Manuel d'utilisation du logiciel de calcul par éléments finis Herezh++ (version 6.949)

Gérard Rio

IDDN.FR.010.0106078.000.R.P.2006.035.20600

1er août 2020

Table des matières

I Introduction **27**
- 0.1 Présentation . . . 28
- 0.2 Historique . . . 29
- 0.3 Coté développeur . . . 29
- 0.4 Coté utilisateur . . . 30

II Généralités et entête **31**

1 Entrée des données **32**
- 1.1 Choix de la langue . . . 33
- 1.2 Commentaires . . . 33
- 1.3 Ordre sur plusieurs lignes . . . 33
- 1.4 Inclusion de fichier . . . 34

2 Liste exhaustive mots clés **34**

3 Liste des mots clés principaux **35**

4 Déclaration de constantes utilisateurs **37**

5 Déclaration de variables utilisateurs **38**

6 Dimension du problème **44**

7 Niveau de commentaire **44**

III Type de problème traité 45

8 Intro et liste algo **46**

9 Galerkin continu et utilitaires **48**

10 Paramètres communs **50**
10.1 Accélération de convergence 52

11 Méthode de Tchamwa-Wielgoz **52**

12 Différences finies centrées **56**

13 Famille de Newmark **57**

14 Méthode de Chung-Lee **60**

15 Méthode Zhai **61**

16 Méthode De Runge-Kutta **62**

17 Umat **63**
17.1 Cas général d'une loi 3D 63
17.2 Cas d'une loi de contrainte plane 64

18 NB : cond. limites et initiales **73**

19 NB : var. pas de temps **73**

20 Relaxation dynamique **74**
20.1 Algos et amor. cinétique 75
20.2 Algos et amor. visqueux 77
20.3 Relaxation dynamique 77
20.3.1 Amor. cinétique 79
20.3.2 Amor. visqueux 79
20.3.3 Amor. mixte 80
20.3.4 Contrôle masse 84
20.3.5 Extension méth. Barnes 84
20.3.6 Calcul masse avec raideur 88
20.3.7 Contrôle du re-calcul de la masse 90
20.3.8 Cas masse nulle 91
20.3.9 Contrôle matrice viscosité critique 92
20.3.10 Contrôle du paramètre λ 93
20.3.11 Contrôle mode debug 99

20.3.12 Modulation de la précision d'équilibre globale . . . 99
20.3.13 Contact avec relaxation dyn. . . . 100
20.4 Critère arrêt en dyn. expli. . . . 100
20.5 NB Organisation des paramètres . . . 101

21 Utilitaires **102**
21.1 Sauv. maill. en cours . . . 102
21.2 Quadratique incomplet en complet . . . 103
21.3 Reloc. noeud inter quadratique . . . 104
21.4 Sauve maill. .her et .lis . . . 104
21.5 Sup. noeuds non ref. . . . 105
21.6 Renumérotation . . . 105
21.7 Orientation des éléments . . . 106
21.7.1 membranes, plaques et coques . . . 107
21.8 Création d'un maillage SFE . . . 109
21.9 Fusion noeuds voisins . . . 110
21.10 Fusion d'éléments supperposés . . . 110
21.11 Fusion de maillages . . . 111
21.12 Créa. sous-maillage via une ref. . . . 112

22 Informations **113**
22.1 Def. références . . . 113
22.2 Frontières . . . 116
22.3 Calculs géométriques . . . 117

23 Galerkin discontinu **117**
23.1 Algo Bonelli . . . 118

24 Algorithmes combinés **118**

25 Intro. champs valeurs **123**

26 Sous-types de calculs **123**

IV Maillages 126

27 Outils de création de maillage **127**

28 Maillage : syntaxe **128**

29 Mouvements solides **130**

30 Sup. noeuds non ref. **132**

31 Renumérotation **132**

32 Fusion noeuds voisins **134**

33 Fusion d'éléments supperposés **134**

34 Sup élém. 2 noeuds coïncidents **135**

35 Crée ref. noeuds non référencé **136**

36 Fusion de maillages **136**

37 Syntaxe référence **136**
37.1 Ref. auto. frontières . . . 138

38 Eléments disponibles **138**
38.1 POUT BIE1 : élément barre . . . 140
38.2 Barre quadratique . . . 140
38.3 Barre linéaire axi. . . . 141
38.4 Barre quadratique axi. . . . 142
38.5 TRIANGLE LINEAIRE . . . 144
38.6 Triangle quadratique . . . 144
38.7 Triangle cubique . . . 146
38.8 Triangle axi. linéaire . . . 146
38.9 Triangle axi. quadratique . . . 147
38.10SFE1 : élément coque . . . 149
38.11SFE2 : élément coque . . . 151
38.12SFE3 et SFE3C : élément coque . . . 152
38.13SFE3 _cmjpti . . . 153
38.14Quadrangle linéaire . . . 154
38.15Quad. quadratique incomplet . . . 155
38.16Quad. quadratique complet . . . 156
38.17Quad. cubique complet . . . 157
38.18Quad. axi. linéaire . . . 158
38.19Quad. axi. quadratique incomplet . . . 159
38.20Quad. axi. quadratique complet . . . 160
38.21Quad. axi. cubique complet . . . 162
38.22Hexa. linéaires . . . 163
38.23Hexa. quadratique incomplet . . . 163
38.24Hexa. quadratique complet . . . 165
38.25Penta. linéaire . . . 167
38.26Penta. quadratique incomplet . . . 169
38.27Penta. quadratique complet . . . 171
38.28Tétra. linéaire . . . 174
38.29Tétra. quadratique . . . 174

39 Numérotations locales des noeuds **177**
39.1 Géométrie 1D . . . 177
39.2 2D triangulaire . . . 177
39.3 2D quadrangulaire . . . 177
39.4 3D hexaédrique . . . 180
39.5 3D pentaédrique . . . 183
39.6 3D tétraédrique . . . 186

40 Positions des points d'intégrations **188**
40.1 1D . . . 188
40.2 2D tria. . . . 188
40.3 2D quad. . . . 189
40.4 3D hexa. . . . 189
40.5 3D penta. . . . 190
40.6 3D tétra. . . . 190

41 NB : pt. intégration **192**

42 Gestion des modes d'hourglass **192**

43 Stabilisation membrane et biel **194**

44 Var. épaisseur élém. 2D **195**

45 Var. section élém. 1D **196**

V Courbes **198**

46 Utilisation courbe 1D **199**

47 Utilisation fonction nD **199**

VI Lois de comportement **201**

48 Lois de comportement : généralités **202**

49 Type de déformation **203**
49.1 Ex : dilatation thermique . . . 204

50 Niveau de commentaire **206**

51 Liste lois méca-thermo **206**
51.1 Lois iso-élastiques linéaires . . . 209
51.1.1 ISOELAS1D . . . 209
51.1.2 ISOELAS2D_D . . . 210

51.1.3 ISOELAS2D_C . . . 211
51.1.4 ISOELAS . . . 211
51.2 Lois isotropes élastiques non-linéaires . . . 214
51.2.1 ISO_ELAS_ESPO1D . . . 214
51.2.2 ISO_ELAS_ESPO3D . . . 215
51.2.3 ISO_ELAS_SE1D . . . 215
51.3 Lois Hyper-élastiques . . . 218
51.3.1 ISOHYPER3DFAVIER3 . . . 218
51.3.2 ISOHYPERBULK3 . . . 220
51.3.3 ISOHYPERBULK_GENE . . . 223
51.3.4 MOONEY_RIVLIN_1D . . . 226
51.3.5 MOONEY_RIVLIN_3D . . . 228
51.3.6 ISOHYPER3DORGEAS1 . . . 230
51.3.7 ISOHYPER3DORGEAS2 . . . 235
51.3.8 POLY_HYPER3D . . . 236
51.3.9 HART_SMITH3D . . . 238
51.3.10 MAHEO_HYPER . . . 240
51.4 Lois élasto-plastiques. . . . 243
51.4.1 PRANDTL_REUSS1D . . . 243
51.4.2 PRANDTL_REUSS2D_D . . . 244
51.4.3 PRANDTL_REUSS . . . 244
51.5 Lois visco_élastiques isotropes. . . . 245
51.5.1 NEWTON1D . . . 245
51.5.2 NEWTON2D_D . . . 246
51.5.3 NEWTON3D . . . 246
51.5.4 MAXWELL1D . . . 247
51.5.5 MAXWELL2D_C . . . 248
51.5.6 MAXWELL3D . . . 250
51.6 Composition additive . . . 255
51.6.1 LOI_ADDITIVE_EN_SIGMA . . . 255
51.6.2 LOI_DES_MELANGES_EN_SIGMA . . . 262
51.7 3D déformation ou contrainte plane : introduction . . . 272
51.8 3D → déformation plane . . . 272
51.9 3D → contrainte plane . . . 272
51.9.1 Méthode par perturbation (explicite) . . . 273
51.9.2 Méthode de Newton (implicite) . . . 274
51.103D → contrainte plane double . . . 278
51.10.1 Méthode par perturbation (explicite) . . . 278
51.10.2 Méthode de Newton (implicite) . . . 280
51.11 Lois critère . . . 283
51.11.1 Plis sur des membranes . . . 283
51.11.2 Plis sur des biellettes . . . 287
51.11.3 Pilotage loi critère . . . 288
51.11.4 Pilotage de contrôle . . . 291
51.12 Lois d'hystérésis. . . . 293

51.12.1 HYSTERESIS_1D . . . 293
51.12.2 HYSTERESIS_3D . . . 296
51.12.3 HYSTERESIS_BULK . . . 306
51.13 Lois hypo-élastiques . . . 307
51.13.1 HYPO_ELAS3D . . . 308
51.13.2 HYPO_ELAS2D_C . . . 309
51.13.3 HYPO_ELAS1D . . . 312
51.14 Lois qui ne font rien ! . . . 315
51.14.1 LOI_RIEN1D . . . 315
51.14.2 LOI_RIEN2D_C . . . 315
51.14.3 LOI_RIEN2D_D . . . 316
51.14.4 LOI_RIEN3D . . . 316
51.15 Lois élastiques anisotropes . . . 317
51.15.1 ORTHOELA3D . . . 317
51.15.2 Élastique orthotrope entraîné en contraintes planes . . . 320
51.16 Lois hypo-élastiques anisotropes . . . 323
51.16.1 HYPO_ORTHO3D . . . 323
51.17 Lois anisotropes obtenues par une projection d'une loi existante . . . 327
51.17.1 PROJECTION_ANISOTROPE_3D . . . 327
51.18 Loi externe de type Umat . . . 331
51.18.1 Cas d'un état de contraintes planes . . . 339

52 Liste de lois thermo physiques disponibles 340
52.1 Lois isotropes thermiques . . . 340
52.2 Loi simple isotrope 1D 2D 3D . . . 340
52.3 Loi de Tait 1D 2D 3D . . . 341
52.4 Loi d'Hoffman d'évolution de la cristalinité . . . 346
52.5 Loi d'Hoffman2 d'évolution de la cristalinité . . . 346

53 Liste de lois de frottement disponibles 347
53.1 Loi de Coulomb . . . 348

VII Divers stockages 352

54 Généralités 353

55 Liste exhaustive des stockages divers 354

56 Épaisseur 355

57 Section 355

58 Variation de section 356

59 Largeur 356

60 Masse volumique **357**

61 Def. intégration volumique **357**

62 Masses additionnelles **358**

63 Repère d'anisotropie **359**

64 Dilatation thermique **360**

VIII Conditions de contact 361

65 Introduction **362**
65.1 Esclave - maître . . . 362
65.2 Auto-contact . . . 363
65.3 Def. zones contact . . . 364
65.4 Def. zones collages . . . 365
65.5 Para. gestion contact . . . 366
65.6 NB post-traitement . . . 366

IX Types d'efforts 369

66 Chargement **370**
66.1 Exemple : charges ponctuelles . . . 370
66.1.1 Champ de valeurs . . . 371
66.1.2 Avec courbe de charge . . . 372
66.1.3 Avec fonction nD . . . 372
66.1.4 Temps mini-maxi . . . 374
66.1.5 Charge active à t0 . . . 374
66.2 Les types de charge . . . 374
66.2.1 Volumique . . . 375
66.2.2 Pseudo massique . . . 376
66.2.3 Linéique . . . 376
66.2.4 Linéique suiveuse . . . 377
66.2.5 Surfacique fixe . . . 377
66.2.6 Pression . . . 378
66.2.7 Surfacique suiveuse . . . 379
66.2.8 Hydrostatique . . . 379
66.2.9 Aéro hydrodynamique . . . 380
66.2.10 Torseur d'efforts ponctuels . . . 382
66.2.11 Cas axisymétrique . . . 386

X Conditions limites en déplacements 388

67 CL. ddl **389**
67.1 Courbe de charge . . . 390
67.1.1 Fonction nD . . . 391
67.2 Temps mini-maxi . . . 392
67.3 Données-Variables . . . 392
67.4 Champ de valeurs . . . 393
67.5 Initialisation ddl . . . 393
67.6 Mouvement solide . . . 394
67.7 Symétrie et encastrement pour SFE . . . 395
67.7.1 Direction de tangente imposée . . . 395
67.7.2 Encastrement imposé . . . 397
67.8 Axisymétrie . . . 397

68 Conditions limites linéaires (CLL) **397**
68.1 Projection sur plan-droite . . . 398
68.2 Condition linéaire générale . . . 400
68.2.1 NB : noms de maillage . . . 403
68.3 NB : CLL et largeur de bande . . . 404

69 Conditions initiales **405**
69.1 NB : init. position / autres ddl . . . 406
69.2 SFE : CL. symétrie et encastrement . . . 406

XI Chargement global **407**

70 Algorithme de chargement **408**

XII Paramètres de contrôles et de pilotage **412**

71 Résolution : introduction **413**

72 Contrôle général **413**

73 Dynamique **420**

74 Calculs des énergies **423**

75 Résolution des systèmes linéaires **425**

76 Pilotage de la résolution globale **431**

77 Pilotage du contact **435**
77.1 Pilotage d'un paramètre via une fonction nD : . . . 445

78 Affichage des résultats **446**

79 Calculs géométriques **448**

XIII Sortie des résultats 450

80 Sortie des résultats **451**
80.1 Introduction . . . 451
80.2 Sortie sur fichiers . . . 451
80.2.1 Aucune sortie demandée . . . 454
80.3 Sortie pour visu. graphique . . . 455
80.3.1 Utilisation des fichiers de commande . . . 456
80.4 Formats vrml . . . 463
80.5 Format tableaux Maple . . . 465
80.5.1 introduction . . . 465
80.5.2 Cas des grandeurs aux noeuds . . . 467
80.5.3 Ddl aux éléments . . . 470
80.5.4 Grandeurs spécifiques aux élèm. . . . 471
80.5.5 Grandeurs tensorielles aux élèm. . . . 472
80.5.6 Grandeurs aux pti de faces et arêtes. . . . 472
80.5.7 Cas des grandeurs globales . . . 473
80.5.8 Cas des torseurs de réactions . . . 473
80.5.9 Moy. maxi. mini. sur ref. noeuds . . . 474
80.5.10 Moy. maxi. mini. sur ref. élém. . . . 475
80.5.11 Moy. maxi. mini. sur ref. face, arête. . . . 475
80.5.12 Ordre des composantes des tenseurs . . . 475
80.5.13 Ordre de sortie des grandeurs particulières . . . 476
80.6 Formats geomview . . . 476
80.7 Formats Gid . . . 476
80.8 Formats gmsh et comparaison avec le format Gid . . . 477
80.9 Sortie des résultats au fil du calcul . . . 479
80.10 Grandeur aux points d'integ. . . . 480
80.11 Significations des grandeurs disponibles . . . 482
80.11.1 Grandeurs liés à la cinématique . . . 482
80.11.2 Grandeurs liés aux contraintes . . . 482
80.11.3 Grandeurs liées aux énergies . . . 483
80.11.4 Liste des ddl possibles aux noeuds . . . 483
80.12 NB : Contrainte et def. pour élém. 2D . . . 484
80.13 NB : Volume entouré par des élém. 2D . . . 484

XIV Utilitaires 486

81 Utilitaires **487**
81.1 Fonctions 1D . . . 487
81.1.1 Polylinéaire . . . 487
81.1.2 Fonction polylineaire-simple . . . 488

81.1.3 Fonction $f(x) = \gamma + \alpha(|x|^n)$ . . . 489
81.1.4 Fonction $f(x) = \frac{c}{2}(1 - cos(\frac{(x-a)\Pi}{(b-a)}))$ . . . 489
81.1.5 Fonction polynomial $f(x) = \sum_{i=1}^{n} a_i x^i$ . . . 489
81.1.6 Fonctions composées : $f(x) = f1 \ o \ f2(x)$ . . . 490
81.1.7 Fonctions composées : $f(x) = f1(x) \ + \ f2(x)$ . . . 491
81.1.8 Fonctions cycliques : amplification multiplicative . . . 491
81.1.9 Fonctions cycliques : amplification additive . . . 493
81.1.10 Fonctions réunion de domaine . . . 493
81.1.11 Fonctions $f(x) = (1. + \gamma \ cos(3 \ x))^{(-n)}$ . . . 493
81.1.12 Fonctions $f(x) = (1. + \gamma \ (cos(3 \ x))^2)^{(-n)}$ . . . 495
81.1.13 Fonctions $f(x) = (\gamma + \alpha \ x)^n$ . . . 496
81.1.14 Fonctions $f(x) = (\gamma + \alpha \ x^2)^n$ . . . 496
81.1.15 Fonctions $f(x) = (a - b) \ exp(-c \ x) + b$ . . . 496
81.1.16 Fonctions $f(x) = e \ cos(\alpha x + \beta)$ . . . 497
81.1.17 Fonctions $f(x) = e \ sin(\alpha x + \beta)$ . . . 497
81.1.18 Fonction $f(x) = a + b \ tanh((x - c)/d)$ . . . 498
81.1.19 Fonction de type "interpolation d'Hermite" par morceau . . . 499
81.1.20 Fonction littérale : f(x) . . . 499
81.1.21 Fonction littérale : f(x), f'(x), f'(x) . . . 501
81.2 Fonctions nD . . . 504
81.2.1 NB utilisation de variables globales . . . 504
81.2.2 Fonction littérale à n variables . . . 504
81.2.3 Fonction utilisant une courbe 1D . . . 505
81.2.4 Combinaison de fonction nD . . . 507
81.2.5 Fonction nD vectorielle . . . 508
81.2.6 Fonction externe à n variables . . . 511

82 Pondérations par fonctions **519**
82.1 Fct. du temps . . . 519
82.2 Fct. de grandeurs locales . . . 519
82.3 Fct. grandeurs quelconques . . . 520
82.4 Fct. grandeurs globales . . . 520

XV Estimation d'erreur **521**

83 Estimation d'erreur après calcul **522**

84 Calcul d'intégrales de volumes **524**

XVI Interruption et multistep **526**

85 Gestion des interruptions prévues ou non **527**
85.1 Interruptions prévues initialement . . . 527
85.2 Interruptions non prévues initialement . . . 527

86 Multistep **530**

XVII Gestion des temps CPU 532

87 Gestion des temps CPU **533**
87.1 Temps globaux 533
87.2 Temps locaux 534

XVIII Grandeurs 535

88 Grandeurs manipulées **536**
88.1 Grandeurs globales 536
88.2 Degrés de liberté : ddl 538
88.3 Ddl_enum_etendu 538
88.4 Grandeurs dites quelconques 541

89 Grandeurs utilisées pour les pondérations **542**

XIX Chronologie et historique des mises à jour 544

90 Introduction **545**

91 Liste exhaustive **545**

Bibliographie 561

Liste des tableaux

1 liste des mots clés principaux (V. 6.908) . . . 36
2 Exemple générique de déclaration de toutes les variables utilisateur possibles 43
3 liste des type de problèmes . . . 47
4 liste des type de paramètres génériques pouvant être associés aux paramètres particuliers de l'algorithme . . . 50
5 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 . . . 51
6 Exemple de paramètres pour mettre en oeuvre et conduire la méthode d'accélération de convergence . . . 52
7 Exemple de de l'algorithme de Tchamwa avec une fonction de modération sur φ. . . . 53
8 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 . . . 55
9 Exemple de déclaration de la méthode de Newmark avec des paramètres particuliers γ et β . . . 58
10 Exemple de déclaration de la méthode de Newmark avec un amortissement numérique type HHT . . . 59
11 Exemple de déclaration des paramètres de pilotage du Runge-Kutta . . . 63
12 déclaration de la fonction Umat fortran : partie fortran . . . 65
13 déclaration de la fonction Umat fortran : partie en langage C . . . 66
14 entête des routines C appelées par la subroutine fortran . . . 67
15 exemple de l'algorithme permettant d'utiliser Herezh++ en tant qu'Umat . 68
16 exemple de l'algorithme permettant d'utiliser Herezh++ en tant qu'Umat, pour la définition d'une loi thermo-dépendante . . . 69
17 suite exemple de l'algorithme permettant d'utiliser Herezh++ en tant qu'Umat, pour la définition d'une loi thermo-dépendante . . . 70
18 suite exemple de l'algorithme permettant d'utiliser Herezh++ en tant qu'Umat, pour la définition d'une loi thermo-dépendante . . . 71
19 exemple de l'algorithme permettant d'utiliser Herezh++ en tant qu'Umat en contrainte plane . . . 72
20 Exemple de déclaration d'un amortissement cinétique avec l'algorithme de Tchamwa . . . 76
21 Exemple de déclaration d'un amortissement visqueux avec l'algorithme de Tchamwa . . . 77
22 Exemple de déclaration de paramètres de l'algorithme de relaxation dynamique en amortissement mixte avec un paramètre de recalcule de la matrice masse différent en cinétique et en visqueux . . . 79
23 Exemple de déclaration de paramètres de l'algorithme de relaxation dynamique avec amortissement cinétique . . . 80
24 Exemple de déclaration de paramètres de l'algorithme de relaxation dynamique avec amortissement visqueux . . . 81

25 Exemple de déclaration de paramètres de l'algorithme de relaxation dynamique avec amortissement visqueux . . . 82
26 Exemple de déclaration de paramètres de l'algorithme de relaxation dynamique avec amortissement mixte contrôlé par une fonction . . . 83
27 Exemple de déclaration pour l'algorithme de relaxation dynamique avec un contrôle sur le résidu . . . 86
28 Exemple de déclaration pour l'algorithme de relaxation dynamique avec un contrôle sur le déplacement . . . 87
29 Exemple de déclaration d'un algorithme de relaxation dynamique avec utilisation d'un λ qui varie en fonction du temps physique . . . 95
30 Exemple de déclaration d'un algorithme de relaxation dynamique avec utilisation d'un λ qui varie à l'aide d'une fonction nD . . . 96
31 Exemple de déclaration d'un algorithme de relaxation dynamique avec utilisation d'un λ piloté automatiquement . . . 98
32 Exemple de déclaration pour l'algorithme de relaxation dynamique avec utilisation du mode debug . . . 99
33 Exemple de déclaration pour l'algorithme de relaxation dynamique avec un paramètre de gestion du contact . . . 101
34 Exemple d'utilisation de l'algorithme "informations" pour définir de nouvelles références. . . . 114
35 Exemple d'utilisation de l'algorithme de Bonelli. . . . 119
36 Exemple partiel de déclaration d'un algorithme combiner qui intègre 2 sous-algorithmes . . . 120
37 Exemple de mise en données spécifiques pour 2 sous-algorithmes. . . . 122
38 Exemple de positionnement dans le .info, de la lecture sur des fichiers externes, des contraintes aux points d'intégration et de déplacements aux noeuds. . . . 124
39 Liste des différents sous types disponibles . . . 125
40 Exemple de déclaration de deux mouvements solides : une translation suivant z et une rotation autour de l'axe y . . . 131
41 Exemple de déclaration d'un mouvement solide de rotation autour d'un noeud . . . 132
42 Exemple de déclaration d'une homothétie . . . 132
43 Exemple de d'utilisation des méthodes de suppression des noeuds non référencés et de renumérotation . . . 133
44 Exemple de d'utilisation de la méthode de suppression des noeuds voisins d'une distance inférieure à 0.001 . . . 134
45 Exemple de d'utilisation de la méthode de suppression des éléments superposés . . . 135
46 Exemple de d'utilisation de la méthode de suppression des éléments à 2 noeuds de distance inférieure à 1. . . . 135
47 Exemple de l'utilisation successive de deux opérations de fusion de maillage, intégrant des déplacements solides pendant et après les opérations. . . . 137
48 liste d'élément finis mécaniques . . . 139
49 Exemple de déclaration d'un élément biellette . . . 140

50 Exemple de déclaration d'un élément biellette . . . 141
51 Exemple de déclaration d'un élément linéaire biellette axisymétrique 142
52 Exemple de déclaration d'un élément quadratique biellette axisymétrique . 143
53 Exemple de déclaration d'un élément triangle avec interpolation linéaire . . 144
54 Exemple de déclaration d'un élément triangle avec interpolation quadratique 145
55 Exemple de déclaration d'un élément triangle avec interpolation quadratique, et avec 3 points d'intégrations strictement interne à l'élément 145
56 Exemple de déclaration d'un élément triangle avec interpolation quadratique et plusieurs type d'intégration . . . 146
57 Exemple de déclaration de deux éléments triangles axisymétriques avec interpolation linéaire . . . 147
58 Exemple de déclaration d'un élément triangle axisymétriques avec interpolation quadratique et différents nombres de points d'intégration 148
59 numérotation des éléments SFE1 . . . 150
60 Exemple de déclaration d'éléments SFE1 . . . 151
61 Exemple de déclaration d'éléments SFE2 . . . 152
62 Exemple de déclaration d'éléments SFE3 (pour l'élément SFE3C, il suffit de changer SFE3 par SFE3C) . . . 153
63 Exemple de déclaration de deux éléments quadrangles avec interpolation linéaire et plusieurs types de nombres de points d'intégration 154
64 Exemple de déclaration d'un élément quadrangle avec interpolation quadratique incomplete . . . 155
65 Exemple de déclaration d'un élément quadrangle avec interpolation quadratique complete et 2 types de nombres de points d'intégration 157
66 Exemple de déclaration d'un élément quadrangle avec interpolation quadratique complete et plusieurs types de nombres de points d'intégration . . 158
67 Exemple de déclaration de deux éléments quadrangles axisymétriques avec interpolation linéaire et plusieurs types de nombres de points d'intégration 159
68 Exemple de déclaration d'un élément quadrangle axisymétriques avec interpolation quadratique incomplete . . . 160
69 Exemple de déclaration d'un élément quadrangle axisymétriques avec interpolation quadratique complete et plusieurs types de nombres de points d'intégration . . . 161
70 Exemple de déclaration d'un élément quadrangle axisymétriques avec interpolation cubique et plusieurs types de nombres de points d'intégration . 162
71 Exemple de déclaration d'éléments hexaédrique avec interpolation linéaire . 163
72 Exemple de déclaration d'éléments hexaédriques linéaire avec différents nombres de points d'intégration . . . 164
73 Exemple de déclaration d'éléments hexaédrique avec interpolation quadratique incomplète . . . 164
74 Exemple de déclaration d'éléments hexaédriques quadratique incomplet avec différents nombres de points d'intégration . . . 165
75 Exemple de déclaration d'éléments hexaédrique avec interpolation quadratique complète . . . 166

76 Exemple de déclaration d'éléments hexaédriques quadratique avec différents nombres de points d'intégration . . . 167
77 Exemple de déclaration d'éléments pentaédriques avec interpolation linéaire 168
78 Exemple de déclaration d'éléments pentaédriques linéaire avec 6 points d'intégration . . . 168
79 Exemple de déclaration d'éléments pentaédriques avec interpolation quadratique incomplet . . . 169
80 Exemple de déclaration d'éléments pentaédriques quadratique avec différents nombres de points d'intégration . . . 170
81 Exemple de déclaration d'éléments pentaédriques avec interpolation quadratique complet . . . 172
82 Exemple de déclaration d'éléments pentaédriques quadratique complet avec différents nombres de points d'intégration . . . 173
83 Exemple de déclaration d'éléments tétraèdre avec interpolation linéaire . . 174
84 Exemple de déclaration d'éléments tétraèdriques avec interpolation quadratique . . . 175
85 Exemple de déclaration d'éléments tétraèdriques quadratique sous-intégré (un point d'intégration) . . . 176
86 numérotation des éléments 1D de référence . . . 177
87 numérotation des éléments triangulaires de référence . . . 178
88 numérotations des éléments de référence quadrangulaires . . . 179
89 numérotation de l'élément de référence hexaédrique linéaire . . . 180
90 numérotation de l'élément de référence pour les hexaèdres quadratiques incomplet . . . 181
91 numérotation de l'élément de référence pour les hexaèdres quadratiques complets . . . 182
92 numérotation de l'élément pentaédrique linéaire de référence. . . . 183
93 Numérotation de l'élément pentaédrique quadratique incomplet de référence. 184
94 Numérotation de l'élément pentaédrique quadratique complet de référence. 185
95 Numérotation de l'élément tétraédrique linéaire de référence. . . . 186
96 Numérotation de l'élément tétraédrique quadratique de référence. . . . 187
97 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 . . . 190
98 Positions de points d'intégration pour les éléments tétraédriques. . . . 191
99 Exemple de déclaration de contrôle d'hourglass à l'aide d'un comportement matériel simple, et d'un élément interne à intégration complète . . . 193
100 Exemple de déclaration d'une stabilisation transversale d'éléments de membrane à l'aide d'un ratio fixé sur la raideur dans le plan médian de l'élément 195
101 Exemple de déclaration d'une stabilisation transversale d'éléments de membrane à l'aide d'un ratio piloté par une fonction nD ici : la fonction pilotageStab . . . 196
102 Exemple de déclaration d'une liste de courbes 1D. . . . 199
103 Exemple de déclaration d'une liste de fonctions nD. . . . 200

