANGLE LINEAIRE      1   3   2   6
      2   QUADRANGLE LINEAIRE _cm1pti  3   4   2   5

nombre.nbiA = 1; // le nombre de point d'intégration pour
                // le calcul de second membre linéique
nombre.nbiMas = 4; // le nombre de point d'intégration
                // pour le calcul de la matrice masse

```

Il est possible d'utiliser une sous intégration avec 1 points d'intégration (cf. exemple de déclaration dans la table 40). Remarquons que dans ce cas on risque l'apparition de modes d'hourglass , une solution pour y remédier est de bloquer ces modes ce qui est possible avec l'élément à 1 points d'intégration. On se reportera à 22 pour plus d'information.

19.12 Quadrangle quadratique incomplet : QUADRANGLE QUADRATIQUE

L'élément s'appuie sur un élément de référence 2D quadratique (8 noeuds) (cf. 64).

Le nombre de point d'intégration par défaut est 4 (cf.20.1), et le nombre de noeuds est 8.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci de manière uniforme dans l'épaisseur. Ainsi les effets de flexion ne sont pas pris en compte. C'est donc un élément de membrane ayant une épaisseur non nulle.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 2. Il est donc nécessaire d'utiliser une loi de comportement associé 2D : type contraintes planes ou type déformations planes.

Il est également nécessaire de déclarer l'épaisseur de l'élément, ce qui s'effectue après la définition du maillage, par référence à des groupes d'éléments de même section (cf. 31.1).

L'élément peut être utilisé dans un espace de travail à 2 ou 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (41) donne un exemple de syntaxe pour définir un élément quadrangle quadratique incomplet.

En résumé pour les différents nombres on a :

```

nombre.nbne = 8; // le nombre de noeud de l'élément
nombre.nbneA = 3; // le nombre de noeud des aretes
nombre.nbi = 4; // le nombre de point d'intégration

```

TABLE 41 – Exemple de déclaration d’un élément quadrangle avec interpolation quadratique incomplète

```

elements -----
  1 ELEMENTS
#-----
#| NO  |          |          |
#|ELTS|  type element  |          Noeuds          |
#-----
      1      QUADRANGLE QUADRATIQUE      1 2 3 4 5 6 7 8

// pour le calcul mécanique
nombre.nbiEr = 9; // le nombre de point d'intégration
// pour le calcul d'erreur
nombre.nbiS = 4; // le nombre de point d'intégration
// pour le calcul de second membre surfacique
nombre.nbiA = 2; // le nombre de point d'intégration
// pour le calcul de second membre linéique
nombre.nbiMas = 9; // le nombre de point d'intégration
// pour le calcul de la matrice masse

```

Remarquons que dans le cas de 4 points d’intégration, ce nombre n’est pas tout à fait suffisant pour obtenir une matrice locale de singularité 3 en 2D (les 3 mouvements solides, en 3D s’ajoute les mouvements suivants la 3 ième direction). L’élément est donc légèrement sous-intégré et il pourrait théoriquement apparaître des modes d’hourglass. Une solution pour y remédier est de les bloquer ce qui est possible avec cet élément On se reportera à [22](#) pour plus d’information.

19.13 Quadrangle quadratique complet : QUADRANGLE QUADRACOMP

L’élément s’appuie sur un élément de référence 2D quadratique (9 noeuds) (cf. [64](#)).

Le nombre de point d’intégration par défaut est 9 (cf.[20.1](#)), et le nombre de noeuds est 9.

L’élément réagit aux sollicitations de traction, compression cisaillement, ceci de manière uniforme dans l’épaisseur. Ainsi les effets de flexion ne sont pas pris en compte. C’est donc un élément de membrane ayant une épaisseur non nulle.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 2. Il est donc nécessaire d’utiliser une loi de comportement associé 2D : type contraintes planes ou type déformations planes.

Il est également nécessaire de déclarer l’épaisseur de l’élément, ce qui s’effectue après la définition du maillage, par référence à des groupes d’éléments de même section (cf. [31.1](#)).

L’élément peut être utilisé dans un espace de travail à 2 ou 3 dimensions, en association avec n’importe quel autre type d’élément.

La table (42) donne un exemple de syntaxe pour définir un élément quadrangle quadratique complet.

TABLE 42 – Exemple de déclaration d’un élément quadrangle avec interpolation quadratique complete et 2 types de nombres de points d’intégration

```

elements -----
  2 ELEMENTS
#-----
#| NO | | | | | | | | | |
#|ELTS | type element | Noeuds |
#-----
  1 QUADRANGLE QUADRACOMPL 1 2 3 4 5 6 7 8 9
  2 QUADRANGLE QUADRACOMPL _cm4pti 1 2 3 4 5 6 7 8 9

```

En résumé pour les différents nombres on a :

```

nombre.nbne = 9; // le nombre de noeud de l'élément
nombre.nbneA = 3; // le nombre de noeud des aretes
nombre.nbi = 9; // le nombre de point d'intégration
// pour le calcul mécanique
nombre.nbiEr = 9; // le nombre de point d'intégration
// pour le calcul d'erreur
nombre.nbiS = 4; // le nombre de point d'intégration pour
// le calcul de second membre surfacique
nombre.nbiA = 2; // le nombre de point d'intégration pour
// le calcul de second membre linéique
nombre.nbiMas = 9; // le nombre de point d'intégration pour
// le calcul de la matrice masse

```

Il est possible d'utiliser une sous intégration avec 4 points d'intégration (cf. exemple de déclaration dans la table 42). Remarquons que dans ce cas on risque l'apparition de modes d'hourglass , une solution pour y remédier est de bloquer ces modes ce qui est possible avec l'élément à 4 points d'intégration. On se reportera à 22 pour plus d'information.

19.14 Quadrangle cubique complet : QUADRANGLE CUBIQUE

L'élément s'appuie sur un élément de référence 2D quadratique (16 noeuds) (cf. 64).

Le nombre de point d'intégration par défaut est 16 (cf.20.1), et le nombre de noeuds est 16.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci de manière uniforme dans l'épaisseur. Ainsi les effets de flexion ne sont pas pris en compte. C'est donc un élément de membrane ayant une épaisseur non nulle.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 2. Il est donc nécessaire d'utiliser une loi de comportement associé 2D : type contraintes planes ou type déformations planes.

Il est également nécessaire de déclarer l'épaisseur de l'élément, ce qui s'effectue après la définition du maillage, par référence à des groupes d'éléments de même section (cf. 31.1).

L'élément peut être utilisé dans un espace de travail à 2 ou 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (43) donne un exemple de syntaxe pour définir un élément quadrangle cubique.

TABLE 43 – Exemple de déclaration d'un élément quadrangle avec interpolation quadratique complete et plusieurs types de nombres de points d'intégration

```

elements -----
      2 ELEMENTS
#-----
#| NO | | |
#|ELTS | type element | Noeuds |
#-----
      1 QUADRANGLE CUBIQUE 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
      2 QUADRANGLE CUBIQUE _cm9pti 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

```

En résumé pour les différents nombres on a :

```

nombre.nbne = 16; // le nombre de noeud de l'élément
nombre.nbneA = 4; // le nombre de noeud des aretes
nombre.nbi = 16; // le nombre de point d'intégration
// pour le calcul mécanique
nombre.nbiEr = 16; // le nombre de point d'intégration
// pour le calcul d'erreur
nombre.nbiS = 9; // le nombre de point d'intégration pour
// le calcul de second membre surfacique
nombre.nbiA = 3; // le nombre de point d'intégration pour
// le calcul de second membre linéique
nombre.nbiMas = 16; // le nombre de point d'intégration pour
// le calcul de la matrice masse

```

Il est possible d'utiliser une sous intégration avec 9 points d'intégration (cf. exemple de déclaration dans la table 43). Remarquons que dans ce cas on risque l'apparition de modes d'hourglass , une solution pour y remédier est de bloquer ces modes ce qui est possible avec l'élément à 9 points d'intégration. On se reportera à 22 pour plus d'information.

19.15 Quadrangle axisymétrique linéaire : QUAD_AXI LINEAIRE

L'élément s'appuie sur un élément de référence 2D linéaire (cf. 64).

Le nombre de point d'intégration par défaut est 4 (cf.20.1), et le nombre de noeuds est 4.

L'élément réagit aux sollicitations axisymétriques de traction, compression cisaillement. La discrétisation élément finis est de dimension 2, mais l'élément est relatif à un volume engendré par la rotation de l'élément réel autour de l'axe de rotation qui est y.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D : type contraintes volumique ou type déformations volumique.

L'espace de travail est de dimension 3.

La table (44) donne un exemple de syntaxe pour définir un élément quadrangle axisymétrique linéaire.

TABLE 44 – Exemple de déclaration de deux éléments quadrangles axisymétriques avec interpolation linéaire et plusieurs types de nombres de points d'intégration

```

elements -----
  2 ELEMENTS
#-----
#| NO  |          |          |          |
#|ELTS |  type element |          |          |
#-----
      1    QUAD_AXI LINEAIRE      1    3    2    6
      2    QUAD_AXI LINEAIRE _cm1pti    3    4    2    5

```

Concernant les efforts externes, du au fait que l'élément est analytique suivant l'axe de rotation et discrétisé suivant les deux autres axes, on se reportera au §(33.2.9) et §(34.8) pour la description des particularités du chargement.

En résumé pour les différents nombres on a :

```

nombre.nbne = 4; // le nombre de noeud de l'élément
nombre.nbneA = 2; // le nombre de noeud des aretes
nombre.nbi = 4; // le nombre de point d'intégration
// pour le calcul mécanique
nombre.nbiEr = 4; // le nombre de point d'intégration
// pour le calcul d'erreur
nombre.nbiS = 4; // le nombre de point d'intégration
// pour le calcul de second membre surfacique
nombre.nbiA = 1; // le nombre de point d'intégration pour
// le calcul de second membre linéique
nombre.nbiMas = 4; // le nombre de point d'intégration
// pour le calcul de la matrice masse

```

Il est possible d'utiliser une sous intégration avec 1 points d'intégration (cf. exemple de déclaration dans la table 44). Remarquons que dans ce cas on risque l'apparition de modes d'hourglass , une solution pour y remédier est de bloquer ces modes ce qui est possible avec l'élément à 1 points d'intégration. On se reportera à 22 pour plus d'information.

19.16 Quadrangle axisymétrique quadratique incomplet : QUAD_AXI QUADRATIQUE

L'élément s'appuie sur un élément de référence 2D quadratique (8 noeuds) (cf. 64).

Le nombre de point d'intégration par défaut est 4 (cf.20.1), et le nombre de noeuds est 8.

L'élément réagit aux sollicitations axisymétriques de traction, compression cisaillement. La discrétisation élément finis est de dimension 2, mais l'élément est relatif à un volume engendré par la rotation de l'élément réel autour de l'axe de rotation qui est y.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D : type contraintes volumique ou type déformations volumique.

L'espace de travail est de dimension 3.

La table (45) donne un exemple de syntaxe pour définir un élément quadrangle axisymétrique quadratique incomplet.

TABLE 45 – Exemple de déclaration d'un élément quadrangle axisymétriques avec interpolation quadratique incomplete

```

elements -----
  1 ELEMENTS
#-----
#| NO |           |           |
#|ELTS |   type element   |           |
#-----
      1      QUAD_AXI  QUADRATIQUE           1  2  3  4  5  6  7  8

```

Concernant les efforts externes, du au fait que l'élément est analytique suivant l'axe de rotation et discrétisé suivant les deux autres axes, on se reportera au §(33.2.9) pour la description des particularités du chargement.

En résumé pour les différents nombres on a :

```

nombre.nbne = 8; // le nombre de noeud de l'élément
nombre.nbneA = 3; // le nombre de noeud des aretes
nombre.nbi = 4; // le nombre de point d'intégration
                // pour le calcul mécanique
nombre.nbiEr = 9; // le nombre de point d'intégration
                // pour le calcul d'erreur
nombre.nbiS = 4; // le nombre de point d'intégration
                // pour le calcul de second membre surfacique
nombre.nbiA = 2; // le nombre de point d'intégration
                // pour le calcul de second membre linéique
nombre.nbiMas = 9; // le nombre de point d'intégration
                // pour le calcul de la matrice masse

```

Remarquons que dans le cas de 4 points d'intégration, ce nombre n'est pas tout à fait suffisant pour obtenir une matrice locale de singularité 3 en 2D (les 3 mouvements solides, en 3D s'ajoute les mouvements suivants la 3ième direction). L'élément est donc légèrement sous-intégré et il pourrait théoriquement apparaitre des modes d'hourglass. Une solution pour y remédier est de les bloquer ce qui est possible avec cet élément On se reportera à 22 pour plus d'information.

19.17 Quadrangle axisymétrique quadratique complet : QUAD_AXI QUADRACOMP

L'élément s'appuie sur un élément de référence 2D quadratique (9 noeuds) (cf. 64).

Le nombre de point d'intégration par défaut est 9 (cf.20.1), et le nombre de noeuds est 9.

L'élément réagit aux sollicitations axisymétriques de traction, compression cisaillement. La discrétisation élément finis est de dimension 2, mais l'élément est relatif à un volume engendré par la rotation de l'élément réel autour de l'axe de rotation qui est y.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D : type contraintes volumique ou type déformations volumique.

L'espace de travail est de dimension 3.

La table (46) donne un exemple de syntaxe pour définir un élément quadrangle axisymétrique quadratique complet.

TABLE 46 – Exemple de déclaration d'un élément quadrangle axisymétriques avec interpolation quadratique complete et plusieurs types de nombres de points d'intégration

```
elements -----
  2 ELEMENTS
#-----
#| NO |           |           |
#|ELTS|   type element   |           |
#-----
  1   QUAD_AXI QUADRACOMPL           1 2 3 4 5 6 7 8 9
  2   QUAD_AXI QUADRACOMPL _cm4pti   1 2 3 4 5 6 7 8 9
```

Concernant les efforts externes, du au fait que l'élément est analytique suivant l'axe de rotation et discrétisé suivant les deux autres axes, on se reportera au §(33.2.9) et §(34.8) pour la description des particularités du chargement.

En résumé pour les différents nombres on a :

```
nombre.nbne = 9; // le nombre de noeud de l'élément
nombre.nbneA = 3; // le nombre de noeud des aretes
nombre.nbi = 9; // le nombre de point d'intégration
// pour le calcul mécanique
nombre.nbiEr = 9; // le nombre de point d'intégration
// pour le calcul d'erreur
nombre.nbiS = 4; // le nombre de point d'intégration pour
// le calcul de second membre surfacique
nombre.nbiA = 2; // le nombre de point d'intégration pour
// le calcul de second membre linéique
nombre.nbiMas = 9; // le nombre de point d'intégration
// pour le calcul de la matrice masse
```

Il est possible d'utiliser une sous intégration avec 4 points d'intégration (cf. exemple de déclaration dans la table 46). Remarquons que dans ce cas on risque l'apparition de modes d'hourglass , une solution pour y remédier est de bloquer ces modes ce qui est possible avec l'élément à 4 points d'intégration. On se reportera à 22 pour plus d'information.

19.18 Quadrangle axisymétrique cubique complet : QUAD_AXI CUBIQUE

L'élément s'appuie sur un élément de référence 2D cubique (16 noeuds) (cf. 64).

Le nombre de point d'intégration par défaut est 16 (cf.20.1), et le nombre de noeuds est 16.

L'élément réagit aux sollicitations axisymétriques de traction, compression cisaillement. La discrétisation élément finis est de dimension 2, mais l'élément est relatif à un volume engendré par la rotation de l'élément réel autour de l'axe de rotation qui est y.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D : type contraintes volumique ou type déformations volumique.

L'espace de travail est de dimension 3.

La table (47) donne un exemple de syntaxe pour définir un élément quadrangle axisymétrique cubique.

TABLE 47 – Exemple de déclaration d'un élément quadrangle axisymétriques avec interpolation cubique et plusieurs types de nombres de points d'intégration

```

elements -----
  2 ELEMENTS
#-----
#| NO  |          |          |
#|ELTS |   type element   |          |
#-----
   1    QUAD_AXI  CUBIQUE      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
   2    QUAD_AXI  CUBIQUE  _cm9pti      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

```

Concernant les efforts externes, du au fait que l'élément est analytique suivant l'axe de rotation et discrétisé suivant les deux autres axes, on se reportera au §(33.2.9) et §(34.8) pour la description des particularités du chargement.

En résumé pour les différents nombres on a :

```

nbne = 16; // le nombre de noeud de l'élément
nbneA = 4; // le nombre de noeud des aretes
nbi = 16; // le nombre de point d'intégration pour le calcul mécanique
nbiEr = 16; // le nombre de point d'intégration pour le calcul d'erreur
nbiS = 9; // le nombre de point d'intégration pour le calcul de second membre surfacique
nbiA = 3; // le nombre de point d'intégration pour le calcul de second membre linéique
nbiMas = 16; // le nombre de point d'intégration pour le calcul de la matrice masse

```


Il est possible d'utiliser une sous intégration avec 9 points d'intégration (cf. exemple de déclaration dans la table 47). Remarquons que dans ce cas on risque l'apparition de modes d'hourglass , une solution pour y remédier est de bloquer ces modes ce qui est possible avec l'élément à 9 points d'intégration. On se reportera à 22 pour plus d'information.

19.19 Hexaédres linéaires : HEXAEDRE LINEAIRE

L'élément s'appuie sur un élément de référence 3D linéaire (65).

Le nombre de point d'intégration par défaut est 8 (cf.20.1), et le nombre de noeuds est 8.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci dans toutes les direction.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D.

L'élément ne peut être utilisé que dans un espace de travail à 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (48) donne un exemple de syntaxe pour définir des éléments hexaédriques linéaires.

TABLE 48 – Exemple de déclaration d'éléments hexaédrique avec interpolation linéaire

```
elements -----
      50 ELEMENTS
#-----
#| NO |           |           |
#|ELTS |   type element   |           |
#-----
      1   HEXAEDRE LINEAIRE      13   15   3   1   14   16   4   2
      2   HEXAEDRE LINEAIRE      15   17   5   3   16   18   6   4
      3   HEXAEDRE LINEAIRE      17   19   7   5   18   20   8   6
...

```

En résumé pour les différents nombres on a :

```
nbne = 8; // le nombre de noeud de l'élément
nbneS = 4; // le nombre de noeud des facettes
nbneA = 2; // le nombre de noeud des aretes
nbi = 8; // le nombre de point d'intégration pour le calcul mécanique
nbiEr = 8; // le nombre de point d'intégration pour le calcul d'erreur
nbiV = 8; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiS = 4; // le nombre de point d'intégration pour le calcul de second membre surfacique
nbiA = 2; // le nombre de point d'intégration pour le calcul de second membre linéique
nbiMas = 8; // le nombre de point d'intégration pour le calcul de la matrice masse

```

Il est également possible d'utiliser un nombre de point d'intégration (cf.20.1) supérieur 27 ou 64. La table (49) donne un exemple de déclaration.

TABLE 49 – Exemple de déclaration d’éléments hexaédriques linéaire avec différents nombres de points d’intégration

```
#-----
# NO      Type                Noeuds                |
#-----
# brique a interpolation linéaire et a 8 points d'intégration
1 HEXAEDRE LINEAIRE 1 2 3 4 5 6 7 8
# brique a interpolation linéaire et a 27 points d'intégration
2 HEXAEDRE LINEAIRE _cm27pti 1 2 3 4 5 6 7 8
# brique a interpolation linéaire et a 64 points d'intégration
3 HEXAEDRE LINEAIRE _cm64pti 1 2 3 4 5 6 7 8
```

19.20 Hexaédres quadratique incomplet : HEXAEDRE QUADRATIQUE

L’élément s’appuie sur un élément de référence 3D quadratique incomplet (66).

Le nombre de point d’intégration par défaut est 8 (cf.20.1), et le nombre de noeuds est 20.

L’élément réagit aux sollicitations de traction, compression cisaillement, ceci dans toutes les direction.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d’utiliser une loi de comportement associé 3D.

L’élément ne peut être utilisé que dans un espace de travail à 3 dimensions, en association avec n’importe quel autre type d’élément.

La table (50) donne un exemple de syntaxe pour définir des éléments hexaédriques quadratiques.

TABLE 50 – Exemple de déclaration d’éléments hexaédrique avec interpolation quadratique incomplète

```
elements -----
#-----
# NO      Type                Noeuds                |
#-----
# brique a interpolation incomplete a 8 points d'intégration
1 HEXAEDRE QUADRATIQUE 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
                        19 20
...
```

En résumé pour les différents nombres on a :

```
{ nbne = 20; // le nombre de noeud de l'élément
  nbneS = 8; // le nombre de noeud des facettes
```

```

nbneA = 3; // le nombre de noeud des aretes
nbi   = 8; // le nombre de point d'intégration pour le calcul mécanique
nbiEr = 27; // le nombre de point d'intégration pour le calcul d'erreur
nbiV  = 8; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiS  = 4; // le nombre de point d'intégration pour le calcul de second membre surfacique
nbiA  = 2; // le nombre de point d'intégration pour le calcul de second membre linéique
nbiMas = 27; // le nombre de point d'intégration pour le calcul de la matrice masse

```

Remarquons que dans le cas de 8 points d'intégration, ce nombre n'est pas suffisant pour obtenir une matrice locale de singularité 6 (les 6 mouvements solides). On peut alors également utiliser 27 points d'intégrations, ce qui correspond à 3 points de gauss sur chaque direction de l'espace de référence, ou 64 points d'intégrations, ce qui correspond à 4 points de gauss sur chaque direction. Enfin on peut choisir un seul point d'intégration, dans ce cas l'élément est fortement sous-intégré et on risque l'apparition de mode d'hourglass par exemple en dynamique. La table (51) donne un exemple de déclaration.

Cependant, il faut noter que lorsqu'il y a plusieurs éléments, le nombre de points d'intégration totale augmente plus vite que le nombre de ddl. Ainsi, bien souvent pour les éléments incomplets, on obtient au final une matrice non singulière. Herezh++ indique un Warning lorsqu'il y a un risque de singularité.

Le fait d'utiliser un nombre plus grand de points d'intégration peut-être intéressant pour augmenter la précision dans le cas d'une lois de comportement fortement non linéaire.

Enfin, il faut noter que ces éléments sont très sensibles à la distorsion. Ce qui signifie que dans le cas de transformations finies, le calcul n'est pas très robuste.

TABLE 51 – Exemple de déclaration d'éléments hexaédriques quadratique incomplet avec différents nombres de points d'intégration

```

#-----
# NO      Type                               Noeuds                               |
#-----
# brique a interpolation incomplete a 1 point d'intégration
1 HEXAEDRE QUADRATIQUE _cm1pti 1  2  3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
                               19 20
# brique a interpolation incomplete a 8 points d'intégration
1 HEXAEDRE QUADRATIQUE  1  2  3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
                               19 20
# brique a interpolation incomplete a 27 points d'intégration
2 HEXAEDRE QUADRATIQUE  _cm27pti 1  2  3 4 5 6 7 8 9 10 11 12 13 14 15
                               16 17 18 19 20
# brique a interpolation incomplete a 64 points d'intégration
3 HEXAEDRE QUADRATIQUE  _cm64pti 1  2  3 4 5 6 7 8 9 10 11 12 13 14 15
                               16 17 18 19 20

```

19.21 Hexaédres quadratique complet : HEXAEDRE QUADRA-COMPL

L'élément s'appuie sur un élément de référence 3D quadratique complet (67).

Le nombre de point d'intégration par défaut est 8 (cf.20.1), et le nombre de noeuds est 27.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci dans toutes les direction.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D.

L'élément ne peut être utilisé que dans un espace de travail à 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (52) donne un exemple de syntaxe pour définir des éléments hexaédriques quadratiques complet.

TABLE 52 – Exemple de déclaration d'éléments hexaédrique avec interpolation quadratique complète

```
elements -----
#-----
# NO      Type                Noeuds                |
#-----
# brique a interpolation incomplete a 8 points d'intégration
1 HEXAEDRE QUADRA-COMPL  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19
                        20  21  22  23  24  25  26  27
...

```

En résumé pour les différents nombres on a :

```
{ nbne = 27; // le nombre de noeud de l'élément
  nbneS = 9; // le nombre de noeud des facettes
  nbneA = 3; // le nombre de noeud des aretes
  nbi = 8; // le nombre de point d'intégration pour le calcul mécanique
  nbiEr = 27; // le nombre de point d'intégration pour le calcul d'erreur
  nbiV = 8; // le nombre de point d'intégration pour le calcul de second membre volumique
  nbiS = 4; // le nombre de point d'intégration pour le calcul de second membre surfacique
  nbiA = 2; // le nombre de point d'intégration pour le calcul de second membre linéique
  nbiMas = 27; // le nombre de point d'intégration pour le calcul de la matrice masse

```

Remarquons que dans le cas de 8 points d'intégration, ce nombre n'est pas suffisant pour obtenir une matrice locale de singularité 6 (les 6 mouvements solides). On peut alors également utiliser 27 points d'intégrations, ce qui correspond à 3 points de gauss sur chaque direction de l'espace de référence, ou 64 points d'intégrations, ce qui correspond à 4 points de gauss sur chaque direction. Enfin on peut choisir un seul point d'intégration, dans ce cas l'élément est fortement sous-intégré et on risque l'apparition de mode d'hourglass par exemple en dynamique. La table (53) donne un exemple de déclaration.

Cependant, il faut noter que lorsqu'il y a plusieurs éléments, le nombre de points d'intégration totale augmente plus vite que le nombre de ddl. Ainsi, on peut obtenir au final une matrice non singulière, mais l'apparition de singularités est beaucoup plus fréquente qu'avec les éléments quadratiques incomplet. Herezh++ indique un Warning lorsqu'il y a un risque de singularité.

Le fait d'utiliser un nombre plus grand de points d'intégration peut-être intéressant pour augmenter la précision dans le cas d'une lois de comportement fortement non linéaire.

Il faut noter que ces éléments sont beaucoup plus robuste à la distorsion que les quadratiques incomplets. A priori, ils sont plus performants lors de transformations finis.

TABLE 53 – Exemple de déclaration d'éléments hexaédriques quadratique avec différents nombres de points d'intégration

```
#-----
# NO      Type                Noeuds                |
#-----
# brique a interpolation complete a 1 point d'intégration
1 HEXAEDRE QUADRACOMPL _cm1pti 1  2  3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
                                20 21 22 23 24 25 26 27
# brique a interpolation complete a 8 points d'intégration
1 HEXAEDRE QUADRACOMPL  1  2  3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
                                20 21 22 23 24 25 26 27
# brique a interpolation complete a 27 points d'intégration
1 HEXAEDRE QUADRACOMPL _cm27pti 1  2  3 4 5 6 7 8 9 10 11 12 13 14 15 16
                                17 18 19 20 21 22 23 24 25 26 27
# brique a interpolation complete a 64 points d'intégration
1 HEXAEDRE QUADRACOMPL _cm64pti 1  2  3 4 5 6 7 8 9 10 11 12 13 14 15 16
                                17 18 19 20 21 22 23 24 25 26 27
```

19.22 Pentaèdre linéaires : PENTAEDRE LINEAIRE

L'élément s'appuie sur un élément de référence 3D tri-linéaire (68).

Le nombre de point d'intégration par défaut est 2 (cf.20.2), et le nombre de noeuds est 6.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci dans toutes les direction.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D.

L'élément ne peut être utilisé que dans un espace de travail à 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (54) donne un exemple de syntaxe pour définir des éléments pentaédrique linéaires.

En résumé pour les différents nombres on a :

TABLE 54 – Exemple de déclaration d’éléments pentaédriques avec interpolation linéaire

```
#-----
#| NO | | |
#|ELTS | type element | Noeuds |
#-----
# pentaedre a interpolation lineaire et a 2 points d'integration
1 PENTAEDRE LINEAIRE 1 2 3 4 5 6
...

nbne = 6; // le nombre de noeud de l'élément
nbneSQ = 4; // le nombre de noeud des facettes quadrangulaires
nbneST = 3; // le nombre de noeud des facettes triangulaires
nbneAQ = 2; // le nombre de noeud des aretes entres les faces triangulaires
nbneAT = 2; // le nombre de noeud des aretes des facettes triangulaires
nbI = 2; // nombre point d'intég pour le calcul méca pour l'élément
nbIQ = 1; // nombre point d'intég pour le calcul méca pour les triangles
nbIT = 4; // nombre point d'intég pour le calcul méca pour les quadrangles
nbIEr = 6; // le nombre de point d'intégration pour le calcul d'erreur
nbIV = 2; // le nombre de point d'intégration pour le calcul de second membre volumique
nbISQ = 4; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
nbIST = 1; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
nbIAQ = 1; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
nbIAT = 1; // nB pt integ pour calcul second membre linéique des arête des triangles
nbIMas = 6; // le nombre de point d'intégration pour le calcul de la matrice masse
```

Il est également possible d'utiliser un nombre de point d'intégration (cf.20.2) égale à 1 ou 6. La table (55) donne un exemple de déclaration pour 6 points d'intégration.

TABLE 55 – Exemple de déclaration d’éléments pentaédriques linéaire avec 6 points d’intégration

```
#-----
# NO Type Noeuds |
#-----
# pentaedre a interpolation lineaire et a 6 points d'integration
1 PENTAEDRE LINEAIRE _cm6pti 1 2 3 4 5 6
```

Dans le cas de 6 points d'intégrations les différents nombres sont :

```
nbne = 6; // le nombre de noeud de l'élément
nbneSQ = 4; // le nombre de noeud des facettes quadrangulaires
nbneST = 3; // le nombre de noeud des facettes triangulaires
nbneAQ = 2; // le nombre de noeud des aretes entres les faces triangulaires
nbneAT = 2; // le nombre de noeud des aretes des facettes triangulaires
nbI = 6; // nombre point d'intég pour le calcul méca pour l'élément
nbIQ = 1; // nombre point d'intég pour le calcul méca pour les triangles
```

```

nbiT    = 4; // nombre point d'intég pour le calcul méca pour les quadrangles
nbiEr   = 6; // le nombre de point d'intégration pour le calcul d'erreur
nbiV    = 6; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiSQ   = 4; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
nbiST   = 1; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
nbiAQ   = 1; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
nbiAT   = 1; // nB pt integ pour calcul second membre linéique des arête des triangles
      nbiMas = 6; // le nombre de point d'intégration pour le calcul de la matrice masse

```

19.23 Pentaèdre quadratique incomplet : PENTAEDRE QUADRATIQUE

L'élément s'appuie sur un élément de référence 3D tri-quadratique incomplet (69).

Le nombre de point d'intégration par défaut est 6 (cf.20.2), et le nombre de noeuds est 15.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci dans toutes les direction.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D.

L'élément ne peut être utilisé que dans un espace de travail à 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (56) donne un exemple de syntaxe pour définir des éléments pentaédrique quadratique incomplet.

TABLE 56 – Exemple de déclaration d'éléments pentaédriques avec interpolation quadratique incomplet

```

#-----
#| NO  |           |           |
#|ELTS|   type element   |           |
#-----
      # pentaedre a interpolation quadratique incomplete a 6 points d'intégration
      1 PENTAEDRE QUADRATIQUE  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
...

```

En résumé pour les différents nombres on a :

```

      nbne    = 15; // le nombre de noeud de l'élément
nbneSQ  = 8; // le nombre de noeud des facettes quadrangulaires
nbneST  = 6; // le nombre de noeud des facettes triangulaires
nbneAQ  = 3; // le nombre de noeud des aretes entres les faces triangulaires
nbneAT  = 3; // le nombre de noeud des aretes des facettes triangulaires
nbi     = 6; // nombre point d'intég pour le calcul méca pour l'élément
nbiQ    = 3; // nombre point d'intég pour le calcul méca pour les triangles
nbiT    = 4; // nombre point d'intég pour le calcul méca pour les quadrangles
nbiEr   =18; // le nombre de point d'intégration pour le calcul d'erreur
nbiV    = 6; // le nombre de point d'intégration pour le calcul de second membre volumique

```

```

nbiSQ = 4; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
nbiST = 3; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
nbiAQ = 2; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
nbiAT = 2; // nB pt integ pour calcul second membre linéique des arête des triangles
      nbiMas = 18; // le nombre de point d'intégration pour le calcul de la matrice masse

```

Il est également possible d'utiliser un nombre de point d'intégration (cf.20.2) égale à 9, 12 ou 18. La table (57) donne un exemple de déclaration.

TABLE 57 – Exemple de déclaration d'éléments pentaédriques quadratique avec différents nombres de points d'intégration

```

#-----
# NO      Type                Noeuds                |
#-----
# pentaedre a interpolation quadratique incomplete a 6 points d'intégration
1 PENTAEDRE QUADRATIQUE 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
# pentaedre a interpolation quadratique incomplete a 9 points d'intégration
1 PENTAEDRE QUADRATIQUE _cm9pti 1 2 3 4 5 6 7 8 9 10 11 12 13
# pentaedre a interpolation quadratique incomplete a 12 points d'intégration
1 PENTAEDRE QUADRATIQUE _cm12pti 1 2 3 4 5 6 7 8 9 10 11 12 13
# pentaedre a interpolation quadratique incomplete a 18 points d'intégration
1 PENTAEDRE QUADRATIQUE _cm18pti 1 2 3 4 5 6 7 8 9 10 11 12 13

```

Dans le cas de 9 points d'intégrations les différents nombres sont :

```

nbne      = 15; // le nombre de noeud de l'élément
nbneSQ    = 8; // le nombre de noeud des facettes quadrangulaires
nbneST    = 6; // le nombre de noeud des facettes triangulaires
nbneAQ    = 3; // le nombre de noeud des aretes entres les faces triangulaires
nbneAT    = 3; // le nombre de noeud des aretes des facettes triangulaires
nbi       = 9; // nombre point d'intég pour le calcul méca pour l'élément
nbiQ      = 3; // nombre point d'intég pour le calcul méca pour les triangles
nbiT      = 4; // nombre point d'intég pour le calcul méca pour les quadrangles
nbiEr     =18; // le nombre de point d'intégration pour le calcul d'erreur
nbiV      = 6; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiSQ     = 4; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
nbiST     = 3; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
nbiAQ     = 2; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
nbiAT     = 2; // nB pt integ pour calcul second membre linéique des arête des triangles
      nbiMas = 18; // le nombre de point d'intégration pour le calcul de la matrice masse

```

Dans le cas de 12 points d'intégrations les différents nombres sont :

```

nbne      = 15; // le nombre de noeud de l'élément
nbneSQ    = 8; // le nombre de noeud des facettes quadrangulaires
nbneST    = 6; // le nombre de noeud des facettes triangulaires
nbneAQ    = 3; // le nombre de noeud des aretes entres les faces triangulaires
nbneAT    = 3; // le nombre de noeud des aretes des facettes triangulaires
nbi       = 12; // nombre point d'intég pour le calcul méca pour l'élément

```



```

nbiQ    = 3; // nombre point d'intég pour le calcul méca pour les triangles
nbiT    = 4; // nombre point d'intég pour le calcul méca pour les quadrangles
nbiEr   =18; // le nombre de point d'intégration pour le calcul d'erreur
nbiV    = 6; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiSQ   = 4; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
nbiST   = 3; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
nbiAQ   = 2; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
nbiAT   = 2; // nB pt integ pour calcul second membre linéique des arête des triangles
      nbiMas = 18; // le nombre de point d'intégration pour le calcul de la matrice masse

```

Dans le cas de 18 points d'intégrations les différents nombres sont :

```

      nbne    = 15; // le nombre de noeud de l'élément
nbneSQ  = 8; // le nombre de noeud des facettes quadrangulaires
nbneST  = 6; // le nombre de noeud des facettes triangulaires
nbneAQ  = 3; // le nombre de noeud des aretes entres les faces triangulaires
nbneAT  = 3; // le nombre de noeud des aretes des facettes triangulaires
nbi     = 18; // nombre point d'intég pour le calcul méca pour l'élément
nbiQ    = 3; // nombre point d'intég pour le calcul méca pour les triangles
nbiT    = 4; // nombre point d'intég pour le calcul méca pour les quadrangles
nbiEr   =18; // le nombre de point d'intégration pour le calcul d'erreur
nbiV    = 6; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiSQ   = 4; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
nbiST   = 3; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
nbiAQ   = 2; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
nbiAT   = 2; // nB pt integ pour calcul second membre linéique des arête des triangles
      nbiMas = 18; // le nombre de point d'intégration pour le calcul de la matrice masse

```

19.24 Pentaèdre quadratique complet : PENTAEDRE QUADRACOMPL

L'élément s'appuie sur un élément de référence 3D tri-quadratique complet (67).

Le nombre de point d'intégration par défaut est 6 (cf.20.2), et le nombre de noeuds est 18.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci dans toutes les direction.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D.

L'élément ne peut être utilisé que dans un espace de travail à 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (58) donne un exemple de syntaxe pour définir des éléments pentaédrique quadratique complet.

En résumé pour les différents nombres on a :

```

      nbne    = 18; // le nombre de noeud de l'élément
nbneSQ  = 9; // le nombre de noeud des facettes quadrangulaires
nbneST  = 6; // le nombre de noeud des facettes triangulaires
nbneAQ  = 3; // le nombre de noeud des aretes entres les faces triangulaires
nbneAT  = 3; // le nombre de noeud des aretes des facettes triangulaires
nbi     = 6; // nombre point d'intég pour le calcul méca pour l'élément
nbiQ    = 3; // nombre point d'intég pour le calcul méca pour les triangles

```

TABLE 58 – Exemple de déclaration d’éléments pentaédriques avec interpolation quadratique complet

```
#-----
#| NO | | |
#|ELTS | type element | Noeuds |
#-----
# pentaedre a interpolation quadratique complete a 6 points d'integration
1 PENTAEDRE QUADRACOMPL 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
...

nbiT = 4; // nombre point d'intég pour le calcul méca pour les quadrangles
nbiEr =18; // le nombre de point d'intégration pour le calcul d'erreur
nbiV = 6; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiSQ = 4; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
nbiST = 3; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
nbiAQ = 2; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
nbiAT = 2; // nB pt integ pour calcul second membre linéique des arête des triangles
nbiMas = 18; // le nombre de point d'intégration pour le calcul de la matrice masse
```

Il est également possible d'utiliser un nombre de point d'intégration (cf.20.2) égale à 9, 12 ou 18. La table (59) donne un exemple de déclaration.

TABLE 59 – Exemple de déclaration d’éléments pentaédriques quadratique complet avec différents nombres de points d’intégration

```
#-----
# NO Type Noeuds |
#-----
# pentaedre a interpolation quadratique complete a 6 points d'integration
1 PENTAEDRE QUADRACOMPL 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
# pentaedre a interpolation quadratique complete a 9 points d'integration
2 PENTAEDRE QUADRACOMPL _cm9pti 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
# pentaedre a interpolation quadratique complete a 12 points d'integration
3 PENTAEDRE QUADRACOMPL _cm12pti 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
# pentaedre a interpolation quadratique complete a 18 points d'integration
4 PENTAEDRE QUADRACOMPL _cm18pti 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
```

Dans le cas de 9 points d’intégrations les différents nombres sont :

```
nbne = 18; // le nombre de noeud de l'élément
nbneSQ = 9; // le nombre de noeud des facettes quadrangulaires
nbneST = 6; // le nombre de noeud des facettes triangulaires
nbneAQ = 3; // le nombre de noeud des aretes entres les faces triangulaires
nbneAT = 3; // le nombre de noeud des aretes des facettes triangulaires
nbi = 9; // nombre point d'intég pour le calcul méca pour l'élément
```

```

nbiQ   = 3; // nombre point d'intég pour le calcul méca pour les triangles
nbiT   = 4; // nombre point d'intég pour le calcul méca pour les quadrangles
nbiEr  =18; // le nombre de point d'intégration pour le calcul d'erreur
nbiV   = 6; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiSQ  = 4; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
nbiST  = 3; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
nbiAQ  = 2; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
nbiAT  = 2; // nB pt integ pour calcul second membre linéique des arête des triangles
      nbiMas = 18; // le nombre de point d'intégration pour le calcul de la matrice masse

```

Dans le cas de 12 points d'intégrations les différents nombres sont :

```

      nbne   = 18; // le nombre de noeud de l'élément
nbneSQ = 9; // le nombre de noeud des facettes quadrangulaires
nbneST = 6; // le nombre de noeud des facettes triangulaires
nbneAQ = 3; // le nombre de noeud des aretes entres les faces triangulaires
nbneAT = 3; // le nombre de noeud des aretes des facettes triangulaires
nbi    = 12; // nombre point d'intég pour le calcul méca pour l'élément
nbiQ   = 3; // nombre point d'intég pour le calcul méca pour les triangles
nbiT   = 4; // nombre point d'intég pour le calcul méca pour les quadrangles
nbiEr  =18; // le nombre de point d'intégration pour le calcul d'erreur
nbiV   = 6; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiSQ  = 4; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
nbiST  = 3; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
nbiAQ  = 2; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
nbiAT  = 2; // nB pt integ pour calcul second membre linéique des arête des triangles
      nbiMas = 18; // le nombre de point d'intégration pour le calcul de la matrice masse

```

Dans le cas de 18 points d'intégrations les différents nombres sont :

```

      nbne   = 18; // le nombre de noeud de l'élément
nbneSQ = 9; // le nombre de noeud des facettes quadrangulaires
nbneST = 6; // le nombre de noeud des facettes triangulaires
nbneAQ = 3; // le nombre de noeud des aretes entres les faces triangulaires
nbneAT = 3; // le nombre de noeud des aretes des facettes triangulaires
nbi    = 6; // nombre point d'intég pour le calcul méca pour l'élément
nbiQ   = 3; // nombre point d'intég pour le calcul méca pour les triangles
nbiT   = 4; // nombre point d'intég pour le calcul méca pour les quadrangles
nbiEr  =18; // le nombre de point d'intégration pour le calcul d'erreur
nbiV   = 6; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiSQ  = 4; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
nbiST  = 3; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
nbiAQ  = 2; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
nbiAT  = 2; // nB pt integ pour calcul second membre linéique des arête des triangles
      nbiMas = 18; // le nombre de point d'intégration pour le calcul de la matrice masse

```

19.25 Tétraèdre linéaire : TETRAEDRE LINEAIRE

L'élément s'appuie sur un élément de référence 3D tri-linéaire (71).

Le nombre de point d'intégration par défaut est 1 (cf.74), et le nombre de noeuds est 4.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci dans toutes les direction.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D.

L'élément ne peut être utilisé que dans un espace de travail à 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (60) donne un exemple de syntaxe pour définir des éléments tétraédrique linéaire.

TABLE 60 – Exemple de déclaration d'éléments tétraèdre avec interpolation linéaire

```
#-----
#| NO | | |
#|ELTS | type element | Noeuds |
#-----
#
# tétraèdre a interpolation linéaire et a 1 point d'intégration
1 TETRAEDRE LINEAIRE 1 2 3 4
...

```

En résumé pour les différents nombres on a :

```
nbne = 4; // le nombre de noeud de l'élément
nbneS = 3; // le nombre de noeud des facettes
nbneA = 2; // le nombre de noeud des aretes
nbi = 1; // le nombre de point d'intégration pour le calcul mécanique
nbiEr = 4; // le nombre de point d'intégration pour le calcul d'erreur
nbiV = 1; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiS = 1; // le nombre de point d'intégration pour le calcul de second membre surfacique
nbiA = 1; // le nombre de point d'intégration pour le calcul de second membre linéique
nbiMas = 4; // le nombre de point d'intégration pour le calcul de la matrice masse

```

19.26 Tétraèdre quadratique : TETRAEDRE QUADRATIQUE

L'élément s'appuie sur un élément de référence 3D tri-quadratique (72).

Le nombre de point d'intégration par défaut est 1 (cf.74), et le nombre de noeuds est 4.

L'élément réagit aux sollicitations de traction, compression cisaillement, ceci dans toutes les direction.

La dimension des tenseurs locaux de contraintes, déformation, vitesse de déformation ... est 3. Il est donc nécessaire d'utiliser une loi de comportement associé 3D.

L'élément ne peut être utilisé que dans un espace de travail à 3 dimensions, en association avec n'importe quel autre type d'élément.

La table (61) donne un exemple de syntaxe pour définir des éléments tetraédriques linéaires.

En résumé pour les différents nombres on a :

```
nbne = 10; // le nombre de noeud de l'élément
nbneS = 6; // le nombre de noeud des facettes
nbneA = 3; // le nombre de noeud des aretes
nbi = 4; // le nombre de point d'intégration pour le calcul mécanique

```

TABLE 61 – Exemple de déclaration d’éléments tétraédriques avec interpolation quadratique

```

#-----
#| NO | | |
#|ELTS | type element | Noeuds |
#-----
#
# tétraèdre a interpolation quadratique et a 4 points d'intégration
1 TETRAEDRE QUADRACOMPL 1 2 3 4 5 6 7 8 9 10
...

nbiEr = 15; // le nombre de point d'intégration pour le calcul d'erreur
nbiV = 4; // le nombre de point d'intégration pour le calcul de second membre volumique
nbiS = 3; // le nombre de point d'intégration pour le calcul de second membre surfacique
nbiA = 2; // le nombre de point d'intégration pour le calcul de second membre linéique
nbiMas = 15; // le nombre de point d'intégration pour le calcul de la matrice masse

```

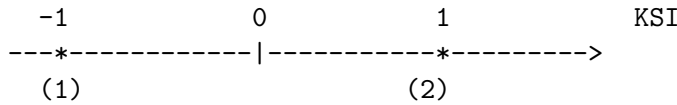
Pour mémoire on rappelle les différentes numérotations des éléments de référence.

19.27 Eléments de référence s'appuyant sur une géométrie 1D

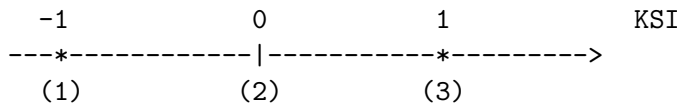
La table (62) montre la numérotation des différents éléments 1D de référence utilisés.

TABLE 62 – numérotation des éléments 1D de référence

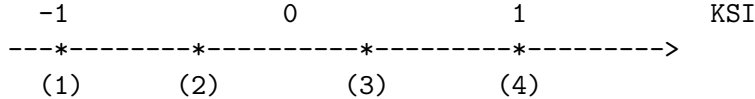
linéaire -> 2 noeuds



quadratique -> 3 noeuds



cubique -> 4 noeuds



On remarque que les noeuds sont numéroté dans l'ordre de la coordonnée croissante.

19.28 Position des points d'intégrations pour les éléments de géométrie 1D

La position des points d'intégrations est la suivante par la coordonnée ξ sur l'élément de référence (on indique également les poids d'intégration associés w) :

- 1 point d'intégration : $\xi = 0, w(1) = 2$
- 2 points d'intégration : $\xi = -1/\sqrt{3}$ et $\xi = 1/\sqrt{3}$,
 $w(1) = w(2) = 1$
- 3 points d'intégration : $\xi = -\sqrt{(3./5.)}$, $\xi = 0.$, $\xi = \sqrt{(3./5.)}$
 $w(2) = 8./9.$; $w(1) = 5./9.$; $w(3) = 5./9.$;
- 4 points d'intégration : $\xi = -T_1$, $\xi = -T$, $\xi = T$, $\xi = T_1$, avec $T_1 = \sqrt{\frac{3+S}{7}} \approx 0.86$,
 $T = \sqrt{\frac{3-S}{7}} \approx 0.33$ et $S = 2.\sqrt{\frac{6}{5}}$.
 $w(3) = 0.5 + S$; $w(2) = 0.5 + S$; $w(4) = 0.5 - S$; $w(1) = 0.5 - S$; avec $S = 1./(6.\sqrt{(6./5.)})$;

19.29 Eléments de référence à géométrie surfacique 2D triangulaire

La numérotation des éléments de référence 2D triangulaire est indiquée à la table (63).

TABLE 63 – numérotation des éléments triangulaires de référence

lineaire:	quadratique :	(3)	cubique
(3) \	(3) \	\	
\	\	\	
\	(6) \ (5)	(8) \ (7)	
(1) ____\ (2)	\	\	
	(1) ____\ (2)	(9) (10)\ (6)	
	(4)	\	
		\	
		(1) ----- (2)	

concernant les aretes
elles sont lineaire quadratique ou cubique comme l'element (4) (5)

numerotation des arrêtes :

en linéaire : 1 : 1 2, 2 : 2 3, 3 : 3 1
 en quadratique : 1 : 1 4 2, 2 : 2 5 3, 3 : 3 6 1
 en cubique: 1 : 1 4 5 2 2: 2 6 7 3 3: 3 8 9 1

leur nombre de point d'integration nbil depend du nombre nbi de l'element

nbi = 1 -> nbil = 1

nbi = 3 ou 4 -> nbil = 2

concernant la triangulation linéaire de l'élément :

- pour l'élément linéaire : décomposé en lui même
- pour l'élément quadratique : décomposé en 4 éléments triangulaires de connexion :
 1e : 6 4 5, 2e : 1 4 6, 3e : 4 2 5, 4e : 6 5 3
- pour l'élément cubique : décomposé en 9 éléments triangulaires de connexion
 1e : 1 4 9 2e : 10 9 4 3e: 4 5 10 4e: 6 10 5
 5e : 5 2 6 6e : 9 10 8 7e: 7 8 10 8e: 10 6 7 9e: 8 7 3

19.30 Position des points d'intégrations pour les éléments de géométrie 2D triangulaires

La position des points d'intégrations de coordonnées (ξ, η) sur l'élément de référence, et la valeur des poids d'intégration sont les suivantes :

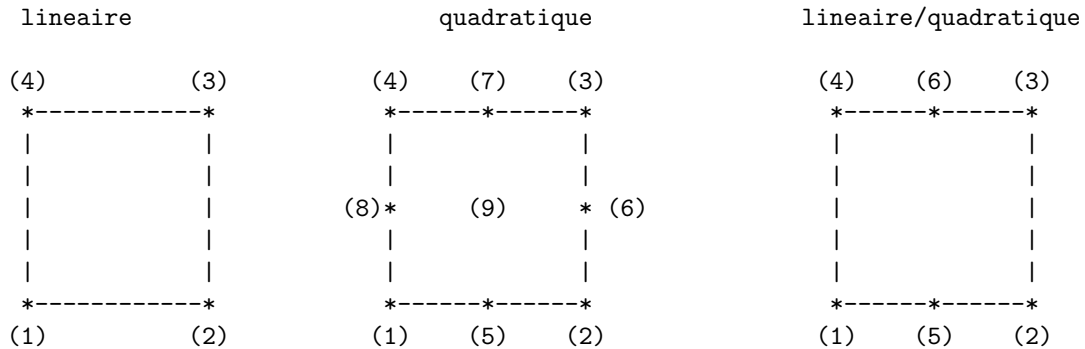
- 1 point d'intégration : $\xi(1) = 1/3.$, $\eta(1) = 1/3.$, $w_1 = 0.5$
- 3 points d'intégration sur les arrêtes :
 $\xi(1) = 0.5$ et $\eta(1) = 0.5$, $w(1) = 1/6$;
 $\xi(2) = 0.$ et $\eta(2) = 0.5$, $w(2) = 1/6$;
 $\xi(3) = 0.5$ et $\eta(3) = 0.$, $w(3) = 1/6$;
- 3 points d'intégration internes :
 $\xi(1) = 1/6$ et $\eta(1) = 1/6$, $w(1) = 1/6$;

- $\xi(2) = 2/3$ et $\eta(2) = 1/6$, $w(2) = 1/6$;
 $\xi(3) = 1/6$ et $\eta(3) = 2/3$, $w(3) = 1/6$;
 – 4 points d'intégration internes :
 $\xi(1) = 1/3$ et $\eta(1) = 1/3$, $w(1) = -27/96$;
 $\xi(2) = 1/5$ et $\eta(2) = 1/5$, $w(2) = 25/96$;
 $\xi(3) = 3/5$ et $\eta(3) = 1/5$, $w(3) = 25/96$;
 $\xi(4) = 1/5$ et $\eta(4) = 3/5$, $w(4) = 25/96$;
 – 6 points d'intégration internes : on pose $a = a = 0.445948490915965$, $b = 0.091576213509771$
- $\xi(1) = a$ et $\eta(1) = a$, $w(1) = 0.111690794839005$;
 $\xi(2) = 1 - 2a$ et $\eta(2) = a$, $w(2) = 0.111690794839005$;
 $\xi(3) = a$ et $\eta(3) = 1 - 2a$, $w(3) = 0.111690794839005$;
 $\xi(4) = b$ et $\eta(4) = b$, $w(4) = 0.054975871827661$;
 $\xi(5) = 1 - 2b$ et $\eta(5) = b$, $w(5) = 0.054975871827661$;
 $\xi(6) = b$ et $\eta(6) = 1 - 2b$, $w(6) = 0.054975871827661$;
 – 7 points d'intégration internes : on pose $a = a = (6. + \sqrt{(15.)})/21.$, $b = 4./7. - a$,
 $A = (155. + \sqrt{(15.)})/2400.$,
 $\xi(1) = 1/3$ et $\eta(1) = 1/3$, $w(1) = 9./80.$;
 $\xi(2) = a$ et $\eta(2) = a$, $w(1) = A$;
 $\xi(3) = 1 - 2a$ et $\eta(3) = a$, $w(2) = A$;
 $\xi(4) = a$ et $\eta(4) = 1 - 2a$, $w(3) = A$;
 $\xi(5) = b$ et $\eta(5) = b$, $w(4) = 31/240 - A$;
 $\xi(6) = 1 - 2b$ et $\eta(6) = b$, $w(5) = 31/240 - A$;
 $\xi(7) = b$ et $\eta(7) = 1 - 2b$, $w(6) = 31/240 - A$;

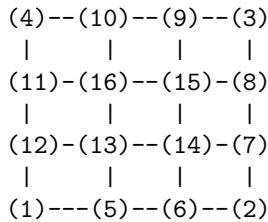
19.31 Eléments de référence à géométrie surfacique 2D quadrangulaire

La table (64) est relatives aux quadrangles.

TABLE 64 – numérotations des éléments de référence quadrangulaires



cubique complet



dans le cas d'un quadratique incomplet, nbne = 8, le noeud (9) est supprime

face 1 : noeuds de l'élément

on attribue le même nombre de points d'integration pour la face que pour l'élément

pour les aretes, 4 aretes

1) pour le quadrangle bilinéaire

1 2 2 3 3 4 4 1

2) pour le quadrangle quadratique complet ou incomplet

1 5 2 2 6 3 3 7 4 4 8 1

3) pour le quadrangle cubique complet

1 5 6 2 2 7 8 3 3 9 10 4 4 11 12 1

On attribue la racine carré du nombre de point d'intégration de l'élément pour l'arrête. D'où en général : 1 point d'integration par arete pour les bilinéaires et 2 points pour les quadratiques.

19.32 Position des points d'intégrations pour les éléments de géométrie 2D quadrangulaires

La position des points d'intégrations de coordonnées (ξ, η) sur l'élément de référence, et la valeur des poids d'intégration sont calculées à partir de celles de l'élément de référence 1D, appliquée d'abord suivant ξ puis suivant η . Ainsi par exemple pour 4 points d'intégration on obtient :

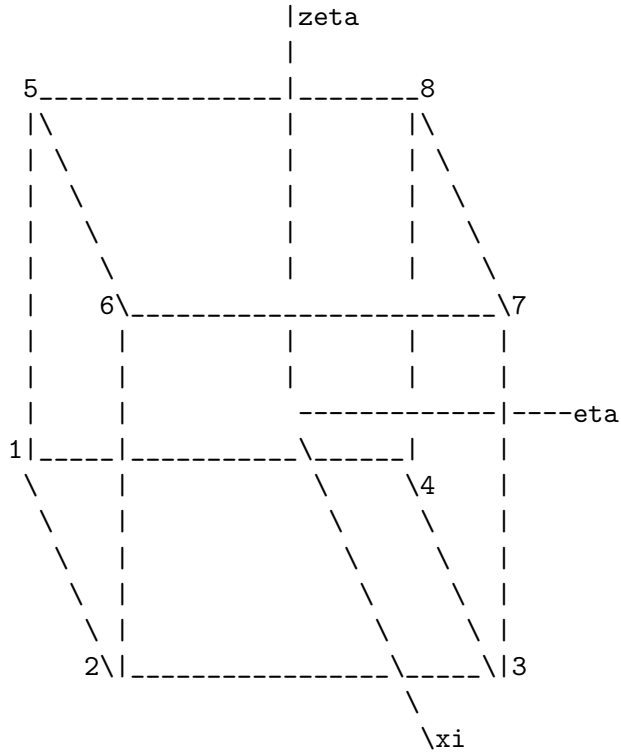
- 4 points d'intégration
- $\xi(1) = -1/\sqrt{3}$ et $\eta(1) = -1/\sqrt{3}$, $w(1) = 1$;
- $\xi(2) = 1/\sqrt{3}$ et $\eta(2) = -1/\sqrt{3}$, $w(2) = 1$;
- $\xi(3) = -1/\sqrt{3}$ et $\eta(3) = 1/\sqrt{3}$, $w(3) = 1$;
- $\xi(4) = 1/\sqrt{3}$ et $\eta(4) = 1/\sqrt{3}$, $w(4) = 1$;

19.33 Eléments de référence à géométrie 3D hexaédrique

La numérotation des éléments 3D est indiquée par les tables suivantes :

- la table (65) concerne les hexaèdres linéaires,
- la table (66) concerne les hexaèdres quadratiques,
- la table (68) concerne les pentaèdres linéaires,
- la table (69) concerne les pentaèdres quadratiques,
- la table (71) concerne les tétraèdres linéaires,
- la table (72) concerne les tétraèdres quadratiques,

TABLE 65 – numérotation de l'élément de référence hexaédrique linéaire

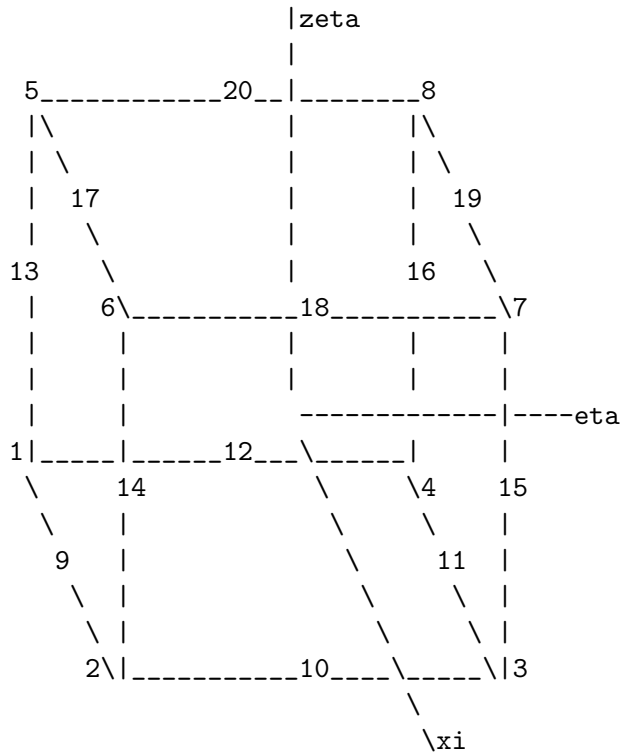


face 1 : noeud 1 4 3 2, face 2 : noeud 1 5 8 4,
 face 3 : noeud 1 2 6 5, face 4 : noeud 5 6 7 8,
 face 5 : noeud 2 3 7 6, face 6 : noeud 3 4 8 7,
 les normales sortent des faces des elements

Les 12 arrêtes

1:1 2	2:2 3	3:3 4	4:4 1
5:1 5	6:2 6	7:3 7	8:4 8
9:5 6	10:6 7	11:7 8	12:8 5

TABLE 66 – numérotation de l'élément de référence pour les hexaèdres quadratiques incomplet



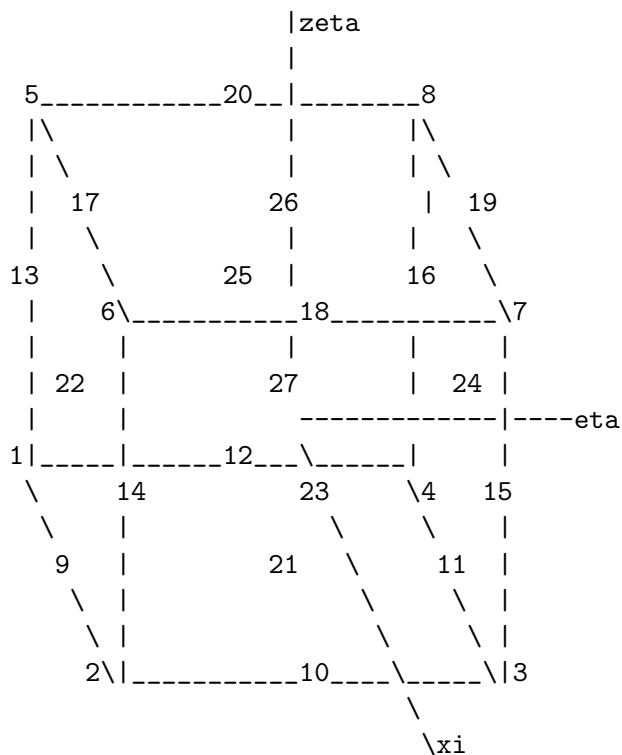
face 1 : noeud 1 4 3 2 12 11 10 9,
 face 2 : noeud 1 5 8 4 13 20 16 12,
 face 3 : noeud 1 2 6 5 9 14 17 13,
 face 4 : noeud 5 6 7 8 17 18 19 20,
 face 5 : noeud 2 3 7 6 10 15 18 14,
 face 6 : noeud 3 4 8 7 11 16 19 15,

les normales sortent des faces des elements

Les 12 aretes :

1:1 9 2	2:2 10 3	3:3 11 4	4:4 12 1
5:1 13 5	6:2 14 6	7:3 15 7	8:4 16 8
9:5 17 6	10:6 18 7	11:7 19 8	12:8 20 5

TABLE 67 – numérotation de l'élément de référence pour les hexaèdres quadratiques complets



par rapport au quadratique incomplet, 21 est au centre de la face 1,
 22 sur la face 3, 23 sur la face 5, 24 sur la face 6
 25 sur la face 2, 26 sur la face 4, 27 au centre de l'élément

face 1 : noeud 1 4 3 2 12 11 10 9 21, face 2 : noeud 1 5 8 4 13 20 16 12 25,
 face 3 : noeud 1 2 6 5 9 14 17 13 22, face 4 : noeud 5 6 7 8 17 18 19 20 26,
 face 5 : noeud 2 3 7 6 10 15 18 14 23, face 6 : noeud 3 4 8 7 11 16 19 15 24,

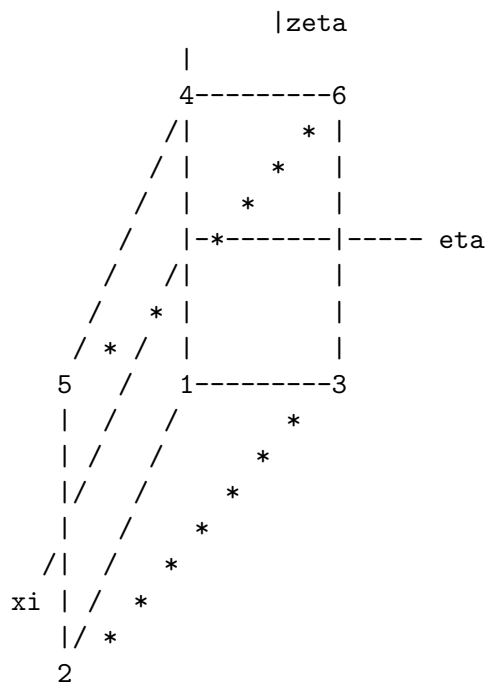
les normales sortent des faces des elements

Les 12 aretes :

1:1 9 2	2:2 10 3	3:3 11 4	4:4 12 1
5:1 13 5	6:2 14 6	7:3 15 7	8:4 16 8
9:5 17 6	10:6 18 7	11:7 19 8	12:8 20 5

19.34 Éléments de référence à géométrie 3D pentaédrique

TABLE 68 – numérotation de l'élément pentaédrique linéaire de référence.



pentaedre trilineaire

Description des faces , puis des arêtes

face 1 : noeud 1 3 2, face 2 : noeud 1 4 6 3,

face 3 : noeud 1 2 5 4, face 4 : noeud 4 5 6,

face 5 : noeud 2 3 6 5

les normales sortent des faces des elements

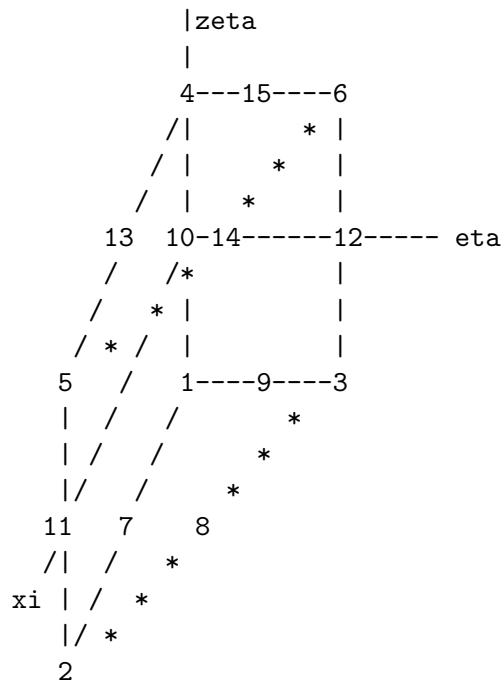
Les 9 aretes :

1-> 1 2 2->2 3 3->3 1

4-> 1 4 5->2 5 6->3 6

7-> 4 5 8->5 6 9->6 4

TABLE 69 – Numérotation de l'élément pentaédrique quadratique incomplet de référence.



pentaedre triquadratique incomplet

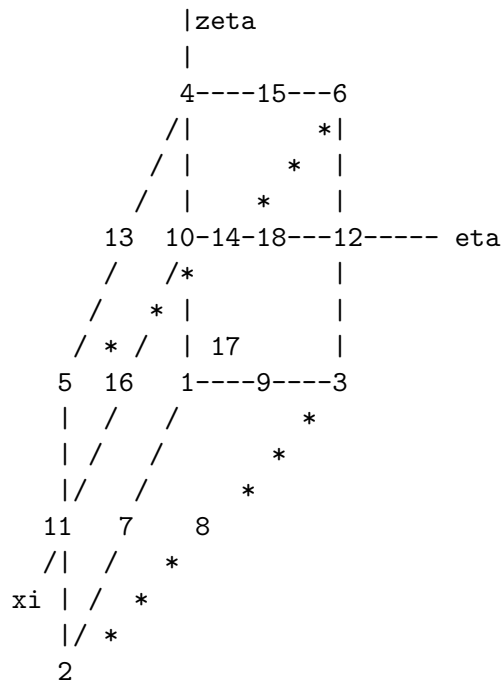
Cas du triquadratique -> description des faces

- face 1 : noeud 9 3 8 2 7 1,
 - face 2 : noeud 9 1 10 4 15 6 12 3,
 - face 3 : noeud 7 2 11 5 13 4 10 1,
 - face 4 : noeud 15 6 14 5 13 4,
 - face 5 : noeud 8 3 12 6 14 5 11 2,
- Les normales sortent des faces des elements,

Les 9 aretes :

- 1->1 7 2 2->2 8 3 3->3 9 1
- 4->1 10 4 5->2 11 5 6->3 12 6
- 7->4 13 5 8->5 14 6 9->6 15 4

TABLE 70 – Numérotation de l'élément pentaédrique quadratique complet de référence.



pentaèdre triquadratique complet

Description des faces

face 1 : noeud 1 3 2 9 8 7,

face 2 : noeud 1 4 6 3 10 15 12 9 18,

face 3 : noeud 1 2 5 4 7 11 13 10 16, face 4 : noeud 4 5 6 13 14 15,

face 5 : noeud 2 3 6 5 8 12 14 11 17

les normales sortent des faces des elements

9 aretes

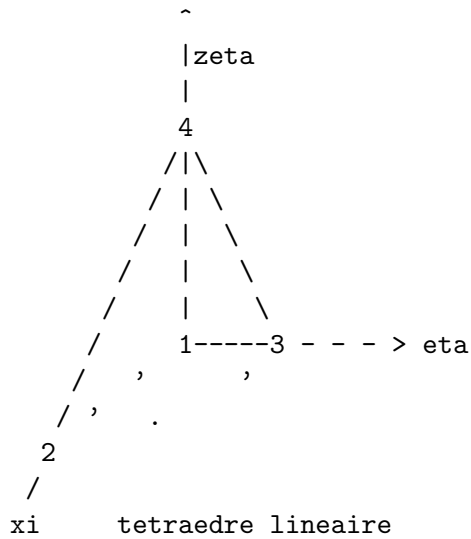
1->1 7 2 2->2 8 3 3->3 9 1

4->1 10 4 5->2 11 5 6->3 12 6

7->4 13 5 8->5 14 6 9->6 15 4

19.35 Eléments de référence à géométrie 3D tétraédrique

TABLE 71 – Numérotation de l'élément tétraédrique linéaire de référence.

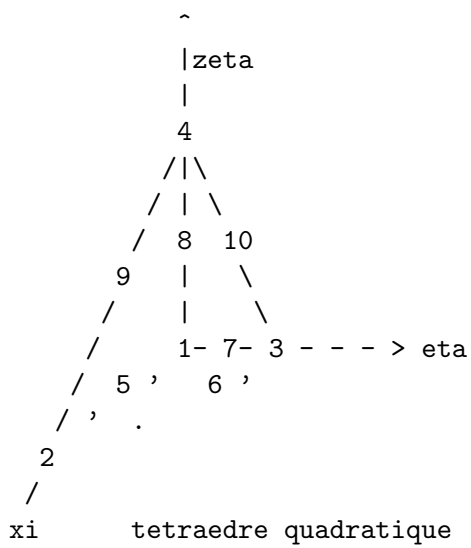


face 1 : noeud 1 3 2 , face 2 : noeud 1 4 3,
 face 3 : noeud 1 2 4, face 4 : noeud 2 3 4,
 Les normales sortent des faces des elements

Les 6 aretes :

A1-> 1 2 A2-> 2 3 A3-> 3 1
 A4-> 1 4 A5-> 2 4 A6-> 3 4

TABLE 72 – Numérotation de l'élément tétraédrique quadratique de référence.



face 1 : noeud 7 3 6 2 5 1 ,
 face 2 : noeud 5 2 9 4 8 1 ,
 face 3 : noeud 7 1 8 4 10 3 ,
 face 4 : noeud 6 3 10 4 9 2 ,

Les normales sortent des faces des elements.

Les 6 aretes :

A1-> 1 5 2 A2-> 2 6 3 A3-> 3 7 1
 A4-> 1 8 4 A5-> 2 9 4 A6-> 3 10 4

20 Positions des points d'intégrations

20.1 Position des points d'intégrations pour les éléments hexaédriques

La position des points d'intégrations de coordonnées (ξ, η, ζ) sur l'élément de référence, et la valeur des poids d'intégration sont calculées à partir de celles de l'élément de référence 1D, appliquée d'abord suivant ξ puis suivant η puis enfin sur ζ sauf pour 8 points d'intégration où l'on a :

– 8 points d'intégration

$$\begin{aligned} \xi(1) &= a, \eta(1) = a, \zeta(1) = a, w(1) = 1; \\ \xi(2) &= a, \eta(1) = a, \zeta(1) = -a, w(1) = 1; \\ \xi(3) &= a, \eta(1) = -a, \zeta(1) = a, w(1) = 1; \\ \xi(4) &= a, \eta(1) = -a, \zeta(1) = -a, w(1) = 1; \\ \xi(5) &= -a, \eta(1) = a, \zeta(1) = a, w(1) = 1; \\ \xi(6) &= -a, \eta(1) = a, \zeta(1) = -a, w(1) = 1; \\ \xi(7) &= -a, \eta(1) = -a, \zeta(1) = a, w(1) = 1; \\ \xi(8) &= -a, \eta(1) = -a, \zeta(1) = -a, w(1) = 1; \end{aligned}$$

avec $a = 1./\sqrt{3}$.

Pour les autres nombres de points d'intégration : 1, $27 = 3 \times 3 \times 3$, $64 = 4 \times 4 \times 4$, on suit l'ordre des points d'intégration donnée en 1D (19.28). Le poids d'intégration est le produit des poids correspondants 1D selon les directions ξ , η et ζ

20.2 Position des points d'intégrations pour les éléments pentaédriques

La position des points d'intégrations de coordonnées (ξ, η, ζ) sur l'élément de référence, et la valeur des poids d'intégration sont calculées à partir de celles d'une part de l'élément de référence 2D triangulaire puis d'autre part de l'élément de référence 1D. ξ et η correspondent aux directions du triangle, et ζ celle du segment. Les numéros varient d'abord suivant les numéros du triangle selon (19.30) puis globalement suivant le segment selon (19.28). Par exemple si l'on a 3 points d'intégration dans le triangle et 2 dans l'épaisseur : la correspondance locale globale est donnée par la table (73).

TABLE 73 – exemple de correspondance de numéro de point d'intégration pour un pentaèdre, entre la numérotation globale et la numérotation du triangle et du segment associé

numéro global	numéro triangle	numéro segment
1	1	1
2	2	1
3	3	1
4	1	2
5	2	2
6	3	2

La position des points d'intégration selon ξ , η est donnée par la position dans le triangle (19.30). La position selon ζ est donnée par la position dans le segment (19.28).

Le poids d'intégration est le produit des poids correspondants 2D et 1D.

20.3 Position des points d'intégrations pour les éléments tétraédriques

TABLE 74 – Positions de points d'intégration pour les éléments tétraédriques.