104 Exemple de déclaration d'un nom de loi de comportement associée à une référence, ceci dans le cas d'un maillage. 202
105 Exemple de déclaration d'un nom de loi de comportement associée à une référence, ceci dans le cas de plusieurs maillages. 202
106 Exemple de déclaration de loi de comportement associée à deux matériaux. 203
107 Exemple de déclaration d'une loi de comportement élastique associée à une déformation logarithmique. 204
108 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. 205
109 liste des différentes lois 207
110 suite de la liste des différentes lois 208
111 liste des différentes lois isotropes élastiques disponibles 209
112 Exemple de déclaration de la loi élastique 1D dont le module d'Young dépend de la température selon une courbe indiquée explicitement. 210
113 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. 210
114 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. 212
115 Exemple de déclaration de la loi élastique 3D dont le module d'Young et le coefficient de Poisson dépendent d'une fonction nD. 212
116 liste des différentes lois isotropes élastiques non linéaires disponibles 214
117 Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_ESPO1D . 214
118 Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_ESPO3D . 215
119 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. 216
120 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 216
121 Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_SE1D incluant un coefficient de Poisson non nul. 217
122 liste des différentes lois isotropes hyper-élastiques disponibles 218
123 Exemple de déclaration de la loi hyperélastique ISOHYPER3DFAVIER3 sans phase. 219
124 Exemple de déclaration de la loi hyperélastique ISOHYPER3DFAVIER3 avec phase. 219
125 Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 220
126 Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 thermodépendante. 221
127 Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 dont le module de compressibilité dépend de la variation de volume. 221

128 Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 avec une dépendance à la température et à la variation de volume. Les courbes "courbe1" et "courbe2" doivent avoir été définies au niveau des courbes générales. 221
129 Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 avec une dépendance à la température et à la variation de volume, via des courbes définies dans la loi, ici il s'agit de courbes poly-linéaires. 222
130 Exemple de déclaration de la loi hyperélastique ISOHYPERBULK_GENE, "courbe2" est le nom d'une fonction qui doit avoir été préalablement définie 223
131 Exemple de déclaration de la loi hyperélastique ISOHYPERBULK_GENE via la définition d'une fonction 224
132 Exemple de déclaration de la loi hyperélastique ISOHYPERBULK_GENE avec thermodépendance multiplicative préalablement définie. 224
133 Exemple de déclaration de la loi hyperélastique ISOHYPERBULK_GENE avec thermodépendance via la définition d'une fonction. 224
134 Exemple de déclaration de la loi hyperélastique ISOHYPERBULK_GENE avec thermodépendance via la définition d'une fonction. 225
135 Exemple de déclaration de la loi de Mooney Rivlin en 1D. 226
136 Exemple de déclaration de la loi de Mooney Rivlin en 1D, avec les deux paramètres thermo-dépendants. 227
137 Exemple de déclaration de la loi de Mooney Rivlin en 1D, avec le second paramètre thermo-dépendant, le premier étant fixe. 227
138 Exemple de déclaration de la loi de Mooney Rivlin en 3D. 228
139 Exemple de déclaration de la loi de Mooney Rivlin en 3D, dans le cas où les trois coefficients matériaux sont thermo-dépendants. 229
140 Exemple de déclaration de la loi de Mooney Rivlin en 3D avec un terme de raidissement. 230
141 Exemple de déclaration de la loi de Mooney Rivlin en 3D avec un terme de raidissement qui dépend de la température. 230
142 Exemple de déclaration de la loi d'hyper-élasticité Orgéas 1. 231
143 Exemple de déclaration de la loi d'hyper-élasticité Orgéas 1 avec dépendance à la phase. 232
144 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. 233
145 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. 234
146 Exemple de déclaration de la loi d'hyper-élasticité Orgéas avec dépendance à la phase et une dépendance à la température des paramètres Q_{0s} selon une évolution donnée par la courbe courbe_evol_Q0s., et Q_{0e} selon la loi interne 234
147 Exemple de déclaration de la loi d'hyper-élasticité Orgéas avec dépendance à la phase selon des fonctions courbes 235
148 Exemple de déclaration de la loi polynomiale en 3D. 236

149 Exemple de déclaration de la loi polynomiale en 3D, dans le cas où les trois coefficients matériaux sont thermo-dépendants. 237
150 Exemple de déclaration de la loi de hart-smith3D en 3D. 238
151 Exemple de déclaration de la loi de Hart Smith en 3D, dans le cas où les quatre coefficients matériaux sont thermo-dépendants. 239
152 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. 240
153 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 240
154 Exemple de déclaration de la MAHEO_HYPER. 241
155 Exemple de déclaration de la loi MAHEO_HYPER, dans le cas où les trois coefficients matériaux sont thermo-dépendants. 242
156 liste des différentes lois isotropes élasto-plastiques disponibles 243
157 Exemple de déclaration de la loi élasto-plastique de Prandtl Reuss en 1D . 243
158 liste des différentes lois visco-élastiques isotropes disponibles 245
159 Exemple de déclaration d'une loi de Newton en 1D 245
160 Exemple de déclaration d'une loi de Newton en 1D dont le coefficient mu dépend de la température 246
161 Exemple de déclaration d'une loi de Maxwell en 1D 247
162 Exemple de déclaration d'une loi de Maxwell en 1D dont les coefficients dépendent de la température au travers de courbes définies explicitement . 248
163 Exemple de déclaration d'une loi de Maxwell en 1D dont les coefficients dépendent de la température au travers de courbes définies explicitement . 248
164 Exemple de déclaration d'une loi de Maxwell en 2D contraintes planes . . . 249
165 Exemple de déclaration d'une loi de Maxwell en 3D, dont les paramètres E et mu dépende de la déformation au sens de Mises 251
166 Exemple de déclaration d'une loi de Maxwell en 3D 251
167 Exemple de déclaration d'une loi de Maxwell en 3D avec une viscosité non linéaire 252
168 Exemple de déclaration d'une loi de Maxwell en 3D avec les deux viscosités dépendantes d'une courbe de température "mu1dfT" définie par ailleurs dans le fichier .info 252
169 Exemple de déclaration d'une loi de Maxwell en 3D avec une viscosité dépendante de la cristalinité 253
170 liste des différentes compositions additives disponibles de lois élémentaires 255
171 Exemple de déclaration d'une loi additive en contrainte 255
172 Exemple de déclaration d'une loi additive en contrainte 259
173 Exemple de déclaration d'une fonction littérale qui dépend de 3 pressions . 260
174 Exemple de loi de comportement additive utilisant l'opérateur tangent par rapport à la déformation 261
175 Exemple de déclaration d'une loi des mélanges en contrainte 267

176 Exemple de déclaration d'une loi des mélanges en contrainte avec une proportion sur les accroissements des contraintes (et non sur la contrainte totale) . . . 268
177 Exemple de déclaration d'une loi additive en contrainte, dans le cas où on utilise une proportion directement accessible au point d'intégration (donc non interpolée à partir de grandeurs aux noeuds) . . . 269
178 Liste des différentes grandeurs susceptibles d'être disponibles aux noeuds . 269
179 exemple de loi des mélanges pilotée par la trace des contraintes, la loi de mélange appelle elle-même deux lois additives . . . 270
180 Exemple de loi des mélanges pilotée par la norme de convergence : cas de la nouvelle mise en données . . . 271
181 exemple de loi de déformations planes intégrant un comportement 3D et des conditions de déformations planes . . . 273
182 Exemple de loi de contraintes planes intégrant un comportement 3D et des conditions de contraintes planes. Dans cet exemple, la prise en compte de la condition de contraintes planes s'effectue par perturbation. . . . 275
183 Exemple de loi de contraintes planes intégrant un comportement 3D et des conditions de contraintes planes. Dans cet exemple, la prise en compte de la condition de contraintes planes s'effectue par une méthode de Newton. . 277
184 Exemple de loi de contraintes planes intégrant un comportement 3D et des conditions de contraintes planes doubles. Dans cet exemple, la prise en compte de la condition de contraintes planes s'effectue par perturbation. . 279
185 Exemple de loi de contraintes planes intégrant un comportement 3D et des conditions de contraintes planes doubles. Dans cet exemple, la prise en compte de la condition de contraintes planes double s'effectue avec la méthode de Newton. . . . 282
186 Exemple de déclaration d'une loi critère dans le cas de plissement de membrane. . . . 284
187 Exemple de déclaration d'une loi mélange, comportant une partie loi critère de plissement de membrane avec une définition différente des paramètres de contrôle pour les contraintes planes et les contraintes doublement planes 286
188 Exemple de déclaration d'une loi critère dans le cas de plissement de biellette. 287
189 liste des différentes lois d'élasto-hystérésis . . . 293
190 Exemple de déclaration d'une loi d'hystérésis 1D . . . 295
191 Exemple de déclaration d'une loi d'hystérésis 1D avec paramètres de réglage pour une méthode de résolution de l'équation constitutive par linéarisation 295
192 Exemple de déclaration d'une loi d'hystérésis 1D avec paramètres de réglage pour une méthode de résolution Runge-Kutta . . . 295
193 valeur par défaut des paramètres de réglage du calcul de l'hystérésis 3D . . 301
194 Exemple de déclaration d'une loi d'hystérésis 3D . . . 302
195 Exemple de déclaration d'une loi d'hystérésis 3D avec des paramètres de contrôle de l'algorithme de résolution . . . 302
196 Exemple de déclaration d'une loi d'hystérésis 3D avec des paramètres matériau thermo-dépendants . . . 303

197 Exemple de déclaration d'une loi d'hystérésis 3D avec un premier paramètre matériau fixe puis les autres thermo-dépendants . . . 303
198 Exemple de déclaration d'une loi d'hystérésis 3D avec paramètres de réglage pour une méthode de résolution Runge-Kutta . . . 303
199 Exemple de déclaration d'une loi d'hystérésis 3D avec dépendance à la phase 304
200 Exemple de déclaration d'une loi d'hystérésis bulk . . . 308
201 liste des différentes lois isotropes élastiques disponibles . . . 308
202 Exemple de déclaration de la loi hypo-élastique linéaire 3D. . . . 309
203 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. . . . 310
204 Exemple de déclaration d'une loi hypo-élastique 3D dont la compressibilité provient d'une loi thermo-physique . . . 311
205 Exemple de déclaration d'une loi hypo-élastique 2D en contraintes planes . 311
206 Exemple de déclaration d'une loi hypo élastique 1D . . . 312
207 liste des différentes lois qui ne font rien, disponibles . . . 315
208 Exemple de déclaration d'une loi 1D ne faisant rien mécaniquement. . . . 315
209 Exemple de déclaration d'une loi 2D en contrainte plane ne faisant rien mécaniquement. . . . 315
210 Exemple de déclaration d'une loi 2D en déformation plane ne faisant rien mécaniquement. . . . 316
211 Exemple de déclaration d'une loi 3D ne faisant rien mécaniquement. . . . 316
212 Exemple de déclaration d'une loi 3D orthotrope élastique entraînée. Cas basique utilisant des paramètres matériels constants. . . . 318
213 Exemple de déclaration d'une loi 3D orthotrope élastique entraînée. Cas où certains paramètres matériels sont définis à partir de fonctions nD, avec utilisation de certains paramètres de réglage. . . . 319
214 Exemple de déclaration d'une loi 2D contrainte plane orthotrope élastique entraînée. Cas basique utilisant des paramètres matériels constants. . . . 321
215 Exemple de déclaration d'une loi 2D CP orthotrope élastique entraînée. Cas où certains paramètres matériels sont définis à partir de fonctions nD, avec utilisation de certains paramètres de réglage. . . . 322
216 Exemple de déclaration d'une loi 3D hypo-élastique orthotrope entraînée. Cas basique utilisant des paramètres matériels constants. . . . 325
217 Exemple de déclaration d'une loi 3D hypo-élastique orthotrope entraînée. Cas où certains paramètres matériels sont définis à partir de fonctions nD, avec utilisation de certains paramètres de réglage. . . . 326
218 Exemple de déclaration d'une loi de projection anisotrope 3D qui s'appuie sur un comportement initialement élastique. . . . 328
219 Exemple de déclaration d'une loi de projection anisotrope 3D à coefficients variables. Cas où certains paramètres matériels sont définis à partir de fonctions nD, avec utilisation de certains paramètres de réglage. . . . 329
220 Exemple de déclaration d'une loi Umat utilisable par Abaqus. . . . 332
221 Exemple de déclaration d'une loi Umat utilisé en interne, pour vérification. 332
222 Exemple de déclaration d'une loi Umat avec redéfinition des pipes. . . . 332

223 Définition du stockage des informations d'entrée-sortie pour la création d'une Umat externe au programme Herezh 333
224 procédure d'écriture sur le pipe, dans le cas de la création d'une Umat externe à Herezh 334
225 procédure d'écriture sur le pipe, dans le cas de la création d'une Umat externe à Herezh 335
226 procédure d'écriture sur le pipe, dans le cas de la création d'une Umat externe à Herezh 336
227 procédure d'écriture sur le pipe, dans le cas de la création d'une Umat externe à Herezh 337
228 procédure d'écriture sur le pipe, dans le cas de la création d'une Umat externe à Herezh 338
229 Cas d'un état de contraintes planes : exemple d'utilisation par Herezh d'une Umat extérieure 339
230 liste des lois de comportement thermophysique 340
231 Exemple de déclaration d'une loi thermo physique simple dont les coefficients sont constants. 340
232 Exemple de déclaration d'une loi thermo physique simple dont les coefficients sont thermo dépendants. 341
233 Exemple de déclaration d'une loi thermo physique de Tait. 342
234 Exemple de déclaration d'une loi thermo physique de Tait avec préparation pour le postraitement des résultats. 343
235 Exemple de déclaration d'une loi thermo physique de Tait avec calcul du taux de cristalinité par le modèle d'Hoffman 345
236 Exemple de déclaration d'une loi d'Hoffman permettant le calcul de la cristalinité en fonction de la pression et de la température. 347
237 Exemple de déclaration d'une loi d'Hoffman permettant le calcul de la cristalinité en fonction de la pression et de la température. 348
238 liste des lois disponibles de frottement lié au contact 348
239 Exemple de déclaration d'une loi de frottement classique de Coulomb. . . . 349
240 Exemple de déclaration d'une loi de frottement de Coulomb, avec un coefficient de frottement qui dépend de la vitesse tangentielle. 349
241 Exemple de déclaration d'une loi de frottement de Coulomb régularisée. . . 351
242 exemple de définition d'épaisseur et de section. 353
243 exemple de définition d'épaisseur et de section dans le cas d'un seul maillage. 353
244 exemple de définition d'épaisseur et de section dans le cas de plusieurs maillages. 354
245 Liste exhaustive des stockages divers 354
246 exemple de définition d'une épaisseur variable en fonction d'une fonction nD. 355
247 exemple de définition d'une épaisseur variable en fonction d'une fonction nD. 355
248 exemple de définition de la section et de la variation de section d'un groupe d'éléments. Ici on ne veut pas de variation de section. 356
249 exemple de définition de la largeur d'un groupe d'éléments. 356
250 exemple de définition d'une épaisseur variable en fonction d'une fonction nD. 357
251 exemple de définition de la masse volumique d'un groupe d'éléments. . . . 357

252 exemple de définition d'une masse volumique variable en fonction d'une fonction nD. 357
253 Exemple de définition d'intégrale sur un groupe d'éléments. 358
254 exemple de définition de masse additionnelle sur un groupe de noeud. . . . 359
255 exemple de définition d'un repère d'anisotropie sur un groupe d'éléments. . 360
256 Exemple de mise en place d'un chargement ponctuel dans le cas d'un seul maillage 370
257 Exemple le mise en place d'un chargement ponctuel dans le cas où il existe plusieurs maillages 370
258 Exemple le mise en place d'un chargement ponctuel dans le cas où il existe plusieurs maillages 372
259 Exemple le mise en place d'un chargement ponctuel avec un contrôle via une fonction nD, un temps mini et un temps maxi 373
260 Exemple le mise en place d'un chargement ponctuel avec une courbe de charge, un temps mini et un temps maxi 374
261 liste des différents types de chargement 375
262 Exemple de déclaration d'un chargement volumique 375
263 Exemple de déclaration d'un chargement volumique se rapportant au volume initial 376
264 Exemple de déclaration d'un chargement linéique 376
265 Exemple de déclaration d'un chargement linéique suiveur 377
266 Exemple de déclaration d'un chargement uniforme surfacique 378
267 Exemple de déclaration d'un chargement pression 378
268 Exemple de déclaration d'un chargement de type surfacique directionnelle . 379
269 Exemple de déclaration d'un chargement hydrostatique 380
270 Exemple de déclaration d'un chargement en pression avec une évolution linéaire, sans limitation de position (noter le caractère de continuation l'antislash, pour l'écriture sur 2 lignes) 380
271 Exemple de déclaration d'un chargement en pression avec une évolution linéaire, sans limitation de position et avec l'utilisation de fonctions dépendantes du temps 381
272 Exemple de déclaration d'un chargement hydrodynamique 382
273 Exemple de déclaration d'une fonction nD de pondération pour un torseur d'effort 384
274 Exemple de déclaration simple d'un torseur de charge correspondant à l'application d'une résultante excentrée par rapport au centre du torseur . . . 384
275 Exemple de déclaration d'un torseur de charge dont la résultante dépend d'une fonction nD et qui utilise une fonction de pondération de poids de type sinusoïdal 384
276 Exemple de déclaration d'un torseur de charge dont le moment dépend d'une fonction nD 384
277 Exemple de déclaration des blocages des degrés de liberté dans le cas d'un seul maillage 389
278 Exemple de déclaration des blocages des degrés de liberté dans le cas de plusieurs maillages 390

279 Exemple de déclaration de degrés de liberté imposés suivant une courbe de charge, selon une procédure relative $X(t+\Delta t) = X(t) + U(t+\Delta t) - U(t)$ 390
280 Exemple de déclaration des blocages des degrés de liberté à l'aide d'une fonction nD . . . 391
281 Exemple de déclaration des blocages des degrés de liberté à l'aide d'une fonction nD . . . 392
282 Exemple de déclaration des blocages des degrés de liberté dans le cas d'un champ de valeurs fixes . . . 393
283 Exemple de déclaration des blocages des degrés de liberté dans le cas d'un champ de valeurs déterminées par des courbes de charge . . . 394
284 Exemple de déclaration d'un déplacement imposé par mouvements solides . 395
285 Exemple de déclaration d'un déplacement imposé par mouvements solides . 395
286 Exemple de déclaration d'une direction de tangente imposée à une surface moyenne d'une coque SFE . . . 396
287 Exemple de mise en place de conditions linéaire par projection sur un plan, avec utilisation de fonction de charge et centre fixe . . . 398
288 Exemple de conditions linéaires par projection sur un plan, avec un centre noeud initial et à t . . . 399
289 Exemple de conditions linéaires par projection sur un plan, avec un temps mini et un temps maxi . . . 400
290 Exemple de conditions linéaires entre ddl de plusieurs noeuds . . . 400
291 Exemple de conditions linéaires avec l'utilisation de fonctions de charge . . 402
292 Exemple de conditions linéaires entre noeuds de maillages différents . . . 404
293 Exemple de mise en place de conditions initiales dans le cas d'un seul maillage 405
294 Exemple de mise en place de conditions initiales dans le cas de plusieurs maillages . . . 405
295 liste des algorithmes de chargement . . . 408
296 liste des sous-mots clés associés aux paramètres de contrôle . . . 414
297 liste des sous-mots clés associés aux paramètres de contrôle liés à la dynamique . . . 421
298 liste des sous-mots clés associés aux paramètres de calcul d'énergie . . . 425
299 liste des sous-mots clés associés aux paramètres de contrôle liés à la dynamique . . . 426
300 liste des sous-mots clés associés aux paramètres de contrôle du pilotage . . 432
301 liste des sous-mots clés associés aux paramètres de contrôle liés au contact - frottement . . . 436
302 liste des sous-mots clés associés aux paramètres de contrôle du pilotage . . 447
303 liste des sous-mots clés associés aux paramètres de contrôle du pilotage . . 448
304 Exemple de fichier de sortie de contraintes : suffixe "_cab.isoe ", ici cas 1D 453
305 Exemple de fichier de sortie de contraintes : suffixe "_dpl.points", ici seule les x varient car il s'agit d'un cas 1D . . . 454
306 liste des différentes options de post-traitement graphique, à indiquer après le type de calcul . . . 455
307 Exemple d'en-tête de fichier .maple . . . 469
308 liste des fonctions 1D . . . 487

309 Exemple de déclaration d'une fonction d'écrouissage plastique polylinéaire 488
310 Exemple de déclaration d'une courbe polylinéaire avec translations initiales 489
311 Exemple de déclaration d'une courbe polylinéaire simplifiée 489
312 Exemple de déclaration d'une fonction de type $\sigma = \gamma + \alpha|x|^n$ 489
313 Exemple de déclaration d'une fonction de type $f(x) = \frac{c}{2}(1 - cos(\frac{(x-a)\Pi}{(b-a)}))$. 490
314 Exemple de déclaration d'une fonction polynomiale $f(x) = \sum_{i=1}^{n} a_i x^i$ . . . 490
315 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. 490
316 Exemple de déclaration d'une fonction composée $f(x) = f1of2(x)$. Les 2 fonctions sont définies ici explicitement. 490
317 Exemple de déclaration d'une fonction composée $f(x) = f1(x)+f2(x)$. Les 2 fonctions sont définies par un nom qui est une référence à des fonctions définies auparavant. 491
318 Exemple de déclaration d'une fonction cyclique avec un facteur d'amplification multiplicatif. 492
319 Exemple de déclaration d'une fonction cyclique avec un facteur d'amplification additif. 494
320 Exemple de déclaration d'une fonction union permettant de globaliser plusieurs fonctions déjà définis. 495
321 Exemple de déclaration d'une fonction de type $f(x) = (1. + \gamma\ cos(3\ x))^{(-n)}$ 495
322 Exemple de déclaration d'une fonction de type $f(x) = (1. + \gamma\ (cos(3\ x))^2)^{(-n)}$ 496
323 Exemple de déclaration d'une fonction de type $f(x) = (\gamma + \alpha\ x)^n$ 496
324 Exemple de déclaration d'une fonction de type $f(x) = (\gamma + \alpha\ x^2)^n$ 497
325 Exemple de déclaration d'une fonction de type $f(x) = (a - b)\ exp(-c\ x) + b$ 497
326 Exemple de déclaration d'une fonction de type $f(x) = e\ cos(\alpha x + \beta)$ 498
327 Exemple de déclaration d'une fonction de type $f(x) = e\ sin(\alpha x + \beta)$ . . . 498
328 Exemple de déclaration d'une fonction de type $f(x) = a + b\ tanh((x - c)/d)$ 499
329 Exemple de déclaration d'une fonction de charge 499
330 Exemple de déclaration d'une courbe poly-Hermite avec translations initiale 500
331 Exemple de déclaration d'une courbe définie sous forme d'une expression littérale 501
332 liste des fonctions mathématiques disponibles 501
333 liste des opérations mathématiques binaires disponibles. **NB** L'opération = est particulière, il s'agit de transférer le continue de droite dans la variable de gauche, donc ne peut s'appliquer qu'à des variables. 502
334 liste des opérations mathématiques ternaires disponibles 502
335 Exemple de déclaration d'une courbe définie sous forme d'une expression littérale 503
336 liste des fonctions nD 504
337 Exemple de déclaration d'une liste de fonctions nD. 505
338 Deux exemples de déclaration d'une fonctions nD qui utilise une courbe 1D déjà existante 506

339 Deux exemples de déclaration d'une fonctions nD FONCTION_EXPRESSION_LITTERALE_nD ? . La première utilise deux fonctions existantes par ailleurs. La seconde utilise deux fonctions définies in situe. 509
340 Un exemple de déclaration d'une fonction qui combine des fonctions existantes et l'utilisation d'une variable globale ENERGIE_CINETIQUE 510
341 Exemple de déclaration d'une fonction externe nD. 512
342 Exemple de programme principal externe 513
343 Exemple d'une classe pour fonction externe : partie .h 514
344 Exemple d'une classe pour fonction externe : partie 1 .cc 515
345 Exemple d'une classe pour fonction externe : partie 2 .cc 516
346 Exemple d'une classe pour fonction externe : partie 3 .cc 517
347 Exemple d'une classe pour fonction externe : partie 4 .cc 518
348 Exemple de déclaration permettant d'activer un calcul d'erreur 523
349 Exemple de .CVisu contenant une visualisation d'erreur et de contraintes continues aux noeuds 523
350 Liste des options disponibles après une interruption via un contrôle c . . . 528

Table des figures

1 exemple d'une résultante de répartition sinusoïdale dans l'épaisseur d'une poutre (cf. 275) 385
2 exemple d'un moment dépendant d'une fonction nD (cf. 276) 385

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 rappel 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 généré 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 "grapheurs" 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++ à 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 coordonnees et il y a une vérification des dimensions en phase de mise au point grace 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 a 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 déja existante, mécanique thermique ..., est utilisé au travers de la définition particulière des fonctions virtuelle génériques.

— Utilisation sytématique 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 sytématique 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 possèder l'extension .info, exemple : biel1.info . On se reportera au fichier biel1.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 biel1

indique au programme Herezh "HZPP" que le fichier de commande a pour nom "biel1.info". Notez qu'il ne faut pas indiquer le suffixe ".info". La chaine "-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 inclut peu 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 blocages cinématiques,
- les paramètres éventuels de pilotage
- 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 certains cas, les mots clés principaux sont suivis de sous-mots clés qui permettent un découpage plus fin de l'information.

La liste exhaustive 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

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 spécial, 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. Lorque 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
```

Vu 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 caractères 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 :

- définition éventuelle de constantes utilisateurs (cf.4).
- dimension du problème (cf. 6). Cette information est optionnelle, par défaut c'est 3.
- le niveau de commentaire (cf. 7). Cette information est optionnelle, par défaut c'est 0.
- le type de problème (cf. III). Cette information est obligatoire.
- lecture éventuelle de fichiers de données préexistantes au calcul (cf. 25)
- la description des maillages (cf. 28). Cette information est obligatoire. On y décrit les maillages et les références.
- le nombre éventuel de maillages esclaves (cf.65.1). Cette information est optionnelle, par défaut c'est 0.
- la description des courbes1D (cf. 46). Cette information est optionnelle. Elle permet ensuite par exemple, de définir les chargements au travers de courbes de charges.
- la description des fonctions nD (cf. 81.2). Cette information est optionnelle. Les courbes nD sont susceptibles d'être utilisées partout dans la mise en données en particulier au niveau des lois comportement et du chargement.
- la description des lois de comportement (cf. 48). La définition d'une loi de comportement pour chaque élément est obligatoire.
- divers stockages (cf. 54). Ces informations sont obligatoires, mais varient en fonction des informations précédentes. Il s'agit :
  - de la masse volumique à 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.65.2). Cette information est optionnelle, par défaut c'est 0.

— Les zones éventuelles, présumées en contact (cf.65.3). 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. 66). 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. 67). 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. 69). 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. 54). Il s'agit ici de la seconde lecture des stockages divers. Cela concerne la lecture éventuelle des masses additionnelles. Cette information est optionnelle.

— l'algorithme de chargement (cf. 70). Information optionnelle. Elle décrit comment le chargement global s'effectue. Cela ne concerne pas les conditions limites qui utilisent 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. 71). Cette information comprend en fait un ensemble de paramètres qui sont pour la plupart optionnels. Par contre le mot clé de départ est obligatoire.

— la sortie des résultats (cf. 80). Information obligatoire. Par contre comprend en interne un ensemble de paramètres qui sont eux optionnels, et qui permettent de spécifier le type de sortie de l'information que l'on désir.

— le mot clé " `_fin_point_info_` " suivi de plusieurs lignes vides.

3 Liste des mots clés principaux

Les mots clés principaux, qui ont une signification particulière dans le fichier .info, sont indiqués dans le tableau 1. Bien noter qu'il existe d'autres mots clés qui sont internes aux différentes procédures.

TABLE 1 – liste des mots clés principaux (V. 6.908)

TYPE_DE_CALCUL	PARA_TYPE_DE_CALCUL
domaine_esclave	pas_de_sortie_finale_
nom_maillage	mvt_maitre
les_courbes_1D	les_fonctions_nD
choix_materiaux	materiaux
epaisseurs	largeurs
inerties	sections
variation_section	
integrale_sur_volume_	integrale_sur_vol_et_temps_
orthotropie	reperes_locaux
charges	blocages
masse_addi	initialisation
typecharge	controle
controle_contact	noeuds
elements	resultats
nom_maillage	flotExterne
para_syteme_lineaire	para_pilotage_equi_global
para_dedies_dynamique	para_affichage
masse_volumique	dilatation_thermique
para_contact	def_mouvement_solide_initiaux_
zone_contact	auto_contact
glue_contact	contact_solide_deformable
para_calculs_geometriques	para_energie
condition_limite_lineaire_	renumerotation_tous_maillages_
renumerotation_des_noeuds_	hourglass_gestion_
stabilisation_transvers_membrane_biel_	suppression_noeud_non_references_
fusion_avec_le_maillage_precedent_	def_auto_ref_frontiere_
_suite_point_info_	_fin_point_info_
_pause_point_info_	repere_anisotropie_
Constantes_utilisateurs_	Mise_A_jour_Constantes_utilisateurs_

4 Déclaration de constantes utilisateurs

En préambule, l'utilisateur peut éventuellement introduire des constantes qui ensuite peuvent être utilisées dans la mise en données, en tant que variable globale (cf.88.1). Ces variables sont utilisables dans les fonctions nD et dans les paramètres de contrôle (cf. 71).

La syntaxe est la suivante :

1. première ligne, le mot clé : `Constantes_utilisateurs_`
2. lignes suivantes (une ligne par définition d'une constante)
   - le mot clé `double_` suivi d'un nom (une chaîne de caractères) dont les 3 premiers caractères doivent obligatoirement être " `C__` " , puis un réel (notation scientifique ou uniquement décimale), ou un entier
   - ...
3. dernière ligne, le mot clé `fin_Constantes_utilisateurs_`

La constante scalaire est ainsi systématiquement stockée à l'aide d'un réel. Lors de son utilisation, elle est transformée en un entier ou un booléen équivalent en fonction du de la grandeur à affecter.