```
//-----
// Points d'integration
//-----
// 1 point : (ordre 1)
//          Pt1 (1/4,1/4,1/4)
// 4 points : (ordre 2)  a = (5. - sqrt(5))/20., b = (5+3.*sqrt(5))/20.
//          Pt1 (a,a,a) ; Pt2 (a,a,b) ; Pt3 (a,b,a) ; Pt4 (b,a,a)
//
// 5 points : (ordre 3)  a = 1/4, b=1/6, c=1/2,
//          Pt1 (a,a,a) ; Pt2 (b,b,b) ; Pt3 (b,b,c) ; Pt4 (b,c,b) ; Pt4 (c,b,b);
//
// 15 points : (ordre 5)  a = 1/4, b1=(7+sqrt(15))/34,  b2=(7-sqrt(15))/34,
//                          c1=(13+3sqrt(15))/34, c2=(13-3sqrt(15))/34,
//                          d=(5-sqrt(15))/20,  e=(5+sqrt(15))/20,
//          Pt1 (a,a,a) ; Pt2 (b1,b1,b1) ; Pt3 (b2,b2,b2) ; Pt4 (b1,b1,c1)
//          Pt5 (b2,b2,c2) ; Pt6 (b1,c1,b1) ; Pt7 (b2,c2,b2) ; Pt8 (c1,b1,b1)
//          Pt9 (c2,b2,b2) ; Pt10 (d,d,e) ; Pt11 (d,e,d) ; Pt12 (e,d,d)
//          Pt13 (d,e,e) ; Pt14 (e,d,e) ; Pt15 (e,e,d) ;
```

21 Remarques sur le nombres de points d'intégration

Pour un certain nombre d'éléments, il est possible de choisir entre plusieurs nombres maxi de points d'intégration. Le temps de calcul est directement lié à ce nombre de point d'intégration. Ainsi si l'on veut diminuer le temps de calcul, on a tout intérêt à diminuer le nombre de points d'intégration (nbpti). Mais la précision du calcul dépend également explicitement de nbpti. Il y a donc nécessairement un compromis. En général, l'élément pas défaut est associé à un nbpti optimum, et en général, on n'a pas à ce soucier de changer ce nombre.

Cependant, le choix d'un nbpti peut également entraîner dans certaines circonstance, une singularité de la matrice de raideur, si celle-ci est calculée. Une méthode pour étudier la possibilité d'une singularité est par exemple en 3D est de comparer $a = nbpti * 6$ (représentant le rang possible de la matrice) avec le nombre de $nbdll_{libre} = ddl_{total} - 6$ (représentant le rang nécessaire de la matrice après suppression des mouvements solides). Si "a" est inférieur au nombre de ddl libre, alors il y a risque de singularité. Lorsque le niveau d'affichage est supérieur à 0, le programme calcul ces deux nombres, effectue le test et indique s'il y a un risque de singularité. Dans le cas où le risque existe, il est préférable d'augmenter le nombre de point d'intégration! ou en tout cas il faut être vigilant au déroulement du calcul.

22 Gestion des modes d'hourglass

Les éléments sous-intégrés (cf. 21), possèdent plusieurs avantages, par exemple : rapidité, réduction des phénomènes de blocage volumétrique. Cependant, ils présentent en général le défaut de permettre l'apparition de modes à énergie nulle (ou puissance nulle), nommées couramment "modes d'hourglass". Différentes techniques existent pour limiter et contrôler l'apparition de ces modes.

Une première technique, a priori peu courante, est implantée dans Herezh++. On associe à l'élément sous-intégré, un élément interne utilisant une intégration complète avec une loi différente (a priori plus simple) que celle de l'élément principal. Par exemple, supposons un maillage d'hexaèdres linéaires avec 1 point d'intégration et une loi de comportement complexe (élasto-plastique). La stabilisation consiste alors à associer une loi élastique (donc peu coûteuse en temps de calcul) avec une intégration complète.

La table (75) donne un exemple d'utilisation qui peut être placée au même niveau dans le fichier .info, que la définition des masses volumiques, ou épaisseurs, sections etc... On remarque tout d'abord le mot clé : "hourglass_gestion." puis sur la ligne active suivante : une référence d'éléments sur lesquels s'applique une gestion de modes d'hourglass, le mot clé "STABHOURGLASS_PAR_COMPORTEMENT" qui indique le type de stabilisation, puis un nom de référence de loi de comportement : ici "acier_mou". Un paramètre scalaire est également présent, il indique un facteur d'échelle "fac" qui agit sur la raideur et le second membre de stabilisation. Par exemple si fac=0.1, cela signifie que seulement 1/10 de la raideur et du second membre de stabilisation sont appliqués.

À noter également qu'il est possible d'accéder en sortie de résultat, à l'énergie totale d'hourglass générée dans le calcul, ainsi qu'à l'énergie d'hourglass générée pour chaque

TABLE 75 – Exemple de déclaration de contrôle d’hourglass à l’aide d’un comportement matériel simple, et d’un élément interne à intégration complète

```
hourglass_gestion_
#-----
#  ref  | type et parametre          |
#-----
# la référence des elements concernes, le type de stabilisation
# le nom de la loi associee,
# le coefficient modérateur: 0.5 -> echelle sur la raideur
# et le second membre de stabilisation
E_to      STABHOURGLASS_PAR_COMPORTEMENT  acier_mou  0.5
```

élément. Le contrôle de ces valeurs permet de limiter l’impact de la stabilisation d’hourglass.

Remarque : Pour l’instant les éléments biellettes, point et les éléments SFE n’intègrent pas de blocage d’hourglass.

23 Remarques sur la prise en compte de la variation d’épaisseur pour les éléments 2D

Dans le cas des éléments 2D (par exemple à découpage triangulaire), et pour des lois utilisant l’hypothèse de contraintes planes, l’épaisseur peut varier pendant le calcul. Sa mise à jour sur un incrément de temps, s’effectue en fonction de la valeur du coefficient de compressibilité, calculé dans la loi de comportement à $t + \delta t$, selon la formule :

$$\frac{V - V_t}{V} = \frac{S.h - S_t.h_t}{S.h} = \frac{\text{trace}(\boldsymbol{\sigma})}{3 K_t} \quad (25)$$

avec $S h$ la section et l’épaisseur finales (à $t + \delta t$), $S_t h_t$ la section et l’épaisseur au début du pas de temps. Cette formule constitue donc une linéarisation sur un pas de temps de la formule générale :

$$\frac{\dot{I}_\sigma}{3} = -\dot{P} = K_t \frac{\dot{V}}{V} = K_s \times \text{taux de variation relative de volume} \quad (26)$$

Comme toute linéarisation, dans le cas de forts changements d’épaisseurs, on peut observer une dépendance du résultat à la taille du pas de temps.

La variation d’épaisseur est prise en compte dans la vérification des équations d’équilibres.

On se reportera à la documentation théorique pour plus d’information sur ces calculs.

Dans le cas de déformation plane, normalement l’épaisseur reste constante.

24 Remarques sur la prise en compte de la variation de section pour les éléments 1D

Dans le cas des éléments 1D (par exemple les biellettes), la section peut varier pendant le calcul. Sa mise à jour sur un incrément de temps, s'effectue en fonction de la valeur du coefficient de compressibilité, calculé dans la loi de comportement à $t + \delta t$, selon la formule :

$$\frac{V - V_t}{V} = \frac{S.l - S_t.l_t}{S.h} = \frac{\text{trace}(\boldsymbol{\sigma})}{3 K_t} \quad (27)$$

avec $S l$ la section et longueur de la fibre moyenne finales (à $t + \delta t$) , $S_t l_t$ la section et la longueur de la fibre moyenne au début du pas de temps. Cette formule constitue donc une linéarisation sur un pas de temps de la formule générale :

$$\frac{\dot{\mathbf{I}}_{\sigma}}{3} = -\dot{P} = K_t \frac{\dot{V}}{V} = K_s \times \text{taux de variation relative de volume} \quad (28)$$

Comme toute linéarisation, dans le cas de forts changements de section, on peut observer une dépendance du résultat à la taille du pas de temps.

La variation de section est prise en compte dans la vérification des équations d'équilibres. On se reportera à la documentation théorique pour plus d'information sur ces calculs.

Cinquième partie
Courbes

25 Introduction et utilisation de courbe 1D : “liste de courbes 1D”

Les courbes 1D sont utilisées par exemple dans les chargements en forces ou en déplacements ou encore dans la définition des lois de comportements. De manière à pouvoir utiliser une même courbe pour différentes entrées de données, il est nécessaire de lui associer un nom, c’est l’objet de ce paragraphe.

La définition d’une liste de courbe est facultative. Le mot clé à activer est : `les_courbes_1D`. Ensuite les différentes courbes sont décrites successivement, avec avant chaque courbe un nom de baptême (ou identificateur).

La table (76) donne un exemple de liste de courbes. Dans cet exemple le nom de la première courbe est “courbe1”, il s’agit d’une courbe poly-linéaires.

On se reportera au chapitre (48.1) pour une description exhaustive des différentes courbes.

TABLE 76 – Exemple de déclaration d’une liste de courbes 1D.

```
les_courbes_1D -----
#-----

courbe1  COURBEPOLYLINEAIRE_1_D
  Debut_des_coordonnees_des_points
  Coordonnee dim= 2 0. 100.
  Coordonnee dim= 2 1. 200.
  Fin_des_coordonnees_des_points

courbe34 COURBE_EXPOAFF
# def des coeff de la courbe expoaff
gamma= 10. alpha= -2. n= 1.3
```

Sixième partie

Lois de comportement

26 Lois de comportement : généralités

La définition des lois de comportement s'effectue en deux temps.

Tout d'abord on indique le nom de la loi utilisé. Ce nom de loi, qui peut être quelconque, est associé à une région de matière. Le ou les solides peuvent être décomposés en plusieurs régions. Dans le cas simple général où il n'y a qu'un solide composé d'une seule matière, on utilise une seule région. Le principe de repérage des régions est l'utilisation de liste d'éléments. L'exemple de la table (77) indique la syntaxe dans le cas d'un maillage :

TABLE 77 – Exemple de déclaration d'un nom de loi de comportement associée à une référence, ceci dans le cas d'un maillage.

```
choix_materiaux
#-----
# Elements      |      Matériau      |
#-----
ELEMENTS1      MATE1
ELEMENTS2      MATE2
```

Dans le cas de plusieurs maillages, il faut indiquer le nom du maillage associé à la référence. L'exemple de la table (78) indique la syntaxe dans le cas de plusieurs maillages.

TABLE 78 – Exemple de déclaration d'un nom de loi de comportement associée à une référence, ceci dans le cas de plusieurs maillages.

```
choix_materiaux
#-----
# nom de maillage |Elements      |      Matériau      |
#-----
nom_mail= piece  ELEMENTS1      MATE1
nom_mail= outil  ELEMENTS2      MATE2
```

La séquence débute par le mot clé "choix_matériaux", puis sur les lignes suivantes on trouve "ELEMENTS1" qui représente le nom d'une première référence d'éléments, "MATE1" qui représente le nom du matériaux associé aux éléments contenus dans la liste "ELEMENTS1". La ligne suivante définit une seconde région associée à un second matériaux.

Le second temps de la définition des lois de comportements consiste à définir effectivement les lois de comportement associés aux matériaux introduit précédemment. Le tableau (79) présente un exemple de déclaration de loi de comportement.

La séquence commence par le mot clé "materiaux" qui indique que l'on va définir les lois de comportement. Ensuite on associe au nom de matériau ici "MATE1" une loi de comportement repéré par un identificateur ici "ISOELAS", ce qui correspond à une loi 3D, isotrope élastique. Sur la ligne suivant on trouve alors les coefficients de la loi : E et

TABLE 79 – Exemple de déclaration de loi de comportement associée à deux matériaux.

```

matériaux
#-----
# Nom Matériau | Type loi | Potentiel |
#-----
      MATE1      ISOELAS

#-----
#      E      |      nu      |
#-----
      10000.      0.

#-----
# Nom Matériau | Type loi | Potentiel |
#-----
      MATE2      ISOELAS

#-----
#      E      |      nu      |
#-----
      20000.      0.3

```

nu. La syntaxe des coefficients dépend de la loi que l'on choisit. Nous allons donc passer en revue différentes lois actuellement disponibles.

Actuellement trois catégories de loi de comportement sont disponibles : les lois mécaniques et/ou thermo-mécaniques, les lois thermo-physiques, les lois de frottements dans le cas d'un contact avec frottement. Les premières sont relatives au comportement mécaniques de la pièce ou structure, les secondes sont relatives aux comportements thermo-physiques, on entend par là l'évolution des caractéristiques thermo-physiques : dilatation, conductivité, capacité calorifique ... les dernières sont associées au contact.

Il n'est pas possible d'appliquer deux lois d'une même catégorie sur une même référence, par contre il est possible d'avoir une loi de chaque catégorie sur une même référence d'éléments. Les deux lois doivent cependant avoir un nom différent. Par exemple si l'on veut travailler sur de l'acier, en tenant compte de la dilatation. Il est nécessaire d'avoir une loi de mécanique ou thermo-mécanique et une loi de thermo-physique. La table (81) présente un exemple complet de déclaration permettant le calcul de l'élongation.

27 Type de déformation

Par défaut le type de déformation utilisé dans Herezh++ est la déformation d'Almansi.

$$\varepsilon = \frac{1}{2}(\hat{g}^{ij} - g^{ij})\hat{g}^i \otimes \hat{g}^j$$

. En 1D, on obtient dans le repère global :

$$\varepsilon_{11} = \frac{1}{2} \left(1 - \frac{(l_0)^2}{l^2} \right)$$

On observe que cette mesure est simple, mais qu'elle n'est pas symétrique par rapport à l'élongation $\Delta l/l_0$, en particulier pour une élongation positive infinie, la mesure tend vers 0.5 et pour une élongation qui tend vers $-l_0/l_0$ c'est-à-dire pour une longueur l s'annulant on tend vers $-\infty$.

Il est possible d'utiliser la mesure de déformation logarithmique, qui est plus complexe à calculer, mais qui possède certains avantages. Pour une sollicitation 1D, en traction ou compression, les limites sont ici ∞ et $-\infty$, et à tout instant, la trace du tenseur de déformation est égale à la variation relative de volume (contrairement aux autres mesures). Par contre les temps de calcul induits sont plus importants.

Pour changer le type de déformation employée, on indique à la suite de la loi de comportement un mot clé "type_de_deformation" suivi du type de déformation, qui par défaut est "DEFORMATION_STANDARD" c'est-à-dire la déformation d'Almansi.

La table (80) donne un exemple d'utilisation de déformation logarithmique.

TABLE 80 – Exemple de déclaration d'une loi de comportement élastique associée à une déformation logarithmique.

```

acier          ISOELAS1D
#-----
#|              E              |              NU              |
#-----
                200000.          0.3
#  -- definition du type de deformation (par default: DEFORMATION_STANDARD) --
           type_de_deformation  DEFORMATION_LOGARITHMIQUE

```

27.1 Exemple de déclaration pour intégrer la dilatation thermique dans un calcul mécanique

La table (81) donne un exemple complet pour intégrer la dilatation thermique dans un calcul mécanique.

TABLE 81 – Exemple de déclaration d’une loi de comportement mécanique associée à une loi de comportement thermo-physique en vue d’intégrer la dilatation thermique dans le calcul mécanique.

```

#----- fin des courbes 1D -----

          choix_materiaux -----
#-----
# ref d'Elements | Materiau      |
#-----
          E_to      acier_meca
          E_to      acier_therm2

          materiaux -----

#----- debut cas d'une loi isoelas acier -----
          acier_meca      ISOELAS1D
#-----
#|      E      |      NU      |
#-----
          200000.      0.3
#----- fin cas d'une loi isoelas acier -----

#----- debut cas d'une loi thermo physique acier -----
          acier_therm      LOI_ISO_THERMO

# ..... loi de comportement thermique isotrope .....
# | coefficient de dilatation | conductivite | capacite calorifique |
# |      alphaT /degr      | lambda W/mm.deg |      cp J/gr.deg      |
# .....
          alphaT= 12.3e-6      lambda= 0.062      cp= 0.465
          fin_thermique_isotrope

#----- fin cas d'une loi thermo physique acier -----

#----- debut cas d'une loi thermo physique acier -----
          acier_therm2      LOI_ISO_THERMO

# ..... loi de comportement thermique isotrope .....
# | coefficient de dilatation | conductivite | capacite calorifique |
# |      alphaT /degr      | lambda W/mm.deg |      cp J/gr.deg      |
# .....
          alphaT= alphaT_thermo_dependant_      courbealpha
          lambda= lambda_thermo_dependant_      courbelambda
          cp= cp_thermo_dependant_      courbecp
          fin_thermique_isotrope

#----- fin cas d'une loi thermo physique acier -----

          masse_volumique -----
#-----
# RHO |
#-----
          E_to      8.e-9

          sections -----
          E_to      2.

          dilatation_thermique -----
#-----
#| ref element | oui ou non on veut la dilatation |
#-----
          E_to      1.
# definition du chargement

```

28 Liste de lois mécaniques et thermo-mécaniques disponibles

TABLE 82 – liste des différents lois

identificateur	indications	ref
	élastique isotrope	
ISOELAS1D	1D : Hooke	(cf.84)
ISOELAS2D_D	2D : Hooke déformations planes	(cf.84)
ISOELAS2D_C	2D : Hooke contraintes planes	(cf.84)
ISOELAS	3D : Hooke	(cf.84)
	élastique isotrope non-linéaire	
ISO_ELAS_ESPO1D	1D : $\sigma = E(\varepsilon)\varepsilon = f(\varepsilon)E\varepsilon$	(cf.88)
ISO_ELAS_ESPO3D	3D : $\sigma = E(\varepsilon)\varepsilon = f(\gamma)E\varepsilon$ avec $\gamma = \sqrt{2./3.\epsilon : \epsilon}$	(cf.88)
ISO_ELAS_SE1D	1D : $\sigma = f(\epsilon)$	(cf.88)
	hyperélastique isotrope	
ISOHYPER3DFAVIER3	3D : potentiel proposé par Denis Favier	(cf.93)
ISOHYPERBULK3	3D : potentiel pour la partie volumique seuil	(cf.28.3.2)
MOONEY_RIVLIN_1D	1D : loi classique de Mooney Rivlin	(cf.93)
ISOHYPER3DORGEAS1	3D : potentiel proposé par Laurent Orgéas	(cf.93)
ISOHYPER3DORGEAS2	3D : idem ISOHYPER3DORGEAS1 + amélioration des fonctions de dépendance à la phase	(cf.28.3.6)
POLY_HYPER3D	3D : loi polynomiale en invariants de même type que ceux de Mooney Rivlin	(cf.93)
	élastoplastique isotrope	
PRANDTL_REUSS1D	1D : Prandtl Reuss grandes déformations	(cf.114)
PRANDTL_REUSS2D_D	2D : Prandtl Reuss grandes déformations déformations planes	(cf.114)
PRANDTL_REUSS	3D : Prandtl Reuss grandes déformations	(cf.114)
	visco-élastique isotrope	
NEWTON1D	1D : $\sigma = \mu \mathbf{D}$ ou $\sigma = \mu(\mathbf{D} : \mathbf{D})^{n/2} \mathbf{D}$	(cf.116)
NEWTON2D_D	2D : déformation plane $\sigma = \mu \mathbf{D}$ ou $\sigma = \mu(\mathbf{D} : \mathbf{D})^{n/2} \mathbf{D}$	(cf.116)
NEWTON3D	3D : $\sigma = \mu \mathbf{D}$ ou $\sigma = \mu(\mathbf{D} : \mathbf{D})^{n/2} \mathbf{D}$	(cf.116)
MAXWELL1D	1D : un ressort et un amortisseur en série	(cf.116)
MAXWELL3D	3D : un ressort et un amortisseur en série	(cf.116)
	composition de lois élémentaires	
LOI_ADDITIVE_EN_SIGMA	1D, 2D, 3D : $\sigma = \sum \sigma_{(i)}$	(cf.128)
LOI_DES_MELANGES_EN_SIGMA	1D, 2D, 3D : $\sigma = (\alpha)\sigma_{(1)} + (\alpha - 1)\sigma_{(2)}$	(cf.128)
	lois d'élasto-hystérésis	
HYSTERESIS_1D	1D : loi d'hystérésis classique (modèle de Guélin-Favier-Pégon)	(cf.137)

TABLE 83 – suite de la liste des différents lois

identificateur	indications	ref du commentaire
	passage 3D 2D d'une loi quelconque	
LOI_DEFORMATIONS_PLANES	loi 3D transformée en déformations planes	(cf.28.7.1)
LOI_CONTRAINTE_PLANES	loi 3D transformée en contraintes planes	(cf.28.7.2)
	lois hypo-élastiques	
HYPO_ELAS3D	3D : loi hypo-élastique $\dot{\mathbf{S}} = \mu \bar{\mathbf{D}}$ et $\dot{I}_\sigma = K_c I_{\mathbf{D}}$	(cf.147)
HYPO_ELAS2D_C	2D : loi hypo-élastique contrainte plane $\dot{\mathbf{S}} = \mu \bar{\mathbf{D}}$ et $\dot{I}_\sigma = K_c I_{\mathbf{D}}$	(cf.147)
LOI_RIEN1D	1D : loi qui ne fait rien mécaniquement	(cf.28.10)
LOI_RIEN2D	2D : loi qui ne fait rien mécaniquement	(cf.28.10)
LOI_RIEN3D	3D : loi qui ne fait rien mécaniquement	(cf.28.10)

28.1 Lois iso-élastiques linéaires

Les lois isotropes élastiques linéaires de type Hooke, disponibles sont données dans la table (cf.84).

TABLE 84 – liste des différents lois isotropes élastiques disponibles

identificateur	indications	ref du commentaire
ISOELAS1D	élastique isotrope 1D	(28.1.1)
ISOELAS2D_D	élastique isotrope 2D déformations planes	(28.1.2)
ISOELAS2D_C	élastique isotrope 2D contraintes planes	(28.1.3)
ISOELAS	élastique isotrope 3D	(28.1.4)

Il s'agit de la loi classique élastique de Hooke qui suppose une relation linéaire entre contraintes et déformations. La loi peut se représenter par exemple à l'aide de deux coefficients de proportionnalité :

$$-P = \text{trace}(\sigma) = K \frac{\text{trace}(\varepsilon)}{3} \quad \text{et} \quad \mathbf{S} = 2 G \bar{\varepsilon} \quad (29)$$

où K est le module de compressibilité et G le module de cisaillement. L'expression $\frac{\text{trace}(\varepsilon)}{3}$ est sensée représenter la variation relative volumique $\frac{\Delta V}{V}$. Ceci est exacte dans le cas de l'utilisation de la mesure de déformation logarithmique, et est également une bonne approximation de la réalité dans le cas des petites déformations. Par contre dans le cas des grandes déformations avec une mesure d'Almansi (mesure par défaut dans Herezh++) ou la mesure classique de Green Lagrange, l'expression représente grossièrement la variation relative de volume. Dans ce dernier contexte, reste utilisable, mais la signification des coefficients change, en particulier le coefficient K ne représente plus un module de compressibilité.

On retiendra donc que les résultats dépendent du type de mesure de déformation utilisé, dans le cas des grandes déformations (ou déformations finies).

On a :

$$K = \frac{E}{3(1-2\nu)} \quad \text{et} \quad G = \frac{E}{2(1+\nu)} \quad (30)$$

avec E et ν le module d'Young et le coefficient de Poisson.

La contrainte peu également est calculée via l'expression équivalente suivante :

$$\boldsymbol{\sigma} = \frac{(E \nu)}{((1. - 2.\nu) (1 + \nu))} \mathbf{I}_\varepsilon + \frac{E}{(1 + \nu)} \varepsilon \quad (31)$$

28.1.1 ISOELAS1D

ISOELAS1D identificateur d'une loi 1D isotrope de type Hooke (cf.28.1) . Cette loi convient pour les éléments 1D, de type poutre par exemple. Elle nécessite la donnée de deux paramètres, le module d'Young E et le coefficient de poisson nécessaire pour tenir compte de la variation de la section (voir doc théorique pour la méthode utilisée dans

Herezh++). Il est également possible de définir un module d'Young qui dépend de la température. Dans ce cas il faut s'assurer que la température est définie aux noeuds soit en tant que donnée, soit en tant que variable. La table (85) donne un exemple de déclaration de loi thermo-dépendante avec une courbe $E=f(T)$ explicitement défini.

TABLE 85 – Exemple de déclaration de la loi élastique 1D dont le module d'Young dépend de la température selon une courbe indiquée explicitement.

```
#----- debut cas d'une loi isoelas acier thermodependante -----
acier      ISOELAS1D
#-----
#|          E          |
#-----
      thermo_dependant_  CPL1D Dd1P 0. 100000. 100. 50000.0 Fd1P 0.3
#----- fin cas d'une loi isoelas acier -----
```

A la place de la valeur de E il y a le mot clé : "thermo_dependant_" et ensuite dans l'exemple il y a la définition d'une courbe simplifiée poly-linéaire (cf. § courbes). Au lieu de définir la courbe il est également possible d'indiquer simplement le nom d'une courbe précédemment défini. La table (86) donne un exemple de déclaration de loi thermo-dépendante avec une courbe $E=f(T)$ repéré par un nom de référence.

TABLE 86 – Exemple de déclaration de la loi élastique 1D dont le module d'Young dépend de la température selon une courbe repérée par un nom de référence.

```
#----- debut cas d'une loi isoelas acier thermodependante -----
acier      ISOELAS1D
#-----
#|          E          |
#-----
      thermo_dependant_  courbe1 0.3
#----- fin cas d'une loi isoelas acier -----
```

28.1.2 ISOELAS2D_D

Identificateur d'une loi 2D isotrope en déformation plane, c'est-à-dire le cas où les déformations suivant 3 sont nulles. Dans ce cas, en général les contraintes suivant cette direction sont non nulles.

Le tenseur de déformation étant entièrement connu, les contraintes s'obtiennent directement en utilisant la relation (31) vrai quelque soit l'état élastique dans le cas du modèle de Hooke.

L'axe 3 ici est l'axe local, c'est-à-dire perpendiculaire à l'élément 2D. Cette loi convient pour des éléments 2D : quadrangles, triangles. La loi nécessite la donnée de deux paramètres comme ISOELAS : le module d'Young et le coefficient de poisson. Comme dans le cas de la loi 1D, le module d'Young peut-être thermodépendant. Dans ce cas la lecture des données suit la même syntaxe que dans le cas 1D, avec après la courbe, la définition du coefficient de Poisson.

28.1.3 ISOELAS2D_C

Identificateur d'une loi 2D isotrope en contrainte plane, c'est-à-dire le cas où les contraintes suivant 3 sont nulles.

La relation (31) vraie quelque soit l'état élastique dans le cas du modèle de Hooke, permet le calcul de la déformation suivant l'axe 3, compte tenu de la nullité de la contrainte, selon par exemple en mixte :

$$\varepsilon_3^3 = \frac{-\nu}{1-\nu} (\varepsilon_1^1 + \varepsilon_2^2) \quad (32)$$

La trace de ε s'en déduit. Les termes σ_β^α , ($\alpha, \beta = 1, 2$) s'obtiennent en utilisant de nouveau la relation (31).

Comme pour la loi ISOELAS2D_D, l'axe 3 est l'axe local normal à l'élément et la loi convient pour des éléments 2D : quadrangles et triangles. La loi nécessite la donnée de deux paramètres : le module d'Young et le coefficient de poisson. Comme dans le cas de la loi 1D, le module d'Young peut-être thermodépendant. Dans ce cas la lecture des données suit la même syntaxe que dans le cas 1D, avec après la courbe, la définition du coefficient de Poisson.

Dans le cas de l'utilisation de la loi avec des éléments 2D (plaques, coques) l'épaisseur de l'élément varie, et est mise à jour dans le calcul, en particulier l'équilibre mécanique tiens compte de la variation de l'épaisseur.

28.1.4 ISOELAS

Identificateur d'une loi élastique 3D isotrope. Cette loi convient par exemple pour les éléments volumiques : hexaèdre, tétraèdre, pentaèdre, par contre ne peut pas être utilisée pour des éléments plaques ou des éléments linéaires. Elle nécessite la donnée d'un module d'young et du coefficient de poisson (exemple : cf. 79). Comme dans le cas de la loi 1D, le module d'Young peut-être thermo-dépendant. Dans ce cas la lecture des données suit la même syntaxe que dans le cas 1D, avec après la courbe, la définition du coefficient de poisson. Il est possible également de n'utiliser que la partie sphérique de la loi ou la partie déviatorique. Pour cela, on indique après le coefficient de Poisson, les mots clés : "seule_spherique" ou "seule_deviatorique". La table (87) donne un exemple de déclaration de loi.

Par défaut, comme il a été précisé au chapitre (27) la déformation utilisée est la mesure d'Almansi. Il est possible également d'utiliser la mesure de déformation logarithmique (cf.27). Dans tous les cas, en général la loi de Hooke n'est valable que pour de faibles déformations, quelques % au maximum, au delà il est préférable d'utiliser des lois hyperélastiques par exemple.

TABLE 87 – Exemple de déclaration de la loi élastique 3D dont le module d’Young dépend de la température selon une courbe repérée par un nom de référence.

```
#----- debut cas d'une loi isoelas acier thermodependante -----
      acier      ISOELAS
# ..... loi de comportement isoelastique 3D themodependante.....
#:  definition de la courbe donnant l'evolution du module d'young en fonction de la temperature :
#:  suivi de la definition du coefficient de poisson (independant de la temperature)           :
#:.....:
      thermo_dependant_      courbe1      0.3      seule_deviatorique
# NB: courbe1 est le nom d'une courbe deja defini, on peut egalement definir directement une
# nouvelle courbe apres le mot cle thermo_dependant_ puis la courbe sans nom de reference.
# Il est possible d'indiquer que l'on souhaite calculer seulement la partie spherique de la loi
# pour cela on met le mot cle: seule_spherique a la fin des donnees sur la meme ligne
# D'une maniere identique il est possible d'indiquer que l'on souhaite calculer seulement la partie
# la partie deviatorique de la loi, pour cela on met le mot cle: seule_deviatorique
#----- fin cas d'une loi isoelas acier -----
```

28.2 Lois isotropes élastiques non-linéaires

Les lois isotropes élastiques non linéaires disponibles sont données dans la table (cf.88). Ces lois sont toutes construites à partir du modèle de Hooke, en introduisant une non-linéarité simple au niveau de l'évolution tangente (par exemple à travers un module d'Young dépendant de la déformation). Ce sont donc des lois principalement phénoménologiques de caractérisation très simple.

TABLE 88 – liste des différents lois isotropes élastiques non linéaires disponibles

identificateur	indications	ref du commentaire
ISO_ELAS_ESPO1D	1D : $\sigma = E(\varepsilon)\varepsilon = f(\varepsilon)E\varepsilon$	(28.2.1)
ISO_ELAS_ESPO3D	3D : $\sigma = E(\varepsilon)\varepsilon = f(\gamma)E\varepsilon$ avec $\gamma = \sqrt{2./3.\epsilon : \epsilon}$	(28.2.2)
ISO_ELAS_SE1D	1D : $\sigma = f(\epsilon)$	(28.2.3)

28.2.1 ISO_ELAS_ESPO1D

Identificateur d'une loi 1D isotrope élastique non linéaire. Cette loi constitue une extension de la loi classique d'Hooke pondéré par une fonction multiplicative qui dépend de la valeur absolue de la déformation. La loi est donc du type : $\sigma = E(\varepsilon)\varepsilon = f(|\varepsilon|)E\varepsilon$. La fonction f est générale, elle est choisie parmi la liste des fonctions 1D disponible (cf. 220).

La loi nécessite donc la donnée du module d'young et du coefficient de poisson (ce dernier n'est pas utilisé, mais cette présentation permet d'être compatible avec les dimensions 2 et 3), et d'une fonction multiplicative $f(x)$: La table (89) donne un exemple de déclaration.

TABLE 89 – Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_ESPO1D .

```
#-----
# Nom Materiau | Type loi      |
#-----
#          MATE3          ISO_ELAS_ESPO1D
#-----
#      E          |      nu          |
#-----
E=      10000.          nu=      0.3

# lecture du type de courbe multiplicative
f_coefficient COURBE_EXPOAFF
# def des coeff de la courbe de type expoaff
gamma= 1. alpha= -20. n= 1.01
```

28.2.2 ISO_ELAS_ESPO3D

Identificateur d'une loi 3D isotrope élastique non linéaire. La loi est identique au cas 1D pour l'entrée des données. Ainsi comme en 1D, cette loi constitue une extension de la loi classique d'Hooke pondéré par une fonction multiplicative. Par contre la fonction multiplicative dépend du deuxième invariant de la déformation sous la forme du paramètre : $\gamma = \sqrt{2./3.}\epsilon : \epsilon$. La loi est donc du type : $\sigma = E(\epsilon)\epsilon = f(\gamma)E\epsilon$. La fonction f est générale, elle est choisie parmi la liste des fonctions 1D disponible (cf. 220).

La loi nécessite donc la donnée du module d'young et du coefficient de poisson, et d'une fonction multiplicative $f(x)$. L'entrée des données est identique au cas 1D sauf pour le nom de la loi. La table (90) donne un exemple de déclaration.

TABLE 90 – Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_ESPO3D .

```
#-----
# Nom Materiau | Type loi      |
#-----
      MATE3      ISO_ELAS_ESPO3D
#-----
#      E          |      nu        |
#-----
E=    10000.      nu=    0.3

# lecture du type de courbe multiplicative
f_coefficient courbe1
```

28.2.3 ISO_ELAS_SE1D

Identificateur d'une loi 1D isotrope élastique non linéaire. Cette loi est plus simple que la loi "ISO_ELAS_ESPO1D". Il s'agit de définir directement une relation $\sigma = f(\epsilon)$. La fonction f est générale, elle est choisie parmi la liste des fonctions 1D disponible (cf. 220). La loi peut-être symétrique par rapport à $\epsilon = 0$. ou avoir un comportement différent en traction et en compression. Par défaut le comportement est symétrique. La présence du mot clé "non_symetrique" permet de spécifier que le comportement est différent en traction et compression. Dans ce dernier cas, il faut donner une courbe ayant une partie négative et une partie positive.

La loi nécessite donc uniquement la donnée d'une fonction f(x). Les tables (91) et (92) donnent des exemples de déclaration.

TABLE 91 – Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_SE1D, ceci dans le cas d'un comportement symétrique et de l'utilisation d'une courbe déjà existante.

```
#-----
# Nom Materiau | Type loi |
#-----
acier2          ISO_ELAS_SE1D
#-----
#|..... loi de comportement isoelastique non lineaire 1D de type sigma = f(epsilon).....|"
#|          .. definition de la courbe f() ..                                     |"
#-----
          f_coefficient  courbe_f_epsilon

#          .. fin de la definition de la courbe f(epsilon)..
```

TABLE 92 – Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_SE1D, cas d'un comportement non symétrique et définition directe de la courbe d'évolution .

```
# -----"
# |..... loi de comportement isoelastique non lineaire 1D de type sigma = f(epsilon).....|"
# |          .. cas non symetrique definition de la courbe f() ..                                     |"
# -----"
non_symetrique f_coefficient COURBEPOLYLINEAIRE_1_D"
  Debut_des_coordonnees_des_points"
    Coordonnee dim= 2 -0.03 -350.
    Coordonnee dim= 2 -0.02 -300.
    Coordonnee dim= 2 -0.01 -200.
    Coordonnee dim= 2 0. 0.
    Coordonnee dim= 2 0.05 200.
    Coordonnee dim= 2 0.1 300.
    Coordonnee dim= 2 0.15 350.
  Fin_des_coordonnees_des_points "
# .. fin de la definition de la courbe sigma = f(epsilon).. \n" << endl;
```


28.3 Lois Hyper-élastiques

Les lois isotropes hyper-élastiques disponibles sont données dans la table (cf.93).

TABLE 93 – liste des différents lois isotropes hyper-élastiques disponibles

identificateur	indications	ref du commentaire
ISOHYPER3DFAVIER3	3D : potentiel proposé par Denis Favier	(28.3.1)
ISOHYPERBULK3	3D : potentiel pour la partie volumique seuil	(cf.28.3.2)
MOONEY_RIVLIN_1D	1D : loi classique de Mooney Rivlin	(28.3.3)
MOONEY_RIVLIN_3D	3D : loi classique de Mooney Rivlin	(28.3.4)
ISOHYPER3DORGEAS1	3D : potentiel proposé par Laurent Orgéas	(28.3.5)
ISOHYPER3DORGEAS2	3D : idem ISOHYPER3DORGEAS1 + amélioration des fonctions de dépendance à la phase	(cf.28.3.6)
POLY_HYPER3D	3D : loi polynomiale en invariants de même type que ceux de Mooney Rivlin	(28.3.7)
HART_SMITH3D	3D : loi de Hart Smith en 3D	(28.3.8)

28.3.1 ISOHYPER3DFAVIER3

Identificateur d'une loi 3D isotrope hyperélastique. Le potentiel est défini par l'expression suivante :

$$w = \frac{K}{6} \ln^2(V) + \frac{Q_{or}^2}{2\mu_0} \ln \left(\cosh \left(\frac{2\mu_0 Q_\varepsilon}{Q_{or}} \right) \right) + \mu_\infty Q_\varepsilon^2 \quad (33)$$

La loi nécessite la donnée de 4 paramètres dans le cas d'une utilisation sans phase :

- le coefficient de compressibilité volumique : K (équivalent au cas de Hooke à : $E/(1 - 2\nu)$)
- le seuil : Q_{or}
- la pente à l'infini : $\mu_{o\infty}$
- la pente à l'origine - celle de l'infini : μ_0

La table (94) donne un exemple de déclaration.

Il est également possible d'introduire l'influence de la phase, selon la formule suivante :

$$\begin{aligned} Q_r &= \frac{Q_{or}}{(1 + \gamma_Q \cos(3\varphi_\varepsilon))^{n_Q}} \\ \mu_\infty &= \frac{\mu_{o\infty}}{(1 + \gamma_\mu \cos(3\varphi_\varepsilon))^{n_\mu}} \end{aligned} \quad (34)$$

Dans ce cas il faut spécifier sur la fin de la ligne des premiers paramètres le mot clé : "avec_phase" et ensuite sur la ligne suivante 4 paramètres supplémentaires :

- deux coefficients pour la variation du seuil : n_Q et γ_Q
- deux coefficients pour la variation de la pente à l'infini : n_μ et γ_μ

La table (95) donne un exemple de déclaration.

TABLE 94 – Exemple de déclaration de la loi hyperélastique ISOHYPER3DFAVIER3 sans phase.

```

#-----
# Nom Materiau | Type loi      | Potentiel    |
#-----
      MATE3      ISOHYPER3DFAVIER3

#-----
#      K      |      Qor |      mur |      mu_inf |
#-----
      270000      400      28000      10000

```

TABLE 95 – Exemple de déclaration de la loi hyperélastique ISOHYPER3DFAVIER3 avec phase.

```

#-----
# Nom Materiau | Type loi      | Potentiel    |
#-----
      MATE3      ISOHYPER3DFAVIER3

#-----
#      K      |      Qor |      mur |      mu_inf |
#-----
      270000      400      28000      10000      avec_phase

#-----
#      n_Q      |      gamma_Q |      n_mu |      gamma_mu |
#-----
      0.25      0.4      0.25      0.4

```

28.3.2 ISOHYPERBULK3

Identificateur d'une loi 3D isotrope hyperélastique. Le potentiel est défini par l'expression suivante :

$$w = \frac{K}{6} \ln^2(V) \quad (35)$$

La loi nécessite la donnée de 1 paramètres, le coefficient de compressibilité volumique : K, équivalent au cas de Hooke à : $E/(1 - 2\nu)$

La table (96) donne un exemple de déclaration. Cette loi correspond à la partie volumique de la loi de Favier. Elle permet, conjointement à de l'hystérésis, de modéliser un comportement complet intégrant l'hystérésis (qui ne modélise que la partie déviatoire de

la contrainte).

TABLE 96 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 .

```
#-----
# Nom Materiau | Type loi      | Potentiel    |
#-----
      MATE3      ISOHYPERBULK3

#-----
#      K      |
#-----
      270000
```

28.3.3 MOONEY_RIVLIN_1D

Identificateur d'une loi 1D isotrope de type Mooney Rivlin hyperélastique. Le potentiel est défini par l'expression suivante :

$$\sigma_1^1 = 2.C_{10} \left(\frac{1.}{(1. - 2.\epsilon_1^1)} - \sqrt{1. - 2.\epsilon_1^1} \right) + 2.C_{01} \frac{1.}{\sqrt{1. - 2.\epsilon_1^1}} \quad (36)$$

où les deux coefficients matériaux sont C_{10} et C_{01} . La contrainte est celle de Cauchy et la déformation est celle d'Almansi. La table (97) donne un exemple de déclaration.

TABLE 97 – Exemple de déclaration de la loi de Mooney Rivlin en 1D.

```
#-----
# Nom Materiau | Type loi      |
#-----
      elastomere      MOONEY_RIVLIN_1D

# -----
# |... loi de comportement hyper elastique 1D de type mooney rivlin ...|
# |                               .. deux coefficients C01 et C10 ..      |
# -----
#
#      C01= 0.0167      C10= 0.145
#      .. fin de la definition de la loi mooney rivlin
```

Il est également possible de définir des paramètres thermo-dépendants (un ou les deux paramètres). La table (98) donne un exemple de déclaration dans le cas où les deux

paramètres sont thermo-dépendants, la table (99) donne un exemple lorsque seul le second paramètre est thermo-dépendant.

TABLE 98 – Exemple de déclaration de la loi de Mooney Rivlin en 1D, avec les deux paramètres thermo-dépendants.

```
#-----
# Nom Materiau | Type loi          |
#-----
    elastomere      MOONEY_RIVLIN_1D

# -----
# |... loi de comportement hyper elastique 1D de type mooney rivlin ...|
# |                               .. deux coefficients C01 et C10 ..      |
# -----
#
    C01= C01_thermo_dependant_ courbe1
    C10= C10_temperature_ courbe2
# .. fin de la definition de la loi mooney rivlin
```

TABLE 99 – Exemple de déclaration de la loi de Mooney Rivlin en 1D, avec le second paramètre thermo-dépendant, le premier étant fixe.

```
#-----
# Nom Materiau | Type loi          |
#-----
    elastomere      MOONEY_RIVLIN_1D

# -----
# |... loi de comportement hyper elastique 1D de type mooney rivlin ...|
# |                               .. deux coefficients C01 et C10 ..      |
# -----
#
    C01= 0.0167 C10= C10_temperature_ courbe2
# .. fin de la definition de la loi mooney rivlin
# noter qu'apres la definition de chaque courbe, on change de ligne, a l'inverse
# si la valeur du parametre est fixe, on poursuit sur la meme ligne.
```

28.3.4 MOONEY_RIVLIN_3D

Identificateur d'une loi 3D isotrope de type Mooney Rivlin hyperélastique. Le potentiel est défini par l'expression suivante :

$$W_{Mooney-Rivlin} = (C_{10} (J_1 - 3) + C_{01} (J_2 - 3)) + \left(K \left[1 - \frac{1 + \ln(\sqrt{I_3})}{\sqrt{I_3}} \right] \right) \quad (37)$$

où les trois coefficients matériaux sont C_{10} , C_{01} et K . La contrainte est celle de Cauchy et la déformation de travail est celle d'Almansi. Cependant il faut noter que cette déformation, ici n'intervient pas directement, le potentiel étant défini à partir des élongations, eux même calculées à partir du tenseur de Cauchy-Green droit : $\mathbf{C} = {}^t_{0..} \mathbf{G} = \hat{g}_{ij} \bar{g}^i \otimes \bar{g}^j$.

La table (100) donne un exemple de déclaration.

TABLE 100 – Exemple de déclaration de la loi de Mooney Rivlin en 3D.

```
#-----
# Nom Materiau | Type loi          |
#-----
           polymere  MOONEY_RIVLIN_3D
# -----
# |... loi de comportement hyper elastique 3D de type mooney rivlin ....|
# |                               .. trois coefficients C01 et C10 ..      |
# -----
C01= 0.0167    C10= 0.145  K= 3000
# .. fin de la definition de la loi money rivlin
```

Comme dans le cas 1D, les coefficients matériels peuvent être thermo-dépendants. La table (101) donne un exemple de déclaration. On se référera au cas 1D pour une déclaration de certains paramètres thermo-dépendants et les autres fixes.

On observe que le potentiel est composé d'une partie relative au changement de forme et une partie relative au changement de volume. Il est possible de changer la partie relative au changement de volume. Quatre cas sont disponible :

$$\begin{aligned} W_{v1} &= K \left[1 - \frac{1 + \ln(\sqrt{I_3})}{\sqrt{I_3}} \right] \\ W_{v2} &= \frac{K}{2} (V - 1) \\ W_{v3} &= \frac{K}{2} (\log(V))^2 \\ W_{v4} &= \frac{K}{2} (V - 1)^2 \end{aligned} \quad (38)$$

Par défaut c'est le potentiel volumique 1 (W_{v1}) qui est utilisé dans le potentiel global. Dans le cas où l'on veut le potentiel volumique 3 par exemple on utilise le mot clé "type_potvol_"

suivi du numéro 3, ceci à la suite des paramètre de la loi, ce qui donne par exemple :

” C01= 0.0167 C10= 0.145 K= 3000 type_potvol_ 3 ”

TABLE 101 – Exemple de déclaration de la loi de Mooney Rivlin en 3D, dans le cas où les trois coefficients matériaux sont thermo-dépendants.

```
#-----
# Nom Materiau | Type loi |
#-----
          polymere    MOONEY_RIVLIN_3D
# -----
# |... loi de comportement hyper elastique 3D de type mooney rivlin ....|
# |                .. trois coefficients C01 et C10 ..                |
# -----
C01= C01_thermo_dependant_ courbe1
C10= C10_temperature_ courbe2
K= K_temperature_ courbe3
# .. fin de la definition de la loi money rivlin
```

28.3.5 ISOHYPER3DORGEAS1

Identificateur d'une loi 3D isotrope hyperélastique dont le potentiel est proposé par Laurent Orgéas (cf. Thèse de Doctorat Grenoble). Le potentiel est défini par l'expression suivante :

$$\begin{aligned}
\omega_4 = & \frac{K_{rev}}{6} \ln^2(V) + Q_{\sigma_{rev}} Q_\epsilon + \mu_{2rev} Q_\epsilon^2 \\
& + \frac{\mu_{1rev}}{2} \left[Q_\epsilon (Q_\epsilon + 2Q_{ec}) - (Q_\epsilon + Q_{ec}) (\alpha_{1rev}^2 + (Q_\epsilon + Q_{ec})^2)^{1/2} + Q_{ec} (\alpha_{1rev}^2 + Q_{ec}^2)^{1/2} \right. \\
& - \alpha_{1rev}^2 \left(\ln \left| Q_\epsilon + Q_{ec} + (\alpha_{1rev}^2 + (Q_\epsilon + Q_{ec})^2)^{1/2} \right| - \ln \left| Q_{ec} + (\alpha_{1rev}^2 + Q_{ec}^2)^{1/2} \right| \right) \left. \right] \\
& + \frac{\mu_{3rev}}{2} \left[Q_\epsilon \left(Q_\epsilon - 2(Q_{\epsilon_{rev}}^2 + \alpha_{3rev}^2)^{1/2} \right) \right. \\
& + \alpha_{3rev}^2 \left(\ln \left| Q_\epsilon - Q_{\epsilon_{rev}} + (\alpha_{3rev}^2 + (Q_\epsilon - Q_{\epsilon_{rev}})^2)^{1/2} \right| - \ln \left| -Q_{\epsilon_{rev}} + (\alpha_{3rev}^2 + Q_{\epsilon_{rev}}^2)^{1/2} \right| \right) \left. \right] \\
& + (Q_\epsilon - Q_{\epsilon_{rev}}) (\alpha_{3rev}^2 + (Q_\epsilon - Q_{\epsilon_{rev}})^2)^{1/2} + Q_{\epsilon_{rev}} (\alpha_{3rev}^2 + Q_{\epsilon_{rev}}^2)^{1/2} \quad (39)
\end{aligned}$$

La table (102) donne un exemple de déclaration. La loi nécessite 8 paramètres matériaux : K est le module de compressibilité (équivalent au cas de Hooke à : $E/(1 - 2\nu)$), Q_s est le seuil plateau en contrainte, $\mu_1 + \mu_2$ est la pente à l'origine, μ_2 est la pente du plateau, $\mu_3 + \mu_2$ est la pente après le plateau, Q_e est la déformation caractéristique pour sortir du plateau, et α_1 et α_2 sont deux paramètres qui règlent la courbure entre le plateau et les deux branches.

TABLE 102 – Exemple de déclaration de la loi d’hyper-élasticité Orgéas 1.

# loi de comportement 3D hyperélastique isotrope Orgeas 1									
#	K	Qs	mu1	mu2	mu3	alpha1	alpha2	Qe	
#	40000	400	30000	500.	30000	0.001	0.003	0.06	

Il est également possible de prendre en compte la dépendance à la phase sous la forme suivante :

$$\begin{aligned}
 Q_{\sigma_{rev}} &= \frac{Q_{\sigma_{0rev}}}{(1 + \gamma_{Q_{\sigma_{rev}}} \cos(3\varphi_{\epsilon}))^{n_{Q_{\sigma_{rev}}}}} \\
 Q_{\epsilon_{rev}} &= \frac{Q_{\epsilon_{0rev}}}{(1 + \gamma_{Q_{\epsilon_{rev}}} \cos(3\varphi_{\epsilon}))^{n_{Q_{\epsilon_{rev}}}}} \\
 \mu_{1rev} &= \frac{\mu_{01rev}}{(1 + \gamma_{\mu_{1rev}} \cos(3\varphi_{\epsilon}))^{n_{\mu_{1rev}}}} \\
 \mu_{2rev} &= \frac{\mu_{02rev}}{(1 + \gamma_{\mu_{2rev}} \cos(3\varphi_{\epsilon}))^{n_{\mu_{2rev}}}} \\
 \mu_{3rev} &= \frac{\mu_{03rev}}{(1 + \gamma_{\mu_{3rev}} \cos(3\varphi_{\epsilon}))^{n_{\mu_{3rev}}}} \tag{40}
 \end{aligned}$$

La table (103) donne un exemple de déclaration avec phase. $Q_{\sigma_{0rev}}$ est représenté par Q_s , μ_{02rev} par Q_e , μ_{01rev} par $mu1$, μ_{02rev} par $mu2$ et μ_{03rev} par $mu3$. Pour les paramètres supplémentaires, $\gamma_{Q_{\sigma_{rev}}}$ et $n_{Q_{\sigma_{rev}}}$ sont représentés par n_{Q_s} et $gamma_{Q_s}$, etc. pour les autres coefficients. La déclaration de tous les paramètres supplémentaires n’est pas obligatoire. Certains des paramètres peuvent être omis, seul l’ordre d’apparition doit-être respecté, c’est à dire les paramètres pour : Q_s , Q_e , $\mu1$, $\mu2$, $\mu3$. Enfin il est également possible, de prendre en compte une dépendance à la température. Dans une première étape seule les paramètre Q_s et Q_e (ou Q_{0s} et Q_{0e}) peuvent dépendre de la température. Cette dépendance pour Q_{0s} peut-être soit selon une évolution proposée par Laurent Orgéas dans sa thèse (page 54), soit une évolution quelconque donnée par une fonction 1D (d’une manière analogue aux autres lois). Dans le cas d’une évolution selon la méthode proposée par Laurent Orgéas :

$$\begin{aligned}
 \text{si } T \geq T_c \text{ alors } Q_s &= \frac{1}{2}G_r \left[(T - T_c + T_s) - \sqrt{(T - T_c + T_s)^2 + a_r^2} \right] + Q_{rmax} \\
 \text{sinon } Q_s &= \frac{1}{2}G_r \left[(T - T_c - T_s) + \sqrt{(T - T_c - T_s)^2 + a_r^2} \right]
 \end{aligned} \tag{41}$$

TABLE 103 – Exemple de déclaration de la loi d’hyper-élasticité Orgéas 1 avec dépendance à la phase.

```

hyperAvecPhase          ISOHYPER3DORGEAS1
# ..... loi de comportement 3D hyperelastique isotrope Orgéas 1 .....
#-----
#      K      |      Qs      |      mu1      |      mu2      |      mu3      |      alpha1      |      alpha2      |      Qe      |
#-----
#      270000      |      200      |      19000      |      300      |      10000      |      0.003      |      0.003      |      0.074 \
#      avec_phase
#      nQs= 0.1 gammaQs= 0.9  nQe= 0.2 gammaQe= 0.5
# ici seule les deux premiers paramètres dépendent de la phase,
# la ligne qui suit donne un exemple de déclaration de tous les paramètres
# nQs= 0.1 gammaQs= 0.9  nQe= 0.2 gammaQe= 0.5  nMu2= 1 gammaMu2= 0.7 \
# nMu2= 1 gammaMu2= 0.7  nMu3= 1 gammaMu3= 0.6

```

avec

$$\begin{aligned}
 T_c &= T_{0r} + \frac{Q_{rmax}}{2 G_r} \\
 T_s &= \frac{a_r^2 - (Q_{rmax}/G_r)^2}{2 (Q_{rmax}/G_r)} \quad (42)
 \end{aligned}$$

L’évolution correspond globalement à deux positions extrêmes (dans le plan Q_s en fonction de T , cela correspond à 2 segments de droites horizontaux) et une zone intermédiaire de raccordement représentée par un segment oblique. L’ensemble de l’évolution comprend donc 3 tronçons, qui sont raccordés de manière lissée. Les paramètres qui contrôlent l’évolution, sont :

- T_{0r} : température à partir de laquelle il y a une modification de Q_s ,
- G_r : la pente de l’évolution oblique,
- Q_{rmax} : la valeur maxi de Q_s ,
- a_r : un paramètre qui contrôle le rayon de courbure au niveau du raccordement entre les différents tronçons.

Dans le cas de Q_e , deux cas de thermo-dépendance sont proposés : soit une évolution suivant la fonction suivante :

$$Q_e(T) = Q_e(T_{0rQ_e}) + \frac{Q_s(T) - Q_s(T_{0rQ_e})}{3 h_1 (mu3 + mu2)} \quad (43)$$

Cette évolution est contrôlée par trois paramètres : une valeur a donner de $Q_e = Q_e(T_{0rQ_e})$ pour une température de référence T_{0rQ_e} a indiquer, et enfin un paramètre d’ajustement h_1 . La deuxième solution pour décrire une thermo-dépendance pour Q_e est de définir une fonction quelconque de T (comme pour Q_s).

La table (104) donne un exemple de loi dépendant de la phase et de la température selon l’évolution proposée par Laurent Orgéas dans sa thèse.

TABLE 104 – Exemple de déclaration de la loi d’hyper-élasticité Orgéas avec dépendance à la phase et une dépendance à la température du paramètre Q_{0s} selon l’évolution proposée par Laurent Orgéas dans sa thèse.

```
hyperAvecPhaseEtTemp      ISOHYPER3DORGEAS1
#-----
#   K   | Q0s       | mu1     | mu02 | mu03 | alpha1| alpha2 | Q0e   |"
#-----"
160000   Q0s_thermo_dependant_ 17000 220   17000  0.002   0.004   0.01   avec_phase
nQs= 1.1 gammaQs= 1.   nQe= 1.1 gammaQe= 1.   nMu2= 1.1 gammaMu2= 1.   nMu3= 1.1 gammaMu3= 1.
cas_Q0s_thermo_dependant_ 1  T0r= 308 Gr= 7.5   Qrmax= 570   ar= 9
```

On remarque que la valeur de Q_{0s} est remplacée par le mot clé “Q0s_thermo_dependant_” ensuite après la fin des autres paramètres, sur la ligne qui suit, on indique successivement :

1. le mot clé : “cas_Q0s_thermo_dependant_” suivi du numéro de cas, pour l’instant “1”,
2. puis les 4 paramètres selon la syntaxe présentée dans l’exemple. Tous les paramètres sont obligatoires.

Dans le cas d’une fonction quelconque la table (105) donne un exemple d’une telle dépendance. On observe que l’évolution de Q_{0s} est ici gérée par la courbe “courbe_evol_Q0s”

TABLE 105 – Exemple de déclaration de la loi d’hyper-élasticité Orgéas avec dépendance à la phase et une dépendance à la température du paramètre Q_{0s} selon une évolution donnée par la courbe courbe_evol_Q0s.

```
hyperAvecPhaseEtTemp      ISOHYPER3DORGEAS1
#-----
#   K   | Q0s       | mu1     | mu02 | mu03 | alpha1| alpha2 | Q0e   |"
#-----"
160000   Q0s_thermo_dependant_ 17000 220   17000  0.002   0.004   0.01   avec_phase
nQs= 1.1 gammaQs= 1.   nQe= 1.1 gammaQe= 1.   nMu2= 1.1 gammaMu2= 1.   nMu3= 1.1 gammaMu3= 1.
cas_Q0s_thermo_dependant_ 2  courbe_evol_Q0s
```

qui doit donc avoir été défini auparavant. Cette courbe peut-être quelconque.

Le cas de Q_e suit la même logique que pour Q_s . Pour activer la thermo-dépendance de Q_e , il faut indiquer le mot clé “Q0e_thermo_dependant_” à la place d’une valeur numérique pour Q_e . Ensuite après, la description de la thermo-dépendance de Q_s , on décrit (après avoir passé à la ligne) celle de Q_e avec successivement : le mot clé : “cas_Q0e_thermo_dependant_” suivi de “1” si l’on veut le premier cas d’évolution, “2”, si l’on veut définir une fonction quelconque. Ensuite dans le cas “1” on définit les trois paramètres selon la syntaxe : “T0rQe= <un reel> Qe.T0rQe= <un reel> h1= <un reel> ”. Dans le cas “2”, on écrit simplement le nom de la courbe, ou on la définit directement. La table (107) donne un

exemple de dépendance pour Q_s et Q_e (remarquer les barres obliques inverses (antislash) qui permettent de couper les lignes trop longues).

TABLE 106 – Exemple de déclaration de la loi d’hyper-élasticité Orgéas avec dépendance à la phase et une dépendance à la température des paramètre Q_{0s} selon une évolution donnée par la courbe courbe_evol_Q0s., et Q_{0e} selon la loi interne

```
hyperAvecPhaseEtTemp      ISOHYPER3DORGEAS1
#-----
#      K      | Q0s          | mu1      | mu02 | mu03 | alpha1| alpha2 | Q0e      |"
#-----
160000      Q0s_thermo_dependant_ 17000 220   17000   0.002   0.004   \
  Q0e_thermo_dependant_   avec_phase
nQs= 1.1 gammaQs= 1.   nQe= 1.1 gammaQe= 1.   nMu2= 1.1 gammaMu2= 1.   \
  nMu3= 1.1 gammaMu3= 1.
cas_Q0s_thermo_dependant_ 2   courbe_evol_Q0s
cas_Q0e_thermo_dependant_ 1   T0rQe= 308 Qe_T0rQe= 0.01 h1= 1.2
```

28.3.6 ISOHYPER3DORGEAS2

Identificateur d’une loi 3D isotrope hyper-élastique qui fonctionne au niveau théorique, de manière identique à ISOHYPER3DORGEAS1. Les différences avec cette dernière se situe au niveau des fonctions de dépendance à la phase. Dans ISOHYPER3DORGEAS1 la forme des fonctions de dépendance à la phase, est fixée, alors que dans ISOHYPER3DORGEAS2 les fonctions utilisables peuvent-être quelconque. La syntaxe d’entrée des données est différentes et la prise en compte de la phase est également légèrement différente. La table (107) donne un exemple d’utilisation. Dans cette exemple on remarque que les fonctions courbes sont données explicitement, mais il est également possible de donner le nom d’une courbe qui a été préalablement défini (comme habituellement pour les autres lois de comportement), ici c’est par exemple le cas pour les variables μ_{03_phi} avec la courbe3 (nom de courbe devant être défini au début du fichier .info).

Une fois ces informations lue, la valeur des paramètres est multipliés par la fonction associée dépendant de la phase. Ainsi par exemple $\mu_2 = \mu_{02} \cdot (\gamma + \alpha \cdot (\cos(3\varphi))^2)^n$.

Il est possible d’utiliser un ou plusieurs coefficients dépendant de la phase, chacun sur une ligne séparée, mais l’ensemble doit se terminer par le mot clé “fin_parametres_avec_phase_” seul sur une ligne.

Comme dans ISOHYPER3DORGEAS1 on peut également utiliser une dépendance à la température qui fonctionne de manière identique.

28.3.7 POLY_HYPER3D

Identificateur d’une loi 3D isotrope hyper-élastique de type polynomiale en invariants, les mêmes que ceux utilisés pour Mooney Rivlin. Le potentiel est défini par l’expression

TABLE 107 – Exemple de déclaration de la loi d’hyper-élasticité Orgéas avec dépendance à la phase selon des fonctions courbes

```

exemple_hyper          ISOHYPER3DORGEAS2
# ..... loi de comportement 3D hyperelastique isotrope Orgéas 1 .....
#-----
#      K      |      Q0s      |      mu1      |      mu02      |      mu03      |      alpha1      |      alpha2      |      Q0e      |
#-----
          370000      250      20000      1850      10000      0.0035      0.003      0.084 avec_phase

mu02_phi= COURBE_EXPO2_N # courbe (gamma+alpha*x)**n
          gamma= 1. alpha= 0.9 n= -4.
mu03_phi= courbe3
Q0s_phi=  COURBE_EXPO_N # courbe (gamma+alpha*x)**n
          gamma= 1. alpha= 0.9 n= 0.1
Q0e_phi= courbe5

fin_parametres_avec_phase_

```

suivante :

$$W_{polynomiale} = \sum_{i+j=1}^n (C_{ij} (J_1 - 3)^i (J_2 - 3)^j) + \left(K \left[1 - \frac{1 + \ln(\sqrt{I_3})}{\sqrt{I_3}} \right] \right) \quad (44)$$

où les coefficients matériaux sont C_{ij} , et K . La contrainte est celle de Cauchy et la déformation de travail est celle d’Almansi. Cependant il faut noter que cette déformation, ici n’intervient pas directement, le potentiel étant défini à partir des élongations, eux même calculées à partir du tenseur de Cauchy-Green droit : $\mathbf{C} = {}^t_{0..} \mathbf{G} = \hat{g}_{ij} \vec{g}^i \otimes \vec{g}^j$.

La table (108) donne un exemple de déclaration. On déclare tout d’abord le degré maxi “n”, puis on indique “tous” les coefficients, qu’ils soient nuls ou non. Il n’y a pas de limitation sur le degré.

Tous les coefficients matériels peuvent être thermo-dépendants. La table (109) donne un exemple de déclaration.

On observe que le potentiel, comme dans le cas de Mooney-Rivlin 3D, est composé d’une partie relative au changement de forme et une partie relative au changement de volume. Il est possible de changer la partie relative au changement de volume d’une manière identique au cas de la loi de Mooney-Rivlin, on s’y référera pour plus de détails sur la syntaxe et les potentiels proposés.

TABLE 108 – Exemple de déclaration de la loi polynomiale en 3D.

```

#-----
# Nom Materiau | Type loi |
#-----
          polymere    POLY_HYPER3D
# -----
# |... loi de comportement hyper elastique 3D de type polynomiale en invariants .. |
# |                .. coefficients de type Cij .. |
# -----
#          exemple de definition de loi
#    degre_maxi= 2 C01= 0.0167 C10= 0.145 C02= 0.0167 C11= 0.145 C20= 0.0167 K= 100
#    .. fin de la definition de la loi polynomiale
#
# on note que l'on met tout d'abord le groupe des coeff de degre 1, puis le groupe
# de degre 2. Tous les coefficients doivent etre presents
#-----

```

28.3.8 HART_SMITH3D

Identificateur d'une loi 3D isotrope de type Hart Smith hyperélastique. Le potentiel est défini par l'expression suivante :

$$\begin{aligned}
 W_{HartSmith} = & C_1 \int_3^{J_1(finale)} \exp[C_3 (J_1 - 3)^2] dJ_1 + C_2 \log\left(\frac{J_2}{3}\right) \\
 & + \left(K \left[1 - \frac{1 + \ln(\sqrt{I_3})}{\sqrt{I_3}} \right] \right) \quad (45)
 \end{aligned}$$

où les quatre coefficients matériaux sont C_1 , C_2 , C_3 et K . La contrainte est celle de Cauchy et la déformation de travail est celle d'Almansi. Cependant il faut noter que cette déformation, ici n'intervient pas directement, le potentiel étant défini à partir des élongations, eux même calculées à partir du tenseur de Cauchy-Green droit : $\mathbf{C} = {}^t_0 \mathbf{G} = \hat{g}_{ij} \bar{g}^i \otimes \bar{g}^j$.

La table (110) donne un exemple de déclaration.

Comme dans le cas de Mooney Rivlin, les coefficients matériels peuvent être thermo-dépendants. La table (111) donne un exemple de déclaration.

On observe que le potentiel est composé d'une partie relative au changement de forme et une partie relative au changement de volume. Il est possible de changer la partie relative

TABLE 109 – Exemple de déclaration de la loi polynomiale en 3D, dans le cas où les trois coefficients matériaux sont thermo-dépendants.

```
# -----
# |... loi de comportement hyper elastique 3D de type polynomiale en invariants .. |
# | .. coefficients de type Cij .. |
# -----
degre_maxi= 2  C01= C01_thermo_dependant_ courbe1
               C10= C10_temperature_ courbe2
               C02= C02_temperature_ courbe2
               C11= C10_temperature_ courbe2
               C20= 0.0167  K=  K_temperature_ courbe4
#-----
# noter qu'apres la definition de chaque courbe, on change de ligne, a l'inverse
# si la valeur du parametre est fixe, on poursuit sur la meme ligne.
#-----
#-----
# un dernier parametre facultatif permet d'indiquer le type de variation volumique
# que l'on desire: par default il s'agit de : K(1-(1+log(V))/V) qui correspond au type 1
# mais on peut choisir:
#           ou:          K/2(V-1)          qui correspond au type 2
#           ou:          K/2(log(V))^2      qui correspond au type 3
#           ou:          K/2(V-1)^2        qui correspond au type 4
# en indiquant (en fin de ligne) le mot cle: type_potvol_ suivi du type
# par default type_potvol_ a la valeur 1
#-----"
```

TABLE 110 – Exemple de déclaration de la loi de hart-smith3D en 3D.

```
#-----
# Nom Materiau | Type loi |
#-----
          polymere  HART_SMITH3D
# -----
# |.. loi de comportement hyper elastique 3D de type Hart-Smith |
# | .. quatre coefficients C1, C2, C3 et K .. |
# -----
#           exemple de definition de loi
C1= 1.    C2= 0.145  C3= 1.e-4  K= 100
# .. fin de la definition de la loi Hart Smith
```

au changement de volume. Quatre cas (idem Mooney-Rivlin) sont disponibles :

$$W_{v1} = K \left[1 - \frac{1 + \ln(\sqrt{I_3})}{\sqrt{I_3}} \right]$$

$$W_{v2} = \frac{K}{2} \frac{(V-1)}{165}$$

$$W_{v3} = \frac{K}{2} (\log(V))^2$$

Par défaut c'est le potentiel volumique 1 (W_{v1}) qui est utilisé dans le potentiel global. Dans le cas où l'on veut le potentiel volumique 3 par exemple on utilise le mot clé "type_potvol_3" suivi du numéro 3, ceci à la suite des paramètres de la loi, ce qui donne par exemple :

C1= 1. C2= 0.145 C3= 1.e-4 K= 100 type_potvol_3

TABLE 111 – Exemple de déclaration de la loi de Hart Smith en 3D, dans le cas où les quatre coefficients matériaux sont thermo-dépendants.

```
#-----
# Nom Matériau | Type loi |
#-----
          polymere   HART_SMITH3D
# -----
# |.. loi de comportement hyper elastique 3D de type Hart-Smith |
# |           .. quatre coefficients C1, C2, C3 et K .. |
# -----
C1= C1_thermo_dependant_courbe1
C2= C2_thermo_dependant_courbe2
C3= C3_temperature_courbe3
K= K_temperature_courbe4
#-----
# noter qu'après la définition de chaque courbe, on change de ligne, à l'inverse
# si la valeur du paramètre est fixe, on poursuit sur la même ligne.
# par exemple supposons C1 et C2 fixes et K thermo-dépendant, on écrit:
#-----
# C1= 1. C2= 0.145 C3= C3_temperature_courbe4
# K= K_temperature_courbe4
#-----
# .. fin de la définition de la loi Hart-Smith
```

Il est également possible d'adjoindre au potentiel une partie permettant de prendre en compte un raidissement.

Ce raidissement peut cependant être obtenue à l'aide d'un potentiel additionnel proposé par Moreau-Rio-Thuillier selon :

$$W_{courbure} = \frac{1}{\sqrt{I_3}} \left((J_1 - 3) \left(\frac{J_1 - 3}{a} \right)^{2r} \frac{1}{(2r + 1)} \right) \quad (47)$$

Le potentiel additionnel dépend de deux paramètres : a qui positionne le départ de la branche finale, et r qui pilote la courbure pour le passage sur la dernière branche.

On pourra se reporter au mémoire de thèse de Cécile Moreau, pour plus d'informations.

Pour introduire cette partie additionnelle on indique le mot clé facultatif : " avec_courbure_" à la fin des informations précédentes. Ensuite on passe une ligne et on définit les deux

paramètres : a et r. La table (112) donne un exemple de déclaration pour des paramètres constants.

TABLE 112 – Exemple de déclaration de la loi de Hart Smith en 3D avec un terme additionnelle dans le potentiel, permettant de décrire un raidissement en fin de chargement.

```
#-----
# Nom Materiau | Type loi          |
#-----
      polymere  HART_SMITH3D
C1= 1.    C2= 0.145  C3= 1.e-4  K= 100  type_potvol_ 2  avec_courbure_
a_courbure= 94 r_courbure= 1.24
#    .. fin de la definition de la loi Hart-Smith
```

Comme pour les autres paramètres, a et r peuvent dépendre de la température. La syntaxe est identique aux cas des autres paramètres. La table (113) donne un exemple de déclaration pour des paramètres dépendants de la température.

TABLE 113 – Exemple de déclaration de la loi de Hart Smith en 3D avec un terme additionnelle dans le potentiel, permettant de décrire un raidissement en fin de chargement : ici les paramètres a et r sont thermo-dépendants .

```
#-----
# Nom Materiau | Type loi          |
#-----
      polymere  HART_SMITH3D
C1= 1.    C2= 0.145  C3= 1.e-4  K= 100  type_potvol_ 2  avec_courbure_
a_courbure= a_temperature
r_courbure= r_temperature
#    .. fin de la definition de la loi Hart-Smith
```

28.4 Lois élasto-plastiques.

Les lois élasto-plastiques disponibles sont données dans la table (cf.114).

TABLE 114 – liste des différents lois isotropes élasto-plastiques disponibles

identificateur	indications	ref du commentaire
PRANDTL_REUSS1D	1D : Prandtl Reuss grandes déformations	(28.4.1)
PRANDTL_REUSS2D_D	2D : Prandtl Reuss grandes déformations déformations planes	(28.4.2)
PRANDTL_REUSS	3D : Prandtl Reuss grandes déformations	(28.4.3)

Les lois disponibles sont :

28.4.1 PRANDTL_REUSS1D

Identificateur d'une loi 1D isotrope élasto-plastique. Il s'agit ici de la loi classique de Prandtl Reuss classique. La loi nécessite en plus des données élastiques classique (E et ν), la donnée d'un seuil d'érouissage et d'une courbe d'érouissage présentée sous forme d'une suite de points. La table (115) donne un exemple de déclaration. Le mot clé "loi_ecrouissage" indique le type de description de la courbe, ici de type polylinéaire c'est-à-dire formée d'un ensemble de segment joignant des points. Les points sont donnés entre deux mots clés : "Debut_des_coordonnees_des_points" et "Fin_des_coordonnees_des_points". Chaque points est défini par également un mot clé : "Coordonnee", ensuite on indique la la dimension puis les 2 coordonnées.

TABLE 115 – Exemple de déclaration de la loi élasto-plastique de Prandtl Reuss en 1D .

```
#-----
# Nom Materiau | Type loi |
#-----
      MATE1      PRANDTL_REUSS1D

#-----
#      E      |      nu      |
#-----
E=      210000.      nu=      0.3
loi_ecrouissage COURBEPOLYLINEAIRE_1_D
  Debut_des_coordonnees_des_points
    Coordonnee dim= 2 0. 100.
    Coordonnee dim= 2 0.4 500.
    Coordonnee dim= 2 0.5 210000.
  Fin_des_coordonnees_des_points
```


28.4.2 PRANDTL_REUSS2D_D

Identificateur d'une loi 2D isotrope élasto-plastique en déformation plane. Il s'agit ici de la loi classique de Prandtl Reuss classique. La loi nécessite en plus des données élastiques classique (E et ν), la donnée d'un seuil d'érouissage et d'une courbe d'érouissage présentée sous forme d'une suite de points. Les données sont identique au cas Prandtl Reuss 1D, sauf le titre de la loi, ici "PRANDTL_REUSS2D_D".

28.4.3 PRANDTL_REUSS

Identificateur d'une loi 3D isotrope élasto-plastique. Il s'agit ici de la loi classique de Prandtl Reuss classique. La loi nécessite en plus des données élastiques classique (E et ν), la donnée d'un seuil d'érouissage et d'une courbe d'érouissage présentée sous forme d'une suite de points. Les données sont identique au cas Prandtl Reuss 1D, sauf le titre de la loi, ici "PRANDTL_REUSS".

28.5 Lois visco_élastiques isotropes.

Les lois visco_élastiques isotropes disponibles sont données dans la table (cf.116).

TABLE 116 – liste des différents lois visco-élastiques isotropes disponibles

identificateur	indications	ref du commentaire
NEWTON1D	1D : $\boldsymbol{\sigma} = \mu \mathbf{D}$ ou $\boldsymbol{\sigma} = \mu (\mathbf{D} : \mathbf{D})^{n/2} \mathbf{D}$	(28.5.1)
NEWTON2D_D	2D : déformation plane $\boldsymbol{\sigma} = \mu \bar{\mathbf{D}}$ ou $\boldsymbol{\sigma} = \mu (\bar{\mathbf{D}} : \bar{\mathbf{D}})^{n/2} \bar{\mathbf{D}}$	(28.5.2)
NEWTON3D	3D : $\boldsymbol{\sigma} = \mu \mathbf{D}$ ou $\boldsymbol{\sigma} = \mu (\bar{\mathbf{D}} : \bar{\mathbf{D}})^{n/2} \bar{\mathbf{D}}$	(28.5.3)
MAXWELL1D	1D : un ressort et un amortisseur en série	(28.5.4)
MAXWELL3D	3D : un ressort et un amortisseur en série en 3D	(28.5.6)

28.5.1 NEWTON1D

Identificateur d'une loi 1D isotrope viscoélastique de type Newton : $\boldsymbol{\sigma} = \mu \mathbf{D}$. La loi nécessite comme paramètre matériel un seul coefficient μ dans le cas linéaire. Il est également possible de lui adjoindre un coefficient non linéaire "n" sous forme de puissance, on obtient alors la relation : $\boldsymbol{\sigma} = \mu (\mathbf{D} : \mathbf{D})^{n/2} \mathbf{D}$ (ici la distinction déviatoire ou non n'a pas lieu d'être).

La table (117) donne un exemple de déclaration.

TABLE 117 – Exemple de déclaration d'une loi de Newton en 1D

```
#-----
# Nom Materiau      |      Type loi      |
#-----
      bois           NEWTON1D
# ..... loi de comportement Newton 1D .....
#      viscosite : et eventuellement une puissance
#      mu (obligatoire)                xn (facultatif)
      mu= 1.500000e-01                xnu= 1.2
      fin_coeff_NEWTON1D      # mot clé obligatoire de fin de définition
```

Il est également possible de définir une loi de Newton dont le coefficient dépendant de la température. La table (118) donne un exemple de déclaration de coefficient thermo-dépendant.

TABLE 118 – Exemple de déclaration d’une loi de Newton en 1D dont le coefficient μ dépend de la température

```
#----- debut cas d'une loi de newton thermodependante -----
polymere          NEWTON1D
# ..... loi de comportement Newton 1D .....
# | viscosite      |          et eventuellement une puissance      |
# |                |          pour une evolution non lineaire          |
# | mu (obligatoire) |          xn (facultatif)          |
#.....
      mu= mu_thermo_dependant_ CPL1D Dd1P 0. 15 100. 50 Fd1P
          fin_coeff_NEWTON1D

#----- fin cas d'une loi loi de newton thermodependante -----
```

Dans cet exemple la fonction $\mu=f(T)$ est explicitement définie. Il est également possible de mettre un nom de courbe déjà existante à la suite du mot clé : "mu_thermo_dependant_".

28.5.2 NEWTON2D_D

Identificateur d’une loi 2D isotrope viscoélastique de type Newton : $\sigma = \mu \bar{D}$, ceci en déformation plane. La loi nécessite comme paramètre matériel un seul coefficient μ dans le cas linéaire. Il est également possible de lui adjoindre un coefficient non linéaire "n" sous forme de puissance, on obtient alors la relation : $\sigma = \mu(\bar{D} : \bar{D})^{n/2} \bar{D}$.

Les relations sont donc exactement les mêmes, à la dimension près que dans le cas 1D. En particulier on remarque que le choix a été de relier la contraintes avec le déviateur des déformations.

Pour les entrées des données, la syntaxe est identique au cas 1D, en remplaçant "1D" par "2D_D".

28.5.3 NEWTON3D

Identificateur d’une loi 3D isotrope viscoélastique de type Newton : $\sigma = \mu \bar{D}$. La loi nécessite comme paramètre matériel un seul coefficient μ dans le cas linéaire. Il est également possible de lui adjoindre un coefficient non linéaire "n" sous forme de puissance, on obtient alors la relation : $\sigma = \mu(\bar{D} : \bar{D})^{n/2} \bar{D}$.

Les relations sont donc exactement les mêmes, à la dimension près que dans le cas 1D. En particulier on remarque que le choix a été de relier la contraintes avec le déviateur des déformations.

Pour les entrées des données, la syntaxe est identique au cas 1D, en remplaçant "1D" par "3D".

28.5.4 MAXWELL1D

Identificateur d'une loi 1D isotrope viscoélastique de type Maxwell, c'est-à-dire un ressort et un amortisseur linéaire en série. La loi nécessite 2 paramètres matériels et un paramètre de réglage. Paramètres matériels : le module d'Young du ressort linéaire E , et la viscosité de l'amortisseur μ . De plus l'intégration de la loi de comportement qui est de type incrémental, nécessite un choix de la dérivée matérielle du tenseur des contraintes. Ici trois cas de dérivée matérielle sont implémentés : Jauman, deux fois covariantes (valeur par défaut), deux fois contravariantes. Un paramètre de réglage optionnel permet de choisir entre ces 3 cas. La table (119) donne un exemple de déclaration. Concernant le type de dérivée nous avons :

- "type_derivee" = 1 - > derivee deux fois contravariantes
- "type_derivee" = 0 - > derivee deux fois covariantes (valeur par défaut)
- "type_derivee" = -1 - > derivee de jauman

TABLE 119 – Exemple de déclaration d'une loi de Maxwell en 1D

```
#-----
# Nom Materiau      |      Type loi      |
#-----
#      acier          MAXWELL1D
# ..... loi de comportement maxwell 1D .....
#      module d'young, viscosite et type de derivee objective utilisee
#      pour le calcul de la contrainte:
#      1.100000e+05      1.500000e-01 type_derivee  0
#                          fin_coeff_MAXWELL1D
```

Il est également possible de définir une loi de Maxwell dont les coefficients (certains ou tous) dépendent de la température. La table (120) donne un exemple de déclaration de coefficients thermo-dépendants.

TABLE 120 – Exemple de déclaration d'une loi de Maxwell en 1D dont les coefficients dépendent de la température au travers de courbes définies explicitement

```
#----- debut cas d'une loi de maxwell thermodépendante -----
#      polymere          MAXWELL1D
# ..... loi de comportement maxwell 1D .....
# | module d'young | viscosite | type de derivee objective utilisee | et éventuellement une puissance |
# |                |          | pour le calcul de la contrainte | pour une evolution non lineaire|
# |      E          |mu (obligatoire)| type_derivee (facultatif) |          xn (facultatif) |
# .....
#      E=      E_thermo_dependant_  CPL1D Dd1P 0. 100000. 100. 50000.0  Fd1P
#      mu=      mu_thermo_dependant_ CPL1D Dd1P 0. 0.15 100. 0.05  Fd1P
#                          fin_coeff_MAXWELL1D
#----- fin cas d'une loi loi de maxwell thermodépendante -----
```

Dans cet exemple toutes les fonction $f(T)$ sont explicitement définies. Il est également possible de mettre un nom de courbe déjà existante à la suite des mots clés de thermo-dépendance. La table (121) donne un exemple de déclaration de coefficients thermo-dépendants au travers de courbes référencées.

TABLE 121 – Exemple de déclaration d’une loi de Maxwell en 1D dont les coefficients dépendent de la température au travers de courbes définies explicitement

```
#----- debut cas d'une loi de maxwell thermodependante -----
polymere          MAXWELL1D
# ..... loi de comportement maxwell 1D .....
# | module d'young | viscosite      | type de derivee objective utilisee | et eventuellement une puissance |
# |               |                | pour le calcul de la contrainte   | pour une evolution non lineaire |
# |      E        | mu (obligatoire) | type_derivee (facultatif)         |          xn (facultatif)        |
# .....
E=  E_thermo_dependant_ courbe_E
mu=  mu_thermo_dependant_ courbe_mu
cn=  xn_thermo_dependant_ courbe_cn

          fin_coeff_MAXWELL1D

#----- fin cas d'une loi loi de maxwell thermodependante -----
```

Remarque A chaque fois que l’on utilise une courbe (explicite ou référencée), à sa suite, il faut passer à la ligne pour définir les coefficients qui suivent.

28.5.5 MAXWELL2D contraintes planes

Il est possible d’utiliser une loi de Maxwell en contraintes planes, suivant une méthodologie très proche du cas 3D, sans avoir cependant toutes ses fonctionnalités.

Les points communs :

- possibilité d’avoir, ou de ne pas avoir, une viscosité sur la partie sphérique,
- dépendance de tous les paramètres à la température,
- les différents types de dérivées,
- fonction multiplicative de mu, dépendant de $\mathbf{II}_{\bar{D}} = \mathbf{D} : \mathbf{D}$

Les fonctionnalités absentes :

- pas de dépendance possible à la déformation équivalente au sens de mises,
- pas de cas : seulement déviatorique, ou seulement sphérique,
- dépendante de la cristallinité,

Si les fonctionnalités absentes sont recherchées, il est possible d’utiliser la loi de passage en contraintes planes (28.7.2) avec la loi 3D.

La table (122) donne un exemple de déclaration avec les commentaires inclus dans le Herezh.

TABLE 122 – Exemple de déclaration d’une loi de Maxwell en 2D contraintes planes

```

#-----
# Nom Materiau      |      Type loi      |
#-----
polymere            MAXWELL2D_C
# ..... loi de comportement visco Maxwell 2D contraintes planes .....
# ..... loi de comportement maxwell 2D contraintes planes .....
# | module d'young | coeff |viscosite sur Dbarre| type de derivee objective utilisee |et eventuellem
# |                | de   |mu(obligatoire)    | pour le calcul de la contrainte  | multiplicativ
# |      E        |poisson |puis sur trace(D)  | type_derivee (facultatif)        | pour une evol
# |                |      |mu_p(optionnelle)  |                                     |
#.....
E= 500 nu= 3.00000000e-01 mu= 2000 mu_p= 2000 type_derivee -1
      fin_coeff_MAXWELL2D_C
# le type de derivee est optionnel: = -1 -> derivee de jauman,
# = 0 -> derivee deux fois covariantes (valeur par default),
# = 1 -> derivee deux fois contravariantes
# dans le cas ou l'on veut une valeur differente de la valeur par default il faut mettre le mot cle
# <type_derivee> suivi de la valeur -1 ou 0 ou 1
# \n# chaque parametre peut etre remplace par une fonction dependante de la temperature
# pour ce faire on utilise un mot cle puis une nom de courbe ou la courbe directement comme avec
# les autre loi de comportement
# exemple pour le module d'young:          E=  E_thermo_dependant_  courbe1
# exemple pour la viscosite sur Dbarre:    mu=  mu_thermo_dependant_  courbe2
# exemple pour la viscosite sur trace(D):  mu_p= mu_p_thermo_dependant_ courbe2
# exemple pour la viscosite:              mu=  mu_thermo_dependant_  courbe2
# dans le cas d'une thermo-dependance et d'une non linearite: mu = mu(T) * fac_mu_cissionD
# IMPORTANT: a chaque fois qu'il y a une thermodependance, il faut passer une ligne apres la description
# de la grandeur thermodependante, mais pas de passage à la ligne si se n'est pas thermo dependant
# la derniere ligne doit contenir uniquement le mot cle:      fin_coeff_MAXWELL2D_C

```

28.5.6 MAXWELL3D

Identificateur d'une loi 3D isotrope viscoélastique de type Maxwell, c'est-à-dire un ressort et un amortisseur linéaire en série. La partie sphérique élastique peut être soit identique à une loi de Hooke c'est-à-dire sans viscosité ou soit avec une évolution visqueuse. La partie déviatorique est systématiquement visqueuse. Ainsi la loi s'écrit pour la partie sphérique dans le cas sans viscosité :

$$\mathbf{I}_\sigma = \frac{E}{(1 - 2\nu)} \mathbf{I}_\varepsilon \quad (48)$$

\mathbf{I}_σ et \mathbf{I}_ε étant les traces des tenseurs contraintes et déformations. Et dans le cas avec viscosité :

$$\mathbf{I}_D = \frac{1}{3K} \dot{\mathbf{I}}_\sigma + \frac{\mathbf{I}_\sigma}{\mu_p} \quad (49)$$

Avec $3K = E/(1 - 2\nu)$

Pour la partie déviatoire on a :

$$\bar{\mathbf{D}} = \frac{1}{2G} \dot{\mathbf{S}} + \frac{\mathbf{S}}{\mu} \quad (50)$$

avec $\bar{\mathbf{D}}$ le déviateur du tenseur vitesse de déformation et $\dot{\mathbf{S}}$ une dérivée matérielle du déviateur des contraintes. Les paramètres de la loi sont ainsi le module d'Young E , le coefficient de Poisson ν , la viscosité μ , qui peuvent être comme dans le cas 1D, dépendants ou non de la température, et éventuellement (mais ce n'est pas obligatoire) une viscosité sur la partie sphérique μ_p . Trois types de dérivée matérielle sont implantées : Jauman (c'est-à-dire 1/2 de Lee en mixte dans les deux sens), de Lee deux fois covariantes (valeur par défaut), et de Lee deux fois contravariantes. Un paramètre de réglage optionnel permet de choisir entre ces 3 cas. La syntaxe est identique au cas 1D, excepté le coefficient de poisson qui est ici présent. La table (124) donne un exemple de déclaration. Concernant le type de dérivée nous avons :

- "type_derivee" = 1 - > dérivée deux fois contravariantes (ou d'Oldroyd)
- "type_derivee" = 0 - > dérivée deux fois covariantes (ou de Rivlin)
- "type_derivee" = -1 - > dérivée de Jauman (valeur par défaut)

Il est également possible de choisir une viscosité non linéaire pour la partie cisssion. ceci s'effectue par la définition d'une fonction multiplicative $f(\mathbf{II}_{\bar{\mathbf{D}}})$ définie à la suite du mot clé "fac_mu_cissionD=". La viscosité indiquée est alors multipliée par $f(\mathbf{II}_{\bar{\mathbf{D}}})$ calculée en fonction du taux de cisaillement en cours (cf.125) avec :

$$\mathbf{II}_{\bar{\mathbf{D}}} = \mathbf{D} : \mathbf{D} \quad (51)$$

De manière équivalente au cas précédent, il est également possible de choisir un module d'Young et ou une viscosité dépendant de la déformation, en fait de la déformation équivalente au sens de mises c'est-à-dire : $\varepsilon_{mises} = \sqrt{(2./3. (\bar{\boldsymbol{\varepsilon}} : \bar{\boldsymbol{\varepsilon}}))}$. Ceci s'effectue par la définition d'une fonction multiplicative $f_1(\varepsilon_{mises})$ définie à la suite du mot clé "fac_E_cissionEps=". Le module d'Young est alors multipliée par la fonction $f_1(\varepsilon_{mises})$. De même on peut indiquer le mot clé "fac_mu_cissionEps=" suivi d'une fonction multiplicative $f_2(\varepsilon_{mises})$. La viscosité indiquée est alors multipliée par $f_2(\varepsilon_{mises})$. La table (cf.123)

TABLE 123 – Exemple de déclaration d’une loi de Maxwell en 3D, dont les paramètres E et mu dépende de la déformation au sens de Mises

```
#-----
# Nom Materiau      |      Type loi      |
#-----
polymere            MAXWELL3D
# ..... loi de comportement visco Maxwell 3D .....
E= 500 nu= 3.00000000e-01 mu= 2000 type_derivee 1 \
    fac_E_Mises_Eps courbe1 # courbe1 = f_1(mises_eps) pour E
    fac_mu_Mises_Eps courbe2 # courbe 2 = f_2(mises_eps) pour mu
fin_coeff_MAXWELL3D
```

donne un exemple de déclaration (le "backslash" permet la continuation de la ligne sur les lignes suivantes). Ne pas oublier de passer à la ligne après chaque définition de fonction.

Enfin, de manière indépendante des paramètres précédents, il est également possible d’indiquer, que seule la contribution déviatorique de la contrainte est finalement calculée et prise en compte. Pour cela il suffit de mettre le mot clé "seule_deviatorique" à la fin des données c’est-à-dire sur la ligne précédent le mot clé "fin.coeff_MAXWELL3D" (cf. 124 et 126). Dans ce dernier cas, il faudra obligatoirement associer à la loi de maxwell une autre loi qui permettra de contribuer à une partie sphérique de contrainte de manière à éviter une indétermination (raideur nulle en déformation volumique!).

TABLE 124 – Exemple de déclaration d’une loi de Maxwell en 3D

```
#-----
# Nom Materiau      |      Type loi      |
#-----
polymere            MAXWELL3D
# ..... loi de comportement visco Maxwell 3D .....
E= 500 nu= 3.00000000e-01 mu= 2000 type_derivee 1
# 1) exemple dans le cas ou l'on veut une viscosite sur la partie spherique
# E= 500 nu= 3.00000000e-01 mu= 2000 mu_p= 2000 type_derivee 0
# 2) exemple dans le cas ou l'on ne veut que la contrainte ce cission
# E= 500 nu= 3.00000000e-01 mu= 2000 type_derivee 0 seule_deviatorique
    fin_coeff_MAXWELL3D
```

Comme dans le cas 1D, il est également possible de définir une loi de Maxwell dont les coefficients (certains ou tous) dépendent de la température. On se reportera aux tables (120) et (121) pour la syntaxe (en n’oubliant pas d’ajouter le coefficient de Poisson! et en notant que le coefficient xn pour le cas 1D n’a pas cours ici). On notera que le mot clé indiquant la thermo-dépendance pour la partie sphérique est "mu_p thermo-dependant_" et non "mu thermo-dependant_" ceci pour la différencier de la thermo-dépendance de la partie déviatorique. La table (126) donne un exemple d’entrée de données avec des viscosités thermo-dépendantes.

On introduit également la possibilité d’une viscosité dépendante de la cristallinité. La

TABLE 125 – Exemple de déclaration d’une loi de Maxwell en 3D avec une viscosité non linéaire

```
# ..... loi de comportement maxwell 3D .....
# | module | coeff |viscosite sur Dbarre| type de derivee objective utilisee |et eventuellement 1 fonction |"
# | d'young | de | mu(obligatoire) | pour le calcul de la contrainte | multiplicative de viscosite |"
# | E |poisson |puis sur trace(D) | type_derivee (facultatif) | pour une evolution |"
# | | |mu_p(optionnelle) | | non lineaire |"
#.....
E= 500 nu= 3.00000000e-01 mu= 2000 type_derivee 1 fac_mu_cissionD= courbe_fac_mu_cissionD
fin_coeff_MAXWELL3D
# le type de derivee est optionnel: = -1 -> derivee de jauman (valeur par défaut),
# = 0 -> derivee deux fois covariantes ,
# = 1 -> derivee deux fois contravariantes
#dans le cas ou l'on veut une valeur differente de la valeur par default il faut mettre le mot cle
# <type_derivee> suivi de la valeur -1 ou 0 ou 1
# chaque parametre peut etre remplace par une fonction dependante de la temperature
# pour ce faire on utilise un mot cle puis une nom de courbe ou la courbe directement comme avec
# les autre loi de comportement
# exemple pour le module d'young: E= E_thermo_dependant_ courbe1
# exemple pour la viscosite sur Dbarre: mu= mu_thermo_dependant_ courbe2
# exemple pour la viscosite sur trace(D): mu_p= mu_p_thermo_dependant_ courbe2
# exemple pour la viscosite: mu= mu_thermo_dependant_ courbe2
# dans le cas d'une thermo-dependance et d'une non linearite: mu = mu(T) * fac_mu_cissionD
# IMPORTANT: a chaque fois qu'il y a une thermodépendance, il faut passer une ligne apres la description
# de la grandeur thermodépendante, mais pas de passage à la ligne si se n'est pas thermo dependant
# la derniere ligne doit contenir uniquement le mot cle: fin_coeff_MAXWELL3D
```

TABLE 126 – Exemple de déclaration d’une loi de Maxwell en 3D avec les deux viscosités dépendantes d’une courbe de température "mu1dfT" défini par ailleurs dans le fichier .info

```

materiaux -----
#-----
# Nom Materiau | Type loi |
#-----
polymeremec MAXWELL3D
# ..... loi de comportement visco 3D .....
E= 1.9888e3 nu= 3.8e-01 mu= mu_thermo_dependant_ mu1dfT
mu_p= mu_p_thermo_dependant_ mu1sfT
# seule_deviatorique # mot cle a mettre pour ne calculer que S
fin_coeff_MAXWELL3D
```

viscosité est obtenue à l’aide d’une modélisation proposée par Hieber, S.Han et K.K. Wang. Nous avons :

$$\mu(\dot{\gamma}, T, p, \alpha) = \frac{\mu_0(T, p, \alpha)}{1 + \left(\mu_0 \frac{\dot{\gamma}}{\tau}\right)^{1-n}} \quad (52)$$

avec

$$\mu_0(T, p, \alpha) = \mu_0^* (T, p) \exp(C_1 \alpha^2) \quad (53)$$

et

$$\mu_0^*(T, p) = D_1 \exp\left(-\frac{A_1 (T - T^*)}{A_2 + (T - T^*)}\right) \text{ et } T^* = D_2 + D_3, A_2(p) = A_2^* + D_3 p \quad (54)$$

La viscosité dépend ainsi de 8 paramètres matériels : n , τ^* , D_1 , D_2 , D_3 , A_1 , A_2^* , C_1 . On notera que le calcul de la viscosité dépend également de la température et de la pression.

TABLE 127 – Exemple de déclaration d’une loi de Maxwell en 3D avec une viscosité dépendante de la cristallinité

```

          materiaux -----
#-----
# Nom Materiau      |      Type loi      |
#-----
polymere           MAXWELL3D
# ..... loi de comportement visco 3D .....
# avec dependance a la cristalinite

# 1) parametres elastiques
E= 1.9888e3 nu= 3.8e-01 depend_cristalinite_

# 2) parametres pour le calcul de la viscosite
nc= 0.3452 tauStar= 1.128e4 D1= 2.780e14 D2= -15.
D3= 1.4e-7 A1= 29.94 At2= 51.6 C1= 2171 # volumique_visqueux_ crista_aux_noeuds_

# 3) type de derivee et calcul uniquement deviatorique ou pas
type_derivee -1 # seule_deviatorique

          fin_coeff_MAXWELL3D

```

La table (127) donne un exemple de définition de loi utilisant une viscosité dépendante de la cristallinité. Le mot clé “mu=” est remplacé par “depend_cristalinite_”, ensuite sur les deux lignes qui suivent on indique de manière exhaustive, la liste des paramètres matériels qui permet le calcul de la viscosité. Le paramètre optionnel “volumique_visqueux_” indique lorsqu’il est présent, que la viscosité s’applique également à la partie sphérique, alors que par défaut, la partie volumique est non visqueuse. Ensuite sur la ligne qui suit on peut indiquer (paramètre optionnel) le type de dérivée et également le mot clé “seule_deviatorique” qui indique lorsqu’il est présent, que seule la partie déviatorique du tenseur des contraintes, est calculée.

Remarque Il n’est pas possible d’utiliser le mot clé “fac_mu_cissionD=” dans le cas de la dépendance à la cristallinité, car cette dépendance impose déjà la non linéarité.

Lorsque l’on utilise ce modèle, il est impératif que la température soit présente dans le calcul, ainsi que la cristallinité. Cette dernière peut-être accessible de deux manière. Par défaut elle est supposée connue au point d’intégration (c’est-à-dire au point où est calculé la loi de comportement). Dans ce cas, la cristallinité doit-être calculée au sein d’une loi

thermo-physique (cf. 163) via un modèle de calcul de la cristallinité par exemple celui de Hoffman (cf.29.4). Une seconde possibilité est d'accéder à la cristallinité via des valeurs supposées connues aux noeuds. Pour ce faire on doit mettre le mot clé "crista_aux_noeuds_" avant ou après le mot clé "volumique_visqueux_" si ce dernier existe ou à sa place s'il n'existe pas. De plus il faut que les noeuds contiennent effectivement la variable de cristallinité, via une lecture de champ de cristallinité par exemple.

28.6 Composition additive de différentes lois de base (ex : loi des mélanges).

Les composition additive disponibles sont données dans la table (cf.128).

TABLE 128 – liste des différentes compositions additives disponibles de lois élémentaires

identificateur	indications	ref du commentaire
LOI_ADDITIVE_EN_SIGMA	1D, 2D, 3D : $\sigma = \sum \sigma_{(i)}$	(28.6.1)
LOI_DES_MELANGES_EN_SIGMA	1D, 2D, 3D : $\sigma = (\alpha)\sigma_{(1)} + (\alpha - 1)\sigma_{(2)}$ ou $\Delta\sigma = (\alpha)\Delta\sigma_{(1)} + (\alpha - 1)\Delta\sigma_{(2)}$	(28.6.2)

28.6.1 LOI_ADDITIVE_EN_SIGMA

Identificateur d'une loi additive en contrainte. Le principe est de déclarer une succession de loi de comportement, qui chacune contribueront au calcul de la contrainte finale de la manière suivante. A partir de la déformation, chaque loi détermine une contrainte et éventuellement un comportement tangent. Ces informations sont sommées pour obtenir en finale la contrainte et le comportement tangent totales. La table (129) donne un exemple de déclaration.

TABLE 129 – Exemple de déclaration d'une loi additive en contrainte

```
#-----
# Nom Materiau      |      Type loi      |
#-----
plastique           LOI_ADDITIVE_EN_SIGMA
# ..... loi de comportement loiAdditiveEnSigma .....
# il faut donner le nom de chaque loi suivi des parametres sur les lignes suivantes
# exemple de deux lois elastiques
      ISOELAS # premiere loi isoelas 3D
# ..... loi de comportement isoelastique 3D .....
#   module d'young :   coefficient de poisson
1.100000e+05  1.500000e-01

      ISOELAS # seconde loi isoelas 3D
# ..... loi de comportement isoelastique 3D .....
#   module d'young :   coefficient de poisson
1.100000e+05  1.500000e-01

      fin_liste_lois_elementaires   # ----- fin de loiAdditiveEnSigma
```

28.6.2 LOI DES MELANGES EN SIGMA

Identificateur d'une loi des mélanges en contrainte. Le principe est de déclarer une succession de 2 lois de comportement, qui chacune contribueront au calcul de la contrainte finale de la manière suivante. À partir de la déformation, chaque loi détermine une contrainte et éventuellement un comportement tangent, on obtient alors deux comportements. Ensuite la contrainte finale est obtenue à partir d'une pondération des deux comportements initiaux. Soit par exemple "α" le facteur de pondération, on obtient pour la contrainte finale :

$$\sigma_{final} = \alpha\sigma_1 + (1. - \alpha)\sigma_2 \quad (55)$$

Le même calcul est effectué pour la raideur. Le facteur de pondération est associé à une grandeur qui peut soit être définie à chaque noeud, soit être directement accessible au point d'intégration (le point où la loi est calculée) cependant pour l'instant, les cas suivants sont implantés :

- un identificateur d'un ddl définit au noeud, par exemple la température TEMP (cf. 133),
- "PROP_CRISTA" Le mot clé pour le choix entre grandeurs aux noeuds ou au point d'intégration est "valeur_aux_noeuds=" et la table (132) donne un exemple de déclaration.
- "def_equivalente", "def_duale_mises", "Spherique_eps" et "Spherique_sig" : le fonctionnement est identique au cas précédant. La table 134 montre un exemple (complexe) de loi de mélange pilotée par la pression, et faisant appelle à deux lois additives.

La grandeur de proportion peut-être soit une donnée, soit une variable. La table (130) donne un exemple de déclaration. Dans cette exemple la grandeur qui sert de proportion est "PROP_CRISTA" ce qui correspond au taux de cristallinité. Dans tous les cas la grandeur doit être comprise entre 0. et 1. sinon on peut imaginer qu'il y a peut-être une erreur. Au niveau du programme, la grandeur de contrôle est systématiquement bornée pour appartenir à l'intervalle [0, 1]. Ainsi si la grandeur est inférieure à 0, elle est arbitrairement mise à 0 (sans message particulier). Si la grandeur est supérieure à 1., elle est bornée à 1. également de manière arbitraire. La table (133) donne la liste des grandeurs potentiellement disponible aux noeuds (mais il faut soit les définir directement à l'aide de conditions de blocage, ou soit s'assurer qu'elles seront effectivement calculées durant la résolution).

Il est également possible de définir une loi des mélanges sur les accroissements de contraintes. Dans ce cas on a la relation suivante :

$$\Delta\sigma_{final} = \alpha\Delta\sigma_1 + (1. - \alpha)\Delta\sigma_2 \text{ et } \sigma_{final}^{t+\Delta t} = \sigma_1^t + \sigma_2^t + \Delta\sigma_{final} \quad (56)$$

Dans ce dernier cas, la contrainte finale cumule ainsi toute l'histoire du facteur de pondération, ce qui peut être dans certain cas intéressant. Pour utiliser cette seconde loi des mélanges il faut indiquer à la suite de la définition de la grandeur proportion, le mot clé type_melange= suivi du chiffre 2 . Par défaut le type_melange vaut 1 ce qui correspond au premier fonctionnement. La table (131) donne un exemple de fonctionnement.

La valeur finale de la contrainte peut-être multipliée par un coefficient "A" :

$$\sigma_{final} = A(\alpha\sigma_1 + (1. - \alpha)\sigma_2) \text{ ou } \Delta\sigma_{final} = A(\alpha\Delta\sigma_1 + (1. - \alpha)\Delta\sigma_2) \quad (57)$$

Pour activer ce cas, à la suite du type de mélange on indique le mot clé "somme_pondere_etendue_" puis sur la ligne suivante : le mot clé "A=" suivi de la valeur numérique du coefficient "A" (voir exemple [134](#)).

Il est possible également d'utiliser une fonction pour calculer α à partir de la grandeur de pondération (par exemple on peut vouloir une dépendance à la température de α , c'est-à-dire une fonction $\alpha(TEMP)$). Pour ce faire on indique après les précédents mots clés, le mot clé : "avec_fonction_proportion_", puis sur la ligne suivante la description de la fonction sous la forme d'un nouveau mot clé : "fonction_prop=" suivi du nom de la fonction (si elle existe déjà) ou de la définition de la fonction. Ne pas oublier de passer à la ligne après la description de la fonction (comme pour toutes les lois). La table [134](#) donne un exemple de fonction.

TABLE 130 – Exemple de déclaration d’une loi additive en contrainte

```

# ..... loi de comportement LoiDesMelangesEnSigma .....
polymere          LOI_DES_MELANGES_EN_SIGMA

# ..... loi de comportement LoiDesMelangesEnSigma .....
# sur la premiere ligne on indique successivement :
# a) obligatoirement: le type de grandeur (mot cle: grandeur_proportion= ) qui regle
# la proportion entre les deux lois, on peut utiliser n'importe quel grandeur defini
# (mais qui doit exister par ailleurs) aux noeuds par exemple, cf. documentation
# b) optionnellement: le type de melange (mot cle: type_melange= ) (1 ou 2) cf.
# expli qui suit
# c) optionnellement: si la proportion provient des noeuds ou du pt d'integ directement
# (mot cle: valeur_aux_noeuds= ) (1 ou 0) la valeur par defaut est 1 (en fait
# tout nb diff de 0), dans le cas de 0, pour l'instant le seul cas implante est
# relatif a PROP_CRISTA
# --> ex:  grandeur_proportion=  PROP_CRISTA  type_melange= 2  valeur_aux_noeuds= 0
#
#
# ---- explications complementaires ----
# en fait il y a deux type de loi des melanges possible:
# soit : sigma = grandeur * sigma(loi1) + (1.-grandeur) * sigma(loi2) , ce qui est le
# type 1 (par defaut)
# soit : delta sigma = grandeur * delta sigma(loi1) + (1.-grandeur) * delta sigma(loi2) ,
# ce qui est le type 2
# en tenant compte que les contraintes sont cumulees dans le type 2.
# le type 1 est par defaut, mais on peut indiquer le type 2 apres le mot cle
# type_melange=, puis il faut donner le nom de chaque loi (2 maxi) suivi des parametres
# sur les lignes suivantes
#
#
# ----- exemple de deux lois elastiques -----
#
grandeur_proportion=  PROP_CRISTA  type_melange= 1  valeur_aux_noeuds= 1

# autre possibilite equivalente vue les grandeurs par defaut:
# grandeur_proportion=  PROP_CRISTA
#
# definition de la premiere loi
#----- debut cas d'une premiere loi isoelas acier -----
ISOELAS1D
#-----
#|          E          |          NU          |
#-----
#          200.          |          0.3          |
#----- fin cas d'une premiere loi isoelas acier -----

# definition de la seconde loi
#----- debut cas d'une seconde loi isoelas acier -----
ISOELAS1D
#-----
#|          E          |          NU          |
#-----
#          200000.          |          0.3          |
#----- fin cas de la seconde loi isoelas acier -----

fin_liste_lois_elementaires  # ----- 183 ----- fin de LoiDesMelangesEnSigma

```

TABLE 131 – Exemple de déclaration d’une loi additive en contrainte avec une proportion sur les accroissements des contraintes (et non sur la contrainte totale)

```

polymere      LOI_DES_MELANGES_EN_SIGMA
# ..... loi de comportement LoiDesMelangesEnSigma .....

      grandeur_proportion=  PROP_CRISTA  type_melange= 2 ";

#          definition de la premiere loi
#----- debut cas d'une premiere loi isoelas acier -----
      ISOELAS1D
#-----
#|          E          |          NU          |
#-----
      200.          0.3
#----- fin cas d'une premiere loi isoelas acier -----

#          definition de la seconde loi
#----- debut cas d'une seconde loi isoelas acier -----
      ISOELAS1D
#-----
#|          E          |          NU          |
#-----
      200000.          0.3
#----- fin cas de la seconde loi isoelas acier -----

      fin_liste_lois_elementaires  # ----- fin de LoiDesMelangesEnSigma

```


TABLE 132 – Exemple de déclaration d’une loi additive en contrainte, dans le cas ou on utilise une proportion directement accessible au point d’intégration (donc non interpolée à partir de grandeurs aux noeuds)

```
# ..... loi de comportement LoiDesMelangesEnSigma .....

polymere      LOI_DES_MELANGES_EN_SIGMA
# ..... loi de comportement LoiDesMelangesEnSigma .....

grandeur_proportion=  PROP_CRISTA  type_melange= 2  valeur_aux_noeuds= 0

#          definition de la premiere loi
          ISOELAS1D
#-----
#|          E          |          NU          |
#-----
          200.          0.3

#          definition de la seconde loi
          ISOELAS1D
#-----
#|          E          |          NU          |
#-----
          200000.          0.3

          fin_liste_lois_elementaires  # ----- fin de LoiDesMelangesEnSigma
```

TABLE 133 – Liste exhaustive des différentes grandeurs susceptibles d’être disponible aux noeuds

```
X1 , X2 , X3,          # les 3 coordonnées
EPAIS , TEMP ,        # l’épaisseur, la température
UX, UY, UZ , V1 , V2 , V3,  # déplacements, vitesses
PR, GAMMA1, GAMMA2, GAMMA3,  # pression, accélération
SIG11,SIG22,SIG33,SIG12,SIG23,SIG13  # contraintes
EPS11,EPS22,EPS33,EPS12,EPS23,EPS13, # déformations
DEPS11,DEPS22,DEPS33,DEPS12,DEPS23,DEPS13, # delta déformation
ERREUR,              # estimation d’erreur
PROP_CRISTA,         # taux de cristallinité
DELTA_TEMP,          # delta température
R_X1,R_X2,R_X3,      # réaction en position bloquée
,R_V1,R_V2,R_V3,    # réaction en vitesse
R_GAMMA1,R_GAMMA2,R_GAMMA3,  # réaction en accélération
```

TABLE 134 – exemple de loi des mélanges pilotée par la trace des contraintes, la loi de mélange appelle elle même deux lois additives

```

loi_comp_trac LOI_DES_MELANGES_EN_SIGMA

# ..... en tete de la loi de comportement melange progressif .....

grandeur_proportion= Spherique_sig type_melange= 1 valeur_aux_noeuds= 0 somme_pondere_etendue_
A= 1. avec_fonction_proportion_
# ici on definit la courbe de progression en fonction de la def spherique
fonction_prop= COURBEPOLYLINEAIRE_1_D
Debut_des_coordonnees_des_points
Coordonnee dim= 2 -10. 1. # en compression c'est la premiere loi
Coordonnee dim= 2 -0.01 1. #
Coordonnee dim= 2 0.01 0. #
Coordonnee dim= 2 10 0. # en traction c'est la seconde loi
Fin_des_coordonnees_des_points

# ..... def des deux lois membres

# --- loi 1) variation de volume negative (compression) -----
LOI_ADDITIVE_EN_SIGMA
HYSTERESIS_3D
np= 0.318 mu= 0.556 Qzero= 0.103 avec_parametres_de_reglage_
type_de_resolution_2 cas_kutta_5 erreurAbsolue_ 1.e-3
erreurRelative_ 1.e-3 tolerance_coincidence_ 1.e-5 nb_maxInvCoinSurUnPas_ 10
fin_parametres_reglage_Hysteresis_
HART_SMITH3D
C1= 0.4 C2= 0.08 C3= 0.09 K= 2700. type_potvol_ 4
MAXWELL3D
E= 2.21 nu= 0.45 mu= 28.86 mu_p= 0.001 type_derivee -1
fin_coeff_MAXWELL3D
MAXWELL3D
E= 0.672 nu= 0.45 mu= 135.63 mu_p= 0.001 type_derivee -1
fin_coeff_MAXWELL3D
fin_liste_lois_elementaires
# --- fin loi 1) -----

# --- loi 2) variation de volume positive (expansion) -----
LOI_ADDITIVE_EN_SIGMA
HYSTERESIS_3D
np= 0.48 mu= 0.994 Qzero= 0.08 avec_parametres_de_reglage_
type_de_resolution_2 cas_kutta_5 erreurAbsolue_ 1.e-3
erreurRelative_ 1.e-3 tolerance_coincidence_ 1.e-5 nb_maxInvCoinSurUnPas_ 10
fin_parametres_reglage_Hysteresis_
HART_SMITH3D
C1= 0.368 C2= 0.28 C3= 0.32 K= 2700. type_potvol_ 4
MAXWELL3D
E= 3.29 nu= 0.45 mu= 33.74 mu_p= 0.001 type_derivee -1
fin_coeff_MAXWELL3D
MAXWELL3D
E= 1.001 nu= 0.45 mu= 177.26 mu_p= 0.001 type_derivee -1
fin_coeff_MAXWELL3D
fin_liste_lois_elementaires
# --- fin loi 2) -----

fin_liste_lois_elementaires # ----- fin de LOI_DES_MELANGES_EN_SIGMA

```

28.7 Passage en déformations et contraintes planes pour une loi 3D quelconque

Il est possible d'utiliser une loi 3D quelconque associée à des conditions de déformations planes ou de contraintes planes, ceci pour travailler avec des éléments finis de type 2D : i.e. des éléments 2D utilisés en membranes ou représentant une cinématique telle que "UZ=0", et des éléments coques (les contraintes planes étant plus particulièrement dédiées aux éléments coques et membranes dans un espace 3D).

Tout d'abord il faut noter que l'espace de travail doit obligatoirement est en 3D, car la loi de comportement interne utilisant des tenseurs d'ordre 3, ceci n'est pas possible dans un espace de travail 2D. Il faut donc veillez à utiliser des conditions limites correctes sur les 3 dimensions (par exemple UZ).

28.7.1 LOI_DEFORMATIONS_PLANES

Le fonctionnement de l'algorithme est le suivant :

- il y a calcul dans le plan de l'élément fini : des bases et éléments de métriques, des déformations 2D, ainsi que leurs variations par rapport aux degrés de liberté,
- les tenseurs d'ordre 2 sont transformés en ordre 3 avec addition des conditions de déformations planes suivant l'axe local 3,
- la loi de comportement 3D est alors appelée avec les tenseurs d'ordre 3,
- le tenseur des contraintes et ses variations, sont ensuite transformés d'ordre 3 en 2
- les résultats sont transmis aux méthodes générales de calcul des puissances et de la raideurs, internes.

En post-traitement, seules les coordonnées d'ordre 2 du tenseur de contrainte sont directement accessible via les grandeurs classiques. Cependant les grandeurs 3D sont également accessibles, mais via les grandeurs particulières associées à la loi de comportement.

Au niveau de l'utilisation, la méthodologie est simple.

- sur une première ligne on indique le nom choisit pour la loi (nom donné par l'utilisateur) suivi du mot clé : "LOI_DEFORMATIONS_PLANES"
- ensuite suit sur les lignes suivantes, la définition d'une loi 3D quelconque (qui peut intégrer des combinaisons de lois élémentaires par exemple)
- puis sur la dernière ligne on doit trouver le mot clé (seul) : "fin_loi_deformation_plane"

La table 135 donne un exemple de loi de déformation plane associée à une loi 3D additive en sigma.

28.7.2 LOI_CONTRAINTES_PLANES

Plusieurs techniques peuvent être retenues pour imposer les conditions de contraintes planes i.e. $\sigma^{i3} = 0$ avec "3" la direction normale à l'élément 2D. La première technique implantée s'appuie sur la méthode de perturbation suivante :

- Dans le cas d'un algorithme implicite de recherche d'équilibre (type Newton), à chaque itération à la suite de l'équilibre, l'épaisseur de l'élément est mise à jour en fonction de la trace du tenseur des contraintes dans lequel on impose $\sigma^{i3} = 0$, du module de compressibilité et de la variation de la surface de l'élément. Cette nou-

TABLE 135 – exemple de loi de déformations planes intégrant un comportement 3D et des conditions de déformations planes

```

#-----
# Nom Materiau |      Type loi      |
#-----
  polymere_visco_elastique          LOI_DEFORMATIONS_PLANES

      LOI_ADDITIVE_EN_SIGMA
# ..... loi de comportement loiAdditiveEnSigma .....

# partie hyper-élastique de type Mooney Rivlin
      MOONEY_RIVLIN_3D
C01= 0.0167 C10= 0.145 K= 300

# partie visco-elastique de type maxwell
      MAXWELL3D
# ..... loi de comportement Maxwell 3D .....
#   E : nu : mu
E= 500 nu= 3.00000000e-01 mu= 2000 type_derivee -1 seule_deviatorique
fin_coeff_MAXWELL3D

      fin_liste_lois_elementaires # ---- fin de loiAdditiveEnSigma

fin_loi_deformation_plane # --- fin de la loi de deformation plane

```

velle épaisseur est pris en compte au cours de l’itération suivante, pour calculer la déformation ϵ_{33} , les autres déformations $\epsilon_{\alpha 3}$, $\alpha = 1..2$, étant nulles.

- Dans le cas d’un algorithme explicite d’avancement temporel par exemple, à chaque incrément l’épaisseur d’élément est mise à jour en fonction des contraintes et du module de compressibilité de manière analogue aux cas des itérations en implicite. La nouvelle épaisseur est utilisée pour l’incrément qui suit. Cette algorithme n’est licite que dans le cas de très petites variations entre deux incréments de temps consécutif, ce qui est le cas en général, en dynamique explicite.

Remarques sur le déroulement :

- Les base, composantes de métriques, déformations 2D dans le plan de l’élément 2D, ainsi que leurs variations par rapport aux ddl, sont calculés directement à partir de la cinématique des noeuds de l’éléments de manière classique (via les coordonnées entraînées).
- Le vector normal, \vec{g}_3 est systématiquement normé, on a donc $\vec{g}_3 = \vec{g}^3$ et $g_{33} = g^{33} = 1$ et la déformation suivant la direction 3 n’est pas calculée à l’aide de la variation des métriques suivant ”33”, mais à l’aide de la variation d’épaisseur, avec un calcul différent suivant la mesure de déformation retenue (Almansi, Logarithmique, Cumulée Logarithmique)
- ensuite la loi de comportement 3D est alors appelée avec les tenseurs d’ordre 3,
- le tenseur des contraintes et ses variations, sont ensuite transformés d’ordre 3 en 2. La contrainte suivant 3 est mise à 0 (cf. l’introduction)
- les résultats sont transmis aux méthodes générales de calcul des puissances et de la raideurs, internes.

En post-traitement, seules les coordonnées d'ordre 2 du tenseur des déformations sont directement accessible via les grandeurs classiques. Cependant la déformation d'épaisseur peut être récupérée via les grandeurs particulières associées à la loi de comportement (mot clé proposé en interactif : DEF_EPAISSEUR).

Au niveau de l'utilisation, la méthodologie est simple.

- sur une première ligne on indique le nom choisit pour la loi (nom donné par l'utilisateur) suivi du mot clé : "LOI.CONTRAINTE.PLANES"
- puis sur la ligne qui suit on indique le type de technique utilisée pour tenir compte de la condition de contraintes planes,
- ensuite suit sur les lignes suivantes, la définition d'une loi 3D quelconque (qui peut intégrer des combinaisons de lois élémentaires par exemple)
- puis sur la dernière ligne on doit trouver le mot clé (seul) : "fin_loi_contrainte_plane"

La table 136 donne un exemple de loi avec condition de contrainte plane associée à une loi 3D additive en sigma.

TABLE 136 – Exemple de loi de contraintes planes intégrant un comportement 3D et des conditions de contraintes planes. Dans cette exemple, la prise en compte de la condition de contraintes planes s'effectue par perturbation.

```
#-----
# Nom Materiau | Type loi |
#-----
PPC LOI_CONTRAINTE.PLANES
PERTURBATION deformation_epaisseur
LOI_ADDITIVE_EN_SIGMA
ISOHYPER3DFAVIER3
2000. 4. 240. 9.
MAXWELL3D
E= 345.56 nu= 0.45 mu= 2661.44 mu_p= 0.1000000000E-04 type_derivee -1
fin_coeff_MAXWELL3D
MAXWELL3D
E= 102.15 nu= 0.45 mu= 7541.05 mu_p= 0.1000000000E-04 type_derivee -1
fin_coeff_MAXWELL3D
MAXWELL3D
E= 136.95 nu= 0.45 mu= 50177.97 mu_p= 0.1000000000E-04 type_derivee -1
fin_coeff_MAXWELL3D
HYSTERESIS_3D
np= 0.3 mu= 300. Qzero= 8. avec_parametres_de_reglage_
type_de_resolution_ 2 cas_kutta_ 5 erreurAbsolue_ 1.e-3
erreurRelative_ 1.e-3 tolerance_coincidence_ 1.e-5 nb_maxInvCoinSurUnPas_ 10
fin_parametres_reglage_Hysteresis_
fin_liste_lois_elementaires
fin_loi_contrainte_plane
```

28.8 Lois d'hystérésis.

Les loi d'élasto-hystérésis disponibles sont données dans la table (cf.137).

TABLE 137 – liste des différents lois d'élasto-hystérésis

identificateur	indications	ref du commentaire
HYSTERESIS_1D	1D : loi d'hystérésis classique (modèle de Guélin-Favier-Pégon)	(28.8.1)
HYSTERESIS_3D	3D : loi d'hystérésis classique modifiée (modèle de Guélin-Favier-Pégon-Bless-Rio)	(28.8.2)

28.8.1 HYSTERESIS_1D

Identificateur d'une loi d'hystérésis 1D. La loi comporte d'une part une partie incrémentale dont l'équation est :

$$\dot{\sigma} = 2\mu\bar{D} + \beta\phi\Delta_R^t\bar{\sigma} \quad (58)$$

avec :

$$\begin{aligned} Q_{\Delta\sigma} &= \sqrt{\text{trace}(\Delta_R^t\sigma.\Delta_R^t\sigma)} \\ \phi &= \Delta_R^t\sigma : \bar{D} \\ \beta &= \frac{-2\mu}{(w'Q_0)^{np}(Q_{\Delta\sigma})^{2-np}} \end{aligned} \quad (59)$$

w' est le paramètre de Masing. Dans le cas 1D il vaut 1. sur la courbe de première charge et 2 pour toutes les autres évolutions. La loi comporte d'autre part un algorithme de gestion des boucles, en particulier gestion des points d'inversions et gestion des points de coïncidence.

On se reportera aux travaux théoriques de Guelin, Pegon, Favier, Manach, Orgeas, Tourabi, Couty, Bless, Rio pour plus d'informations.

Les paramètres de la loi de comportement sont ainsi : μ qui représente la pente initiale, Q_0 qui représente le maxi des boucles en contrainte, np le paramètre de Prager qui contrôle le passage de la pente initiale au seuil maxi avec la limitation : $0 < np$.

La table (138) donne un exemple de déclaration.

Il est possible de définir des paramètres matériaux qui dépendent de la température. La syntaxe est identique au cas 3D, on s'y référera donc (28.8.2). De même les paramètres de l'algorithme de résolution peuvent être modifiés de manière identique au cas 3D (syntaxe identique). Ces paramètres ne sont pas obligatoires. Dans le cas où on veut les modifier on indique le mot clé : "avec_parametres_de_reglage_" à la fin des paramètres matériaux et on va à la ligne pour indiquer les paramètres dont l'apparition doit respecter l'ordre suivant (mais certain paramètre peuvent être absents) :

- "type_de_resolution_" : par défaut 1 c'est-à-dire par linéarisation de l'équation constitutive, voir l'exemple (139), 2 indique que l'on veut une résolution explicite avec une méthode de Runge-Kutta imbriquée, voir l'exemple (140),

- “nb_iteration_maxi_” : utilisé avec la méthode de linéarisation, indique le nombre d’itération maxi permit dans la méthode de Newton de résolution, par défaut 100,
- “nb_dichotomie_maxi_” : utilisé avec la méthode de linéarisation, indique le nombre de sous pas qu’il est permis pour décomposer le pas initial dans le cas de non convergence de l’algorithme de Newton, par défaut 4,
- “tolerance_residu_” : utilisé avec la méthode de linéarisation, indique la tolérance sur le résidu à convergence de la méthode de Newton, par défaut $1.e - 6$,
- “cas_kutta_” : utilisé avec la méthode de Runge-Kutta, indique le type de méthode de Runge-Kutta imbriquée à utiliser : “3” pour une méthode imbriquée 2-3, “4” pour une méthode imbriquée 3-4, “5” pour une méthode imbriquée 4-5, par défaut 5,
- “erreurAbsolue_” puis “erreurRelative_” : utilisé avec la méthode de Runge-Kutta, la précision utilisée est = l’erreur absolue + l’erreur relative . la valeur de la fonction, par défaut l’erreur absolue est $1.e - 3$ et l’erreur relative est $1;e - 5$,
- “nbMaxiAppel_” : utilisé avec la méthode de Runge-Kutta, indique le nombre maxi d’appel de la fonction dérivée (sigma point ici), par défaut 120,
- “tolerance_coincidence_” indique la tolérance que l’on accepte sur les coïncidences.

TABLE 138 – Exemple de déclaration d’une loi d’hystérésis 1D

```
#----- debut cas d'une loi d'hysteresis -----
#-----
# Nom Materiau      |      Type loi      |
#-----
# AMF                HYSSTERESIS_1D
#-----
# ..... loi_de_comportement d'hysteresis 1D ..... |
# para de prager    :      mu      : limite de plasticite |
#-----
# np= 10.000000e+00      mu= 1.000000e+03      Qzero= 2.000000e+02
#
# -- definition du type de deformation (par defaut: DEFORMATION_STANDART) --
#      type_de_deformation      DEFORMATION_STANDART
#----- fin cas d'une loi d'hysteresis -----
```

28.8.2 HYSSTERESIS_3D

Identificateur d’une loi d’hystérésis 3D. La loi comporte d’une part une partie incrémentale de même type que le cas 1D, dont l’équation est :

$$\dot{\sigma} = 2\mu\bar{D} + \beta\phi\Delta_R^t\bar{\sigma} \quad (60)$$

TABLE 139 – Exemple de déclaration d’une loi d’hystérésis 1D avec paramètres de réglage pour une méthode de résolution de l’équation constitutive par linéarisation

```
#-----
# ..... loi_de_comportement d'hysteresis 1D ..... |
# para de prager      :      mu      : limite de plasticite |
#-----
      np= 2.000000e+00      mu= 2.      Qzero= 0.2. avec_parametres_de_reglage_
nb_iteration_maxi_ 20 nb_dichotomie_maxi_ 20 tolerance_residu_ 1.e-5 tolerance_coincidence_ 1.e-3
#----- fin cas d'une loi d'hysteresis -----
```

TABLE 140 – Exemple de déclaration d’une loi d’hystérésis 1D avec paramètres de réglage pour une méthode de résolution Runge-Kutta

```
#-----
# ..... loi_de_comportement d'hysteresis 1D ..... |
# para de prager      :      mu      : limite de plasticite |
#-----
      np= 2.000000e+00      mu= 2.      Qzero= 0.2. avec_parametres_de_reglage_
# tous les parametres qui suivent doivent être sur une seule ligne contrairement à l'exemple qui suit
      type_de_resolution_ 2      cas_kutta_ 5      erreurAbsolue_ 1.e-3      erreurRelative_ 1.e-3
      tolerance_coincidence_ 1.e-4      tolerance_coincidence_ 1.e-4
#----- fin cas d'une loi d'hysteresis -----
```

avec :

$$\begin{aligned}
 Q_{\Delta\sigma} &= \sqrt{\text{trace}(\Delta_R^t \boldsymbol{\sigma} \cdot \Delta_R^t \boldsymbol{\sigma})} \\
 \phi &= \Delta_R^t \boldsymbol{\sigma} : \bar{\mathbf{D}} - \frac{(Q_{\Delta\sigma})^2 \dot{w}'}{2\mu w'} \\
 \beta &= \frac{-2\mu}{(w' Q_0)^{np} (Q_{\Delta\sigma})^{2-np}} \\
 w' &= w \cos(\alpha)
 \end{aligned} \tag{61}$$

w est le paramètre de Masing. Il vaut 1. sur la courbe de première charge et 2 pour toutes les autres évolutions. α est l’angle de phase entre $\Delta_R^t \bar{\boldsymbol{\sigma}}$ et $\Delta_{O_i}^R \bar{\boldsymbol{\sigma}}$, O_i étant le centre de la i^{ieme} sphère (en dim 6) limite.

La loi comporte d’autre part un algorithme de gestion des boucles, en particulier gestion des points d’inversions et gestion des points de coïncidence.

On se reportera aux travaux théoriques de Guélin, Pegon, Favier, Manach, Orgeas, Tourabi, Couty, Bless, Rio pour plus d’informations.

Les paramètres de la loi de comportement sont ainsi : μ qui représente la pente initiale, Q_0 qui représente le maxi des boucles en contrainte, np le paramètre de Prager qui contrôle le passage de la pente initiale au seuil maxi avec la limitation : $0 < np$. D’une manière plus précise Q_0 est tel que :

- en cisaillement la contrainte maxi est $\tau = Q_0 / \sqrt{2}$

- en traction simple $\sigma_{maxi} = \sqrt{(3/2)} * Q0$

Ainsi en résumé : la limite ultime de plasticité (sachant qu’il y a de la plasticité dès le début de la sollicitation) est en traction : $\sqrt{(3/2)} * Q0$ et en cisaillement : $Q0/\sqrt{(2)}$

On notera que la partie hystérésis ne participe qu’à la partie déviatorique de la contrainte. Il est donc nécessaire d’adjoindre une partie sphérique pour que le tenseur soit complet.

La table (142) donne un exemple de déclaration.

Il est possible d’indiquer une thermo-dépendance des paramètres de la loi de comportement. D’une manière analogue à toutes les lois thermo-dépendante, à la suite du paramètre on indique un mot clé précisent la thermo-dépendance puis le nom d’une courbe ou la définition directe de la courbe. La définition du paramètre suivant doit alors se faire sur une nouvelle ligne. La table (144) donne un exemple de déclaration avec 3 paramètres thermo-dépendants, et la La table (145) donne un exemple de déclaration pour un premier paramètres fixe, les deux autres étant thermo-dépendants.

De manière a contrôler plus précisément l’algorithme de résolution de l’équation constitutive, il est possible de modifier les paramètres de l’algorithme. Ces paramètres ne sont pas obligatoires. Dans le cas où on veut les modifier on indique le mot clé : “avec_parametres_de_reglage_” à la fin des paramètres matériaux et on va à la ligne pour indiquer les paramètres dont l’ordre d’apparition peut-être quelconque (ces paramètres sont tous optionnels). Chaque paramètre est précédé d’un mot clé, on a donc une succession de mot clé suivi d’une grandeur, on peut avoir un ou plusieurs couple parametre-grandeur sur chaque ligne, par contre la dernière ligne doit comporter uniquement le mot cle : “fin_parametres_reglage_Hysteresis_”.

Les différents parametres sont :

- “type_de_resolution_” : 1 indique que la résolution s’effectue par linéarisation de l’équation différentielle constitutive ce qui conduit à une équation tensorielle non linéaire, qui est résolu par une méthode itérative de Newton (avec sous-découpage de l’incrément de déformation, si la précision voulu n’est pas obtenue), voir l’exemple (139). 2 indique que l’on veut une intégration explicite de l’équation différentielle avec une méthode de Runge-Kutta imbriquée, voir l’exemple (140). Dans ce dernier cas, il n’y a pas linéarisation de l’équation constitutive. Lorsque la précision ne peut-être atteinte, la méthode bascule automatiquement vers une linéarisation de Newton. Si cette dernière ne marche toujours pas, la méthode fait un appel directe à un Runge-Kutta d’ordre 5 et l’erreur estimée est écrite (si le niveau d’impression est supérieure à 0). Différentes variantes sont également disponible :
 1. “type_de_resolution_” 3 : idem pour le début que le cas 2, mais dans le cas où la méthode initiale de Kutta imbriqué ne converge pas, arrêt et génération d’une exception de non convergence, qui est récupérée au niveau du calcul de la raideur et du pilotage globale du calcul.
 2. “type_de_resolution_” 4 : idem pour le début que le cas 2, mais dans le cas où la méthode de Newton ne converge pas, arrêt et génération d’une exception de non convergence, qui est récupérée au niveau du calcul de la raideur et du pilotage globale du calcul.
 3. “type_de_resolution_” 5 : idem pour le début que le cas 2, mais dans le cas où la méthode de Newton ne converge pas, on rebasculer sur la méthode de Kutta

imbriqué, mais en relâchant la précision demandée. En fait c'est une boucle, qui à chaque itération, multiplie par 10 les précisions relative et absolue demandées. En sortie, impression d'un Warning avec la précision estimée obtenue.

Sans autre information, le plus simple est d'utiliser le type 2 ou 4. Ce dernier est le seul qui garantie que l'on n'utilisera pas une contrainte calculée avec une précision moins bonne que celle demandée.

“Ex : type_de_resolution_ 2”

- “type_calcul_comportement_tangent_” suivi de 1 ou 2 : dans le cas où =1 cela signifie que le comportement tangent est effectué à l'aide de l'équation constitutive linéarisée. C'est le cas par défaut lorsque le type de résolution est 1. Dans le cas où =2 cela signifie que le comportement tangent est effectué à l'aide de l'équation constitutive sans linéarisation (dernière méthode implantée). Cette dernière méthode est a priori plus robuste. “Ex : type_calcul_comportement_tangent_ 2 ”
- “nb_iteration_maxi_” : utilisé avec la méthode de linéarisation, indique le nombre d'itération maxi permit dans la méthode de Newton de résolution. “Ex : nb_iteration_maxi_ 10 ”
- “nb_dichotomie_maxi_” : utilisé avec la méthode de linéarisation, indique le nombre de sous pas qu'il est permis pour décomposer le pas initial dans le cas de non convergence de l'algorithme de Newton. “Ex : nb_dichotomie_maxi_ 4 ”
- “tolerance_residu_” : utilisé avec la méthode de linéarisation, indique la tolérance sur le résidu à convergence de la méthode de Newton. “Ex : tolerance_residu_ 1.10⁻⁵ ”.
- “cas_kutta_” : utilisé avec la méthode de Runge-Kutta, indique le type de méthode de Runge-Kutta imbriquée à utiliser : “3” pour une méthode imbriquée 2-3, “4” pour une méthode imbriquée 3-4, “5” pour une méthode imbriquée 4-5. “Ex : cas_kutta_ 5 ”
- “erreurAbsolue_” puis “erreurRelative_” : utilisé avec la méthode de Runge-Kutta, la précision utilisée est = l'erreur absolue + l'erreur relative . la valeur de la fonction. “Ex : erreurAbsolue_ 1.10⁻⁵ ” et “erreurRelative_ 1.10⁻⁴”.
- “nbMaxiAppel_” : utilisé avec la méthode de Runge-Kutta, indique le nombre maxi d'appel de la fonction dérivée (sigma point ici). “Ex : nbMaxiAppel_ 1000”
- “tolerance_coïncidence_” indique la tolérance que l'on accepte sur les coïncidences. “Ex : tolerance_coïncidence_ 1.e-5”
- “mini_rayon_” : le mini en dessous duquel on considere $\Delta_{0_i}^R \bar{\sigma}$ et $\Delta_R^{t+\Delta t} \bar{\sigma}$ nuls “Ex : mini_rayon_ 1. e-8”
- “nb_maxInvCoinSurUnPas_ ” indique le nombre maximum permis d'inversion et/ou de coïncidence sur un pas. Dans le cas ou ce maxi est un nombre négatif, on utilise sa valeur absolue et lorsque le maxi dépasse ce nombre, on retient la valeur final de la contrainte, sinon cas normal : on génère une erreur. “Ex : nb_maxInvCoinSurUnPas_ -10”
- “min_angle_trajet_neutre_” indique la valeur mini du $\cos(\Delta\phi)$ en dessous de laquelle on considère que l'évolution est neutre, $\Delta\phi$ étant la valeur le l'angle entre $\Delta_{0_i}^R \bar{\sigma}$ et $\Delta_R^{t+\Delta t} \bar{\sigma}$. Une valeur typique est 0.008, ce qui correspond à environ 0.5 degré. “Ex : min_angle_trajet_neutre_ 0.0001 ”
- “possibilite_cosAlphaNegatif_ ” indique si oui (diff de 0) on non (0) on accepte transitoirement une valeur de $\cos(\Delta\phi)$ négative. Par défaut c'est oui. Si c'est non, dans

le cas d'une valeur négative, le traitement est identique au cas neutre : "ex : possible_cosAlphaNegatif_ 0 "

- "avecVarWprimeS_" permet de tenir compte ou pas de la variation de l'angle de phase et de sa dérivée dans le calcul du comportement tangent. Par défaut, il n'y a pas de prise en compte de ces variations. Pour en tenir compte, il faut mettre indiquier le mot cle "avecVarWprimeS_" suivi de 1 (0 étant la valeur par défaut). "Ex : avecVarWprimeS_ 1 "
- "permet_affichage_" permet l'affichage des erreurs et des warning suivant un niveau particulier qui suit la même logique que le niveau d'affichage global. L'affichage s'effectuera donc en fonction de l'affichage normale et de l'affichage particulier. "Ex : permet_affichage_ 5"

Le tableau (141) donne les valeurs par défaut des différents paramètres.

TABLE 141 – valeur par défaut des paramètres de réglage du calcul de l'hystérésis 3D

paramètre	valeur par défaut	valeur préconisée a priori
type_de_resolution_	1	2
type_calcul_comportement_tangent_	1	2
nb_iteration_maxi_	6	10
nb_dichotomie_maxi_	4	4
tolerance_residu_	1.10^{-3}	1.10^{-3}
cas_kutta_	5	5
erreurAbsolue_	1.10^{-3}	1.10^{-3}
erreurRelative_	1.10^{-5}	1.10^{-3}
nbMaxiAppel_	1000	1000
tolerance_coincidence_	tolerance_residu_ * 100	1.10^{-5}
mini_rayon_	1.10^{-12}	1.10^{-12}
nb_maxInvCoinSurUnPas_	4	-10
min_angle_trajet_neutre_	0.008	0.008
avecVarWprimeS_	0	0
permet_affichage_	0	0

La table (143) donne un exemple de déclaration de la partie hystérétique, avec utilisation des paramètres de contrôle.

Dépendance à la phase : Il est possible d'introduire une dépendance à la phase des paramètres de la loi. Bien noter que la phase considérée est celle de la contrainte d'hystérésis seule (et non la contrainte totale lors d'une loi additive). Ainsi dans la pratique, cette dépendance n'est a priori correcte que dans le cas d'une loi d'élastohystérésis pure (partie sphérique élastique, partie déviatorique hystérétique pure).

La phase du déviateur concerne l'angle de phase (dans le plan déviatoire) de $\Delta \mathbf{S}_{O_i}^{t+\Delta t}$, O_i étant le centre de référence actuellement actif. Pour introduire la dépendance à la

TABLE 142 – Exemple de déclaration d’une loi d’hystérésis 3D

```

#-----
# Nom Materiau | Type loi |
#-----
niti LOI_ADDITIVE_EN_SIGMA
# ..... loi de comportement loiAdditiveEnSigma .....
#----- partie spherique : loi isoelas -----
      ISOELAS
#-----
#|          E          |          NU          |          type          |
#-----
      200000.          0.3          seule_spherique
#----- fin partie spherique: loi isoelas -----
#----- debut cas d'une loi d'hysteresis (partie deviatorique) -----
#-----
# Nom Materiau      |          Type loi          |
#-----
      HYSTERESIS_3D
#-----
#|..... loi_de_comportement d'hysteresis 3D ..... |
#| para de prager      :          mu          : limite de plasticite |
#-----
      np= 0.500000e+00          mu= 60000          Qzero= 400

#----- fin cas d'une loi d'hysteresis -----
fin_liste_lois_elementaires
# ..... fin de loiAdditiveEnSigma .....

```

TABLE 143 – Exemple de déclaration d’une loi d’hystérésis 3D avec des paramètres de contrôle de l’algorithme de résolution

```

#-----
# ..... loi_de_comportement d'hysteresis 3D ..... |
# para de prager      :          mu          : limite maxi de plasticite |
#-----
      np= 0.500000e+00          mu= 60000          Qzero= 400          avec_parametres_de_reglage_
      nb_iteration_maxi_ 20          nb_dichotomie_maxi_ 20
      tolerance_residu_ 1.e-5          tolerance_coincidence_ 1.e-3
      fin_parametres_reglage_Hysteresis_

```

phase, à la suite de la déclaration du dernier paramètre matériau, on met le mot cle : “avec_phase” puis sur la ligne qui suit on met les fonctions de controle de la dependance.

TABLE 144 – Exemple de déclaration d’une loi d’hystérésis 3D avec des paramètres matériau thermo-dépendants

```
#-----
# ..... loi_de_comportement d'hysteresis 3D ..... |
# para de prager      :      mu      : limite maxi de plasticite |
#-----
np= np_thermo_dependant_ courbe1
mu= mu_thermo_dependant_ courbe4
Qzero= Qzero_thermo_dependant_ courbe3
```

TABLE 145 – Exemple de déclaration d’une loi d’hystérésis 3D avec un premier paramètre matériau fixe puis les autres thermo-dépendants

```
#-----
# ..... loi_de_comportement d'hysteresis 3D ..... |
# para de prager      :      mu      : limite maxi de plasticite |
#-----
np= 2.    mu= mu_thermo_dependant_ courbe4
Qzero= Qzero_thermo_dependant_ courbe3
```

TABLE 146 – Exemple de déclaration d’une loi d’hystérésis 3D avec paramètres de réglage pour une méthode de résolution Runge-Kutta

```
#-----
# ..... loi_de_comportement d'hysteresis 3D ..... |
# para de prager      :      mu      : limite de plasticite |
#-----
np= 2.000000e+00      mu= 2.      Qzero= 0.2. avec_parametres_de_reglage_
type_de_resolution_ 2 cas_kutta_ 5 erreurAbsolue_ 1.e-3 erreurRelative_ 1.e-3
tolerance_coincidence_ 1.e-4 tolerance_coincidence_ 1.e-4 nb_maxInvCoinSurUnPas_ 4
min_angle_trajet_neutre_ 0.01
fin_parametres_reglage_Hysteresis_
#----- fin cas d'une loi d'hysteresis -----
```

Il est possible d’omettre une fonction mais il faut respecter l’ordre. Comme pour toutes les courbes on peut soit mettre directement la courbe, soit mettre un nom de courbe (ne pas oublier de passer à la ligne apres chaque courbe). Voici un exemple partiel de déclaration de dépendance à la phase, qui intègre également des paramètres de réglage de l’algorithme :

```
# ---- les parametres de la loi suivis du mot cle avec_phase

np= 1      mu= 4000      Qzero= 80      avec_phase
```

```

xnp_phase= courbe_xnp_phase
xmu_phase= courbe_xmu_phase
Qzero_phase= courbe_Qzero_phase

# ---- suivi de parametres de reglage

                avec_parametres_de_reglage_
type_de_resolution_ 2
cas_kutta_ 5 erreurAbsolue_ 1.e-3 erreurRelative_ 1.e-5
nbMaxiAppel_ 600 tolerance_coincidence_ 1.e-4
nb_maxInvCoinSurUnPas_ -20
mini_Qsig_pour_phase_sigma_Oi_tdt_ 10
                fin_parametres_reglage_Hysteresis_

```

Dépendance à une déformation équivalente De manière à mieux maîtriser le passage du régime principalement d'élasticité au régime principalement de plasticité, qui est géré par le paramètre de Prager, il est possible d'introduire une dépendance de ce dernier à une mesure de déformation équivalente, utiliser pour le calcul de l'énergie : ϵ_{equi}

$$\epsilon_{equi} = \int_R^t \frac{\phi}{\omega' Q_{\Delta\sigma}} d\tau \quad (62)$$

Cette dépendance a été introduite par Pierre Pégon (cf. document de thèse : annexe II.2 formule A.2.1), puis reprise par Laurent Orgéas (cf. document de thèse : chapitre II.55 formule II-38). Dans notre cas l'implantation est légèrement différente, mais doit conduire à une évolution du même ordre :

$$np = (2 - c1) f\left(-\epsilon_{equi} \left(\frac{2 Q_0^2}{\omega' \mu}\right) Q_{\Delta\sigma}\right) + c1 \quad (63)$$

Pour indiquer la dépendance, on indique après le dernier paramètre matériau, le mot clé "avec_defEqui". Le paramètre np peut avoir une dépendance, ainsi sur la ligne qui suit on met la fonction de contrôle de la dépendance $f()$ qui est utilisée dans la formule (??). Suit un exemple de déclaration. Comme pour toutes les courbes on peut soit mettre directement la courbe, soit mettre un nom de courbe. Ne pas oublier de passer à la ligne après chaque courbe.