```
#
#   exemple de declaration de constantes utilisateurs
#
Constantes_utilisateurs_
    double_  C__aprecisionRunge        1.e-6
    double_   C__le_nombre_de_cycle    3
fin_Constantes_utilisateurs_
```

Au cours de l'exécution du calcul, ces constantes ne changent pas. Cependant l'utilisateur peut demander une modification de leur valeur au cours du calcul via deux méthodes.

1. Première méthode : lors de l'utilisation d'un calcul en plusieurs étapes (cf. 86) après chaque mot clé : " `_suite_point_info_` " il est possible de changer les valeurs des constantes. La suite du déroulement du calcul utilisera ces nouvelles valeurs pour les fonctions qui utilisent ces grandeurs et également les grandeurs de contrôle qui sont redéfinies après le mot clé " `_suite_point_info_` ". L'intérêt ainsi est de pouvoir adapter au cours des étapes, certains paramètres de contrôle ou certaines constantes utilisées par les fonctions nD.

   La syntaxe de modification est identiques à la déclaration initiale à l'exception des mots clés :

   (a) première ligne, le mot clé : `Mise_A_jour_Constantes_utilisateurs_`

   (b) lignes suivantes, une ligne par re-définition d'une constante. La constante doit déjà avoir été définie initialement.
   - le mot clé `double_` suivi d'un nom (une chaîne de caractères) dont les 3 premiers caractères doivent obligatoirement être " `C__` " , puis un réel (notation scientifique ou uniquement décimale),

— ...

(c) dernière ligne, le mot clé `fin_Mise_A_jour_Constantes_utilisateurs_`

Bien noter que :

(a) les paramètres de contrôle qui ne sont pas redéfinis, gardent leur valeur d'origine, même si initialement ils ont été défini à partir d'une constante utilisateur qui est modifiée,

(b) toutes les fonctions nD qui utilisent des constantes utilisateurs, prennent en compte directement les nouvelles valeurs des constantes utilisateurs.

2. Seconde méthode : l'utilisateur peut générer à tout moment, par exemple en fonction des résultats qu'il observe, une interruption du calcul, changer la valeur d'une constante, puis continuer le calcul qui va donc utiliser la nouvelle valeur de la constante. On se réfèrera à 85.2 pour plus d'indications pratiques sur la méthodologie.
Bien noter qu'après la modification d'une constante après une interruption : :

(a) seules les fonctions nD qui utilisent cette constante, sont impactées par sa nouvelle valeur ;

(b) la prise en compte de la nouvelle valeur est immédiate.

5 Déclaration de variables utilisateurs

En préambule, l'utilisateur peut éventuellement introduire des variables qui correspondent à des grandeurs calculées par Herezh pendant la résolution. Ces variables seront déclarées en tant que variable globale (cf.88.1) et ainsi sont utilisables "pendant le calcul" par les fonctions nD.

Les grandeurs disponibles calculées par Herezh sont celles accessibles en post-traitement au niveau des noeuds, des points d'intégration d'élément, des points d'intégration de face et/ou arête d'élément. Pour en connaître la liste il suffit d'utiliser le mode interactif de post-traitement au format .maple (cf. 80.5).

Les variables utilisateurs représentent une copie des grandeurs calculées par Herezh. Leur mise à jour s'effectue au début de chaque itération pour un calcul implicite et au début de chaque incrément pour un calcul explicite. Il s'agit ici de la résolution de l'équilibre globale.

Pour constituer une mise en données correcte du point de vue de la syntaxe, il est possible :

— soit d'utiliser le mode interactif d'Herezh++ de constitution d'un fichier .info,

— soit d'utiliser les règles syntaxiques qui suivent.

Règles initiales :

— Chaque variable est déclarée sur une ligne indépendante.

— Chaque variable est repérée par un nom de référence dont les 3 premiers caractères doivent obligatoirement être " `V__` " .

— Deux variables différentes doivent avoir un nom de référence différent.

— Une variable est systématiquement un scalaire du type "réel en double précision".

À partir des ces règles initiales les cas de variables possibles sont les suivants :

— un ddl pur défini à un noeud $->$ exemple :
" Ddl_noeud_ V__x1 nom_mail= poutre num_NE_ 3 ddl_ X1 temps_ TEMPS_0 "

1. Ddl_noeud_ V__x1 : mot clé qui indique qu'il s'agit d'un ddl à un noeud suivi du nom donné par l'utilisateur à la variable ici V__x1 : déclaration obligatoire
2. nom_mail= poutre : permets d'indiquer à quel maillage appartient le noeud, ici au maillage "poutre". Cette déclaration est optionnelle. Si elle est absente, le noeud est réputé appartenir au premier maillage déclaré.
3. num_NE_ 3 : permet de définir le numéro du noeud, ici "3" : déclaration obligatoire
4. ddl_ X1 : permet de définir le nom du ddl, ici " X1 " : déclaration obligatoire
5. temps_ TEMPS_0 : permet de définir à quel temps la variable est mise à jour avec 3 possibilités :
   — TEMPS_0 : valeur initiale,
   — TEMPS_t : valeur à la fin du précédent incrément de temps,
   — TEMPS_tdt : valeur actuelle

   la déclaration est obligatoire.

— un ddl étendu défini à un noeud $->$ exemple :
" Ddl_etendu_a_un_noeud_ V__m2 nom_mail= poutre num_NE_ 2 /
ddl_etendu_ Masse_diago_noeud "

Remarque importante : Le ddl étendu Masse_diago_noeud n'est pas définit systématiquement au noeud. Il est définit uniquement dans le cas de certains algorithmes : ceux relatifs à des calculs dynamiques ou dans le cas de l'algorithme de relaxation dynamique.

1. Ddl_etendu_a_un_noeud_ V__m2 : mot clé qui indique qu'il s'agit d'un ddl étendu à un noeud suivi du nom donné par l'utilisateur à la variable ici V__m2 : déclaration obligatoire
2. nom_mail= poutre : permets d'indiquer à quel maillage appartient le noeud, ici au maillage "poutre". Cette déclaration est optionnelle. Si elle est absente, le noeud est réputé appartenir au premier maillage déclaré.
3. ddl_ Masse_diago_noeud : permet de définir le nom du ddl étendu, ici " Masse_diago_noeud " : déclaration obligatoire
4. num_NE_ 2 : permet de définir le numéro du noeud, ici "2" : déclaration obligatoire

— une grandeur quelconque définie à un noeud $->$ exemple :
" Quelconque_a_un_noeud_ V__M1 nom_mail= poutre num_NE_ 2 /
grandeur_quelconque_ FORCE_GENE_EXT composante: 1 "

1. Quelconque_a_un_noeud_ V__M1 : mot clé qui indique qu'il s'agit d'une grandeur quelconque à un noeud suivi du nom donné par l'utilisateur à la variable ici V__M1 : déclaration obligatoire

2. nom_mail= poutre : permets d'indiquer à quel maillage appartient le noeud, ici au maillage "poutre". Cette déclaration est optionnelle. Si elle est absente, le noeud est réputé appartenir au premier maillage déclaré.
3. num_NE_ 2 : permet de définir le numéro du noeud, ici "2" : déclaration obligatoire
4. grandeur_quelconque_ FORCE_GENE_EXT : permet de définir le nom de la grandeur quelconque, ici " FORCE_GENE_EXT " : déclaration obligatoire
5. composante: 1 : définis le numéro de la composante ou le numéro d'ordre dans la grandeur quelconque, ici il s'agira de la composante suivant x de la résultante des forces extérieures appliquée au noeud : déclaration obligatoire.

— un ddl pur défini à un pti d'élément, analogue au post-traitement : 80.5.3, − > exemple :
" Ddl_a_un_element_ V__sig11_21 nom_mail= poutre num_NE_ 2 /
temps_ TEMPS_0 absolu_ 1 num_pti_ 1 ddl_ SIG11 "

1. Ddl_a_un_element_ : mot clé qui indique qu'il s'agit d'un ddl de pti d'élément suivi du nom donné par l'utilisateur à la variable ici V__sig11_21 : déclaration obligatoire
2. nom_mail= poutre : permets d'indiquer à quel maillage appartient le noeud, ici au maillage "poutre". Cette déclaration est optionnelle. Si elle est absente, le noeud est réputé appartenir au premier maillage déclaré.
3. num_NE_ 2 : permet de définir le numéro de l'élément, ici "2" : déclaration obligatoire
4. absolu_ 1 : indique si l'on veut la grandeur dans le repère absolu (rep. 1 par défaut) ou dans un repère ad hoc (rep. 0). Utile uniquement pour les grandeurs vectorielles ou tensorielles : déclaration optionnelle.
5. temps_ TEMPS_0 : permet de définir à quel temps la variable est mise à jour avec 3 possibilités :
   — TEMPS_0 : valeur initiale,
   — TEMPS_t : valeur à la fin du précédent incrément de temps,
   — TEMPS_tdt : valeur actuelle

   la déclaration est obligatoire.
6. ddl_ SIG11 : permet de définir le nom du ddl, ici " SIG11 " : déclaration obligatoire

— une grandeur particulière définie à un pti d'élément, analogue au post-traitement 80.5.4, − > exemple :
" Particulier_a_un_element_ V__VOLUME_PTI_21 /
nom_mail= poutre num_NE_ 2 absolu_ 1 num_pti_ 1 /
grandeur_quelconque_ VOLUME_PTI composante: 1 "

1. Particulier_a_un_element_ V__VOLUME_PTI_21 : mot clé qui indique qu'il s'agit d'une grandeur particulière à un pti d'élément suivi du nom donné par l'utilisateur à la variable ici V__VOLUME_PTI_21 : déclaration obligatoire

2. nom_mail= poutre : permets d'indiquer à quel maillage appartient le noeud, ici au maillage "poutre". Cette déclaration est optionnelle. Si elle est absente, le noeud est réputé appartenir au premier maillage déclaré.
3. num_NE_ 2 : permet de définir le numéro de l'élément, ici "2" : déclaration obligatoire
4. absolu_ 1 : indique si l'on veut la grandeur dans le repère absolu (rep. 1 par défaut) ou dans un repère ad hoc (rep. 0). Utile uniquement pour les grandeurs vectorielles ou tensorielles : déclaration optionnelle.
5. grandeur_quelconque_ VOLUME_PTI : définition de la grandeur quelconque associée à la variable utilisateur, ici : VOLUME_PTI : déclaration obligatoire.
6. composante: 1 : définis le numéro de la composante ou le numéro d'ordre dans la grandeur quelconque : déclaration obligatoire.

— une grandeur évoluée définie à un pti d'élément, analogue au post-traitement 80.5.5, − > exemple :

" Evoluee_a_un_element_ V__epslog_PTI_21 /
nom_mail= poutre num_NE_ 2 absolu_ 1 num_pti_ 1 /
grandeur_quelconque_ LOGARITHMIQUE composante: 1 "

1. Evoluee_a_un_element_ V__epslog_PTI_21 : mot clé qui indique qu'il s'agit d'une grandeur évoluée à un pti d'élément suivi du nom donné par l'utilisateur à la variable ici V__epslog_PTI_21 : déclaration obligatoire
2. nom_mail= poutre : permets d'indiquer à quel maillage appartient le noeud, ici au maillage "poutre". Cette déclaration est optionnelle. Si elle est absente, le noeud est réputé appartenir au premier maillage déclaré.
3. num_NE_ 2 : permet de définir le numéro de l'élément, ici "2" : déclaration obligatoire
4. absolu_ 1 : indique si l'on veut la grandeur dans le repère absolu (rep. 1 par défaut) ou dans un repère ad hoc (rep. 0). Utile uniquement pour les grandeurs vectorielles ou tensorielles : déclaration optionnelle.
5. grandeur_quelconque_ LOGARITHMIQUE : définition de la grandeur quelconque associée à la variable utilisateur, ici : la mesure logarithmique du tenseur de déformation LOGARITHMIQUE : déclaration obligatoire.
6. composante: 1 : définis le numéro de la composante ou le numéro d'ordre dans la grandeur quelconque, ici il s'agit de la première composante donc ϵ_{11} : déclaration obligatoire. L'ordre des composantes est identique à celui utilisé en post-traitement (cf. 80.5.12).

— une grandeur quelconque définie à un pti de face d'élément, analogue au post-traitement 80.5.6, − > exemple :

" Quelconque_face_element_ V__press_n2_21 /
nom_mail= poutre num_NE_ 2 absolu_ 1 num_pti_ 1 /
grandeur_quelconque_ VECT_PRESSION composante: 2 num_FA_ 1 "

1. Quelconque_face_element_ V__press_n2_21 : mot clé qui indique qu'il s'agit d'une grandeur évoluée à un pti d'élément suivi du nom donné par l'utilisateur à la variable ici V__press_n2_21 : déclaration obligatoire

2. nom_mail= poutre : permets d'indiquer à quel maillage appartient le noeud, ici au maillage "poutre". Cette déclaration est optionnelle. Si elle est absente, le noeud est réputé appartenir au premier maillage déclaré.
3. num_NE_ 2 : permet de définir le numéro de l'élément qui contient la face, ici "2" : déclaration obligatoire
4. absolu_ 1 : indique si l'on veut la grandeur dans le repère absolu (rep. 1 par défaut) ou dans un repère ad hoc (rep. 0). Utile uniquement pour les grandeurs vectorielles ou tensorielles : déclaration optionnelle.
5. grandeur_quelconque_ VECT_PRESSION : définition de la grandeur quelconque associée à la variable utilisateur, ici : le vecteur pression VECT_PRESSION : déclaration obligatoire.
6. composante: 2 : définis le numéro de la composante ou le numéro d'ordre dans la grandeur quelconque, ici il s'agit de la deuxième composante dans le repère absolu du vecteur pression : déclaration obligatoire.
7. num_FA_ 1 : définis le numéro de la face d'élément, ici 1 : déclaration obligatoire.

— une grandeur quelconque définie à un pti d'une arête d'élément, analogue au post-traitement 80.5.6, − > exemple :

" Quelconque_arete_element_ V__press_n2_21 /
nom_mail= poutre num_NE_ 2 absolu_ 1 num_pti_ 1 /
grandeur_quelconque_ POSITION_GEOMETRIQUE composante: 2 num_FA_ 1 "

1. Quelconque_arete_element_ V__press_n2_21 : mot clé qui indique qu'il s'agit d'une grandeur évoluée à un pti d'élément suivi du nom donné par l'utilisateur à la variable ici V__press_n2_21 : déclaration obligatoire
2. nom_mail= poutre : permets d'indiquer à quel maillage appartient le noeud, ici au maillage "poutre". Cette déclaration est optionnelle. Si elle est absente, le noeud est réputé appartenir au premier maillage déclaré.
3. num_NE_ 2 : permet de définir le numéro de l'élément qui contient la face, ici "2" : déclaration obligatoire
4. absolu_ 1 : indique si l'on veut la grandeur dans le repère absolu (rep. 1 par défaut) ou dans un repère ad hoc (rep. 0). Utile uniquement pour les grandeurs vectorielles ou tensorielles : déclaration optionnelle.
5. grandeur_quelconque_ POSITION_GEOMETRIQUE : définition de la grandeur quelconque associée à la variable utilisateur, ici : le vecteur position du pti de l'arête POSITION_GEOMETRIQUE : déclaration obligatoire.
6. composante: 2 : définis le numéro de la composante ou le numéro d'ordre dans la grandeur quelconque, ici il s'agit de la deuxième composante dans le repère absolu de la position : déclaration obligatoire.
7. num_FA_ 1 : définis le numéro de l'arête d'élément, ici 1 : déclaration obligatoire.

La table (2) donne un exemple générique de toutes les déclarations possibles.

TABLE 2 – Exemple générique de déclaration de toutes les variables utilisateur possibles

```
#-----------------------------------------------------------------
 Variables_utilisateurs_
   Ddl_noeud_ V__a num_NE_ 1 ddl_  X1 temps_ TEMPS_0
   Ddl_etendu_a_un_noeud_ V__b num_NE_ 1 ddl_etendu_  Masse_diago_noeud
   Quelconque_a_un_noeud_ V__c num_NE_ 1 \
       grandeur_quelconque_ POSITION_GEOMETRIQUE composante: 1
   Ddl_a_un_element_ V__d num_NE_ 1 absolu_ 1 num_pti_ 1  \
       temps_ TEMPS_0  ddl_  SIG11
   Particulier_a_un_element_ V__e num_NE_ 1 absolu_ 1 num_pti_ 1  \
       grandeur_quelconque_ POSITION_GEOMETRIQUE composante: 1
   Evoluee_a_un_element_ V__f num_NE_ 1 absolu_ 1 num_pti_ 1  \
       grandeur_quelconque_ DEF_PRINCIPALES composante: 1
   Quelconque_face_element_ V__g num_NE_ 1 absolu_ 1 num_pti_ 1  \
       grandeur_quelconque_ NN_SURF_t0 composante: 1  num_FA_ 1
   Quelconque_arete_element_ V__h num_NE_ 1 absolu_ 1 num_pti_ 1  \
      grandeur_quelconque_ POSITION_GEOMETRIQUE composante: 1  num_FA_ 1
 fin_Variables_utilisateurs_
```

6 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 demandée. 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 toutes les dimensions d'espace. À 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.

7 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

8 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'equilibre non dynamique
#-----------------------------------
non_dynamique
```

La liste exhaustive des problèmes que l'on peut traiter est donnée par le tableau (3).

TABLE 3 – 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 (10.1)
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 méhode utilisée est les différences finis centrées cf.(12) un paramètre facultatif associé cf.(12)
dynamique_explicite_tchamwa	dynamique explicite sans contact, il s'agit de la méhode de Tchamwa-Wielgoz cf.(11) 1 paramètres associé cf. (11)
dynamique_implicite	dynamique implicite sans contact, correspond à la méthode de Newmark cf.(13) ou à la méthode HHT (Hilbert-Hughes-Taylor) deux ou 1 paramètres associés (cf. 13) ou cf. (13)
dynamique_explicite_chung_lee	dynamique explicite sans contact correspond à la méthode proposée par Chung-Lee cf.(14) 1 paramètre associé cf. (14)
dynamique_explicite_zhai	dynamique explicite sans contact correspond à la méthode proposée par Zhai. cf.(15) 4 paramètres associés
dynamique_Runge_Kutta	dynamique : avancement temporel résolu par la méthode de Runge-Kutta. cf.(16) plusieurs paramètres associés cf. (16)
dynamique_relaxation_dynam	relaxation dynamique recherche de la solution statique via la relaxation dynamique et la masse optimisée cf. (20.3)
umat_abaqus	utilisation d'herezh++ comme umat, permet l'utilisation de toutes les lois d'herezh++ par abaqus cf. (17)
utilitaires	différents utilitaires en pré ou post calcul par exemple de modification de maillage (cf. 21)
informations	recuperation d'informations par exemple récupération et création interactives de references (cf. 22)
dynamique_explicite_bonelli	dynamique explicite de type Galerkin Discontinu (cf. 23)
combiner	combinaison de plusieurs algorithmes déjà existants par exemple : implicite et explicite (cf. 24)

9 Liste des algorithmes Galerkin continu et utilitaires

Chaque nom de type de calcul (cf. table 3) peut être suivi d'un sous-mot clé indiquant des calculs suplémentaires a effectué. Par exemple :

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

indique que le calcul explicit 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.

On se réfèrera au tableau (39) pour avoir la liste exhaustive des sous-types possibles.

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 : " PARA_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 :

```
PARA_TYPE_DE_CALCUL
#----------------------------------
#  type de parametre   |           |
#  associe au calcul   | valeur    |
#      phie            |           |
#----------------------------------
       phi=                1.05
```

indique que le paramètre phi (ce reporter au fonctionnement de l'algorithme 11 pour la signification de ce paramètre) à la valeur 1.05. Notons que ce paramètre n'est utilisable qu'avec la méhode de Thamwa-Wielgoz, aussi dans le cas de l'utilisation d'une autre méthode une erreur à l'exécution peut-être générée, ou bien le paramètre est ignoré.

Remarque

— Pendant l'exécution d'un algorithme on peut consulter (via une fonction nD cf.81.2) la variable globale :

algo_global_actuel
qui contient le numéro de code interne de l'algorithme en cours. Ce numéro correspond à un type énuméré (type C ou C++) défini de la façon suivante :

```
enum EnumTypeCalcul {DYNA_IMP = 1, DYNA_EXP,DYNA_EXP_TCHAMWA,
                     DYNA_EXP_CHUNG_LEE,DYNA_EXP_ZHAI,
                     DYNA_RUNGE_KUTTA,NON_DYNA, NON_DYNA_CONT,
                     FLAMB_LINEAIRE,INFORMATIONS,UTILITAIRES,
                     DEF_SCHEMA_XML,RIEN_TYPECALCUL,
                     UMAT_ABAQUS, DYNA_EXP_BONELLI ,
                     RELAX_DYNA, STAT_DYNA_EXP ,
                     COMBINER
                    };
```

ce qui signifie que l'on a la correspondance suivante :

1. DYNA_IMP
2. DYNA_EXP
3. DYNA_EXP_TCHAMWA
4. DYNA_EXP_CHUNG_LEE
5. DYNA_EXP_ZHAI
6. DYNA_RUNGE_KUTTA
7. NON_DYNA
8. NON_DYNA_CONT
9. FLAMB_LINEAIRE
10. INFORMATIONS
11. UTILITAIRES
12. DEF_SCHEMA_XML
13. RIEN_TYPECALCUL
14. UMAT_ABAQUS
15. DYNA_EXP_BONELLI
16. RELAX_DYNA
17. STAT_DYNA_EXP
18. COMBINER

Cette variable interne est en particulier utile lorsque l'on utilise l'algorithme combiner (cf. 24) qui combine plusieurs sous-algorithmes.

10 Paramètres communs 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 4 – liste des type de paramètres génériques pouvant être associés aux paramètres particuliers de l'algorithme

objectif des paramètres	reférence	types de calcul associé
amortissement cinétique	(20.3.9)	algorithmes de dynamique et de relaxation dynamique
amortissement visqueux	(20.1)	algorithmes de dynamique et de relaxation dynamique
critère d'arrêt sur le résidu statique hors viscosité numérique	(20.4)	algorithmes de dynamique et de relaxation dynamique
contrôle mode debug	(20.3.11)	tous les algorithmes de calcul
modulation de la précision d'équilibre globale	(20.3.12)	tous les algorithmes de calcul

La table 5 donne un exemple d'utilisation de paramètres génériques.

TABLE 5 – 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
```

10.1 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 a obtenir une solution optimisée. Pour utiliser cette méthode, on définit des paramètres associés à l'algorithme. La table (6) donne un exemple d'utilisation.

TABLE 6 – Exemple de paramètres pour mettre en oeuvre et conduire la méthode d'accélération de convergence

```
#-------------------------------------------------------------------
#|  parametres (falcultatifs ) 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 accroissement du vecteur solution $S_n = \Delta X_{n+1}^{t+\Delta t} - \Delta X_{n+1}^{t+\Delta t}$. 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.

11 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 définie une fonction "f" et le coefficient résultant devient pour chaque ddl : $(1+f(|\gamma(i)|).(\varphi-1))$ (au lieu de φ initialement), "i" étant le numéro du ddl. La table (7) 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 7 – 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 rafinement est possible en se referent à une moyenne des accelerations sur n pas.

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

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" :

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

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 (8) donne un exemple d'utilisation.

TABLE 8 – 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
```

12 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 :

$$\dot{\boldsymbol{X}}_{n-1/2} = \frac{\boldsymbol{X}_n - \boldsymbol{X}_{n-1}}{\Delta t} \tag{3}$$

$$\dot{\boldsymbol{X}}_{n+1/2} = \frac{\boldsymbol{X}_{n+1} - \boldsymbol{X}_n}{\Delta t} \tag{4}$$

d'où

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

D'une manière pratique, le calcul s'effectue dans Herezh++ en un pas. Dans la version initiale (voir 12 avec `type_cal_equilibre=` 1 ou 2), on définit le vecteur des inconnues généralisées au temps "n" suivant :

$$\left\{ \begin{array}{c} \boldsymbol{X}_n \\ \dot{\boldsymbol{X}}_n \\ \dot{\boldsymbol{X}}_{n-1/2} \\ \ddot{\boldsymbol{X}}_n \end{array} \right\} \tag{6}$$

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

— calcul de $\dot{\boldsymbol{X}}_{n+1/2}$ avec (5), première expression,

— calcul de $\boldsymbol{X}_{n+1}$ avec (4),

— calcul d'une version explicite de la vitesse au temps "n+1" :

$$\dot{\boldsymbol{X}}_{n+1} = \dot{\boldsymbol{X}}_{n+1/2} + \frac{\Delta t}{2}\, \ddot{\boldsymbol{X}}_n \tag{7}$$

— calcul de $\ddot{\boldsymbol{X}}_{n+1}$ à l'aide de l'équation d'équilibre

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

— calcul de la vitesse réelle (au sens des DFC) au temps "n+1"

$$\dot{\boldsymbol{X}}_{n+1} = \dot{\boldsymbol{X}}_{n+1/2} + \frac{\Delta t}{2} \ddot{\boldsymbol{X}}_{n+1} \tag{9}$$

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 $\boldsymbol{X}_{n+1}$ 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 $\dot{\boldsymbol{X}}_{n-1/2}$ et $\dot{\boldsymbol{X}}_{n+1/2}$ en utilisant la séquence suivante :

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

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'ou la valeur de l'accélération, pour initier le calcul.

13 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.

D'une manière classique on a les conditions suivantes :

— $\gamma < 0.5$ le calcul est instable (définitivement !) donc à ne pas utiliser,

— $\gamma = 0.5$ et $\beta = 0$ cela correspond globalement à la méthode DFC qui est conditionnellement stable,

— $\gamma \geq 0.5$ et $2\beta < \gamma$ la méthode est conditionnellement stable,

— $\gamma \geq 0.5$ et $2\beta \geq \gamma$ la méthode est inconditionnellement stable,

La table (9) donne un exemple de déclaration de la méthode de Newmark avec des paramètres γ et β particuliers.

TABLE 9 – Exemple de déclaration de la méthode de Newmark avec des paramètres particuliers γ et β

```
        dynamique_implicite   #avec plus visualisation

#-------------------------------------------------------------------
#|  parametres (falcultatifs) associes l'algorithme de Newmark      |
#-------------------------------------------------------------------

    PARA_TYPE_DE_CALCUL
  #    .....................................................
  #   / facteur beta  puis facteur gamma,  (ou facteur hht)     /
  #  / (dans le cas de la methode hht: beta et gamma sont fixe)/
  # / ( il ne sont donc pas à indiquer )                      /
  #.....................................................
        beta_et_gamma= 0.25  0.5
```

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 modifié. 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 $\gamma = 1/2 - \theta$. La table (10) donne un exemple de déclaration.

TABLE 10 – Exemple de déclaration de la méthode de Newmark avec un amortissement numérique type HHT

```
 dynamique_implicite

#-------------------------------------------------------------------
#|  parametres (falcultatifs) associes l'algorithme de Newmark      |
#-------------------------------------------------------------------

    PARA_TYPE_DE_CALCUL
  #    .....................................................
  #   / facteur beta  puis facteur gamma,  (ou facteur hht)     /
  #  / (dans le cas de la methode hht: beta et gamma sont fixe)/
  # / ( il ne sont donc pas à indiquer )                      /
  #.....................................................
       hht=  -0.05
```

14 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.

L'algorithme proposé par Chung-Lee ([Hulbert et Chung, 1996]) est donné par les expressions suivantes :

$$M\ \ddot{q}_{n+1} + \mathcal{R}_{int}(q_n, \dot{q}_n) = \mathcal{R}_{ext}(q_n, \dot{q}_n) \tag{11}$$

$$q_{n+1} = q_n + \Delta t \dot{q}_n + \Delta t^2(\hat{\beta}\ddot{q}_n + \beta\ddot{q}_{n+1}) \tag{12}$$

$$\dot{q}_{n+1} = \dot{q}_n + \Delta t(\hat{\gamma}\ddot{q}_n + \gamma\ddot{q}_{n+1}) \tag{13}$$

L'analyse de la consistance de l'algorithme montre que la précision est du second ordre lorsque $\hat{\beta} = 1/2 - \beta$, $\gamma = 3/2$ et $\hat{\gamma} = 1 - \gamma$.

L'étude de l'influence des paramètres sur la convergence du schéma numérique conduit au domaine utile de β :

$$1. \leq \beta \leq 28/27 \tag{14}$$

De manière plus précise, le paramètre de réglage correspond au paramètre β . Par défaut, $\beta = 28./27. = 1.037$.

Pour changer sa valeur on utilise le mot clé beta= suivi de la valeur désirée.

On peut également indiquer un arrêt correspondant à un équilibre statique. Dans ce cas le calcul s'arrête lorsque l'on a atteint un régime statique d'équilibre dont le critère est :

- le résidu statique est inférieur au maxi indiqué dans les paramètres généraux de contrôle,
- si l'on utilise de la relaxation cinétique : le critère d'arrêt est celui de l'algorithme associé.

Pour activer cette possibilité d'arrêt il faut indiquer dans les paramètres de l'algorithme, sur la ligne qui suit le mot clé :

" ARRET_A_EQUILIBRE_STATIQUE_ " suivi d'un nombre :

- ARRET_A_EQUILIBRE_STATIQUE_ 1 signifie que le résidu statique est utilisé comme critère,
- ARRET_A_EQUILIBRE_STATIQUE_ 2 signifie que l'on utilise le critère de la relaxation cinétique,
- ARRET_A_EQUILIBRE_STATIQUE_ 3 signifie que l'on utilise le maxi des deux critères précédent

```
#    exemple d'utilisation
#
  PARA_TYPE_DE_CALCUL
#  ....................................................
#   / facteur beta  /
#  ................."
      beta= 1.037
   ARRET_A_EQUILIBRE_STATIQUE 1
#
```

15 Méthode proposée par Zhai

Le schéma d'avancement temporel proposé par Zhai [ZHAI, 1996] est une méthode explicite qui s'appuie sur une résolution en deux étapes types : prédiction puis correction. 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. Comme pour les algorithmes de Chung Lee et Tchamwa, l'amortissement est d'autant plus efficace que le pas de temps est proche de pas de temps critique.

1. prédiction :

$$\begin{aligned} q(z)_{p,n+1} &= q(z)_n + \Delta t \dot{q}(z)_n + \Delta t^2 \left\{(1/2+\Psi)\ddot{q}(z)_n - \Psi \ddot{q}(z)_{n-1}\right\} \\ \dot{q}(z)_{p,n+1} &= \dot{q}(z)_n + \Delta t \left\{(1+\varphi)\ddot{q}(z)_n - \varphi \ddot{q}(z)_{n-1}\right\} \end{aligned} \tag{15}$$

   avec "(z)" désignant les grandeurs introduites par Zhai et "p" pour "predicted". $q(z)_{p,n+1}$ et $\dot{q}(z)_{p,n+1}$ correspondent à la prédiction du déplacement et de la vitesse.

2. calcul des efforts internes et externes correspondants à cette cinématique, puis résolution des équations d'équilibres → prédiction de l'accélération :

$$\ddot{q}(z)_{p,n+1} \tag{16}$$

3. correction du déplacement et de la vitesse avec une formule de type Newmark

$$\begin{aligned} q(z)_{n+1} &= q(z)_n + \Delta t \dot{q}(z)_n + \Delta t^2 \left\{(1/2-\beta)\ddot{q}(z)_n + \beta \ddot{q}(z)_{p,n+1}\right\} \\ \dot{q}(z)_{n+1} &= \dot{q}(z)_n + \Delta t \left\{(1-\gamma)\ddot{q}(z)_n + \gamma \ddot{q}(z)_{p,n+1}\right\} \end{aligned} \tag{17}$$

4. de nouveau, calcul des efforts internes et externes correspondants à cette cinématique corrigée, puis résolution des équations d'équilibres → accélération finale

$$\ddot{q}(z)_{n+1} \tag{18}$$

4 paramètres de contrôle sont disponibles.

Dans la pratique si l'on veut une importante précision, Zhai préconise de retenir $\Psi = \varphi = 1/2$ et les paramètres habituels de Newmark : $\gamma = 1/2$ et $\beta = 1/4$. On obtient la meilleure précision pour $\gamma = 1/2$ et $\beta = 1/12$. Avec ces paramètres on obtient un très

faible amortissement et un rayon de convergence environ deux fois plus faible que celui de DFC classique, ce qui n'est pas du tout intéressant.

Par exemple en utilisant comme paramètres : $\varphi = \gamma = 1/2$ et $\Psi = \beta = 1/6$, on obtient un rayon de convergence un peu inférieur à DFC (5 à 10% plus faible), par contre l'atténuation est très importante, du même ordre (voir un peu supérieure) à celle de Chung Lee. En temps de calcul, la méthode est cependant environ deux fois plus lente que celles de Chung Lee ou Tchamwa compte tenue de l'étape de prédiction.