```

# ---- les parametres de la loi suivis du mot cle avec_defEquivalente

np= 1          mu= 4000      Qzero= 80      avec_defEquivalente
xnp_defEqui= COURBE_RELAX_EXPO
          xa= 1.  xb= 0.  xc= 1. avec_bornesMinMax_
          xmin= 0.  xmax= 50.
          fin_bornesMinMax_

```

```

# ---- suivi de parametres de reglage

          avec_parametres_de_reglage_
type_de_resolution_ 2
cas_kutta_ 5 erreurAbsolue_ 1.e-3 erreurRelative_ 1.e-5
nbMaxiAppel_ 600 tolerance_coincidence_ 1.e-4
nb_maxInvCoinSurUnPas_ -20
mini_Qsig_pour_phase_sigma_Oi_tdt_ 10
          fin_parametres_reglage_Hysteresis_

```

Il est également possible de définir un écrouissage isotrope au travers d'une dépendance du paramètre Q_0 à la déformation équivalente maxi observée sur la courbe de première charge. Cette fonctionnalité produit un comportement qui n'est pas toujours correcte dans le cas de l'algorithme de fermeture de boucle, il est donc à utiliser avec précaution. Le texte littéral qui suit donne un exemple partiel de déclaration :

```

# ---- les parametres de la loi suivis du mot cle avec_defEquivalente

      np= 1      mu= 4000      Qzero= 80      avec_defEquivalente

xnp_defEqui= COURBE_RELAX_EXPO
      xa= 1.  xb= 0.  xc= 1.  avec_bornesMinMax_
      xmin= 0.  xmax= 50.
      fin_bornesMinMax_

Qzero_defEqui= COURBEPOLYLINEAIRE_1_D
      Debut_des_coordonnees_des_points
      Coordonnee dim= 2 0. 1.
      Coordonnee dim= 2 0.1 1.64
      Fin_des_coordonnees_des_points

          avec_parametres_de_reglage_
type_de_resolution_ 2
cas_kutta_ 5 erreurAbsolue_ 1.e-3 erreurRelative_ 1.e-5
nbMaxiAppel_ 600 tolerance_coincidence_ 1.e-4
nb_maxInvCoinSurUnPas_ -20
mini_Qsig_pour_phase_sigma_Oi_tdt_ 10
          fin_parametres_reglage_Hysteresis_

```

Dans cette exemple on a une dépendance de xnp et de Q_0 . Cependant il est possible d'avoir l'une ou l'autre. La courbe d'évolution de Q_0 à la déformation équivalente "maxi" sur la courbe de première charge, peut-être quelconque.

28.9 Lois hypo-élastiques

Les lois hypo-élastiques disponibles sont données dans la table (cf.147).

TABLE 147 – liste des différents lois isotropes élastiques disponibles

identificateur	indications	ref du commentaire
HYPO_ELAS3D	3D : loi hypo-élastique linéaire $\dot{\mathbf{S}} = \mu \bar{\mathbf{D}}$ et $\dot{I}_\sigma = K_c I_{\mathbf{D}}$	(28.9.1)
HYPO_ELAS2D_C	2D : loi hypo-élastique contrainte plane $\dot{\mathbf{S}} = \mu \bar{\mathbf{D}}$ et $\dot{I}_\sigma = K_c I_{\mathbf{D}}$	(28.9.2)

28.9.1 HYPO_ELAS3D

Identificateur d'une loi hypo-élastique 3D isotrope définie par les relations suivantes : $\dot{\mathbf{S}} = \mu \bar{\mathbf{D}}$ et $\dot{I}_\sigma = K_c I_{\mathbf{D}}$. Cette loi convient pour les éléments 3D. Elle nécessite la donnée d'un coefficient de compressibilité tangent K_c , d'un module de cisaillement tangent μ et d'un type de dérivée matérielle pour l'intégration de la loi de comportement. Il est également possible de définir des paramètres matériels dépendant de la température : $K_c(T)$ et/ou $\mu(T)$. Dans ce cas il faut s'assurer que la température est définie aux noeuds soit en tant que donnée, soit en tant que variable. La table (148) donne un exemple de loi simple (pas de thermo-dépendance). La table (149) donne un exemple avec thermo-dépendance.

Il est également possible d'utiliser une compressibilité calculé à l'aide d'une thermo-physique (celle associée au même élément fini). Dans ce cas on ne donne aucune valeur, par contre on utilise le mot clé : "Kc_avec_compressibilite_externe". La table (150) donne un exemple de compressibilité calculée en dehors de la loi. Pour que l'ensemble fonctionne correctement, il ne faut pas oublier de définir une loi thermo-physique pour l'élément !

28.9.2 HYPO_ELAS2D_C

Identificateur d'une loi hypo-élastique 2D en contraintes planes, isotrope définie par les relations suivantes : $\dot{\mathbf{S}} = \mu \bar{\mathbf{D}}$ et $\dot{I}_\sigma = K_c I_{\mathbf{D}}$. Cette loi convient pour les éléments 2D. Elle nécessite la donnée d'un coefficient de compressibilité tangent K_c , d'un module de cisaillement tangent μ et d'un type de dérivée matérielle pour l'intégration de la loi de comportement.

L'entrée des données suit exactement la même logique que dans le cas 3D.

La table (151) donne un exemple de déclaration basique.

TABLE 148 – Exemple de déclaration de la loi hypo-élastique linéaire 3D.

```
#----- debut cas d'une loi hypo-elastique -----
polymere          HYPO_ELAS3D

# ..... loi de comportement HYPO_ELAS 3D .....
# | coef compressibilite | coef cisaillement | type de derivee objective utilisee |
# |   instantane        |   instantane      | pour le calcul de la contrainte   |
# |       Kc            |           mu      | type_derivee (facultatif)         |
# .....
      Kc= 1000  mu= 500 type_derivee -1
          fin_loi_HYPO_ELAS3D
# le type de derivee est optionnel: = -1 -> derivee de jauman,
# = 0 -> derivee deux fois covariantes (valeur par default),
# = 1 -> derivee deux fois contravariantes
# dans le cas ou l'on veut une valeur differente de la valeur par default il faut mettre
# le mot cle <type_derivee> suivi de la valeur -1 ou 0 ou 1
# la derniere ligne doit contenir uniquement le mot cle:      fin_loi_HYPO_ELAS3D

#----- fin cas d'une loi hypo-elastique polymere -----
```

TABLE 149 – Exemple de déclaration d’une loi hypo-élastique 3D dont les coefficients dépendent de la température selon une courbe repérée par un nom de référence.

```
#----- debut cas d'une loi hypo-élastique  thermodépendante -----
polymere          HYPO_ELAS3D

# ..... loi de comportement HYPO_ELAS 3D .....
# | coef compressibilité      | coef cisaillement | type de dérivée objective utilisée |
# |      instantané          |      instantané   | pour le calcul de la contrainte   |
# |      Kc                  |      mu           | type_dérivée (facultatif)         |
# .....
      Kc= Kc_thermo_dépendant_ courbe3
      mu= mu_thermo_dépendant_ courbe2
      type_dérivée -1
          fin_loi_HYPO_ELAS3D

# le type de dérivée est optionnel: = -1 -> dérivée de jauman,
# = 0 -> dérivée deux fois covariantes (valeur par défaut),
# = 1 -> dérivée deux fois contravariantes
# dans le cas où l'on veut une valeur différente de la valeur par défaut il faut mettre
# le mot cle <type_dérivée> suivi de la valeur -1 ou 0 ou 1"
# pour le module de compressibilité: Kc, il est possible d'indiquer
# que le module sera fourni
# par une loi thermophysique (celle associée au même élément),
# pour ce faire on indique uniquement:
# Kc= Kc_avec_compressibilité_externe
# chaque paramètre (à l'exclusion du cas Kc_avec_compressibilité_externe)
# peut être remplacé par une fonction dépendante de la température
# pour ce faire on utilise un mot cle puis un nom de courbe ou la courbe directement
# comme avec les autres lois de comportement
# exemple pour Kc:          Kc= Kc_thermo_dépendant_ courbe3
# exemple pour mu:         mu= mu_thermo_dépendant_ courbe2
# IMPORTANT: à chaque fois qu'il y a une thermodépendance, il faut passer une ligne
# après la description de la grandeur thermodépendante, mais pas de passage à la
# ligne si se n'est pas thermo-dépendant
# la dernière ligne doit contenir uniquement le mot cle:          fin_loi_HYPO_ELAS3D
#----- fin cas d'une loi isoélastique polymère -----
```

TABLE 150 – Exemple de déclaration d’une loi hypo-élastique 3D dont la compressibilité provient d’une loi thermo-physique

```

#----- debut cas d'une loi hypo-élastique thermodependante -----
polymere          HYPO_ELAS3D

# ..... loi de comportement HYPO_ELAS 3D .....
# | coef compressibilite | coef cisaillement | type de derivee objective utilisee |
# | instantane           | instantane         | pour le calcul de la contrainte   |
# | Kc                   | mu                 | type_derivee (facultatif)         |
#.....
      Kc= Kc_avec_compressibilite_externe  mu= mu_thermo_dependant_  courbe2
      type_derivee -1
      fin_loi_HYPO_ELAS3D

#----- fin cas d'une loi isoelas polymere -----

```

TABLE 151 – Exemple de déclaration d’une loi hypo-élastique 2D en contraintes planes

```

acier          HYPO_ELAS2D_C

# ..... loi de comportement HYPO_ELAS 2D_C .....
# | coef compressibilite | coef cisaillement | type de derivee objective utilisee |
# | instantane           | instantane         | pour le calcul de la contrainte   |
# | Kc                   | mu                 | type_derivee (facultatif)         |
#.....
      Kc= 525000.  mu= 162000
      fin_loi_HYPO_ELAS2D_C

```

28.10 Lois qui ne font rien !

Ils s’agit d’une classes de lois qui sont utiles pour des cas très particuliers : elles ne font rien au niveau mécanique. Le programme se contente de les appeler puis continue. Les lois “qui ne font rien” disponibles sont données dans la table (cf.152).

TABLE 152 – liste des différents lois qui ne font rien, disponibles

identificateur	indications	ref du commentaire
LOI_RIEN1D	1D :	(28.10.1)
LOI_RIEN2D	2D :	(28.10.2)
LOI_RIEN3D	3D :	(28.10.3)

28.10.1 LOI_RIEN1D

Identificateur d’une loi 1D qui ne fait rien. Cette loi ne nécessite aucun paramètre. La table (153) donne un exemple de cette loi.

TABLE 153 – Exemple de déclaration d’une loi 1D ne faisant rien mécaniquement.

```
#----- déclaration d’une loi qui ne fait rien mécaniquement -----  
polymere      LOI_RIEN1D  
#----- fin de la loi -----
```

28.10.2 LOI_RIEN2D

Identificateur d’une loi 2D qui ne fait rien. Cette loi ne nécessite aucun paramètre. La table (154) donne un exemple de cette loi.

TABLE 154 – Exemple de déclaration d’une loi 2D ne faisant rien mécaniquement.

```
#----- déclaration d’une loi qui ne fait rien mécaniquement -----  
polymere      LOI_RIEN2D  
#----- fin de la loi -----
```

28.10.3 LOI_RIEN3D

identificateur d’une loi 3D qui ne fait rien. Cette loi ne nécessite aucun paramètre. La table (155) donne un exemple de cette loi.

TABLE 155 – Exemple de déclaration d’une loi 3D ne faisant rien mécaniquement.

```
#----- déclaration d'une loi qui ne fait rien mécaniquement -----  
  polymere          LOI_RIEN3D  
#----- fin de la loi -----
```

28.11 Loi externe de type Umat

Il s'agit de pouvoir utiliser une loi Umat définie extérieurement à Herezh ou définie par un autre processus Herezh, fonctionnant en parallèle. On se reportera à (5.9) pour avoir plus d'information concernant la mise en route d'un processus Herezh fournissant la loi Umat. La loi peut également être définie par un programme externe en C ou autre langage qui puisse appeler des routines C (exemple : scilab, matlab etc.). La table (156) donne un exemple de déclaration d'utilisation de loi externe par Herezh. Dans cet exemple le nom de la loi est "acier" (paramètre en fait non utilisé), la catégorie par le mot clé "CAT_MECANIQUE" la seule catégorie pour l'instant utilisable, puis on indique la dimension. La déclaration est terminée par le mot clé : "fin_loi_Umat".

Dans le but de tester le bon fonctionnement d'un appel d'Umat, sans utiliser de pipe (donc uniquement à l'aide d'un process d'Herezh++) il est possible de déclarer l'utilisation d'une loi Umat, puis ensuite de la loi Umat elle-même (157).

Viennent ensuite le mot clé "utilisation_umat_interne" puis "fin_loi_Umat". Pour cela on indique le mot clé "utilisation_umat_interne" puis on indique la loi qui sert pour l'Umat (dans l'exemple la loi iso-élastique associée à l'acier). Dans ce cas, le nom de la loi (ici acier) indiqué dans la déclaration de l'Umat, est utilisé pour repérer la loi déclarée par la suite. La déclaration de l'Umat doit donc précéder la déclaration de la loi associée.

Par défaut le nom des pipes d'entrée-sortie est : "Umat_reception_Hz" et "Umat_envoi_Hz". Il est possible de changer le nom de ces pipes. La table (158) donne l'exemple d'une déclaration des noms : "pipe_envoi" et "pipe_reception".

On donne également les routines permettant de construire un programme externe, calculant la loi de comportement. Comme il est dit plus haut, ce programme peut-être quelque, du moment qu'il puisse appeler les routines c servant au dialogue. Le formatage des informations est conforme à celui utilisé par le programme Abaqus, on rappelle donc en (160) les variables utilisées par ce programme. La table (28.11) présente la structuration des données, qui est utilisées pour la sérialisations (transformation dans les pipes en octets banalisés) en entrée et sortie des programmes C qui gèrent le dialogue (28.11) et (161). L'ensemble de ces informations et routines, permet de récupérer dans le programme calculant la loi, les infos classiques et après calcul de la loi, de transmettre les résultats sur le pipes de sortie. On se référera à [Rio et al., 2008] pour plus d'information sur le fonctionnement d'ensemble.

TABLE 156 – Exemple de déclaration d'une loi Umat utilisable par Abaqus.

```
MATE2 LOI_VIA_UMAT
# -----
# |      exemple de loi de comportement defini en Umat      |
# |      via un processus qui dialogue avec herezh++        |
# -----
nom_de_la_loi= acier      categorie=   CAT_MECANIQUE      dim_loi= 3
fin_loi_Umat
```

TABLE 157 – Exemple de déclaration d’une loi Umat utilisé en interne, pour vérification.

```

MATE2 LOI_VIA_UMAT
# -----
# |   exemple de loi de comportement defini en Umat   |
# |   via un processus qui dialogue avec herezh++     |
# -----
nom_de_la_loi= acier   categorie=   CAT_MECANIQUE   dim_loi= 3
utilisation_umat_interne
fin_loi_Umat

#-----
# Nom Materiau | Type loi      | Potentiel |
#-----
      acier      ISOELAS

#-----
#   E   |   nu   |
#-----
      210000.      0.3

```

TABLE 158 – Exemple de déclaration d’une loi Umat avec redéfinition des pipes.

```

MATE2 LOI_VIA_UMAT
# -----
# |   exemple de loi de comportement defini en Umat   |
# |   via un processus qui dialogue avec herezh++     |
# -----
nom_de_la_loi= acier   categorie=   CAT_MECANIQUE   dim_loi= 3
nom_pipe_envoi= pipe_envoi nom_pipe_reception= pipe_reception
fin_loi_Umat

```

TABLE 159 – Définition du stockage des informations d’entrée-sortie pour la création d’une Umat externe au programme Herezh

```

/*****
*   UNIVERSITE DE BRETAGNE SUD (UBS)   --- ENSIBS/UFR SSI LORIENT   *
*****
*   LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)           *
*   Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex   *
*   tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
*****
*   DATE:           08/03/2005                                     *
*                                                           $ *
*   AUTEUR:         G RIO/H LAURENT (mailto:gerard.rio@univ-ubs.fr) *
*                   Tel 0297874571 fax : 02.97.87.45.72           *
*                                                           $ *
*****
*   BUT: échange de structures de données: fortran Umat -> C++   *
*   fonction C a implanter du coté d'abaqus (ou autre prog)      *
*   accès directe aux informations.                               *
*                                                           *
*                                                           $ *
*   ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, *
*   VERIFICATION:                                              *
*                                                           *
*   ! date !   auteur   !           but           !           *
*   ----- *
*   !           !           !           !           *
*                                                           $ *
*   ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, *
*   MODIFICATIONS:                                             *
*   ! date !   auteur   !           but           !           *
*   ----- *
*                                                           $ *
*****/

```

```

#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <fcntl.h> /* Pour O_WRONLY, etc */
#include <unistd.h> /* pour les ordres read write close*/

/* définition d'une union qui lie les réels, les entiers et les caractères */
union Tab_car_double_int
{
    char    tampon[928];
    double  x[116];
    int     n[232];
};
Tab_car_double_int t_car_x_n;

```



```

/* --- def des variables globales (a commenter certaines si elles sont declarees autre part) */
/* nom du tube nommé pour l'envoi des donnees */
char* envoi="Umat_envoi_Hz";
/* nom du tube nomme pour la reception des donnees */
char* reception="Umat_reception_Hz";
/* -- declarations des variables de passages avec un nom assez long pour qu'il n'y ait pas */
/* -- de confusion avec les noms du programme appelant */
/* --- en sortie uniquement */
double* u_herezh_DDSDDT = &t_car_x_n.x[0]; /* 6 */
double* u_herezh_DRPLDE = &t_car_x_n.x[6]; /* 6 */
double* u_herezh_DSDDE = &t_car_x_n.x[12]; /* 36 */
/* --- en entrée - sortie */
double* u_herezh_RPL = &t_car_x_n.x[48]; /* 1 */
double* u_herezh_STRESS = &t_car_x_n.x[49]; /* 6 */
double* u_herezh_SSE = &t_car_x_n.x[55]; /* 1 */
double* u_herezh_SPD = &t_car_x_n.x[56]; /* 1 */
double* u_herezh_SCD = &t_car_x_n.x[57]; /* 1 */
double* u_herezh_DRPLDT = &t_car_x_n.x[58]; /* 1 */
double* u_herezh_PNEWDT = &t_car_x_n.x[59]; /* 1 */
/* --- en entrée seulement */
double* u_herezh_STRAN = &t_car_x_n.x[60]; /* 6 */
double* u_herezh_DSTRAN = &t_car_x_n.x[66]; /* 6 */
double* u_herezh_TIME = &t_car_x_n.x[72]; /* 2 */
double* u_herezh_DTIME = &t_car_x_n.x[74]; /* 1 */
double* u_herezh_Temp = &t_car_x_n.x[75]; /* 1 */
double* u_herezh_DTEMP = &t_car_x_n.x[76]; /* 1 */
double* u_herezh_COORDS = &t_car_x_n.x[77]; /* 3 */
double* u_herezh_DRDROT = &t_car_x_n.x[80]; /* 9 */
double* u_herezh_CELENT = &t_car_x_n.x[89]; /* 1 */
double* u_herezh_DFGRDO = &t_car_x_n.x[90]; /* 9 */
double* u_herezh_DFGRD1 = &t_car_x_n.x[99]; /* 9 */

int* u_herezh_NDI = &t_car_x_n.n[216]; /* 1 */
int* u_herezh_NSHR = &t_car_x_n.n[217]; /* 1 */
int* u_herezh_NTENS = &t_car_x_n.n[218]; /* 1 */
int* u_herezh_NSTATV = &t_car_x_n.n[219]; /* 1 */
int* u_herezh_NOEL = &t_car_x_n.n[220]; /* 1 */
int* u_herezh_NPT = &t_car_x_n.n[221]; /* 1 */
int* u_herezh_LAYER = &t_car_x_n.n[222]; /* 1 */
int* u_herezh_KSPT = &t_car_x_n.n[223]; /* 1 */
int* u_herezh_KSTEP = &t_car_x_n.n[224]; /* 1 */
int* u_herezh_KINC = &t_car_x_n.n[225]; /* 1 */

char* u_herezh_CMNAME = &t_car_x_n.tampon[904]; /* 24 */

```

TABLE 160 – rappel des parametres de passage de la subroutine fortran utilisable par Abaqus pour

```

/* ----- rappel des parametres de passage de la routine fortran ----- */

/*          SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
 1 RPL,DDSDDT,DRPLDE,DRPLDT,STRAN,DSTRAN,
 2 TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,MATERL,NDI,NSHR,NTENS,
 3 NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,CELENT,
 4 DFGRDO,DFGRD1,NOEL,NPT,KSLAY,KSPT,KSTEP,KINC)
C
  INCLUDE 'ABA_PARAM.INC'
C
  CHARACTER*80 MATERL
  DIMENSION STRESS(NTENS),STATEV(NSTATV),
 1 DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
 2 STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),
 3 PROPS(NPROPS),COORDS(3),DROT(3,3),
 4 DFGRDO(3,3),DFGRD1(3,3)
C
  DIMENSION EELAS(6),EPLAS(6),FLOW(6)
  PARAMETER (ONE=1.0D0,TWO=2.0D0,THREE=3.0D0,SIX=6.0D0)
  DATA NEWTON,TOLER/10,1.D-6/
*/

/* ----- fin rappel des parametres de passage de la routine fortran ----- */

```

TABLE 161 – Procédure de lecture des informations sur le pipes

```

/* procedure de lecture de recuperation sur le pipe des infos calculees */
void LectureDonneesUmat(double* STRESS,double* DDSDE
                        ,double* SSE,double* SPD,double* SCD
                        ,const int* NDI,const int* NSHR,const int* NTENS
                        ,double* PNEWDT
                        ,double* RPL,double* DDSDDT,double* DRPLDE,double* DRPLDT)
{
    /* Creation d'un processus de reception des donnees */
    int tub;
    /* ouverture du tube nomme en lecture */
    tub = open(envoi,0_RDONLY);
    /* lecture dans le tampon */
    read (tub,t_car_x_n.tampon,480);
    close (tub); /* fermeture du tampon */
    /* transformation de l'information */
    if ((*NDI == 3) && (*NSHR == 3))
    { /* cas classique 3D */
        int ij;
        for (ij=0;ij<6;ij++)
        {STRESS[ij] = u_herezh_STRESS[ij];
        DDSDDT[ij] = u_herezh DDSDDT[ij];
        DRPLDE[ij] = u_herezh_DRPLDE[ij];
        int kl;
        for (kl=0;kl<6;kl++)
        {int r=ij*6+kl;
        DDSDE[r] = u_herezh DDSDE[r];
        }
        }
    };
    *SSE = *u_herezh_SSE; *SPD = *u_herezh_SPD; *SCD = *u_herezh_SCD;
    *PNEWDT = *u_herezh_PNEWDT;
    *RPL = *u_herezh_RPL;
    *DRPLDT = *u_herezh_DRPLDT;
};

```

TABLE 162 – procédure d’écriture sur le pipe, dans le cas de la création d’une Umat externe à Herezh

```

union Tab_car_et_double
{ char tampon[21];
  double truc[2];
} ;

/* procedure d'écriture sur le pipe des infos passees par le programme principal */
void EcritureDonneesUmat(double* STRESS
                        ,double* SSE,double* SPD,double* SCD
                        ,const double* STRAN,const double* DSTRAN
                        ,const double* TIME,const double* DTIME,const double* Temp,const double* D
                        ,const char* CMNAME
                        ,const int* NDI,const int* NSHR,const int* NTENS
                        ,const int* NSTATV
                        ,const double* COORDS,const double* DROT
                        ,double* PNEWDT,const double* CELENT
                        ,const double* DFGRD0,const double* DFGRD1
                        ,const int* NOEL,const int* NPT,const int* LAYER
                        ,const int* KSPT,const int* KSTEP,const int* KINC)
{
  if ((*NDI == 3) && (*NSHR == 3))
  { /* cas classique 3D */
    int ij;
    for (ij=0;ij<6;ij++)
    {u_herezh_STRESS[ij] = STRESS[ij];
     u_herezh_STRAN[ij] = STRAN[ij];
     u_herezh_DSTRAN[ij] = DSTRAN[ij];
    }
  };
  *u_herezh_SSE = *SSE; *u_herezh_SPD = *SPD; *u_herezh_SCD = *SCD;
  *u_herezh_PNEWDT= *PNEWDT;
  u_herezh_TIME[0]= TIME[0]; u_herezh_TIME[1]= TIME[1]; *u_herezh_DTIME = *DTIME;
  *u_herezh_Temp = *Temp; *u_herezh_DTEMP = *DTEMP;
  *u_herezh_NDI = *NDI; *u_herezh_NSHR = *NSHR; *u_herezh_NTENS = *NTENS; *u_herezh_NSTATV = *NSTATV;
  int i;
  for (i=0;i<3;i++)
    {u_herezh_COORDS[i]= COORDS[i];
  int j;
    for (j=0;j<3;j++)
      {int r=i*3+j;
        u_herezh_DROT[r]= DROT[r]; u_herezh_DFGRD0[r]=DFGRD0[r]; u_herezh_DFGRD1[r]=DFGRD1[r];
      }
    };
  *u_herezh_CELENT = *CELENT;
  *u_herezh_NOEL = *NOEL; *u_herezh_NPT = *NPT; *u_herezh_LAYER = *LAYER; *u_herezh_KSPT = *KSPT;
  *u_herezh_KSTEP = *KSTEP; *u_herezh_KINC = *KINC;
  for (i=0;i<20;i++) u_herezh_CMNAME[i] = CMNAME[i];
}

```

```

/*cout << "\n ***** grandeurs expedier du sous prog c abaqus ***** ";
cout << "\n sigma_t "; for (int i=0;i<6;i++) cout << " ("<<i<<")=" << u_herezh_STRESS[i];
    cout << "\n def "; for (int i=0;i<6;i++) cout << " ("<<i<<")=" << u_herezh_STRAN[i];
    cout << "\n deltatdef "; for (int i=0;i<6;i++) cout << " ("<<i<<")=" << u_herezh_DSTRAN[i];
    cout << "\n rotation "; for (int i=0;i<9;i++) cout << " ("<<i<<")=" << u_herezh_DROT[i];
    cout << "\n gradient_t "; for (int i=1;i<9;i++) cout << " ("<<i<<")=" << u_herezh_DFGRD0[i];
    cout << "\n gradient_tdt "; for (int i=1;i<9;i++) cout << " ("<<i<<")=" << u_herezh_DFGRD1[i];
cout << endl; */
/* Creation d'un processus de pour l'écriture des donnees */
/* ouverture du tube nomme en écriture */
int tub = open(reception,O_WRONLY);
/* écriture dans le tampon */
/* en fait l'objectif est d'écrire que les variables d'entrees ou */
/* d'entree sortie */
char* tampon_envoi = &(t_car_x_n.tampon[384]);
write (tub,tampon_envoi,544); /*sizeof(tampon)); */
close (tub); /* fermeture du tampon */
return;
};

```

29 Liste de lois thermo-physiques disponibles

29.1 Lois isotropes thermiques

Les lois disponibles sont données par la table (cf. 163).

TABLE 163 – liste des lois de comportement thermophysique

nom	commentaire simplifié	référence
LOI_ISO_THERMO	loi simple isotrope 1D 2D 3D	(29.2)
LOI_DE_TAIT	loi de Tait 1D 2D 3D isotrope	(29.3)

29.2 Loi simple isotrope 1D 2D 3D

LOI_ISO_THERMO : identificateur d’une loi simple isotrope adaptée aux dimensions 1 2 et 3. Cette loi convient donc pour la plupart des éléments finis. La table (164) donne un exemple de déclaration de loi à coefficients constants.

TABLE 164 – Exemple de déclaration d’une loi thermo physique simple dont les coefficients sont constants.

```
#----- debut cas d'une loi thermo physique acier -----
acier_therm      LOI_ISO_THERMO

# ..... loi de comportement thermique isotrope .....
# | coefficient de dilatation | conductivite | capacite calorifique |
# | | | | |
# | alphaT /degr | lambda W/mm.deg | cp J/gr.deg |
# .....
alphaT= 12.3e-6 lambda= 0.062 cp= 0.465
fin_thermique_isotrope

#----- fin cas d'une loi thermo physique acier -----
```

Chaque coefficient peut également être dépendant de la température. La table (165) donne un exemple de déclaration de loi à coefficients thermo-dépendants.

Il est possible de récupérer en post-traitement, différentes grandeurs internes à la loi, qui peuvent éventuellement varier dans le temps (parce qu’elles dépendent de la température qui varient par exemple). Les grandeurs disponibles sont : le coefficient de dilatation α , le coefficient de conductivité λ , la capacité calorifique c_p , le coefficient de compressibilité K . Ces grandeurs sont accessibles via le menu “grandeurs particulières” associées aux points d’intégration (dans le format maple par exemple).

Comme pour la loi de Tait (cf. 29.3), il est également possible de calculer un taux de cristallinité en même temps que les autres grandeurs thermo-physiques. Cette prise en compte s’effectue de la manière suivante :

TABLE 165 – Exemple de déclaration d’une loi thermo physique simple dont les coefficients sont thermo-dépendants.

```
#----- debut cas d'une loi thermo physique acier -----
acier_therm2      LOI_ISO_THERMO

# ..... loi de comportement thermique isotrope .....
# | coefficient de dilatation | conductivite | capacite calorifique |
# |                          |              |                       |
# |      alphaT /degr       | lambda W/mm.deg |      cp J/gr.deg      |
# .....
alphaT=  alphaT_thermo_dependant_  courbealpha
lambda=  lambda_thermo_dependant_  courbelambda
cp=      cp_thermo_dependant_      courbecp
              fin_thermique_isotrope

#----- fin cas d'une loi thermo physique acier -----
```

- tout d’abord un indicateur (facultatif) indiquant le type de calcul voulu, par défaut = 0
- type_de_calcul_ =1 → calcul du taux de cristallinité, mais pas de dépendance des variables Thermo-physiques (α , λ , la compressibilité) avec la cristallinité
- type_de_calcul_ =2 → calcul du taux de cristallinité, avec dépendance des variables Thermo-physiques.

Dans le cas où “type_de_calcul_” n’est pas nul, il faut ensuite définir la méthode de calcul de la cristallinité, par exemple pour la loi d’hoffman (cf. 29.4) :

“type_de_calcul_ = 2 Cristallinite_ = HOFFMAN”

On se reportera à la table (168) pour un exemple d’application.

29.3 Loi de Tait 1D 2D 3D

LOI_DE_TAIT : identificateur d’une loi thermophysique de Tait adaptée aux dimensions 1 2 et 3. Cette loi convient pour la plupart des éléments finis. La table (166) donne un exemple de déclaration de ce type de loi.

On remarquera que les coefficients lambda et cp peuvent être dépendant de la température.

Il est possible, en post-traitement de récupérer certaines grandeurs internes à la loi, qui varient éventuellement en cours de calcul. Cependant, l’accès à ces grandeurs nécessite que Herezh++ définit des zones de mémoire, de sauvegarde supplémentaires par rapport à celles nécessaires strictement pour le calcul. Aussi est-il nécessaire d’indiquer dans les paramètres d’entrée de la loi, que l’on utilisera les grandeurs de post-traitement. Pour ce faire, il faut utiliser un drapeau : “prepa_sortie_post=” suivi de valeur “1” (cf. table 167).

Actuellement, les grandeurs disponibles (pour le post-traitement) sont : la température de transition, le volume spécifique, le coefficient de dilatation, la conductivité, la capacité calorifique, et enfin le module de compressibilité. Ces grandeurs sont accessibles via le menu “grandeurs particulières” associées aux points d’intégration (dans le format maple

TABLE 166 – Exemple de déclaration d'une loi thermo physique de Tait.

```
#----- debut cas d'une loi thermo physique -----

poly_therm2      LOI_DE_TAIT

# ----- loi de comportement thermique isotrope avec le comportement d'etat de Tait -----
# | la loi necessite 13 coefficients qui permettent le calcul du volume specifique Vspe |
# | Vspe = V0(T)*(1.-C*log(1.+P/B(T))) + Vt(T,P) avec les relations |
# | C = 0.0894, Ttrans = b5 + b6 * P |
# | si T < Ttrans: V0(T) = b1s+b2s*(T-b5), B(T) = b3s*exp(-b4s*(T-b5)) |
# | si T >= Ttrans: V0(T) = b1m+b2m*(T-b5), B(T) = b3m*exp(-b4m*(T-b5)) |
# | Vt(T,P) = b7*exp(-b8(T-b5)-b9*P) |
# | En fonction du volume specifique on obtient la masse volumique ro=1./ Vspe |
# | et le coefficient de dilatation: alpha = 1./ro * d ro / d T |
# -----

# definition des coefficients
coefficients_bi
b1s= 0. b2s= 0. b3s= 0. b4s= 0.
b1m= 1. b2m= 1. b3m= 1. b4m= 1.
b5= 2. b6= 2. b7= 2. b8= 2. b9= 2.

# puis on definit la conductivite et la capacite calorifique
# .....
# | conductivite | capacite calorifique |
# | | |
# | lambda | cp |
# .....
lambda= 0.062 cp= 0.465

# chaque parametre peut etre remplace par une fonction dependante de la temperature
# pour ce faire on utilise un mot cle puis une nom de courbe ou la courbe directement comme avec
# les autre loi de comportement
# exemple pour la conductivite: mlambda= lambda_thermo_dependant_ courbe2
# exemple pour la capacite calorifique: cp= cp_thermo_dependant_ courbe3

# IMPORTANT: a chaque fois qu'il y a une thermodependence, il faut passer
# une ligne apres la description
# de la grandeur thermodependante, mais pas de passage à la ligne si se
# n'est pas thermo dependant

fin_loi_tait

# la derniere ligne doit contenir uniquement le mot cle: fin_loi_tait
```

par exemple). Dans le cas du calcul de la cristallinité, on a également acces à la valeur du taux de cristallinité.

Il est également possible de calculer un taux de cristallinité en même temps que les autres

TABLE 167 – Exemple de déclaration d’une loi thermo physique de Tait avec préparation pour le posttraitement des résultats.

```
#----- debut cas d'une loi thermo physique -----

poly_therm2      LOI_DE_TAIT

# ----- loi de comportement thermique isotrope avec le comportement d'etat de Tait -----
# | la loi necessite 13 coefficients qui permettent le calcul du volume specifique Vspe |
# | Vspe = V0(T)*(1.-C*log(1.+P/B(T))) + Vt(T,P)      avec les relations |
# | C = 0.0894, Ttrans = b5 + b6 * P |
# | si T < Ttrans: V0(T) = b1s+b2s*(T-b5), B(T) = b3s*exp(-b4s*(T-b5)) |
# | si T >= Ttrans: V0(T) = b1m+b2m*(T-b5), B(T) = b3m*exp(-b4m*(T-b5)) |
# | Vt(T,P) = b7*exp(-b8(T-b5)-b9*P) |
# | En fonction du volume specifique on obtient la masse volumique ro=1./ Vspe |
# | le coefficient de dilatation lineaire : alpha = 1./(3*ro) * d ro / d T |
# | et le coefficient de compressibilite: ksi = -1./ro * d ro / d p |
# -----

      # definition des coefficients
coefficients_bi
b1s= 0. b2s= 0. b3s= 0. b4s= 0.
b1m= 1. b2m= 1. b3m= 1. b4m= 1.
b5= 2.  b6= 2. b7= 2. b8= 2. b9= 2.

# puis on definit la conductivite et la capacite calorifique
# .....
# |   conductivite           |   capacite calorifique   |
# |           |             |           |
# |   lambda                 |   cp                     |
# .....
      lambda= 0.062           cp= 0.465   prepa_sortie_post= 1

      fin_loi_tait

# la derniere ligne doit contenir uniquement le mot cle:      fin_loi_tait
```

grandeurs thermo-physiques. Cette prise en compte s’effectue de la maniere suivante :

- tout d’abord un indicateur (facultatif) indiquant le type de calcul voulu, par default = 0
- type_de_calcul_ =1 → calcul du taux de cristalinité, mais pas de dépendance des variables Thermo-physiques (α , λ , la compressibilité) avec la cristalinité
- type_de_calcul_ =2 → calcul du taux de cristalinité, avec dépendance des variables Thermo-physiques.

Dans le cas où “type_de_calcul_” n’est pas nul, il faut ensuite définir la méthode de calcul de la cristalinité, par exemple pour la loi d’hoffman (cf. 29.4) :

“type_de_calcul_ = 2 Cristalinite_ = HOFFMAN1”

On se reportera à la table (168) pour un exemple d'application.

TABLE 168 – Exemple de déclaration d'une loi thermo physique de Tait avec calcul du taux de cristallinité par le modèle d'Hoffman

```
#----- debut cas d'une loi thermo physique -----

poly_therm2      LOI_DE_TAIT

# ----- loi de comportement thermique isotrope avec le comportement d'etat de Tait -----
# | la loi necessite 13 coefficients qui permettent le calcul du volume specifique Vspe |
# | Vspe = V0(T)*(1.-C*log(1.+P/B(T))) + Vt(T,P)      avec les relations |
# | C = 0.0894, Ttrans = b5 + b6 * P |
# | si T < Ttrans: V0(T) = b1s+b2s*(T-b5), B(T) = b3s*exp(-b4s*(T-b5)) |
# | si T >= Ttrans: V0(T) = b1m+b2m*(T-b5), B(T) = b3m*exp(-b4m*(T-b5)) |
# | Vt(T,P) = b7*exp(-b8(T-b5)-b9*P) |
# | En fonction du volume specifique on obtient la masse volumique ro=1./ Vspe |
# | le coefficient de dilatation lineaire : alpha = 1./(3*ro) * d ro / d T |
# | et le coefficient de compressibilite: ksi = -1./ro * d ro / d p |
# -----

      # definition des coefficients
coefficients_bi
b1s= 0. b2s= 0. b3s= 0. b4s= 0.
b1m= 1. b2m= 1. b3m= 1. b4m= 1.
b5= 2.  b6= 2. b7= 2. b8= 2. b9= 2.

# puis on definit la conductivite et la capacite calorifique \n";
# .....
# |   conductivite           |   capacite calorifique           |
# |           |               |               |
# |   lambda                |   cp                            |
# .....
      lambda= 0.062           cp= 0.465   prepa_sortie_post= 1

      prepa_sortie_post= 0 # valeur par default -> pas de sauvegarde

#---- prise en compte de la cristalinite -----
# il est possible de tenir compte d'un tau de cristalinite de la maniere suivante:
# tout d'abord un indicateur (facultatatif) indiquant le type de calcul voulu, par default = 0
#   type_de_calcul : =1 -> calcul du taux de cristalinite, mais pas de dependance des variables
#                   de ThermoDonnee(alpha, lambda, compressibilite) avec la cristalinite
#                   : =2 -> calcul du taux de cristalinite, avec dependance des variables de
#                   ThermoDonnee
# Dans le cas ou le type_de_calcul n'est pas nul, il faut ensuite definir la methode
# de calcul de la cristalinite
# ex: pour la loi d'hoffman1

      type_de_calcul_= 2 Cristalinite_= HOFFMAN
#
# puis on donne les parametres de la loi d'Hoffman, cf. la documentation

coefficientsLoiHoffman_
      n= 3   G0= 2.83e2  N01= 0.156  N02= 15.1
      Ustar= 6250.  Tinf= 243.  Kg= 5.5e5  Tm= 483.
      alphaP= 0.283e-6   betaP= -2.08e-16

      fin_coefficientsLoiHoffman_

```

29.4 Loi d'Hoffman d'évolution de la cristalinité

Dans l'implantation actuelle, ce type de loi est utilisée conjointement avec une loi thermo-physique de type par exemple loi de Tait (29.3) ou la loi LOLISO_THERMO (29.2). Elle permet le calcul du taux de cristalinité d'un polymère en fonction de la température, et de la pression. Soit $K(T, P)$ la constante cinétique donnée par la loi :

$$K(T, P) = G_0 \left(\left(\frac{4}{3} \pi N_0 \right)^{1/3} \exp \left(- \frac{U^*}{R (T_{corr} - T_\infty)} \right) \exp \left(- \frac{Kg}{(T_{corr} (T_m - T_{corr}))} \right) \right) \quad (64)$$

avec les données suivantes :

- $DT_{corrPres} = P (\alpha_P + P \beta_P)$
- $T_{corr} = T - DT_{corrPres} + 273.$
- $N_0 = \exp(N_{01} (T_m - T_{corr}) + N_{02})$
- et : P la pression relative, T la température celsius et R la constante des gaz parfaits.

Les paramètres de la loi sont : $n, G_0, N_{01}, N_{02}, U^*, T_\infty, Kg, T_m, \alpha_P, \beta_P$

Le taux de cristalinité $c(t)$ se calcul à l'aide de l'expression suivante :

$$c(t) = X_\infty \left\{ 1 - \exp \left(- \left[\int_0^t K(t') dt' \right]^n \right) \right\} \quad (65)$$

X_∞ correspond au taux maximal de cristalinité atteignable pour les conditions données de pression et de température.

La table (169) donne un exemple de description de la loi d'Hoffman.

29.5 Loi d'Hoffman2 d'évolution de la cristalinité

La loi est très proche de la loi d'Hoffman1, si ce n'est la forme de la fonction. Elle permet le calcul du taux de cristalinité d'un polymère en fonction de la température, et de la pression. Soit $K(T, P)$ la constante cinétique donnée par la loi, dans le cas de Hoffman2 nous avons : $f=2*T_{corr}/(T_{corr} + T_m)$

$$K(T, P) = (\log(2))^{1/n} \frac{1}{T_{1/2}} \left(\exp \left(- \frac{U^*}{R (T_{corr} - T_{inf})} \right) \exp \left(- \frac{Kg}{(T_{corr} (T_m - T_{corr}) * f)} \right) \right) \quad (66)$$

avec les données suivantes :

- $DT_{corrPres} = P (\alpha_P + P \beta_P)$
- $T_{corr} = T - DT_{corrPres} + 273.$
- $f = 2. \frac{T_{corr}}{(T_{corr} + T_m)}$
- et : P la pression relative, T la température celsius et R la constante des gaz parfaits.

Les paramètres de la loi sont : $n, T_{1/2}, U^*, T_\infty, Kg, T_m, \alpha_P, \beta_P$

Comme pour Hoffman1, Le taux de cristalinité $c(t)$ se calcul à l'aide de l'expression suivante :

$$c(t) = X_\infty \left\{ 1 - \exp \left(- \left[\int_0^t K(t') dt' \right]^n \right) \right\} \quad (67)$$

TABLE 169 – Exemple de déclaration d’une loi d’Hoffman permettant le calcul de la cristallinité en fonction de la pression et de la température.

```
# ----- loi de Hoffman1 permettant le calcul de K(T,P) -----
# |la loi necessite 10 coefficients  n,G0,N01,N02,Ustar,Tinf,Kg,Tm,alphaP,betaP,X_inf |
# | le calcul de K(T,P) s'effectue alors suivant la formule: |
# | K(T,P) = G0*((4.0/3.0*PI*N0)**(1/3) |
# | *exp(- Ustar/(R*(Tcorr - Tinf)))*exp(- Kg/(Tcorr*(Tm - Tcorr))) |
# | avec les relations: |
# | DTcorr_Pres = P*(alphaP + betaP*P); |
# | Tcorr = T - DTcorr_Pres + 273.; |
# | N0 = exp(N01*(Tm - Tcorr) + N02); |
# | et: P la pression relative, T la temperature celsius |
# -----

# definition des coefficients, ils peuvent être donnés dans un ordre quelconque et
# sur une ou plusieurs lignes

coefficientsLoiHoffman_
  n= 3  G0= 2.83e2  N01= 0.156  N02= 15.1
  Ustar= 6250.  Tinf= 243.  Kg= 5.5e5  Tm= 483.
  alphaP= 0.283e-6  betaP= -2.08e-16  X_inf= 0.684

fin_coefficientsLoiHoffman_

# la derniere ligne doit contenir uniquement le mot cle:  fin_coefficientsLoiHoffman_ "
```

X_∞ correspond au taux maximal de cristallinité atteignable pour les conditions données de pression et de température.

La table (170) donne un exemple de description de la loi d’Hoffman.

30 Liste de lois de frottement disponibles

Le frottement intervient lors du contact entre deux matériaux. Les paramètres matériaux dépendent en général du couple de matériaux en présence. Dans le cas d’Herezh++, le contact est simulé au travers d’“éléments de contact” qui sont constitué d’un noeud, appelé “noeud projectile” et d’une surface “cible”. On parle également de “noeud esclave” et surface maître. La surface est toujours constituée par une frontière d’un élément fini. La loi de frottement est donc associée à “cette” élément fini.

Les éléments de contact se créent et disparaissent au gré de l’apparition ou non de contact. Les lois disponibles sont données par la table (cf. 171).

30.1 Loi de Coulomb

TABLE 170 – Exemple de déclaration d’une loi d’Hoffman permettant le calcul de la cristallinité en fonction de la pression et de la température.

```
# ----- loi de Hoffman2 permettant le calcul de K(T,P) -----
# |la loi necessite 11 coefficients  n,inv012,Ustar,Tinf,Kg,Tm,alphaP,betaP,X_inf |
# | le calcul de K(T,P) s'effectue alors suivant la formule: |
# | K(T,P) = (log(2))**(1/n) * inv012 * exp(- Ustar/(R*(Tcorr - Tinf))) |
# | * exp(- Kg/(Tcorr * (Tm - Tcorr) * f)) |
# | avec les relations: |
# | f=2*Tcorr/(Tcorr + Tm) ; |
# | DTcorr_Pres = P*(alphaP + betaP*P); |
# | Tcorr = T - DTcorr_Pres + 273.; |
# | et: P la pression relative, T la temperature celsius |
# -----
# definition des coefficients, ils peuvent être donnés dans un ordre quelconque et
# sur une ou plusieurs lignes

coefficientsLoiHoffman_
  n= 3 inv012= 1.528e8
  Ustar= 6284. Tinf= 215. Kg= 3.81e5 Tm= 445.
  alphaP= 0. betaP= 0. X_inf= 0.443
fin_coefficientsLoiHoffman_

# la derniere ligne doit contenir uniquement le mot cle: fin_coefficientsLoiHoffman_ "
```

TABLE 171 – liste des lois disponibles de frottement lié au contact

nom	commentaire simplifié	référence
LOI_COULOMB	loi de coulomb 1D 2D 3D	(30.1)

La loi de Coulomb est une loi phénoménologique qui est largement employée. On distingue la loi classique et la loi régularisée.

Dans le cas classique, deux types de régime peuvent survenir : soit les points en contact ne glissent pas l’un par rapport à l’autre, c’est le contact collant, soit il y a glissement. Le comportement est régi par “le coefficient de frottement” qui détermine le cône de frottement. La force de contact est telle qu’elle demeure toujours à l’intérieur du cône : strictement dans le cas du contact collant, sur le cône dans le cas du glissement. Dans ce dernier cas on a la relation :

$$\vec{F}_T = -\mu \|\vec{F}_N\| \cdot \frac{\vec{V}_{rt}}{\|\vec{V}_{rt}\|} \quad (68)$$

\vec{F}_T est la force tangentielle, μ est le coefficient de frottement qui dépend du couple de matériau en contact, \vec{F}_N est la force normale, et \vec{V}_{rt} est la vitesse relative tangentielle du point en contact.

Le coefficient de frottement peut également dépendre de la vitesse : en général au démarrage le coefficient est plus important que lorsque le frottement est en régime établi.

Une modélisation de cette dépendance peut s'exprimer sous la forme suivante :

$$\mu(v) = \mu_S + (\mu_S - \mu_C).e^{-c.v} \quad (69)$$

où μ_S est le frottement statique, μ_C est le frottement cinématique, $v = \|\vec{V}_{rt}\|$ est l'intensité de la vitesse relative, $\mu(v)$ est le coefficient variable.

La table (172) donne un exemple de description d'une loi de Coulomb classique.

TABLE 172 – Exemple de déclaration d'une loi de frottement classique de Coulomb.

```
# -----
# |..... loi de comportement de frottement de coulomb ..... |
# -----
#           exemple de definition de loi
#     mu_statique= 0.3
#     fin_loi_frottement_coulomb_
# -----
# mu_statique= le coefficient de frottement statique
# -----
```

La table (173) donne un exemple de description d'une loi de Coulomb dont le coefficient de frottement dépend de la vitesse tangentielle selon la loi (69).

TABLE 173 – Exemple de déclaration d'une loi de frottement de Coulomb, avec un coefficient de frottement qui dépend de la vitesse tangentielle.

```
# -----
# |..... loi de comportement de frottement de coulomb ..... |
# -----
#     mu_statique= 0.3      mu_cine= 0.2  x_c= 2.
#     fin_loi_frottement_coulomb_
# -----
# mu_statique= le coefficient de frottement statique
# mu_cine= est le coefficient de frottement cinématique
# x_c= regle le passage du frottement statique au frottement cinématique
# selon mu = mu_statique + (mu_statique-mu_cine)* exp(-x_c*V),
# V etant la vitesse tangente
# -----
```

La loi de Coulomb classique introduit une indétermination lorsque l'on se trouve dans le cône de frottement. Ceci peut-être résolu par différentes méthodes générales (cf. théorie du contact) mais il est également possible d'utiliser une loi de contact dite "régularisée" qui permet d'éviter cette indétermination. Dans le cas d'un frottement régularisée, la relation $\vec{F}_T = f(\vec{V}_{rt})$ est toujours vérifiée (il n'y a pas deux régimes) . Cependant la forme de la fonction "f" tend à reproduire le fonctionnement initial de la loi classique de Coulomb.

Normalement, due au fait qu'il n'y ait plus d'indétermination, le calcul global est plus robuste.

Dans le cas d'un calcul régularisé on adopte le comportement suivant :

$$\vec{F}_T = -\mu \cdot \varphi(v) \|\vec{F}_N\| \cdot \frac{\vec{V}_{rt}}{\|\vec{V}_{rt}\|} \quad (70)$$

où la fonction $\varphi(v)$ peut prendre plusieurs formes. Les formes implantées sont extraites de l'ouvrage de Peter Wriggers (Computational Contact Mechanics, Wiley, ISBN 0-471-49680-4).

$$\varphi_1(v) = \frac{v}{\sqrt{(v^2 + \varepsilon^2)}} \quad (71)$$

$$\varphi_1(v) = \tanh\left(\frac{v}{\varepsilon}\right) \quad (72)$$

$$\varphi_1(v) = \begin{cases} -1 & \text{if } v < -\varepsilon \\ \frac{v}{\varepsilon} & \text{if } -\varepsilon \leq v \leq \varepsilon \\ 1 & \text{if } v > \varepsilon \end{cases} \quad (73)$$

La première fonction est qualifiée par l'auteur de "square root regularization", la seconde de "hyperbolic tangent regularization" la dernière de "piecewise polynomial regularization". ε est un paramètre de réglage qui permet de contrôler le passage à la saturation de la loi. Lorsque ε tend vers 0, le comportement tend vers celui du modèle original de Coulomb.

La table (174) donne un exemple de description d'une loi de Coulomb régularisé. On remarque dans les commentaires de l'exemple, qu'il est également possible d'introduire sa propre loi de régularisation $\varphi(v)$.

TABLE 174 – Exemple de déclaration d’une loi de frottement de Coulomb régularisée.

```

# -----
# |..... loi de comportement de frottement de coulomb ..... |
# -----
#   mu_statique= 0.3
#   regularisation= 3   epsilon= 1.
#   fin_loi_frottement_coulomb_
# -----
# regularisation= donne le type de regularisation que l'on desire: en
# fonction de la vitesse
# = 1: régularisation par morceau de droite fonction de la vitesse
# = 2: régularisation quadratique; = 3: régularisation
#     en tangente hyperbolique
# = 4: régularisation par une fonction donnée par l'utilisateur
#     f(norme(vitesse))
# epsilon= est le parametre de réglage de la regularisation
# exemple d'une regularisation de type 4:
#   mu_statique= 0.3
#   regularisation= 4   fonction_regul_   courbe4
#   fin_loi_frottement_coulomb_
# -----
#   comme pour toutes les autres courbes dans les lois de comportement
#   il est possible de declarer directement la courbe au lieu de donner
#   une reference a la fin de la description de la courbe, on doit
#   revenir a la ligne
# -----

```

Septième partie

Informations particulières liées au contexte

31 Stockages divers

Un certain nombre d'entités sont intéressantes à définir globalement à l'aide de liste de référence. Par exemple lorsqu'on introduit un maillage d'élément plaque, il est nécessaire d'en définir l'épaisseur. Cette dernière aurait pu être définie à la lecture de chaque élément. Cette démarche aurait nécessité la duplication de la valeur de l'épaisseur pour chaque élément.

De manière à faciliter l'introduction de telle valeur répétitive, on définit globalement une entité épaisseurs associée à une valeur et à une référence d'élément. Ainsi tous les éléments de la référence se verront associé automatiquement l'épaisseur indiquée.

La syntaxe est donnée par l'exemple de la table (176) dans le cas d'un seul maillage.

TABLE 175 – exemple de définition d'épaisseur et de section.

```
epaisseurs -----
EMAT0001      240.

sections-----
EMAT0002      30000.
```

Cette exemple permet d'introduire une référence d'épaisseurs. Tous les éléments de la référence d'éléments "EMAT0001" auront l'épaisseurs 240. De la même manière il est possible d'introduire également une sections pour un groupe d'élément. Ici tous les éléments de la référence "EMAT0002" auront la section 30000.

D'une manière générale lorsque l'on définit des éléments 2D dans un maillage, il est nécessaire de leur associer une épaisseur. De même, tous les éléments 1D d'un maillage doivent posséder une section. Dans le cas où l'épaisseur ou la section n'est pas définit, les éléments sont considérés comme incomplet, et le calcul ne peut se faire. En général, le programme indique un message d'erreur en se sens.

La syntaxe est donnée par l'exemple de la table (176) dans le cas d'un seul maillage.

TABLE 176 – exemple de définition d'épaisseur et de section dans le cas d'un seul maillage.

```
epaisseurs -----
EMAT0001      240.

sections-----
EMAT0002      30000.
```

Dans le cas où il y a plusieurs maillage il est obligatoire d'indiquer le nom du maillage auquel la référence se rapporte, avant le nom de la référence. La syntaxe est donnée par l'exemple de la table (177) dans le cas de plusieurs maillages.

TABLE 177 – exemple de définition d'épaisseur et de section dans le cas de plusieurs maillages.

```

    epaisseurs -----
nom_mail= piece      EMAT0001      240.

    sections-----
nom_mail= piece      EMAT0002      30000.

```

Remarque : Dans toute la suite nous nous plaçons dans le cas de l'existence d'un seul maillage pour simplifier la présentation. Dans le cas de l'utilisation de plusieurs maillage, il est obligatoire d'ajouter la définitions du nom de maillage.

31.1 Définition de l'épaisseur d'un groupe d'éléments

L'exemple précédent (cf. 176) donne un exemple de l'entrée d'une épaisseur.

NB : Bien noter que tous les éléments qui ont une épaisseur, doivent avoir une épaisseur défini. Il n'y a pas d'épaisseur par défaut. Par contre tous les éléments n'ont pas une épaisseur.

31.2 Définition de la section d'un groupe d'éléments

L'exemple précédent (cf. 176) donne un exemple de l'entrée d'une section.

NB : Bien noter que tous les éléments qui ont une section, doivent avoir une section défini. Il n'y a pas de section par défaut. Par contre tous les éléments n'ont pas une section.

31.3 Définition de la largeur d'un groupe d'éléments

La définition d'une largeur s'effectue à l'aide du mot clé : largeurs, d'une manière identique à celle des épaisseurs (cf. 178).

TABLE 178 – exemple de définition de la largeur d'un groupe d'éléments.

```

    largeurs -----
EMAT0001      24.

```

NB : Bien noter que tous les éléments qui ont une largeur, doivent avoir une largeur défini. Il n'y a pas de largeur par défaut. Par contre tous les éléments n'ont pas une largeur.

TABLE 179 – exemple de définition de la masse volumique d’un groupe d’éléments.

```
      masse_volumique -----  
EMAT0001      24.
```

TABLE 180 – exemple de définition de masse additionnelle sur un groupe de noeud.

```
      masse_addi -----  
N_tout      0.03.
```

31.4 Définition de la masse volumique d’un groupe d’éléments

La définition de la masse volumique s’effectue à l’aide du mot clé : `masse_volumique`, d’une manière identique à celle des épaisseurs (cf. 179).

NB : Bien noter que tous les éléments doivent avoir une masse volumique défini. Il n’y a pas de masse volumique par défaut.

31.5 Définition des masses additionnelles

La définition de la masse volumique s’effectue à l’aide du mot clé : `masse_addi`, d’une manière identique à celle des épaisseurs (cf. 179). La référence doit être une référence de noeud

et non d’élément, car l’objectif est d’installer des masses ponctuelles supplémentaires sur des noeuds.

NB : Cette information est facultative.

Huitième partie

Conditions de contact

32 Introduction

Il est possible de prendre en compte des conditions de contact entre plusieurs solides. Le contact est a priori de type déformable-déformable, c'est-à-dire que deux (ou plus) solides en contact se déforment. Le contact s'active dans la partie "paramètres de contact" cf.(§44).

Ensuite on indique le ou les maillages maîtres c'est-à-dire les maillages dont le déplacement est imposé, ainsi que le ou les maillages esclaves dont les déplacements seront imposés entre autre par des contacts avec les maillages maîtres cf.(32.1). On peut également prendre en compte un auto-contact entre les noeuds et éléments d'un même maillage cf.(32.2). A priori toutes les frontières sont susceptibles d'entrer en contact. Cependant dans le cas de pièces comportant un grand nombre d'éléments frontières, l'examen du contact peut s'avérer particulièrement pénalisant au niveau du temps de calcul. Il est alors intéressant de restreindre les zones candidates au contact cf.(32.3).

Les conditions de contact sont prises en compte de manière exacte selon une méthode qui traduit le contact sous forme de conditions linéaires entre les éléments en contact (méthode originale développée par l'auteur) ou par une méthode de pénalisation. La première méthode fournit des résultats identiques à la méthode classique des multiplicateurs de Lagrange, sans toutefois introduire de degré de liberté supplémentaires ce qui en fait son intérêt. Par contre la structure de la matrice de raideur est modifiée à chaque modification de l'étendue de la zone de contact. La seconde méthode par pénalisation est classique dans le fonctionnement d'ensemble, cependant plusieurs particularités ont été mise en place, en particulier les opérations classiques entre cible et projectile s'effectue via la prise en compte de la trajectoire du noeud contrairement aux méthodes classiques qui utilisent la projection du noeud sur la cible. On se reportera aux paramètres de gestion du contact pour plus d'information cf.(§44).

32.1 Introduction des notions de maillages esclaves et de maillages maîtres

Dans le cas de contact, il est nécessaire de définir un ou plusieurs maillages, successivement les uns après les autres cf.(§11). Ensuite on indique le nombre de maillage esclave selon l'exemple suivant :

```
domaine_esclave
#-----
#  definition du nombre de domaine esclave          |
#-----
1
```

Cette déclaration contient le mot clé "domaine_esclave", suivi d'un entier "n" donnant le nombre de domaine esclave. Le programme considère que les "n" premiers maillages définis sont des maillages esclaves, les maillages restant étant des maillages maîtres.

Remarques

- La déclaration du nombre de maillage esclave s’effectue juste après la définition des maillages et des déplacements solides associés. En particulier, la déclaration doit s’effectuer avec celle des courbes éventuelles.
- Dans le cas d’un contact classique, sans auto-contact, il faut au minimum 2 maillages pour qu’il y ait un contact potentiel,
- dans le cas d’un auto-contact éventuelle, un seul maillage est possible,
- par défaut tous les couples possibles ”maillage esclave” ↔ ”maillage maître”, sont pris en compte pour le contact. Dans le cas d’un grand nombre de maillages, le nombre de combinaisons peut devenir important et donc entraîner des temps de calcul important. Dans le cas où l’on veut limiter le nombre de couples possibles, il faut définir plus précisément les zones de contact cf.(32.3),
- il n’y a pas de limitation sur le nombre de maillages possibles.

32.2 Auto-contact

Il est possible de déclarer des maillages en auto-contact. Cela signifie que des noeuds du maillage peuvent entrer en contact avec des éléments du même maillage. Seule les maillages esclaves peuvent être déclarés en auto-contact. La présence d’auto-contact s’effectue à l’aide du mot clé ”auto_contact” suivit d’un nombre ”m” qui indique que les ”m” derniers maillages esclaves seront en auto-contact.

Remarques :

- La déclaration d’auto-contacts éventuelles doit s’effectuer juste avant la déclaration du chargement dans le fichier ”.info” et avant la déclaration de zones spécifiques de contacts éventuelles cf.(32.3).
- Par défaut, il n’y a pas d’auto-contact (m=0),
- le nombre de maillage en auto-contact doit évidemment être inférieur ou égal au nombre de maillage esclave,
- les zones en auto-contact peuvent-être restreintes à l’aide de la définition de zones potentiels de contact cf.(32.3).

Exemple :

```
auto_contact
#-----
#  definition du nombre de domaine esclave  en auto-contat  |
#-----
      1
```

32.3 Introduction explicite des zones présumées de contact

De manière à diminuer le temps de calcul lié à la recherche des zones de contact, il est possible d’indiquer explicitement les zones qui sont présumées en contact. Cette déclaration s’effectue juste avant le chargement, après les données particulières. Elle est

constituée du mot clé "zone_contact" suivi d'une liste de référence de frontière cf.(§18). Voici un exemple de déclaration :

```
zone_contact -----  
#-----  
# liste des références ou le contact est recherche |  
#-----  
N_deb  F_contact
```

Fonctionnement : pendant la recherche du contact, avant de prendre en compte une frontière, le programme vérifie qu'elle fait partie de la zone de contact ainsi défini. Les références doivent contenir la liste des noeuds esclaves susceptibles d'entrer en contact "et" la liste des frontières (points et/ou lignes et/ou surfaces) des maillages maîtres s'il n'y a pas auto-contact, ou du même maillage esclave s'il y a auto-contact pour ce maillage. Ainsi dans l'exemple précédent, "N_deb" indique une liste de noeud esclave et "F_contact" une liste de surfaces du maillage maître.

Remarques :

- Lorsque le mot clé "zone_contact" existe dans le fichier .info, seules les zones indiquées sont prises en compte à l'exclusion de tous les autres frontières !
- Les références de toutes les frontières et de tous les noeuds peuvent-être automatiquement générées à la lecture du maillage cf.(18.1).

32.4 Paramètres de gestion de la méthode de contact

Les paramètre liés à la gestion des méthodes de contact sont défini dans la zone des paramètres cf.(§44). On se reportera à cette section pour plus d'information.

Neuvième partie
Types d'efforts

33 Conditions de chargement imposé

33.1 Mot clé et exemple du chargement ponctuel

La mise en place d'un chargement ponctuel, s'effectue à l'aide du mot clé "charges". L'exemple de la table (181) montre comment implanter un chargement ponctuelle lorsqu'il n'y a qu'un seul maillage.

TABLE 181 – Exemple de mise en place d'un chargement ponctuel dans le cas d'un seul maillage

```
charges -----
#-----
# Ref noeud | Type de charge | valeurs |
#-----
NI          PONCTUELLE      0. -20340
```

Ici on indique qu'on impose un chargement ponctuelle sur tous les noeuds de la liste de nom "NI". Le vecteur charge a deux coordonnées : 0. en x et -20340 en y. Il faut remarquer que ce type de définition n'est valide qu'en 2D, en 3D il serait nécessaire de donner une troisième composante. Par contre le fait de donner plus de composantes que nécessaire ne gêne pas la lecture des données, seulement il faut se rappeler, que seule les n premières composantes, n étant la dimension, seront prise en compte pour le calcul.

Dans le cas où il y a plusieurs maillage il est obligatoire d'indiquer le nom du maillage auquel la référence se rapporte, avant le nom de la référence. L'exemple de la table (182) montre comment implanter un chargement ponctuelle lorsqu'il y a plusieurs maillages.

TABLE 182 – Exemple le mise en place d'un chargement ponctuel dans le cas où il existe plusieurs maillages

```
charges -----
#-----
# nom du maillage | Ref noeud | Type de charge | valeurs |
#-----
nom_mail= outil   NI          PONCTUELLE      0. -20340
```

Ici "outil" est le nom du maillage auquel se rapporte la référence NI. On peut donc avoir au même nom de référence pour plusieurs maillage.

Remarque : Dans toute la suite nous nous plaçons dans le cas de l'existence d'un seul maillage pour simplifier la présentation. Dans le cas de l'utilisation de plusieurs maillage, il est obligatoire d'ajouter la définitions du nom de maillage

TABLE 183 – Exemple le mise en place d’un chargement ponctuel avec une courbe de charge, un temps mini et un temps maxi

charges -----							
#	Ref face	Type de charge	valeurs	courbe de charge	echelle	temps mini	temps maxi
#	NI	PONCTUELLE	0. -20340 0	COURBE_CHARGE: courbe1	EHELLE: 1.2	TEMPS_MINI= 0.	TEMPS_MAXI= 3.

33.1.1 Chargement particulier suivant une courbe de charge

Durant le calcul, l’application de la charge s’effectue par défaut en fonction d’un ”algorithme de chargement” (cf.). Ceci signifie, que l’intensité des efforts que l’on a prescrit est multipliée par un coefficient, fonction du temps et de l’algorithme de chargement. On se reportera au §() pour la description du fonctionnement des algorithmes de chargement. Cependant il est également possible de définir une courbe de chargement spécifique pour un effort particulier. Par exemple la table (183) indique que l’intensité de la force ponctuelle suit la courbe de charge qui pour nom ”courbe1” (voir 25).

D’une manière identique, on peut indiquer pour tout type d’effort, à la suite des paramètres particulière à la définition de l’effort, la définition d’une courbe de charge et éventuellement d’une échelle. L’intensité finale appliquée sera en fonction du temps : l’effort indiqué multiplié par $f(t)$ multiplié par l’échelle, $f(t)$ étant la valeur de la courbe de charge au temps t considéré. Par défaut l’échelle vaut 1.

33.1.2 Mise en place d’un temps mini et/ou d’un temps maxi

L’exemple de la table (183) montre également la mise en place d’un temps mini et d’un temps maxi. La condition de chargement ne sera activée que lorsque le temps sera compris entre le temps maxi et le temps mini indiqués. Ces deux paramètres sont optionnels, ils peuvent être présent pour tous les types d’efforts, tous les deux, ou un des deux ou encore aucun, dans ce dernier cas par défaut le temps maxi est l’infini et le temps mini 0.

La définition d’un temps mini et/ou maxi doit obligatoirement être faite après la définition éventuelle d’une courbe de charge.

Bien noter que la borne inférieure est exclus de l’intervalle, à l’inverse du temps maxi qui lui est inclus dans l’intervalle.

Le paragraphe(33.2) indique les différents type de chargement que l’on peut introduire.

33.1.3 Particularité de la mise en place d’un chargement actif pour $t=0$

Par défaut le chargement n’est actif que sur l’intervalle $[t_{mini}, t_{maxi}]$ ce qui peut poser un problème lorsque l’on veut un chargement actif pour t strictement nul. Dans ce cas la solution est d’introduire un chargement réglé par une courbe de charge, seul chargement pour lequel on peut changer le temps mini et le temps maxi. Ensuite on définit un temps mini qui vaut un temps négatif très proche de 0 par exemple $t_{mini} = -1.10^{-14}$. Dans ce cas le chargement sera actif pour $t=0$.

33.2 Différent type de chargement

Chaque type de chargement est défini par un nom de référence en adéquation avec le type de chargement, par exemple une référence de noeud pour des charges ponctuelles, ou une référence d'élément pour des charges volumiques ... Puis un identificateur de type de chargement, et enfin les valeurs numériques attachés à dimensionner le chargement.

La liste exhaustive des identificateurs disponibles est donné dans la table (184).

Il est possible évidemment d'avoir plusieurs type de chargement en même temps. Il suffit dans ce cas d'indiquer successivement les différents chargement. Par exemple :

```

charges -----
#-----
# Ref face | Type de charge | valeurs |
#-----
F_02      PRESSDIR      1. 1. 0.
F_02      PRESSION      8.
E_25      VOLUMIQUE     0. 0.1 5.
A_34      LINEIQUE     0. 0. 2.

```

TABLE 184 – liste des différents types de chargement

identificateur	ref du commentaire
PONCTUELLE	(33.1)
VOLUMIQUE	(33.2.1)
LINEIQUE	(33.2.2)
LINEIC_SUIVEUSE	(33.2.3)
UNIFORME	(33.2.4)
PRESSION	(33.2.5)
PRESSDIR	(33.2.6)
PHYDRO	(33.2.7)

33.2.1 Chargement volumique

Identificateur "VOLUMIQUE" : identificateur de charge volumique. Nécessite une référence d'élément, et un vecteur donnant la valeur de la charge volumique dans le repère absolu. La charge sur l'élément dépend donc du volume final de l'élément. En particulier s'il y a changement de volume, il y a changement de charge. Exemple de syntaxe : (185).

33.2.2 Chargement linéique

Identificateur "LINEIQUE" : identificateur de charge linéique. Nécessite une référence d'arrêtes d'élément ou d'élément linéaire et un vecteur donnant la valeur de la charge linéique dans le repère absolu. La charge résultante sur l'élément dépend de la longueur de l'arrête, elle varie donc pendant le calcul. Exemple de syntaxe : (186).

TABLE 185 – Exemple de déclaration d’un chargement volumique

```

charges -----
#-----
# Ref face | Type de charge | valeurs |
#-----
E_25      VOLUMIQUE          0. 0.1 5.

```

TABLE 186 – Exemple de déclaration d’un chargement linéique

```

charges -----
#-----
# Ref face | Type de charge | valeurs |
#-----
A_34      LINEIQUE          0. 0. 2.

```

33.2.3 Chargement : densité linéique qui suit l’évolution de la frontière

Identificateur "LINEIC_SUIVEUSE" : identificateur de charge linéique suiveuse, c’est-à-dire dont la direction varie avec la normale à l’élément. Ce type de chargement n’est défini que pour les éléments 2D, car en 3D la normale à une arête n’est pas définie. Le chargement nécessite une référence d’arêtes d’élément ou d’élément linéaire et un vecteur donnant la valeur de la charge linéique initiale dans le repère absolu. La charge résultante sur l’élément dépend de la longueur de l’arête et de la normale à l’arête, elle varie donc pendant le calcul. Exemple de syntaxe : (187).

TABLE 187 – Exemple de déclaration d’un chargement linéique suiveur

```

charges -----
#-----
# Ref face | Type de charge | valeurs |
#-----
A_34      LINEIC_SUIVEUSE    0. 0.2

```

33.2.4 Chargement : densité d’effort dont la direction reste fixe

Identificateur "UNIFORME" : identificateur de charge surfacique (ou densité) dont la direction reste fixe pendant la déformation. Nécessite une référence de surface et les coordonnées du vecteur densité de charge. La charge résultante sur l’élément dépend de

la surface de l'élément et de l'orientation de cette surface avec la direction de la charge. Exemple de syntaxe : (188).

TABLE 188 – Exemple de déclaration d'un chargement uniforme surfacique

```

charges -----
#-----
# Ref face | Type de charge   |   valeurs |
#-----
F_02      UNIFORME           0. 0.  2.

```

33.2.5 Chargement de type pression

Identificateur "PRESSION" : identificateur de charge surfacique de type pression, c'est-à-dire une charge qui est normale à la surface sur laquelle elle s'applique. La direction du chargement demeure à tout moment normale à la surface, on parle de charge suiveuse. Nécessite une référence de surface et une valeur indiquant la pression que l'on veut exercer. La charge résultante sur l'élément dépend de la surface de l'élément. Exemple de syntaxe : (189).

Il faut noter qu'une pression positive indique que le chargement tend à réduire le volume de la pièce, ainsi le sens de la pression est inverse de celui de la normale! En clair, à la surface du solide on a la relation

$$P = -\mathbf{I}\boldsymbol{\sigma}/dim$$

avec dim la dimension de l'espace de travail.

TABLE 189 – Exemple de déclaration d'un chargement pression

```

charges -----
#-----
# Ref face | Type de charge   |   valeurs |
#-----
F_02      PRESSION           8.

```

33.2.6 Chargement : densité d'effort dont la direction suit l'évolution des frontières

Identificateur "PRESSDIR" : identificateur de charge surfacique de type densité d'effort dont l'orientation initiale par rapport à la surface sur laquelle elle s'applique, demeure fixe durant la transformation mécanique. On a affaire également à une charge suiveuse comme

dans le cas de la pression, mais ici la direction de la densité n'est pas forcément normale à la paroi sur laquelle elle s'applique. Nécessite une référence de surface et un vecteur qui indique la valeur et la direction initiale du chargement. Ensuite cette direction sera amenée à évoluer en fonction de la variation de la surface, par contre l'intensité du chargement reste fixe. Exemple de syntaxe : (190).

TABLE 190 – Exemple de déclaration d'un chargement de type surfacique directionnelle

```

charges -----
#-----
# Ref face | Type de charge | valeurs |
#-----
      F_02      PRESSDIR      1. 1. 0.

```

33.2.7 Chargement hydrostatique

Le type de chargement hydrostatique correspond par exemple à la poussée qu'exerce un liquide sur des parois externes à un solide. Ainsi la définition de ce type de chargement nécessite la référence d'une surface, le mot clef "PHYDRO", la définition de la direction normale à la surface libre du liquide dans le repérage du maillage ce qui permet d'adopter une direction différente de Z, un point de la surface libre du liquide, le poids volumique du liquide (ρg du liquide). Exemple de syntaxe : (191).

TABLE 191 – Exemple de déclaration d'un chargement hydrostatique

```

charges -----
#-----
# Ref face | Type de charge | direction normale | point surf libre | coef hydro |
#-----
      F_02      PHYDRO      0. 0. 1.      0. 0. 0.      4.

```

33.2.8 Chargement aéro-hydrodynamique

Le chargement concerne un solide déformable se déplaçant à une vitesse \vec{V} que l'on considère être sa vitesse par rapport à un milieu fluide englobant et considéré fixe.

Le type de chargement aéro-hydrodynamique est mis en place pour modéliser plusieurs comportements différents :

- les forces dues à un frottement fluide visqueux,
- les forces aérodynamique.

Pour ce faire on considère trois contributions sur la frontière considérée :

1. La première contribution est suivant la direction de la vitesse : de type traînée aérodynamique locale

$$\vec{F}_n = \omega f_n(V) S(\vec{n} \cdot \vec{u}) \vec{u}$$

où ω représente le poids volumique du liquide externe, V est l'intensité de la vitesse ($\|\vec{V}\|$), $f_n(V)$ est une fonction scalaire de l'intensité de la vitesse, fonction quelconque donnée par l'utilisateur permettant ainsi de prendre en compte une non-linéarité du comportement en fonction de la vitesse, S est la surface de la frontière considérée, \vec{n} est la normale à la frontière au point considéré, $\vec{u} = \vec{V}/\|\vec{V}\|$ représente le vecteur unitaire colinéaire avec la vitesse \vec{V} .

2. La deuxième contribution est suivant la normale à la vitesse, de type portance locale

$$\vec{F}_t = \omega f_t(V) S(\vec{n} \cdot \vec{u}) \vec{w}$$

avec $f_t(V)$ une fonction quelconque donnée par l'utilisateur, permettant comme pour $f_n(V)$ de prendre en compte une non-linéarité en vitesse, \vec{w} est unitaire, normal à V et situé dans le plan défini par \vec{n} et \vec{v} . Dans le cas où le vecteur normal est colinéaire avec le vecteur vitesse, on considère qu'il n'y a pas de portance $\vec{F}_t = \vec{0}$

3. La troisième contribution est de type frottement visqueux :

$$\vec{T} = to(V_t) S \vec{u}_t$$

où $\vec{V}_t = \vec{V} - (\vec{V} \cdot \vec{n}) \vec{n}$ est la vitesse tangentielle le long de la surface de la frontière, $\vec{u}_t = \vec{V}_t / \|\vec{V}_t\|$.

En fait, a priori soit les deux premières contributions sont utilisées ce qui est le cas de l'action hydro ou aérodynamique, soit uniquement la dernière qui représente un frottement très visqueux par exemple dans le cas d'un polymère liquide. Cependant il n'y a aucune limitation à utiliser conjointement les trois contributions.

La table (192) donne un exemple de déclaration. Le mot clé "P_HYDRODYNA", doit être suivi du nom de la courbe $to(V_t)$ (défini au début du fichier .info), puis on doit trouver le nom de la courbe $f_n(V)$, ensuite le nom de la courbe $f_t(V)$ et enfin un réel donnant la masse volumique du fluide. Chaque nom de courbe peut être remplacé par le mot clé "NULL" ce qui indique alors que la fonction est nulle quelque soit V , ce qui permet ainsi d'annuler la contribution associée.

TABLE 192 – Exemple de déclaration d'un chargement hydrodynamique

charges -----						
#	Reference	TYPE DE	frot	coeff	coeff	masse
#	surface	CHARGE	fluid to(V)	trainee f_n(V)	portance f_t(V)	volumique
#	F_surface_externe	P_HYDRODYNA	courbe_frot_fluid	courbe_coef_aero_n	NULL	1.e-9

33.2.9 Cas particulier de chargement avec des éléments axisymétriques

Les éléments axisymétriques sont en fait des éléments 3D, dont une dimension est analytique, et les deux autres dimensions sont discrétisées par éléments finis. Aussi, les chargements que l'on impose sont de type 2D, mais sont interprétés comme des chargements 3D selon le fonctionnement général suivant :

- Lorsque l'on indique une charge ponctuelle en un point N_A, celle-ci est interprétée comme répartie sur la ligne générée par une rotation de 2Π autour de l'axe y, du point N_A. La valeur de la charge donnée par l'utilisateur = l'intégrale de la charge répartie.
- Lorsque l'on indique une charge linéique et linéique suivieuse (la charge est un vecteur à trois composantes dont la troisième doit-être nulle) sur une arête A_i, celle-ci est interprétée comme répartie sur la surface générée par une rotation de 2Π autour de l'axe y, de l'arête A_i. Mais contrairement au cas de la force ponctuelle, la valeur de la charge linéique " q_l " donnée par l'utilisateur = la charge par unité de surface. La charge imposée sera donc au finale : $\int_{ligne} q_l 2 \Pi r dl$
- Il est possible d'indiquer une charge de type "PRESSION" (la charge est un scalaire) sur une arête A_i. Dans ce cas, la charge pression s'applique sur la surface générée par une rotation 2Π autour de l'axe y, de l'arête A_i. La pression donnée en argument (= la valeur du scalaire donnée après le mot clé "PRESSION") est interprétée exactement de la même manière que la charge linéique précédente.
- Lorsque l'on indique une charge surfacique (mot clé "UNIFORME") sur une surface F_i, celle-ci est interprétée comme répartie sur le volume généré par une rotation de 2Π autour de l'axe y, de la surface F_i. Comme pour la charge linéique, la valeur de la charge surfacique " q_s " donnée par l'utilisateur = la charge par unité de volume. Ainsi la charge imposée sera donc au finale : $\int_{surface} q_s 2 \Pi r ds$
- on n'indique pas de charge volumique.

Remarque concernant le nombre de points d'intégration pour les conditions limites

Supposons que l'on veuille un chargement uniforme sur une face d'un cylindre étudié avec des éléments axisymétriques. D'une manière générale nous avons :

$$\text{résidu}(V_{ar}^*) = \sum_{i=1}^{i=npti} \vec{F} \cdot \vec{I}_a \varphi_r \sqrt{g(i)} \rho i ds_i \quad (74)$$

Il faut alors considérer 3 points :

1. le résidu au noeud, résultant de la charge uniforme, n'est pas forcément également réparti sur les noeuds, ceci est due à la position des points d'intégration qui sont relatifs à un volume qui dépend de l'excentrement. Supposons par exemple un maillage en quadrangles axi. On a par exemple 4 points d'intégration dans la surface, deux à un rayon r1 et 2 à un rayon r2. Le jacobien volumique attaché aux points d'intégration = jacobien de surface. $\pi.r$. Ainsi le jacobien volumique sera différent pour r1 et r2. Conséquence, s'il y a un seul élément suivant le rayon, on observera un résidu différent pour les 2 noeuds internes (les plus près de l'axe) et les noeuds externes (les plus éloignés de l'axe).

2. d'une manière générale en 2D classique (membrane par exemple) lorsque l'on assemble plusieurs éléments, qui par exemple individuellement supportent la même charge uniforme, on obtient pour les noeuds extrêmes une valeur plus faible que pour les noeuds internes. Cela provient du fait que pour les noeuds internes, plusieurs éléments contribuent, alors que pour les noeuds extrêmes, le nombre d'éléments qui contribuent est plus faible. Par exemple, dans un cas très simple 1D, avec deux éléments biellettes, le noeud interne aura un résidu 2 fois plus grand que les noeuds externes, et ceci si l'on a un chargement uniforme.
3. l'exactitude de l'intégration, dépend du nombre de points d'intégration. Actuellement, ce nombre ne correspond pas toujours à une intégration exacte, ceci pour des économies de temps calcul. Il faut se reporter aux différents nombres attachés à l'élément pour connaître le nombre de points d'intégration réellement utilisé.

Calculons l'élément de surface correspondant à l'élément linéique dr . On a : $ds = dr \cdot r \cdot 2 \cdot \pi$ Supposons que l'on veuille une interpolation linéique : linéaire. Cela signifie que r est linéaire donc que ds est quadratique. Ainsi pour une intégration exacte d'un chargement uniforme, il faut au minimum 2 pt d'integ de Gauss. Supposons que l'on veuille une interpolation linéique : quadratique. Cela signifie que ds est de degré 4, il faut 3 points d'intégration etc.

En résumé, plusieurs facteurs contribuent au fait que même pour un chargement uniforme, les résidus résultants aux noeuds ne sont pas identiques. Néanmoins, si l'on veut exactement représenter cette condition de chargement, il faut veiller à utiliser un nombre suffisant de points d'intégration (cf. les nombres attachés à l'élément).

Dixième partie

Conditions limites en déplacements

34 Conditions limites pour les degrés de liberté (déplacements, positions ...)

Les conditions limites sur les degrés de liberté consistent à imposer soit une valeur numérique aus ddl concernés, soit une fonction de charge. Ceci s'effectue à l'aide du mot clé "blocage", cf. l'exemple (193).

TABLE 193 – Exemple de déclaration des blocages des degrés de liberté dans le cas d'un seul maillage

```

                                blocages -----
#-----
# Ref noeud | Bloquages
#-----
N_W          UX,UY,UZ
N_S          'UZ=3.',UY
N_3  'UX= COURBE_CHARGE: courbe1 ECHELLE: 1.2' TEMPS_MINI= 0. TEMPS_MAXI= 3.

```

Dans cet exemple les noeuds de la liste N_W ont les degrés de libertés en x,y,z bloqués, les noeuds de la liste N_S ont le déplacement suivant y bloqué, et le déplacement suivant z est imposé à 3. La dernière ligne indique que les degrés de liberté de déplacement en x pour les noeuds de la liste N_3, seront soumis à une courbe de charge, dont le nom est courbe1. Cette condition sera active lorsque le temps sera compris entre 0 (exclu) et 3secondes.

Actuellement les degrés de libertés que l'on peut imposer en statique sont : UX pour le déplacement en x, UY pour le déplacement en y, et UZ pour le déplacement en z. En dynamique s'y ajoutent les vitesses suivant x y z : V1 V2 V3 et pour les accélérations : gamma1 gamma2 gamme3.

Pour toutes ces grandeurs, le fait d'indiquer le nom du ddl, sans valeur associée, signifie que la valeur imposée est 0 par défaut. Une valeur non nulle sera spécifié entre cote ex : 'UX=2.'

L'application d'un blocage non nul sans courbe de charge s'effectue de manière pratique suivant la courbe de chargement globale (cf.37). L'application du chargement s'effectue par incrément, qui s'ajoutent à la valeur existant auparavant c'est-à-dire à "t".

Dans le cas de l'application d'un déplacement imposé $U(t)$ en fonction d'une courbe de charge, la position finale est par défaut $X(0) + U(t)$.

Il est également possible d'indiquer un déplacement relatif de "t" à "t + Δt" dans ce cas nous avons à la fin d'un incrément : $X(t + \Delta t) = X(t) + U(t + \Delta t) - U(t)$. Pour cela il faut utiliser un mot clé additionnel "blocage_relatif_" à la suite des temps mini et maxi (cf. l'exemple 195).

Remarque : Attention !, le blocage relatif ne fonctionne que dans le cas d'utilisation de courbe de charge indivituelle. Donc ne fonctionne pas avec les valeurs fixes!!

NB : Sur certain clavier il y a deux types de simple cote, ' et ' . Pour trouver le bon symbole il n'y a que l'essai!

Dans le cas où il y a plusieurs maillage, il est nécessaire (obligatoire) d'indiquer avant la référence, le nom du maillage auquel cette référence est attachée (cf.(194))

TABLE 194 – Exemple de déclaration des blocages des degrés de liberté dans le cas de plusieurs maillages

```

                blocages -----
#-----
#  nom du maillage | Ref noeud |  Blocages
#-----
nom_mail= outil      N_W      UX,UY,UZ
nom_mail= outil      N_S      'UZ=3.',UY
nom_mail= piece      N_S      'UZ=4.',UY
nom_mail= piece      N_3      'UX= COURBE_CHARGE: courbe1 ECHELLE: 1.2' TEMPS_MINI= 0. TEMPS_MAXI= 3.

```

TABLE 195 – Exemple de déclaration de degrés de liberté imposé suivant une courbe de charge, selon une procédure relative $X(t + \Delta t) = X(t) + U(t + \Delta t) - U(t)$

```

                blocages -----
#-----
#  nom du maillage | Ref noeud |  Blocages
#-----
nom_mail= piece N_3 'UX= COURBE_CHARGE: c1 ' TEMPS_MINI= 0. TEMPS_MAXI= 3. BLOQUAGE_RELATIF_

```

Remarque : Dans toute la suite nous nous plaçons dans le cas de l'existence d'un seul maillage pour simplifier la présentation. Dans le cas de l'utilisation de plusieurs maillage, il est obligatoire d'ajouter la définitions du nom de maillage

34.1 Chargement particulier suivant une courbe de charge

Au lieu d'indiquer une valeur bloquée, il est également possible de mentionner le nom d'une courbe de charge préalablement définie dans la liste de courbe 1D(cf. 25). Dans ce cas la valeur de blocage sera calculée à chaque temps à l'aide de la fonction de charge. De plus, il est possible d'indiquer un facteur d'échelle multiplicatif à la fonction de charge, ceci à l'aide du mot clé ECHELLE : suivi de la valeur du facteur d'échelle voulue. Par défaut le facteur d'échelle vaut 1.

En définitif le blocage au temps t sera pour une fonction de charge f et un facteur d'échelle α par exemple : $\alpha f(t)$. Bien noter que dans ce cas la courbe de charge globale (cf.37)

n'intervient pas dans le calcul, par contre le coefficient multiplicatif global (mot clé MULTIPLICATEUR cf.210) lui intervient.

34.2 Mise en place d'un temps mini et d'un temps maxi pour l'application des conditions

Pour chaque enregistrement, on peut indiquer un intervalle de temps pendant lequel la condition est valide. En dehors de cette intervalle, le noeud sera considéré libre de conditions limites. Par défaut cette intervalle est $]0, \infty[$. Une borne inférieure différente s'indique par le mot clé TEMPS_MINI= suivi de la valeur du temps mini. D'une manière analogue une borne supérieure peut-être mentionnée à l'aide du mot clé TEMPS_MAXI= suivi de la valeur du temps maxi. En dehors de cette intervalle, le degré de liberté est libre. Bien noter que la borne inférieure est exclus de l'intervalle, à l'inverse du temps maxi qui lui est inclus dans l'intervalle.

Enfin il faut veiller à ne pas indiquer deux blocages sur un même ddl en même temps. Si cela se produit, le programme signal l'erreur et s'arrête. Cette erreur peut survenir par exemple lorsque l'on indique plusieurs différents sur un même noeud, pour des intervalles de temps différents et que par erreur l'intersection des intervalles n'est pas nulle.

NB : Dans le cas de l'utilisation d'une fonction de charge, la valeur utilisée pour le blocage est celle calculée au temps t ajouté à celle au temps 0. Par exemple dans le cas de ddl de déplacement, la position imposée est celle du temps 0 plus la valeur calculée avec la fonction de charge muni du coefficient d'échelle et du coefficient multiplicateur global. Dans le cas on l'on indique également une condition initiale sur le noeud (cf.36), il faut veiller à ce qu'elle soit cohérente avec la condition d'évolution pour éviter des à-coups.

34.3 Notion de données et de variables

Une grandeur est considérée soit comme une donnée soit comme une variable. Lorsque la grandeur fait partie des inconnues du problème, est a le statut de variable. Lorsqu'elle est consultée durant le déroulement elle a le statut de donnée.

Lorsque l'on indique des conditions limites sur une grandeurs, dans un premier temps le programme considère que ces grandeurs sont des données fixées par l'utilisateur. Dans la suite du déroulement, plusieurs cas sont à considérer.

1. soit à aucun moment les données ou ne sont consultées ou utilisées, dans ce cas les grandeurs sont déclarées hors-service. Par exemple si des noeuds ne sont jamais utilisées, ces noeuds sont ignorés.
2. soit au moment de l'initialisation certains objets (éléments finis, lois de comportement ...) indiqués qu'ils vont utiliser les grandeurs, dans ce cas les grandeurs sont mise en service.

Ainsi durant le déroulement du calcul, seules les grandeurs en service sont utilisés. Le statut de donnée ou variable dépend par exemple du type de calcul que l'on effectue. Par exemple en mécanique classique, la température peut être une donnée alors que la position peut être une variable. Dans le cas d'un problème de thermique stationnaire, la température est une variable alors que la position est une donnée.

34.4 Prise en compte d'un champ de valeurs

Il est possible de définir un champ de valeur pour une référence au lieu d'une grandeur unique. Prenons par exemple le cas indiqué sur le tableau (196). Dans cette exemple N_tout est référence de 2 noeuds. Ensuite entre les deux mots clés "champ_de :" et "fin_champ_de_" sont indiquée une suite de ddl dont le nombre doit être exactement le même que celui du nombre de noeud. Ainsi le premier noeud de la référence N_tout comprendra un ddl bloqué de température à 10, le second à 20. L'ensemble du champ sera activé entre les temps mini 0.5 s et 3. s.

TABLE 196 – Exemple de déclaration des bloquages des degrés de liberté dans le cas d'un champ de valeurs fixes

```
#-----  
# Ref noeud | Bloquages  
#-----  
N_fi          'UX= 2.'  
N_tout champ_de:  
              'TEMP=10 '  
              'TEMP=20 '  
fin_champ_de_ TEMPS_MINI= 0.5  TEMPS_MAXI= 3.
```

Il faut noter qu'ici le chargement suit l'algorithme générale de mise en place du chargement.

Il est également possible de définir un champ de valeur qui utilisent chacune une courbe de charge. Le tableau (197) donne un exemple de ce type de déclaration. Dans cet exemple

TABLE 197 – Exemple de déclaration des bloquages des degrés de liberté dans le cas d'un champ de valeurs déterminées par des courbes de charge

```
#-----  
# Ref noeud | Bloquages  
#-----  
N_fi          'UX= 2.'  
N_tout champ_de:  
              'TEMP=COURBE_CHARGE: c_temp_1 '  
              'TEMP=COURBE_CHARGE: c_temp_2 '  
fin_champ_de_ #TEMPS_MINI= 0.5  TEMPS_MAXI= 3.
```

les courbe de nom c_temp_1 et c_temp_2 doivent au préalable avoir été évidemment défini.

34.5 Cas d'un ddl imposé en initialisation et au cours du temps : exemple de la température

Le chargement est en fait l'association des conditions limites et des conditions qui évoluent avec le temps. Supposons par exemple que l'on fixe une température initiale à 200°C et une courbe de chargement en température qui évolue de 0°C à 100°C lorsque le temps évolue linéairement de 0 à 10s . Dans ce cas la température réellement imposée sera de 200°C+ t.100°C, avec t le temps. On voit donc que la condition de température imposée est en fait une condition de "variation" de température imposée. Dans le cas où il n'y a pas de condition d'initialisation, par défaut la température initiale est mise à 0. Dans ce cas, tout ce passe "comme si" on imposait réellement la température et non la variation de température, cependant il faut bien noter que c'est une conséquence d'une initialisation par défaut, et non le mécanisme réel.

Il en est de même pour tous les ddl, sauf pour les ddl de position : Xi, qui eux sont systématiquement initialisés au moment de la lecture des coordonnées des noeuds (lire le paragraphe (36.1) pour plus d'informations).

Toujours pour le cas de la température, bien noter que si l'on tient compte de la dilatation, celle si dépend de la variation de température entre la température finale et la température initiale c'est-à-dire en reprenant l'exemple précédent :

$$(200^{\circ}C + t.100^{\circ}C) - (200^{\circ}C) = t.100^{\circ}C$$

On voit donc que pour la dilatation, la température initiale n'intervient pas, c'est directement la variation que l'on impose qui est utilisée!!

34.6 Cas de la mise en place d'un mouvement solide

Il est possible d'imposer à un groupe de noeud, un déplacement solide. La table (198) donne un exemple de mouvement solide global. La syntaxe suit la même logique que pour les mouvements initiaux appliqués aux maillages (cf. 12). On définit une référence de noeuds, et ensuite une suite de mouvements solides : translation, nouveau centre de rotation, rotation. On peut enchaîner autant d'opérations que l'on souhaite . Ces opérations seront appliquées dans l'ordre où elles sont lues. **NB** Après le mot clé "mouvement_solide_" on passe une ligne, et chaque opération est sur une seule ligne, sur la dernière ligne on indique le mot clé "fin_mouvement_solide_"

TABLE 198 – Exemple de déclaration d'un déplacement imposé par mouvements solides

```
#-----
# Ref noeud | Bloquages
#-----
N_bout   mouvement_solide_   # def de mouvements solides
          translation_=  0.   0.   0.2
          centre_=      10.  0.   0.
          rotation_=    0.   0.2  0.
          fin_mouvement_solide_
```

Il est également possible de définir une courbe de charge global, qui agira comme une courbe d'amplitude des différentes opérations que l'on a défini. Cependant, la modulation ne concerne évidemment pas les changements de centre, qui s'appliquent toujours intégralement. On peut également indiquer une échelle, un temps mini et un temps maxi. Toutes ces informations doivent être indiquées sur la même ligne qui contient le mot clé "fin_mouvement_solide_" selon la procédure habituelle et en respectant l'ordre. La table (199) donne un exemple complet.

TABLE 199 – Exemple de déclaration d'un déplacement imposé par mouvements solides

```
#-----
# Ref noeud | Bloquages
#-----
      N_bout   mouvement_solide_   # def de mouvements solides
                translation_=  0.   0.   0.2
                centre_=     10.  0.   0.
                rotation_=    0.   0.2  0.
fin_mouvement_solide_ COURBE_CHARGE: courbe_amplitude ECHELLE: 1.3 TEMPS_MINI= 0.5 TEMPS_MAXI= 3.
```

34.7 Condition de symétrie et encastrement pour les SFE

Dans le cas des éléments SFE, il n'y a pas de degré de liberté de rotation. Les conditions d'encastrement s'effectuent à l'aide d'une condition de blocage sur les déplacements des noeuds d'arêtes encastrees plus une condition de direction imposée à la tangente de surface. Ces conditions peuvent également être imposées via les conditions initiale (cf.36), dans ce cas elles sont imposées au début du calcul, et ne varient plus ensuite. Par contre dans le cadre de conditions limites bloquées, elles peuvent évoluer en fonction du temps (par exemple la tangente peut évoluer).

La condition de direction imposée à la tangente de la surface permet également d'imposer des conditions de symétrie, nous allons donc commencer par cette seconde condition qui est commune aux deux conditions limites.

34.7.1 Direction de tangente imposée

La condition s'applique à des références d'arêtes d'éléments. Elle consiste à imposer la direction de la tangente à la surface courbe médiane de la coque. Cette direction est obligatoirement contenue dans un plan normal à l'arête. Pour la définir, l'utilisateur indique un vecteur \vec{w} et la direction de la tangente sera

$$\vec{d} = \vec{w} \times \vec{u}$$

avec \vec{u} la direction de l'arête. Cette technique permet de définir une tangente qui s'adapte au contour. Par exemple si l'on veut définir une tangente horizontale pour un contour circulaire, contenu dans le plan xy, on introduit : $\vec{d} = \vec{z}$.

La table (200) indique un exemple de condition.

A_externe : est le nom d'une référence d'arête (la première lettre est "A"),
typeCL_ : un mot clé qui indique que l'on introduit un type particulier de condition limite. Ce mot clé est suivi d'un identificateur qui peut être dans la version actuelle un des deux suivants :

TANGENTE_CL : indique une condition de tangente imposée d'où une symétrie locale,

RIEN_TYPE_CL : indique aucune condition,

Dans le cas d'une tangente imposée, on indique ensuite les coordonnées de d. *Le dernier cas est le cas par défaut, pour toutes les autres conditions limites, il est donc traité selon les conditions habituelles. Mais actuellement, les conditions classiques sur des arêtes ne servent à rien dans Herezh++, il y aura donc un message d'erreur.*

X1 : la première composante sera nulle. En fait ceci est inutile, car par défaut les trois composantes sont nulles,

X2= COURBE_CHARGE : c1.2 : la seconde composante suivra l'évolution d'une courbe de charge,

X3=1. : la troisième composante est fixée à 1.

TEMPS_MINI= 0. TEMPS_MAXI= 3. : l'intervalle de temps pendant lequel la condition est active.

TABLE 200 – Exemple de déclaration d'une direction de tangente imposée à une surface moyenne d'une coque SFE

```
#-----
# Ref aretes | mot cle      | Bloquages
#-----
A_externe   typeCL_ TANGENTE_CL   X1, 'X2= COURBE_CHARGE: c1.2' 'X3=1.' TEMPS_MINI= 0. TEMPS_MAXI=
```

La condition de symétrie, s'indique donc à l'aide d'une direction tangente normale au plan de symétrie.

34.7.2 Encastrement imposé

Comme il a été indiquée en introduction, l'encastrement s'impose via une condition de symétrie plus une condition de déplacement imposé. Cette dernière s'exprime sur les noeuds, selon les méthodes générales indiquées plus haut. Il est donc nécessaire d'imposer 2 conditions, pour obtenir l'encastrement !

34.8 Cas particulier de géométries et chargements axisymétrique

Dans le cas de géométries et chargements axisymétriques, la dimension d'espace à utiliser est "3D". Cependant, la discrétisation utilise des éléments finis à supports 2D ou 1D, et la géométrie est décrite dans le plan xy. L'axe de symétrie en rotation est l'axe y.

Pendant une déformation ou un déplacement, due aux symétries, il ne doit pas y avoir de déplacement suivant l'axe z aussi il faut bloquer impérativement tous les ddl "UZ". Ceci évite par exemple une singularité de la matrice de raideur dans le cas de la recherche de solution statique.

35 Conditions limites linéaires (CLL) entre degrés de liberté (déplacements, positions ...)

Cette partie concerne la possibilité d'introduire des conditions limites linéaires entre des degrés de libertés existant. Ces conditions sont applicables aux calculs statiques et dynamiques implicites. Dans le cas de calculs dynamiques explicites, il est actuellement possible d'introduire des conditions linéaires sur les accélérations. Le cas des autres ddl, bien que possible, n'est pas actuellement traité.

Actuellement deux principales options sont disponibles avec plusieurs conditions de contrôles possibles. Il faut noter que ces conditions sont complexes, et leur manipulations peuvent-être délicates.

La première option concerne une condition linéaire entre des degrés de liberté de position ou de déplacement d'un même noeud, correspondant à un positionnement sur un plan en 3D ou une droite en 2D.

La seconde option concerne des degrés de libertés quelconques, mais d'une même famille, pouvant appartenir à plusieurs noeuds. Bien que la seconde option contienne a priori la première, dans l'utilisation courante, il est difficile de produire la première option à l'aide de la seconde, dans le cas d'un déplacement du plan de projection (ou droite en 2D).

Les conditions linéaires sont groupé dans un ensemble qui démarre par le mot clé "condition_limite_lineaire_", au même titre que les conditions limites classiques et les conditions initiales. Cet ensemble peut contenir plusieurs conditions linéaires, sa position dans le fichier .info, doit se situer entre les condition limites classiques et les conditions initiales.

35.1 Déplacement ou positionnement dans un plan (3D) ou sur une droite (2D)

Pour fixer les idées, prenons un exemple (201).

TABLE 201 – Exemple de mise en place de conditions linéaire par projection sur un plan, avec utilisation de fonction de charge et centre fixe

```
condition_limite_lineaire_
#-----
# Noeuds   blocage
#-----
N_droit   enu= X1
def_auto_coef_planOuDroite_ centre_fixe_ 0. 0. 0. coefficients= 1. 0. 0.4 fin_list_coefficients_ AvecFonctionsDeCharges_
deb_fonction_de_charge= cfixe cfixe c45monte fin_list_fonctions_de_charges_
```

La condition se situe ici dans un espace 3D (le cas 2D sera explicité par la suite). Dans l'exemple une seule condition est indiquée. On trouve tout d'abord une référence de noeud, ici "N_droit", qui peut-être précédé par un nom de maillage, ce qui permet de choisir entre plusieurs maillage si le cas se présente. Ensuite on trouve une référence de type de degré de liberté ici "X1" précédé obligatoirement du mot clé "enu=". "X1" indique que la condition linéaire concerne la famille des positions, donc pas uniquement "X1", mais également "X2" et "X3". Ensuite sur la ligne qui suit on trouve le mot clé "def_auto_coef_planOuDroite.". C'est ce mot clé qui permet de faire la différence entre les deux types de conditions limites présentés en introduction. Suit la définition d'un point fixe du plan, ici "0. 0. 0." précédé par le mot clé "centre_fixe_", puis les composantes de la normale au plan (qui n'ont pas besoin d'être normées), ici "1. 0. 0.4" précédé par le mot clé obligatoire "coefficients=" et terminés par le mot clé "fin_list_coefficients.". Ensuite, dans l'exemple on souhaite que les coefficients du plan évolue pendant le calcul suivant des fonctions de charge, on indique donc le mot clé optionnel "AvecFonctionsDeCharges_" suivi sur la ligne suivante d'une liste de nom de fonction de charge encadrée par deux mots clés obligatoire "deb_fonction_de_charge=" et "fin_list_fonctions_de_charges.". On remarque qu'il y a 3 fonctions de charge, ce qui est obligatoire (on ne peut pas en mettre 1 ou 2). Les trois fonctions de charge servent à multiplier les composantes de la normale du plan. Supposons que les fonctions de charge soient les suivantes :

```

cfixe COURBEPOLYLINEAIRE_1_D
  Debut_des_coordonnees_des_points
    Coordonnee dim= 2 0. 1.
    Coordonnee dim= 2 1. 1.
  Fin_des_coordonnees_des_points

c45monte COURBEPOLYLINEAIRE_1_D
  Debut_des_coordonnees_des_points
    Coordonnee dim= 2 0. 0.
    Coordonnee dim= 2 1. 1.
  Fin_des_coordonnees_des_points

```

Durant l'application du chargement, les composantes de la normale seront : "(1. * cfixe(t), 0. * cfixe(t), 0.4 * c45monte(t)) ". On modélise ainsi une rotation autour de l'axe z, de tous les noeuds de la référence "N_droit". On niveau du calcul, que ce passe-t-il : à chaque incrément, les noeuds sont projetés sur le plan, puis au cours des itérations d'équilibres, on leur impose de se déplacer dans le plan. Ainsi, pour garantir un chargement progressif, il est préférable que le plan ne soit pas trop distant de la position du noeud.

Il est également possible d'indiquer comme point du plan la position d'un noeud, soit initiale, soit à l'instant t. Un exemple de syntaxe est donné par la table (202).

Pour la première condition on a choisit la position initiale du noeud "2" du maillage "plaque". Pour la seconde condition, il s'agit de la position à t du noeud 8 du maillage courant.

TABLE 202 – Exemple de conditions linéaires par projection sur un plan, avec un centre noeud initial et à t

```

condition_limite_lineaire_
#-----
# Noeuds   blocage
#-----
N_droit   enu= X1
def_auto_coef_planOuDroite_ centre_noeud_a_t0_ nom_mail= plaque 2 coefficients= 1. 0. 0.4 fin_list_coefficients_ AvecFonctionsDeCharges_
deb_fonction_de_charge= cfixe cfixe c45monte fin_list_fonctions_de_charges_
N_gauche  enu= X1
def_auto_coef_planOuDroite_ centre_noeud_a_t_ 8 coefficients= 1. 0. 0.4 fin_list_coefficients_ AvecFonctionsDeCharges_
deb_fonction_de_charge= cfixe cfixe c45monte fin_list_fonctions_de_charges_

```

Dans le cas 2D, toutes les grandeurs doivent être données en 2D (2 coordonnées). De plus les coefficients (2 en 2D) représentent la direction de la droite et non la normale comme dans le cas 3D. A part ces deux particularités, le reste du fonctionnement est identique au cas 3D.

Par comparaison au cas des conditions linéaire générale (cf.35.2), on peut noter les points suivants :

- seul le type de ddl “X1” est acceptable, tous les autres provoquent une erreur,
- Il n’est pas possible d’indiquer un facteur d’échelle (ou alors = 1, ce qui signifie “pas de facteur d’échelle ”)
- Il est possible d’indiquer un temps mini, et un temps maxi, définissant ainsi une plage pendant laquelle la condition est appliquée (cf.35.2), voir l’exemple : (203),
- Il n’est pas possible d’indiquer que la condition est relative (comme dans le cas générale).
- Il n’est pas possible d’indiquer des références secondaire (comme dans le cas générale), car cela n’a aucune signification ici.

TABLE 203 – Exemple de conditions linéaires par projection sur un plan, avec un temps mini et un temps maxi

```

condition_limite_lineaire_
#-----
# Noeuds   blocage
#-----
N_droit   enu= X1
          TEMPS_MINI= 0.1 TEMPS_MAXI= 5.
def_auto_coef_planOuDroite_ centre_noeud_a_t0_ nom_mail= plaque 2 coefficients= 1. 0. 0.4 fin_list_coefficients_ AvecFonctionsDeCharges_
deb_fonction_de_charge= cfixe cfixe c45monte fin_list_fonctions_de_charges_

```

Enfin notons que le cas 1D n’est pas prévu, car il ne représente pas de cas physique a priori.

35.2 Condition linéaire générale

Il s’agit ici d’une relation entre des degrés de liberté d’une même famille, mais pouvant appartenir à plusieurs noeuds. La présentation est faite à travers un exemple.

TABLE 204 – Exemple de conditions linéaires entre ddl de plusieurs noeuds

```

condition_limite_lineaire_
#-----
# Noeuds    blocage
#-----
N_deb    enu= UX
      refs_associe= N_fi  fin_list_refs_associe_
      val_condi_lineaire= 0  coefficients= 1 1  fin_list_coefficients_

```

La table (204) présente le cas d’une condition linéaire entre plusieurs noeuds en 1D. Ce type de condition peut donc s’établir en toute dimension : 1D, 2D ou 3D. On trouve en premier le nom d’une référence principale, ici “N_deb” qui donne la liste des noeuds qui piloteront la condition. Ensuite on trouve le type de degré de liberté de la famille de ddl, sur laquelle s’applique la condition limite, par exemple s’il s’agit de “UX” et que l’on est en 3D, la condition limite concerne “UX”, “UY” et “UZ” qui sont tous les trois de la famille des déplacements. A noter qu’une condition linéaire sur “UX” est différente d’une condition linéaire sur “X1”, la première concerne des déplacements, la seconde des positions.

Viennent ensuite éventuellement, sur la ligne qui suit, une liste de références associées de noeuds pouvant être liés aux noeuds principaux. Ces références associées sont optionnelles, mais si elles existent, chacune des références associées doit contenir exactement le même nombre de noeuds que la référence principale. Ainsi supposons que l’on ait 4 noeuds dans la liste principale, chaque référence associées doit contenir 4 noeuds, et la condition linéaire s’applique sur tous les groupes : un noeud “i” de la liste principale, et un noeud “i” de chaque référence associée. Dans notre exemple il y a une seule référence associée : “N_fi”, la référence principale et la référence associée comporte 1 seule noeud. La relation linéaire s’applique au deux noeuds (et non à chaque noeud séparément). Les références associées s’introduisent sur la ligne, et sont encadrées par les mots clés : “refs_associe=” et “fin_list_refs_associe_”. Suit alors une série de grandeurs facultatives, dans l’ordre suivant :

- l’échelle précédée du mot clé “ECHELLE :” , qui sera utilisé pour le coefficient de la condition linéaire,
- le temps mini précédée du mot clé ”TEMPS_MINI=” , à partir duquel la condition est active,
- le temps maxi précédée du mot clé ”TEMPS_MAXI=” , au delà duquel la condition est inactive,
- le fait que la condition soit relative ou pas (1 ou 0), précédée du mot clé ”CONDITION_RELATIVE=”

Ensuite sur la ligne qui suit on trouve tout d’abord la valeur de la condition linéaire précédé par le mot clé : “val_condi_lineaire=”. Ensuite on trouve les coefficients de la

condition linéaire encadrés par les mots clés : “coefficients=” et “fin_list_coefficients_”. Le nombre de coefficient est égale au nombre de références c’est-à-dire le nombre de références associés + 1 (correspondant à la référence principale) , multiplié par le nombre de ddl de la famille introduite sur la première ligne. Dans le cas de l’exemple, on est en 1D, donc un seul ddl dans la famille des déplacements, il y a deux références en tout donc il faut 2 coefficients, ce qui est le cas. En résumé, la condition correspond sur l’exemple à : $1*UX(\text{du noeud "i" de "N_deb"}) + 1*UX(\text{du noeud "i" de "N_fin"}) = 0$, ceci pour $i=1$ au nombre total de noeud de chaque référence

Comme dans le cas des conditions limites (35.1) il est possible d’indiquer avec la même syntaxe l’existence de fonctions de charges, permettant de faire varier les coefficients de la condition linéaire pendant le calcul. pour cela apres le mot clé “fin_list_coefficients_” on indique le mot clé “AvecFonctionsDeCharges_” (cf. 205 par exemple). S’il y a des fonctions de charge, leur nombre doit-être identique à celui des coefficient+1 (pour la valeur de la condition limite) on a donc une liste de nom de fonction de charge encadree par 2 mots clef : “deb_fonction_de_charge=” et “fin_list_fonctions_de_charges_”. La première fonction de charge s’applique sur la condition, la deuxième sur le coefficient 1, la ieme sur le coefficient $i-1$. La liste des noms des fct de charge s’indique sur la ligne qui suit

Et, comme on l’a vu précédemment, il est possible d’indiquer un facteur d’échelle global à tous les paramètres et une condition relative ou pas. Par défaut il s’agit de condition non relative. L’option “relatif” est utile au niveau des ddl des noeuds, qui sont des valeurs issues du calcul d’équilibre donc non prévisibles a priori. Par contre les coefficients, fixes ou dépendants de fonctions de charge, sont connues à l’avance, donc pour ces dernières il n’y a aucun intérêts de prévoir une procédure particulière à l’option “relatif”.

TABLE 205 – Exemple de conditions linéaires avec l’utilisation de fonctions de charge

```
condition_limite_lineaire_
#-----
N_avant enu= UX
refs_associe= N_arriere N_milieu fin_list_refs_associe_ TEMPS_MINI= 0.1 TEMPS_MAXI= 5.
val_condi_lineaire= 10 coefficients= 1 2 3 4 5 6 7 8 9 fin_list_coefficients_ AvecFonctionsDeCharges_
deb_fonction_de_charge= chc ch1 ch2 ch3 ch4 ch5 ch6 ch7 ch8 ch9 fin_list_fonctions_de_charges_
```

D’où en résumé le fonctionnement est le suivant pour la valeur fixée de la condition linéaire :

- Lorsque la valeur est fixe, celle-ci (multipliée éventuellement par le facteur d’échelle) est appliquée intégralement, elle ne dépend donc pas de l’amplitude du chargement. Par exemple si l’on veut “ $UX(\text{noeud 1}) + UX(\text{noeud 2}) = 2$.” avec un facteur d’échelle unitaire (valeur par défaut), cette relation sera valide quelque soit l’amplitude et le temps (ou le paramètre d’avancement) du chargement. Cette relation est également indépendante du fait que la condition soit relative ou pas.
- Lorsque la valeur dépend d’une fonction de charge, elle dépend donc du temps (ou du paramètre d’avancement) de chargement. Que se soit une condition absolue (situation par défaut) ou relative, la valeur est tout simplement multipliée par la fonction de

charge et le facteur d'échelle globale.

Concernant les coefficients de la condition linéaire, le fonctionnement est le suivant :

- Lorsqu'il n'y a pas de fonction de charge les coefficients sont fixes tout au long du calcul. Ceux-ci (multipliés éventuellement par le facteur d'échelle) sont appliqués intégralement, ils ne dépendent donc pas de l'amplitude du chargement.
- Lorsque les coefficient dépendent d'une fonction de charge, ils dépendent donc du temps (ou du paramètre d'avancement) de chargement. Que se soit une condition absolue (situation par défaut) ou relative, la valeur des coefficient est tout simplement multipliée par la fonction de charge associée et le facteur d'échelle globale.

Concernant la répercussion de la condition linéaire au niveau des ddl, c'est-à-dire le blocage à une valeur dépendant de la condition sur un ddl particulier (cf. théorie de la méthode), le fonctionnement est le suivant :

- Lorsque la condition est absolue, la valeur finale de blocage s'obtient à l'aide des paramètres de la condition et s'applique alors directement sur la valeur finale du ddl,
- Lorsque la condition est relative, c'est un accroissement de la valeur de blocage qui est calculé à l'aide des paramètres de la condition. Cette accroissement s'ajoute alors à la valeur à t du ddl.

Il est possible de superposer un grand nombre de condition linéaire. Cependant, il faut noter que chaque condition conduit à un blocage sur un ddl du noeud principal. En particulier si l'on considère une condition en déplacement ou en position, le blocage s'appliquera sur un des ddl "Xi", par exemple en 3D, il y aura 3 possibilités. Si l'on veut que le noeud participe à plus que 3 conditions linéaires, il faut changer de noeud principal.

Remarque : Toutes les conditions linéaires ne conduisent pas forcément à une condition licite. Prenons par exemple le cas simple suivant : soit une barre en 1D, modélisé par un élément linéaire à 2 noeuds, et soumise à une force de traction au noeud 2. Supposons de plus la condition linaire : " $U_1(\text{noeud } 1) - U_1(\text{noeud } 2) = 0$ ", ce qui signifie que l'on souhaite le même déplacement pour le noeud 1 et 2. Supposons que l'on retient comme noeud principal le noeud 1, dans ce cas la condition est impossible, et conduit à une division par zéro. Si l'on retient le noeud 2, dans ce cas on impose un déplacement solide, mais la force ne peut pas être prise en compte car le déplacement du noeud 2 est imposé par la condition, cette dernière est donc toujours illicite. Par contre la condition " $U_1(\text{noeud } 1) + U_1(\text{noeud } 2) = 0$ " avec comme noeud principale 1 conduit à un problème bien posé et ne pose aucun problème de résolution.

35.2.1 Particularités liées aux noms de maillage

Lorsqu'il n'y a qu'un seule maillage dans les données, il n'y a aucun problème de nom maillage. Par contre il est possible d'introduire plusieurs maillages. Dans ce cas, chaque nom de référence doit en principe être précédé par un nom de maillage ce qui permet par exemple d'utiliser un même nom de référence pour plusieurs maillages. Donc supposons que l'on soit dans un cas où plusieurs maillages sont présents. Tout d'abord, le nom de la référence principale doit-être obligatoirement précédé par un nom de maillage (lui-même précédent du mot clé : "nom_mail="). Ensuite, concernant la liste des noms de références secondaires on peut avoir les fonctionnements suivants :

- il est possible d'omettre un nom de maillage avant chaque nom de référence secon-

daire. Dans ce cas, par défaut on suppose que le nom de maillage associé à ces références secondaires, est celui de la référence principale.

- il est possible de mettre un nom de maillage devant certain ou tous les noms de référence secondaire. A chaque fois que l’on indique un nom de maillage, il doit-être précédé du mot clé : “nom_mail=”. Les différents noms de maillage peuvent-être identiques ou différents de celui de la référence principale. Ainsi on peut définir des conditions linéaires entre des noeuds de maillages différents.

Il faut noter qu’à chaque fois qu’il manque un nom de maillage devant une référence secondaire, c’est le nom de maillage de la référence principale qui est utilisée.

La table (206) donne un exemple d’utilisation de nom de maillage.

TABLE 206 – Exemple de conditions linéaires entre noeuds de maillages différents

```

condition_limite_lineaire_
#-----
# condition entre N_avant du solide 1, N_arriere du solide 2 et
# N_milieu du solide 1
#-----
nom_mail= solide1 N_avant enu= UX
refs_associe= nom_mail= solide2 N_arriere N_milieu fin_list_refs_associe_
val_condi_lineaire= 10 coefficients= 1 2 3 4 5 6 7 8 9 fin_list_coefficients_

```

35.3 Conséquences des CLL sur le stockage matriciel (largeur de bande)

Par défaut, le stockage de la raideur est en matrice bande symétrique dont l’encombrement est directement relié à la largeur de bande. Cette dernière dépend des relations qui sont susceptibles d’exister entre les différents noeuds, et en particulier de la différence de numéros des noeuds concernés. Ainsi la largeur de bande dépend en premier lieu des éléments, mais elle dépend également des conditions linéaires. Aussi, lorsque l’on met en place des conditions linéaires il faudrait vérifier que la largeur de bande initialement calculée pour les éléments, soit suffisante pour prendre en compte les CLL. Pour ce faire, au début du calcul, Herezh++ calcul une largeur de bande qui dépend des éléments et des CLL, que celle-ci soit active ou pas (c’est-à-dire quelque soit la valeur des temps mini et maxi des CLL). Ainsi cette largeur de bande sera correcte tout le long du calcul. Par contre elle ne sera pas forcément optimal, du fait qu’en général la largeur de bande est optimisée par le mailleur, uniquement vis-à-vis des éléments. Herezh++ affiche les différentes largeurs calculées pour un niveau de commentaires (cf. 4) supérieur ou égale à 5.

Dans le cas de l’utilisation d’un algorithme explicite (Tchamwa, DFC, relaxation dynamique), il est possible de mettre en place des conditions linéaires sur les accélération (la

seule pour l’instant mise au point et testée). Si la matrice de masse demandée est de type diagonale, elle est remplacée par une matrice bande de largeur suffisante pour prendre en compte les conditions limites linéaires (comme pour un problème statique). Cependant, le calcul de la masse reste néanmoins diagonal, comme demandé initialement. A chaque incrément (ou itération en relaxation dynamique), la matrice masse est inversée en tenant compte des conditions linéaires, qui peuvent dépendre du temps. En conséquence, le temps de calcul peut être sensiblement augmenté par rapport à un calcul explicite classique utilisant une matrice diagonale dont l’inversion est triviale.

36 Conditions initiales

Dans le cas d’un calcul dynamique par exemple il est nécessaire d’indiquer des conditions limites sur les positions et/ou sur les vitesses initiales. Dans Herezh++ il est également possible d’initialiser les accélérations. Le principe d’utilisation est très proche de celui des déplacement (ou autre degré de liberté) imposés. Le mot clé à utiliser est : ”initialisation”.

L’exemple de la table (207) montre comment implanter des conditions initiales lorsqu’il n’y a qu’un seul maillage.

TABLE 207 – Exemple de mise en place de conditions initiales dans le cas d’un seul maillage

```

initialisation -----
#-----
# Ref noeud | Bloquages
#-----
N_fi          'V1=1000.' # vitesse initiale
N_2           'X1=41.'  # position initiale

```

L’exemple précédent initialise les composantes V1 c’est-à-dire la première composante de vitesse, des noeuds de la référence ”N_fi” à la valeur 1000. De même les composantes X1, c’est-à-dire la première composante de position, des noeuds de la référence ”N_2” à la valeur 41 (voir cependant le paragraphe (36.1) pour les particularités de l’initialisation des positions). Dans le cas de l’accélération on utilise les identificateurs : ”GAMMA”, c’est-à-dire GAMMA1 ou/et GAMMA2 ou/et GAMMA3.

Dans le cas où il y a plusieurs maillage il est obligatoire d’indiquer le nom du maillage auquel la référence se rapporte, avant le nom de la référence. L’exemple de la table (208) montre comment implanter des conditions initiales lorsqu’il y a plusieurs maillages.

Dans le cas de la dynamique on se reportera à (5.10) pour des précisions supplémentaires.

36.1 Particularité de l’initialisation des positions par rapport aux autres degrés de libertés

TABLE 208 – Exemple de mise en place de conditions initiales dans le cas de plusieurs maillages

```

initialisation -----
#-----
#   nom du maillage | Ref noeud |           Bloquages           |
#-----
nom_mail= piece     N_tout           'V1=1000.' # vitesse initiale
nom_mail= outil     N_2              'X1=41.'  # position initiale

```

La seule solution disponible pour initialiser les différents degrés de liberté à $t=0$ est d'utiliser les conditions d'initialisation. Cependant ceci n'est pas vrai pour les ddl de position, dont les valeurs initiales correspondent aux coordonnées des noeuds lues au moment de l'acquisition du maillage. Aussi, contrairement aux autres ddl, dans le cas d'une initialisation des ddl X_i , il y a en fait initialisation des X_i à t c'est-à-dire à " $0+\Delta t$ ", Δt étant le premier incrément de temps utilisé. L'initialisation peut créer ainsi un champ de déformation initiale, qui est développé sur le premier pas de temps.

Dans le cas des autres degrés de libertés, par exemple les températures, le fait d'indiquer une température initiale, par exemple 'TEMP=200', fixe la valeur initiale c'est-à-dire la valeur à $t=0$. Dans le cas où de plus un chargement en fonction du temps est imposé sur ce ddl, la valeur initiale viendra s'ajouter au chargement imposé (cf. lire la Remarque (34.5) pour plus d'information sur le fonctionnement).

Dans le cas de la dynamique on se reportera à (5.10) pour des précisions supplémentaires.

36.2 Conditions de symétrie ou d'encastrement initiale

Comme dans le cas des blocages, il est possible d'introduire des conditions initiales de symétrie ou d'encastrement pour les coques, en particulier de type SFE. Comparée aux blocages, la condition initiale à l'intérêt d'être appliquée qu'une fois d'où un gain de temps de calcul.

La syntaxe est identique à celle des blocages, on se reportera donc à (34.7) pour sa description précise.

Onzième partie

Chargement global

37 Algorithme de chargement

Par défaut l'ensemble du chargement, c'est-à-dire les actions prescrites que se soit en force ou en déplacement, est imposé dès le début du calcul. Il est cependant souvent préférable et même nécessaire d'imposer ces conditions suivant un algorithme particulier. En général cet algorithme est guidé par le paramètre temps. Ce paramètre varie pendant le calcul, selon d'une part les indications données par l'utilisateur dans la zone "paramètres de contrôle général", par exemple l'incrément de pas de temps le temps final ..., et d'autre part par les résultats du calcul, à la fin de chaque incrément de temps si le calcul a convergé le temps de calcul est incrémenté sinon on diminue le temps en espérant ainsi faciliter le calcul.

Une première utilité d'un algorithme de chargement est donc l'implantation d'un chargement dépendant directement du temps. Par exemple lors d'un comportement statique fortement non linéaire, il est souhaitable de pouvoir imposer le chargement de manière progressive, ce qui permettra d'éviter les divergences des algorithmes itératifs de recherche de zéro au niveau des équations d'équilibre global ou au niveau de la loi de comportement. Le mot clé à utiliser est : typecharge, cf. l'exemple suivant.

```

                                typecharge -----
#-----
#  NOM DU      |      temps      |
#  TYPE        |                      |
#-----
      TYPE1          1.0

```

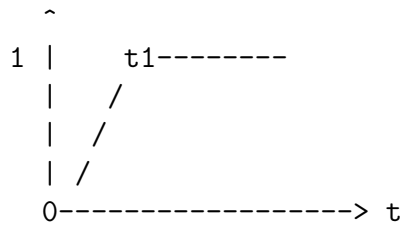
La liste exhaustive des algorithmes de chargement disponibles est donné dans la table (209).

TABLE 209 – liste des algorithmes de chargement

nom	paramètres	commentaire simplifié	référence
TYPE1	un réel donnant le temps t1	rampe avec palier de stabilisation	1
TYPE2	deux réels donnant les temps t1 et t2	rampe, palier puis arrêt brutal	2
TYPE3	aucun paramètre	uniquement un palier	3
TYPE4	une courbe	le chargement est piloté par une fonction $y=f(x)$	4
TYPE5	une liste de points	le chargement est piloté par la liste de points	5

Avec les commentaires détaillés suivants :

1. TYPE1 : le chargement correspond au schéma suivant :

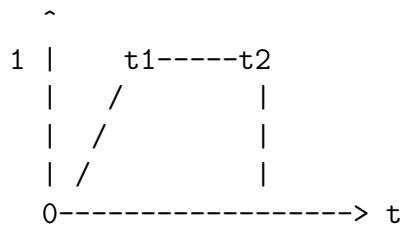


Du temps $t=0$ à $t=t_1$, le chargement évolue de manière linéaire pour atteindre la valeur finale au temps t_1 , en ordonnée est indiqué le facteur multiplicatif global du chargement. A partir du temps t_1 , le chargement demeure constant.

Cet algorithme constitue le schéma de base pour imposer progressivement un chargement. Exemple de déclaration de chargement de type 1 :

typecharge -----		
#-----		
# NOM DU		temps
# TYPE		
#-----		
TYPE1		1.0

2. TYPE2 : le chargement correspond au schéma suivant :

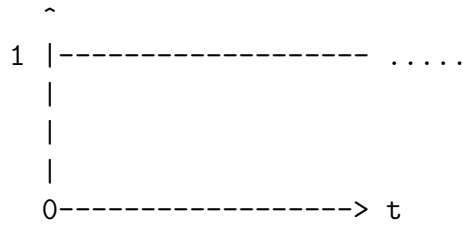


Du temps $t=0$ à $t=t_1$, le chargement évolue de manière linéaire pour atteindre la valeur finale au temps t_1 , en ordonnée est indiqué le facteur multiplicatif global du chargement. A partir du temps t_1 , le chargement demeure constant jusqu'au temps t_2 , puis tout le chargement est supprimé de manière instantanée, pour ensuite garder une valeur nulle. Exemple de déclaration de chargement de type 2 :

typecharge -----		
#-----		
# NOM DU		temps
# TYPE		
#-----		
TYPE2		1.0 2.0

Ce chargement est intéressant pour imposer un chargement non nul de manière quasi-statique, c'est-à-dire pour un temps t_1 grand. De t_1 à t_2 on attend la stabilisation. En suite la suppression du chargement permet d'étudier par exemple les oscillations de la structure libre.

3. TYPE3 : le chargement correspond au schéma suivant :



Aussitôt le calcul lancé, l'ensemble du chargement est imposé, et ceci de manière constante dans le temps. Le chargement est actif également pour le temps $t=0$ inclus (contrairement aux intervalles de temps t_{min} , t_{max} que l'on peut définir après les chargements individuels, qui n'incluent pas la borne mini). Par contre pour prendre en compte un chargement il faut également indiquer un temps mini inférieur à 0 pour éviter l'exclusion du temps 0 (cf. 33.1.3). Exemple de déclaration de chargement de type 3 :

```

                                typecharge -----
#-----
#  NOM DU      |      temps      |
#  TYPE        |                  |
#-----
                TYPE3          # pas de temps caractéristique

```

4. TYPE4 : le chargement s'effectue au travers d'une fonction multiplicative quelconque (cf.220). Exemple de déclaration de chargement de type 4 :

```

                                typecharge -----
#-----
#  NOM DU      |      temps      |
#  TYPE        |                  |
#-----
                TYPE4          COURBEPOLYLINEAIRE_1_D
                Debut_des_coordonnees_des_points
                Coordonnee dim= 2 0.    0.
                Coordonnee dim= 2 0.5  1.
                Coordonnee dim= 2 1.    1.
                Coordonnee dim= 2 1.5  0.
                Fin_des_coordonnees_des_points

```

5. TYPE5 : le chargement s'effectue au travers d'une fonction multiplicative qui doit être uniquement de type polylinéaire normale ou simplifiée (cf.220). c'est-à-dire constituée d'une liste de points. Il doit y avoir au moins deux points. Les coordonnées x des points représentent les différents temps où l'on veut effectuer les

calculs. Les coordonnées y des points représentent un facteur multiplicatif du chargement que l'on impose. Il faut noter que tous les temps doivent être différents, et qu'il ne peut pas y avoir décroissance du temps !

Exemple de déclaration de chargement de type 5 :

```

      typecharge -----
#-----
# NOM DU      | temps   et facteurs multiplicatifs |
# TYPE       |                               |
#-----
      TYPE5          COURBEPOLYLINEAIRE_1_D
      Debut_des_coordonnees_des_points
      Coordonnee dim= 2 0.  0.
      Coordonnee dim= 2 0.5  1.
      Coordonnee dim= 2 1.   1.
      Coordonnee dim= 2 1.5  0.
      Fin_des_coordonnees_des_points

```

Douzième partie

Paramètres de contrôles et de pilotage

38 Résolution : introduction

Herezh++ est dédié au calcul non linéaire. La résolution est en général réglée par un certain nombre de paramètres, ceux-ci étant actuellement partagés en plusieurs groupes :

- les paramètres de contrôle général (39) : liste exhaustive (210) ,
- les paramètres dédiés à la dynamique (40) : liste exhaustive (211) ,
- les paramètres de contrôle liés aux calculs des énergies (41) : liste exhaustive (212) ,
- les paramètres de gestion de la résolution du système linéaire (42) : liste exhaustive (213) ,
- les paramètres de contrôle du pilotage de la résolution globale (43) : liste exhaustive (214) ,
- les paramètres liés au pilotage du contact (44) : liste exhaustive (215) ,
- les paramètres liés à l’affichage des résultats (45) : liste exhaustive (216) ,
- les paramètres liés à certains calculs géométriques (46) : liste exhaustive (217) .

La plupart des paramètres sont optionnels, les valeurs par défaut essayant de répondre aux cas les plus courant.

39 Contrôle général

L’exemple suivant présente une liste de paramètres classiques de contrôle. Le mot clé est ”controle” suivi d’une liste de sous-mots clés, renseigné ou non par une valeur.