```
#    exemple d'utilisation
#
dynamique_explicite_zhai     avec plus lectureCommandesVisu

PARA_TYPE_DE_CALCUL
   phi_minus= 0.5   grand_phi= 0.16667  gamma= 0.5 beta= 0.16667
```

Comme pour Chung Lee, on peut également indiquer un arrêt correspondant à un équilibre statique (cf. 14).

16 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 résolut 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 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_` * $max(max(|X_i|), \Delta t * max(|\dot{X}_i|))$". 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 évité 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 (11) donne un exemple de déclaration de paramètres.

Il est a 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 11 – 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
```

17 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.*, 2008b] pour plus d'information sur les fondements scientifiques dans le cas d'un comportement 3D. Le cas des contraintes planes est particulier, le chapitre ? est dédié à ce type de comportement. Il est également possible à l'aide de la technique présentée si-dessous, d'utiliser dans Herezh++ une loi quelconque extérieure à Herezh++, programmée à l'aide d'un outil quelconque (par exemple via octave, ou directement en C ou C++).

Au niveau d'Abaqus (ou d'Herezh++) il est nécessaire de définir la fonction utilisateur Umat selon le format donné par la table 12. Cette fonction appelle une routine C définie par les tables 13 et 14. Les routines de lecture et écriture sont données au chapitre (51.18).

17.1 Cas général d'une loi 3D

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 15 donne un exemple d'un fichier de commande permettant de définir l'utilisation d'une loi hyper-élastique de type Mooney-Rivlin. On voit que le maillage contient un seul élément qui est d'un type particulier : "POINT CONSTANT" . Il ne faut utiliser que ce type de maillage qui est spécifique et ne peut-être utilisé pour autre chose.

Dans le cas d'une loi qui ne dépend pas de degré de liberté au noeud, cas de l'exemple 15, il n'est pas nécessaire de définir un seul noeud, et donc le maillage peut ne contenir aucun noeud et l'élément " `POINT CONSTANT` " n'est associé à aucun noeud.

Au contraire, dans le cas d'une loi dépendante d'une grandeur normalement définie aux noeuds, ce qui est le cas de la température, il est nécessaire de suivre la procédure habituelle dans Herezh pour introduire la dépendance du point d'intégration à un noeud. La table (18) donne un exemple pour une thermodépendance. Les ajouts par rapport au cas précédent sont les suivants :

1. définition d'un point dont les coordonnées peuvent être quelconques (elles ne seront pas utilisées dans le calcul)

2. ajout d'une connexion à l'élément (le point)
3. ajout d'un blocage sur le noeud (N_tout TEMP 0) dont la valeur n'a pas d'importance. Mais le fait que ce blocage existe conduit Herezh a définir un ddl de température au noeud
4. utilisation d'une loi thermodépendante.

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 suboutine 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 (51.18) pour plus d'information.

17.2 Cas d'une loi de contrainte plane

Le tenseur de contrainte calculé, est tel que les composantes $\sigma^{i3} = 0$, i=1..3. L'axe 3 est ainsi particulier. Dans le cas de la simulation du comportement de membranes ou de plaques ou coques, la condition de contrainte plane est en général retenue. L'axe 3 représente alors la normale au plan moyen (ou médian) de l'élément. Aussi, dans le cas d'une coque par exemple, la direction 3 varie suivant la position du point de matière considéré. Il est nécessaire dans le programme appelant l'Umat (Abaqus ou Herezh++) d'exprimer tous les tenseurs nécessaires au calcul de la loi, dans un repère local tel que la direction 3 soit celle de la condition de contrainte plane.

Comme pour l'utilisation d'une Umat 3D, l'appel dans Herezh d'un comportement particulier s'effectue à l'aide d'un fichier .info classique qui contient l'algorithme " `umat_abaqus` ", sans autre paramètre particulier. Par rapport au cas 3D, les différences sont :

— le type d'élément à utiliser est " `POINT_CP` " (cf. 19)

— la loi de comportement doit-être d'un type contrainte plane

Remarque : Le fait d'utiliser une loi qui dépend d'une grandeur définie au noeud, comme par exemple la température, suit la même logique que pour le cas 3D.

TABLE 12 – déclaration de la fonction Umat fortran : partie fortran

```
C -----------------------------------------------------------
C     UMAT_Herezh
c     appel a subroutine c
C     pour execution  de HZ++
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 DFGRD0,DFGRD1,NOEL,NPT,KSLAY,KSPT,KSTEP,KINC)
C
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER(LEN=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 DFGRD0(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/
C
cc    Appel fonction C pour lancement de Herezh
      EXTERNAL appelc

      IF (NDI.NE.3) THEN
         WRITE(6,1)
 1       FORMAT(//,30X,'***ERROR - THIS UMAT MAY ONLY BE USED FOR ',
     1          'ELEMENTS WITH THREE DIRECT STRESS COMPONENTS')
         STOP
      ENDIF
```

TABLE 13 – 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 14 – 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* DDSDDE,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* DFGRD0,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 ,DFGRD0
                    ,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC);
   /*-----------------------------------------------------------------------------------*/
   /* C function for reading data in the pipe */
   /*-----------------------------------------------------------------------------------*/
   LectureDonneesUmat(STRESS,DDSDDE,SSE,SPD,SCD,NDI,NSHR,NTENS,PNEWDT ,RPL
                    ,DDSDDT,DRPLDE,DRPLDT);
}
```

TABLE 15 – 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_  #------------------
```

TABLE 16 – exemple de l'algorithme permettant d'utiliser Herezh++ en tant qu'Umat, pour la définition d'une loi thermo-dépendante

```
# ------------------------------------------------
#   exemple Umat   thermodependante  #
# ------------------------------------------------
dimension 3
niveau_commentaire 4
TYPE_DE_CALCUL
umat_abaqus     # definition de l'algorithme
#-------------------------------------------------------------------
#   maillage
#-------------------------------------------------------------------
noeuds
1 NOEUDS
1  0  0  0

elements
1 ELEMENTS

1 POINT CONSTANT  1
   E_to 1

#-------------------------------------------------------------------


les_courbes_1D
   C1_fonction_T COURBEPOLYLINEAIRE_1_D
      Debut_des_coordonnees_des_points
        Coordonnee dim= 2 -34.  0.006
        Coordonnee dim= 2 -29.  0.057
        Coordonnee dim= 2 -17.  0.127
        Coordonnee dim= 2 -13.  0.156
        Coordonnee dim= 2 -9.   0.282
        Coordonnee dim= 2 0.5   0.465
        Coordonnee dim= 2 10.   0.547
        Coordonnee dim= 2 20.   0.556
        Coordonnee dim= 2 80.   0.607
      Fin_des_coordonnees_des_points
```

TABLE 17 – suite exemple de l'algorithme permettant d'utiliser Herezh++ en tant qu'Umat, pour la définition d'une loi thermo-dépendante

```
   C2_fonction_T COURBEPOLYLINEAIRE_1_D
      Debut_des_coordonnees_des_points
        Coordonnee dim= 2 -34.  0.255
        Coordonnee dim= 2 -29.  0.255
        Coordonnee dim= 2 -17.  0.255
        Coordonnee dim= 2 -13.  0.305
        Coordonnee dim= 2 -9.   0.101
        Coordonnee dim= 2 0.5   0.008
        Coordonnee dim= 2 10.   0.021
        Coordonnee dim= 2 20.   0.065
        Coordonnee dim= 2 80.   0.062
      Fin_des_coordonnees_des_points

   C3_fonction_T COURBEPOLYLINEAIRE_1_D
      Debut_des_coordonnees_des_points
        Coordonnee dim= 2 -34.  23.128
        Coordonnee dim= 2 -29.  43.128
        Coordonnee dim= 2 -17.  23.128
        Coordonnee dim= 2 -13.  18.175
        Coordonnee dim= 2 -9.   10.426
        Coordonnee dim= 2 0.5   0.031
        Coordonnee dim= 2 10.   0.003
        Coordonnee dim= 2 20.   0.008
        Coordonnee dim= 2 80.   0.014
      Fin_des_coordonnees_des_points


choix_materiaux
E_tout H_AVEC_T_C1_C2_C3


materiaux

#C1, C2, C3 dependant T
H_AVEC_T_C1_C2_C3 HART_SMITH3D
  C1= C1_thermo_dependant_ C1_fonction_T
  C2= C2_thermo_dependant_ C2_fonction_T
  C3= C3_thermo_dependant_ C3_fonction_T
  K= 0 type_potvol_ 4

masse_volumique
E_tout 1.
```

TABLE 18 – suite exemple de l'algorithme permettant d'utiliser Herezh++ en tant qu'Umat, pour la définition d'une loi thermo-dépendante

```
charges

blocages

N_tout  'TEMP=  0.'

controle
SAUVEGARDE 1

resultats  pas_de_sortie_finale_
COPIE 0
_fin_point_info_
```

TABLE 19 – exemple de l'algorithme permettant d'utiliser Herezh++ en tant qu'Umat en contrainte plane

```
dimension 3
#------------------------------------------------------------------
niveau_commentaire 3
#------------------------------------------------------------------
TYPE_DE_CALCUL
umat_abaqus
#------------------------------------------------------------------
#   maillage
#------------------------------------------------------------------
   noeuds ------------
    0  NOEUDS
   elements ----------
    1  ELEMENTS
    1   POINT_CP  CONSTANT
   E_to 1
#------------------------------------------------------------------
   choix_materiaux

#-----------------------------------------
# Nom Materiau    |      Type loi         |
#-----------------------------------------
  acier_interne_CP         LOI_CONTRAINTES_PLANES
 NEWTON_LOCAL  avec_parametres_de_reglage_
                nb_iteration_maxi_  20
                nb_dichotomie_maxi_  20
                tolerance_residu_ 1.e-3
                permet_affichage_  0
             fin_parametres_reglage_Algo_Newton_
    ISOELAS
      100000  0.3
fin_loi_contrainte_plane  # --- fin de la loi de contrainte plane

#------------------------------------------------------------------
masse_volumique
 E_to    1.00
#------------------------------------------------------------------
charges
#------------------------------------------------------------------
blocages
#------------------------------------------------------------------
controle
SAUVEGARDE 1
#--------------------------
resultats  pas_de_sortie_finale_
COPIE 0
           _fin_point_info_
```

18 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 (69), 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 découles. 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 69.1). 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)

19 Remarques concernant les variations de pas de temps

Les différents schémas d'avancement temporel sont établis en général pour un pas de temps constant.

Supposons qu'entre deux pas de temps, celui-ci varie, il serait nécessaire d'adapter les formules obtenues pour un pas de temps constant. Deux cas sont considérés : soit il s'agit d'un calcul implicite soit d'un calcul explicite.

Cas d'un calcul explicite

En général le pas de temps est petit voir très petit. On considère alors que la variation de pas de temps n'est pas trop importante et que sur l'ensemble des pas de temps traités, il y a peu de changement de pas de temps relativement. En conséquence, on peut faire l'hypothèse que l'impact d'un changement de pas de temps est négligeable sur la précision globale de la réponse.

Cas d'un calcul implicite

En général le pas de temps peut-être important. On suppose alors que l'algorithme intègre la prise en compte d'un changement de pas de temps de manière précise.

C'est le partie pris actuellement retenue pour Herezh. Cela amène 2 remarques :

1. en explicite, dans le cas de nombreux changements de pas de temps durant le calcul, il est possible que cela impact la solution finale. Donc il faut garder ce point en tête lors de l'analyse des résultats,
2. dans le cas d'un temps maxi fixé en dynamique explicite, contrairement aux schémas implicites (ou quasi-statiques), le calcul s'arrête dès que le temps en court est suffisamment proche du temps final (à la précision près). Il n'y a pas de modification du dernier pas de temps pour finir exactement au temps fin demandé. Pour l'implicite, au contraire, le dernier pas de temps est modifié pour obtenir exactement le temps final.

20 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 a noté 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 réservé 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.20.1) : de l'amortissement cinétique ou de l'amortissement visqueux de préférence proche de l'amortissement critique (cf. 20.3.9) . 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 croitre, 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.20.3). Enfin là également il y a le choix entre l'amortissement cinétique et l'amortissement visqueux critique approché.

20.1 Utilisation d'algorithmes classiques avec de l'amortissement cinétique

Après avoir définit l'algorithme classique, on introduit à la fin des paramètres de l'algorithme un certain nombre de mots clé. La table (20) 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_cinetique_ " : mot clé facultatif, le nombre minimal d'itérations a partir duquel on démarre l'algorithme d'amortissement cinétique, par défaut = 1 ;

— " max_nb_decroit_pourRelaxDyn_ " : mot clé facultatif, indique combien de fois l'énergie cinétique doit diminuer, pour valider la mise en place de la relaxation, par défaut = 1. Par exemple si = 3, cela signifie qu'il faut attendre 3 baisses de l'énergie cinétique pour valider la relaxation. Ces 3 baisses peuvent apparaître au cours de "n" itérations, "n" étant plus grand que 3 ! De plus pendant ces "n" itérations, on peut également avoir des élévations de l'énergie cinétique suivi de diminutions. Seules les diminutions sont prises en compte. Après une relaxation, le compteur de diminution est remis à 0.

— " inter_nb_entre_relax_ " : mot clé facultatif, indique le nombre minimal d'itération accepté entre deux relaxations, par défaut = 1. Par exemple si = 10, cela signifie qu'après une relaxation, une nouvelle relaxation ne sera accepté qu'après 10 nouvelles itérations. Il est également possible de piloter ce nombre à l'aide d'une fonction nD. Pour cela, on indique à la place du scalaire, le mot clé
nom_fct_nD_inter_nb_entre_relax_ suivi (toujours sur la même ligne) du nom (une chaine de caratères) de la fonction nD. Par exemple :
inter_nb_entre_relax_ nom_fct_nD_inter_nb_entre_relax_ fonctionToto

— " 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 pics 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 inférieur 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_cinétique_ ” : mot clé obligatoire indiquant la fin des paramètres

NB : Pour les paramètre ” mode_debug_= ” et ” ARRET_A_EQUILIBRE_STATIQUE_ ” voir 5 pour un exemple et 14 , 20.3.11 pour plus de précision.

Chaque paramètre doit-être sur une ligne différente !

TABLE 20 – 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
```

20.2 Utilisation d'algorithmes classiques avec de l'amortissement visqueux critique

Après avoir définit 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 (21) donne un exemple de déclaration. On se reportera à 20.3.9.

NB : Pour les paramètre " mode_debug_= " et " ARRET_A_EQUILIBRE_STATIQUE_ " voir 20.3.11 et 20.4.

Chaque paramètre doit-être sur une ligne différente !

TABLE 21 – 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
```

20.3 Algorithme de Relaxation dynamique

L'algorithme se décline en plusieurs versions, en particulier trois types d'amortissement principaux peuvent être invoqués : cinétique, visqueux critique ou un mixte des deux. 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. 20.3.1) Il s'agit d'une relaxation avec amortissement cinétique, qui utilise différente techniques pour le calcul de la pseudo-masse. Ces techniques sont différenciéeé a 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.20.3.2) Il s'agit d'une relaxation avec amortissement visqueux critique, qui utilise les mêmes technique de calcul de la pseudo-masse que le cas typeCalRelaxation= 1 par contre l'amortissement est différent car visqueux critique.
3. typeCalRelaxation= 4 : (cf.20.3.3) Il s'agit d'une relaxation avec amortissement mixte qui démarre en cinétique et ensuite bascule sur un amortissement visqueux critique,

— suivi des paramètres principaux détaillés en 20.3.1 et 20.3.2

À la suite de ces paramètres, il peut-être intéressant (en particulier pour typeCalRelaxation= 4) d'indiquer une méthode de recalcule de la matrice masse différente pour l'amortissement cinétique et l'amortissement visqueux. Si oui, sur la même ligne, en dernier, on indique le mot clé opt_visqueux_recal_mass= suivi du type de recalcule pour l'amortissement visqueux. Bien noter que la présence du paramètre opt_visqueux_recal_mass= est prise en compte quelque soit la valeur de typeCalRelaxation= . Dans le cas de typeCalRelaxation= 4, et dans le cas où le paramètre opt_visqueux_recal_mass= existe, c'est typeCalRelaxation= qui est utilisé pendant la phase de relaxation par amortissement cinétique et c'est opt_visqueux_recal_mass= qui est utilisé pendant la phase de relaxation par amortissement visqueux (voir exemple 22).

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.20.3.4),
2. éventuellement les paramètres de contrôle du re-calcul de la matrice masse (cf. 20.3.7),
3. éventuellement les paramètres permettant de contrôler le calcul de la matrice visqueuse (cf. 20.3.9),
4. éventuellement le paramètre de contrôle du mode "debug" (cf. 20.3.11),
5. éventuellement les paramètres de contrôle de l'amortissement cinétique (voir 20.1 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 20.4 pour plus d'informations).
7. éventuellement un paramètre particulier de contrôle du contact voir (20.3.13) pour plus d'informations.

Le paramètre " typeCalRelaxation= " est par défaut : 1

TABLE 22 – Exemple de déclaration de paramètres de l'algorithme de relaxation dynamique en amortissement mixte avec un paramètre de recalcule de la matrice masse différent en cinétique et en visqueux

```
 PARA_TYPE_DE_CALCUL
typeCalRelaxation= 4   lambda= 1  type_calcul_mass= 2  option_recalcul_mass= 3 \
opt_visqueux_recal_mass= 0
```

20.3.1 Cas d'une relaxation avec amortissement cinétique

Exemple de declaration :
typeCalRelaxation= 1 lambda= 0.7 type_calcul_mass= 1 option_recalcul_mass= 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_mass= 1 : permet de choisir le type de calcul de la matrice de pseudo-masse
4. option_recalcul_mass= 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 : 23)

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

— typeCalRelaxation= 2 ;
— lambda= 0.605 ;
— type_calcul_mass= 2 ;
— option_recalcul_mass= 0 ;

La table 23 donne un exemple de déclaration.

20.3.2 Cas d'une relaxation avec amortissement visqueux

Exemple de declaration :
typeCalRelaxation= 2 lambda= 0.7 type_calcul_mass= 2 option_recalcul_mass= 4

1. typeCalRelaxation= 2, indique qu'il s'agit d'une relaxation avec amortissement visqueux. On se reportera à 24 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

TABLE 23 – 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
```

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 23 donne un exemple de déclaration.

20.3.3 Cas d'une relaxation mixte amortissement cinétique ou visqueux

L'objectif est ici de combiner les deux amortissements avec l'idée que dans le cas de grands déplacements solides, l'amortissement cinétique est en général plus performant que l'amortissement visqueux. Par contre lorsque l'on approche de l'équilibre, l'amortissement visqueux permet (a priori) une évolution plus régulière vers la solution ce qui peut se révéler intéressant pour des lois de comportement complexe par exemple.

La table 25 donne un exemple de déclaration d'algorithme mixte dont les paramètres spécifiques sont :

TABLE 24 – 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
```

- " typeCalRelaxation= 4" : indique le type 4 qui correspond au choix de l'algorithme mixte,
- " proportion_cinetique= 0.1" : permet de déterminer le seuil de la bascule entre l'amortissement cinétique et l'amortissement visqueux. Au début du calcul l'amortissement est cinétique. Lorsque 0.1× la précision de l'équilibre atteint est inférieure à la norme visée, l'amortissement cinétique est remplacé par l'amortissement visqueux. La norme visée est définie avec le mot clé " NORME " au niveau des paramètres généraux (cf. 72).

Remarque importante : Le mot clé " proportion_cinetique= " doit apparaître avant la déclaration des paramètres relatifs à l'amortissement visqueux.

À noter qu'une fois que l'algorithme est passé en amortissement visqueux, au cours d'un incrément, il reste en amortissement visqueux quelque soit l'évolution du contrôle du basculement de cinétique à visqueux. Ceci a pour but d'éviter des oscillations. À chaque début d'incrément, l'algorithme repasse par défaut en amortissement cinétique. Il est possible néanmoins de rester en visqueux grâce à un pilotage via une fonction nD qui regarde le numéro d'incrément (cf. plus bas le mot clé propCinetiqueAvecPonderation_Globale_).

Il est possible de contrôler le basculement de cinétique à visqueux à l'aide d'une fonction dépendante du temps et/ou d'une fonction nD. Mots clés :

- propCinetiqueAvecPonderation_temps_ pour une dépendance spécifique au temps,
- propCinetiqueAvecPonderation_Globale_ pour une fonction nD

Le mot clé doit-être suivi du nom de la fonction (qui sera ensuite dans le fichier .info). La table 26 donne un exemple de déclaration de ce type de contrôle. Bien noter que cette fonction est censée produire un booléen : c'est-à-dire soit une grandeur supérieure à 1 qui sera alors considérée comme vrai, soit 0 ou compris entre 0 et 1 (exclus) qui sera

TABLE 25 – 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= 4   lambda= 0.6   type_calcul_mass= 2  option_recalcul_mass= 0
parametre_calcul_de_la_masse_   casMass_relax= 3
proportion_cinetique= 0.1
parametre_calcul_de_la_viscosite_  type_calcul_visqu_critique= 2 \
opt_cal_C_critique= 1  f_= 0.9

mode_debug_= 100

avec_amortissement_cinetique_
   max_nb_decroit_pourRelaxDyn_ 1
   coef_arret_pourRelaxDyn_ 0#0.001
   coef_redemarrage_pourRelaxDyn_ 0.0
   max_deltaX_pourRelaxDyn_ 0.005
   nb_max_dX_OK_pourRelaxDyn_ 10
   nb_deb_testfin_pourRelaxDyn_ 250
fi_parametre_amortissement_cinetique_

ARRET_A_EQUILIBRE_STATIQUE_ 2
```

alors compris comme faux. Vrai conduira à rester en cinétique, faux conduira à passer en visqueux.

Remarque Pendant le calcul, il est possible d'accéder à la variable globale `amor_cinet_visqueux` qui indique à tout moment si l'amortissement est en cinétique (la variable vaut alors 1) ou en visqueux (la variable vaut alors 0). On peut ainsi piloter une fonction nD avec en interne dans la fonction, un test sur la variable `amor_cinet_visqueux` , ce qui peut être utile pour le calcul de λ par exemple.

TABLE 26 – Exemple de déclaration de paramètres de l'algorithme de relaxation dynamique avec amortissement mixte contrôlé par une fonction

```
dynamique_relaxation_dynam avec plus visualisation

PARA_TYPE_DE_CALCUL
# .....................................................
# / type d'algorithme /
#.....................................................
# variable_emplacement
typeCalRelaxation= 4   lambda= 0.6   type_calcul_mass= 2  option_recalcul_mass= 0
parametre_calcul_de_la_masse_   casMass_relax= 3

proportion_cinetique= 0.1
#propCinetiqueAvecPonderation_temps_   co_depend_temps
propCinetiqueAvecPonderation_Globale_  fct_var_glob

parametre_calcul_de_la_viscosite_  type_calcul_visqu_critique= 2\
          opt_cal_C_critique= 1  f_= 0.9

mode_debug_= 100

avec_amortissement_cinetique_
   max_nb_decroit_pourRelaxDyn_ 1
   coef_arret_pourRelaxDyn_ 0#0.001
   coef_redemarrage_pourRelaxDyn_ 0.0
   max_deltaX_pourRelaxDyn_ 0.005
   nb_max_dX_OK_pourRelaxDyn_ 10
   nb_deb_testfin_pourRelaxDyn_ 250
fi_parametre_amortissement_cinetique_

ARRET_A_EQUILIBRE_STATIQUE_ 2
```

20.3.4 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 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 20.3.5.

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 20.3.6.

20.3.5 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'Herezh++, 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 :

$$k_{imax} = \sum_e \frac{ep}{4} \left(\alpha\ K + \beta\mu + \gamma \frac{\boldsymbol{I}_\sigma}{3} + \frac{\theta}{2}\ \sigma_{mises}) \right) \tag{19}$$

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 (19) 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 (19)

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

(c) =3 : on retient la valeur moyenne de (19) :

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

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

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

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 linéïque, la masse est calculée selon une formule équivalente :

$$k_{imax} = \sum_{e} \frac{l_e}{4} \left(\alpha \ K + \beta\mu + \gamma \frac{\boldsymbol{I}_\sigma}{3} + \frac{\theta}{2} \ \sigma_{mises}) \right) \tag{20}$$

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. 20.4 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 (27) donne un exemple d'utilisation de l'algorithme dans le cas d'un critère sur le résidu, tandis que la table (28) 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 ;

Post traitement : Il est possible de visualiser les isovaleurs "scalaires" de la valeur de la masse à chaque noeud à l'aide de la grandeur masse_relax_dyn définie au niveau des ddl_etendu_aux_noeuds

Remarque : Noter que dans le cas de l'utilisation de casMass_relax= 2 , en post-traitement c'est un vecteur que l'on obtient et que le mot clé est alors MASSE_RELAX_DYN défini au niveau des GrandEvoluee_noeud

TABLE 27 – 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 28 – 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_
```

20.3.6 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 :

$$m_i = \frac{\lambda(\Delta t)^2}{2} S_i \tag{21}$$

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" :

$$S(a_i) = \sum_{e}^{NBE} \left\{ \sum_{j=1,b=1}^{n,dim} |K_e\ (a_i, b_j)| \right\} \tag{22}$$

Ici on fait une somme des valeurs propres de chaque matrice de raideur locale K_e ce qui doit conduire à une majoration des valeurs propres via le théorème de Gershgorin appliqué localement à chaque matrice.

— " casMass_relax= 3" :

$$S(c_i) = MAX_{a=1}^{3} \sum_{e}^{NBE} \left\{ \sum_{j=1,b=1}^{n,dim} |K_e\ (a_i, b_j)| \right\} \tag{23}$$

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 (majoration des 3 valeurs propres obtenue via le théorème de Gershgorin appliqué localement à chaque matrice).

— " casMass_relax= 4" :

$$S(a_i) = MAX \left(2.\ |K(a_i, a_i)|, \sum_{e}^{NBE} \left\{ \sum_{j=1,b=1}^{n,dim} |K_e\ (a_i, b_j)| \right\} \right) \tag{24}$$

Correspond au maximum entre le cas " casMass_relax= 2" et 2 fois le cas " casMass_relax= 0"

— " casMass_relax= 5" :

$$S(c_i) = MAX_a \left(MAX \left(2.\ |K(a_i, a_i)|, \sum_{e}^{NBE} \left\{ \sum_{j=1,b=1}^{n,dim} |K_e\ (a_i, b_j)| \right\} \right) \right) \tag{25}$$

$c = 1, dim$; Correspond au maximum entre le cas " casMass_relax= 3" et 2 fois le cas " casMass_relax= 1"

— " casMass_relax= 6" : Ici on utilise la matrice de raideur K totalement calculée.

$$S(a_i) = \sum_{j=1,b=1}^{n,dim} |K(a_i, b_j)| \tag{26}$$

équivalent du cas casMass_relax= 2. On utilise la forme classique du théorème de Gershgorin.

— " casMass_relax= 7" :

$$S(c_i) = MAX_{a=1}^{3} \sum_{j=1,b=1}^{n,dim} |K(a_i, b_j)| \tag{27}$$

avec c=1,dim. équivalent du cas casMass_relax= 3 mais ici avec la matrice de raideur complètement calculée. 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= 8" :

$$S(a_i) = MAX\left(2.\ |K(a_i, a_i)|,\ \sum_{j=1,b=1}^{n,dim} |K(a_i, b_j)|\right) \tag{28}$$

équivalent du cas casMass_relax= 4 mais ici avec la matrice de raideur complètement calculée.

— " casMass_relax= 9" :

$$S(c_i) = MAX_a\left(MAX\left(2.\ |K(a_i, a_i)|,\ \sum_{j=1,b=1}^{n,dim} |K(a_i, b_j)|\right)\right) \tag{29}$$

$c = 1, dim$; équivalent du cas casMass_relax= 5 mais ici avec la matrice de raideur complètement calculée.

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

— casMass_relax= 3

La table (23) donne un exemple d'utilisation.

Post traitement : Il est possible de visualiser les isovaleurs "vectorielles" aux noeuds, de la valeur de la masse à chaque noeud et dans chaque direction de coordonnée, à l'aide de la grandeur MASSE_RELAX_DYN définie au niveau des GrandEvoluee_noeud .

Remarque : Noter que dans le cas de l'utilisation de casMass_relax= 1 , en post-traitement c'est un "scalaire" que l'on obtient et que le mot clé est alors masse_relax_dyn défini au niveau des ddl_etendu_aux_noeuds .