```
                controle -----
#-----
# PARAMETRE      |  VALEUR      |
#-----
SAUVEGARDE       1
DELTAtMAXI      1.
TEMPSFIN         2.
DELTAt          1.
MULTIPLICATEUR   10000.
ITERATIONS       200
PRECISION        1e-8
RESTART          1
```

La liste exhaustive des sous-mots clés disponible est donné dans la table (210).

Avec les commentaires suivants :

1. SAUVEGARDE < INTER_TEMPS > < un entier ou un réel > : Le mot cle ”INTER_TEMPS” est optionnel. Dans le cas où il n’y a pas le mot clé ”INTER_TEMPS”, le nombre qui suit est transformée en entier, qui donne alors la fréquence de sauvegarde sur disque des incréments de charge. En effet, lors d’un chargement incrémental, le nombre d’incrément calculé peut être très élevé (plusieurs milliers par exemple). Dans ce cas il peut être utile de ne sauvegarder qu’un incrément sur n. Il est

TABLE 210 – liste des sous mots clés associés aux paramètres de contrôle

sout mot clé	valeur par défaut	ref du commentaire
SAUVEGARDE	1	(1)
DELTA _t	1.	(2)
DELTA _t MAXI	1.	(3)
DELTA _t MINI	0.	(4)
TEMPSFIN	1.	(5)
PRECTEMPS	1.E-12	(6)
ITERATIONS	100	(7)
PRECISION	1.E-3	(8)
NORME	"Residu/Reaction"	(9)
MAXINCRE	400	(10)
RESTART	0	(11)
MULTIPLICATEUR	1.	(12)
LINE_SEARCH		(13)
VARIATION_CHARGE_EXTERNE_SUR_RAIDEUR	0	(14)
VARIATION_JACOBIEN_SUR_RAIDEUR	(obsolète)	(15)
VARIATION_VITESSE_DEFORMATION_SUR_RAIDEUR	1	(16)

également possible de mettre le mot clé optionnel "INTER_TEMPS", dans ce cas le nombre qui suit est interprété comme un réel, qui donne alors l'intervalle de temps approximatif qui sépare deux sauvegardes. En fait comme chaque calcul est effectué à un temps discret, il y a sauvegarde, dès que le temps du calcul est supérieur au dernier temps de sauvegarde + l'intervalle de temps indiqué. Le temps réel entre deux sauvegardes n'est donc qu'approximativement le temps indiqué à l'aide du paramètre "INTER_TEMPS". L'approximation est d'autant meilleure que le pas de temps de calcul Δt est petit par rapport au pas de temps de sauvegarde.

Ce mode de sauvegarde est intéressant par exemple lorsque l'on travaille en dynamique, et que le pas de temps donc le nombre d'itération pour un avancement temporel donné, change pendant le calcul.

Noter également que par défaut la sauvegarde est effectuée à tous les incréments, ceci dans le fichier .BI, dans le cas où l'on ne veut aucune sauvegarde, il faut indiquer "SAUVEGARDE 0". Dans ce cas il y a création du fichier .BI, mais il reste vide.

Il est également possible de ne sauvegarder que le dernier temps calculé (effectivement valide). Dans ce cas on indique "SAUVEGARDE < DERNIER_CALCUL >" (le mot clé < DERNIER_CALCUL > doit suivre directement le mot clé "SAUVEGARDE"). Dans ce cas, l'incrément 0 est également sauvegardé. Ce cas peut permettre de reprendre un calcul qui c'est arrêté faute de convergence, après avoir modifié les paramètres de réglage.

2. DELTA_t <un réel> : fixe la valeur de l'incrément de temps pour chaque incrément de charge. Dans le cas d'un calcul statique, le paramètre temps joue le rôle de paramètre d'avancement, il sert à calculer l'avancement du chargement. Dans le cas d'un calcul de dynamique, il est possible de fixer la valeur du pas de temps en fonction

du pas de temps critique de l'avancement temporel qui dépend de la méthode, de la géométrie ... En fait ce pas de temps critique est approché par le programme. On peut définir le pas de temps comme : $\Delta t = \alpha \Delta t_{critique(approchee)}$. Pour cela, à la suite du mot clé : DELTA_t on indique COEF_PASCRIQUE suivi de la valeur du coefficient α . Dans ce cas le pas de temps Δt est évalué en début d'algorithme puis mis à jour à chaque changement de pas de temps critique. Enfin lorsque l'on effectue un restart, il est possible d'indiquer au programme que l'on veut utiliser le pas de temps indiqué dans le .info plutôt que de reprendre de pas temps de l'incrément précédent. Pour cela on indique sur la ligne de donné à la suite des autres informations, le mot clé FORCE_DELTAT_DU_INFO. Ainsi, si ce dernier mot clé n'existe pas, ce qui est le cas par défaut, le programme reprend le dernier pas de temps sauvegardé. (voir également la remarque 39). Il est également possible d'utiliser le pas de temps critique de la condition de courant, ce qui correspond au pas de temps critique de la méthode classique DFC. Pour ce faire on utilise le mot clé COEF_PASCRIQUE_DFC à la place de COEF_PASCRIQUE. L'avantage de ce pas de temps critique est qu'il sera identique, quelque soit l'algorithme d'avancement temporel utilisé.

3. DELTA_tMAXI <un réel> : fixe la valeur de l'incrément de temps maxi pour l'incrément de charge. Lorsque le pilotage automatique du pas de temps conduit à augmenter le pas de temps, celui-ci ne peut dépasser la valeur donnée pour DELTA_tMAXI. Comme pour le pas de temps, il est possible de fixer le pas de temps maxi en fonction du pas de temps critique, dans le cas d'un calcul dynamique. Dans ce cas on aura $\Delta t_{maxi} = \alpha \Delta t_{critique(approchee)}$. Pour cela, à la suite du mot clé : DELTA_tMAXI on indique COEF_PASCRIQUE suivi de la valeur du coefficient α . Dans ce cas le pas de temps Δt_{maxi} est évalué en début d'algorithme puis mis à jour à chaque changement de pas de temps critique. De manière identique au paramètre DELTA_t, on peut également utiliser le pas de temps critique de la méthode DFC en indiquant COEF_PASCRIQUE_DFC à la place de COEF_PASCRIQUE.
4. DELTA_tMINI <un réel> : fixe la valeur de l'incrément de temps mini pour l'incrément de charge, cet incrément doit être évidemment = ou plus petit que l'incrément de temps maxi. Lorsque le pilotage automatique du pas de temps conduit à diminuer le pas de temps, celui-ci ne peut être inférieur à la valeur donnée par DELTA_tMINI. Comme pour le pas de temps, il est possible de fixer le pas de temps mini en fonction du pas de temps critique, dans le cas d'un calcul dynamique. Dans ce cas on aura $\Delta t_{mini} = \alpha \Delta t_{critique(approchee)}$. Pour cela, à la suite du mot clé : DELTA_tMINI on indique COEF_PASCRIQUE suivi de la valeur du coefficient α . Dans ce cas le pas de temps Δt_{mini} est évalué en début d'algorithme puis mis à jour à chaque changement de pas de temps critique. De manière identique au paramètre DELTA_t, on peut également utiliser le pas de temps critique de la méthode DFC en indiquant COEF_PASCRIQUE_DFC à la place de COEF_PASCRIQUE.
5. TEMPSFIN <un réel> : fixe la valeur maxi du temps à atteindre avant de stopper le calcul.
6. PRECTEMPS <un réel> : fixe la précision utilisée lors des tests effectués sur les temps. En particulier, le temps fin peut être satisfait à la précision PRECTEMPS donnée.

7. ITERATIONS <un entier> : fixe le nombre d'itération maxi au delà desquelles, on considère que le calcul n'a pas convergé. Ensuite soit le calcul s'arrête, soit le pas de temps est modifié par le pilotage automatique.
8. PRECISION <un réel> : fixe la précision sur la norme de convergence, en dessous de laquelle le calcul est réputé avoir convergé.
9. NORME <une chaîne de caractère> : les choix possibles sont :
 - Residu : correspond à une norme infini sur le résidu, c'est-à-dire que l'on teste le maximum des composantes du résidu.
 - Residu/Reaction : c'est une norme relative qui correspond à la norme infinie du résidu divisé par le maximum des réactions.
 - Residu/PVExterne : norme relative correspondant à la norme infinie du résidu divisée par le maximum des résidus externes c'est-à-dire causés par les efforts externes.
 - Residu/PVInterne : norme relative correspondant à la norme infinie du résidu divisée par le maximum des résidus internes c'est-à-dire causés par les efforts de cohésion.
 - Residu/maxPVI : norme relative correspondant à la norme infinie du résidu divisée par le maximum des : résidus internes, résidus externes et maximum des réactions.
 - min(Res,Res/Reaction) : le minimum du la norme infinie du résidu et de la norme relative correspondant à la norme infinie du résidu divisée par le maximum des réactions.
 - E_cinetique/E_statique_ET_ResSurReact : norme utilisable **uniquement** avec un algorithme dynamique. Correspond au maxi d'une part de : l'énergie cinétique divisé par l'énergie interne plus l'énergie externe, et d'autre part : du résidu divisé par le maxi des réactions.
 - Bilan_puissance/Max(puiss) : correspond au bilan des puissances réellement en jeux, divisé par le maxi des puissances : interne, externe, et éventuellement d'accélération si elle existe.

Cas d'un calcul implicite, contrôle de variation maxi de ddl : Il est également possible d'indiquer dans le cas d'un calcul implicite (cf. 43) que l'on souhaite contrôler la variation des ddl. Dans le cas où cette variation devient trop faible, on considère qu'il y a divergence, ou plutôt que la convergence sera très longue, d'où l'arrêt des itérations avec les paramètres actuels. Pour mettre en oeuvre ce contrôle, on indique après le nom du résidu, la chaîne de caractère : "_et_miniVarDdl". Par exemple dans le cas de la norme relative aux réaction avec une limitation minimale des variations ddl, le mot clé à utiliser devient : "Residu/Reaction_et_miniVarDdl". La chaîne terminale "_et_miniVarDdl" peut ainsi être rajouté à toutes les normes présentées précédemment.

Contrôle de variation mini de ddl : D'une manière identique au contrôle de la variation minimale de ddl, on peut également introduire un contrôle maximal de la variation des ddl. Dans ce cas la chaîne de caractère à rajouter est "_et_maxiVarDdl".

Contrôle de l'évolution du résidu : Lors d'une "bonne" convergence, le résidu doit normalement diminuer. Supposons qu'entre "n" itération, n=3 par exemple, le

résidu ne diminue pas, et cela "m" fois, m=3 par exemple, on peut conclure qu'il y a des oscillations, qui ne permettront pas de converger. D'une manière analogue aux contrôles sur les variations de ddl, le contrôle sur le résidu s'active avec la chaîne de caractère "_et_VarRes".

Le contrôle sur la variation minimale ou maximale de ddl et sur l'évolution du résidu ne s'effectue pas dans le cas d'une convergence.

Il est possible de cumuler tous les contrôles et cela dans n'importe quel ordre. Par exemple les deux mots clés suivant sont correctes : "Residu_et_VarRes_et_miniVarDdl-et_maxiVarDdl" , "Residu/Reaction_et_maxiVarDdl_et_VarRes_et_miniVarDdl".

Les contrôles sur les mini et maxi des variations de ddl dépend de paramètres qui doivent être modifiés pour que le contrôle soit efficace. Ainsi, par défaut la valeur minimale de contrôle des variations de ddl est 0., pour que le contrôle soit effectif, il faut lui donner une valeur non nulle. De même, la valeur par défaut des variations maxi de ddl est un nombre très grand. On se reportera au chapitre (43) et plus particulièrement au tableau (214) pour la modification de ces paramètres. On peut également modifier au même endroit, les paramètres qui règlent le contrôle sur l'évolution du résidu. **Important** : ces différents contrôles additionnels ne sont pertinents que dans le cas d'un pilotage (cf. 43) qui a pour objectif de contrôler la taille du pas d'incrément de manière à minimiser le temps de calcul, ceci en agissant sur le nombre d'itérations par exemple. Ainsi, dans le cas d'un calcul explicite, ces paramètres n'ont a priori aucune utilité.

10. MAXINCRE <un entier> : le calcul s'arrête lorsque l'on dépasse le maximum d'incrément autorisé. Ceci permet entre autre d'éviter qu'un calcul ne s'arrête jamais.
11. RESTART <un entier> : ce paramètre est très utile pour permettre de redémarrer un calcul à un incrément préalablement sauvegardé, ce qui évite de devoir redémarrer le calcul tout au début, ceci pour une nouvelle panoplie de valeurs de contrôle par exemple. (voir également la remarque 39).
12. MULTIPLICATEUR <un réel> : fixe un coefficient multiplicateur de l'ensemble du chargement.
13. LINE_SEARCH : la présence de ce paramètre signifie que la procédure d'accélération de convergence de type line search est actif.
14. VARIATION_CHARGE_EXTERNE_SUR_RAIDEUR <1 ou 0> : 1 signifie que l'on prend en compte dans le calcul de la raideur, la variation du chargement externe. Ceci est nécessaire lorsque l'on a un chargement qui varie en fonction du déplacement.
15. VARIATION_JACOBIEN_SUR_RAIDEUR <1 ou 0> : Indique si oui ou non, on tient compte de la variation du jacobien dans le calcul de la raideur. En fait ce paramètre vaut 1 par défaut, et il n'est plus possible de le changer, car l'expérience a montré qu'il est préférable d'en tenir compte dans tous les cas.
16. VARIATION_VITESSE_DEFORMATION_SUR_RAIDEUR <1 ou 0> : Indique si oui ou non, on tient compte de la variation de la vitesse de déformation virtuelle dans le calcul de la raideur. En fait ce paramètre vaut 1 par défaut pour tous les éléments sauf les éléments SFE, pour lesquels on peut ou non le mettre en oeuvre. Par

contre pour les autres éléments il n'est plus possible de le changer, car l'expérience a montré qu'il est préférable d'en tenir compte dans tous les cas. Dans le cas des SFE, pour l'instant, il semble que dans le cas d'un comportement principalement de flexion, il est préférable de ne pas l'activer, par contre dans le cas d'un comportement principalement de membrane, il est préférable de l'activer.

Remarque Concernant le pas de temps : lorsque l'on a un pas de temps qui est supérieur au pas de temps critique, et que le calcul est en explicite, le programme re-calcule un pas de temps inférieur au pas de temps critique. Lorsque l'on effectue un calcul en "restart", le programme commence par examiner si le pas de temps lu dans le fichier .info est inférieur au pas de temps critique. Si oui, il retiens ce pas de temps, si non, il re-calcule un pas de temps qui est inférieur au pas de temps critique pour le maillage actuel. Il est tout à fait possible que ce pas de temps ne soit pas exactement celui qu'on aurait obtenu si le calcul n'était pas en restart ! même si l'on refait le même calcul que sans restart. Ceci est due à l'algorithme de pilotage qui ne met pas à jour à "chaque" incrément le pas de temps, mais par défaut essaie plutôt d'anticiper les variations en augmentant les modifications à chaque intervention, ce qui permet de diminuer le nombre d'intervention.

40 Dynamique

Le fonctionnement de ce groupe de paramètres suit la même logique que dans le cas des paramètres de contrôle généraux. L'exemple suivant présente une liste de paramètres classiques. Le mot clé est "para_dedies_dynamique" suivi d'une liste de sous-mots clés, renseigné ou non par une valeur.

```

                para_dedies_dynamique -----
#-----
# PARAMETRE      |  VALEUR      |
#-----
TYPE_CALCUL_MATRICE_MASSE      MASSE_DIAG_COEF_VAR
LIMITATION_TEMPS_MAXI_STABLE      1

```

La liste exhaustive des sous-mots clés disponible est donné dans la table (211). Bien noter que ces paramètres sont facultatifs, le groupe de paramètre est lui-même également facultatif.

Avec les commentaires suivants :

1. TYPE_CALCUL_MATRICE_MASSE <une chaîne de caractère> : les choix possibles sont :
 - MASSE_DIAG_COEF_EGAUX : Dans ce cas les coefficients de la matrice masse sont calculés de la manière suivante : pour chaque élément la masse totale de l'élément est répartie uniformément sur chaque noeud, de manière à obtenir une matrice diagonale représentant le comportement de masses ponctuelles situées à

TABLE 211 – liste des sous mots clés associés aux paramètres de contrôle liés à la dynamique

sout mot clé	valeur par défaut	ref
TYPE_CALCUL_MATRICE_MASSE	MASSE_DIAG_COEF_EGAUX	(1)
LIMITATION_TEMPS_STABLE	1	(2)
AMORTISSEMENT_VISCOSITE_ARTIFICIELLE	0	(3)
VISCOSITE_ARTIFICIELLE	0.1	(4)
COEFFICIENT_DE_RAYLEIGH_POUR_LA_MASSE	1.	(5)
COEFFICIENT_DE_RAYLEIGH_POUR_LA_RAIDEUR	0.	(6)
BULK_VISCOSITY	0.	(7)
COEFF_TRACE	0.06	(8)
COEFF_CARRE_TRACE	1.5	(9)

chaques noeuds, dont la somme est équivalente à la masse de l'élément initiale. Ce type de calcul convient très bien au cas des éléments linéaires. Par contre dans le cas des éléments quadratiques ou de degré plus élevé, le comportement obtenu n'est pas toujours correcte. L'expression définissant les termes de la matrice masse élémentaire est de la forme suivante :

$$m_i = \frac{1}{n} \int_D \rho dv \quad (75)$$

“i” étant un numéro de noeud courant, n étant le nombre de noeud de l'élément, D le volume de l'élément, ρ la masse volumique de l'élément. m_i est la masse au noeud i, D est le domaine d'intégration, φ_i est la fonction d'interpolation du noeud i, ρ est la masse volumique, n est le nombre de noeud de l'élément.

- MASSE_CONSISTANTE : Dans ce cas la matrice masse est calculée suivant la formule théorique

$$M(ar, bs) = \int_D \rho \varphi_r \varphi_s \delta^{ab} \sqrt{g} dv \quad (76)$$

avec r et s les numéros des noeuds horizontalement et verticalement dans la matrice, a et b le numéro de coordonnée : de 1 à 3 en 3D par exemple, ρ la masse volumique, D le volume de la pièce, φ_r la fonction d'interpolation du noeud r. Dans ce cas on modélise bien un solide continue. Convient bien aux éléments linéaires, peut poser des problèmes dans le cas de polynômes de degrés plus élevé.

- MASSE_DIAG_COEF_VAR : Le calcul de la matrice masse est effectuée selon une formule proposée dans l'ouvrage de Batoz sur les coques (??) la matrice est diagonale, la répartition est réalisée au prorata des fonctions d'interpolation (cf page 304, tome 2 Batoz). A priori le modèle adopté permet d'effectuer des calculs de dynamique avec des éléments d'interpolations supérieurs à 1, ce que ne permettent pas facilement les autres modèles de matrices masses. L'expression définissant les termes de la matrice masse élémentaire est de la forme suivante :

$$m_i = \alpha \int_D \varphi_i^2 dv, \quad \text{avec } \alpha = \frac{\int_D \rho dv}{\sum_{j=1}^n \int_D \rho \varphi_j^2 dv} \quad (77)$$

m_i est la masse au noeud i , D est le domaine d'intégration, φ_i est la fonction d'interpolation du noeud i , ρ est la masse volumique, n est le nombre de noeud de l'élément.

2. `LIMITATION_TEMPS_STABLE` <un booléen = 1 ou 0> : lors d'un calcul dynamique explicite, il existe un incrément de temps supérieur critique, au delà duquel, la stabilité de calcul ne sera plus assurée. D'une manière simplifiée, après n incréments (n pouvant être petit 5 par exemple) il y aura systématiquement divergence numérique, typiquement les déplacements obtenus tendront vers l'infini. Par défaut le programme teste et limite le temps proposé par l'utilisateur par rapport à un temps critique estimé à partir du temps de propagation d'une onde élastique au travers de l'élément le plus petit du maillage. Cette limitation n'est réellement efficace qu'avec le premier type de matrice masse : `MASSE_DIAG_COEF_EGAUX`. Pour les autres types de matrice masse, ce temps critique peut-être mal estimé, en particulier pour les matrices masses consistantes, il est nécessaire de choisir un temps en général plus faible, par exemple 0.5 le temps critique estimé. Dans ce cas il suffit d'indiquer un temps critique plus faible que celui proposé par le programme. Ce dernier est affiché dès lors que l'on demande un temps trop élevé : paramètres `DELTA_t` et `DELTA_tMAXI`. Dans le cas où au contraire on veut essayer un temps supérieur au temps critique proposé par le programme il est nécessaire de désactiver le paramètre `LIMITATION_TEMPS_MAXI_STABLE` en le mettant à 0.
3. `AMORTISSEMENT_VISCOSITE_ARTIFICIELLE` <un booléen = 1 ou 0> : indique si l'on désire l'introduction d'une matrice de viscosité artificielle dont la construction est gérée par les variables : `VISCOSITE_ARTIFICIELLE`, `COEFFICIENT_DE_RAYLEIGH_POUR_LA_MASSE`, `COEFFICIENT_DE_RAYLEIGH_POUR_LA_RAIDEUR`. En fait la matrice de viscosité $[C]$ est construite à partir de la matrice de masse $[M]$ et de la matrice de raideur $[K]$. On a $[C] = \eta(\alpha[M] + \beta[K])$ qui correspond à la formule classique de Rayleigh. ν représente le coefficient de viscosité, il est introduit par la variable `VISCOSITE_ARTIFICIELLE`. α représente la proportion de la matrice masse dans la matrice d'amortissement. Classiquement on retient $\alpha = 1.$, en particulier c'est le seul choix disponible pour les calculs en explicite pour lesquels il n'y a pas de construction explicite de la matrice de raideur. α est donné par la variable `COEFFICIENT_DE_RAYLEIGH_POUR_LA_MASSE`. β représente la proportion de la matrice de raideur dans $[C]$, il est donné par la variable `COEFFICIENT_DE_RAYLEIGH_POUR_LA_RAIDEUR`.
4. `VISCOSITE_ARTIFICIELLE` <un réel > : permet de définir le coefficient de viscosité η (cf. 3),
5. `COEFFICIENT_DE_RAYLEIGH_POUR_LA_MASSE` <un réel > : permet de définir le coefficient α de la formule de Rayleigh (cf. 3), bien noter que ce coefficient n'est pas pris en compte en explicite, car il vaut systématiquement 1.
6. `COEFFICIENT_DE_RAYLEIGH_POUR_LA_RAIDEUR` <un réel > : permet de définir le coefficient β de la formule de Rayleigh (cf. 3).
7. `BULK_VISCOSITY` < 0 , 1 ou 2 > : indique si l'on n'utilise pas le bulk viscosity (=0) ou on l'utilise (différent de 0) pour l'amortissement des fréquences numériques

. La méthode du bulk viscosity est une méthode classique qui permet de filtrer automatiquement une partie des hautes fréquences numériques introduites par le schéma numérique d'avancement temporel : par exemple classiquement avec les différences finis centrées. La méthode consiste à introduire un terme de pression hydrostatique P tel que : $P = \rho l(C_1 l I_D^2 - C_2 c I_D)$ si la trace est négative, 0 sinon. On obtient donc la contrainte finale : $\boldsymbol{\sigma}_{finale} = \boldsymbol{\sigma} - P \mathbf{I} = \boldsymbol{\sigma} - \rho l(C_1 l I_D^2 - C_2 c I_D) \mathbf{I}$. Deux cas d'utilisation :

- (i) soit l'utilisation classique mentionnée précédemment, dans ce cas on donne un paramètre = 1,
 - (ii) soit on souhaite une utilisation constante du bulk quelque soit le signe de la trace de la vitesse de déformation. Dans ce cas on indique un paramètre = 2
8. COEFF_TRACE <un réel > : permet de définir le coefficient C_1 facteur de la trace du tenseur de vitesse de déformation, dans la formule du bulk viscosity (cf. 7).
 9. COEFF_CARRE_TRACE <un réel > : permet de définir le coefficient C_2 facteur de la trace du tenseur de vitesse de déformation, dans la formule du bulk viscosity (cf. 7).

41 Calculs des énergies

Dans le cas des différents calculs, on cherche à assurer l'équilibre de la structure au sens des puissances virtuelles. Cela signifie qu'il y a équilibre (statique ou dynamique) à un moment précis. Cependant, ceci ne garantit pas qu'il y ait globalement équilibre à tout instant. En particulier entre deux instants, il peut y avoir un déséquilibre d'autant plus marqué que le comportement est non linéaire. Les indications des différentes énergies calculées permettent d'avoir une idée des différents déséquilibres existant tout au long du chargement. Il s'agit de l'énergie cinétique, de l'énergie fournie par les forces généralisées interne, et par les forces généralisées externes. Le calcul de ces différentes énergies est approché, il utilise la méthode des trapèzes avec les grandeurs calculées à chaque pas de temps, ceci pour les forces généralisés. Pour l'énergie cinétique, elle est calculée en fonction de la vitesse finale. En parallèle il y a également le calcul des puissances réelles : d'accélération, interne et externe. À noter que ces grandeurs sont accessibles en tant que grandeurs globales (comme les énergies).

Il faut bien noter que l'énergie externe et interne, à l'exclusion de l'énergie cinétique, représentent les énergies échangées, mais non les énergies stockées!! qui elles dépendent du caractère réversible ou pas des comportements.

Cependant, durant le calcul, l'énergie interne est également systématiquement calculée sous forme de trois composantes : l'énergie élastique emmagasinée, la dissipation plastique et la dissipation visqueuse. Ces composantes sont calculées au niveau des lois de comportement, donc pour chaque point d'intégration. Il est possible ensuite de récupérer ces trois types d'énergie au niveau des grandeurs globales, et au niveau des grandeurs aux points d'intégrations (grandeurs à post-traiter comme les contraintes et les déformations). On remarquera que la somme de ces trois énergies est différentes de l'énergie interne. Ceci

provient du mode de calcul qui est différent pour ces deux grandeurs. Dans le cas de l'énergie interne (totale), on utilise une méthode des trapèzes sur l'incrément :

$$energie_{interne} = \frac{(R_{int(t)} + R_{int(t+\Delta t)})}{2} \cdot \Delta X \quad (78)$$

Dans le cas des trois énergies, l'intégration sur le pas de temps dépend de la loi : par exemple peut-être en implicite pure ou bien peut-être exacte (exemple de l'élasticité linéaire).

Remarque : Dans le cas d'un comportement statique, il n'y a pas évidemment de calcul de l'énergie cinétique.

Un certains nombres de paramètres permettent de contrôler le calcul et l'affichage des énergies globales : interne, externe et cinétique, (alors que les 3 parties de l'énergie interne sont systématiquement calculées).

Le fonctionnement de ce groupe de paramètres suit la même logique que dans le cas des paramètres de contrôle généraux. L'exemple suivant présente une liste de paramètres classiques. Le mot clé est "para.energie" suivi d'une liste de sous-mots clés, renseigné par une valeur.

```

                para_energie -----
#-----
# PARAMETRE      | VALEUR  |
#-----
NB_INCR_CAL_ENERG      1
AFFICHE_INCR_ENERGIE   0

```

La liste exhaustive des sous-mots clés disponible est donné dans la table (212). Bien noter que ces paramètres sont facultatifs, le groupe de paramètre est lui-même également facultatif.

TABLE 212 – liste des sous mots clés associés aux paramètres de calcul d'énergie

sout mot clé	valeur par défaut	ref
NB_INCR_CAL_ENERG	1	(1)
AFFICHE_INCR_ENERGIE	0	(2)

Avec les commentaires suivants :

1. NB_INCR_CAL_ENERG <un entier> : indique à partir de quel incrément on commence à cumuler les différentes énergies globales échangées, à l'exclusion de l'énergie cinétique qui nécessite pas de cumule, et les énergies internes : élastique visqueuse et plastique qui sont toujours cumulées.
2. AFFICHE_INCR_ENERGIE <un booléen> : indique si de plus on veut l'affichage des différentes énergies sur l'incrément (indépendamment des énergies cumulées).

42 Résolution des systèmes linéaires

Ce jeux de paramètres est facultatif. Ces paramètres servent à définir et préciser la méthode de résolution du système linéaire global que l'on a à résoudre soit pour déterminer l'équilibre dans le cas d'une résolution implicite statique ou dynamique, soit pour résoudre l'équilibre dynamique explicite, c'est-à-dire en intégrant les puissances d'inertie, lorsque la matrice de masse est consistante.

Le fonctionnement de ce groupe de paramètres suit toujours la même logique que dans le cas des paramètres de contrôle généraux. Le mot clé est "para_syteme_lineaire" suivi d'une liste de sous-mots clés, renseigné ou non par une valeur. L'exemple suivant présente une liste de paramètres classiques dans le cas d'un stockage bande symétrique avec la méthode de résolution de Cholesky.

```
                para_syteme_lineaire -----
#-----
# PARAMETRE          |          VALEUR          |
#-----
TYPE_MATRICE          BANDE_SYMETRIQUE
TYPE_RESOLUTION        CHOLESKY
```

Le second exemple suivant présente le cas d'un stockage en matrice creuse, type compressé colonne, avec une méthode de résolution gradient conjugué.

```
                para_syteme_lineaire -----
#-----
# PARAMETRE          |          VALEUR          |
#-----
TYPE_MATRICE          CREUSE_COMPRESSEE_COLONNE
SYMETRIE_MATRICE      0
TYPE_RESOLUTION        CONJUG_GRAD
TYPE_PRECONDITIONNEMENT ILU
NB_ITER_NONDIRECTE    200
TOLERANCE              1.e-15
```

La liste exhaustive des sous-mots clés disponible est donné dans la table (213).
Avec les commentaires suivants :

1. TYPE_MATRICE <un une chaine de caractères > : les choix possibles sont :
 - CARREE : la matrice de raideur sera stockée sous forme d'une matrice carrée, dont le nombre de colonne = nombre de ligne = nombre de degré de liberté du problème. Ce mode de stockage n'est pas en général économique en élément finis. Il faut le réserver pour des structures ayant un faible nombre de degrés de libertés, ou pour des cas particuliers : des tests par exemple.

TABLE 213 – liste des sous mots clés associés aux paramètres de contrôle liés à la dynamique

sout mot clé	valeur par défaut	ref du commentaire
TYPE_MATRICE	BANDE_SYMETRIQUE	(1)
SYMETRIE_MATRICE	1	(2)
TYPE_RESOLUTION	CHOLESKY	(3)
TYPE_PRECONDITIONNEMENT	DIAGONAL	(4)
NB_ITER_NONDIRECTE	30	(5)
TOLERANCE	1.e-7	(6)
NB_VECT_RESTART	32	(7)

- RECTANGLE : D’une manière théorique, ce stockage permet de différencier le nombre de colonne et le nombre de ligne. Dans les faits, l’utilisateur n’ayant aucun moyen pour l’instant de modifier ces données, le type de matrice utilisé sera identique au cas CARREE .
 - BANDE_SYMETRIQUE : la matrice de raideur sera stockée en bande. Seule la partie symétrique de la matrice sera sauvegardée. Dans le cas d’une raideur symétrique, ceci permet d’économiser de la place mémoire. Dans le cas d’une raideur a priori non symétrique, l’utilisation d’un stockage non symétrique entraîne une symétrisation forcée de la matrice : $\frac{(M(i,j)+M(j,i))}{2} \rightarrow M(i, j)$.
 - BANDE_NON_SYMETRIQUE : la matrice de raideur sera stockée en bande. Contrairement au cas BANDE_SYMETRIQUE, ici toute la matrice est stockée. Ce type de stockage est à retenir dans le cas de problème conduisant à des matrices fortement non symétriques, par exemple dans certain cas de couplages ou de frottement par exemple.
 - BANDE_NON_SYMETRIQUE_LAPACK : la matrice de raideur est stockée en bande complète. L’intérêt ici est d’utiliser la bibliothèque Lapack associé avec la bibliothèque blas, optimisée pour la machine utilisée. Par exemple sur machine apple, l’optimisation est automatique et on observe des vitesses de calcul bien supérieur (facteur 2 à 3) par rapport au cas BANDE_SYMETRIQUE ! La méthode classique de résolution est GAUSS, pour laquelle on a optimisation des pivots.
 - CREUSE_COMPRESSEE_COLONNE : la matrice de raideur sera stockée en matrice creuse, compressée colonne suivant le format Boeing-Harwell. Ce type de stockage est très économique, seules les termes non nuls étant stockés. Par contre la résolution à utiliser ne peut être que de type gradient conjugué ou dérivé, c’est-à-dire ne conduisant pas à une modification de la matrice pendant la résolution.
2. SYMETRIE_MATRICE <un booléen = 1 ou 0 > : Indique que l’assemblage est symétrique ou pas. Par exemple pour les matrices creuses compressées colonnes actuellement seul l’assemblage non symétrique est actuellement possible, il faut donc explicitement l’indiquer, la résolution ne fonctionnera pas dans le cas contraire. Pour les autres matrices c’est au choix.
 3. TYPE_RESOLUTION <une chaîne de caractères > : les choix possibles sont soit la méthode directe de Cholesky soit les autres méthodes qui sont itératives de type gradient conjugué.

- CHOLESKY : utilisable uniquement avec les matrices symétriques "BANDE_SYMETRIQUE" (stockées en symétrique ou non). Correspond à la résolution classique de la méthode de Cholesky. La résolution s'effectue en deux phases : tout d'abord il y a triangulation de la matrice, puis dans la seconde phase il y a résolution de deux problèmes linéaires triviaux constitués de matrices triangulaires.
 - GAUSS : utilisable uniquement avec les matrices BANDE_NON_SYMETRIQUE_LAPACK , il s'agit de la méthode classique "pivot de Gauss" et factorisation LU.
 - BL_CONJUG : utilisable avec tous les types de matrices symétriques ou non. Méthode du bi-gradient conjugué.
 - BL_CONJUG_STAB : Méthode du bi-gradient conjugué stabilisé, utilisable avec tous les types de matrices symétriques ou non.
 - CONJUG_GRAD : Méthode du gradient conjugué, utilisable avec les matrices symétriques.
 - CONJUG_GRAD_SQUARE : variante de la méthode du bi-gradient conjugué, utilisable avec tous les types de matrices symétriques ou non.
 - CHEBYSHEV : méthode des itérations de Chebyshev, utilisable les matrices symétriques définies positives.
 - GENE_MINI_RESIDUAL : Méthode de gradient conjugué qui utilise une minimisation au sens des moindres carrés, nécessite le stockage de résultat intermédiaire dont le nombre limite est fixé par le paramètre NB_VECT_RESTART, utilisable avec tous les types de matrices symétriques ou non.
 - ITERATION_RICHARSON : itération de Richardson, utilisable avec les matrices symétriques .
 - QUASI_MINI_RESIDUAL : variante de la méthode du bi-gradient conjugué stabilisé, qui a pour objectif de lisser les variations de convergence, utilisable avec tous les types de matrices symétriques ou non.
4. TYPE_PRECONDITIONNEMENT <une chaîne de caractères > : les méthodes itératives de type gradient conjugué nécessitent de préconditionner les matrices pour optimiser la résolution. Il s'agit de modifier la matrice originale pour accélérer la convergence. Les choix possibles sont :
- DIAGONAL : il s'agit du préconditionnement diagonal classique. En général l'efficacité est correcte sans plus !
 - ICP : cas de la factorisation de cholesky incomplete. Ce préconditionnement est très efficace.
 - ILU : cas de la factorisation LU : Ce préconditionnement est comme le précédent très efficace.
5. NB_ITER_NONDIRECTE < un entier > : s'utilise avec les méthodes itératives, spécifie le nombre d'itération maxi du processus de résolution que l'on tolère. Si la précision spécifiée est atteinte avant ce nombre maxi, le calcul a convergé normalement, sinon le calcul s'arrête pour le nombre maxi d'itération et on affiche la précision qui est atteinte. Souvent dans ce dernier cas la précision atteinte est suffisante et la résolution globale éléments finis peut continuer ! Néanmoins cela signifie que la convergence est difficile, et qui est peut-être souhaitable de changer de méthode ou d'adapter les paramètres de contrôle.

6. TOLERANCE < un réel > : s'utilise avec les méthodes itératives, spécifie la tolérance maxi sur le résidu de la résolution que l'on tolère. Cette valeur constitue donc la précision de la résolution du système linéaire.
7. NB_ITER_NONDIRECTE < un entier > : s'utilise avec la méthode GENE_MINI_RESIDUAL, spécifie le nombre de vecteur maxi de restart que l'on accepte de stocker.

Remarque : Le type de stockage de la matrice masse suit le principe suivant :

- soit on demande le calcul d'une matrice masse diagonale, dans ce cas quel que soit le type de stockage de la matrice de raideur, le stockage de la matrice masse sera naturellement sous forme d'une matrice diagonale.
- soit le calcul de la matrice masse conduit à une matrice masse non diagonale, dans ce cas le type de stockage adopté pour la matrice masse sera identique à celui de la matrice de raideur.

43 Pilotage de la résolution globale

Ce jeux de paramètres est facultatif. L'objectif de ce groupe de paramètre est de piloter la résolution globale. Avant chaque nouvel incrément on se pose la question : est-il possible d'augmenter le pas de chargement, ou au contraire lorsqu'il n'y a pas convergence, de combien faut-il diminuer l'incrément pour éviter une trop grande non-linéarité sur l'incrément et permettre une convergence.

Le fonctionnement de ce groupe de paramètres suit toujours la même logique que dans le cas des paramètres de contrôle généraux. Le mot clé est "para_pilotage_equi_global" suivi d'une liste de sous-mots clés, renseigné ou non par une valeur. L'exemple suivant présente une liste de paramètres.

```

                para_pilotage_equi_global -----
#-----
# PARAMETRE          |          VALEUR          |
#-----
FACTEUR_DIMINUTION      1.732
FACTEUR_AUGMENTATION    1.414
NB_BONNE_CONVERGENCE    3
INIT_COMP_TANGENT_SIMPLE -1
SUR_SOUS_RELAXATION     1.2

```

La liste exhaustive des sous-mots clés disponible est donné dans la table (214).

Avec les commentaires suivants :

1. TYPE_DE_PILOTAGE < une chaîne de caractère > : Indique quel type de pilotage on désire. Pour l'instant deux cas sont possible, soit un pilotage basique (PILOTAGE_BASIQUE) qui utilise les paramètres : FACTEUR_DIMINUTION, FACTEUR_AUGMENTATION, NB_BONNE_CONVERGENCE, NB_ITER_POUR_BONNE_CONVERGENCE

TABLE 214 – liste des sous mots clés associés aux paramètres de contrôle du pilotage

sout mot clé	valeur par défaut	ref du commentaire
TYPE_DE_PILOTAGE	PILOTAGE_BASIQUE	(1)
FACTEUR_DIMINUTION	$\sqrt{(3)}$	(2)
FACTEUR_AUGMENTATION	$\sqrt{(2)}$	(3)
NB_BONNE_CONVERGENCE	3	(5)
FACT_DIM_EN_MAUVAISE_CONVERGENCE	3	(4)
NB_ITER_POUR_BONNE_CONVERGENCE	0.25 ITERATIONS	(6)
NB_ITER_POUR_MAUVAISE_CONVERGENCE	0.5 ITERATIONS	(7)
INIT_COMP_TANGENT_SIMPLE	-1	(8)
SUR_SOUS_RELAXATION	1	(9)
NORME_MAXI_INCREMENT	nombre très grand	(10)
MAXI_VARIATION_DDL_POUR_CONVERGENCE	nombre très grand	(11)
MINI_VARIATION_DDL_POUR_CONVERGENCE	0.	(12)
NB_CYCLE_CONTROLE_RESIDU	3	(13)
PLAGE_CONTROLE_RESIDU	3	(14)
INIT_INCRE_AVEC_PAS_PREC	1	(15)
CAS_JACOBIEN_NEGATIF	1	(16)
VAR_MAXI_JACOBIEN	0.	(17)

Le second cas est indiqué par le mot : AUCUN_PILOTAGE. Dans ce dernier cas il n’y a pas de pilotage, ceci signifie, que les paramètres FACTEUR_DIMINUTION, FACTEUR_AUGMENTATION, NB_BONNE_CONVERGENCE, NB_ITER_POUR_BONNE_CONVERGENCE, ne sont pas utilisés. Du début jusqu’à la fin du calcul, de pas de temps reste fixe. Et enfin le pilotage (PILOT_GRADIENT) qui regarde l’évolution du résidu. Soit $PLAGE_CONTROLE_RESIDU = \alpha$ et soit $NB_CYCLE_CONTROLE_RESIDU = \beta$. Si le résidu ne diminue pas sur une plage α d’itération c’est-à-dire $residu(n+\alpha) > residu(n)$, n étant le nombre d’itération, on considère qu’un premier critère n’est pas rempli. Ensuite si ce premier critère n’est pas rempli β fois, là on considère qu’il y a divergence.

2. FACTEUR_DIMINUTION <un nombre réel > : Ce nombre indique de combien il faut diminuer l’incrément de temps lorsque l’algorithme conduit à cette augmentation.
3. FACTEUR_AUGMENTATION <un nombre réel > : Ce nombre indique de combien il faut augmenter l’incrément de temps lorsque l’algorithme conduit à cette augmentation.
4. FACT_DIM_EN_MAUVAISE_CONVERGENCE <un nombre réel > : Ce nombre indique de combien il faut diminuer l’incrément de temps lorsque l’on a eu une convergence mais pas de bonne qualité (c’est-à-dire avec un nombre d’itération supérieur à NB_ITER_POUR_MAUVAISE_CONVERGENCE).
5. NB_BONNE_CONVERGENCE <un nombre entier > : Ce nombre indique le nombre de fois consécutives, ou il y a ”bonne convergence”, à partir duquel on augmente l’incrément de temps.

6. NB_ITER_POUR_BONNE_CONVERGENCE <un nombre entier > : Lorsqu'il y a convergence pour un nombre d'itération inférieur ou égale à ce nombre, on considère qu'il y a eu "bonne convergence".
7. NB_ITER_POUR_MAUVAISE_CONVERGENCE <un nombre entier > : Ce nombre indique le nombre d'itération au dessus duquel on considère que la convergence si elle se fait est de toute manière mauvaise (ou pas très bonne!).
8. INIT_COMP_TANGENT_SIMPLE < un entier > : Ce nombre indique combien pendant combien d'itération, on utilise un calcul simplifié de la loi de comportement. En général il s'agit d'un comportement tangent du type $\sigma = \mathbf{T} \ \varepsilon$ qui permet par sa simplicité de stabiliser rapidement la géométrie de la pièce, avant d'utiliser un comportement plus complexe.
9. SUR_SOUS_RELAXATION <un nombre réel > : ce facteur est utilisé uniquement dans le cas d'un calcul implicite sans contact, statique. Il permet d'augmenter arbitrairement la valeur de l'accroissement de ddl qui a été déterminé dans une itération de Newton-Raphson. Ainsi à l'itération suivante, pour $(ddl_{n+1} = (ddl_n) + facteur * (\delta ddl)$. Dans le cas normal le facteur vaut 1.
10. NORME_MAXI_INCREMENT <un nombre réel > : Dans le cas d'une divergence, où la norme des ddl tant vers une valeur supérieur au facteur, l'ensemble du vecteur ddl est modifié selon la formule : $(ddl) \rightarrow facteur * (ddl) / \|(ddl)\|$. Cela signifie que l'on garde la direction de l'incrément de ddl obtenu par la résolution de l'équation d'équilibre, par contre on en limite la norme. Dans le cas où l'on indique un nombre négatif, le fonctionnement est différent. En fait pour toutes les composantes du vecteur ddl dont la valeur absolue est inférieure au facteur, il n'y a pas de modification. Dans le cas contraire la valeur des composantes est plafonnée au facteur muni du signe de la composante initiale.
11. MAXI_VARIATION_DDL_POUR_CONVERGENCE <un nombre réel > : définit la limite maxi autorisée pour les variations de ddl. Au delà de cette limite on considère que le calcul diverge. Voir (9) pour l'utilisation de ce paramètre.
12. MINI_VARIATION_DDL_POUR_CONVERGENCE <un nombre réel > : définit la limite mini autorisée pour les variations de ddl. En dessous de cette limite on considère que le calcul diverge. Voir (9) pour l'utilisation de ce paramètre.
13. NB_CYCLE_CONTROLE_RESIDU < un entier >, et PLAGES_CONTROLE_RESIDU < un entier > : définissent les paramètres de réglage du contrôle de la variation de résidu. Ce paramètre est utilisé avec le type de pilotage PILOT_GRADIENT : voir (voir 1) pour l'utilisation de ces paramètres.
14. PLAGES_CONTROLE_RESIDU < un entier > : indique combien de fois consécutive il faut que le contrôle de variation de résidu soit faux pour conclure à la non convergence (voir 1). Ce paramètre est utilisé dans le cas du pilotage : PILOT_GRADIENT.
15. INIT_INCRE_AVEC_PAS_PREC < un coefficient réel "coef" > typiquement compris entre 1 ou 0 (0 le facteur est inactif) : indique si l'on veut initialiser la position finale proposée à la première itération à l'aide de l'incrément de ddl du pas précédent multiplié par le coefficient. Ainsi supposons qu'au pas précédent (n-1) on a trouvé un incrément de ddl : $\delta ddl(n-1) = ddl_{t+\Delta t, (n-1)} - ddl_{t, (n-1)}$ ceci

pour un incrément de temps Δt_{n-1} . Au pas suivant on initialise les ddl finaux selon $ddl_{t+\Delta t, (n)} = ddl_{t, (n)} + coef * \delta ddl(n-1) * \Delta t_n / \Delta t_{n-1}$ avec "coef" le coefficient indiqué. La méthode sert ainsi à définir une prédiction pour le pas suivant, obtenue par extrapolation linéaire du pas précédent. Cette initialisation ne fonctionne que pour les algorithmes globaux statiques (pour l'instant, Newmark en est exclus).

16. CAS_JACOBIEN_NEGATIF < un entier > : permet d'indiquer ce que l'on fait lorsque l'on rencontre le cas d'un jacobien négatif.
 - = 0 : on ne tient pas du tout compte du jacobien négatif.
 - = 1 : au niveau du calcul du second membre et/ou de la raideur on annule la contribution du point d'intégration se rapportant au jacobien négatif.
 - = 2 : Dans le cas d'un calcul implicite, on considère qu'il y a divergence d'équilibre dès le premier jacobien négatif rencontré. La suite dépend du type de pilotage retenu.
17. VAR_MAXI_JACOBIEN < un nombre réel > : n'est valide que s'il est supérieur strictement à 1. Indique la valeur maxi de la variation de jacobien (en valeur absolue) autorisé pendant le calcul. Si le nouveau jacobien J est supérieur au jacobien initiale J_0 c'est le rapport J/J_0 qui est comparé à VAR_MAXI_JACOBIEN sinon c'est le rapport J_0/J qui est utilisé. Lorsque le rapport réel est supérieur à la limite imposé, on considère qu'il y a divergence. La suite dépend du type de pilotage retenu.

44 Pilotage du contact

Ce jeux de paramètres est facultatif.

Le fonctionnement de ce groupe de paramètres suit toujours la même logique que dans le cas des paramètres de contrôle généraux. Le mot clé est "para_contact" suivi d'une liste de sous-mots clés, renseigné ou non par une valeur. Voici un exemple de déclaration :

```

                para_contact -----
#-----
# PARAMETRE          |  VALEUR  |
#-----
PRECISION_POINT_INTERNE_DEBUT          1.e-2

```

La liste exhaustive des sous-mots clés disponible est donné dans la table (215).

Avec les commentaires suivants :

1. PRECISION_POINT_INTERNE_DEBUT < un réel > indiquait la précision utilisée par le programme pour déterminer si un point est interne ou non à la matière d'un autre solide avant le démarrage du calcul. Ce paramètre n'est plus utilisé, le calcul utilise maintenant les paramètres de calculs géométriques cf. 46, en particulier les paramètres : POINT_INTERNE_DELTA_THETA_MAXI et POINT_INTERNE_PREC_THETA_INTE
2. CONTACT_TYPE < un entier > indique tout d'abord s'il y a contact (paramètre différent de 0) ou pas (paramètre = 0) puis le type de contact (le numéro indique le type de contact). Dans le cas où le type de contact = 1, il s'agit d'une méthode sans

TABLE 215 – liste des sous mots clés associés aux paramètres de contrôle liés à la dynamique

sout mot clé	valeur par défaut	ref du commentaire
PRECISION_POINT_INTERNE_DEBUT	(obsolete) 1.e-6	(1)
CONTACT_TYPE	0	(2)
PENALISATION_PENETRATION	0.1	(3)
TYPE_PENALISATION_PENETRATION	2	(4)
PENALISATION_TANGENTIELLE	1.e6	(5)
TYPE_DE_DECOLLEMENT	0	(6)
NB_DECOLLEMENT_MAXI	1	(7)
PRECISION_BOITE_PRELOCALISATION	1.05	(8)
MINI_EXTRA_BOITE_PRELOCALISATION	0.001	(9)
PENETRATION_CONTACT_MAXI	(obsolete) 0.1	(10)
PENETRATION_BORNE_REGULARISATION	un nombre très grand	11
FORCE_CONTACT_NOEUD_MAXI	un nombre très grand	12
DISTANCE_MAXI_AU_PT_PROJETE	un nombre très grand	13
FACT_POUR_RAYON_ACCOSTAGE	1	14

multiplicateur de Lagrange ni pénalisation, utilisable en implicite avec l’algorithme particulier : “non_dyna_contact” cf. (III). Dans le cas où le type de contact = 2, il s’agit d’une méthode avec pénalisation utilisable avec la plupart des algorithmes (sauf indication contraire dans la description de l’algorithme).

3. PENALISATION_PENETRATION <un réel α > indique un des éléments permettant le calcul du facteur de pénalisation qui est employé pour définir la force de réaction due à une pénétration $\Delta\vec{X}$ tel que la force vaut $\vec{F} = -\alpha\beta\Delta\vec{X}$. Ce paramètre n’est utilisé que dans une méthode associant la pénalisation. Le paramètre β est défini par le mot clé TYPE_PENALISATION_PENETRATION.
4. TYPE_PENALISATION_PENETRATION <un entier ”indic”> qui indique la méthode utilisée pour calculer le facteur β . En fonction de la valeur de ”indic” on a :
 - =1 : dans ce cas $\beta = 1$, ce qui veut dire que α est directement le facteur de pénalisation,
 - =2 : deux cas sont à distinguer : soit le noeud esclave impacte un élément solide, soit il impacte un élément surfacique.
 Dans le cas de l’impact sur la surface A_e de l’élément solide de volume V_e dont le module de compressibilité volumique vaut K_e on a :

$$\beta = \frac{K_e A_e^2}{V_e}$$

Dans le cas de l’impact sur un élément coque ou plaque de surface A_e , dont le module de compressibilité volumique vaut K_e on a :

$$\beta = \frac{K_e A_e}{\max(\text{distance entre les noeuds})}$$

=3 : même fonctionnement que le cas 2, mais avec l'algorithme supplémentaire suivant. On sauvegarde d'un pas sur l'autre la pénétration. A chaque fois que la pénétration augmente on double la valeur du facteur de pénalisation calculé. Ainsi après "n" augmentations successives de la pénalisation, le facteur initiale est multiplié par 2^n (pour mémoire, $2^{10} = 1024 \approx 10^3$ et $2^{100} \approx 10^{30}$!). Dans le cas où la pénétration diminue, "n" diminue selon le même algorithme. Cette méthode (exploratoire) permet dans certain cas de mieux maîtriser les pénétrations importantes.

=5 : calcul d'un facteur β_{base} identiquement au cas 2 puis on a l'algorithme supplémentaire suivant :

- si la pénétration "e" est telle que : $-e_{regul} \leq e \leq e_{regul}$ alors : $\beta = \beta_{base} \times (e - e_{regul})^2 / (e_{regul})^2$, fonction qui varie de 1. (pour $e = -e_{regul}$) à 0. (pour $e = e_{regul}$)
- si $e > e_{regul}$ dans ce cas $\beta = 0$
- si $e < -e_{regul}$ dans ce cas $\beta = \beta_{base}$

La grandeur e_{regul} est donnée par le paramètre PENETRATION_BORNE_REGULARISATION (cf.11).

5. PENALISATION_TANGENTIELLE <un réel α_t > indique le facteur de pénalisation qui est employé pour définir la force tangentielle de réaction due à un déplacement tangentiel $\Delta \vec{X}_t$ tel que la force vaut $\vec{F}_T = -\alpha_t \Delta \vec{X}_t$. Ce paramètre n'est utilisé que dans d'une méthode associant la pénalisation d'une part et d'autre part que dans le régime "collant" c'est-à-dire durant l'étape où il n'y a pas glissement. Si ce paramètre n'est pas présent, par défaut il est mis égal au facteur de pénalisation en pénétration.
6. TYPE_DE_DECOLLEMENT <un entier "m" > qui indique la méthode utilisée pour identifier le décollement ou non.
 - $m=0$: (valeur par défaut) le décollement est identifié lorsque la réaction de contact devient positive,
 - $m \neq 0$: le décollement est identifié lorsque la réaction devient positive "et" la distance entre la surface maître et le noeud esclave est supérieur à $m \times e_{regul}$ (cf. 11)
7. NB_DECOLLEMENT_MAXI <un entier "n" > qui indique le nombre consécutif de fois que le test de décollement du noeud doit être satisfait, avant que le contact soit supprimé.
8. PRECISION_BOITE_PRELOCALISATION <un réel > = α' . Pour optimiser la recherche des noeuds entrant en contact, on définit pour chaque face d'élément frontière, une boite d'encombrement qui est égale à la boite rectangulaire minimale contenant la face concernée. Cette boite est tout d'abord définie dans Herezh par ces deux points extrêmes \vec{X}_{max} et \vec{X}_{min} . Puis pour prendre en compte les incertitudes de position et le fait qu'une dimension (l'épaisseur) puisse être nulle, elle est agrandie de la manière suivante :
 - tout d'abord sa taille est multiplier par le facteur α' . A priori on doit donc avoir $\alpha' \geq 1$. Sans raison particulière, il est préférable de laisser le paramètre par défaut.

- puis la boîte est agrandie dans toutes les directions d’une même valeur : $\beta' \times \text{MAX}_{i=1}^{\text{dim}} | X_{\text{max}}^i - X_{\text{min}}^i |$. Là aussi il est préférable de laisser la paramètre β' (cf.9) par défaut.

Ces paramètres (α' et β') sont également utilisés pour définir la boîte d’encombrement de l’élément géométrique attaché à la facette. Cette boîte sert pour la détection des noeuds internes (à la matière) au début du calcul, et également pendant certaine phase de la détection de contact.

9. MINLETRA_BOITE_PRELOCALISATION <un réel > = β' . (cf.8)
10. PENETRATION_CONTACT_MAXI <un réel e_{max} > : ce paramètre est obsolète, actuellement il n’est plus utilisé.
11. PENETRATION_BORNE_REGULARISATION <un réel e_{regul} > : qui indique une borne de régularisation sur la penetration (dont l’utilisation dépend de l’algorithme qui gère la pénalisation (cf. 4 et 6).
12. FORCE_CONTACT_NOEUD_MAXI <un réel f_{max} > : qui indique une force maxi pour la force de réaction par pénalisation. Si la force calculée est supérieure, elle est réduite arbitrairement à la valeur f_{max} .
13. DISTANCE_MAXI_AU_PT_PROJETE <un réel d_{max} > : qui indique la distance maxi admissible entre le noeud et sa projection (ou plutôt l’intersection de sa trajectoire avec la facette cible). Dans le cas où la distance relevée pendant le calcul, est plus grande que d_{max} on considère que le contact est erroné, donc on n’en tient pas compte. En fait, la distance qui est considérée est $\min(d_{\text{max}}, 2.\alpha \|\Delta X\|_{\infty})$ où $\alpha \|\Delta X\|_{\infty}$ est détaillé en 14.
14. FACT_POUR_RAYON_ACCOSTAGE <un réel α > : qui sert à calculer la distance maxi admissible entre le noeud et sa projection. Cette distance doit être inférieur à $\alpha \|\Delta X\|_{\infty}$ pour être licite, ΔX étant la variation des coordonnées. $\alpha >$ est utilisé conjointement avec d_{max} (cf.13). La distance maxi admissible entre le noeud et sa projection étant : $\min(d_{\text{max}}, 2.\alpha \|\Delta X\|_{\infty})$.

45 Affichage des résultats

Ce jeux de paramètres est facultatif. L’objectif de ce groupe de paramètre est de piloter l’affichage en général à l’écran. Dans certain cas, cet affichage peut ralentir considérablement l’avancement du calcul, alors qu’il est en fait difficilement exploitable. Par exemple, lors d’un calcul explicite utilisant un maillage avec peu de noeuds et d’éléments, mais nécessitant un grand nombre d’incrément, plus de 10000 par exemple. En général l’affichage est tellement rapide que l’on ne peut le lire, alors que cela ralentit beaucoup l’avancement du calcul. Il est dans cas plus intéressant de demander un affichage tous les n incréments, et de même lorsqu’il y a beaucoup d’itération pour un calcul implicite on peut également d’une manière analogue limiter le nombre d’itération affiché.

Deux paramètres sont donc introduit pour gérer cet affichage. Ce groupe de paramètre suit la même logique que dans le cas des paramètres de contrôle généraux. Le mot clé est ”para.affichage” suivi d’une liste de sous-mots clés, renseigné ou non par une valeur.

Il est également possible de changer le nombre de chiffre significatif utilisé pour afficher les nombres en double précision. Par défaut l’affichage s’effectue avec 22 chiffres significatifs ce qui est juste un peu supérieur au nombre de diggit utilisé sur macos. Dans le cas d’un restart, on est ainsi sûr de ne pas perdre d’information. Par contre lorsque la sauvegarde n’est utilisée que pour la visualisation, 10 chiffres significatifs sont suffisants ce qui diminue de moitié la taille des fichiers de restart ! En fait deux paramètres sont proposés : un pour toutes les sorties relatives à la visualisation, le second spécifiques à l’archivage.

```

                para_affichage -----
#-----
# PARAMETRE      |  VALEUR      |
#-----
FREQUENCE_AFFICHAGE_INCREMENT      500
FREQUENCE_AFFICHAGE_ITERATION      1

```

Il est également possible de régler la fréquence de sortie des données au fil du calcul ce qui permet un affichage au cours du calcul. Ainsi la fréquence de sauvegarde peut être différentes de la sortie des données au fil de calcul. Par exemple on peut vouloir un affichage continue au fil du calcul (tous les incréments par exemple), alors que la sauvegarde soit plus espacé (tous les 10 incréments par exemple). Cependant il faut noter que le post-traitement ne s’effectuera que sur les incréments sauvegardés.

La liste exhaustive des sous-mots clés disponible est donné dans la table (216).

TABLE 216 – liste des sous mots clés associés aux paramètres de contrôle du pilotage

sout mot clé	valeur par défaut	ref du commentaire
FREQUENCE_AFFICHAGE_INCREMENT	1	(1)
FREQUENCE_AFFICHAGE_ITERATION	1	(2)
FREQUENCE_SORTIE_FIL_DU_CALCUL	-1000000	(3)
NB_CHIFFRE_POUR_DOUBLE_CALCUL	22	(4)
NB_CHIFFRE_POUR_DOUBLE_GRAPHIQUE	12	(5)

Avec les commentaires suivants :

1. FREQUENCE_AFFICHAGE_INCREMENT <un nombre entier relatif n > : Dans le cas ou le nombre "n" est positif, il indique la fréquence des itérations que l’on affiche, c’est-à- dire que l’affichage se fera tous les "n" incréments. Dans le cas où "n" est négatif cela indique que la fréquence d’affichage sera la même que la fréquence de sauvegarde (cf. 210, le paramètre SAUVEGARDE) dans le cas ou la fréquence de sauvegarde est supérieur à 0. Dans le cas ou la fréquence de sauvegarde est inférieur à 0 il n’y a aucun affichage ! Dans le cas où la fréquence de sauvegarde est telle que seul de dernier incrément est demandé, et que la fréquence d’affichage est inférieur ou égal à 0, l’affichage se fait tous les incréments.
2. FREQUENCE_AFFICHAGE_ITERATION <un nombre entier relatif m > : Le nombre "m" indique qu’il y aura affichage tous les "m" itérations. Si "m" est négatif, ou nul, il n’y a aucun affichage.

3. FREQUENCE_SORTIE_FIL_DU_CALCUL <un nombre entier positif m > : donne la fréquence de sortie de résultats au fil du calcul. En fait, il y a plusieurs autres possibilités pour affiner cette sortie au fil du calcul. On se reportera au paragraphe (47.4) pour plus de précision à se sujet.
4. NB_CHIFFRE_POUR_DOUBLE_CALCUL <un nombre entier positif m > : donne le nombre de chiffre significatif utilisé par le programme pour afficher les nombres en double précision sur fichier ou à l'écran (quand la sortie est formatée), en particulier utilisé pour le fichier de restart.
5. NB_CHIFFRE_POUR_DOUBLE_GRAPHIQUE <un nombre entier positif m > : donne le nombre de chiffre significatif utilisé par le programme pour l'affichage des nombres en double précision utilisé pour les sorties graphiques.

46 Calculs géométriques

Ce jeux de paramètres est facultatif. L'objectif de ce groupe de paramètre est de piloter des calculs géométriques, qui sont relativement indépendant des calculs mécaniques. Le mot clé est "para_calculs_geometriques" suivi d'une liste de sous-mots clés, renseigné ou non par une valeur.

Un premier groupe de paramètres est relatif à une action (une méthode) qui recherche si un point est interne à un élément. Les paramètres associés ont un identificateur qui débute par "POINT_INTERNE_".

La table (217) donne une liste exhaustive des différents paramètres et de leur signification.

```

                para_calculs_geometriques -----
#-----
# PARAMETRE      | VALEUR      |
#-----
POINT_INTERNE_DELTA_THETA_MAXI      1.e-4
POINT_INTERNE_PREC_THETA_INTERNE    1.e-6
POINT_INTERNE_NB_BOUCLE_SUR_DELTA_THETA  10
POINT_INTERNE_NB_EXTERNE              4

```

TABLE 217 – liste des sous mots clés associés aux paramètres de contrôle du pilotage

sout mot clé	valeur par défaut	ref du commentaire
POINT_INTERNE_DELTA_THETA_MAXI	1.e-4	(1)
POINT_INTERNE_PREC_THETA_INTERNE	1.e-6	(2)
POINT_INTERNE_NB_BOUCLE_SUR_DELTA_THETA	10	(3)
POINT_INTERNE_NB_EXTERNE	3	(4)
CAL_VOL_TOTAL_ENTRE_SURFACE_ET_PLANS_REF	0	(5)
TYPE_CALNUM_INVERSION_METRIQUE	CRAMER	(6)

Avec les commentaires suivants :

1. POINT_INTERNE_DELTA_THETA_MAXI <un nombre réel > : indique la précision que l'on veut sur la recherche des coordonnées locales θ^i par rapport à un élément donné. Bien voir que la précision réelle est fonction de la taille de l'élément, car les $|\theta_{maxi}^i|$ sont en général de l'ordre de 1 pour la plupart des éléments.
2. POINT_INTERNE_PREC_THETA_INTERNE <un nombre réel " e " > : indique la précision à laquelle on détermine si un point est interne, sur la surface ou extérieur à l'élément. D'une manière plus précise une fois calculées les coordonnées locales $\theta^i(M)$ d'un point M, on peut déterminer en fonction de la géométrie de l'élément de référence, l'appartenance stricte ou non du point à l'élément. Cependant, si le point appartient à l'élément, deux sous-cas sont distingués : si $||\theta^i(M) - \theta^i(point\ frontiere)|| < e$ alors on considère que le point est sur la frontière, sinon le point est réputé à l'intérieur de l'élément. Ceci explique pourquoi on peut avoir une précision POINT_INTERNE_PREC_THETA_INTERNE éventuellement plus faible que celle POINT_INTERNE_DELTA_THETA_MAXI (?)
3. POINT_INTERNE_NB_BOUCLE_SUR_DELTA_THETA <un nombre entier positif m > : nombre de boucle maxi pour la recherche des coordonnées locales θ^i par rapport à un élément donné.
4. POINT_INTERNE_NB_EXTERNEL <un nombre entier positif m > : nombre de fois qu'un point est trouvé extérieur avant le quel on déclare qu'il est vraiment à l'extérieur. Ce calcul est effectué dans la même boucle que celle du calcul des coordonnées locales θ^i du point par rapport à un élément donné. Ainsi, la recherche se termine si : soit la précision est atteinte, soit le nombre maxi de boucle est atteint, soit on a trouvé le point extérieur à l'élément "m" fois.
5. CAL_VOL_TOTAL_ENTRE_SURFACE_ET_PLANS_REF : suivi de "1" ou "0" (valeur par défaut). Indique si oui ou non, on veut le calcul du volume situé entre les plans de base (xy, xz, yz) et la surface. Ne fonctionne que pour des structures 2D (membrane, coques, plaques) qui sont calculées en 3D. Une fois cette option activée, on obtient l'accès à 3 nouvelles grandeurs globales en post-traitement (format .maple) : vol_total2D_avec_plan_yz, vol_total2D_avec_plan_xz, vol_total2D_avec_plan_xy.
6. TYPE_CALNUM_INVERSION_METRIQUE : suivi d'un indicateur qui donne le type de méthode numérique à utiliser pour inverser le tenseur métrique. Deux méthodes sont disponibles : la méthode historique qui utilise la méthode des déterminants (méthode de Cramer) : indicateur "CRAMER". La deuxième méthode implantée, utilise une décomposition LU avec un équilibrage des lignes : indicateur "LU_EQUILIBRE". Cette dernière méthode entraîne a priori moins d'erreur d'arrondi, lorsque le conditionnement des matrices des composantes des tenseurs, sont mal conditionnées.

Treizième partie

Sortie des résultats

47 Sortie des résultats

47.1 Introduction

L'accès aux résultats s'effectue soit à l'aide de fichiers de valeurs, soit via une visualisation graphique. Pour éviter d'être lié à un programme particulier de visualisation graphique, le formatage des résultats graphiques c'est porté sur des standards :

- le standard vrml qui est un format texte, utilisable par la plupart des navigateurs de web, après avoir récupéré un PLUGING vrml par exemple sur le site de silicon graphics, ou Cortoma.
- le format geomview. En fait geomview est un programme de visualisation de données graphiques, très général. Ce programme est disponible gratuitement sur les plateformes : Unix, Linux et Mac osX.
- le format GID. Le logiciel GID permet de faire du pré et post-traitement de calcul éléments finis. Une version gratuite et limitée en nombre d'éléments est disponible sur le web pour les plates-formes : Unix, Linux et Mac osX, Windows.
- le format gmsh. Le logiciel gmsh (www.geuz.org/gmsh/) est un logiciel libre, disponible sur le web pour toutes les plates-formes.

On parlera également du format "Maple". En fait, il s'agit d'un format texte particulier, qui est constitué essentiellement de tableaux multi-colonnes d'informations exploitables graphiquement par les logiciels classiques de visualisation de courbes en 2D ou 3D voir même avec une évolution avec un paramètre temps. Par exemple, on pourra utiliser les logiciels :

- Gnuplot : disponible sur Windows, Unix, Linux, et Mac osX ou 9,
- Exel : disponible sur Windows ou Mac, et d'une manière générale tous les tableaux qui permettent un post-traitement graphique,
- Xmgr ou Xmgrace sur Linux et Mac osX,
- Maple sur toutes plateformes,
-

Maple ayant été le premier logiciel interfacé, j'ai gardé le type "format Maple" pour distinguer ce type de sortie des autres.

Remarque *De plus une interface partielle "historique" est également implantée avec le logiciel Livan qui utilise également le programme de visualisation Geomview, disponible sur station UNIX ou LINUX.*

47.2 Sortie des résultats sur fichiers

À la suite de l'exécution du programme, un certain nombre de fichiers sont générés de manière automatique. Le nom des fichiers est identique au fichier .info, avec le suffixe .info remplacée par un suffixe particulier à chaque type de sortie. Il s'agit :

- suffixe .reac : ce fichier texte contient les réactions des noeuds bloqués. En effet le fait d'imposer un ddl, engendre une réaction dans la structure qui est calculée de manière à être en équilibre avec l'ensemble des sollicitations.

- suffixe .ddl : le fichier contient les valeurs des degrés de liberté aux noeuds et leurs coordonnées. On trouve les coordonnées initiales, les déplacements et les coordonnées finales. Concernant les ddls, on trouve leurs valeurs initiales, leurs valeurs finales et la variation initiale et finale.

Dans le cas d'un problème de mécanique statique, les positions des noeuds sont les degrés de liberté du système, en sortie on obtient alors ces informations en double.

- suffixe .res : ce fichier contient les grandeurs calculées aux points d'intégrations. Dans le cas de la mécanique, il s'agit des grandeurs relatives aux coordonnées du tenseur de déformation, aux coordonnées du tenseur de contrainte, et aux coordonnées du point d'intégration sur l'élément réel avant et après transformation.

De manière à rendre la lecture de ce fichier plus aisé, il est possible de choisir quelles informations ont désire aux points d'intégration. Pour cela on indique à la fin du fichier .info, après le mot clé "resultats", sur la ligne suivant on indique le mot clé "POINTS_INTEGRATION" suivi d'une référence d'une liste d'éléments, les éléments pour lesquels on désire des informations, puis sur la ligne suivante une liste de sous-mots-clés qui indique le type des informations que l'on veut à chaque point d'intégration.

D'une manière exhaustive on peut avoir :

- les coordonnées du tenseur de déformation de Green-Lagrange dans le repère global, sous-mot-clé : "Green-Lagrange"
- les coordonnées du tenseur de déformation d'Almansi dans le repère global, sous-mot-clé : "Almansi"
- les coordonnées du tenseur de vitesse de déformation dans le repère global, sous-mot-clé : "Vitesse_deformation"
- les coordonnées du tenseur de l'incrément de déformation courant d'Almansi dans le repère global, sous-mot-clé : "Increment_deformation"
- les coordonnées locales du tenseur de contrainte de Cauchy en mixte, sous-mot-clé : "Cauchy_global"
- les coordonnées locales du tenseur de déformation d'Almansi en mixte, sous-mot-clé : "Def_mixte_local"
- les valeurs des contraintes principales de Cauchy et la contrainte de Mises, sous-mot-clé : "Sigma_principale"
- les valeurs des déformations principales d'Almansi, sous-mot-clé : "Def_principale"
- les valeurs principales des vitesses de déformations, sous-mot-clé : "VitDef_principale"

Un exemple d'indication de sortie de résultats dans le fichier .info :

```

                resultats -----
#-----
# PARAMETRE      |  VALEUR      |
#-----

COPIE                0
POINTS_INTEGRATION ELEMENTS

```

Green-Lagrange Almansi Cauchy_global Def_mixte_local Sigma_mixte_local

On ne demande pas de recopie du fichier d'entrée à l'écran, le mot clé "COPIE" est suivi de 0, ensuite on indique que l'on veut des informations aux points d'intégration pour les éléments de la liste "ELEMENTS". Les informations demandées sont : les coordonnées des tenseurs de Green-Lagrange, d'Almansi, de Cauchy dans le repère global, et également les coordonnées locales de contraintes et de déformation en mixte.

47.2.1 Aucune sortie demandée

Dans certains cas, il est préférable d'interdire la sortie de résultats finale. Par exemple, lorsque l'on utilise un utilitaire de maillage, qui va conduire à créer un nouveau maillage, mais incomplet (dans le sens, qu'il ne contient pas par exemple de loi de comportement). Dans ce cas, aucun calcul mécanique n'a été réalisé et la sortie d'information aura des manques et affichera des messages d'erreurs ou de Warning. Pour éviter cette sortie automatique, on indique uniquement la ligne suivante :

```
resultats    pas_de_sortie_finale_ # -----
```

Aucun fichier de sortie automatique n'est alors rempli.

Remarque Il est néanmoins possible de laisser le texte correspondant à la définition de COPIE et de la sortie aux points d'intégrations. Il n'y aura pas d'erreur générée, mais aucune action ne sera faite (sauf si COPIE = 1, dans ce cas il y a recopie des infos lues à l'écran).

47.3 Sortie des résultats pour une visualisation graphique directe.

Cette sortie de résultats s'effectue soit en interactif à la fin du calcul soit d'une manière automatique à partir d'un fichier de commande contenant les paramètres de post-traitement graphique.

Pour avoir accès à la sortie graphique il est nécessaire d'indiquer une option au début du fichier .info, par exemple pour un calcul en statique on indiquera :

```
non_dynamique avec plus visualisation
```

La liste des options de post-traitement graphique est donnée par le tableau [218].

Entre chaque option de post-traitement, il ne faut pas oublier de mettre les deux mots clés : "avec plus". Il est possible de cumuler plusieurs mots clés, dans ce cas l'ensemble des mots clés doit être sur une même ligne. Ainsi l'exemple suivant :

```
non_dynamique avec plus lectureCommandesVisu avec plus visualisation avec plus sauveCommandesVisu
```

TABLE 218 – liste des différentes options de post-traitement graphique, a indiquer après le type de calcul

identificateur	ref du commentaire
visualisation	menus interactifs après le calcul
sauveCommandesVisu	création d'un fichier de commande de visualisation
lectureCommandesVisu	lecture d'un fichier de commande et exécution automatique

signifie que l'on commence par récupérer un fichier de commande et exécuter les commandes lues, ensuite le programme passe en mode interactif, ce qui permet de modifier éventuellement certains paramètres lus dans le fichier de commande. Ensuite le jeu de nouveaux paramètres est sauvegardé dans le fichier de commande.

Dans le cas d'une demande de travail interactive, à la fin de l'exécution du programme apparaîtra le menu suivant :

```

===== choix du module de visualisation interactive =====
sauvegarde des commandes de visualisation      ? (rep 1)
visualisation automatique                      ? (rep 2)
visualisation au format vrml ?                 (rep 3)
visualisation par fichier de points, format maple ? (rep 4)
visualisation au format geomview              ? (rep 5)
visualisation au format Gid                   ? (rep 6)
fin                                             (rep 0 ou f)
reponse ?

```

Suivant la réponse indiquée, on accède à une série de sous-menus qui permettent de préparer la sortie graphique finale. Les différents formats sont explicités dans les paragraphes suivants.

Rappelons cependant que Herezh n'effectue lui-même aucune visualisation graphique, en fait il construit un fichier texte contenant les informations graphiques nécessaires pour une visualisation directe à l'aide d'un autre programme spécialisé entre autres dans l'affichage graphique : geomview, Exel, navigateur avec plugin VRML etc..

Rappelons également que le format maple est plutôt dédié aux graphes, tandis que les deux autres formats sont plutôt prévus pour des visualisations volumiques.

47.3.1 Utilisation des fichiers de commande

Il est possible de sauvegarder les paramètres de visualisation dans un fichier, de manière à être réutilisées par la suite. Ce fichier a pour objectif principal de simplifier la sortie graphique, lorsque l'on doit utiliser plusieurs fois le même jeu de paramètres pour différents calculs. Le fichier de commande est un fichier texte éditable avec un éditeur quelconque. Il est auto-commenté, c'est-à-dire qu'il inclut des explications de fonctionnement, et les paramètres. On peut donc l'éditer, et modifier directement un paramètre. Ce mode d'intervention peut également être intéressant lorsque l'on veut changer un petit nombre de paramètre et éviter l'ensemble des questions/réponses du menu interactif.

Exemple de fichier de commande :

```
#####
# Fichier de commande pour la visualisation elements finis #
# Herezh++ V4.80 #
# Copyright (c) 1997-2003, Gérard Rio (gerard.rio@univ-ubs.fr) http://www-lg2m.univ-ubs.fr #
# http://www-lg2m.univ-ubs.fr #
#####
```

```
debut_fichier_commande_visu # >>>>> le mot cle: <debut_fichier_commande_visu>
# permet au programme de se positionner au debut du fichier
# il est indispensable
```

```
# =====
# || ***** demande d'une visualisation geomview: ***** ||
# =====
# un mot cle de debut (debut_visualisation_geomview)
# un mot cle de fin ( fin_visualisation_geomview) apres tous les ordres particuliers
# la seule presence du premier mots cle suffit a activer la visualisation geomview
```

```
# ----- definition des parametres du maillage initial: -----
debut_maillage_initial # un mot cle de debut de liste
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
# les parametres avec des valeurs: (sur une meme ligne)
en_filaire 1 # en_filaire <0 ou 1> (indique si l'on veut le dessin en filaire)
surface 1 # surface <0 ou 1> (indique si l'on veut le dessin des faces ou surfaces)
les_numeros 0 # numero <0 ou 1> (indique si l'on veut le dessin des numeros)
# couleurs_filaires_RGB <3 réels> (indique la couleur en RGB du trace filaire)
couleurs_filaires_RGB 0 0 1
# couleurs_surfaces_RGB <3 réels> (indique la couleur en RGB du trace des surfaces)
couleurs_surfaces_RGB 0 1 0
# couleurs_numeros_RGB <3 réels> (indique la couleur en RGB du trace des numeros)
couleurs_numeros_RGB 1 0 0
fin_maillage_initial # le mot cle de fin
```

```
# ----- definition des parametres pour les isovaleurs : -----
debut_isovaleur_vrml # mot cle de debut des parametres pour les isovaleurs
actif 0 # <0 ou 1> indique si l'ordre est actif ou non
choix_peau 1 # <0 ou 1> si 1 visualisation uniquement de la peau
choix_legende 1 # <0 ou 1> si 1 affichage de la legende
choix_min_max 1 # <0 ou 1> si 1 min max calcule automatiquement
valMini -7.47864e+240 # mini <un reel> utilises si choix_min_max = 0
valMaxi -7.47864e+240 # maxi <un reel> utilises si choix_min_max = 0
nb_iso_val 5 # <un entier> nombre d'isovaleurs dans la legende
# tableau de ddl a visualiser, un par maillage
debut_tableau_ddl fin_tableau_ddl
fin_isovaleur_vrml # mot cle de fin des parametres pour les isovaleurs
```

```
# ----- definition des parametres de deformee: -----
debut_deformee # un mot cle de debut de liste
actif 0 # <0 ou 1> indique si l'ordre est actif ou non
# les parametres avec des valeurs: (sur une meme ligne)
en_filaire 1 # en_filaire <0 ou 1> (indique si l'on veut le dessin en filaire)
```

```

surface 1 # surface <0 ou 1> (indique si l'on veut le dessin des faces ou surfaces)
les_numeros 0 # numero <0 ou 1> (indique si l'on veut le dessin des numeros)
# couleurs_filaires_RGB <3 réels> (indique la couleur en RGB du trace filaire)
couleurs_filaires_RGB 0 1 1
# couleurs_surfaces_RGB <3 réels> (indique la couleur en RGB du trace des surfaces)
couleurs_surfaces_RGB 1 1 0
# couleurs_numeros_RGB <3 réels> (indique la couleur en RGB du trace des numeros)
couleurs_numeros_RGB 1 0 0
amplification 1 # amplification <1 réel> (indique le facteur d'amplification
                #de la deformee par rapport au calcul)
fin_deformee # un mot cle de fin

# ----- definition de la liste des increments a balayer: -----
debut_list_increment # un mot cle de debut de liste
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
# une liste d'entier separee par des blancs,
# un mot cle de fin de liste ( fin_list_increment)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 fin_list_increment

# ----- choix des maillages a visualiser: -----
# la liste est facultative, par default la visualisation concerne le premier maillage
debut_choix_maillage # un mot cle de debut,
actif 0 # <0 ou 1> indique si l'ordre est actif ou non
# une liste d'entiers , puis <fin_choix_maillage>, sur une meme ligne
1 fin_choix_maillage

fin_visualisation_geomview
# =====
# || fin de la visualisation geomview ||
# =====

fin_fichier_commande_visu # <<<<<< le mot cle <fin_fichier_commande_visu> permet
# l'arret de la lecture des commandes, apres ce mot cle,
# aucune commande n'est lu, de plus
# sans le mot cle de fin de fichier, le fichier n'est pas valide

#####

```

Il est possible de créer un fichier de commande entièrement à l'éditeur, mais dans ce cas il est évidemment difficile d'éviter une faute de syntaxe, ce n'est donc pas la bonne méthode. En fait, la méthode naturelle est lors d'une première exécution du programme, de passer par le menu interactif et de créer une sortie graphique. Ensuite, deux possibilités, soit vous avez prévu dans le fichier .info une sauvegarde des paramètres avec le mot clé "sauveCommandesVisu" cf.(218), soit vous utilisez l'option de sauvegarde interactive à la sortie du programme de visualisation.

On remarque que d'une manière analogue au cas du fichier .info, les commentaires sont précédés du signe # . Le mode de construction du fichier est simple, chaque jeu

de paramètre est encadré par deux mots clés du type "debut_ ..." et "fin_ ...", de plus certains paramètres sont précédés d'un mot clé indiquant leur fonction, ceci pour faciliter l'intervention directe à l'éditeur, dans le fichier.

Le fichier de commande prend comme nom le même que celui du .info, suivi de .CVvisu. Il y a donc un fichier de commande différent par fichier de calcul.

Lors d'un calcul ultérieur si l'on veut utiliser le fichier de commande, il faut utiliser le mot clé "lectureCommandesVisu" cf.(218). Ce mot clé peut également être associé au mot clé "sauveCommandesVisu" cf.(218), dans ce cas après la visualisation automatique, le programme revient en mode interactif, mais ce n'est pas le cas par défaut. Enfin, il est possible de faire cohabiter plusieurs formats de sortie dans un même fichier de commande, c'est-à-dire qu'il est par exemple possible de sortir en format maple et geomview, dans ce cas les paramètres des deux formats seront successivement décrits dans le fichier de commande. Pour créer un tel fichier de commande, il faut, en mode interactif, activer successivement une sortie maple et une sortie geomview.

Les paramètres graphiques pour tous les formats sont susceptibles d'être sauvegardés. On trouvera ainsi dans la suite du document, un exemple pour chaque autre format.

```
#####
# Fichier de commande pour la visualisation elements finis #
# Herezh++ V4.80 #
# Copyright (c) 1997-2003, Gérard Rio (gerard.rio@univ-ubs.fr) http://www-lg2m.univ-ubs.fr #
# http://www-lg2m.univ-ubs.fr #
#####
```

```
debut_fichier_commande_visu # >>>> le mot cle: <debut_fichier_commande_visu>
# permet au programme de se positionner au debut du fichier,
#il est indispensable
```

```
# =====
# || **** demande d'une visualisation vrml: **** ||
# =====
# un mot cle de debut (debut_visualisation_vrml)
# un mot cle de fin ( fin_visualisation_vrml) apres tous les ordres particuliers
# la seule presence du premier mots cle suffit a activer la visualisation vrml
# la presence du second permet une meilleur lisibilite du fichier, mais n'est pas indispensable
debut_visualisation_vrml

# ----- definition des parametres du maillage initial: -----
debut_maillage_initial # un mot cle de debut de liste
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
# les parametres avec des valeurs: (sur une meme ligne)
en_filaire 1 # en_filaire <0 ou 1> (indique si l'on veut le dessin en filaire)
surface 1 # surface <0 ou 1> (indique si l'on veut le dessin des faces ou surfaces)
les_numeros 0 # numero <0 ou 1> (indique si l'on veut le dessin des numeros)
# couleurs_filaires_RGB <3 réels> (indique la couleur en RGB du trace filaire)
couleurs_filaires_RGB 0 0 1
# couleurs_surfaces_RGB <3 réels> (indique la couleur en RGB du trace des surfaces)
couleurs_surfaces_RGB 0 1 0
```

```

# couleurs_numeros_RGB <3 réels> (indique la couleur en RGB du trace des numeros)
couleurs_numeros_RGB 1 0 0
fin_maillage_initial # le mot cle de fin

# ----- definition des parametres pour les isovaleurs : -----
debut_isovaleur_vrml # mot cle de debut des parametres pour les isovaleurs
actif 0 # <0 ou 1> indique si l'ordre est actif ou non
choix_peau 1 # <0 ou 1> si 1 visualisation uniquement de la peau
choix_legende 1 # <0 ou 1> si 1 affichage de la legende
choix_min_max 1 # <0 ou 1> si 1 min max calcule automatiquement
valMini -7.47864e+240 # mini <un reel> utilises si choix_min_max = 0
valMaxi -7.47864e+240 # maxi <un reel> utilises si choix_min_max = 0
nb_iso_val 5 # <un entier> nombre d'isovaleurs dans la legende
# tableau de ddl a visualiser, un par maillage
debut_tableau_ddl fin_tableau_ddl
fin_isovaleur_vrml # mot cle de fin des parametres pour les isovaleurs

# ----- definition des parametres de deformee: -----
debut_deformee # un mot cle de debut de liste
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
# les parametres avec des valeurs: (sur une meme ligne)
en_filaire 1 # en_filaire <0 ou 1> (indique si l'on veut le dessin en filaire)
surface 1 # surface <0 ou 1> (indique si l'on veut le dessin des faces ou surfaces)
les_numeros 0 # numero <0 ou 1> (indique si l'on veut le dessin des numeros)
# couleurs_filaires_RGB <3 réels> (indique la couleur en RGB du trace filaire)
couleurs_filaires_RGB 0 1 1
# couleurs_surfaces_RGB <3 réels> (indique la couleur en RGB du trace des surfaces)
couleurs_surfaces_RGB 1 1 0
# couleurs_numeros_RGB <3 réels> (indique la couleur en RGB du trace des numeros)
couleurs_numeros_RGB 1 0 0
amplification 1 # amplification <1 réel> (indique le facteur d'amplification de la deformee
# par rapport au calcul)
fin_deformee # un mot cle de fin

# ----- definition de la liste des increments a balayer: -----
debut_list_increment # un mot cle de debut de liste
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
# une liste d'entier separee par des blancs,
# un mot cle de fin de liste ( fin_list_increment)
10 fin_list_increment

# ----- choix des maillages a visualiser: -----
# la liste est facultative, par default la visualisation concerne le premier maillage
debut_choix_maillage # un mot cle de debut,
actif 0 # <0 ou 1> indique si l'ordre est actif ou non
# une liste d'entiers , puis <fin_choix_maillage>, sur une meme ligne
1 1 fin_choix_maillage

# ----- definition des parametres d'animation: -----
debut_animation # un mot cle de debut de liste (debut_animation)
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
# des parametres avec des valeurs: (sur une meme ligne)
cycleInterval 8 # cycleInterval <un reel> (indique le temps en seconde du cycle de
# l'animation)

```

```

loop 0 # loop <0 ou 1> (indique si l'on veut une animation en boucle continue ou non)
startTime 1 # startTime <un reel> (temps de démarrage de l'animation en absolu depuis
# 1970)
stopTime 0 # stopTime <un réel> (temps de fin en absolu depuis 1970, si < a starttime
# on n'en tiend pas compte)
debut_auto 0 # debut_auto <0 ou 1> (indique si l'on veut ou non un debut automatique
# de l'animation)
inter_2pas 2 # inter_2pas <un réel> (temps entre deux cycle d'animation, si loop est
# actif)
fin_animation # un mot cle de fin

fin_visualisation_vrml
# =====
# || fin de la visualisation vrml ||
# =====

# =====
# || ***** demande d'une visualisation maple: ***** ||
# =====
# un mot cle de debut (debut_visualisation_maple)
# un mot cle de fin ( fin_visualisation_maple)
# la seule presence de ces deux mots cle suffit a activer la visualisation maple
debut_visualisation_maple

# ----- definition de la liste des increments a balayer: -----
debut_list_increment # un mot cle de debut de liste
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
# une liste d'entier separee par des blancs,
# un mot cle de fin de liste ( fin_list_increment)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 fin_list_increment

# ----- choix des maillages a visualiser: -----
# la liste est facultative, par default la visualisation concerne le premier maillage
debut_choix_maillage # un mot cle de debut,
actif 0 # <0 ou 1> indique si l'ordre est actif ou non
# une liste d'entiers , puis <fin_choix_maillage>, sur une meme ligne
1 fin_choix_maillage

# ----- definition des grandeurs a visualiser (maple): -----
debut_grandeurs_maple # un mot cle de debut (debut_grandeurs_maple),
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
# ensuite pour chaque maillage:,
# le numero du maillage <un entier>,
# les infos pour la visualisation eventuelle aux noeud,
# les infos pour la visualisation eventuelle aux elements,
# enfin un mot cle de fin ( fin_grandeurs_maple)
1 # le numero de maillage
debut_liste_ddl_et_noeud # ** debut des grandeurs aux noeuds
# debut de la liste de noeuds, puis une liste de numero de noeud <entier>,
# puis <fin_list_noeud>
deb_list_noeud 2 fin_list_noeud
# debut de la liste des ddl a considerer aux noeuds, (une liste de ddl),
# puis <fin_list_ddl_noeud>

```

```

deb_list_ddl_noeud      X1 V1 GAMMA1      fin_list_ddl_noeud
fin_liste_ddl_et_noeud      # fin des grandeurs aux noeuds
debut_liste_ddl_ptinteg # ** debut des grandeurs aux elements
# debut de la liste des elements et points d'integration, une liste de
# (un element, un numero de pt d'integ), puis <fin_list_NbElement_NbPtInteg>
deb_list_NbElement_NbPtInteg      fin_list_NbElement_NbPtInteg
# debut de la liste des ddl a considerer pour les elements, (une liste de ddl),
# puis <fin_list_ddl_element>
deb_list_ddl_element      fin_list_ddl_element # fin de la liste de ddl a
# considerer pour les elements
fin_liste_ddl_ptinteg      # fin des grandeurs aux elements
fin_grandeurs_maple # fin des grandeurs a visualiser au format maple

fin_visualisation_maple
# =====
# ||                      fin de la  visualisation maple                      ||
# =====

# =====
# ||      *****      demande d'une visualisation Gid:      *****      ||
# =====
# un mot cle de debut (debut_visualisation_Gid)
# un mot cle de fin ( fin_visualisation_Gid) apres tous les ordres particuliers
# la seule presence du premier mots cle suffit a activer la visualisation Gid
# la presence du second permet une meilleur lisibilite du fichier, mais n'est pas indispensable
debut_visualisation_Gid

# ----- definition des parametres du maillage initial: -----
debut_maillage_initial # un mot cle de debut de liste
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
fin_maillage_initial # le mot cle de fin

# ----- definition des parametres pour les isovaleurs : -----
debut_isovaleur_Gid      # mot cle de debut des parametres pour les isovaleurs
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
1 # le numero de maillage
# tableau de ddl aux noeuds a visualiser, un par maillage
debut_tableau_ddl_aux_noeuds X1 183 X2 183 X3 183 fin_tableau_ddl_aux_noeuds
# tableau de ddl aux elements a visualiser, un par maillage
debut_tableau_ddl_aux_elements contrainte_tresca fin_tableau_ddl_aux_elements
# tableau de grandeurs particulieres aux elements a visualiser, un par maillage
deb_list_GrandParticuliere_element      fin_list_GrandParticuliere_element
fin_isovaleur_Gid      # mot cle de fin des parametres pour les isovaleurs

# ----- definition des parametres de deformee: -----
debut_deformee # un mot cle de debut de liste
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
# definition des alertes: deb_list_alerte
# un mot clef de debut
# puis deux nombres: un mini et un maxi, et un nom
# un mot clef de fin: fin_list_alerte
deb_list_alerte
fin_list_alerte

```

```

fin_deformee # un mot cle de fin

# ----- definition de la liste des increments a balayer: -----
debut_list_increment # un mot cle de debut de liste
actif 1 # <0 ou 1> indique si l'ordre est actif ou non
# une liste d'entier separee par des blancs, ou le mot cle (tous_les_increments)
# un mot cle de fin de liste ( fin_list_increment)
dernier_increment fin_list_increment

# ----- choix des maillages a visualiser: -----
# la liste est facultative, par default la visualisation concerne le premier maillage
debut_choix_maillage # un mot cle de debut,
actif 0 # <0 ou 1> indique si l'ordre est actif ou non
# une liste d'entiers , puis <fin_choix_maillage>, sur une meme ligne
1 fin_choix_maillage

fin_visualisation_Gid
# =====
# || fin de la visualisation Gid ||
# =====

fin_fichier_commande_visu # <<<<<< le mot cle <fin_fichier_commande_visu> permet
# l'arret de la lecture des commandes, apres ce mot cle,
# aucune commande n'est lu, de plus
# sans le mot cle de fin de fichier, le fichier
# n'est pas valide

```

#####

47.3.2 Formats vrml

On obtient le menu suivant :

```

===== module de visualisation format vrml =====
-----
                                maillage initiale:      mi
                                isovaleurs:             iso
                                deformee:              de
                                choix numeros d'increment: cni
                                choix du ou des maillages a visualiser: cmv
                                animation:             ani
                                visualisation :         visu
                                arret de la visualisation interactive: fin
reponse ?

```

L'objectif est ici de produire un fichier texte qui a toujours le même nom : HZ.wrl, qui contient les données pour une visualisation graphique. Le programme Herezh++ n'effectue pas lui-même la visualisation, il se sert d'outils déjà existants.

On va étudier les différents éléments du menu.

– mi : on obtient le menu :

```
----> preparation de la visualisation du maillage initiale
parametre par default ? : affichage en filaire, couleur bleu,
                           et affichage des facettes, couleur vert,
                           pas d'affichage de numeros.    (rep 'o')
pour accepter ces parametres sinon autre
reponse ?
```

On voit qu'il s'agit de paramétrer la visualisation du maillage initial. Si l'on décide de ne pas prendre le choix par défaut, il est nécessaire d'indiquer systématiquement, toutes les informations : affichage filaire ou facettisé, couleur du filaire et des facettes, numéro. Dans ce dernier cas, on peut également définir la taille des numéros. Cette taille dépend du visualisateur, il est donc souvent nécessaire de trouver la bonne taille à partir d'essais successifs.

– de : on obtient le menu :

```
----> preparation de la visualisation des deformeés
parametre par default ? : affichage en filaire, couleur cyan,
                           affichage des facettes, couleur jaune,
                           pas d'affichage de numéros,
                           amplification du déplacement = 1.    (rep 'o')
pour accepter ces parametres sinon autre
reponse ?
```

L'objectif est ici de paramétrer la déformée : quel type de visualisation d'une manière analogue au maillage initial, et quel facteur d'amplification.

– cni : ce qui conduit au premier menu :

```
liste des increments disponibles (en plus de l'increment 0 mis par default): 1
parametre par default ? le dernier increment    (rep 'o') pour accepter sinon autre
```

L'objectif est de définir une liste d'incrément à visualiser. Dans le cas où un seul incrément est défini, il s'agit d'une déformée classique. Dans le cas où plusieurs incréments sont définis, la visualisation s'effectuera par une animation, qui consistera à visualiser chronologiquement les différentes déformées. En général, il est dans ce cas préférable d'effacer la liste actuelle, constituée par défaut et ensuite de définir le ou les incréments à visualiser. A priori, le programme propose la liste des incréments disponibles. Ceux-ci sont définis par le paramètre SAUVEGARDE dans la liste des paramètres globaux (cf.210). Par défaut tous les incréments de calcul sont sauvegardés, cependant pour certains calculs longs ou nécessitant de nombreux pas de temps il est possible de ne sauvegarder que certains pas de temps. Dans ce dernier cas, au moment de la visualisation, seuls ces pas de temps sauvegardés seront évidemment disponibles. On obtient donc le menu suivant :

```
differentes options :
un increment                    rep : 1
un interval d'increments par pas de 1    rep : 2
un interval d'increments avec choix du pas
```

```
entre deux increment choisit          rep : 3
une liste d'increments                  rep : 4
```

Le premier cas permet de définir un incrément. Le cas 2, permet la définition d'un ensemble d'incrément, défini par le premier et le dernier. Le cas 3, permet en plus de définir le pas. Il s'agit du pas pour les incréments disponibles. Supposons par exemple que l'on a sauvegardé tous les 10 pas de temps, et que l'on choisit le cas 3 avec un intervalle de 2, cela signifie que l'on considérera la déformée tous les 20 pas de temps de calcul. Le cas 4, permet de définir une liste manuellement, à l'éditeur.

- cmv : le menu obtenu est :

```
liste des maillage disponibles : de 1 a 1
parametre par default ? tous les maillages (rep 'o') pour accepter sinon autre
voulez-vous effacer la liste actuelle qui est : 1
rep 'o' pour effacer sinon autre ?
```

Il s'agit de définir la liste des maillages que l'on veut représenter. Par défaut on retient le maillage 1.

- ani : paramètre d'animation :

```
----> changement des parametres de l'animation
parametres actuels ? : duree de l'animation : 8
                        bouclage de l'animation : non (rep 'o')
pour accepter ces parametres sinon autre
reponse ?
```

Le menu est explicite.

- visu : correspond à l'ordre d'activation de l'écriture du fichier HZ.wrl
- fin : correspond à la fin du menu de visualisation vrml.

Il n'est pas nécessaire de répondre à tous les éléments du menu, un certain nombre de valeurs sont définies par défaut. Il est possible de sortir plusieurs visualisations, avec des paramètres différents, cependant à chaque fois que l'on active l'ordre visu, le fichier HZ.wrl est écrasé. Il faut donc le sauvegarder si on veut en garder une trace.

Dans le cas où l'on désire une animation, il est nécessaire d'effectuer successivement les opérations suivantes : définition du maillage initial (mi), définition de la déformée (de), définition des différents incréments en gardant le maillage initiale (cni), puis visualisation avec auparavant définition automatique ou non de l'animation (ani) et (visu).

47.3.3 Formats Maple : introduction

En fait, il s'agit ici de sortir un fichier texte en vue de tracer des courbes d'évolution. Ces courbes peuvent ensuite être tracées par n'importe quel utilitaire de tracé, pourvu que cet utilitaire sache isoler les différentes colonnes du fichier de sortie. Le format de sortie était initialement prévu pour une visualisation à l'aide de l'utilitaire Maple, d'où le nom du paragraphe.

Au niveau du fichier de sortie, les commentaires sont précédés du signe # . Ces commentaires contiennent cependant des informations utiles à la structuration des données du fichier, de manière à pouvoir utiliser ce fichier de manière automatique via un script par exemple.

L'exemple d'utilisation est la sortie de l'évolution d'une grandeur à un noeud en fonction du temps : position, vitesse, accélération. Il est également possible de sortir l'évolution

d'une grandeur à un point d'intégration. Il s'agit alors de la contrainte ou de la déformation ou de grandeurs dérivées comme la contrainte de Mises ou les valeurs principales ... Le premier menu obtenu est le suivant :

```
=== choix des increments utilises pour l'initialisation de la visualisation ====
option par default : tous les increments                (rep 1)
choix d'un nombre plus petit d'increment              (rep 2)
reponse ?
```

Le programme doit lire un certain nombre de pas de calcul, pour connaître les différentes grandeurs à visualiser. Le plus sûr serait que tous les pas soient lus, cependant dans le cas où le nombre de pas sauvegardé est grand et que le nombre de grandeurs qui nous intéressent reste identique pendant tout le calcul, il est suffisant d'initialiser avec seulement un incrément, c'est donc dans ce cas la réponse 2, d'où le menu suivant :

```
liste des increments disponibles (en plus de l'increment 0 mis par default): 0 1
parametre par default ? le dernier increment    (rep 'o') pour accepter sinon autre

auquel on choisira a priori la réponse par default.
On obtient alors le menu suivant :
```

```
liste finale : 0 34
```

```
.....choix numeros d'increment:          cni
.....choix du ou des maillages a visualiser:      cmv
.....choix grandeurs:                          cg
.....animation_maple:                            ani
.....visualisation:                               visu
.arret visualisation interactive pour format maple:    f
reponse ?
```

On reconnait certains choix identiques au cas de la sortie vrml, il s'agit de : de, cni, cmv, visu. Le cas de cg est particulier, il permet de définir exactement quelle grandeur doit être visualisée. Le menu obtenu est alors :

```
grandeurs globales                -> rep : glo
torseurs de reactions              -> rep : tre
grandeurs aux noeuds (par default) -> rep : noe
grandeurs generique aux elements  -> rep : ele
grandeurs particulieres aux elements -> rep : elp
style de sortie                    -> rep : sty
pour accepter la valeur par default -> rep : o
pour arreter les questions         -> rep : fin (ou f)
reponse ?
```


47.3.4 Cas des grandeurs aux noeuds pour le format maple

La valeur par défaut est la sortie au noeud, ce qui conduit par exemple à (ici la réponse dépend du problème traité) :

```
maillage nb : 1
la liste des grandeurs disponibles est la suivante : X1 X2 X3 R_X1 R_X2 R_X3
donnez la ou les grandeurs que vous voulez visualiser (rep grandeurs?)
ou toutes les grandeurs sans les reactions (rep : to)
ou toutes les grandeurs (rep : tr)
effacer la liste actuelle (rep : ef)
type de sortie de ddl retenue (rep : ts)
                                     (pour terminer tapez : fin (ou f))
grandeur ?
```

Par exemple on répond to puis fin, c'est-à-dire toutes les grandeurs. Il reste à déterminer le ou les numéros de noeud :

```
choix du ou des noeuds ou l'on veut la visualisation
le reperage d'un noeuds peut se faire de differentes manieres :
par son numero dans le maillage -> (choix : 1)
par ses coordonnees, meme approximatives -> (choix : 2)
par une reference de liste de noeuds -> (choix : 3)
donnez pour chaque noeud le type de choix puis le numero
ou les cordonnees
effacer la liste actuelle de noeuds (rep : ef)
effacer la liste actuelle des references de noeuds (rep : efref)
afficher la liste des references de noeuds existants (rep : affref)
                                     (pour finir tapez : fin (ou f))
choix ?
```

on choisit la définition qui est la plus pratique, soit par numéro de noeud soit par coordonnées. On peu définir plusieurs numéros de noeud, dans ce cas les informations seront stockées dans le fichier de sortie, de manière contigüe.

Ensuite après activation de l'ordre visu, on obtient un fichier HZ_maple dont un exemple est donné par la table (219)

TABLE 219 – Exemple d'en-tête de fichier .maple

```
#fichier au format maple6
#####
# Visualisation elements finis : Herezh+ V6.565 #
# Copyright (c) 1997-2011, Gerard Rio (gerard.rio@univ-ubs.fr) http://www-lg2m.univ-ubs.fr #
# http://www-lg2m.univ-ubs.fr #
#####

# entete des donnees : informations generales: on trouve successivement:
# >> le nombre de grandeurs globales (peut etre nul) suivi des identificateurs
# precedes du numero de colonne entre crochet
# >> le nombre de maillages m, et dimension de l'espace de travail
# puis pour chaque maillage,
# >> le nombre de torseurs de reaction (peut etre nul), le nombre total de reel qui va etre ecrit
# correspondant aux composantes des torseurs, puis les noms de ref associee suivi des positions
# des composantes entre crochet accolées a un identificateur: R pour reaction, M pour moment
# puis pour chaque maillage
# >> le nombre de noeud n (peut etre nul) ou il y a des grandeurs en sortie ,
# puis le nombre des grandeurs p1 correspondantes, la position entre crochet des coordonnees
# et enfin l'identificateur de ces grandeurs(p1 chaines de caractere)
# precedes du numero de colonne correspondant entre crochet
# puis pour chaque maillage
# >> le nombre de couples element-pt_integ (peut etre nulle) ou il y a des grandeurs en sortie ,
# les grandeurs aux elements sont decomposees en 2 listes: la premiere de quantite P2 correspondant
# a des grandeurs generiques, la seconde de quantite P3 correspond aux grandeurs specifiques,
# on trouve donc a la suite du nombre d'element: le nombre P2, suivi de P2 identificateurs de ddl
# chacun precedes du numero de colonne entre crochet
# puis le nombre P3, suivi de P3 identificateurs+categorie+type (chaines de caracteres),
# suivi entre crochet, de la plage des numeros de colonnes, correspondant
# chacun sur une ligne differentes
# == NB: pour les grandeurs specifique tensorielle: exemple d'ordre en 2D:
# tenseur symetrique, A(1,1) A(2,1) A(2,2), non symetrique A(1,1) A(1,2) A(2,1) A(2,2)
# en 3D c'est: tenseur symetrique, A(1,1) A(2,1) A(2,2) A(3,1) A(3,2) A(3,3)
# non symetrique A(1,1) A(1,2) A(2,1) A(2,2) A(2,3) A(3,1) A(3,2) A(3,3)
# ** dans le cas ou il n'y a qu'un seul increment en sortie, pour les grandeurs aux noeuds
# et aux elements,
# ** les informations peuvent etre decoupees selon: une ligne = un noeud, et le temps
# n'est pas indique
# ** ( cf: parametre_style_de_sortie = 0)

#####
#|| recapitulatif des differentes grandeurs par colonne ||
#####
#----- grandeur globales -----
#4 (nombre de grandeurs globales) [2]energie_elastique [3]energie_externes
# [4]energie_elastique [5]energie_houress
#----- maillage et dimension -----
#1 3 (nombre de maillages et dimension)
#----- torseurs de reactions -----
#1 6 (nombre de torseurs et nombre total de grandeurs associees)
# N_bas_arriere_droit [7]Rx [8]Ry [9]Rz [10]Mx [11]My [12]Mz ;
#
#----- grandeurs aux noeuds -----
#0 0 (nombre de noeuds, nombre total de grandeurs associees)
#----- grandeurs aux elements -----
#0 0 (nombre total d'elements, nombre totale de grandeurs associees)
#####
#|| fin du recapitulatif des differentes grandeurs ||
#####

# ensuite les donnees sont organisees sur differentes lignes, chaque lignes correspondant
# a un calcul (par exemple un pas de temps), sur chaque ligne il y a m enregistrement, chacun
# correspondant a un maillage. On trouve pour chaque enregistrement successivement :
# s'il y a des grandeurs globales: le temps puis les grandeurs globales,
# puis s'il y a des torseurs de reaction :
# de nouveau le temps, les composantes de la resultante puis les composantes du moments
# donc en 1D -> 1 reels (resultante), en 2D -> 3 reels (resultante 2, moment 1) et en 3D 6 reels
# puis s'il y a des grandeurs aux noeuds: de nouveau le temps
# les coordonnees a t du premier noeud suivi des p1 grandeurs correspondant au premier noeud
# puis les coordonnees du second noeud, les p1 grandeurs etc. pour tous les noeuds
# puis s'il y a des grandeur aux elements:
# le temps, puis les coordonnees a t du point d'integration d'un element (pour les grandeurs generiques)
# suivi des p2 grandeurs correspondantes puis les coordonnees a t du point d'integration
# correspondant aux grandeurs specifiques suivi des p3 grandeurs correspondantes
# puis les coordonnees d'un second point d'integration d'un element, les p2 grandeurs
# etc. pour tous les points d'integration - element

1.000000000000e-02 2.157281941235e-04 2.157330256490e-04 ....
2.000000000000e-02 8.629081223132e-04 8.629541106361e-04 ....
3.000000000000e-02 1.941488793077e-03 ....
etc...
```

Toutes les lignes qui commencent par # correspondent à des commentaires explicatifs.

Il est possible de sortir non pas les grandeurs, mais leurs accroissements entre 0 et t, ou encore les valeurs à t=0 suivit de celle à t, ce qui permet en suite un post-traitement plus aisé. Pour cela, dans le premier menu, on l'utilise l'option : "ts" (qui correspond à la question : type de sortie de ddl retenue) et on obtient alors le sous-menu suivant, dans lequel on a choisi par exemple le cas 2 :

```
grandeur ? ts

actuellement type_sortie_ddl_retenue= 1

sortie des grandeurs a t (par default) : (rep: 0)
sortie des grandeurs entre t et 0      : (rep: 1)
sortie des grandeurs a 0 et a t       : (rep: 2)
laisser tel-quel                       : (rep: f)
2
```

47.3.5 Cas des grandeurs de type degré de liberté, aux éléments pour le format maple

Après avoir choisi la réponse "le" on obtient une réponse semblable au texte suivant :

```
maillage nb : 1
la liste des grandeurs disponibles est la suivante : SIG11 EPS11 Green-Lagrange11
Almansi11 Cauchy_local11 Almansi_local11 Def_principaleI Sigma_principaleI
contrainte_mises contrainte_tresca
donnez la ou les grandeurs que vous voulez visualiser      (rep grandeurs?)
REMARQUE : il faut donner uniquement un jeu de ddl
de meme type (contraintes ou(exclusif) erreur ou ...)
effacer la liste actuelle                                  (rep : ef)
                                                           (pour terminer tapez : fin)

grandeur ?
```

Le programme indique la liste des grandeurs actuellement disponibles. Cette liste dépend du type d'élément utilisé de la dimension de l'espace de travail, etc. Il est possible de définir une liste de plusieurs grandeurs à visualiser. Par contre, toutes ces grandeurs doivent être exprimées au même point, en général un point d'intégration. De plus, il n'est pas possible de sortir des grandeurs de type de base différent. Par exemple : pour une déformation et une contrainte, il faut faire deux sorties. Il est possible également de supprimer des grandeurs dans une précédente liste. Ensuite après le mot clé "fin" on obtient le menu suivant :

```
grandeur ? SIG11
grandeur ? fin

choix de la position sur le ou les elements ou l'on veut la visualisation
le reperage peut se faire de differentes manieres :
par un numero d'element dans le maillage + un
numero d'un pt d'integration (au sens classique)          -> (choix : 1)
```

```

par des coordonnees d'un point, meme approximatives
la grandeur affichee sera a un point le plus proche ou elle existe -> (choix : 2)
par une reference de liste d'element -> (choix : 3)
donnez le type de choix puis les infos correspondantes
effacer la liste actuelle -> (rep : ef)
effacer la liste actuelle des references d'elements -> (rep : efref)
afficher la liste des references d'elements existants -> (rep : affref)
(tapez fin pour finir(ou f))

choix ?

```

Les points où l'on veut les grandeurs sont en général les points d'intégrations. La première méthode la plus simple est d'entrée le numéro du point d'intégration et le numéro de l'élément.

Exemple de séquence :

```

choix ? 1
numero d'element ? 2
numero du point d'integration ? 1

choix ?

```

Cependant, dans un maillage complexe il est difficile de connaître les coordonnées exactes des points d'intégration. Dans ce cas, on choisit l'option 2 et on donne un point approximativement le plus proche du point d'intégration que l'on recherche, le programme recherche le point et le propose. On peut soit accepter soit l'effacer.

Exemple de séquence :

```

choix ? 2

coordonnee (1 nombre(s)) ? 2

les coordonnees sont à: t = 0 ou t ou tdt (répondre 0 ou t ou tdt) ? 0

element contenant le point nb= 1 et le numéro d'ordre le plus proche 1 de coordonnee à
t: 14.2868
choix ?

```

47.3.6 Cas des grandeurs de type spécifique, aux éléments pour le format maple

Contrairement au paragraphe précédent qui concernait les grandeurs de type ddl (contrainte, déformation ...), il s'agit ici de grandeurs particulières. Par exemple, supposons que l'on utilise une loi de comportement constituée de la somme de n lois élémentaires. Il est possible d'accéder aux contraintes individuelles à chaque lois, c'est-à-dire à la contribution de chaque loi. Il est clair que ces grandeurs ne sont pas disponibles pour toutes les lois, c'est pourquoi elles sont qualifiées de "grandeurs particulières".

Après avoir choisi la réponse "elp" on obtient une réponse semblable au texte suivant :

```

maillage nb : 1
la liste des grandeurs particulieres disponibles est la suivante :
contrainte_individuelle_a_chaque_loi_courant_et_a_t

```

```

contrainte_individuelle_a_chaque_loi_a_t
grandeurs que vous voulez visualiser          ( rep : grandeurs?)
REMARQUE : il faut donner uniquement un jeux de grandeurs
definit au meme point d'integration
effacer la liste actuelle                      (rep : ef)
                                             (pour terminer tapez : fin (ou f))
grandeur ?

```

Le programme indique la liste des grandeurs particulières actuellement disponibles. Cette liste dépend du type de loi de comportement, du type de déformation, du type de loi thermophysique...

Il est possible de définir une liste de plusieurs grandeurs à visualiser. Par contre, toutes ces grandeurs doivent être exprimées au même point, en général un point d'intégration. Il est possible également de supprimer des grandeurs dans une précédente liste. Ensuite après le mot clé "fin" on accède au menu du choix d'éléments, et de points d'intégration (le même que pour les degrés de liberté aux éléments) :

```

liste actuelle des element:

choix de la position sur le ou les elements ou l'on veut la visualisation
le reperage peut se faire de differentes manieres :
par un numero d'element dans le maillage + un
numero d'un pt d'integration (au sens classique)          -> (choix : 1)
par des coordonnees d'un point, meme approximatives
la grandeur affichee sera a un point le plus proche ou elle existe -> (choix : 2)
par une reference de liste d'element                      ->          (choix : 3)
donnez le type de choix puis les infos correspondantes
effacer la liste actuelle                                -> (rep : ef)
effacer la liste actuelle des references d'elements     -> (rep : ehref)
afficher la liste des references d'elements existants    -> (rep : affref)
                                                         (tapez fin pour finir(ou f))
choix ?

```

On suit ensuite la même démarche que celle vue au paragraphe (47.3.5);

47.3.7 Cas des grandeurs globales

Cf. les menus proposés : les grandeurs globales sont par exemple : l'énergie cinétique globale, la quantité de mouvement, l'énergie interne, la puissance d'accélération, les différentes énergies de dissipation et.. Il faut se reporter à l'entête du fichier .maple pour retrouver le placement dans le fichier des différentes grandeurs que l'on souhaite visualiser (voir l'exemple (219)).

47.3.8 Cas des torseurs de réactions

Au moment du calcul des réactions, le programme calcul également un torseur de réaction correspondant à chaque référence de condition limite appliquée à au moins un déplacement donc conduisant à un déplacement imposé. Ensuite il est possible d'obtenir les composantes de ces torseurs. Au moment du choix interactif des différentes grandeurs

que l'on souhaite visualiser, on choisit dans le menu l'option "tre" (cf. 47.3.3) et on peut alors indiquer pour chaque maillage quel torseur on souhaite obtenir.

Remarque : Les réactions (sens et intensités) correspondent aux efforts et moments exercés par la pièce "sur" l'environnement (et non l'inverse!!).

Dans le cas 1D, seul la composante unique de la résultante est indiquée. En 2D il y a 2 composantes pour la résultante et une composante pour le moment. En 3D il y a 6 composantes pour chaque référence.

Dans le cas des conditions linéaires, il y a une réaction induite sur toutes les références concernant la condition linéaire (cf.35). Dans le cas où une référence est utilisée pour une condition bloquée et pour une condition linéaire (comme référence secondaire par exemple), il y a création de deux torseurs un pour la condition bloquée et un pour la condition linéaire, pour ce dernier le nom de la référence est postfixé par "_CLL".

La constitution des différents torseurs suit une logique particulière. Supposons par exemple deux références "N_avant" et "N_avant_droit", tel que pour la première référence on bloque le déplacement UY et pour la seconde référence on bloque UX. Supposons de plus que les noeuds de la seconde référence appartiennent également à la première référence. On s'attendrait à ce que le torseur correspondant à la première référence, intègre les réactions en X générés par la seconde référence : en fait non. Le premier torseur est construit à l'aide "uniquement" des réactions correspondantes aux ddl bloqués relatifs à la première référence. Cela permet de pouvoir retrouver et distinguer les répercussions de chaque blocage imposé.

Cependant, supposons maintenant que l'on définisse sur la première référence une première condition UX, puis par la suite (pas forcément dans la même définition de blocage) un second blocage UZ. En sortie il n'y aura qu'un seul torseur, qui globalisera les deux réactions, car d'une part les deux blocages se rapportent à exactement une même référence, et d'autre part, il est possible de retrouver dans les coordonnées du torseur ce qui est dû à chaque condition.

Le cas des conditions linéaires suit la même logique. Cependant, la prise en compte des CLL ne simplifie pas la présentation des résultats due à la remarque suivante :

- un ddl ne peut être bloqué qu'une seule fois par une condition de blocage. D'une manière pratique, si un ddl est bloqué plusieurs fois il y a un message d'avertissement qui est généré (à condition que le niveau de commentaire soit, suffisent) et seul le dernier blocage est pris en compte (en fait, il écrase les autres blocages!).
- par contre plusieurs CLL peuvent modifier un même degré de liberté (à condition que ce ne soit pas un ddl de la référence principale).

Le choix qui a été pris est de différencier les torseurs issus des ddl bloqués et ceux des CLL. Pour ce faire, dans le cas des torseurs issus des CLL, le nom de la référence associée est postfixé par "_CLL". On peut donc avoir deux torseurs identiques associés à une même référence, le premier dans le cadre d'un ddl bloqué, le second dans le cadre d'une CLL.

47.3.9 Remarques concernant l'ordre de sortie des composantes des grandeurs tensorielles dans le cas du format Maple

Les grandeurs tensorielles sont représentées par la succession des composantes du tenseur, c'est-à-dire la succession des composantes de la matrice représentant les composantes du tenseur. Dans le cas d'un tenseur non symétrique, les grandeurs apparaissent ligne par ligne. Par exemple pour la dimension 2 on a : $A(1, 1) A(1, 2) A(2, 1) A(2, 2)$. Dans le cas d'un tenseur symétrique, seule la partie triangulaire inférieure est représentée, également ligne par ligne. Par exemple pour la dimension 2 on aura : $A(1, 1) A(2, 1) A(2, 2)$ et en 3D on aura : $A(1, 1) A(2, 1) A(2, 2) A(3, 1) A(3, 2) A(3, 3)$.

47.3.10 Formats geomview

On obtient le menu suivant :

```

===== module de visualisation format vrml =====
-----
                                maillage initiale:          mi
                                frontieres initiales:       fri
                                isovaleurs:                 iso
                                deformee:                   de
                                choix numeros d'increment:   cni
                                choix du ou des maillages a visualiser: cmv
                                animation:                   ani
                                visualisation :               visu
                                arret de la visualisation interactive: fin
reponse ?

```

Globalement, la signification et l'utilisation de chaque ordre sont identiques au cas de la sortie vrml (pour l'instant). Notons également que plusieurs ordres présents ne fonctionnent pas encore, ils sont en développement. Il s'agit des isovaleurs, des frontières initiales, et de l'animation.

47.3.11 Formats Gid

Gid est un logiciel de pré et post traitement dédié éléments finis. Il comporte de nombreuses possibilités de visualisation :

- isovaleurs
- création d'animations,
- coupes,
- lignes de flux
- ...

On se référera à la documentation en ligne de Gid pour plus d'information.

La version académique de Gid (version gratuite) permet de traiter des maillages jusqu'à 3000 éléments. Enfin, Gid est un logiciel disponible sur toutes les plates-formes : windows, Linux, Mac osX, Unix. Le logiciel est téléchargeable sur le web.

À la suite de l'utilisation du menu Gid, Herezh++, par exemple pour un fichier de commande "toto.info", crée deux fichiers "toto_Gid.msh" et "toto_Gid.res". Le premier

contient les éléments nécessaires à la définition du maillage pour Gid. Le second contient toutes les indications pour le traitement graphique des résultats demandés.

On obtient le menu suivant :

```
===== module de visualisation format Gid =====

=== choix des increments utilises pour l'initialisation de la visualisation ===
option par default : tous les increments          (rep 1)
choix d'un nombre plus petit d'increment        (rep 2)
reponse ? 1
-----

                maillage initiale:           mi
                   isovaleurs:             iso
                       deformee:           de
                   choix numeros d'increment: cni
choix du ou des maillages a visualiser:        cmv
                       visualisation :     visu
arret de la visualisation interactive:         f
reponse ?
```

Globalement, les significations et utilisations de chaque ordre sont identiques au cas des précédentes sorties. Notons également que l'ordre de frontières initiales ne fonctionne pas actuellement.

47.3.12 Formats gmsh et comparaison avec le format Gid

gmsh est un logiciel de pré et post traitement dédié éléments finis. Il est libre de droits d'utilisation et est particulièrement ergonomique. Il comporte de nombreuses possibilités de visualisation :

- isovaleurs
- création d'animations,
- coupes,
- lignes de flux
- ...

On se référera à la documentation en ligne de gmsh pour plus d'information. Il existe également des tutoriaux sur le site web de distribution du logiciel. Ces tutoriaux permettent de prendre en main le logiciel très rapidement. Gmsh est un logiciel gratuit disponible sur toutes les plates-formes : windows, Linux, Mac osX, Unix. Le logiciel est téléchargeable sur le web.

À la suite de l'utilisation du menu gmsh, Herezh++, par exemple pour un fichier de commande "toto.info", crée d'une part un fichier "toto_Gmsh.msh" qui permet de visualiser le maillage initial, et d'autre part un répertoire "toto_Gmsh" qui contient un ensemble de fichiers, en fait un fichier par grandeur à visualiser, ce qui permet de créer très simplement des animations. On peut néanmoins charger plusieurs fichiers à la fois dans gmsh si l'on veut superposer différentes grandeurs.

Il y a plusieurs différences entre gmsh et Gid.

Tout d'abord en pré-traitement (construction de maillage). A priori, Gid semble (mais ce n'est pas toujours évident) plus puissant et offre une plus grande étendue de possibilités. Par contre, il est un plus difficile d'accès, moins intuitif, et il n'y a pas d'interface qui permet de transformer automatiquement un maillage au format Gid en maillage au format her (c.-à-d. natif d'Herezh++). Cependant, il faut noter qu'avec un éditeur de texte classique, cette transformation est très facile et rapide dans le cas des noeuds et éléments, par contre pour les références c'est un peu plus laborieux. Dans le cas de gmsh, l'outil gmsh2her.pl permet de traduire automatiquement un maillage gmsh en un maillage her, ceci en transcrivant également les références de noeuds, et les références de faces (pour les éléments à une face) et arêtes (pour les éléments à une arête).

Dans le cadre du post-traitement (visualisation des résultats). A priori, Gid semble également plus complet, mais gmsh est très ergonomique et possède de nombreuses possibilités qui se révèlent performantes. Au bilan, relativement aux différentes possibilités couramment utilisées en pratique (car simples d'accès), gmsh est une très bonne alternative à Gid qui reste néanmoins une bonne solution industrielle.

En résumé, dans la phase actuelle les différences majeures entre ces deux outils sont les suivantes :

- pour un même résultat, les fichiers générés par gmsh sont beaucoup plus volumineux que pour Gid dans l'ancienne version de stockage de gmsh. Par contre dans la nouvelle (sortie par défaut d'Herezh++ à partir de la version 6.590).
- dans le cas de gmsh, seules les grandeurs définies aux noeuds sont visibles en isovaleurs. Aussi, toutes les grandeurs définies aux points d'intégrations sont systématiquement transportées par moyennage, aux noeuds (contrairement à Gid pour lequel on peut visualiser des grandeurs aux noeuds ou aux points d'intégrations),
- l'ergonomie de gmsh est plus aboutie (bien que ça soit un jugement peut-être un peu partial),
- gmsh2her.pl permet de créer automatiquement des maillages au format her à partir des maillages au format gmsh.
- gmsh semble plus robuste pour lire des fichiers à différents formats, par exemple step.

Un grand nombre de formats en entrée et sortie sont directement disponibles.

Au niveau des différents menus offerts par Herezh dans le cas de gmsh, on suit exactement la même logique que pour les autres types de sortie. On se reportera aux autres types de sortie pour plus d'information, ou plus facilement, directement à l'affichage d'herezh++.

En particulier, avec le format gmsh :

- il est possible de sortir une visualisation des différentes références : de noeuds, d'arêtes, de faces, d'éléments et de points d'intégration. C'est sortie est par défaut, mais il est possible de la supprimer pour avoir un fichier plus petit. À signaler que lorsque les références sont présentes, le nombre de noeuds et d'éléments est augmenté par l'ajout d'éléments "noeuds" (notion spécifique à gmsh, les éléments noeuds et les noeuds sont deux choses différentes), qui représentent les noeuds du maillage et les points d'intégration (éventuelles).
- lorsque le nombre de références est très important et/ou la taille du maillage est importante, les ressources en mémoire vive nécessaires pour tout visualiser deviennent

prohibitives. Il est alors possible d'indiquer que l'on souhaite un fichier par référence. Globalement l'ensemble des fichiers ainsi générés, est plus grande que le fichier unique, mais par contre chaque fichier est beaucoup plus petit que le fichier global d'où, en général, cela ne pose plus de problème pour les visualiser un par un.

47.4 Sortie des résultats au fil du calcul

Dans le cas où le traitement de la visualisation après le calcul n'est pas approprié, par exemple dans le cas où les fichiers générés sont trop volumineux du fait d'un grand nombre d'incrément sauvegardés, il est possible de traiter la visualisation au cours du calcul. En général, cette technique est à préconiser.

La démarche à suivre est alors la suivante :

- tout d'abord on définit les caractéristiques de la sortie que l'on veut avoir. Pour ce faire, on crée un fichier `.CVvisu` (voir cf.47.3.1). D'une manière pratique, pour créer le fichier, on peut effectuer un calcul très court (sur un incrément par exemple) à la suite de quoi on définit un fichier de contrôle `.CVvisu`.

- On définit la fréquence de sortie des résultats, qui est donc ici différente de celle des sauvegardes des incréments. Pour cela on utilise le mot clé

`FREQUENCE_SORTIE_FIL_DU_CALCUL` dans les paramètres liés à l'affichage des résultats (cf. 45) pour indiquer la fréquence sur les incréments, de sortie des résultats. Cette fréquence est indépendante de la fréquence de sauvegarde du fichier `.BI`.

Comme pour le paramètre sauvegarde (cf. (1)), il est possible d'indiquer un pas de temps de sauvegarde. Pour cela, à la suite du mot clé

`"FREQUENCE_SORTIE_FIL_DU_CALCUL"` on indique le mot clé `"INTER_TEMPS"` suivi d'un réel représentant le pas de temps que l'on désire entre deux sauvegardes au fil du calcul. En fait comme chaque calcul est effectué à un temps discret, il y a sauvegarde, dès que le temps du calcul est supérieur au dernier temps de sauvegarde + l'intervalle de temps indiqué. Le temps réel entre deux sauvegardes n'est donc qu'approximativement le temps indiqué à l'aide du paramètre `"INTER_TEMPS"`. L'approximation est d'autant meilleure que le pas de temps de calcul Δt est petit par rapport au pas de temps de sauvegarde.

Il est également possible de sauvegarder uniquement la structure finale. Pour cela, à la suite du mot clé

`"FREQUENCE_SORTIE_FIL_DU_CALCUL"` on indique le mot clé `"DERNIER_CALCUL"`.

Remarque *Dans tous les cas : sortie avec une fréquence donnée par numéro d'incrément ou par intervalle de temps, par défaut il y en plus sauvegarde du dernier calcul.*

Les résultats sont alors inscrits dans les différents fichiers habituels (`.maple`, `.vrml` etc..).

Dans le cas où le calcul stop à cause d'une erreur, normalement on récupère le dernier incrément valide. Si par la suite on effectue un redémarrage à un incrément valide, le précédent fichier de résultat est écrasé ! Il faut donc le renommer ou le recopier par exemple pour en garder une trace. En clair, il n'est pas prévu de sauvegarde automatique des fichiers de résultats ni de gestion particulière de ces fichiers. À chaque début de calcul, dans le cas d'une sortie au fil du calcul, tous les fichiers de sortie sont réinitialisés.

47.5 Exportation des grandeurs des points d'intégrations aux noeuds

Cette partie est disponible actuellement pour la sortie Gid et pour la sortie gmsh, mais les méthodes développées sont disponibles pour toutes les sorties. Aussi, en fonction des demandes, le transfert aux noeuds pourra s'utiliser avec tous les types de post-traitement.

Plusieurs méthodes étant possibles, le choix s'est porté sur un compromis précision-vitesse d'exécution. On décrit ici, les techniques de transfert utilisé selon la dimension de l'élément, son nombre de noeuds et le nombre de points d'intégration utilisés.

Dans le cas des éléments 1D :

- pour 1 point d'intégration : on reporte la grandeur du pt d'intégration à chaque noeud.
- pour 2 pt d'intégration : on extrapole linéairement depuis les pt d'intégrations vers les noeuds.
- pour 3, 4, 5, 6, 7 points d'intégration et quelques soit le nombre de noeuds (2, 3, 4), on extrapole linéairement à chaque noeud en utilisant les deux points d'intégration les plus proches du noeud.

Dans le cas des éléments 2D triangulaires :

- pour 1 point d'intégration : on reporte la grandeur du pt d'intégration à chaque noeud.
- pour 3 pt d'intégration, pour les deux types de répartition (au milieu des arrêtes ou près des noeuds sommets) : on extrapole linéairement depuis les pt d'intégrations vers les noeuds, quelque soit leurs nombres.
- pour 4 pt d'intégration, on ne tient pas compte du point central (le premier) et les valeurs aux noeuds sont obtenues par extrapolation des 3 autres points d'intégration (qui sont près des noeuds).
- pour 6 et 7 points d'intégration :
 - pour une interpolation à 3 noeuds (linéaire) ou 6 noeuds (quadratique) vue la proximité entre point d'intégration et noeud, on reporte à chaque noeud la valeur existante au point d'intégration le plus proche.
 - pour une interpolation à 10 noeuds (cubique complet), pour les 3 noeuds sommets, la même technique que pour 3 noeuds est utilisée. Pour les noeuds restants, une extrapolation linéaire est effectuée à partir des 3 points d'intégration les plus proche du noeud considéré.
- pour 12 points d'intégration, quelque soit le nombre de noeuds, vu la proximité entre point d'intégration et noeud, on reporte à chaque noeud la valeur existante au point d'intégration le plus proche.

Dans le cas des éléments 2D quadrangulaires :

- pour 1 point d'intégration : on reporte la grandeur du pt d'intégration à chaque noeud.
- pour 4 pt d'intégration et pour les 4 noeuds sommets, on extrapole linéairement vers les noeuds en considérant à chaque fois les 3 pt d'intégration les plus près du noeud. Pour les noeuds restants :
 - pour une interpolation à 8, 9 ou 16 noeuds (quadratique incomplet, quadratique complet et cubique) les grandeurs aux noeuds proviennent d'une extrapolation

linéaire à partir de 3 points d'intégration arbitraire (il y a 4 possibilités, dont deux toujours acceptables).

- pour 9 points d'intégration, quelque soit le nombre de noeud, il y a extrapolation linéaire à partir des 3 points d'intégration les plus proches du noeud.
- pour 16 points d'intégration, pour les 4 noeuds sommets, on extrapole linéairement vers les noeuds en considérant à chaque fois les 3 pt d'intégration les plus près du noeud. Pour les noeuds restants :
 - pour une interpolation à 8, 9 vu la position des points d'intégration, les grandeurs aux noeuds sont obtenues par une moyenne des valeurs aux deux points d'intégration les plus proches du noeud sauf pour le noeud 9 où la moyenne concerne les 4 points d'intégration l'entourant.
 - pour une interpolation à 16 noeuds (cubique complet) vu la position des noeuds et des points d'intégration, la grandeur à chaque noeud correspond à celle du point d'intégration le plus proche

Dans le cas des éléments 3D : hexaèdre, pentaèdre, tétraèdre :

- pour 1 point d'intégration on reporte la grandeur du pt d'intégration à chaque noeud.
- pour tous les autres cas de nombres de points d'intégration et pour tous les cas d'interpolation, on utilise la valeur du point d'intégration le plus proche, ou une moyenne de 2, 3 ou 4 valeurs aux points d'intégration, selon la position du noeud par rapport aux points d'intégration.

Ensuite l'ensemble des grandeurs aux noeuds est moyenné, c'est-à-dire on fait la somme des apports et on divise par le nombre d'apports.

47.6 Significations des grandeurs disponibles

Il s'agit ici des grandeurs en générales disponibles. Certaines grandeurs particulières liées par exemple à une loi de comportement spécifique, sont décrites avec la loi de comportement.

47.6.1 Grandeurs liés à la cinématique

Aux noeuds on dispose :

- de la position initiale du noeud
- de la position actuelle (à t) du noeud
- du déplacement entre 0 et t , ou entre $t-\Delta t$ et t .

Aux points d'intégration on dispose de :

- les composantes du tenseur de déformation : par défaut celle d'Almansi, on peut également obtenir celles du tenseur de Green Lagrange, et enfin celles du tenseur de déformation logarithmique.
- les déformations principales,
- les composantes de la vitesse de déformation,
- les vitesses de déformation principales,
- la variation des composantes du tenseur d'Almansi entre $t-\Delta t$ et t (ex : "Delta_def11" pour la composante 11)

- dans le cas de la présence d’une dilatation d’origine thermique, on a accès séparément à la déformation mécanique (c’est la déformation sans précision), la déformation totale qui correspond à la déformation thermique + mécanique (par exemple : "Almansi_totale11", "Green_Lagrange_totale11", "logarithmique_totale11").
- la trace/3 du tenseur de déformation : "Spherique_eps" = $trace(\epsilon)/3$.
- l’intensité du déviateur de déformation : "Q_eps" = $\sqrt{\bar{\epsilon} : \bar{\epsilon}}$
- le cosinus de trois fois l’angle de Lode (de phase) du déviateur de déformation : "Cos3phi_eps",
- la déformation duale de Mises : "def_duale_mises" = $\sqrt{2/3 * \bar{\epsilon} : \bar{\epsilon}}$,
- la déformation cumulée appelée certaine fois déformation équivalente : "def_equivalente" = $\int_0^t \sqrt{2./3. * \bar{D} : \bar{D}} dt$,
- la déformation au sens duale de Mises, maximale obtenue entre 0 et t : "def_duale_mises_maxi",
- la vitesse de déformation équivalente : "vitesse_def_equivalente" = $\sqrt{2./3. * \bar{D} : \bar{D}}$

47.6.2 Grandeurs liés aux contraintes

Aux points d’intégration on dispose de :

- Les composantes du tenseur des contraintes de Cauchy : ex "SIG11",
- les contraintes principales : ex "Sigma_principaleI",
- la trace/3 du tenseur de contrainte : "Spherique_sig" = $trace(\sigma)/3$.
- l’intensité du déviateur des contraintes : "Q_eps" = $\sqrt{\bar{\sigma} : \bar{\sigma}}$
- le cosinus de trois fois l’angle de Lode (de phase) du déviateur des contraintes : "Cos3phi_sig",
- la contrainte de Mises : "contrainte_mises",
- la contrainte de Tresca : "contrainte_tresca",

47.6.3 Grandeurs liées aux énergies

Sur l’ensemble de la structure et en chaque point d’intégration on dispose de :

- l’énergie élastique à t : "energie_elastique",
- la dissipation plastique cumulée de 0 à t : "dissipation_plastique",
- la dissipation visqueuse cumulée de 0 à t : "dissipation_visqueuse".

47.7 Remarque concernant les contraintes et déformations pour les membranes, plaques et coques

Dans le cas où l’on veut observer les contraintes ou déformations dans un repère ortho-normé, pour des éléments 2D dans un espace 3D (typiquement une membrane ou coque ou plaque en 3D), il n’est pas pertinent d’exprimer les coordonnées de ces tenseurs dans le repère global. Aussi, on définit un repère orthonormé local à l’élément obtenu de la manière suivante :

- Premier cas : la projection du vecteur \vec{I}_1 sur l’élément est non nulle. Dans ce cas, cette projection est normalisée et tient lieu de premier vecteur de la base locale. Le troisième vecteur est constitué de la normale à l’élément, et le second vecteur est obtenu par produit vectoriel des deux premiers.

- Second cas : la projection du vecteur \vec{I}_1 sur l'élément est nulle, on utilise alors la projection du vecteur \vec{I}_2 qui tient lieu alors de premier vecteur. La suite est identique au premier cas.

Ainsi, lorsque l'on trace des isovaleurs de composantes il peut y avoir basculement entre le premier cas et le second cas !

47.8 Remarque concernant le volume délimité par des membranes, plaques et coques

Le maillage d'une surface peut délimiter un volume, ou seulement une portion de volume compte tenu par exemple de symétries imposées. Il est possible, qu'Herezh calcul ce volume et en particulier son évolution dans le temps. Pour prendre en compte le problème des volumes partielles, on calcul le volume situé entre les 3 plans de base : xy , xz , yz et la surface. Dans le cas où la surface est fermée, ces trois volumes sont identiques. Lorsqu'il y a des symétries, en fonction de leurs positions et à partir des 3 volumes élémentaires, il est facile de remonter au volume global. Lorsqu'il ne s'agit pas d'une surface fermée, on obtient l'intégrale des volumes situés entre les plans de base et la surface.

Pour mettre en oeuvre le calcul de volume, il faut se reporter aux paramètres de calcul géométrique : (cf.5).

La méthode de calcul retenue est la suivante :

- triangulation (linéaire) de la surface
- calcul des volumes élémentaires
- bilans

Quatorzième partie
Utilitaires

48 Utilitaires

48.1 Fonctions 1D

Pour certaines données telle que les lois de comportement (fonction de charge), il est possible d'utiliser des fonctions 1D. Différentes types de fonctions sont disponibles. La déclaration des fonctions s'effectue à l'aide d'un nom qui sert d'identificateur de type puis par les données décrivant la fonction. Le tableau (220) donne la liste des fonctions disponibles.

TABLE 220 – liste des fonctions 1D

nom	commentaire simplifié	référence
COURBEPOLYLINEAIRE_1_D	ensemble de point reliés par des segments	48.1.1
CPL1D	ensemble de point reliés par des segments	48.1.2
COURBE_EXPOAFF	$f(x) = \gamma + \alpha(x^n)$	48.1.3
COURBE_UN_MOINS_COS	$f(x) = c/2(1 - \cos((x - a)\Pi/(b - a)))$	48.1.4
COURBEPOLYNOMIALE	$f(x) = \sum_{i=1}^n a_i x^i$	48.1.5
F1 Rond_F2	$f(x) = f1 \circ f2(x)$	48.1.6
F1 PLUS_F2	$f(x) = f1(x) + f2(x)$	48.1.7
F_CYCLIQUE	courbe qui se répète à intervalles réguliers avec un facteur d'amplification multiplicatif	48.1.8
F_CYCLE_ADD	courbe qui se répète à intervalles réguliers avec un facteur d'amplification additif	48.1.9
F_UNION_1D	courbe qui rassemble plusieurs, chacune définit sur un interval différent	48.1.10
COURBE_TRIPODECOS3PHI	$f(x) = (1. + \gamma \cos(3 x))^{(-n)}$	48.1.11
COURBE_SIXPODECOS3PHI	$f(x) = (1. + \gamma (\cos(3 x))^2)^{(-n)}$	48.1.12
COURBE_EXPO_N	$f(x) = (\gamma + \alpha x)^n$	48.1.13
COURBE_EXPO2_N	$f(x) = (\gamma + \alpha x^2)^n$	48.1.14
COURBE_RELAX_EXPO	$f(x) = (a - b) \exp(-c x) + b$	48.1.15
COURBE_COS	$f(x) = \cos(x)$	48.1.16
COURBE_SIN	$f(x) = \sin(x)$	48.1.17
COURBE_TANH	$f(x) = a + b \tanh((x - c)/d)$	48.1.18

48.1.1 Fonction poly-linéaire

Fonction poly-linéaire de mot clé "COURBEPOLYLINEAIRE_1_D" : construite à partir d'un ensemble de points relié par des segments. La fonction nécessite la donnée d'un ensemble de points dont la séquence démarre par le mot clé "Debut_des_coordonnees_des_points" et termine par le mot clé "Fin_des_coordonnees_des_points". Entre ces deux mots clés, on indique les points, un par ligne. Un point est constitué du mot clé "Coordonnee" suivi du mot clé "dim=" puis la dimension ici 2, ce qui signifie que le point est constitué d'un x et d'un y, et enfin les deux coordonnées en question. Un exemple de ligne représentant un point est :

Coordonnee dim= 2 0. 100.

dans cette exemple, $x=0.$, $y=100.$.La table (221) donne un exemple de déclaration d'une courbe poly-linéaire pour une fonction d'érouissage plastique de mot clé `loi_ecrouissage` (cf.28.4.3)

TABLE 221 – Exemple de déclaration d'une fonction d'érouissage plastique poly-linéaire

```
loi_ecrouissage COURBEPOLYLINEAIRE_1_D
  Debut_des_coordonnees_des_points
    Coordonnee dim= 2 0. 100.
    Coordonnee dim= 2 0.5 150.
    Coordonnee dim= 2 1. 200.
  Fin_des_coordonnees_des_points
```

Il est également possible d'introduire un décalage initiale en x et y . Dans ce cas, à la lecture les coordonnées des points de la courbe sont translatées du décalage. Le décalage en x est soustrait aux coordonnées x des points, tandis que le décalage en y est ajoute aux coordonnées y des points.

La table (222) donne un exemple de déclaration de courbes poly-linéaires avec translations initiales.

TABLE 222 – Exemple de déclaration d'une courbe poly-linéaire avec translations initiales

```
courbe_2 COURBEPOLYLINEAIRE_1_D
decalageX_= 10. decalageY_= 3.
# def des points constituant la courbe
  Debut_des_coordonnees_des_points
    Coordonnee dim= 2 0. 0.
    Coordonnee dim= 2 1. 0.5
  Fin_des_coordonnees_des_points
```

48.1.2 Fonction polylineaire-simple

Fonction identique à celle défini par le mot clé "COURBEPOLYLINEAIRE_1.D". En fait c'est une version simplifiée qui permet de définir une courbe poly-linéaire d'une manière compacte, mais le fonctionnement est identique au cas "COURBEPOLYLINEAIRE_1.D". La table (223) donne un exemple de déclaration de cette courbe simplifiée. `courbe1` est le nom de la courbe, choisit par l'utilisateur. "CPL1D" est l'identificateur du type de courbe. "DdlP" est l'identificateur de début des points. Ensuite il doit y avoir n paires de x et y , n étant quelconque, terminé par le mot clé : "FdlP".

TABLE 223 – Exemple de déclaration d’une courbe poly-linéaire simplifiée

```
courbe1 CPL1D Dd1P 0. 100000. 100. 50000.0 Fd1P
```

48.1.3 Fonction $f(x) = \gamma + \alpha(|x|^n)$

Fonction de type $f(x) = \gamma + \alpha(|x|^n)$. La fonction dépend de trois paramètres γ , α et n qui sont indiqués sur une ligne d’entrée selon la syntaxe :

gamma= <une valeur réelle> alpha= <une valeur réelle> n= <une valeur réelle>

La table (240) donne un exemple de déclaration de la fonction.

TABLE 224 – Exemple de déclaration d’une fonction de type $\sigma = \gamma - \alpha|x|^n$

```
f_coefficient COURBE_EXPOAFF
# def des coeff de la courbe expoaff
gamma= 1. alpha= -20. n= 1.01
```

48.1.4 Fonction $f(x) = \frac{c}{2}(1 - \cos(\frac{(x-a)\Pi}{(b-a)}))$

Fonction : $f(x) = \frac{c}{2}(1 - \cos(\frac{(x-a)\Pi}{(b-a)}))$, pour $x < a$ on a $f=0$, et pour $x > b$ on a $f=c$. La fonction permet de passer d’une manière continue et avec une dérivée continue, de la valeur 0 à c , ce qui peut être intéressant par exemple pour l’application progressive en dynamique d’une charge. La table (225) donne un exemple de déclaration de la fonction.

TABLE 225 – Exemple de déclaration d’une fonction de type $f(x) = \frac{c}{2}(1 - \cos(\frac{(x-a)\Pi}{(b-a)}))$

```
courbe3 COURBE_UN_MOINS_COS
# def des coeff de la fonction f(t)= c/2*(1-cos((x-a)*Pi/(b-a)))
a= 0. b= 1. c= 1.
```

48.1.5 Fonction polynomial $f(x) = \sum_{i=1}^n a_i x^i$

Fonction polynôme : $f(x) = \sum_{i=1}^n a_i x^i$. La table (226) donne un exemple de déclaration de la fonction. Le degré est quelconque.

48.1.6 Fonctions composées : $f(x) = f1 \circ f2(x)$

TABLE 226 – Exemple de déclaration d’une fonction polynomiale $f(x) = \sum_{i=1}^n a_i x^i$

```

courbe_exemple    COURBEPOLYNOMIALE # nom de la courbe puis le type de la courbe
# def des coefficients d'un polynome du troisieme degre 1+3x+6x^2+8x^3
debut_coef= 1. 3. 6. 8. fin_coef

```

Fonctions composées : $f(x) = f1 \circ f2(x)$. Les tables (227) et (228) donnent deux exemples de déclaration de fonctions composées. Les 2 fonctions à composer peuvent être définis soit explicitement cf.(228) soit via des noms de fonctions déjà définis cf.(227).

TABLE 227 – Exemple de déclaration d’une fonction composée $f(x) = f1 \circ f2(x)$. Les 2 fonctions sont définis par un nom qui est une référence à des fonctions définis par ailleurs dans la liste des courbes déjà définis.