20.3.7 Contrôle du re-calcul de la masse

Il y a tout d'abord le paramètre option_recalcul_mass= qui est indiqué sur la même ligne que la définition de l'algorithme (voir ex : 24). Ensuite on trouve le paramètre facultatif : parametre_recalcul_de_la_masse_ qui permet d'abonder

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

$$\epsilon = MAX_{i=1}^{nbddl}(\epsilon_i) \text{ avec } \epsilon_i = \frac{\lambda(\Delta t)^2}{2}\frac{|\Delta \ddot{X}_i|}{|\Delta X_i|} \tag{30}$$

   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 contrôle provient de l'amortissement visqueux critique)

   Exemple de déclaration : parametre_recalcul_de_la_masse_ fac_epsilon= 0.8

   par défaut fac_epsilon= 0.9

   NB : Si on veut également calculer la masse tous les n cycles il faut indiquer le mot clé ncycle_calcul= après l'utilisation du mot clé fac_epsilon=
7. option_recalcul_mass= 5 : la matrice masse est calculée au début du calcul (ou du restart) et ensuite elle reste fixe, même aux changement d'incrément.

Dans le cas d'un amortissement mixte (typeCalRelaxation= 4) il est prévue de recalculer la matrice masse juste au moment du basculement de l'amortissement cinétique à l'amortissement visqueux. Si on veut désactiver ce fonctionnement, il faut indiquer le

mot clé
et_pas_recalcul_masse_a_la_transition_ en dernier enregistrement du paramètre facultatif parametre_recalcul_de_la_masse_ , voici un exemple de déclaration :

```
parametre_recalcul_de_la_masse_ ncycle_calcul= 1152 et_pas_recalcul_masse_a_la_transition_
```

Il est possible de piloter le recalcul de la masse à l'aide d'une fonction nD, qui devra dépendre uniquement des grandeurs globales. Pour cela à la suite du mot clé
option_recalcul_mass= , à la place d'une valeur numérique on indique le mot clé
nom_fct_nD_option_recalcul_mass_ suivi du nom (une chaine de caractère) d'une fonction nD.

- La fonction doit évidemment exister,
- elle doit uniquement utiliser des grandeurs globales,
- la fonction est appelée en particulier à chaque itération, sa valeur de retour est interprété comme l'aurait été une valeur numérique fixe,
- l'utilisation d'une fonction nD est possible avec tous les types de relaxation : cinétique, visqueuse, et également mixte. En particulier si l'on a utilisé le mot clé
  opt_visqueux_recal_mass= (cf. 20.3) on peut également remplacer la valeur numérique qui suit par le mot clé nom_fct_nD_option_recalcul_mass_ suivi du nom d'une fonction nD éventuellement différente (ou non) de la première fonction (s'il y a une première fonction de définition générale de recalcul de masse)

20.3.8 Traitement du cas particulier de masse nulle

Lors du calcul de la matrice masse, il est possible dans certain cas, d'obtenir des masses nulles.

Par exemple, supposons un noeud lié à un ensemble d'éléments dont les contraintes sont entièrement relâchées. Dans ce cas la matrice de raideur associée à ces éléments est nulle ce qui se traduit par une raideur globale nulle d'où un calcul de matrice masse nulle.

Le fait d'avoir une masse nulle va conduire à une accélération infinie et donc une divergence du calcul. Pour contrecarrer cette divergence, deux techniques sont disponibles.

1. modification arbitraire des masses nulles
2. limitation des déplacements et vitesses induites par les masses nulles.

La suite du document présente ces deux méthodologies.

Modification arbitraire des masses nulles : Par défaut la masse nulle est laissée telle quelle. Cependant on peut intervenir de manière arbitraire au travers d'un paramètre de contrôle " choix_mini_masse_nul= " dont on présente un exemple d'utilisation dans le cas d'un amortissement cinétique :

```
typeCalRelaxation= 1   lambda= 2.6  type_calcul_mass= 2  option_recalcul_mass= 0
parametre_calcul_de_la_masse_   casMass_relax= 3
choix_mini_masse_nul= 1
```

et pour un amortissement visqueux :

```
typeCalRelaxation=  2  lambda=  15.  type_calcul_mass=  2  option_recalcul_mass=  0.
parametre_calcul_de_la_masse_  casMass_relax= 3
parametre_recalcul_de_la_masse_  fac_epsilon= 100.
choix_mini_masse_nul= 1
parametre_calcul_de_la_viscosite_  type_calcul_visqu_critique= 1 opt_cal_C_critique= 1 f_= 0.9
```

Remarque : On notera que le mot clé choix_mini_masse_nul= "doit" se trouver avant les paramètres de calcul de la viscosité. En fait l'ensemble des paramètres relatifs à la masse, est traité dans un bloc différent de celui relatif à la viscosité.

En fonction de la valeur du paramètre de " choix_mini_masse_nul= ", le fonctionnement est le suivant :

- " choix_mini_masse_nul= 0 " : cas par défaut, aucune modification des masses nulles,
- " choix_mini_masse_nul= 1 " : les masses nulles sont remplacées par la valeur maxi des masses virtuelles actuellement calculées,
- " choix_mini_masse_nul= 2 " : les masses nulles sont remplacées par la valeur moyenne des masses virtuelles actuellement calculées.

Limitation des déplacements et vitesses induites par les masses nulles : L'idée est ici de limiter les déplacements maxis sous forme d'un écrêtage à une valeur donnée. Par exemple sur l'exemple suivant on limite la valeur absolue de la composantes maxi de l'incrément de déplacement à 0.1 et à 0.15 pour l'incrément de vitesse. Le signe négatif est pour indiquer qu'il s'agit seulement des maxis qui sont concernés, c'est à dire toutes les composantes supérieures à la borne indiquée (ici 0.1).

```
 para_pilotage_equi_global ------------
#-------------------------------------------
# PARAMETRE              |       VALEUR       |
#-------------------------------------------
NORME_MAXI_X_INCREMENT  -0.1
NORME_MAXI_V_INCREMENT  -0.15
```

Ces bornes maxis permettent ainsi d'éviter une divergence brutale. Cependant leur présence ne garantie pas une convergence si, par exemple, aucun équilibre n'existe ! Mais l'idée est que ces limitations permettent de "passer" des étapes transitoires, dans lesquelles par exemple les structures sont totalement relachées avec utilisation d'une loi critère par exemple.

20.3.9 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.

$$[C] = c[M] \tag{31}$$

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

$$\omega_0^2 \approx \frac{\Delta X^T K^n \Delta X}{\Delta X^T M \Delta X} \tag{32}$$

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 :

$$\omega_0^2 \approx \frac{\Delta X^T \ {}^l K^n \ \Delta X}{\Delta X^T M \Delta X} \ avec \quad {}^l K_{ii}^n = \frac{\Delta R^n_{i(statique)}}{\Delta t \ \dot{X}_i^{n-1/2}} \tag{33}$$

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 : $(\omega_0)^2 \approx (\Delta X^T \Delta R^n_{i(statique)})/(\Delta X^T M \Delta X)$
2. `opt_cal_C_critique=` 1 : $(\omega_0)^2 \approx (\dot{X}^T \Delta R^n_{i(statique)})/(\dot{X}^T M \dot{X})$

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 defaut : $f = 0.9$

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 ;

20.3.10 Contrôle du paramètre λ

Trois mots clefs permettent de contrôler le paramètre λ (cf.21) en plus de sa valeur initiale donnée par le mot clef : " `lambda=` ". Il s'agit de :

— " `lambdaAvecPonderation_temps_` " : ce paramètre permet de pondérer la valeur de λ à travers l'utilisation d'une dépendance au temps (réel),

— " `lambdaAvecPonderation_Globale_` " : permet de pondérer la valeur de λ à travers l'utilisation d'une dépendance à des grandeurs globales,

— " `pilotageAutolambda_` " : introduit un pilotage automatique de λ dont l'objectif est de limiter des amortissements intempestifs.

Rappelons que le paramètre λ (cf.21) est en grande partie responsable de la stabilité du processus d'amortissement. Sur un incrément de temps, l'algorithme de relaxation dynamique utilise un pseudo pas de temps critique dont la valeur est arbitrairement $\Delta t = 1$. La masse est ensuite ajustée pour satisfaire la condition de Courant (C F L) via la relation 21 dans le cas de l'utilisation de la raideur réelle. L'objectif est que le pseudo pas de temps demeure inférieur à la limite de stabilité (c.-à-d. le pas de temps critique).

Néanmoins, il s'agit d'un calcul approché et la stabilité n'est jamais parfaitement garantie. Le paramètre λ est alors utilisé pour augmenter la valeur de la masse de manière à améliorer la stabilité. À noter que plus λ est grand, plus le calcul est stable, mais à l'inverse plus l'amortissement est faible d'où une convergence vers la solution stable plus longue. Il s'agit donc de trouver un compromis. Typiquement (cf. le manuel théorique d'Herezh++) la limite inférieure de λ est de l'ordre de 0.6 (en considérant une marge de 10% comme préconisée par Underwood).

Enfin rappelons qu'à l'intérieur d'un pas de temps physique (le temps qui sert pour piloter le chargement par exemple), chaque itération de convergence de l'algorithme de relaxation correspond à un pas de pseudo-temps = 1 (donc qui est totalement indépendant du temps physique).

Par défaut la valeur de λ utilisée pendant le calcul est celle donnée par le mot clé : " `lambda=` ". Cette valeur est fixe pendant tout le calcul.

Pendant le calcul, il est néanmoins possible si on le désire, de faire varier le paramètre λ à l'aide d'une des 3 possibilités suivantes chacune pilotée par un des 3 mots clefs présentés précédemment.

1. **Variation de λ au cours de l'évolution du temps physique** : Typiquement, l'idée est de pouvoir modifier la valeur de λ entre deux incréments de temps physique. Pour mettre en place ce pilotage, on utilise le mot clé " `opt_cal_C_critique=` " suivi du nom d'une courbe 1D. Supposons par exemple que la courbe s'appelle "evolLambda", à chaque début d'un nouvel incrément, la fonction est appelée et λ prend la valeur : $\lambda(t) = \lambda(0) \times evolLambda(t)$ avec $\lambda(0)$ la valeur initiale de λ indiquée après le mot clé : " `lambda=` ". La table (29) donne un exemple de déclaration.
2. **Variation de λ à l'aide d'une fonction nD** : Le fonctionnement est proche du cas précédent avec l'idée d'étendre les possibilités de contrôle de λ du fait que la fonction nD peut dépendre simultanément de plusieurs variables. Seules sont autorisées pour ces variables, les grandeurs globales. On se reportera à (88.1) pour leur liste exhaustive. Pour mettre en place ce pilotage, on utilise le mot clé " `lambdaAvecPonderation_Globale_` " suivi du nom d'une fonction nD (cf.81.2). La table (30) donne un exemple de déclaration.

   Contrairement au cas de la dépendance au temps, il est possible ici de faire varier λ pendant les itérations ce qui donne une large latitude de contrôle. Par exemple il peut-être intéressant de prévoir un λ différent en début et en fin de convergence.

   Bien noter que le λ utilisé au final dans le calcul vaut le λ introduit initialement, multiplié par la valeur de la fonction nD. Si on veut avoir la valeur de λ directement égale à la fonction nD, il faut donc indiquer un λ initial égal à 1.

TABLE 29 – Exemple de déclaration d'un algorithme de relaxation dynamique avec utilisation d'un λ qui varie en fonction du temps physique

```
TYPE_DE_CALCUL
#-----------------------------------
# probleme d'equilibre non dynamique
#-----------------------------------
dynamique_relaxation_dynam

PARA_TYPE_DE_CALCUL
# ----- KDR_temps
typeCalRelaxation= 1   lambda= 0.65   type_calcul_mass= 2  option_recalcul_mass= 0
parametre_calcul_de_la_masse_   casMass_relax= 3
lambdaAvecPonderation_temps_   evolLambda

# ----- Amortissement cinetique
avec_amortissement_cinetique_
.... (ici la liste des paramètres) ...
fi_parametre_amortissement_cinetique_

ARRET_A_EQUILIBRE_STATIQUE_ 2
...
... (définition des maillages) ...
...
#----------------------------
# Definition des courbes
#----------------------------
les_courbes_1D

evolLambda COURBEPOLYLINEAIRE_1_D
      Debut_des_coordonnees_des_points
         Coordonnee dim= 2 0.0  1.
         Coordonnee dim= 2 1.   3.
         Coordonnee dim= 2 2.   3.
      Fin_des_coordonnees_des_points
...
... (suite et fin de la mise en données)
...
```

TABLE 30 – Exemple de déclaration d'un algorithme de relaxation dynamique avec utilisation d'un λ qui varie à l'aide d'une fonction nD

```
TYPE_DE_CALCUL
#-------------------------------------
# probleme d'equilibre non dynamique
#-------------------------------------
dynamique_relaxation_dynam

PARA_TYPE_DE_CALCUL
# ----- KDR
typeCalRelaxation= 1   lambda= 0.65   type_calcul_mass= 2  option_recalcul_mass= 0
parametre_calcul_de_la_masse_   casMass_relax= 3
lambdaAvecPonderation_Globale_ fonct_cinetique

# ----- Amortissement cinetique
avec_amortissement_cinetique_
.... (ici la liste des paramètres) ...
fi_parametre_amortissement_cinetique_

ARRET_A_EQUILIBRE_STATIQUE_ 2
...
... (définition des maillages) ...
...
 les_fonctions_nD #------------

#    f(energie_cinetique)
fonct_cinetique FONCTION_EXPRESSION_LITTERALE_nD
  un_argument= energie_cinetique
  # si l'energie dépasse 1.e4 on augmente lambda
  fct= ((energie_cinetique > 1.e4) ? 1.1  : 1  )
fin_parametres_fonction_expression_litterale_
...
... (suite et fin de la mise en données)
...
```

3. **Pilotage automatique de λ en fonction de l'évolution du calcul** : Il s'agit d'un fonctionnement exploratoire avec pour objectif de trouver une méthode qui garantit automatiquement la stabilité du calcul. On observe dans la pratique que lorsque λ est proche de l'instabilité, il apparaît de multiples relaxations consécutives. Cela a pour conséquence de limiter drastiquement les déplacements et par là même une convergence correcte. Pour combattre ce phénomène, l'idée est ici d'observer l'évolution des relaxations consécutives et à partir d'un critère, d'augmenter automatiquement la valeur de λ dans le cas d'une suite d'amortissements intempestifs.

   Actuellement (version 6.789) le fonctionnement est le suivant. Soit la mise en données suivante :

   ```
   pilotageAutolambda_   lambda_min= 0.6 lambda_max= 7  delta_lambda= 0.1
   ```

   À chaque relaxation on sauvegarde l'itération correspondante dans une liste, de telle manière que le premier élément de la liste correspond à la dernière relaxation. Puis l'analyse suivante est menée :

   — si on a 1 relaxation tous les 2 itérations, 3 fois à suivre, cela veut dire que l'on a 3 relaxations sur 6 itérations, donc 3 enregistrements dans la liste de relaxations, tels que le (3ième - le premier) < 5 par exemple : ex de suite de relaxations : 14, 12, 10 $--> 14\text{-}10 = 4 < 5$. En fait on prend une marge et la limite réellement utilisée est 7 au lieu de 5. Si effectivement le (3ième - le premier) < 7, on suppose qu'il y a oscillation et on augmente λ d'un incrément qui a été donné par le mot clé "delta_lambda= ".

   — on étend le raisonnement à 1 relaxation tous les 3 itérations 4 fois à suivre, ou un mixte avec le premier cas. Le test implanté est alors si (4ième - le premier) < 10 alors on augmente λ d'un incrément.

   — Enfin, un dernier test similaire est implanté : si (5ième - le premier) < 21 alors on augmente λ d'un incrément.

   Le mot clé " `lambda_max=` " permet de limiter l'augmentation de λ. De même le mot clé " `lambda_min=` " constitue une borne mini qui n'est pas utilisée pour l'instant.

   La table (31) donne un exemple de déclaration.

TABLE 31 – Exemple de déclaration d'un algorithme de relaxation dynamique avec utilisation d'un λ piloté automatiquement

```
PARA_TYPE_DE_CALCUL
# ----- KDR
typeCalRelaxation= 1   lambda= 0.65   type_calcul_mass= 2  option_recalcul_mass= 0
parametre_calcul_de_la_masse_   casMass_relax= 3
pilotageAutolambda_   lambda_min= 0.6 lambda_max= 7  delta_lambda= 0.1
...
... (suite et fin de la mise en données)
...
```

20.3.11 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 32 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.

TABLE 32 – 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
```

20.3.12 Modulation de la précision d'équilibre globale

La précision sur l'équilibre global est par principe, gérée par les paramètres de contrôle globaux (cf. 72) avec comme paramètre principale une précision fixée par le mot clé PRECISION .

Il est possible d'ajouter une modulation de cette précision via la définition d'une fonction nD qui doit dépendre uniquement de variables globales (ex : temps, incrément, itération, énergie etc.). La valeur de la fonction est multipliée par la précision fixée par le mot clé PRECISION pour produire la précision réellement utilisée pour tester l'équilibre.

Pour indiquer le nom de la fonction nD à utiliser, on utilise le mot clé :
modulation_precision_equilibre_= suivi du nom de la fonction qui devra être déclaré par la suite dans le fichier .info

Le mot clé modulation_precision_equilibre_= se positionne une ligne soit avant soit après le mot clé mode_debug_= (cf. 20.3.11).

20.3.13 Contrôle particulier du contact avec l'algorithme de relaxation

Le contact peut-être très consommateur de temps calcul, aussi dans le cas d'un calcul implicite, par défaut la vérification de l'apparition ou de la suppression de nouveau contact ne s'effectue qu'après la convergence de chaque incrément. Par contre dans le cas d'un algorithme explicite, cette mise à jour du contact est effectuée à chaque incrément de temps, mais comme dans ce cas il n'y a pas d'itération, cette fréquence de vérification est très grande.

Dans le cas de l'algorithme de relaxation, on peut considérer qu'il s'agit d'un calcul implicite ou d'un calcul explicite, en fait c'est les deux en même temps sous des aspects différents. Ainsi au niveau du contact, par défaut le contact est recherché à chaque incrément explicite, ce qui correspond à chaque itération de l'équilibre, mais on peut modifier cet état à l'aide d'un paramètre de contrôle. Un exemple typique est donné par la table (33). On observe le mot clé : " parametre_activation_du_contact_ " suivis de " type_activation_contact= " et d'un entier ici 1.

- type_activation_contact= 0 : la recherche de nouveaux contacts est activée à la fin de chaque incrément "
- type_activation_contact= 1 (valeur par défaut) : la recherche de nouveaux contacts est effectuée à la fin de chaque itération "
- type_activation_contact= 2 : la recherche de nouveaux contacts est effectuée après chaque amortissement cinétique et à la fin de chaque incrément (s'il n'y a pas d'amortissement cinétique c'est équivalent au cas = 0)

20.4 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 :

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,

TABLE 33 – Exemple de déclaration pour l'algorithme de relaxation dynamique avec un paramètre de gestion du contact

```
dynamique_relaxation_dynam   #avec plus visualisation

PARA_TYPE_DE_CALCUL
#-------------------------------------
#/type d'algorithme/
#-------------------------------------

typeCalRelaxation= 1 lambda= 0.7 type_calcul_mass= 2 option_recalcul_mass= 0
parametre_calcul_de_la_masse_  casMass_relax= 3
parametre_recalcul_de_la_masse_    fac_epsilon= 100
parametre_activation_du_contact_            type_activation_contact=  1
mode_debug_= 50

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
```

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 (32) pour un exemple de déclaration.

L'utilisation de ce paramètre est dédiée aux algorithmes dynamiques explicites classiques (sauf Runge Kutta cf.16 et les algorithme de type Galerkin-discontinu) . Dans le cas des algorithmes non concerné il est a priori ignoré!!

20.5 Remarque sur l'organisation des paramètres

Compte tenu du fait que les paramètres liés à l'amortissement peuvent-être utilisés pour d'autres algorithmes que la relaxation dynamique, l'organisation des paramètres doit-être la suivante :

— déclaration du type d'algo de relaxation, par exemple :
" typeCalRelaxation= 4 lambda= 1 type_calcul_mass= 2 option_recalcul_mass= 0"

— déclaration dans l'ordre que l'on veut et au choix :
" parametre_recalcul_de_la_masse_ "

" parametre_calcul_de_la_masse_ "
" parametre_activation_du_contact_ "
" choix_mini_masse_nul= "
" proportion_cinetique= "
" propCinetiqueAvecPonderation_temps_ "
" propCinetiqueAvecPonderation_Globale_ "
" lambdaAvecPonderation_temps_ "
" lambdaAvecPonderation_Globale_ "
" pilotageAutolambda_ "

— puis déclaration du (ou des) paramètre(s) de l'amortissement, et/ou mode débug, et/ou type d'arrêt (dans l'ordre que l'on veut). Les paramètres propres à l'amortissement doivent suivre la déclaration du type d'amortissement avant de déclarer un autre amortissement ou un des autres paramètres si dessous :
" parametre_calcul_de_la_viscosite_ "
" avec_amortissement_cinetique_ "
" avec_au_noeud_amortissement_cinetique_ "
" ARRET_A_EQUILIBRE_STATIQUE_ "
" mode_debug_= "
" modulation_precision_equilibre_= "

21 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.

21.1 Sauvegarde des maillages en cours

À la suite de l'utilitaire (mais également pour les autres algorithmes) on peut utiliser le sous-mot clé "sauveMaillagesEnCours " pour sauvegarder le maillage en cours. L'intérêt est par exemple de sauvegarder des maillages après avoir effectué différentes opérations sur le maillage, par exemple après un déplacement solide, une opération de renumérotation, de suppression de noeuds non référencés, etc.

Remarque

1. Bien noter que le nom de fichier utilisé pour sauvegarder un maillage sera constitué à l'aide du "nom de maillage" indiqué en début du maillage. Si ce nom est généré automatiquement, en général il est identique à celui du fichier qui contient le maillage. Dans ce cas la sauvegarde va écraser le fichier initial ! Aussi, si l'on veut un nouveau fichier il est important de changer le nom du maillage en conséquence.

2. Dans le cas où le sous-mot clé "sauveMaillagesEnCours " est utilisé après un algorithme de calcul (par exemple un calcul statique ou dynamique mécanique), alors c'est la position finale du (ou des) maillage(s) qui est sauvegardée. Cela constitue une des méthodes simples pour obtenir un maillage déformé.

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

21.2 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 modifié, aussi toutes les références initiales sont modifiés 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 tout action. Cependant les informations autres que celles directement liées aux maillages, ne sont pas utilisées.

21.3 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.

21.4 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
```

21.5 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 30), 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".

21.6 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
```

Lorsqu'il y a plusieurs maillages le traitement est un peu plus complexe :

1. l'ensemble des maillages est tout d'abord globalisé,

2. les conditions linéaires sur chaque maillage et entre les maillages, sont prises en compte,
3. ensuite une première passe d'optimisation globale est effectuée ce qui conduit à une numérotation globale unique, optimisée,
4. enfin une deuxième passe de traitement conduit à transformer cette numérotation globale en une numérotation spécifique pour chaque maillage.

En résumé, dans le cas où plusieurs maillages sont présents dans le fichier .info, l'opération concerne systématiquement l'ensemble des maillages.

Remarque

— Il est également possible d'effectuer cette opération, juste après la lecture du maillage (cf 31), 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". Par contre la renumérotation globale de tous les maillages n'est pas possible au moment de la lecture.

— Bien noter que l'utilisation de l'utilitaire, permet d'optimiser la numérotation en tenant compte des conditions initiales linéaires éventuelles.

— Dans le cas où il y a plusieurs maillages. Il faut absolument que des liaisons existent (via des conditions linéaires par exemple) entre ces différents maillages, pour que l'algorithme de renumérotation fonctionne. Si les maillages sont totalement séparés, alors seules les renumérotations indépendantes pour chaque maillage sont possibles.

21.7 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.

Important : Il est nécessaire d'utiliser la même dimension d'espace de travail, pour la vérification "et" pour l'utilisation par la suite du maillage. En effet, supposons que l'on utilise des éléments 2D (type triangle ou quadrangle). En dimension d'espace de travail 2D, le jacobien est calculé par projection sur l'axe z, du produit vectoriel des vecteurs de base de l'élément. En dimension d'espace de travail 3D, l'axe z n'est pas pertinent, en fait l'orientation des éléments peut-être quelconque, donc le jacobien se calcul par le produit scalaire du produit vectoriel des 2 vecteurs tangents à l'élément 2D et de la normale à l'élément, ce qui constitue le produit mixte qui est toujours positif compte tenu de la manière où il est calculé. Ainsi le jacobien est toujours positif. En fait ce qui est important en espace 3D, c'est la variation des normales d'un élément à l'autre, ce qui est réalisé ou vérifié par l'item "orientation des faces d'élément 2D" du menu (cf. 21.7.1). Car dans le cas où deux éléments 2D sont contigus, en espace 3D, avec une normale inversée en sens, cela conduit à des efforts locaux généralisés et des raideurs qui se retranchent au lieu classiquement de s'ajouter !

21.7.1 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 $< 50°$ 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'élément sont définit 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 terminé. 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 ?
```

21.8 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 transformées 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 chaine "_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".

21.9 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 fusionnés s'ils appartiennent à des éléments différents,

— et à un même maillage.

Les noeuds supprimés du maillage initiale, sont également remplacés par les nouveaux noeuds dans les 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.

21.10 Fusion d'éléments supperposés

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 remplacés par les éléments identiques restants, dans les listes de références d'éléments, et également les 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.

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.

21.11 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 :

  — soit la référence initiale a un nom différent de toutes les autres références, dans ce cas, elle conserve son nom

  — soit plusieurs références ont le même nom. Dans ce cas, la première référence (dans l'ordre de lecture du fichier .info) conserve son nom, puis la seconde a un suffixe "$_i$" ajouté, $i = 1$ pour la deuxième apparition, $= 2$ pour la troisième, etc. Le programme affiche dans le terminal la modification des noms de référence si le niveau d'affichage est différent de 1. On peut donc récupérer dans un fichier log les nouveaux noms avec un ordre de redirection du type "ordre de lancement d'herezh" suivi de | tee nom.log . Cet ordre permet d'avoir l'affichage à l'écran + la sauvegarde de l'affichage dans le fichier "nom.log"

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 nfus)
 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.

21.12 Création d'un sous maillage à partir d'une référence d'éléments

L'option a pour objectif de créer un nouveau maillage construit à partir d'une référence d'éléments, que nous appellerons par exemple "E_ref", concernant un maillage particulier. Le nouveau maillage correspond à une copie de l'ancien maillage qui contient la référence d'éléments "E_ref". Dans ce nouveau maillage, tous les éléments qui ne sont pas référencés dans "E_ref" ont été supprimés. De même tous les noeuds qui ne sont pas référencés par un élément conservé, sont supprimés. Les références du nouveau maillage correspondent aux références de l'ancien maillage, mises à jour en fonction des suppressions.

La syntaxe de ce type de calcul est la suivante :

```
#------------------------------
# definition du type de calcul |
#------------------------------
 TYPE_DE_CALCUL

utilitaires   plus cree_sous_maillage
```

Après lecture des maillages, le programme propose différents choix :

```
========== choix de la reference d'elements       ===============
 liste des ref d'elements disponibles                 (rep list)
 nom de la ref d'element du sous maillage             (rep nref)
 nom du maillage associe a la ref (defaut: le premier)(rep noma)
 nom du nouveau maillage                              (rep nmai)
 fin (appel de la methode de creation)                (rep    f)
 reponse ?
```

- "list" : permet d'obtenir la liste des références d'éléments actuellement disponibles,
- "nref" : permet de spécifier la référence d'éléments qui permettra de créer le sous-maillage,
- "noma" : dans le cas où il y a plusieurs maillages, permet de spécifier le maillage auquel la référence d'éléments est associée,
- "nmai" : permet de spécifier le nom du sous-maillage qui va être construit. Dans le cas où ce nom est déjà utilisé par un maillage existant, le nom final du sous-maillage comprendra le suffixe "_nevez".

22 Informations

22.1 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.37).

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 (34) 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 34 – 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 certaine soit toutes les indiquer). Pour le cas particulier des référence 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 différent 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 concernants 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.

22.2 Frontières

Cette option permet de générer automatiquement des frontières pour les maillages lus. La syntaxe de ce type de calcul est la suivante :

```
TYPE_DE_CALCUL
```

```
  #----------------
  #  TYPE DE      |
  #  CALCULS      |
  #----------------
informations  avec plus frontieres
```

À la suite du calcul des frontières, ces dernières sont enregistrées dans un fichier avec le suffixe : "i_front.faces", ou "i" est le numéro du maillage.

Les frontières ainsi déterminées sont calculées de manière automatique selon le critère suivant :

— une arête ou une surface est considérée comme une frontière si elle n'est pas commune à plusieurs éléments.

Remarque Avec ce critère simple, par exemple toutes les arêtes doubles ne sont pas considérées comme des frontières. Lorsque l'on utilise des éléments superposés volontairement, cela conduit automatiquement à supprimer les arêtes des éléments. Sachant que les frontières sont essentiellement utilisées pour le contact, lorsque l'on veut utiliser ces arêtes pour le contact, il faut alors imposer de les prendre en compte. On utilise alors une définition explicite des ces zones : cf. (65.3) et (65.4) pour une information détaillée.

22.3 Calculs géométriques

Cette option a pour objectif de fournir interactivement des informations géométriques calculées à partir des informations lues dans le .info

La syntaxe de ce type de calcul est la suivante :

```
#------------------------------
# definition du type de calcul |
#------------------------------
 TYPE_DE_CALCUL

informations avec plus calcul_geometrique
```

On obtient alors un menu où l'on choisit l'information désirée. Pour l'instant (version 6.797) un seul calcul est implanté : celui de la distance initiale entre deux noeuds.

23 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.

23.1 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 a 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 (35) donne un exemple de l'utilisation de l'algorithme.

24 Combinaison de plusieurs algorithmes de base : ex implicite et explicite

L'algorithme de mot clé : `combiner` permets d'enchaîner (ou de choisir entre) plusieurs algorithmes de base. La table (36) donne un exemple de base permettant d'enchaîner deux algorithmes de base.

NB : Par la suite les algorithmes internes seront appelés des sous-algorithmes.

D'une manière plus précise, l'algorithme `combiner` permet d'encapsuler et de contrôler plusieurs sous-algorithmes à l'intérieur d'un algorithme maître.

NB :

- Il n'y a pas de limitation sur le nombre de sous algorithme,
- un même type d'algorithme peut apparaitre plusieurs fois,
- les sous-algorithmes sont repérés par leur numéro d'apparition lors de la lecture (et non par leur type).

Par défaut le fonctionnement est le suivant :

- Pour chaque incrément de charge, l'ensemble des sous-algorithmes est exécuté dans l'ordre de la lecture des sous-algorithmes,
- le programme sort d'un sous algorithme lorsque typiquement la précision requise pour cet algorithme en fonction de la norme choisit, est satisfaite (à noter que chaque algorithme peut redéfinir sa propre norme et sa propre précision via des fonctions nD et/ou des paramètres de contrôle spécifique)
- seul le dernier algorithme valide (s'il y a eu convergence) le résultat final.

TABLE 35 – Exemple d'utilisation de l'algorithme de Bonelli.

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

#-------------------------------------------------------------------------------
#|  parametres (falcultatifs ) 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                                /
```

Il est possible d'étendre ce fonctionnement avec les paramètres facultatifs propres à l'algorithme combiner définit par le mot cle : PARA_TYPE_DE_CALCUL positionné après la définition de tous les sous algorithmes (cf. le mot clé fin_liste_algorithmes_a_combiner_),

— on peut définir une fonction nD, dépendante de grandeurs globales qui doit retourner 1 scalaire. Ce scalaire indique le choix du numéro du sous algorithme désiré. Dans ce

TABLE 36 – Exemple partiel de déclaration d'un algorithme combiner qui intègre 2 sous-algorithmes

```
  combiner  avec  plus visualisation
# ++++++++++++ liste des sous-algorithmes  ++++++++++++
  liste_algorithmes_a_combiner_ # mot cle de debut de liste
       # ============ algo 1 ============
       # ----- Dynamic Relaxation
        dynamique_relaxation_dynam
          # puis les parametres particuliers de l'algorithme de relaxation ...
          #  ... a completer
       # ============ algo 2 ============
       # ----- newton statique
        non_dynamique
        PARA_TYPE_DE_CALCUL
        mode_debug_= 1 # il s'agit ici du mode debug uniquement pour le non_dynamique
       # ...
        fin_liste_algorithmes_a_combiner_ # mot cle de fin de liste de sous algo

       # ++++++++++++ parametres de l'algorithme combiner  ++++++++++++
       #   ils sont facultatifs et ici commentes
        PARA_TYPE_DE_CALCUL
        ## --- gestion du choix de l'algo
       #   choix_algo_ fonc_choix_algo
       ##    --- gestion de la sauvegarde de chaque algo
       #   gestion_sauvegarde_ fonc_gestion_sauvegarde # idem sauvegarde
       ##     ---   puis la sortie du sous algo si convergence
       #   gestion_sortie_a_convergence_ fonc_gestion_sortie_a_convergence
```

cas, à chaque incrément de charge, la fonction est appelée et le nombre (entier) que retourne la fonction indique le numéro du sous algorithme à utiliser. En sortie du sous algorithme on examine s'il s'agissait d'un calcul avec validation ou non. Le sous algorithme ne rend la main que si le calcul s'est déroulé correctement (cf. remarques si dessous). Aussi si le calcul était avec validation, la charge est incrémentée sinon il n'y a pas d'incrémentation de la charge.

Puis la fonction de choix d'algorithme est appelée pour désigner le nouveau sous algorithme à appeler.

La définition de la fonction s'effectue avec le mot clé `choix_algo_` suivi du nom d'une fonction nD définie dans la suite de la mise en données.

exemple de définition :

`choix_algo_` fonc-choix-algo

où fonc-choix-algo est le nom d'une fonction nD qui doit être définie dans la suite de la mise en données.

— Par défaut, c'est le dernier sous algorithme qui valide le calcul. On peut définir

une fonction nD, dépendante de grandeurs globales, qui redéfinit quand le sous algorithme doit valider le calcul. Cette fonction est appelée à l'intérieur du sous algorithme à la fin de chaque itération (implicite) ou incrément (explicite) et permet à l'utilisateur (par exemple en fonction de la précision obtenue) de valider le calcul.

la définition de la fonction s'effectue avec le mot clé `gestion_sauvegarde_` suivi du nom d'une fonction nD définit dans la suite de la mise en données.

Exemple de définition :
`gestion_sauvegarde_` fonc-gestion-sauvegarde
où fonc-gestion-sauvegarde est le nom d'une fonction nD qui doit être définie dans la suite de la mise en données.

— Par défaut, le programme sort d'un sous algorithme lorsque typiquement la précision requise pour cet algorithme en fonction de la norme choisit, est satisfaite, ceci pour un incrément de charge. Il est possible d'effectuer plusieurs incréments de charge avec un même sous-algorithme. A priori il faut dans ce cas utiliser conjointement les fonctions de gestion de sauvegarde et de validation de calcul, et on utilise une troisième fonction nD qui indique quand il faut sortir du sous-algorithme.

La définition de la fonction s'effectue avec le mot clé :
`gestion_sortie_a_convergence_`
suivi du nom d'une fonction nD définie dans la suite de la mise en données.

Exemple de définition :
`gestion_sortie_a_convergence_` fonc-gestion-sortie-a-convergence
où fonc-gestion-sortie-a-convergence est le nom d'une fonction nD qui doit être définie dans la suite de la mise en données.

Autres remarques importantes :

— Un sous algorithme ne rend la main que s'il a fonctionné correctement. Par exemple s'il ne converge pas, il suit sa logique propre (cf. ses propres paramètres de contrôle) jusqu'à aboutir aux résultats ciblés (incrément, précision etc.). Aussi, tant que le sous-algorithme ne rend pas la main, un nouveau choix de sous-algorithme n'est pas possible. Cependant, chaque test de convergence du sous algorithme en cours peut se contrôler via une fonction nD qui prend en compte l'évolution du calcul. On peut ainsi définir une stratégie spécifique, qui permet de sortir du sous-algorithme.

— Pendant l'exécution de l'algorithme `combiner` on peut consulter (via une fonction nD) la variable globale :
`algo_global_actuel`
qui contient le numéro de code interne du sous-algorithme en cours. On se reportera à (9) dans la partie "Remarques", pour plus d'information concernant la signification du numéro.

— Au cours de la mise en données générale (c'est-à-dire via le .info) on définit un ensemble de paramètres de contrôle (ex : pas de temps, incrément, nombre d'itérations etc.)." Ces paramètres s'appliquent à l'algorithme principal c-a-d `combiner` . Par défaut, l'ensemble de ces paramètres de contrôle, est ré-attribué à chaque sous algorithme. Mais il est également possible (et souvent souhaitable) de modifier spécifiquement certains paramètre pour un algorithme particulier. Ceci s'effectue

entre les mots clés :
==PARA_SPECIFIQUE_SOUS_ALGO==
et
==FIN_PARA_SPECIFIQUE_SOUS_ALGO_==
Ces 2 mots clés permettent de définir des paramètres de contrôle spécifiques pour un sous algorithme. Ils doivent être indiqués après la définition des paramètres de contrôle généraux (paramètres par défaut de tous les sous-algorithmes). Le premier mot clé (==PARA_SPECIFIQUE_SOUS_ALGO==) doit être suivi du numéro de sous-algorithme auquel il s'applique. Le second mot clé (==FIN_PARA_SPECIFIQUE_SOUS_ALGO_==) termine la définition spécifique. La table (37) donne un exemple partiel de définition. La syntaxe des paramètres spécifiques est identique à celles des paramètres généraux (cf. 71 et par exemple plus particulièrement 72).

TABLE 37 – Exemple de mise en données spécifiques pour 2 sous-algorithmes.

```
     ...
 para_affichage
#---------------------------
# PARAMETRE    |    VALEUR    |
#---------------------------
 FREQUENCE_AFFICHAGE_INCREMENT    1
 FREQUENCE_AFFICHAGE_ITERATION    1
 FREQUENCE_SORTIE_FIL_DU_CALCUL   1
# ... fin des parametres generaux

 ==PARA_SPECIFIQUE_SOUS_ALGO== 1 # para specifiques sous algorithme 1
  controle #------------
 #--------------------
 # PARAMETER | VALUE |
 #--------------------
  NORME E_cinetique/E_statique_ET_ResSurReact
  PRECISION 5e-1
 etc.
 ==FIN_PARA_SPECIFIQUE_SOUS_ALGO_==

==PARA_SPECIFIQUE_SOUS_ALGO== 2 # para specifiques sous algorithme 2
 ....
==FIN_PARA_SPECIFIQUE_SOUS_ALGO_==
```

25 Introduction de champs de valeurs préexistantes aux calculs

En fait il y a plusieurs méthodes pour prendre en compte des champs de valeurs préexistantes aux calculs. La méthode qu'il faut privilégier est la définition de grandeurs aux noeuds ou aux éléments que l'on peut fixer via les conditions limites ou les conditions initiales. On se reportera à (67) pour plus d'informations.

Néanmoins il existe une méthode simplifiée pour lire deux types d'informations : les contraintes aux points d'intégration et les déplacements aux noeuds. Cette méthode est en association avec une méthode particulière de sauvegarde, voir les paragraphes (4) et (5) pour plus d'information sur cette méthode de sauvegarde et les formats de données associés.

Pour lire un champ de contraintes préexistantes au calcul, ou de déplacement, on utilise la syntaxe illustrée par l'exemple suivant.

```
# ------------------------------------------------------------------------
# ---lecture sur fichiers externes de grandeurs preexistantes au calcul---
# ------------------------------------------------------------------------

 flotExterne  #--- #  mot cle pour introduire la lecture

1 equilibre1_cab.isoe   barre5

2 equilibre1_dpl.points   barre5
```

On trouve successivement :

— sur la première ligne, le mot clé : " flotExterne"

— ensuite une suite de lignes (non limitée), qui commence par :

  — "1" : dans ce cas il s'agit d'un fichier de contraintes

  — "2" : dans ce cas il s'agit d'un fichier de déplacements

  Puis après le chiffre 1 ou 2, on a un nom de fichier qui est celui dans lequel le programme va lire les informations. Et enfin, un nom de maillage qui doit être cohérent avec un nom de maillage réellement utilisé dans le calcul. On peut ainsi prendre en compte plusieurs maillages, ou au contraire ne prendre en compte que certains maillages (lorsqu'il y en a plusieurs).

Ces informations doivent être positionnées dans le fichier .info juste avant la lecture du maillage, donc après la définition des paramètres particuliers de l'algorithme. La table (38) donne un exemple.

26 Introduction de sous-types de calculs

D'une manière générale, on peut associer au type principal de calcul, des sous-types qui complètent le type principal. Par exemple la table (38) définie le type principal de calcul

TABLE 38 – Exemple de positionnement dans le .info, de la lecture sur des fichiers externes, des contraintes aux points d'intégration et de déplacements aux noeuds.

```
....
#-------------------------------
# definition du type de calcul |
#-------------------------------
 TYPE_DE_CALCUL
non_dynamique  avec plus uniquement_remontee  plus visualisation


   flotExterne  #--- #  mot cle pour introduire la lecture
1 equilibre1_cab.isoe   barre5 # contraintes aux pti sur le maillage barre5
2 equilibre1_dpl.points   barre5 # déplacements aux noeuds sur le maillage barre5

#--------------------------------------
#| definition du (ou des) maillage(s) |
#--------------------------------------
 < barre5.her   # lecture du maillage barre5
 ....
```

" non_dynamique " et les deux sous types : " uniquement_remontee " qui permettra de calculer la "remontée" des contraintes aux noeuds, et "visualisation" qui permettra d'activer la présentation d'un menu interactif (cf. 80.3) à la fin du calcul. Quelques remarques concernant les sous-types :

1. Lorsque le type principal est suivi du mot "avec" , cela signifie que le calcul et les sous-types sont mis oeuvre (selon une méthodologie qui dépends de leur nature).
2. Lorsque le type principal n'est pas suivi du mot "avec", cela signifie qu'il n'y a pas de calcul relatif au type principal, ce sont seulement les sous-types qui sont activés. Supposons par exemple :

   ```
   ....
   #-------------------------------
   # definition du type de calcul |
   #-------------------------------
    TYPE_DE_CALCUL
   non_dynamique   plus uniquement_remontee  plus visualisation


      flotExterne  #--- #  mot cle pour introduire la lecture
   1 equilibre1_cab.isoe   barre5 # contraintes aux pti sur le maillage barre5
   2 equilibre1_dpl.points   barre5 # déplacements aux noeuds sur le maillage barre5

   #--------------------------------------
   #| definition du (ou des) maillage(s) |
   #--------------------------------------
    < barre5.her   # lecture du maillage barre5
    ....
   ```

   Dans ce cas, il y a lecture des valeurs de contraintes, des déplacements. Il n'y a pas de calcul d'équilibre. Il y a calcul de la "remontée" des contraintes aux noeuds puis

présentation du menu interactif de post-traitement.

3. Il peut y avoir plusieurs sous type, à condition que ce soit possible au niveau numérique. Chaque sous-type est séparé par le mot "plus".

TABLE 39 – Liste des différents sous types disponibles

identificateur	indications	ref éventuelle
uniquement_remontee	calcul la remontée des contraintes aux noeuds	83
remontee_et_erreur	remontée des contraintes aux noeuds + calcul d'un estimateur d'erreur	83
frontieres	calcul de frontières	22.2
LinVersQuad	création d'un maillage quadratique à l'aide d'un linéaire	ne concerne que les hexaèdres
QuadIncVersQuadComp	création d'un maillage quadratique complet à partir d'un incomplet	21.2
relocPtMilieuQuad	relocalisation des points milieux d'un maillage quadratique	21.3
visualisation	menus interactifs après le calcul	80.3
sauveCommandesVisu	création d'un fichier de commande de visualisation	80.3
lectureCommandesVisu	lecture d'un fichier de commande et exécution automatique	80.3
sauveMaillagesEnCours	sauvegarde du(des) maillage(s) en cours	21
creation_reference	création de référence	22.1
modif_orientation_element	modification de l'orientation des éléments	21.7
creationMaillageSFE	création des éléments SFE	21.8
suppression_noeud_non_references	suppression des noeuds non référencées	21.5
renumerotation_des_noeuds	renumérotation des noeuds	21.6
fusion_de_noeuds	fusion de noeuds	21.9
fusion_elements	fusion d'éléments	21.10
fusion_maillages	fusion de maillage	21.11

Quatrième partie
Maillages

27 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 structurées 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 frustre.
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 (22.1) pour la méthodologie à adopter pour créer des références.

28 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 chacunes des coordonnées des noeuds.

— de références de liste de noeud (cf.37) : 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 plurieurs 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.37). Ces listes fonctionnent comme celles des noeuds. Elles sont également optionnelles.

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

```
# un paralelepipede rectangle:
#  - longueur    1.0000000000000
#  - largeur    1.0000000000000
#  - hauteur    1.0000000000000
#  - 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 clartée 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 contiend effectivement le maillage
< cube.lis  # ce nom indique un fichier qui contiend 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 !

29 Mouvements solides

À la suite de la définition de chaque maillage, il est possible d'effectuer une suite de mouvements solides qui affectent 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.69).

La définition de mouvements solides s'effectue en indiquant à la suite des références 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 définie par un mot clé “ translation_= ” suivi par les composantes d'un vecteur déplacement.

Un nouveau centre de rotation est défini 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 définie 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 composante 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 voulu.

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

La table (40) donne un exemple de syntaxe pour définir deux mouvements solides.

TABLE 40 – Exemple de déclaration de deux mouvements solides : une translation 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 41 pour un exemple).

Il est également possible d'utiliser la position initiale 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 41 donne un exemple de déclaration.

En complément des mouvements solides, on peut également effectuer une opération d'homothétie. La table 42 donne un exemple de déclaration. Il ne s'agit pas d'un mouvement solide au sens mathématique, puisque la forme et le volume de la pièce finale changent par rapport à la pièce initiale. Néanmoins pour des facilités d'implémentation et a priori d'utilisation (via le positionnement dans le .info), la commande se situe dans la déclaration des mouvements solides. L'homothétie est réalisée de la manière suivante : soit $\vec{OC}$ la position du dernier centre défini par le mot clé ” centre_= ”, tous les points

TABLE 41 – 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_
```

$O\vec{M}$ sont transformés en des points $O\hat{\vec{M}}$ tels que :

$$X^i_{\hat{M}} = H^i \times (X^i_M - X^i_C) + X^i_C \tag{34}$$

Avec i = 1 à 3. Les trois composantes H^i de l'homothétie sont définies par le mot clé " homothétie_= ". Dans l'exemple 42, la pièce est ainsi dilatée de 0.005 % suivant x et y, et de 0.003% suivant z.

TABLE 42 – Exemple de déclaration d'une homothétie

```
def_mouvement_solide_initiaux_
  mouvement_solide_
     translation_= 0. 0. 3.e-3
     centre_= 0. 0. 0.
     homothetie_= 1.0005 1.0005 1.0003
 fin_mouvement_solide_
```

30 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 (43) 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 29).

Il est possible également d'utiliser un algorithme utilitaire, pour modifier définitivement le maillage (cf 21.5).

31 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. 30) et après l'application de mouvement solide (si c'est demandé cf 29), et après la création automatique de références de frontières (si c'est demandé cf. 37.1). La table (43) donne un exemple d'utilisation de renumérotation.

Il est possible également d'utiliser un algorithme utilitaire, pour modifier définitivement le maillage (cf 21.6).

TABLE 43 – Exemple de d'utilisation des méthodes de suppression des noeuds non référencés et de renumérotation

```
# importation du mailllage

< eprou1.her
< eprou1.lis

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

     suppression_noeud_non_references_

# demande de renumerotation

     renumerotation_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 qui tienne compte des liens pouvant exister entre les maillages il faut utiliser l'utilitaire (21.6). En effet il n'est pas possible de renuméroter l'ensemble des maillages sans l'existence de relation entre les maillages par exemple sous forme de relations linéaires. Or au moment de la lecture des maillages, les conditions linéaires ne sont pas encore lues !

Remarque importante Dans le cas où on utilise l'option de renumérotation, il n'est pas possible (actuellement) d'effectuer un "restart" à partir d'un fichier de résultat. Aussi, si l'on prévoit d'effectuer des "restarts", la solution est d'utiliser tout d'abord l'utilitaire (algorithme global cf. 21.6) de renumérotation de manière à obtenir un maillage optimisé et sauvegardé, préalablement au calcul. On peut également demander une renumérotation des pointeurs d'assemblages sans modification de la numérotation des numéros de noeuds (cf. OPTIMISATION_POINTEURS_ASSEMBLAGE au niveau des paramètres 75 et/ou le mot

clé OPTIMISATION_NUMEROTATION au niveau des paramètres de contact 77). Dans ce cas, il sera possible d'effectuer une opération de restart.

32 Fusion de noeuds très voisins

De manière similaire à l'utilitaire de fusion de noeud (cf. 21.9), 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 (44) 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 44 – 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
```

33 Fusion d'éléments supperposés

De manière similaire à l'utilitaire de fusion d'éléments superposés (cf. 21.10), 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 (45) donne un exemple de syntaxe pour supprimer des éléments superposés.

TABLE 45 – 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_
```

34 Suppression d'éléments à 2 ou 3 noeuds dont la distance est très proche

Il est possible de demander de supprimer les éléments à 2 ou 3 noeuds (segment, triangles) qui ont des noeuds proches, ceci durant la phase de lecture. Il n'y a pas de création d'un nouveau maillage (sauf si une demande de sauvegarde est faite), l'opération de suppression est donc réalisée à chaque lecture, avant le calcul. La suite du calcul s'effectuera alors sur le maillage dont les éléments repérés, auront été exclus. Les références sont mises à jours, en tenant compte des suppressions.

Dans le cas des triangles, la suppression d'un triangle impose de supprimer les éléments adjacents au coté dont les noeuds sont proches. Cette opération n'est possible "que " si l'élément adjacent est également de type triangle ou segment. Dans le cas où l'élément adjacent est différent (ex : quadrangle, élément volumique et.) il n'y a pas de suppression de l'élément !

La table (46) donne un exemple de syntaxe pour supprimer les éléments dont les noeuds ont une distance inférieure à 1.

On indique à la suite du mot clé " suppression_elements_2noeuds_tres_proches_ " la distance minimale en dessous de laquelle il y a suppression.

TABLE 46 – Exemple de d'utilisation de la méthode de suppression des éléments à 2 noeuds de distance inférieure à 1.

```
# -- def maillage
 < ../maillages/chapiteau_toile7.her

# demande de fusion des noeuds de distances inférieures a 0.001
suppression_elements_2noeuds_tres_proches_   1e0
```

35 Création d'une référence sur les noeuds non référencés

Lorsqu'un maillage comporte des noeuds qui ne sont pas connectés avec un élément, cela produit une erreur dans le cas d'un calcul d'équilibre par exemple, car aucune raideur n'est associée au noeud. En général une solution est de les supprimer cf. 30. Néanmoins, pendant la phase de mise en donnée, il peut-être intéressant de repérer ces noeuds. Une solution est d'utiliser le mot clé : " `creation_ref_noeud_non_references_` " à la place de la suppression. Dans ce cas il y a une création automatique d'une référence nommée " `N_noeud_libre` " que l'on peut récupérer grâce à la sauvegarde du maillage en cours avant tout calcul cf. 21.1.

Ensuite on peut examiner le contenu de la nouvelle référence " `N_noeud_libre` " dans les fichier .her et .lis sauvegardés.

36 Fusion de maillages

De manière similaire avec l'utilitaire de fusion de maillage (21.11), 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 (21.11). 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 (47) 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.

On se reportera à (21.11) pour l'information concernant la création des noms de références résultant de la fusion.

37 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.

TABLE 47 – 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_
```

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

- 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 définit, il y a création d'une référence contenant tous les éléments. Elle a pour nom : " `E_tout` ". Par contre si cette référence est déjà définit, elle n'est pas remplacée.

Certaine fois, pour améliorer la clartée 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.

37.1 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.

38 Eléments disponibles

On dispose d'élément :

— 1D : des biellettes 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(48).

TABLE 48 – liste d'élément finis mécaniques

géométrie	commentaire	interpolation	commentaire	ref
POUT	poutre	BIE1	lineaire 2 noeuds	38.1
POUT	poutre	BIE2	quadratique 3 noeuds	38.2
SEG_AXI	poutre	BIE1	axisymétrique linéaire 2 noeuds	38.3
TRIANGLE	triangle	LINEAIRE	linéaire 3 noeuds	38.5
TRIANGLE	triangle	QUADRACOMPL	quadratique 6 noeuds	38.6
TRIANGLE	triangle	CUBIQUE	quadratique 10 noeuds	38.7
TRIA_AXI	triangle	LINEAIRE	axisymétrique linéaire 3 noeuds	38.8
TRIA_AXI	triangle	QUADRACOMPL	axisymétrique quadratique 6 noeuds	38.9
TRIANGLE	triangle	SFE1	élément coques à 6 noeuds	38.10
TRIANGLE	triangle	SFE2	élément coques à 6 noeuds	38.11
TRIANGLE	triangle	SFE3	élément coques à 6 noeuds	38.12
QUADRANGLE	quadrangle	LINEAIRE	linéaire 4 noeuds	38.14
QUADRANGLE	quadrangle	QUADRATIQUE	quadratique incomplet 8 noeuds	38.15
QUADRANGLE	quadrangle	QUADRACOMPL	quadratique complet 9 noeuds	38.16
QUADRANGLE	quadrangle	CUBIQUE	cubique complet 16 noeuds	38.17
QUAD_AXI	quadrangle	LINEAIRE	axisymétrique linéaire 4 noeuds	38.18
QUAD_AXI	quadrangle	QUADRATIQUE	axisymétrique quadratique incomplet 8 noeuds	38.19
QUAD_AXI	quadrangle	QUADRACOMPL	axisymétrique quadratique complet 9 noeuds	38.20
HEXAEDRE	brique, 6 faces	LINEAIRE	linéaire 8 noeuds	38.22
HEXAEDRE	brique, 6 faces	QUADRATIQUE	quadratique incomplet 20 noeuds	38.23
HEXAEDRE	brique, 6 faces	QUADRACOMPL	quadratique complet 27 noeuds	38.24
PENTAEDRE	1/2 brique 5 faces	LINEAIRE	6 noeuds	38.25
PENTAEDRE	1/2 brique 5 faces	QUADRATIQUE	incomplets 15 noeuds	38.26
PENTAEDRE	1/2 brique 5 faces	QUADRACOMPL	complets 18 noeuds	38.27
TETRAEDRE	tétraèdre à 4 faces	LINEAIRE	4 noeuds	38.28
TETRAEDRE	tétraèdre à 4 faces	QUADRACOMPL	10 noeuds	38.29

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.

38.1 POUT BIE1 : élément barre

L'élément s'appuie sur un élément linéaire 1D de référence (cf. 86).

Le nombre de point d'intégration par défaut est 1 (cf.40.1), et le nombre de noeuds est 2.

L'élément réagit comme une biellette 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. 57).

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. 45).

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 (49) donne un exemple de syntaxe pour définir un élément biellette.

TABLE 49 – Exemple de déclaration d'un élément biellette

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

Il est possible de bloquer la variation de la section (cf. 58). Dans ce cas seule la section initiale est prise en compte.

38.2 POUT BIE2 : élément barre quadratique

L'élément s'appuie sur un élément quadratique 1D de référence (cf. 86).

Le nombre de point d'intégration par défaut est 2 (cf.40.1), et le nombre de noeuds est 3.

L'élément réagit comme une biellette 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. 57).

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. 45).

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 (50) donne un exemple de syntaxe pour définir un élément biellette quadratique.

TABLE 50 – Exemple de déclaration d'un élément biellette

```
elements ----------
   1  ELEMENTS
#-------------------------------------------------------------------
#| NO  |                          |                                 |
#|ELTS |     type element         |       Noeuds                    |
#-------------------------------------------------------------------
     1       POUT  BIE2                1      2     3
```

Il est possible de bloquer la variation de la section (cf. 58). Dans ce cas seule la section initiale est prise en compte.

En résumé pour les différents nombres on a :

```
nbne = 3; // le nombre de noeud de l'élément
nbneA = 3;// le nombre de noeud des aretes
nbi = 2;  // le nombre de point d'intégration pour le calcul mécanique
nbiEr = 3;// le nombre de point d'intégration pour le calcul d'erreur
nbiA = 2; // le nombre de point d'intégration pour le calcul de second membre linéique
nbiMas = 3; // le nombre de points d'intégration pour le calcul de la matrice masse
nbiHour = 0; // le nombre de point d'intégration un blocage d'hourglass
```

38.3 SEG_AXI BIE1 : élément barre linéaire axisymétrique

L'élément s'appuie sur un élément linéaire 1D de référence (cf. 86).

Le nombre de point d'intégration par défaut est 1 (cf.40.1), et le nombre de noeuds est 2.

L'élément réagit aux sollicitations axisymétriques de traction, compression. La discrétisation élément finis est de dimension 1, mais l'élément est relatif à une surface engendrée 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 2. Il est donc nécessaire d'utiliser une loi de comportement associé 2D.

L'espace de travail est de dimension 3.

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. 45).

d'élément.

La table (51) donne un exemple de syntaxe pour définir un élément biellette linéaire axisymétrique.

Il est possible de bloquer la variation de la section (cf. 58). Dans ce cas seule la section initiale est prise en compte.

TABLE 51 – Exemple de déclaration d'un élément linéaire biellette axisymétrique

```
elements ----------
   1  ELEMENTS
#-------------------------------------------------------------------
#| NO  |                        |                                  |
#|ELTS |     type element       |       Noeuds                     |
#-------------------------------------------------------------------
     1       SEG_AXI  BIE1               1      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 §(66.2.11) et §(67.8) pour la description des particularités du chargement.

En résumé pour les différents nombres on a :

```
nbne = 2; // le nombre de noeud de l'élément
nbneA = 2;// le nombre de noeud des aretes
nbi = 2;  // le nombre de point d'intégration pour le calcul mécanique
nbiEr = 2;// le nombre de point d'intégration pour le calcul d'erreur
nbiA = 2; // le nombre de point d'intégration pour le calcul de second membre linéique
nbiMas = 2; // le nombre de point d'intégration pour le calcul de la matrice masse
nbiHour = 0; // le nombre de point d'intégration un blocage d'hourglass
```

38.4 SEG_AXI BIE2 : élément barre quadratique axisymétrique

L'élément s'appuie sur un élément quadratique 1D de référence (cf. 86).

Le nombre de point d'intégration par défaut est 2 (cf.40.1), et le nombre de noeuds est 3.

L'élément réagit aux sollicitations axisymétriques de traction, compression. La discrétisation élément finis est de dimension 1, mais l'élément est relatif à une surface engendrée 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 2. Il est donc nécessaire d'utiliser une loi de comportement associé 2D.

L'espace de travail est de dimension 3.

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. 45).

La table (52) donne un exemple de syntaxe pour définir un élément biellette quadratique axisymétrique.

Il est possible de bloquer la variation de la section (cf. 58). Dans ce cas seule la section initiale est prise en compte.

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 §(66.2.11) et §(67.8) pour la description des particularités du chargement.

En résumé pour les différents nombres on a :

TABLE 52 – Exemple de déclaration d'un élément quadratique biellette axisymétrique

```
elements ----------
   1  ELEMENTS
#-------------------------------------------------------------------
#| NO  |                        |                                  |
#|ELTS |     type element       |       Noeuds                     |
#-------------------------------------------------------------------
     1       SEG_AXI  BIE2               1      2     3
```

```
  nbne = 3; // le nombre de noeud de l'élément
  nbneA = 3;// le nombre de noeud des aretes
  nbi = 4;//  // le nombre de point d'intégration pour le calcul mécanique
  nbiEr = 4;// le nombre de point d'intégration pour le calcul d'erreur
  nbiA = 4;// // 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
  nbiHour = 0; // le nombre de point d'intégration un blocage d'hourglass
```

38.5 TRIANGLE LINEAIRE

L'élément s'appuie sur un élément de référence 2D linéaire (cf. 87).

Le nombre de point d'intégration par défaut est 1 (cf.40.2), 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. 56). L'épaisseur peut varier pendant le calcul (voir 44 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 (53) donne un exemple de syntaxe pour définir un élément triangle.

TABLE 53 – Exemple de déclaration d'un élément triangle avec interpolation linéaire

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

38.6 TRIANGLE quadratique (QUADRACOMPL)

L'élément s'appuie sur un élément de référence 2D quadratique (6 noeuds) (cf. 87).

Le nombre de point d'intégration par défaut est 3 (cf.40.2), sur les arrêtes de 2 (cf.40.1), et le nombre de noeuds est 6.

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 44 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. 56).

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 (54) donne un exemple de syntaxe pour définir un élément triangle quadratique.