```

courbe_exemple1    F1_ROND_F2 # nom de la courbe
courbe1= fonction_temperature # def de la premiere courbe par un nom de reference
courbe2= transfo1 # def de la seconde courbe par un nom de reference

```

TABLE 228 – Exemple de déclaration d’une fonction composée $f(x) = f1 \circ f2(x)$. Les 2 fonctions sont définis ici définis explicitement.

```

courbe_exemple2    F1_ROND_F2 # nom de la courbe
courbe1= COURBE_UN_MOINS_COS # def explicite de la premiere courbe
# def des coeff de la courbe demi sinus f(t)= c/2*(1-cos((x-a)*Pi/(b-a)))
# pour x < a => f=0, pour x>b => f=c
a= 0. b= 1. c= 1.
courbe2= COURBEPOLYNOMIALE # def explicite de la seconde courbe
# def des coefficients d'un polynome du troisieme degre 1+3x+6x^2+8x^3
debut_coef= 1. 3. 6. 8. fin_coef

```

48.1.7 Fonctions composées : $f(x) = f1(x) + f2(x)$

Fonctions composées : $f(x) = f1(x) + f2(x)$. Cette fonction permet la composition par addition de deux fonctions. Elle fonctionne au niveau des entrées/sorties de manière semblable à la fonction F1_ROND_F2. La table (229) donne un exemples de déclaration de fonctions somme. Comme pour la fonction F1_ROND_F2, les 2 fonctions à composer peuvent être définis soit explicitement cf.(228) soit via des noms de fonctions déjà définis cf.(227).

TABLE 229 – Exemple de déclaration d’une fonction composée $f(x) = f1(x) + f2(x)$. Les 2 fonctions sont définis par un nom qui est une référence à des fonctions définis auparavant.

```

courbecos    COURBE_UN_MOINS_COS
# def des coeff de la fonction f(t)= c/2*(1-cos((x-a)*Pi/(b-a)))
a= 0. b= 0.1 c= -0.0

courbe_const COURBEPOLYNOMIALE # nom de la courbe puis le type de la courbe
# def des coefficients d'un polynome constant
debut_coef= 1. fin_coef

courbe_somme F1_PLUS_F2 # nom de la courbe
courbe1= courbecos # def de la premiere courbe par un nom de reference
courbe2= courbe_const # def de la seconde courbe par un nom de reference

```

48.1.8 Fonctions cycliques : amplification multiplicative

Cette fonction, que l’on appellera $g(x)$, permet de définir un comportement cyclique selon une longueur de cycle donnée. Elle est composée d’une fonction de base $f(x)$, qui peut être quelconque, définit par ailleurs et de paramètres contrôlant le cycle. Soit cy la longueur du cycle et a un paramètre d’amplification. Le comportement est le suivant :

- pour $0 \leq (x - x_0) \leq cy$ $g(x) = y_0 + f(x - x_0)$
- pour $cy \leq (x - x_0) \leq 2 cy$ $g(x) = y_0 + f(cy) + a f(x - cy - x_0)$
- pour $2 cy \leq (x - x_0) \leq 3 cy$ $g(x) = y_0 + (1 + a) f(cy) + a^2 f(x - 2 cy - x_0)$
- ...
- pour $n cy \leq (x - x_0) \leq (n + 1) cy$ $g(x) = y_0 + (1 + a + a^2 + \dots + a^{n-1})f(cy) + a^n f(x - n cy - x_0) = y_0 + (a^n - 1)/(a - 1)f(cy) + a^n f(x - n cy - x_0)$

Le facteur “a” est un facteur d’amplification. Par défaut il vaut 1. S’il est différent de 1, pour le cycle “n”, la fonction initiale est multipliée par a^n . Enfin, au début de chaque cycle, la fonction est translatée de la valeur qu’elle avait à la fin du cycle précédent. On a ainsi une fonction continue, d’un cycle à l’autre. cf.(230) donne deux exemple d’utilisation d’une tel fonction. x_0 et y_0 représentent deux ofsets possibles selon x et y. Par défaut ces ofsets sont nuls, mais leurs utilisations permet par exemple de joindre plusieurs courbes.

48.1.9 Fonctions cycliques : amplification additive

Cette fonction, que l’on appellera $g(x)$, permet de définir un comportement cyclique d’une manière analogue à la fonction précédente (48.1.8) mais avec une amplification additive plutôt que multiplicative, selon une longueur de cycle donnée. Elle est composée d’une fonction de base $f(x)$, qui peut être quelconque, définit par ailleurs et de paramètres contrôlant le cycle. Soit cy la longueur du cycle et a un paramètre d’amplification. Le comportement est le suivant :

- n=1 : pour $0 \leq (x - x_0) \leq cy$ $g(x) = y_0 + a f(x - x_0)$

TABLE 230 – Exemple de déclaration d’une fonction cyclique avec un facteur d’amplification multiplicatif.

```

#.....
#  il s’agit d’une fonction qui est cyclique, a chaque debut de cycle
#  elle prend la valeur de la fin du cycle precedent + , la valeur d’une
#  fonction de base multipliee par un facteur d’amplitude puissance n,
#  n etant le nombre de cycle qui vaut 1 par default
#  ainsi on doit indiquer la longueur du cycle, mais le facteur d’amplitude est
#  facultatif. On indique au debut de la declaration, la fonction de base
#  Remarque: le premier cycle commence a x=0.
# *** exemple 1 de definition d’une courbe composee F_CYCLIQUE ****
#  1) ici on indique le nom de la courbe interne qui doit donc etre
#  definis par ailleurs
courbe_exemple1    F_CYCLIQUE # nom de la courbe
    courbe1= fonction_temperature # def de la courbe interne
    longueur_cycle_= 10  amplification_= 2

#  Il est egalement possible d’introduire un decalage en x et y. Le decalage en x
#  est soustrait a la valeur courante de x, tandis que le decalage en y est ajoute
#  a la valeur finale de la fonction. Exemple de syntaxe
#  longueur_cycle_= 10  amplification_= 2  decalageX_= 10. decalageY_= 3.

# *** exemple 2 de definition d’une courbe composee F_CYCLIQUE ****
#  ici on indique explicitement la courbe interne
#  definis par ailleurs
courbe_exemple2    F_CYCLIQUE # nom de la courbe
# def d’une fonction echelon (interne) avec une attenuation des angles
courbe1= COURBEPOLYLINEAIRE_1_D
    Debut_des_coordonnees_des_points
        Coordonnee dim= 2 0.  0.
        Coordonnee dim= 2 0.2 0.05
        Coordonnee dim= 2 0.4 0.2
        Coordonnee dim= 2 0.6 0.8
        Coordonnee dim= 2 0.8 0.95
        Coordonnee dim= 2 1.  1.
        Coordonnee dim= 2 10.  1.
    Fin_des_coordonnees_des_points
# def des parametres internes du cycle
longueur_cycle_= 5.

```

- n=2 : pour $cy \leq (x - x_0) \leq 2 cy$ $g(x) = y_0 + f(cy) + 2 a f(x - cy - x_0) = y_0 + a (f(cy) + 2 f(x - cy - x_0))$
- n=3 : pour $2 cy \leq (x - x_0) \leq 3 cy$ $g(x) = y_0 + a (1+2) f(cy) + 3 a f(x - 2 cy - x_0) =$

- $y_0 + a ((1 + 2)f(cy) + 2 f(x - cy - x_0))$
- ...
- cycle n : pour $(n-1) cy \leq (x-x_0) \leq n cy$ $g(x) = y_0 + a (1+2+3+\dots+(n-1))f(cy) + n a f(x - n cy - x_0) = y_0 + a (((n-1)^2 + n - 1)/2 f(cy) + n f(x - (n-1) cy - x_0))$

Le facteur “a” est un facteur d’amplification. Par défaut il vaut 1. S’il est différent de 1, pour le cycle “n”, la fonction initiale est multipliée par $n a$. Enfin, au début de chaque cycle, la fonction est translatée de la valeur qu’elle avait à la fin du cycle précédent. On a ainsi une fonction continue, d’un cycle à l’autre. cf.(231) donne deux exemple d’utilisation d’une tel fonction. x_0 et y_0 représentent deux ofsets possibles selon x et y. Par défaut ces ofsets sont nuls, mais leurs utilisations permet par exemple de joindre plusieurs courbes.

48.1.10 Fonctions réunion de domaine

Cette fonction, que l’on appellera $g(x)$, permet de globaliser le comportement de n fonctions, que l’on appellera $f_i(x)$, chacune de ces fonctions membres étant définit sur un interval propre $[x_i, x_{i+1}[$ différent des autres fonctions. Cependant l’ensemble des intervals doit être continu de manière à représenter un grand interval plein (sans trou). La table (232) donne un exemple d’utilisation d’une telle fonction. Le nombre de fonction n’est pas limité. La borne mini et la borne maxi sont facultative (par défaut $\pm\infty$. Après chaque fonction, on passe à la ligne suivante et sur chaque ligne on doit avoir le nom d’une courbe (ou la définition d’une sous courbe) et la borne supérieure de l’intervalle de définition. S’il la valeur x est inférieure à la borne mini, on appel la première fonction avec la valeur x, si x est supérieure à la borne maxi, on appel la dernière fonction avec la valeur x.

48.1.11 Fonctions $f(x) = (1. + \gamma \cos(3 x))^{(-n)}$

Fonction de type $f(x) = (1. + \gamma \cos(3 x))^{(-n)}$. La fonction dépend de deux paramètres γ et n qui sont indiqués sur une ligne d’entrée selon la syntaxe :

gamma= <une valeur réelle> n= <une valeur réelle>

La table (233) donne un exemple de déclaration de la fonction.

48.1.12 Fonctions $f(x) = (1. + \gamma (\cos(3 x))^2)^{(-n)}$

Fonction de type $f(x) = (1. + \gamma (\cos(3 x))^2)^{(-n)}$. Fonction du même type que la précédente mais en plus qui est pair en x. La fonction dépend de deux paramètres γ et n qui sont indiqués sur une ligne d’entrée selon la syntaxe :

gamma= <une valeur réelle> n= <une valeur réelle>

La table (234) donne un exemple de déclaration de la fonction.

48.1.13 Fonctions $f(x) = (\gamma + \alpha x)^n$

Fonction du type $f(x) = (\gamma + \alpha x)^n$. La fonction puissance permet d’avoir des valeurs toujours positives. La fonction dépend de 3 paramètres qui sont indiqués selon la syntaxe :

gamma= <une valeur réelle> alpha= <une valeur réelle> n= <une valeur réelle>

La table (235) donne un exemple de déclaration de la fonction.

TABLE 231 – Exemple de déclaration d’une fonction cyclique avec un facteur d’amplification additif.

```

#.....
#  il s’agit d’une fonction qui est cyclique, a chaque debut de cycle
#  elle prend la valeur de la fin du cycle precedent + , la valeur d’une
#  fonction de base multipliee par un facteur d’amplitude fois n,
#  n etant le nombre de cycle qui vaut 1 par default
#  ainsi on doit indiquer la longueur du cycle, mais le facteur d’amplitude est
#  facultatif (par default = 1.).
#  On indique au debut de la declaration, la fonction de base
#  Remarque: le premier cycle commence a x=0.
# *** exemple 1 de definition d’une courbe composee F_CYCLE_ADD ****
#  1) ici on indique le nom de la courbe interne qui doit donc etre
#  definis par ailleurs
courbe_exemple1    F_CYCLE_ADD # nom de la courbe
    courbe1= fonction_temperature # def de la courbe interne
    longueur_cycle_= 10  amplification_= 2

#  Il est egalement possible d’introduire un decalage en x et y. Le decalage en x
#  est soustrait a la valeur courante de x, tandis que le decalage en y est ajoute
#  a la valeur finale de la fonction. Exemple de syntaxe
#  longueur_cycle_= 10  amplification_= 2  decalageX_= 10. decalageY_= 3.

# *** exemple 2 de definition d’une courbe composee F_CYCLE_ADD ****
#  ici on indique explicitement la courbe interne
#  definis par ailleurs
courbe_exemple2    F_CYCLE_ADD # nom de la courbe
# def d’une fonction echelon (interne) avec une attenuation des angles
courbe1= COURBEPOLYLINEAIRE_1_D
    Debut_des_coordonnees_des_points
        Coordonnee dim= 2 0.  0.
        Coordonnee dim= 2 0.2 0.05
        Coordonnee dim= 2 0.4 0.2
        Coordonnee dim= 2 0.6 0.8
        Coordonnee dim= 2 0.8 0.95
        Coordonnee dim= 2 1.  1.
        Coordonnee dim= 2 10.  1.
    Fin_des_coordonnees_des_points
# def des parametres internes du cycle
longueur_cycle_= 5.

```

48.1.14 Fonctions $f(x) = (\gamma + \alpha x^2)^n$

TABLE 232 – Exemple de déclaration d’une fonction union permettant de globaliser plusieurs fonctions déjà définis.

```
#.....
#   il s'agit d'une fonction qui agglomere plusieurs fonctions deja existantes
#   chaque fonction est defini sur un interval disjoint des autres
#   Les intervals doivent se suivre
#   Xmin= precise la borne inferieur de definition de la fonction
#       c'est un parametre facultatif, s'il manque, la borne inf est - l'infini
#   ensuite il y a une liste de courbes:
#       suivi chacune de la fin de l'intervale correspondant
#   ici cf1 est pour x appartenant a [-10,0[, cf2 est pour x appartenant a [0,10[
#       cf3 est pour x appartenant a [10,20[

courbe_exemple    F_UNION_1D # nom de la courbe
  Xmin= -10  courbe= cf1 Xmax= 0
              courbe= cf2 Xmax= 10
              courbe= cf3 Xmax= 20
  fin_courbe_union"

#
#   pour la dernier courbe il est possible de ne pas faire figurer Xmax=, dans
#   ce cas, cela signifie que la borne finale est l'infini
#   enfin, si la valeur de x est en dehors de l'intervale, on applique la premiere
#   (x<Xmin) ou la derniere fonction (x>Xmax)
```

TABLE 233 – Exemple de déclaration d’une fonction de type $f(x) = (1 + \gamma \cos(3x))^{(-n)}$

```
f_coefficient COURBE_TRIPODECOS3PHI
# def des coeff de la courbe tripode
gamma= 0.9  n= 0.1
```

TABLE 234 – Exemple de déclaration d’une fonction de type $f(x) = (1 + \gamma (\cos(3x))^2)^{(-n)}$

```
f_coefficient COURBE_SIXPODECOS3PHI
# def des coeff de la courbe tripode
gamma= 0.9  n= 0.1
```

Fonction du type $f(x) = (\gamma + \alpha x^2)^n$. Cette fonction est très proche de la précédente si ce n’est le fait d’utiliser x^2 à la place de x . Le fait d’utiliser x^2 et la fonction puissance

TABLE 235 – Exemple de déclaration d’une fonction de type $f(x) = (\gamma + \alpha x)^n$

```
#.....
# exemple de definition d’une courbe COURBE_EXPO_N ( f(x) = (gamma + alpha * x)**n )|
#.....
#
courbe_monte    COURBE_EXPO_N # nom de la courbe puis le type de la courbe
                # def des coeff de la courbe COURBE_EXPO_N
                gamma= 10. alpha= -2. n= 1.3
```

permet d’avoir des valeurs toujours positives et une fonction pair. La fonction dépend de 3 paramètres qui sont indiqués selon la syntaxe :

gamma= <une valeur réelle> alpha= <une valeur réelle> n= <une valeur réelle>

La table (236) donne un exemple de déclaration de la fonction.

TABLE 236 – Exemple de déclaration d’une fonction de type $f(x) = (\gamma + \alpha x^2)^n$

```
#.....
# exemple de definition d’une courbe COURBE_EXPO2_N ( f(x) = (gamma + alpha * x*x)**n )|
#.....
#
courbe_monte    COURBE_EXPO2_N # nom de la courbe puis le type de la courbe
                # def des coeff de la courbe COURBE_EXPO2_N
                gamma= 10. alpha= -2. n= 1.3
```

48.1.15 Fonctions $f(x) = (a - b) \exp(-c x) + b$

Fonction du type $f(x) = (a - b) \exp(-c x) + b$. Cette fonction permet de passer progressivement de la valeur a pour $x=0$ à la valeur b pour $x=\infty$, sous forme d’une tangente. La fonction dépend de 3 paramètres qui sont indiqués selon la syntaxe :

xa= <une valeur réelle> xb= <une valeur réelle> xc= <une valeur réelle>

La table (237) donne un exemple de déclaration de la fonction.

48.1.16 Fonctions $f(x) = e \cos(\alpha x + \beta)$

Fonction classique du type $f(x) = e \cos(\alpha x + \beta)$. Il est possible d’indiquer une borne mini (appelée “a” pour la suite) et maxi (appelée “b” pour la suite) pour les valeurs de x. Dans le cas où ces bornes sont différentes des bornes par défaut $(-\infty, +\infty)$ on obtient le fonctionnement suivant :

- pour $x < a$ on a $f(x) = f(a)$
- pour $a \leq x \leq b$, $f(x) = e \cos(\alpha x + \beta)$

TABLE 237 – Exemple de déclaration d’une fonction de type $f(x) = (a - b) \exp(-c x) + b$

```
#.....
# exemple de definition d'une courbe COURBE_RELAX_EXPO ( f(x) = (a-b)*exp(-c*x) + b )
#
courbe_monte    COURBE_RELAX_EXPO # nom de la courbe puis le type de la courbe
                # def des coeff de la courbe COURBE_RELAX_EXPO
                xa= 2. xb= -1. xc= 1.3
```

– pour $b < x$ on a $f(x) = f(b)$

La fonction dépend de 3 coefficients principaux :

1. “e” qui représente un facteur d’amplification du cos (=1 par défaut)
2. “ α ” et “ β ” qui permettent de faire une transformation linéaire sur x. Par défaut $\alpha = 1$ et $\beta = 0$

Par défaut l’angle est supposé exprimé en radian, mais il est également possible d’indiquer que l’on veut une unité d’angle en degré à l’aide de la présence, après les précédents paramètre, du mot clé : “en_degre_”

La table (238) donne un exemple de déclaration de la fonction.

TABLE 238 – Exemple de déclaration d’une fonction de type $f(x) = e \cos(\alpha x + \beta)$

```
courbe_monte    COURBE_COS # nom de la courbe puis le type de la courbe
                # def des coeff de la courbe= mini et maxi de x
                # pour x < a => f=f(a), pour x>b => f=f(b)
                # a et b sont facultatif, par default = -l’infini et + l’infini
                a= 0. b= 1. ampli= 2. alph= 3. bet= 0.5 en_degre_
```

48.1.17 Fonctions $f(x) = e \sin(\alpha x + \beta)$

Fonction classique du type $f(x) = e \sin(\alpha x + \beta)$. Il est possible d’indiquer une borne mini (appelée “a” pour la suite) et maxi (appelée “b” pour la suite) pour les valeurs de x. Dans le cas où ces bornes sont différentes des bornes par défaut ($-\infty, +\infty$ on obtient le fonctionnement suivant :

- pour $x < a$ on a $f(x) = f(a)$
- pour $a \leq x \leq b$, $f(x) = e \sin(\alpha x + \beta)$
- pour $b < x$ on a $f(x) = f(b)$

La fonction dépend de 3 coefficients principaux :

1. “e” qui représente un facteur d’amplification du cos (=1 par défaut)
2. “ α ” et “ β ” qui permettent de faire une transformation linéaire sur x. Par défaut $\alpha = 1$ et $\beta = 0$

Par défaut l'angle est supposé exprimé en radian, mais il est également possible d'indiquer que l'on veut une unité d'angle en degré à l'aide de la présence, après les précédents paramètres, du mot clé : "en_degre_"

La table (239) donne un exemple de déclaration de la fonction.

TABLE 239 – Exemple de déclaration d'une fonction de type $f(x) = e \sin(\alpha x + \beta)$

```

courbe_descente  COURBE_SIN  # nom de la courbe puis le type de la courbe
# def des coeff de la courbe= mini et maxi de x
# pour x < a => f=f(a), pour x>b => f=f(b)
# a et b sont facultatif, par défaut = -l'infini et + l'infini
a= 0. b= 1. ampli= 2. alph= 3. bet= 0.5 en_degre_

```

48.1.18 Fonction $f(x) = a + b \tanh((x - c)/d)$

Fonction de type $f(x) = a + b \tanh((x - c)/d)$. La fonction dépend de 4 paramètres a , b , c , d , qui sont indiqués sur une ligne d'entrée selon la syntaxe :

a= <une valeur réelle> b= <une valeur réelle> c= <une valeur réelle> d= <une valeur réelle>

La table (??) donne un exemple de déclaration de la fonction.

TABLE 240 – Exemple de déclaration d'une fonction de type $f(x) = a + b \tanh((x - c)/d)$

```

f_coefficient  COURBE_TANH
# def des coeff de la courbe
a= 100. b= -32. c= 295. d= 22.

```

Quinzième partie

Chronologie et historique des mises à jour

49 Introduction

Il s'agit ici de retracer l'historique des mises à jour, vues du côté de l'utilisateur. Cette partie débute en 2004, alors que le projet existe depuis 8 ans, ce qui explique que l'on soit à la version 5.

50 Liste exhaustive

1. **Version 5.00** Juin 2004 (version de départ de l'historique)
2. **Version 5.01** Introduction de la thermo-dépendance dans le module d'Young des lois iso-élastiques : 2D en déformations planes, 2D en contraintes planes, et 3D. Modification de la sortie au fil du calcul, en introduisant pour la sortie de type Maple, un contrôle du style de sortie
3. **Version 5.02** Juillet 2004 : Mise au point de la loi d'hyper-élasticité proposée par Laurent Orgéas.
4. **Version 5.10** Octobre 2004 : Mise en place de l'interfaçage avec le post-processeur Gid.
5. **Version 5.20** 10 Octobre 2004 : Début de la Mise en place de l'interfaçage XML.
6. **Version 5.21** 16 Octobre 2004 : Mise en place d'un type de courbe obtenue par composition de deux autres courbes.
7. **Version 5.22** 16 Octobre 2004 : Mise en place de la loi de Tait permettant de définir le coefficient de dilatation en fonction de la pression.
8. **Version 5.23** 24 Octobre 2004 : Amélioration de l'implantation de la loi de tait, avec en particulier la possibilité de sortie en post-traitement : la température de transition, le volume spécifique, le coefficient de dilatation, la conductivité, la capacité calorifique, le module de compressibilité. Modification de l'affichage des ordres de post-traitement, avec cadrage sur 50 caractères, pour palier au défaut d'affichage sur pc linux.
9. **Version 5.30** 4 novembre 2004 : Mise en place d'une sortie de grandeurs évoluées au niveau de Gid (en fait des grandeurs directement tensorielles : tenseurs d'ordre 2, vecteurs, et scalaires). La vitesse de sortie de grandeurs tensorielles est a priori bien plus rapide que celle de la sortie des composantes individuellement. Amélioration du menu et des fonctionnalités de choix des isovaleurs. Mise en place de sortie par défaut pour le format Gid. Mise en place également d'une sortie automatique "a priori" du maillage initiale.
10. **Version 5.31** 7 novembre 2004 : Correction d'un bug (petit mais désagréable) perte de mémoire et vitesse importante dans le cas de plus de 1000 incréments, sortie des ddl actifs associés au ddl maxi à chaque itération (pour le contrôle de la convergence),
11. **Version 5.32** 8 novembre 2004 : Mise en place d'un chargement type 5, piloté entièrement et exactement par une liste de points.
12. **Version 5.33** 21 novembre 2004 : Mise en place d'une loi de Maxwell 3D. La partie sphérique est celle de Hooke, la partie visqueuse est relative à la partie déviatoire des tenseurs.
13. **Version 5.34** 1 décembre 2004 : Mise en place d'une viscosité linéaire optionnelle sur la partie sphérique pour la loi de Maxwell 3D.
14. **Version 5.35** 5 décembre 2004 : Correction de bug sur Maxwell3D, ajout d'une viscosité son-linéaire (pour l'instant en test).
15. **Version 5.40** 22 décembre 2004 : OK pour Maxwell3D, et mise en place d'une sortie des réactions aux noeuds, accessible pendant le calcul, au même titre que les autres ddl.
16. **Version 5.41** 22 décembre 2004 : première version de la mise en place d'une loi hypo-élastique linéaire 3D avec possibilité d'une dépendance thermique pour les paramètres matériels.
17. **Version 5.42** 22 décembre 2004 : mise en place dans la loi de Maxwell3D de la possibilité de ne prendre en compte que la partie déviatorique (l'apport de la partie sphérique est donc nulle ici).
18. **Version 5.50** 5 janvier 2005 : première mise en place d'éléments axisymétriques : des triangles linéaires et des triangles quadratiques, première mise en place de conditions limites de forces hydrodynamique.
19. **Version 5.51** 15 janvier 2005 : première mise en place du mécanisme d'interaction en format XML avec Herezh++.
20. **Version 5.52** 25 janvier 2005 : Mise en place de la possibilité d'utiliser dans les lois hypo-élastiques une compressibilité calculée avec une loi thermo-physique.
21. **Version 5.53** 29 janvier 2005 : Mise en place de la possibilité d'avoir des ddls aux noeuds entrés dans un ordre quelconque. Dans tous les cas, l'ensemble des composantes des ddl sont introduites dans le noeuds même si un seul ddl est fixé par exemple. Il n'y a pas de conséquence pour l'utilisateur, sauf qu'ainsi il peut apparaître des ddl qui n'ont pas été expressément spécifiés aux noeuds (par exemple, dans le .ddl, lorsqu'une seule direction est bloquée, les trois composantes de la réaction sont présentes, et les réactions des ddl libres sont évidemment nulles).
22. **Versions 5.54 - 5.70** mars 2005 : Mise en place :
 - de la possibilité d'utiliser Herezh++ comme Umat extérieure, en particulier avec le logiciel Abaqus, ceci via l'utilisation de deux pipes nommés.
 - de la possibilité d'utiliser des Umat extérieures via l'utilisation de deux pipes nommés,
 - implantation de l'hystérésis en 3D, avec une nouvelle gestion des points d'inversion,
 - possibilité d'utiliser une loi de comportement élastique de Hooke en ne retenant que la partie sphérique ou que la partie déviatorique.
23. **Version 5.71** mi-Avril 2005 : Premières validations de la loi d'hystérésis sur un cas radial, avec inversions et coïncidences.
24. **Version 5.72** 17-Avril 2005 : Mise en place du calcul d'Umat pour les lois additives en contraintes et pour les lois des mélanges en contraintes.
25. **Versions 5.73 .. 5.75** 12 mai 2005 : Correction de bug qui permettent maintenant le fonctionnement effectif de l'umat hystérésis avec abaqus et avec Herezh++. Introduction des paramètres de réglage pour la résolution de l'équation constitutive de l'hystérésis. Introduction de nouvelles variables particulières pour la sortie d'informations propres à l'hystérésis.
26. **Version 5.76** 14 mai 2005 : Correction de bug d'affichage sur Gid, pour les grandeurs aux points d'intégrations, pour les triangles et les quadrangles. Correction de bug dans stamm correspondant. Introduction dans stamm de nouvelles références d'arrêtes et d'éléments pour les triangles et les treillis triangulaires. Introduction dans Herezh d'un élément quadratique triangulaire avec 3 points d'intégration strictement interne à l'élément. Introduction sur l'élément pentaédrique linéaire de la possibilité d'utiliser 6 points d'intégration (au lieu de 2 par défaut).
27. **Version 5.77** 5 juin 2005 : Première mise en place d'une loi hypo-élastique non linéaire qui dépend du second invariant de la déformation.
28. **Version 5.78** 14 juin 2005 : Mise en place d'une fonction 1D qui permet de composer des fonctions existantes sous forme d'une somme.
29. **Versions 5.79 .. 5.80** 28 juin 2005 : Mise en place de la loi de Mooney-Rivlin en 3D. Validations mono dimensionnelle uniquement.
30. **Versions 5.81 .. 5.84** 9 juillet 2005 : Mise en place de la procédure Umat pour la loi de Mooney-Rivlin, correction de Bug, mise en place du calcul sélectif des éléments de métrique à $t=0$ et t , en fait les grandeurs ne sont calculées qu'au premier passage, d'où un gain de vitesse et aussi un encombrement mémoire plus important.
31. **Version 5.86** 21 septembre 2005 : Mise en place d'une thermo-dépendance possible des paramètres de la loi d'hystérésis, ceci pour la loi 3D et la loi 1D. Le paramètre de Prager n'est plus limité aux valeurs supérieures à 2, pour le cas 1D, d'une manière identique au cas 3D.

32. **Versions 5.87** 26 septembre 2005 : Mise en place d'une thermo-dépendance possible des paramètres des lois de Mooney-Rivlin 1D et 3D.
33. **Versión 5.88** 18 octobre 2005 : Correction de bug sur la sortie des réactions. Mise en place de la possibilité de sortir des réactions dans de format Maple et dans le format Gid. Documentation de la mise en place du type de déformation.
34. **Versions 5.89 .. 5.91** Fin de la mise en place d'élément 2D triangles cubiques (par contre pas de sortie possible pour l'instant avec GID et ce type d'element). Mise en place des sorties des différentes sortes d'énergies internes : élastique, plastique, visqueuse, soit sous forme globale soit sous forme locale avec laquelle il est possible de sortir des iso-valeurs.
35. **Versión 5.92** Correction des principaux bugs sur la loi de Mooney-Rivlin 3D y compris la version Umat.
36. **Versión 5.93** Mise en place du calcul des puissances réelles en sortie globale.
37. **Versión 5.94** 13 octobre 2005, modification des sorties : pour Gid, la sortie du maillage initiale devient implicite, pour les numéros d'incrément à sortie sur le .BI on peut mettre 0 (correction d'un bug), ce qui signifie alors que le fichier .BI sera vide, affichage des énergies en fonction des affichages généraux prévus (cf. paramètres d'affichage), correction d'un bug d'affichages des grandeurs aux points d'intégrations en format .maple.
38. **Versions 5.95 - 5.96** 15-23 novembre 2005. Correction de bugs correspondant à la sortie au fil du calcul. Mise en place d'un algorithme DFC plus proche de la théorie classique. L'algorithme actuel n'utilise pas l'équation d'équilibre au premier pas de temps, ce qui peut poser des problèmes dans le cas de restart multiple.
39. **Versión 5.97** 27 novembre 2005. Correction de bugs pour la mise en place de charges ponctuelles dans le cas axi-symétrique. Introduction de la possibilité de faire une sortie au fil du calcul selon un delta t.
40. **Versión 5.98** 4 décembre 2005. Mise en place du calcul systématique de la variation de la vitesse de déformation virtuelle, avec optimisation, car c'est une grandeur constante par élément dans le cas d'interpolations classiques. Essai de l'intégration également du calcul uniquement à l'initialisation des variations en fait on abandonne pour les variations de vecteurs de bases car il faut stocker et recopier, et la recopie est aussi longue que le calcul initial!).
41. **Versión 5.99** 15 janvier 2006. Mise en place d'éléments axisymétriques quadrangulaires : linéaires, quadratiques incomplets, quadratiques complets.
42. **Versions 5.991 et 5.992** 20 et 25 janvier 2006. Correction de bugs de lecture de thermo-dépendance des paramètres pour les lois de Mooney-Rivlin et d'hélasto-hystérésis.
43. **Versión 5.993** 28 janvier 2006. Correction d'un bug sur les sorties d'énergies élastiques internes en explicite.
44. **Versión 5.994** 31 janvier 2006. Introduction de loi 1D 2D et 3D qui ne font rien mécaniquement! .
45. **Versión 5.995** 4 février 2006. Utilisation "recherche" : introduction d'un algorithme expérimental au niveau de Tchamwa permettant de moduler l'application du coefficient φ en fonction de la valeur absolue de l'accélération pour chaque ddl.
46. **Versión 5.996** 22 février 2006. Conception et mise en place d'une classe d'intégrateurs Runge-Kutta imbriquées avec pilotage d'erreur : première application en hystérésis 1D.
47. **Versión 6.00** 22 février 2006. Debuggage de la version 1D hystérésis en Runge-Kutta : fonctionnement correcte. Mise en place, et premiers tests d'une méthode de relaxation dynamique dont l'idée est proposée par Julien Troufflard. La méthode est implantée pour tous les schémas dynamiques (explicites ou dynamiques), mise en place du référencement IDDN!!.
48. **Versión 6.01** 16 mars 2006. Debuggage relaxation dynamique et hystérésis. Informations sur la sortie des grandeurs tensorielles en format maple.
49. **Versión 6.02** 18 mars 2006. Mise en place de l'intégration par Runge-Kutta pour l'élasto-hystérésis 3D. Mise en place d'"exceptions C++" pour la gestion de non convergence des lois de comportement, mise en place d'un paramètre permettant de limiter le nombre de boucle interne d'inversion ou de coïncidence sur un même pas, et d'une manière générale amélioration de la robustesse des algorithmes d'hystérésis.
50. **Versión 6.03** Première mise en place du calcul des énergies élastiques et plastiques en élasto-hystérésis 3D.
51. **Versión 6.031** Correction de bug dans l'entrée des paramètres des différents algorithmes, résultant du mélange maintenant possible de paramètres spécifiques à l'algorithme (ex phi pour Tchamwa) et globales aux algorithmes (ex : relaxation dynamique).
52. **Versión 6.032** Correction de bug en lecture de données lorsque l'on passe d'un calcul implicite à un calcul explicite.
53. **Versión 6.033** 26 mars 2006 : Correction de bug sur le calcul de charge volumique en 3D.
54. **Versión 6.034** 2 avril 2006 : Correction de bug sur la loi visco-élastique de maxwell 3D.
55. **Versions 6.04 - 6.41** 8 avril 2006 : Mise en place du calcul systématique des énergies. Mise en place du calcul de l'énergie visqueuse d'origine numérique que l'on introduite pour l'amortissement.
56. **Versión 6.05** 12 avril 2006 : Première mise en place du contact en explicite DFC. Introduction des mouvements solides au niveau de la lecture des maillages et possibilité de sauvegarder les nouveaux maillages. Mise en place de la possibilité d'imposer des déplacements relatif au calcul précédent (à t).
57. **Versión 6.06** 5 mai 2006 : Introduction des loi de comportement en frottement : loi de Coulomb régularisée ou non avec frottement statique ou cinématique. Optimisation de la recherche d'un point dans un élément. Mise en place de la gestion des énergies liées au contact-frottement, en variables globales et locales.
58. **Versión 6.061** 22 mai 2006 : Introduction dans la doc des éléments quadrangle cubique complet. Passage du nombre de point d'intégration de 9 à 16 pour ces éléments. Modif sortie des résultats pour la loi d'hystérésis. Modif des sorties du nombre d'itération et step pour les Umat.
59. **Versions 6.062-6.065** 25 mai-3juin 2006 : Correction d'un bug dans le calcul de la coïncidence pour la loi d'élasto-hystérésis en 3D.
60. **Versión 6.066** 4 juin 2006 : Correction d'un Bug sur le fonctionnement de la relaxation dynamique en implicite.
61. **Versión 6.067** 5 juin 2006 : Correction d'un Bug sur le fonctionnement du comportement des forces hydrodynamique en 3D.
62. **Versión 6.07** 27 juin 2006 : Mise en place d'un algorithme global de type Runge-Kutta pour résoudre l'équilibre dynamique de la structure.
63. **Versión 6.08** 10 aout 2006 : Première mise en place d'un algorithme de Galerkin discontinu : celui de Bonelli.
64. **Versions 6.081 - 6.082** 20 aout 2006 : Correction de Bug sur l'algo de Bonelli, semble fonctionner correctement. Mise à jour de la doc au niveau des algorithmes : séparation des différents algo, en particulier galerkin continu et discontinu.
65. **Versión 6.083** 1 sept 2006 : Introduction de la possibilité d'utiliser le pas de temps critique de la méthode DFC pour paramétrer les pas de temps courant, mini et maxi.
66. **Versión 6.084** 18 sept 2006 : Correction de bugs mineures. On retire de la sauvegarde celle des métriques à t, normalement c'était superflu, et cela doit diminuer fortement la taille du .BI.
67. **Versions 6.085-6.087** 28 sept 2006 : Correction de bugs mineures et amélioration du calcul des bases naturelles. Mise en place de la possibilité de ne sauvegarder que le dernier incrément valide.
68. **Versión 6.089** 5 oct 2006 : Mise en place dans l'algo de Tchamwa, d'un pilotage permettant de prendre en compte une valeur moyenne d'accélération dans le temps.
69. **Versión 6.090** 8 oct 2006 : Modification et amélioration du contrôle d'avancement temporel pour l'algorithme de Runge-Kutta global (équilibre de la structure).
70. **Versión 6.091** 13 oct 2006 : Mise en place des sorties des contraintes globales sur un repère local orthonormé adapté pour les membranes utilisées en 3D.
71. **Versions 6.092-6.093** 21-23 oct 2006 : Correction de bug sur le chargement des éléments axisymétriques.
72. **Versions 6.094-6.095** 7 nov 2006 : Correction de bug sur l'utilisation de la masse consistance dans le cas de Bonelli et sur la lecture des paramètres de contrôle.

73. **Version 6.096** 8 nov 2006 : Correction d'un bug sur le contact (destruction d'élément contact).
74. **Version 6.097** 14 nov 2006 : Mise en place de la vérification de la singularité de la matrice de raideur. Amélioration de l'affichage du message de Warning, lorsque deux références de conditions limites sont identiques et que les valeurs numériques associées sont différentes.
75. **Version 6.098** 15 nov 2006 : Mise en place d'une limitation sur le facteur de contrôle de la proportion dans la loi de comportement des mélanges. Le facteur est maintenant borné de manière à être toujours entre 0 et 1.
76. **Version 6.099** 16 nov 2006 : Correction d'un bug sur le contact (recherche d'éléments voisins).
77. **Version 6.100** 17 nov 2006 : Amélioration sur la loi de comportement hyper favier 3D. Dans le cas où dans le calcul du potentiel, le cosinus hyperbolique devient infini, il y a un traitement particulier pour éviter un dépassement de capacité.
78. **Version 6.101** 17 nov 2006 : Correction d'un bug sur l'entrée de donnée pour les forces suivieuses surfaciques.
79. **Version 6.102** 21 nov 2006 : Traitement de nouveaux cas particuliers pour l'hystérésis.
80. **Version 6.103** 23 nov 2006 : Correction d'un bug sur la lecture de plusieurs maillages.
81. **Version 6.200** 26 nov 2006 : Définition de conteneurs globaux pour les grandeurs aux points d'intégration.
82. **Version 6.201** 2 dec 2006 : Introduction des trajets neutres dans la loi d'élastohystérésis.
83. **Version 6.210** 4 dec 2006 : Introduction d'une automatisation du calcul du facteur de pénalisation pour le contact, et introduction du calcul du module de compressibilité et de cisaillement pour les lois en général. Cependant pour l'instant, ce calcul n'est valide que pour l'élasticité de Hooke.
84. **Version 6.3** fin dec 2006 : Introduction de la possibilité de visualiser aux noeuds les mêmes grandeurs que celles demandées en sortie aux points d'intégration. Introduction d'une loi hypo-élastique en contraintes planes.
85. **Versions 6.301-6.302** 4-5 janvier 2007 : Correction de bugs sur le transfert des informations aux noeuds dans le cas de plusieurs maillages.
86. **Versions 6.310-6.311** -- > 12 février 2007 : Mise en place de la création de maillage quadratique à partir d'un maillage linéaire ceci bien qu'implanté globalement, n'est activé pour l'instant qu'uniquement pour les hexaèdres. Pour l'instant les références ne sont pas concernées. Correction d'un bug dans le constructeur de copie de l'élément générique. Amélioration de l'affichage d'infos dans le cas d'un blocage relatif : qui ne peut fonctionner qu'avec des courbes de charge. Mise en place d'une procédure de capture d'erreur dans la recherche de racine triple qui conduisait certaine fois à des nan. Amélioration des infos d'erreur d'entrée de données pour la lecture des courbes poly-linéaires.
87. **Version 6.320** 14 février 2007 : Mise en place de rotations solides pour les conditions limites d'une manière identique aux mouvements solides sur les maillages. Ajout de la possibilité de définir une suite de nouveaux centres de rotation.
88. **Version 6.321** 15 février 2007 : Mise en place du calcul des torseurs de réaction, pour chaque référence de conditions limites contenant un déplacement (ou plusieurs) bloqué. Sortie des torseurs de réaction dans le fichier .reac, et mise en place d'une sortie dans les .maple.
89. **Version 6.322** 21 février 2007 : Changement du mot clé de fin de fichier du .info, et correction d'un bug de lecture dans la loi des mélanges.
90. **Version 6.330** 23 février 2007 : Mise en place d'une seconde loi des mélanges fonctionnant sur les accroissement de contraintes.
91. **Version 6.340** 2 mars 2007 : Mise en place de l'algorithme de relaxation dynamique de Barnes.
92. **Versions 6.341 - 6.342** 8 mars 2007 : Correction de différents petits bugs. Correction d'un gros bug sur la loi ISOHYPER3DFAVIER3 (suite à une fausse manip, il y avait une partie de loi qui ne se calculait plus). Introduction d'une nouvelle courbe 1D : F_CYCLIQUE. Introduction de la possibilité de changer de nom de fichier .CVisu, à la fin du calcul.
93. **Version 6.343** 9 mars 2007 : correction d'un bug sur le calcul des énergies sur tous les algos. Mise en place dans l'algo non dyna implicite, de la possibilité de divergence avec restart automatique à plusieurs incréments de distance de l'incrément actuel (phase exploratoire).
94. **Version 6.344** 13 mars 2007 : correction d'un bug sur la lecture des mouvements solides avant le calcul (sur les maillages).
95. **Version 6.345** 16 mars 2007 : mise en place dans stamm de la génération de maillage quadratiques complets, et tests de traction et de flexion simple.
96. **Version 6.350** 19 mars 2007 : mise en place d'outils de recherche et de définitions interactive de références : de noeuds, de faces, d'éléments, de points d'intégrations. Introduction de ce dernier type. Prise en compte dans les sorties .maple des références de points d'intégration.
97. **Version 6.360** 28 mars 2007 : mise en place de la possibilité d'utiliser des conditions linéaires limites.
98. **Version 6.361** 29 mars 2007 : mise en place de la possibilité d'ordonner les listes de noeuds ou de points d'intégrations selon leurs projection sur une droite. Correction de bug sur la sortie en animation en .maple, et amélioration de la sortie avec de nouvelles possibilités.
99. **Version 6.370** 20 avril 2007 : mise en place de la dépendance à la phase dans la loi hyper-élastique de Laurent Orgéas.
100. **Version 6.371** 23 avril 2007 : introduction du calcul et de la récupération de l'énergie élastique pour les lois hyper grenobloises. Introduction d'une dépendance à la température du paramètre Qs pour la loi de Laurent Orgéas : via une fonction proposée par Laurent.
101. **Version 6.372** 27 avril 2007 : introduction dans la sortie .maple de la possibilité d'avoir les ddl également à 0 ou leur variation de 0 à t. Introduction d'une dépendance à la température du paramètre Qs (pour la loi de Laurent Orgéas) via une fonction d'évolution quelconque comme pour les autres lois.
102. **Versions 6.373-374** 3-6 mai 2007 : modification et amélioration de la relaxation dynamique. Introduction d'une loi cyclique additive (en plus de la loi cyclique multiplicative existante).
103. **Version 6.375** 9 mai 2007 : Correction d'un bug sur la création interactive de face et d'arêtes. Mise en place d'une loi polynomiale hyper-élastique dépendant des invariants utilisés par Mooney-Rivlin. Introduction des colonnes dans la description des grandeurs en sortie dans le .maple.
104. **Version 6.376** 15 mai 2007 : Correction d'un bug sur la méthode ordonnant les pt d'integ, dans le cas d'un calcul d'une liste de pt d'integ. Amélioration des fonctions cycliques de charge. Introduction d'une courbe d'union de plusieurs fonctions, chacune étant défini sur un seul intervalle. Amélioration de la sortie des torseurs de réaction : dans le cas où il y a plusieurs conditions (disjointes) sur une même référence, il y a cumul automatique dans un seul torseur, utile pour les sorties. Correction d'un Bug (ou amélioration) : possibilité de mettre plusieurs ddl limite sur une même ligne, même s'il y a un temps mini temps max etc..
105. **Version 6.377** 16 mai 2007 : Mise au point de la loi hyper-élastique polynomiale (sur des cas simples).
106. **Version 6.378** 21 mai 2007 : Introduction d'un utilitaire pour vérifier l'orientation des éléments et créer des éléments à jacobien positif si pb. Amélioration du modèle d'hystérésis : calcul d'énergie, angle de phase.
107. **Versions 6.379 et 6.380** 11 juin 2007 : Correction d'un bug sur le post-traitement, puis d'un second sur une méthode qui détermine si un point est interne à un élément ou pas. Avancement sur la nouvelle implantation des éléments SFE.
108. **Version 6.381** 19 juin 2007 : Correction de Bugs... fin de la mise en place des éléments SFE et premier tests sur 2 éléments ! Mise en place d'un utilitaire permettant de construire un maillage sfe à partir d'un maillage triangulaire.
109. **Version 6.382** 21 juin 2007 : Test d'éléments SFEL sur petit, moyen et grand maillage. Modification du post-traitement pour l'adapter aux sfel. Pour l'instant uniquement la déformée est opérationnelle. Mise à jour de la doc d'utilisation, au niveau des sfel et des éléments volumiques hexaédriques.
110. **Version 6.383** 22 juin 2007 : Fin de la mise à jour de la doc pour les éléments volumiques. Introduction d'un décalage possible en x et y pour les fonctions poly-linéaires.
111. **Version 6.384** 28 juin 2007 : Remplacement du calcul de courbure complexe (car le modèle n'est pas bon : cf doc théorique), par le plus modèle originale le plus simple : somme des normales aux milieux des arêtes. Tests quantitatif sur une poutre en flexion, ok.

112. **Version 6.385** 5 juillet 2007 : Première implantation des éléments SFE2, avec validation sur deux cas tests (idem SFE1).
113. **Version 6.386** 11 juillet 2007 : Correction d'un petit bug sur la création des maillages SFE2. Pas mal de petites modifs et amélioration sur le calcul Hyper-élastique des lois "Grenobloises" Favier, Orgéas.
114. **Version 6.387** 15 juillet 2007 : Introduction des éléments SFE3 pour lesquels la courbure est calculée à partir d'un polynôme quadratique en θ et θ^2 .
115. **Version 6.388** 13 septembre 2007 : introduction d'une nouvelle méthode de calcul des valeurs propres dans le cas 3D (l'ancienne conduisait à des nan dans certains cas).
116. **Version 6.389** 14 septembre 2007 : correction d'un bug sur la sortie des sfe (le repère local orthonormé était dans certains cas mal calculé)
117. **Version 6.390** 17 septembre 2007 : passage de 1 à 2 pt d'integ pour le chargement linéique sur des triangles axi linéaires. Correction d'un bug sur les éléments pentaedriques quadratiques complet. Mise en place d'un critère d'arrêt sur le résidu pour l'algo de relaxation dynamique.
118. **Version 6.391** 11 octobre 2007 : introduction des conditions de symétrie et d'encastrement pour les éléments SFE3.
119. **Version 6.392** 16 octobre 2007 : correction d'un bug sur le post-traitement des grandeurs internes à la loi de Tait. Introduction d'une première version d'accélération de convergence pour l'algorithme non dynamique.
120. **Version 6.393** 16 octobre 2007 : correction d'un bug sur la sortie des grandeurs aux noeuds dans le cas de plusieurs références de noeuds. Modif des indicateurs de post-traitement pour la loi de Tait.
121. **Version 6.394** 22 oct 2007 : correction d'un bug sur le post-traitement dans Gid, dans le cas de plusieurs types différents d'éléments.
122. **Version 6.395** 27 oct 2007 : première modification des class BaseB, BaseH, Coordonnee pour Tuner, et correction d'un bug sur la loi de mélange, intégration d'un nouveau type de sortie particulier.
123. **Version 6.396** 22 nov 2007 : mise en place d'un programme perl pour le transfert de maillage créé par gmsh dans le format herezh++. Dans herezh++, intégration d'un algorithme permettant de supprimer les noeuds non référencés par les éléments, et dans algorithme d'optimisation de largeur de bande.
124. **Version 6.397** 17 décembre 2007 : introduction du potentiel hyper-élastique de Hart Smith 3D. Introduction d'un nouvel algorithme pour le calcul de la matrice tangente pour la loi d'hystérésis dans le cas du calcul de la contrainte avec un algo de Runge-Kutta imbriqué. Mise en place d'un facteur permettant de moduler la prédiction par extrapolation linéaire d'un incrément à l'autre, pour l'algorithme global statique.
125. **Version 6.398** 21 décembre 2007 : modification de l'introduction (à la lecture) des facteurs de pilotage du calcul de l'hystérésis. Introduction d'un facteur permettant de prendre ou non en compte les variations de w' et w'' dans le calcul de la matrice tangente. Amélioration de la doc concernant l'hystérésis 3D.
126. **Version 6.399** 5 janvier 2008 : Sortie isovaleur Gid, mise en place de la possibilité de visualiser des isovaleurs de grandeurs évoluées (vecteur, coordonnées, tenseurs) et également des isovaleurs de grandeurs particulières aux éléments : ex : pour une loi additive, chaque contribution de contrainte. La visualisation fonctionne soit via les grandeurs aux pt d'intégration, soit via le transfert des pt d'integ aux noeuds. Correction d'un bug sur la sortie maple de grandeurs quelconques.
127. **Version 6.400 - 6.401** Correction d'un bug sur la sortie des valeurs propres en 3D, d'un bug sur la sortie des scalaires pour les grandeurs évoluées.
128. **Version 6.402** 8 février 2008 : mise en place de la possibilité d'utiliser le bulk viscosity quelque soit le signe de la trace de la vitesse de déformation.
129. **Version 6.403** 13 février 2008 : introduction des éléments Sfe3C, et amélioration de différents calculs sur les sfe et sur le passage calcul glob locale des tenseurs non-symétriques. Introduction d'un pilotage de la diminution de l'incrément de temps lorsque la convergence est difficile.
130. **Version 6.410** 2 mars 2008 : introduction d'une sortie fichier pour un pos-traitement avec le logiciel gmsh.
131. **Version 6.411** 5 mars 2008 : Prise en compte dans la loi des mélanges, de la cristallinité directement aux pt d'integ (et non via l'interpolation aux noeuds).
132. **Version 6.412** 12 mars 2008 : Amélioration de la sortie Gmsh : création d'un répertoire a doc, et sortie d'un fichier par grandeur.
133. **Version 6.413** 14 mars 2008 : Fin de la mise en place du calcul interne (au choix) de la cristallinité avec le modèle d'Hoffman.
134. **Version 6.414** 19 mars 2008 : Correction d'un bug sur le pilotage de la convergence, et ajout d'une nouvelle norme de convergence.
135. **Version 6.415** 25 mars 2008 : Correction d'un bug sur la sortie gmsh des éléments sfe, et d'un bug sur le calcul de la raideur des sfe.
136. **Version 6.416** 26 mars 2008 : Correction d'un bug sur la sortie des grandeurs physiques (cristallinité par exemple).
137. **Version 6.417** 28 mars 2008 : Correction d'un bug sur gmsh. Introduction des éléments SFE3_cm4pti et SFE3_cm6pti.
138. **Version 6.418** 31 mars 2008 : Intégration dans l'hystérésis 3D, de la possibilité d'utiliser une loi de dépendance à la phase, quelconque (choisit dans les courbes 1D disponibles). Intégration dans la loi de Maxwell 3D, de la prise en compte de la cristallinité.
139. **Version 6.419** 7 avril 2008 : Mise en place des courbes : COURBE.TRIPODECOS3PHI COURBE.SIXPODECOS3PHI COURBE.EXPON COURBE.EXPO2.N. Mise en place des lois hyper élastiques ISOHYPER3DORGEAS2 et ISOHYPERBULK3. Introduction du paramètre mini.Qsig_pour_phase.sigma_Oi.tdt_ dans le contrôle du calcul.
140. **Version 6.420** 29 avril 2008 : Mise en place de sortie d'informations relatives aux courbures des éléments coques SFE, en maple, gid et gmsh.
141. **Version 6.421** 14 mai 2008 : Correction d'un bug sur la redéfinition des références après une optimisation de largeur de bande.
142. **Version 6.422** 19 mai 2008 : Introduction d'une dépendance à la déformation équivalente pour xn en hystérésis. Correction d'un bug d'affichage pour les SFE.
143. **Version 6.423 - 6.425** 28 mai 2008 : correction de bug. Introduction de conditions linéaires non nulles en statique : refonte de toute la structure de prise en compte des CLL.
144. **Version 6.426** 4 juin 2008 : introduction d'une loi Hoffman2.
145. **Version 6.427-6.428** 9 juin 2008 : Correction d'un bug associé au bulk viscosity. Dans le cas DFC, le type 3 est maintenant celui par défaut.
146. **Version 6.429** 13 juin 2008 : Introduction du calcul des torseurs de réaction correspondant aux conditions CLL, avec possibilité de les récupérer en pos-traitement.
147. **Version 6.430** 25 juin 2008 : Amélioration du calcul de l'opérateur tangent d'une loi de Maxwell3D dans le cas où il n'y a pas de viscosité sur la partie sphérique. Correction d'un Bug sur la loi hypo, qui apparaît que dans un cas particulier. Introduction de fonctions de charges type cos et sin avec bornes et translation linéaire.
148. **Version 6.431** 2 juillet 2008 : Intégration de la possibilité d'utiliser des lois hyper Grenobloise dans des Umat. Correction d'un bug sur le calcul de l'énergie élastique pour les lois Hypo-élastiques.
149. **Version 6.440** 1 octobre 2008 : intégration de la prise en compte d'un ddl d'épaisseur aux noeuds au niveau des éléments SFE. Correction d'un bug au niveau de la sortie des déplacements gmsh. Intégration de l'élément TriaSfe3_3D, pour l'instant en test (ne converge pas bien).
150. **Version 6.441** 16 octobre 2008 : correction d'un bug sur la re-numérotation globale, quand elle est appelée via l'algorithme "utilitaire". Amélioration de la prise en compte de toutes les conditions linéaires.
151. **Version 6.442** 9 décembre 2008 : Intégration dans le cas de chargement "type3" du fait que contrairement au temps mini des fonctions par exemple, dès t=0, la fonction de charge est opérationnelle (application immédiate du chargement). Début de l'introduction de la phase au niveau du paramètre mul dans la loi de orgéas1.

152. **Version 6.450** 25 janvier 2009 : fin intégration de la dépendance à la phase de mul pour la loi hyper-élastique d'orgeas1. Tests vérifications ok. Amélioration de la vitesse de lecture sur fichier, et mise en place de la possibilité d'avoir un ordre sur plusieurs lignes.
153. **Version 6.451** 31 janvier 2009 : Intégration de la possibilité de sortir en gmsh, des variations de ddl en plus de la valeur des ddl. Correction d'un bug sur l'assemblage des conditions linéaires pour les matrices bandes.
154. **Version 6.452-6.454** 22 mars 2009 : Correction de différents petits bugs. Introduction de la possibilité d'avoir une dépendance quelconque de E et mu dans une loi de Maxwell, en fonction de l'intensité du déviateur des vitesses de déformation : implantation fini, reste la mise au point.
155. **Version 6.455** 25 mars 2009 : Amélioration de la présentation de l'entête des fichiers .maple (numéro de noeud, d'élément, nom de référence ...).
156. **Version 6.456** 1 avril 2009 : Correction d'un bug sur la suppression des noeuds non référencés. Correction d'un bug sur la visualisation de plusieurs maillage avec gmsh.
157. **Version 6.457** 4 avril 2009 : Mise en place de la possibilité de ne rien sortir comme résultat automatique.
158. **Version 6.458** 2 mai 2009 : Mise en place de la possibilité de création automatique à la suite de la définition de maillage, de références de noeuds, éléments, faces et arêtes de frontière.
159. **Version 6.459** 12 mai 2009 : Correction d'un bug sur la génération interne d'éléments frontière. Ajout d'une référence automatique sur les noeuds des arêtes des éléments frontière. Intégration d'un écrouissage isotrope pour l'hystérésis 3D.
160. **Version 6.460 - 61** 17 mai 2009 : Correction bug sur l'écrouissage + mise en place d'une vérification automatique de la cohérence des références de noeuds, éléments, faces, arêtes. Amélioration du potentiel d'Hart-Smith par la mise en place d'une partie courbure au potentiel.
161. **Version 6.462 - 65** 7 juin 2009 : Correction de bugs. Début de la traduction des I/O en anglais, tout en gardant la partie française.
162. **Version 6.466** 21 juin 2009 : Introduction du contact dans l'algo de Tchamwa, introduction du calcul de la déformation équivalente cumulée.
163. **Version 6.467** 23 juin 2009 : Correction de bugs sur la lecture des paramètres de courbure pour la loi de Hart Smith modifiée
164. **Version 6.468 - 72** 17 juillet 2009 : Correction d'un bug sur la loi des mélanges. Introduction de la possibilité de piloter le mélange par la deformation cumulée. En hyperélasticité Orgeas, possibilité d'une dépendance à la température de Q_e . Introduction de la prise en compte d'une hystérésis dépendante de la température. Lecture de fichier .info contenant que des carriage return. Introduction d'une courbe TangenteHyperbolique.
165. **Version 6.473 - 76** 10 octobre 2009 : généralisation du contact aux cas généraux de statique. Premiers debuggages pour le contact 2D dans ce contexte.
166. **Version 6.477** 15 octobre 2009 : Introduction de la possibilité de sauvegarder au fil du calcul le dernier incrément uniquement, et sortie par défaut de ce dernier incrément. Mise en place de 2 normes de convergence supplémentaires : une sur l'énergie cinétique relativement aux autres énergies, et une sur le bilan puissance relativement aux puissances élémentaires. Mise en place pour tous les algo dynamiques classiques, de la possibilité de s'arrêter à convergence sur le cas statique.
167. **Version 6.478** 19 octobre 2009 : Introduction de la possibilité du calcul du volume entre un élément de surface et les plans de base. Idem pour toute une surface. Sortie possible en .maple et isovaleurs.
168. **Version 6.480** 12 novembre 2009 : passage de toutes les sources en UTF8. Mise en place d'un Makefile qui permet une compilation en ligne sur osX. Idem sur Linux, avec toutes les bibliothèques ad hoc.
169. **Version 6.481** 20 novembre 2009 : introduction de la possibilité de piloter la loi des mélanges avec la partie sphérique de la déformation.
170. **Version 6.482** 24 novembre 2009 : possibilité de visualiser toutes les contributions de lois élémentaires quelque soit la complexité du montage.
171. **Version 6.483** 28 novembre 2009 : introduction de la possibilité de piloter la loi des mélanges par la partie sphérique de la contrainte.
172. **Version 6.484** 18 décembre 2009 : introduction d'un élément pentaédrique à 1 point d'intégration. Correction d'un bug sur le pilotage en implicite.
173. **Version 6.485** 27 janvier 2010 : mise en place des messages en anglais pour la classe LesReferences, correction de bug sur l'algo de changement de numérotation pour éviter les jacobiens négatifs.
174. **Version 6.486-7** 14 mars 2010 : mise à jour de la gestion des Umat, introduction des sauvegardes sur .BI, préparation aux opérations de restart.
175. **Version 6.488-90** 10 avril 2010 : introduction de nombreuses améliorations sur le contact. Mise en place d'une dépendance du module d'Young et ou de la viscosité à la déformation au sens de Mises, dans la loi 3D de Maxwell.
176. **Version 6.491** 11 avril 2010 : introduction de messages de commentaires en anglais pour les classes gérants les références.
177. **Version 6.492** 13 avril 2010 : introduction de la prise en compte de la variation d'épaisseur dans le cas d'éléments triangulaires, avec une loi de contrainte plane. Correction d'un bug sur la loi hypo-élastique en contraintes planes.
178. **Version 6.493** 14 avril 2010 : correction d'un pb de décalage d'un pas de temps sur les sortie en .BI
179. **Version 6.494** 5 mai 2010 : correction d'un bug sur la prise en compte du temps courant et du delta t dans le cas du calcul d'une loi Umat, ces grandeurs sont utilisées en particulier avec les lois visco-élastiques.
180. **Version 6.495 - 6.499** correction de bugs. Amélioration du contact en explicite. Introduction du contact en relaxation dynamique.
181. **Version 6.500 - 6.510** 26 juin 2010 : première mise en place de conditions linéaires en dynamique explicite : pour Tchamwa, DFC, et relaxation dynamique.
182. **Version 6.511 - 6.514** 10 juillet 2010 : correction de bug sur les conditions linéaires en dynamique. Introduction d'éléments à 1 points d'intégration, hexaédriques, quadratiques complets et incomplets
183. **Version 6.515** 13 octobre 2010 : Introduction de la gestion de mode d'hourglass via un calcul avec intégration exacte et une loi de comportement simplifiée.
184. **Version 6.516** 18 octobre 2010 : Extension de la gestion de mode d'hourglass pour les tétraèdres et pour les pentaèdres.
185. **Version 6.517** 23 octobre 2010 : Relaxation dynamique : introduction d'une nouvelle méthode pour le calcul des masses, à partir de la raideur réelle.
186. **Version 6.518-21** 5 décembre 2010 : correction de bug en particulier sur le contact en pénalisation, fonction très correctement sur des cas d'école.
187. **Version 6.522-23** Mise à jour du test de définition de l'épaisseur pour les triangles et quadrangles.
188. **Version 6.523-28** Correction de bugs sur les conditions linéaires et sur la sortie de références via l'utilitaire.
189. **Version 6.529** Correction bug sur une mise à jour des Umats.
190. **Version 6.530-6.532** 3 mai 2011 : Correction de bug et amélioration de la prise en compte de conditions linéaires en entrée et pour le contact. Intégration des blocages de modes d'hourglass sur les éléments 2D sauf sfé.
191. **Version 6.533-6.534** 14 mai 2011 : Mise en place de la relaxation dynamique avec amortissement visqueux. Correction de bug, dans le cas d'un restart en relaxation dynamique, DFC et Tchamwa.
192. **Version 6.535** 4 juin 2011 : Réorganisation complète des paramètres de gestion de l'algorithme de relaxation dynamique avec amortissement visqueux ou cinétique. Introduction d'une nouvelle syntaxe (et plus de possibilités), qui est incompatible avec l'ancienne syntaxe !
193. **Version 6.536-9** 24 juin 2011 : Nouvelle amélioration de la gestion de tous les types de relaxation. Def de par globaux à tous les algos : viscosité critique, mode debug. Introduction de la possibilité d'un critère d'arrêt uniquement sur le résidu hors viscosité numérique.

194. **Version 6.540-43** 20 septembre 2011 : Correction du petits bugs, dont un sur l'utilisation de matrice masse consistante.
195. **Version 6.544** 22 septembre 2011 : Amélioration de la prise en compte du contact dans les algorithmes de relaxation dynamique. Amélioration du contact au niveau de la gestion du décollement.
196. **Version 6.545** 28 septembre 2011 : Correction d'un bug sur le contact avec présence d'interpolation quadratique.
197. **Version 6.546** 10 octobre 2011. Introduction du contact entre membrane et membrane, puis première implantation de l'auto-contact.
198. **Version 6.547-8** 17 octobre 2011. Correction de deux bugs.
199. **Version 6.561** 19 janvier 2012 : Introduction du nouveau format de sauvegarde de gmsh. Dans une première étape ce format est optionnel. Cependant il permet d'obtenir des réductions de taille d'un facteur 10!
200. **Version 6.566** 15 mars 2012 : Introduction des déformations planes utilisable pour toutes les lois 3D implantés.
201. **Version 6.576** 25 avril 2012 : Correction de bugs. Ajout de la possibilité d'utiliser des matrices bandes non-symétriques, de la bibliothèque Lapack (avec accélération sur mac)
202. **Version 6.585** 25 septembre 2012 : Mise en place d'un algorithme d'orientation automatique pour un maillage de membrane-plaque-coque.
203. **Version 6.586** 9 octobre 2012 : Mise en place d'une méthode pour fusionner des noeuds très voisins.
204. **Version 6.587-9** 6 novembre 2012 : Correction de bug.
205. **Version 6.590** 9 novembre 2012 : correction d'un bug empêchant la création automatique des ref de frontières surfaces. Correction d'un bug sur la mise à jour des métriques pour les éléments SFE : empêchait un fonctionnement normal avec des lois de comp incrémentales.
206. **Version 6.591-4** 24 novembre 2012 : Intégration pour les éléments SFE3, d'une intégration de Gauss-Lobatto dans l'épaisseur : 3,5,7,13pti, et intégration de 12pti en Gauss. Mise en place d'une méthode automatique pour prendre en compte une loi de contrainte plane englobant une loi 3D quelconque, ceci par une méthode de perturbation. Correction de bug, pour le chargement axi-symétrique linéique et linéique suiveur. Mise en place d'un chargement de type pression pour les élément axisymétriques.
207. **Version 6.595** 28 novembre : intégration de la possibilité de sortir graphiquement sous gmsh, les références sous forme d'un fichier par référence. L'option n'est utile que pour de très grands maillages ou un grand nombre de références. Correction d'un bug sur la libération de mémoire concernant les matrices MatLapack.
208. **Version 6.596-8** 3 décembre 2012 : Amélioration de la prise en compte d'une sortie d'un nombre restreint de maillages. Correction bugs suite aux changements de la prise en compte des frontières.
209. **Version 6.599-6.600** 2 janvier 2013 : Introduction d'un algorithme de fusion d'éléments superposés, amélioration de l'algorithme d'orientation automatique de facette, ceci au niveau de la génération des références associées ; amélioration de l'algorithme de fusion de noeuds proches.
210. **Version 6.601** 22 janvier 2013 : Correction d'un bug sur l'algorithme d'intégration (in-out) de références pour la création interactive de nouvelles références.
211. **Version 6.602** 23 janvier 2013 : Correction d'un bug sur les sorties .maple.
212. **Version 6.603** 26 janvier 2013 : Ajout d'un algo utilitaires de fusion de maillages. Ajout également des différentes possibilités de fusions, exécutées à la volée pendant la phase de lecture, et intégration d'un ordre arbitraire pour appliquer les raffinements extra maillage.
213. **Version 6.604** 2 février 2013 : sep Bug sur le calcul de la def equi, lorsque très petite. Mise en place du calcul des énergies pour maxwell 2D contraintes planes.

Références

- [Rio et al., 2008] Rio, G., Laurent, H., and Bles, G. (2008). Asynchronous interface between a finite element commercial software abaqus and an academic research code he-rezh++. *Advances in Engineering Software*, 39(12) :1010–1022. (ISSN 0965-9978).