TABLE 54 – Exemple de déclaration d'un élément triangle avec interpolation quadratique

```
elements ----------
   1  ELEMENTS
#-------------------------------------------------------------------
#| NO  |                          |                                  |
#|ELTS |     type element         |       Noeuds                     |
#-------------------------------------------------------------------
     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.40.2). La table (55) donne un exemple de syntaxe pour définir ce type d'élément.

En résumé pour les différents nombres on a :

```
nbne = 6; // le nombre de noeud de l'élément
nbneA = 3;// le nombre de noeud des aretes
nbi = 3;  // le nombre de point d'intégration pour le calcul mécanique
nbiEr = 6;// le nombre de point d'intégration pour le calcul d'erreur
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 = 6; // le nombre de point d'intégration pour le calcul de la matrice masse
nbiHour = 0; // le nombre de point d'intégration un blocage d'hourglass
```

TABLE 55 – 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
```

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 42 pour plus d'information.

38.7 TRIANGLE CUBIQUE

L'élément s'appuie sur un élément de référence 2D cubique (10 noeuds) (cf. 87).

Le nombre de point d'intégration par défaut est 6 (cf.40.2), sur les arrêtes de 3 (cf.40.1), 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 44 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. 56).

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 (56) donne un exemple de syntaxe pour définir un élément triangle quadratique.

TABLE 56 – 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 56). 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 42 pour plus d'information.

38.8 Triangle axisymétrique linéaire : TRIA_AXI LINEAIRE

L'élément s'appuie sur un élément de référence 2D linéaire (cf. 87).

Le nombre de point d'intégration par défaut est 1 (cf.40.2), 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 (57) donne un exemple de syntaxe pour définir un élément triangle axisymétrique linéaire.

TABLE 57 – Exemple de déclaration de deux éléments triangles axisymétriques avec interpolation linéaire

```
elements ----------
   2  ELEMENTS
#-------------------------------------------------------------------------
#| NO  |                            |                                    |
#|ELTS |     type element           |       Noeuds                       |
#-------------------------------------------------------------------------
     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 §(66.2.11) et §(67.8) 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
```

38.9 Triangle axisymétrique quadratique : TRIA_AXI QUADRACOMPL

L'élément s'appuie sur un élément de référence 2D quadratique (6 noeuds) (cf. 87).

Le nombre de point d'intégration par défaut est 3 (cf.40.2), 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 (58) donne un exemple de syntaxe pour définir un élément triangle axisymétrique linéaire.

TABLE 58 – 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 §(66.2.11) et §(67.8) 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 58). 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 42 pour plus d'information.

38.10 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.

$$\vec{OM} = \vec{OP}(\theta^1, \theta^2) + z.\vec{N}(\theta^1, \theta^2) \tag{35}$$

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 ce cette cinématique on obtient le tenseur métrique sous la forme suivante :

$$\begin{aligned} g_{\alpha\beta} &= a_{\alpha\beta} - 2\ z.b_{\alpha\beta} + z^2\ b_{\alpha\gamma}b^\gamma_\beta \\ g_{\alpha 0} &= 0 \\ g_{33} &= 1 \end{aligned} \tag{36}$$

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 :

$$\varepsilon_{\alpha\beta} = 0.5(g_{\alpha\beta}(t) - g_{\alpha\beta}(0)) \tag{37}$$

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.

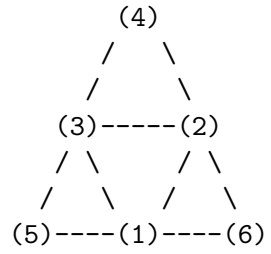
Les conditions limites de type "encastrement" ou "symétrie" sont particulières pour ces éléments. On se reportera à 67.7 pour plus d'informations.

Mode d'emploi de l'élément L'élément s'appuie sur un élément de référence 2D triangulaire linéaire (3 noeuds) (cf. 87).

Le nombre de point d'intégration par défaut est 2, 1 en surface (cf.40.2) et 2 en épaisseur (cf.40.1).

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.59). 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.59) sont les numéros des noeuds, et que le noeud 4 n'existe pas on aurait la connection : 1 2 3 1 5 6

TABLE 59 – numérotation des éléments SFE1




```
 concernant les aretes:   elles sont lineaire
numerotation 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 (60) 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. 56).

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
 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
```

Table 60 – 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
........


  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 ma
  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.21.8) permettant de transformer un maillage triangulaire linéaire en maillage sfe.

38.11 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 arrête "i" est obtenue par une moyenne pondérée :

$$\vec{N}^{(i)} = \frac{(\vec{N}\ H_aA + \vec{N}_e\ DH_d)}{(H_aA + DH_d)} \tag{38}$$

avec $\vec{N}$ la normale de la facette centrale, H_aA la longueur de la hauteur sur la facette

centrale, abaissée sur l'arrête, $\vec{N}_e$ la normale à la facette extérieure, DH_d la longueur de la hauteur sur la facette externe.

La position de la normale sur l'arrête est également optimisée de manière à satisfaire la relation :

$$\vec{GH_i}.\vec{CB} = \vec{H_iG_i}.\vec{CB} \tag{39}$$

où "G" est le centre de gravité du triangle interne, G_i le centre de gravité du triangle externe, $\vec{CB}$ est l'arrête, 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 (61) donne un exemple de syntaxe pour définir un élément sfe2.

Table 61 – 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 (38.10).

38.12 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

$$\theta^3(\theta^1, \theta^2) = a\ \theta^1\ (\theta^1 - 1.) + b\ \theta^2\ (\theta^2 - 1.) + c\ \theta^1\ \theta^2 \tag{40}$$

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 $< \theta 1, \theta^2, \theta^3 >$. On a :

$$b_{11} = 2\ a\ ,\quad b_{22} = 2\ b\ ,\quad b_{12} = c \tag{41}$$

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 (62) donne un exemple de syntaxe pour définir des éléments sfe3 et sfe3C.

TABLE 62 – 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 (38.10).

38.13 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).

38.14 Quadrangle linéaire : QUADRANGLE LINEAIRE

L’élément s’appuie sur un élément de référence 2D linéaire (cf. 88).

Le nombre de point d’intégration par défaut est 4 (cf.40.4), 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. 56).

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 (63) donne un exemple de syntaxe pour définir un élément quadrangle linéaire.

TABLE 63 – 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         |       Noeuds                    |
#-------------------------------------------------------------------
     1     QUADRANGLE  LINEAIRE        1      3      2     6
     2     QUADRANGLE  LINEAIRE  _cm1pti      3      4      2     5
```

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 63). 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 42 pour plus d'information.

38.15 Quadrangle quadratique incomplet : QUADRANGLE QUADRATIQUE

L'élément s'appuie sur un élément de référence 2D quadratique (8 noeuds) (cf. 88).

Le nombre de point d'intégration par défaut est 4 (cf.40.4), 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. 56).

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 (64) donne un exemple de syntaxe pour définir un élément quadrangle quadratique incomplet.

TABLE 64 – Exemple de déclaration d'un élément quadrangle avec interpolation quadratique incomplete

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

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 3 ième 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 42 pour plus d'information.

38.16 Quadrangle quadratique complet : QUADRANGLE QUADRACOMP

L'élément s'appuie sur un élément de référence 2D quadratique (9 noeuds) (cf. 88).

Le nombre de point d'intégration par défaut est 9 (cf.40.4), 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. 56).

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 (65) donne un exemple de syntaxe pour définir un élément quadrangle quadratique complet.

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
```

TABLE 65 – 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
```

```
                    //  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 65). 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 42 pour plus d'information.

38.17 Quadrangle cubique complet : QUADRANGLE CUBIQUE

L'élément s'appuie sur un élément de référence 2D quadratique (16 noeuds) (cf. 88).

Le nombre de point d'intégration par défaut est 16 (cf.40.4), 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. 56).

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 (66) donne un exemple de syntaxe pour définir un élément quadrangle cubique.

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
```

TABLE 66 – 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
```

```
  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 66). 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 42 pour plus d'information.

38.18 Quadrangle axisymétrique linéaire : QUAD_AXI LINEAIRE

L'élément s'appuie sur un élément de référence 2D linéaire (cf. 88).

Le nombre de point d'intégration par défaut est 4 (cf.40.4), 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 (67) donne un exemple de syntaxe pour définir un élément quadrangle axisymétrique linéaire.

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 §(66.2.11) et §(67.8) pour la description des particularités du chargement.

TABLE 67 – 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           |       Noeuds                        |
#-------------------------------------------------------------------------
     1     QUAD_AXI  LINEAIRE         1      3      2     6
     2     QUAD_AXI  LINEAIRE  _cm1pti      3      4      2     5
```

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 67). 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 42 pour plus d'information.

38.19 Quadrangle axisymétrique quadratique incomplet : QUAD_AXI QUADRATIQUE

L'élément s'appuie sur un élément de référence 2D quadratique (8 noeuds) (cf. 88).

Le nombre de point d'intégration par défaut est 4 (cf.40.4), 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 (68) donne un exemple de syntaxe pour définir un élément quadrangle axisymétrique quadratique incomplet.

TABLE 68 – Exemple de déclaration d'un élément quadrangle axisymétriques avec interpolation quadratique incomplete

```
elements ----------
   1  ELEMENTS
#-------------------------------------------------------------------------
#| NO  |                            |                                     |
#|ELTS |     type element           |       Noeuds                        |
#-------------------------------------------------------------------------
     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 §(66.2.11) 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 42 pour plus d'information.

38.20 Quadrangle axisymétrique quadratique complet : QUAD_AXI QUADRACOMP

L'élément s'appuie sur un élément de référence 2D quadratique (9 noeuds) (cf. 88).

Le nombre de point d'intégration par défaut est 9 (cf.40.4), 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 (69) donne un exemple de syntaxe pour définir un élément quadrangle axisymétrique quadratique complet.

TABLE 69 – 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           |       Noeuds                  |
#-------------------------------------------------------------------
     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 §(66.2.11) et §(67.8) 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 69). 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 42 pour plus d'information.

38.21 Quadrangle axisymétrique cubique complet : QUAD_AXI CUBIQUE

L'élément s'appuie sur un élément de référence 2D cubique (16 noeuds) (cf. 88).

Le nombre de point d'intégration par défaut est 16 (cf.40.4), 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 (70) donne un exemple de syntaxe pour définir un élément quadrangle axisymétrique cubique.

Table 70 – 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           |       Noeuds                     |
#-----------------------------------------------------------------------
     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 §(66.2.11) et §(67.8) 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 70). 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 42 pour plus d'information.

38.22 Hexaédres linéaires : HEXAEDRE LINEAIRE

L'élément s'appuie sur un élément de référence 3D linéaire (89).

Le nombre de point d'intégration par défaut est 8 (cf.40.4), 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 (71) donne un exemple de syntaxe pour définir des éléments hexaédriques linéaires.

TABLE 71 – Exemple de déclaration d'éléments hexaédrique avec interpolation linéaire

```
elements ----------
   50  ELEMENTS
#-------------------------------------------------------------------------
#| NO  |                            |                                    |
#|ELTS |     type element           |       Noeuds                       |
#-------------------------------------------------------------------------
     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.40.4) supérieur 27 ou 64. La table (72) donne un exemple de déclaration.

38.23 Hexaédres quadratique incomplet : HEXAEDRE QUADRATIQUE

L'élément s'appuie sur un élément de référence 3D quadratique incomplet (90).

TABLE 72 – 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
```

Le nombre de point d'intégration par défaut est 8 (cf.40.4), 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 (73) donne un exemple de syntaxe pour définir des éléments hexaédriques quadratiques.

TABLE 73 – 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 (74) 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 74 – 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
```

38.24 Hexaédres quadratique complet : HEXAEDRE QUADRA-COMPL

L'élément s'appuie sur un élément de référence 3D quadratique complet (91).

Le nombre de point d'intégration par défaut est 8 (cf.40.4), 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 (75) donne un exemple de syntaxe pour définir des éléments hexaédriques quadratiques complet.

TABLE 75 – 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 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
...
```

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 (76) 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 76 – 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
```

38.25 Pentaèdre linéaires : PENTAEDRE LINEAIRE

L'élément s'appuie sur un élément de référence 3D tri-linéaire (92).

Le nombre de point d'intégration par défaut est 2 (cf.40.5), 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 (77) donne un exemple de syntaxe pour définir des éléments pentaédrique linéaires.

En résumé pour les différents nombres on a :

```
   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
```

Table 77 – Exemple de déclaration d'éléments pentaédriques avec interpolation linéaire

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


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.40.5) égale à 1 ou 6. La table (78) donne un exemple de déclaration pour 6 points d'intégration.

Table 78 – Exemple de déclaration d'éléments pentaédriques linéaire avec 6 points d'intégration

```
   #-------------------------------------------------------------------------
   # NO    Type                    Noeuds                                  |
   #-------------------------------------------------------------------------
     # pentaedre a interpolation lineaire et a 6 points d'intégration
     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
```

38.26 Pentaèdre quadratique incomplet : PENTAEDRE QUADRATIQUE

L'élément s'appuie sur un élément de référence 3D tri-quadratique incomplet (93).

Le nombre de point d'intégration par défaut est 6 (cf.40.5), 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 (79) donne un exemple de syntaxe pour définir des éléments pentaédrique quadratique incomplet.

TABLE 79 – Exemple de déclaration d'éléments pentaédriques avec interpolation quadratique incomplet

```
 #-------------------------------------------------------------------
 #| NO  |                            |                               |
 #|ELTS |     type element           |       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
...
```

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.40.5) égale à 3, 9, 12 ou 18. La table (80) donne un exemple de déclaration.

TABLE 80 – 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 _cm3pti 1  2  3 4 5 6 7 8 9 10 11 12 13
  # 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 3 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     = 3; // 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
nbiHour = 9; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
```

Cette élément présente des modes d'hourglass, il est donc nécessaire de la stabiliser vis-à-vis de ces modes à énergies nulles. Les 3 points sont dans le plan, l'élément ne permet donc pas de représenter un comportement de flexion, par contre il est particulièrement intéressant pour représenter un comportement dans le plan d'une plaque, sans recourir à une loi en def ou contraintes planes, mais en gardant une loi 3D.

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
```

38.27 Pentaèdre quadratique complet : PENTAEDRE QUADRACOMPL

L'élément s'appuie sur un élément de référence 3D tri-quadratique complet (91).

Le nombre de point d'intégration par défaut est 6 (cf.40.5), 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 (81) donne un exemple de syntaxe pour définir des éléments pentaédrique quadratique complet.

TABLE 81 – Exemple de déclaration d'éléments pentaédriques avec interpolation quadratique complet

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

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
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.40.5) égale à 9, 12 ou 18. La table (82) donne un exemple de déclaration.

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
```

TABLE 82 – 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'intégration
  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'intégration
  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'intégration
  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'intégration
  4 PENTAEDRE QUADRACOMPL _cm18pti 1  2  3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
```

```
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
```

38.28 Tétraèdre linéaire : TETRAEDRE LINEAIRE

L'élément s'appuie sur un élément de référence 3D tri-linéaire (95).

Le nombre de point d'intégration par défaut est 1 (cf.98), 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 (83) donne un exemple de syntaxe pour définir des éléments tétraédrique linéaire.

Table 83 – 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
```

38.29 Tétraèdre quadratique : TETRAEDRE QUADRATIQUE

L'élément s'appuie sur un élément de référence 3D tri-quadratique (96).

Le nombre de point d'intégration par défaut est 1 (cf.98), 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 (84) donne un exemple de syntaxe pour définir des éléments tetraédriques linéaires.

Table 84 – Exemple de déclaration d'éléments tétraèdriques 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
...
```

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
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
```

Il est également possible d'utiliser des éléments tétraédriques quadratiques sous intégrés. Dans ce cas le nombre de points d'intégration est 1. La table (85) donne un exemple de déclaration.

TABLE 85 – Exemple de déclaration d'éléments tétraèdriques quadratique sous-intégré (un point d'intégration)

```
 #-------------------------------------------------------------------------
 #| NO  |                            |                                    |
 #|ELTS |     type element           |       Noeuds                       |
 #-------------------------------------------------------------------------
     # tétraèdre a interpolation quadratique et a 4 points d'intégration
     1 TETRAEDRE QUADRACOMPL  _cm1pti   1  2  3   4   5  6  7  8  9  10
...
```

39 Numérotations locales des noeuds des éléments de référence

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

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

La table (86) montre la numérotation des différents éléments 1D de référence utilisés.

TABLE 86 – numérotation des éléments 1D de référence

```
linéaire -> 2 noeuds

        -1            0           1            KSI
      ---*------------|-----------*--------->
        (1)                      (2)

quadratique -> 3 noeuds

        -1            0           1            KSI
      ---*------------|-----------*--------->
        (1)          (2)         (3)

cubique -> 4 noeuds

        -1              0             1            KSI
      ---*--------*----------*---------*--------->
        (1)      (2)        (3)       (4)
```

On remarque que les noeuds sont numéroté dans l'ordre de la coordonnée croissante.

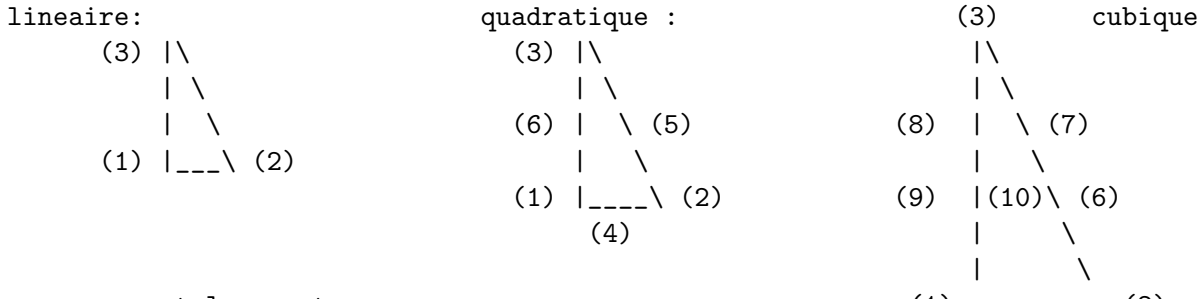
39.2 Eléments de référence à géométrie surfacique 2D triangulaire

La numérotation des éléments de référence 2D triangulaire est indiquée à la table (87).

39.3 Eléments de référence à géométrie surfacique 2D quadrangulaire

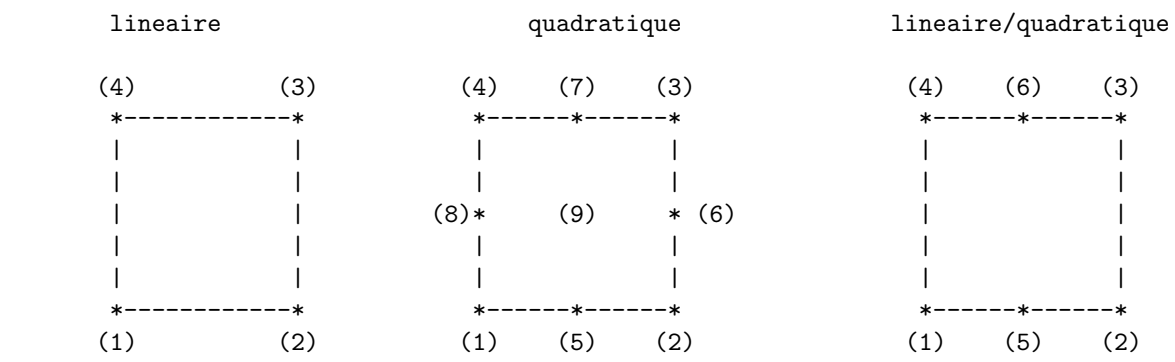
La table (88) est relatives aux quadrangles.

TABLE 87 – numérotation des éléments triangulaires de référence

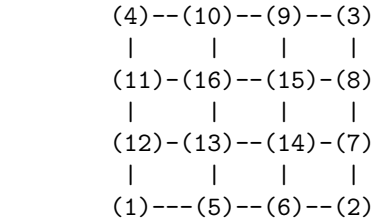


concernant les aretes
elles sont lineaire quadratique ou cubique comme l'element (4) (5)
numerotation des arrêtes :
en linéaire : 1 : 1 2, 2 : 2 3, 3 : 3 1
en quadratique : 1 : 1 4 2, 2 : 2 5 3, 3 : 3 6 1
en cubique: 1 : 1 4 5 2 2: 2 6 7 3 3: 3 8 9 1
leur nombre de point d'integration nbil depend du 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

TABLE 88 – 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.

39.4 Eléments de référence à géométrie 3D hexaédrique

La numérotation des éléments 3D est indiquée par les tables suivantes :

— la table (89) concerne les hexaèdres linéaires,

— la table (90) concerne les hexaèdres quadratiques,

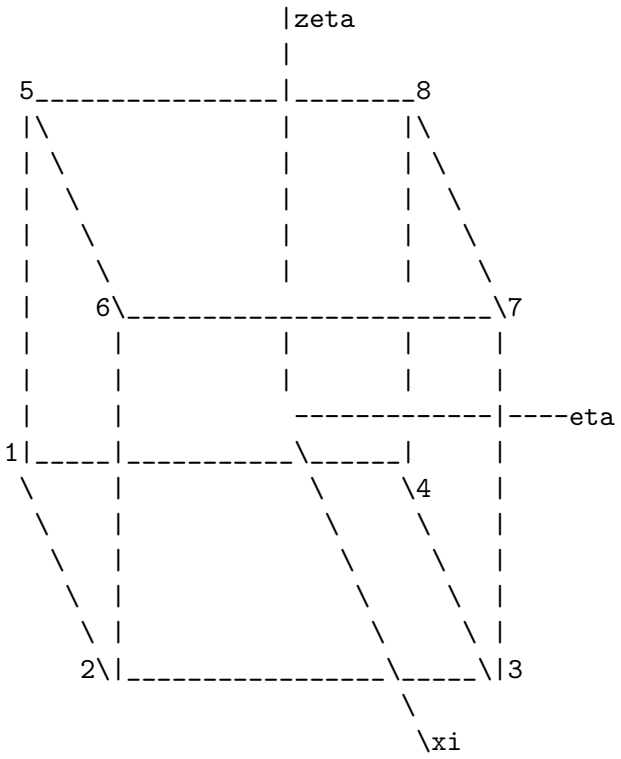
— la table (92) concerne les pentaèdres linéaires,

— la table (93) concerne les pentaèdres quadratiques,

— la table (95) concerne les tetraèdres linéaires,

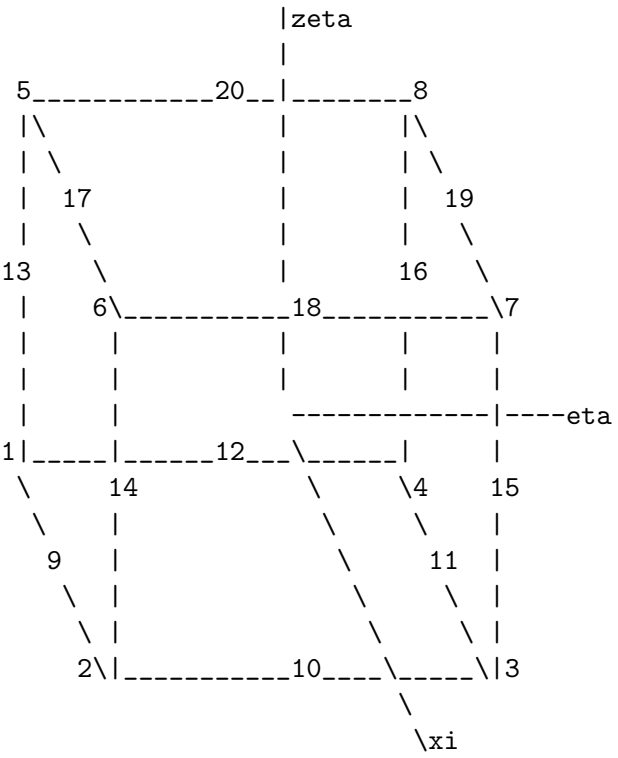
— la table (96) concerne les tetraèdres quadratiques,

TABLE 89 – numérotation de l'élément de référence hexaédrique linéaire



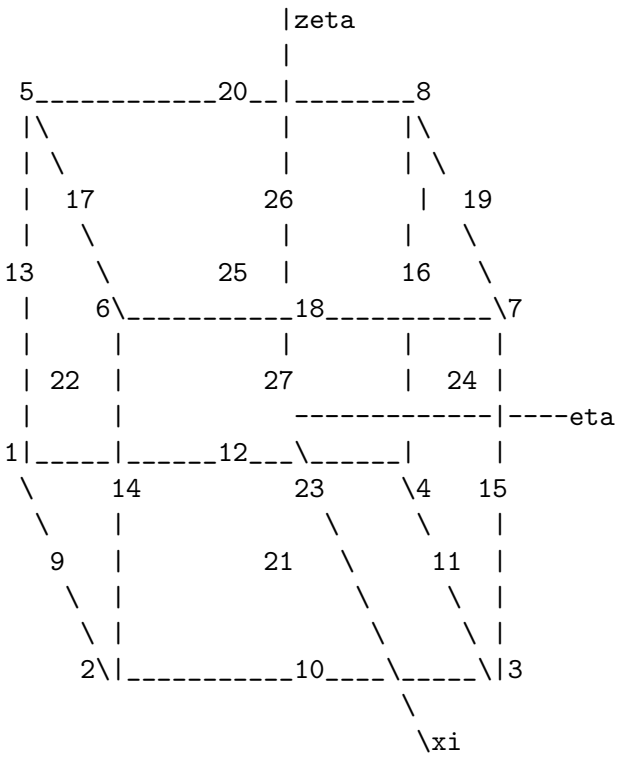

```
face 1 : noeud 1 4 3 2, face 2 : noeud 1 5 8 4,
face 3 : noeud 1 2 6 5, face 4 : noeud 5 6 7 8,
face 5 : noeud 2 3 7 6, face 6 : noeud 3 4 8 7,
 les normales sortent des faces des elements

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 90 – 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 91 – 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,

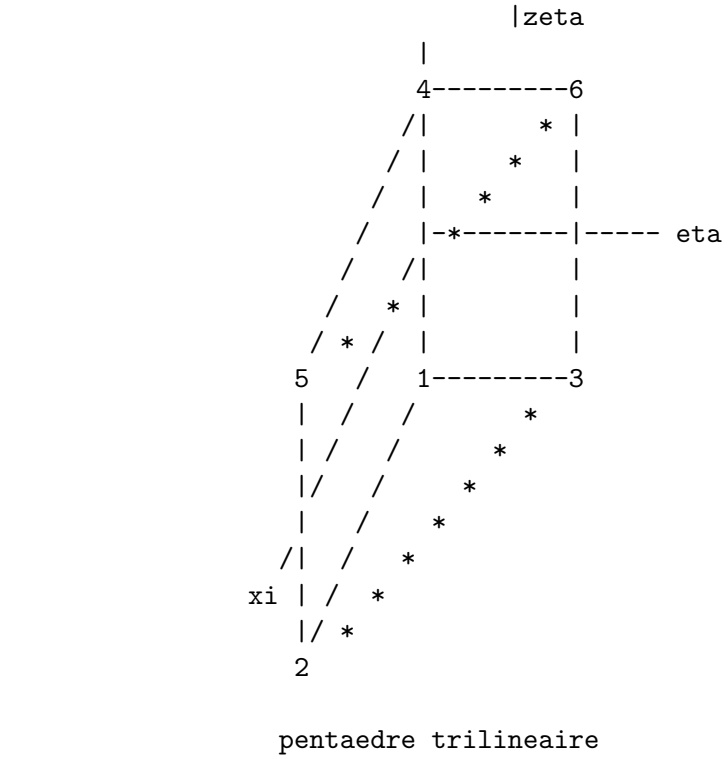
les normales sortent des faces des elements

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

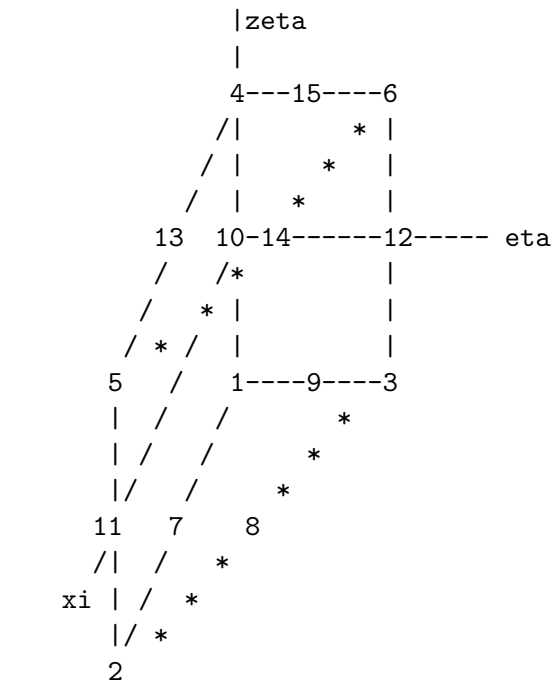
39.5 Eléments de référence à géométrie 3D pentaédrique

TABLE 92 – numérotation de l'élément pentaédrique linéaire de référence.



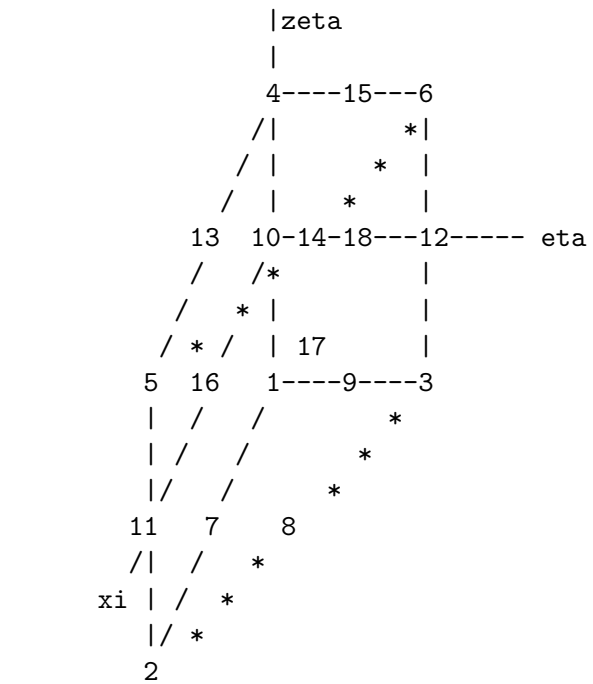

```
Description des faces , puis des arêtes
face 1 : noeud 1 3 2, face 2 : noeud 1 4 6 3,
face 3 : noeud 1 2 5 4, face 4 : noeud 4 5 6,
face 5 : noeud 2 3 6 5
 les normales sortent des faces des elements

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 93 – 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 94 – Numérotation de l'élément pentaédrique quadratique complet de référence.

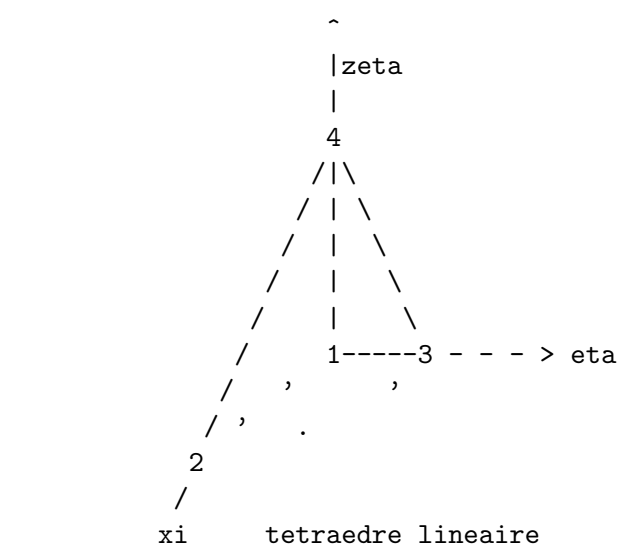


pentaèdre triquadratique complet

```
 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
```

39.6 Eléments de référence à géométrie 3D tétraédrique

TABLE 95 – Numérotation de l'élément tétraédrique linéaire de référence.

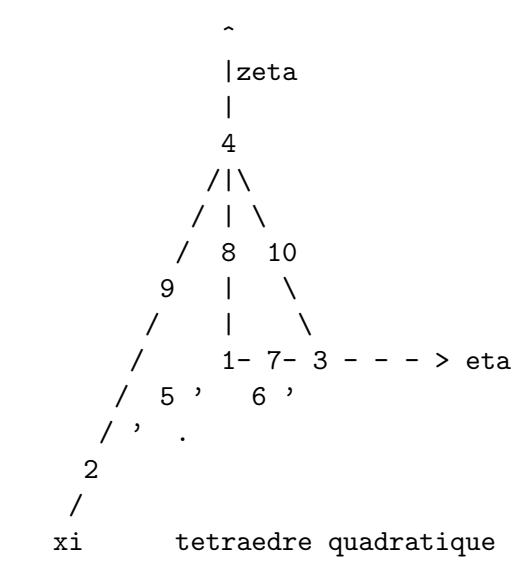


tetraedre lineaire

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 96 – 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

40 Positions des points d'intégrations

40.1 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 $S = 1./(6.\sqrt{(6./5.)})$;

40.2 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 $\eta(7) = 1 - 2b$, $w(6) = 31/240 - A$;

40.3 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 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 $\eta(4) = 1/\sqrt{3}$, $w(3) = 1$;

40.4 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
$\xi(1) = a$, $\eta(1) = a$, $\zeta(1) = a$, $w(1) = 1$;
$\xi(2) = a$, $\eta(1) = a$, $\zeta(1) = -a$, $w(1) = 1$;
$\xi(3) = a$, $\eta(1) = -a$, $\zeta(1) = a$, $w(1) = 1$;
$\xi(4) = a$, $\eta(1) = -a$, $\zeta(1) = -a$, $w(1) = 1$;
$\xi(5) = -a$, $\eta(1) = a$, $\zeta(1) = a$, $w(1) = 1$;
$\xi(6) = -a$, $\eta(1) = a$, $\zeta(1) = -a$, $w(1) = 1$;
$\xi(7) = -a$, $\eta(1) = -a$, $\zeta(1) = a$, $w(1) = 1$;
$\xi(8) = -a$, $\eta(1) = -a$, $\zeta(1) = -a$, $w(1) = 1$;
avec $a = 1./\sqrt{3.}$;

Pour les autres nombres de points d'intégration : 1, 27 = 3 x 3 x 3, 64 = 4 x 4 x 4, on suit l'ordre des points d'intégration donnée en 1D (40.1). Le poids d'intégration est le produit des poids correspondants 1D selon les directions ξ , η et ζ

40.5 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 (40.2) puis globalement suivant le segment selon (40.1). 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 (97).

TABLE 97 – 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 (40.2). La position selon ζ est donnée par la position dans le segment (40.1).

Le poids d'intégration est le produit des poids correspondants 2D et 1D.

40.6 Position des points d'intégrations pour les éléments tétraédriques

TABLE 98 – 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)  ;
```

41 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 $nbddl_{libre} = ddl_{total} - 6$ (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.

42 Gestion des modes d'hourglass

Les éléments sous-intégrés (cf. 41), 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 couteuse en temps de calcul) avec une intégration complète.

La table (99) 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=0.01, cela signifie que seulement 1/100 de la raideur et du second membre de stabilisation sont appliqués.

En fait actuellement, deux cas de stabilisation sont disponibles, différenciés par le mot clé indiquant le type de stabilisation :

TABLE 99 – 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  0.01
```

1. STABHOURGLASS_PAR_COMPORTEMENT : à chaque calcul du résidu en explicite, ou du résidu et de la raideur en implicite, la sous intégration (n points) est utilisée pour la loi principale et l'intégration complète (m points) pour la loi secondaire de stabilisation. Donc au final, si la position des points d'intégration pour les deux méthodes, est différente ce qui est le cas courant, m+n points sont utilisés. Cela signifie que tous les calculs relatifs à la cinématique : métrique, sensibilité de la métrique aux variations des positions etc. ; sont évalués m+n fois. Cette technique est coûteuse, néanmoins si l'évaluation de la loi de comportement principal (par exemple une loi élasto-plastique complexe) demande des temps de calcul beaucoup plus important que celle de la loi de stabilisation (par exemple la partie élastique de la loi élasto-plastique), le gain global est appréciable. À noter que si l'élément est très distordu, les points d'intégration complète peuvent conduire rapidement à des jacobiens négatifs, ce qui est un peu moins vrai pour les points de sous-intégration. Cela provient du fait qu'en général, au centre de l'élément le jacobien demeure plus longtemps positif, à mesure que la qualité de forme de l'élément se dégrade, comparativement aux points excentrés.
2. STABHOURGLASS_PAR_COMPORTEMENT_REDUIT : la technique utilisée ici est une variation de la technique précédente, qui dans la pratique se révèle très efficace. L'idée est de calculer la stabilisation une seule fois, sur le maillage non déformé. Ensuite la même stabilisation est appliquée tout au long du calcul quelque soit la déformation, que ce soit en explicite ou en implicite. Par rapport à la première technique :
   (a) la méthode est beaucoup plus rapide, l'évaluation des grandeurs cinématiques et loi de comportement n'est effectué qu'au niveau des points d'intégration réduite
   (b) comme la stabilisation est effectuée sur la forme initiale non déformée, elle n'entraîne pas, a priori, de jacobien négatif.

   Pour ces raisons, il est recommandé d'utiliser préférentiellement cette seconde techniques.

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 élément. Le contrôle de ces valeurs permet de limiter l'impact de la stabilisation d'hourglass.

À chaque utilisation de la stabilisation des modes d'hourglass **il est recommandé de contrôler le niveau de l'énergie d'hourglass par rapport à celui des énergies physiques !**.

Remarque : Pour l'instant les éléments biellettes, point et les éléments SFE n'intègrent pas de blocage d'hourglass.

43 Stabilisation membrane et biel

les éléments membranes et bielles ne possèdent pas de raideur transversales. Dans certains calculs, notamment lors d'un calcul quasi-statique utilisant une méthode de Newton-Raphson (mais pas seulement), il peut-être intéressant de chercher à stabiliser le comportement transversals des membranes et ou des bielles. Dans le cas contraire, on obtient naturellement une matrice de raideur locale singulière, qui globalement peut conduire à une matrice globale de raideur singulière.

Pour l'instant seule la stabilisation pour les membranes est opérationnelle (V 6.938).

La méthodologie imaginée dans Herezh++ est d'introduire une raideur suivant la direction normale au plan de la membrane. L'intensité de la raideur est proportionnelle au maximum :

- soit de la valeur absolue des raideurs dans le plan
  (mot clé : " STABMEMBRANE_BIEL_PREMIER_ITER "
  ou " STABMEMBRANE_BIEL_PREMIER_ITER_INCR ")
- soit de l'approximation des valeurs propres via le théorème de Gerschgorin
  (mot clé : STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER
  ou STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER_INCR)

suivant un paramètre qui peut être :

- soit fixe (mot clé : " une_valeur_numerique_ " : voir l'exemple 100),
- soit piloté par une fonction nD (mot clé : " une_fonction_nD_ " : voir l'exemple 101) dépendant uniquement de grandeurs globales. Ainsi par exemple il est possible de faire varier cette intensité de la raideur, au cours des itérations de Newton.

La force des raideurs est alors égale à :

$$\vec{F} = -\alpha \times [(\vec{X}(t+\Delta t) - \vec{X}(t)).\vec{N}] \times \vec{N} \tag{42}$$

où α est l'intensité de la raideur de stabilisation, $(\vec{X}(t+\Delta t)-\vec{X}(t))$ représente la variation de position du point (donc de chaque noeud) entre le début et la fin de l'incrément, $\vec{N}$ est la normale à la membrane : pendant le calcul il s'agit en faite de la normale calculée à chaque point d'intégration.

L'intensité de la raideur de stabilisation peut-être calculée à différent moment du calcul. Les cas possibles sont repérés par un mot clé :

— " STABMEMBRANE_BIEL_PREMIER_ITER " signifie que l'intensité est calculée à chaque première itération à partir du maximum des valeurs absolues de la matrice de raideur,

— " STABMEMBRANE_BIEL_PREMIER_ITER_INCR " signifie que l'intensité est calculée à la première itération du premier incrément, à partir du maximum des valeurs absolues de la matrice de raideur,

— " STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER " signifie que l'intensité est calculée à chaque première itération à partir du maximum des valeurs absolues des valeurs propres de la raideur,

— " STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER_INCR " signifie que l'intensité est calculée à la première itération du premier incrément, à partir du maximum des valeurs propres de la raideur.

Remarque L'idée dans ces différentes méthodes de calcul de l'intensité est d'éviter que l'intensité dépende à chaque itération de la raideur de membrane dans le plan, car dans ce cas on peut obtenir un calcul instable dans le cas d'une loi non linéaire par exemple.

TABLE 100 – Exemple de déclaration d'une stabilisation transversale d'éléments de membrane à l'aide d'un ratio fixé sur la raideur dans le plan médian de l'élément

```
stabilisation_transvers_membrane_biel_
#-----------------------------------------------------
#  ref  | type et parametre                          |
#-----------------------------------------------------
# la référence des elements concernes, le type de stabilisation
# STABMEMBRANE_BIEL_PREMIER_ITER
#   le calcul de la raideur de stabilisation  est effectue à l'iteration 1
# le coefficient modérateur: 0.01 ->  la raideur de stabilisation = 0.01
#   la raideur dans le plan de l'élément
# une_valeur_numerique_ -> mot clef
E_to     STABMEMBRANE_BIEL_PREMIER_ITER   une_valeur_numerique_  0.01
```

44 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 :

$$\frac{V - V_t}{V} = \frac{S.h - S_t.h_t}{S.h} = \frac{trace(\boldsymbol{\sigma})}{3\ K_t} \tag{43}$$

TABLE 101 – Exemple de déclaration d'une stabilisation transversale d'éléments de membrane à l'aide d'un ratio piloté par une fonction nD ici : la fonction pilotageStab

```
stabilisation_transvers_membrane_biel_
#-----------------------------------------------------
#  ref  | type et parametre                          |
#-----------------------------------------------------
# la référence des elements concernes, le type de stabilisation
# STABMEMBRANE_BIEL_PREMIER_ITER
# -> le calcul de la raideur de stabilisation est effectue à l'iteration 1
# la force de stabilisation = la valeur de la fonction  pilotageStab
# multiplié par  la raideur dans le plan de l'élément
# une_fonction_nD_ -> mot clef
E_to     STABMEMBRANE_BIEL_PREMIER_ITER   une_fonction_nD_  pilotageStab
```

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 :

$$\frac{\dot{\boldsymbol{I}}_\sigma}{3} = -\dot{P} = K_t \, \frac{\dot{V}}{V} = K_s \times \text{taux de variation relative de volume} \tag{44}$$

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.

45 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 :

$$\frac{V - V_t}{V} = \frac{S.l - S_t.l_t}{S.h} = \frac{trace(\boldsymbol{\sigma})}{3 \; K_t} log\left(\frac{V}{V_0}\right) = log\left(\frac{S.l}{S_0.l_0}\right) \tag{45}$$

avec S l la section et longueur de la fibre moyenne finales (à $t+\delta t$) , S_0 l_0 la section et la longueur de la fibre moyenne initiale. Dans cette formule, on considère une variation globale logarithmique du volume. Cette variation peut se comprendre comme l'intégrale de la formule incrémentale :

$$\frac{\dot{V}}{V} = \frac{\dot{trace}(\boldsymbol{\sigma})}{3 \; K(t)} \tag{46}$$

Il s'agit ici de scalaire,

ceci de manière à pouvoir considérer Cette formule constitue donc une linéarisation sur un pas de temps de la formule générale :

$$\frac{\dot{\boldsymbol{I}}_\sigma}{3} = -\dot{P} = K_t \, \frac{\dot{V}}{V} = K_s \times \text{taux de variation relative de volume} \tag{47}$$

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

46 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 a activer est : les_courbes_1D. Ensuite les différentes courbes sont décrites successivement, avec avant chaque courbe un nom de baptême (ou identificateur).

La table (102) donne un exemple de courbes. Dans cette exemple le nom de la première courbe est "courbe1", il s'agit d'une courbe poly-linéaires.

On se reportera au chapitre (81.1) pour une description exhaustive des différentes courbes.

TABLE 102 – 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
```

47 Introduction et utilisation de fonction nD

Les fonctions nD correspondent à des fonctions multi-dimensionnelles, utilisables dans certaines parties d'Herezh. Par exemple, la loi critère peut-être pilotée par une fonction multi-dimensionnelle. L'utilisation et la déclaration des fonctions nD suivent une méthodologie très proche de celle des courbes 1D. Actuellement un seul type de fonction nD est disponible, il s'agit des fonctions littérales à n variables, n <= 5.

La définition d'une liste de fonctions nD est facultative. Le mot clé a activer est : les_Fonctions_nD. Ensuite les différentes fonctions sont décrites successivement, avec avant chaque fonction un nom de baptême (ou identificateur).

La table (103) donne un exemple de fonction. Dans cette exemple le nom de la première fonction est "fonction1". Elle comporte 2 arguments : "i" et "j" et fonctionne de la manière suivante (fct = le retour de la fonction) :

si ($i < 2$)	alors	si ($j < 1000$)	alors	fct = -50
			sinon	fct = 0.
	sinon	si ($j < 2000$)	alors	fct = -10
			sinon	fct = 0.

On se reportera au chapitre (81.2) pour une description exhaustive.

TABLE 103 – Exemple de déclaration d'une liste de fonctions nD.

```
les_fonctions_nD   #------------

# exemple de definition d'une fonction Fonction_expression_litterale_nD
#                 f(i,j) = une expression de i et j
 fonction1  FONCTION_EXPRESSION_LITTERALE_nD
   un_argument= i un_argument= j
   fct=  (i < 2) ? ((j < 1000) ? -50. : 0. ): ((j < 2000) ? -10. : 0. )
   fin_parametres_fonction_expression_litterale_
```

Sixième partie

Lois de comportement

48 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 (104) indique la syntaxe dans le cas d'un maillage :

TABLE 104 – Exemple de déclaration d'un nom de loi de comportement associée à une référence, ceci dans le cas d'un maillage.

```
choix_materiaux
#-------------------------------------
# Elements   |      Materiau        |
#-------------------------------------
  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 (105) indique la syntaxe dans le cas de plusieurs maillages.

TABLE 105 – 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   |      Materiau        |
#-----------------------------------------------------
  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 matériau associé aux éléments contenus dans la liste "ELEMENTS1". La ligne suivante définit une seconde région associée à un second matériau.

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 (106) 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 suivante on trouve alors les coefficients de la loi : E et

TABLE 106 – Exemple de déclaration de loi de comportement associée à deux matériaux.

```
materiaux
#-----------------------------------------------------
# Nom Materiau  |  Type loi       |   Potentiel     |
#-----------------------------------------------------
     MATE1        ISOELAS

#--------------------
#     E    |     nu  |
#--------------------
        10000.            0.

#-----------------------------------------------------
# Nom Materiau  |  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 mécanique 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 (108) présente un exemple complet de déclaration permettant le calcul de l'élongation.

49 Type de déformation

Par défaut le type de déformation utilisé dans Herezh++ est en général la déformation d'Almansi.

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\hat{g}^{ij} - g^{ij})\hat{\vec{g}}^{i} \otimes \hat{\vec{g}}^{j}$$

. en 1D, on obtient dans le repère global :

$$\varepsilon_{11} = \frac{1}{2}(1 - \frac{(l_0)^2}{l^2})$$

On observe que cette mesure est simple, mais quel 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 $-\infty$.

Dans certains cas, 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 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_STANDART` " c'est-à-dire la déformation d'Almansi. Ce mot clé doit se trouver sur une ligne à part, il peut se positionner avant ou après la définition (facultative) d'un niveau de commentaires spécifiques (cf. 50).

La table (107) donne un exemple d'utilisation de déformation logarithmique.

TABLE 107 – 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 defaut: DEFORMATION_STANDART) --
           type_de_deformation    DEFORMATION_LOGARITHMIQUE
```

Il faut cependant noter que le choix de la mesure de déformation n'est pas toujours possible. Par exemple, pour une loi de type Mooney-Rivlin, de base ou polynomiale, le choix de la mesure de déformation est dicté par le modèle théorique qui s'appuie sur le tenseur de Cauchy-Green, droit ($\boldsymbol{C}$) ou gauche ($\boldsymbol{B}$), suivant que l'on travaille dans le repère initial ou final (cas d'Herezh).

49.1 Exemple de déclaration pour intégrer la dilatation thermique dans un calcul mécanique

La table (108) donne un exemple complet pour intégrer la dilatation thermique dans un calcul mécanique.

TABLE 108 – 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
```

50 Niveau de commentaire

En marge du niveau global de commentaire (cf. 7) il est possible d'introduire un niveau de commentaire spécifiquement à une loi de comportement. Pour ce faire deux méthodes coexistent dans la mise en données.

La première méthode est la méthode générale, valable pour toutes les lois. À la fin de la mise en données d'une loi, sur une ligne à part, on indique le mot clé : `permet_affichage_` suivi d'un entier entre 0 et 10 qui indique le niveau demandé. Ce mot clé peut-être avant ou après la définition du type de déformation (cf. 49).

La seconde méthode est spécifique à certaines lois de comportement, qui pour des raisons historiques, avaient en interne également un paramètre de niveau de commentaire géré par le même mot clé : `permet_affichage_` . Pour ces lois, il est toujours possible d'indiquer une valeur de niveau de commentaire en suivant la mise en données spécifiques de la loi (voir par exemple 51.15.1) souvent au travers de la définition de paramètres de réglage.

Remarque : Il est possible de cumuler les deux méthodes. Dans ce cas, c'est le dernier niveau indiqué qui est retenu.

51 Liste de lois mécaniques et thermo-mécaniques disponibles

TABLE 109 – liste des différentes lois

identificateur	indications	ref
	élastique isotrope	
ISOELAS1D	1D : Hooke	(cf.111)
ISOELAS2D_D	2D : Hooke déformations planes	(cf.111)
ISOELAS2D_C	2D : Hooke contraintes planes	(cf.111)
ISOELAS	3D : Hooke	(cf.111)
	élastique isotrope non-linéaire	
ISO_ELAS_ESPO1D	1D : $\sigma = E(\varepsilon)\varepsilon = f(\lvert\varepsilon\rvert)E\varepsilon$	(cf.116)
ISO_ELAS_ESPO3D	3D : $\sigma = E(\varepsilon)\varepsilon = f(\gamma)E\varepsilon$ avec $\gamma = \sqrt{2./3.\boldsymbol{\epsilon} : \boldsymbol{\epsilon}}$	(cf.116)
ISO_ELAS_SE1D	1D : $\sigma = f(\epsilon)$	(cf.116)
	hyperélastique isotrope	
ISOHYPER3DFAVIER3	3D : potentiel proposé par Denis Favier	(cf.122)
ISOHYPERBULK3	3D : potentiel pour la partie volumique seule potentiel = f(T) × $(log(V))^2$ $K/6$	(cf.51.3.2)
ISOHYPERBULK_GENE	3D : potentiel pour la partie volumique seule potentiel = f(T) × g(V)	(cf.51.3.3)
MOONEY_RIVLIN_1D	1D : loi classique de Mooney Rivlin	(cf.122)
ISOHYPER3DORGEAS1	3D : potentiel proposé par Laurent Orgéas	(cf.122)
ISOHYPER3DORGEAS2	3D : idem ISOHYPER3DORGEAS1 + amélioration des fonctions de dépendance à la phase	(cf.51.3.7)
POLY_HYPER3D	3D : loi polynomiale en invariants de même type que ceux de Mooney Rivlin	(cf.122)
	élastoplastique isotrope	
PRANDTL_REUSS1D	1D : Prandtl Reuss grandes déformations	(cf.156)
PRANDTL_REUSS2D_D	2D : Prandtl Reuss grandes déformations déformations planes	(cf.156)
PRANDTL_REUSS	3D : Prandtl Reuss grandes déformations	(cf.156)
	visco-élastique isotrope	
NEWTON1D	1D : $\boldsymbol{\sigma} = \mu\boldsymbol{D}$ ou $\boldsymbol{\sigma} = \mu(\boldsymbol{D} : \boldsymbol{D})^{n/2}\boldsymbol{D}$	(cf.158)
NEWTON2D_D	2D : déformation plane $\boldsymbol{\sigma} = \mu\boldsymbol{D}$ ou $\boldsymbol{\sigma} = \mu(\boldsymbol{D} : \boldsymbol{D})^{n/2}\boldsymbol{D}$	(cf.158)
NEWTON3D	3D : $\boldsymbol{\sigma} = \mu\boldsymbol{D}$ ou $\boldsymbol{\sigma} = \mu(\boldsymbol{D} : \boldsymbol{D})^{n/2}\boldsymbol{D}$	(cf.158)
MAXWELL1D	1D : un ressort et un amortisseur en série	(cf.158)
MAXWELL3D	3D : un ressort et un amortisseur en série	(cf.158)
	composition de lois élémentaires	
LOI_ADDITIVE_EN_SIGMA	1D, 2D, 3D : $\boldsymbol{\sigma} = \sum \boldsymbol{\sigma}_{(i)}$	(cf.170)
LOI_DES_MELANGES_EN_SIGMA	1D, 2D, 3D : $\boldsymbol{\sigma} = (\alpha)\boldsymbol{\sigma}_{(1)} + (\alpha - 1)\boldsymbol{\sigma}_{(2)}$	(cf.170)

TABLE 110 – suite de la liste des différentes lois

identificateur	indications	ref du commentaire
	lois d'élasto-hystérésis	
HYSTERESIS_1D	1D : loi d'hystérésis classique (modèle de Guélin-Favier-Pégon)	(cf.189)
HYSTERESIS_3D	3D : loi d'hystérésis classique modifiée (modèle de Guélin-Favier-Pégon-Bless-Rio)	(51.12.2)
HYSTERESIS_BULK	3D : loi d'hystérésis appliquée au comportement sphérique (Rio)	(51.12.3)
	passage 3D 2D ou 3D 1D d'une loi quelconque	
LOI_DEFORMATIONS_PLANES	loi 3D transformée en déformations planes	(cf.51.8)
LOI_CONTRAINTES_PLANES	loi 3D transformée en contraintes planes	(cf.51.9)
LOI_CONTRAINTES_PLANES_DOUBLE	loi 3D transformée en contraintes planes doubles	(cf.51.10)
LOI_CRITERE	application d'un critère à une loi 3D	(cf.51.11)
	lois hypo-élastiques	
HYPO_ELAS3D	3D : loi hypo-élastique $\dot{\boldsymbol{S}} = \mu\bar{\boldsymbol{D}}$ et $\dot{I}_\sigma = K_c I_{\boldsymbol{D}}$	(cf.201)
HYPO_ELAS2D_C	2D : loi hypo-élastique contrainte plane $\dot{\boldsymbol{S}} = \mu\bar{\boldsymbol{D}}$ et $\dot{I}_\sigma = K_c I_{\boldsymbol{D}}$	(cf.201)
HYPO_ELAS1D	1D : loi hypo-élastique $\dot{\boldsymbol{\sigma}} = f()\boldsymbol{D}$ et $\dot{I}_\sigma = K_c I_{\boldsymbol{D}}$	(cf.201)
	lois permettant d'annuler la contribution de l'élément	
LOI_RIEN1D	1D : loi qui ne fait rien mécaniquement	(cf.51.14)
LOI_RIEN2D	2D : loi qui ne fait rien mécaniquement	(cf.51.14)
LOI_RIEN3D	3D : loi qui ne fait rien mécaniquement	(cf.51.14)
	Élastique anisotrope	
ORTHOELA3D	3D : orthotropie élastique entraînée	(cf.51.15.1)
	Hypo-élastique anisotrope	
HYPO_ORTHO3D	3D : hypo-élastique orthotrope entraînée	(cf.51.16.1)
	Quelconque anisotrope par projection	
PROJECTION_ANISOTROPE_3D	anisotropie pas projection d'une loi existante 3D	(cf.51.17.1)

51.1 Lois iso-élastiques linéaires

Les lois isotropes élastiques linéaires de type Hooke, disponibles sont données dans la table (cf.111).

TABLE 111 – liste des différentes lois isotropes élastiques disponibles

identificateur	indications	ref du commentaire
ISOELAS1D	élastique isotrope 1D	(51.1.1)
ISOELAS2D_D	élastique isotrope 2D déformations planes	(51.1.2)
ISOELAS2D_C	élastique isotrope 2D contraintes planes	(51.1.3)
ISOELAS	élastique isotrope 3D	(51.1.4)

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 :

$$-P = trace(\sigma) = K\ trace(\varepsilon) \quad et \quad \boldsymbol{S} = 2\ G\ \bar{\boldsymbol{\varepsilon}} \tag{48}$$

où K est le module de compressibilité et G le module de cisaillement. L'expression $\frac{trace(\varepsilon)}{3}$ est censée représenter la variation relative volumique $\frac{\Delta V}{V}$. Ceci est exact 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). Ici par défaut la mesure de déformation est celle d'Almansi, mais il est possible de choisir une mesure logarithmique.

On a :

$$K = \frac{E}{3(1-2\ \nu)}\ et\ G = \frac{E}{2(1+\nu)} \tag{49}$$

avec E et ν le module d'Young et le coefficient de Poisson.

La contrainte peu également est calculée via l'expression équivalente suivante :

$$\boldsymbol{\sigma} = \frac{(E\ \nu)}{((1.-2.\nu)\ (1+\nu))}\boldsymbol{I}_{\varepsilon} + \frac{E}{(1+\nu)}\ \varepsilon \tag{50}$$

51.1.1 ISOELAS1D

ISOELAS1D identificateur d'une loi 1D isotrope de type Hooke (cf.51.1) . 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 (112) donne un exemple de déclaration de loi thermo-dépendante avec une courbe E=f(T) explicitement définie.

TABLE 112 – 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 DdlP 0. 100000. 100. 50000.0  FdlP  0.3
#-------- fin cas d'une loi isoelas acier --------
```

À 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 (113) donne un exemple de déclaration de loi thermo-dépendante avec une courbe E= f(T) repérée par un nom de référence.

TABLE 113 – 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 --------
```

51.1.2 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 (50) vraie, quel que 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 pour `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.

51.1.3 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 (50) 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 :

$$\varepsilon_3^3 = \frac{-\nu}{1-\nu}\,(\varepsilon_1^1 + \varepsilon_2^2) \tag{51}$$

La trace de $\boldsymbol{\varepsilon}$ s'en déduit. Les termes $\sigma_\beta^\alpha, \quad (\alpha,\ \beta\ = 1,2)$ s'obtiennent en utilisant de nouveau la relation (50).

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 tient compte de la variation de l'épaisseur.

51.1.4 ISOELAS

Identificateur d'une loi élastique 3D isotrope. Cette loi convient par exemple pour les éléments volumiques : hexaèdre, tétraèdre, pentaèdre par contre elle ne peut pas être utilisée directement pour des éléments plaques ou des éléments linéaires. Si on veut l'utiliser dans le cas de plaques il faut l'encapsuler dans une loi de contrainte plane (cf. 51.9) et pour des éléments 1D il faut l'encapsuler dans une loi doublement contrainte plane (cf. 51.10).

La loi nécessite la donnée d'un module d'Young et du coefficient de Poisson (exemple : cf. 106). 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 (114) donne un exemple de déclaration de loi.

Par défaut, comme il a été précisé au chapitre (49) la déformation utilisée est la mesure d'Almansi. Il est possible également d'utiliser la mesure de déformation logarithmique (cf.49). 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 114 – 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 --------
```

D'une manière plus générale, chaque paramètre matériau peut être défini à l'aide d'une fonction nD (cf. 47). Ainsi il est par exemple possible de définir un comportement qui dépend de la position initiale, ou à l'incrément précédent ou à l'instant présent (ex : matériau à gradient de propriétés). Les fonctions nD peuvent dépendre d'autres paramètres, en particulier les grandeurs globales, les données locales interpolées en particulier : le temps, la température si elle est définie ...

Un module d'Young dépendant d'une fonction nD se déclare via la série de 2 mots clés `E= E_fonction_nD:` suivis du nom de la fonction nD ou de la définition immédiate de la fonction nD.

De même un coefficient de Poisson dépendant d'une fonction nD se déclare via la série de 2 mots clés `nu= nu_fonction_nD:` suivis du nom de la fonction nD ou de la définition immédiate de la fonction nD.

La table (115) donne un exemple de mise en données.

TABLE 115 – Exemple de déclaration de la loi élastique 3D dont le module d'Young et le coefficient de Poisson dépendent d'une fonction nD.

```
  acier         ISOELAS
# .......  loi de comportement isoelastique 3D ........
#    module d'young :   coefficient de poisson
 E= E_fonction_nD: fct_E  nu= nu_fonction_nD: fct_nu
```

Il est possible de définir une dépendance uniquement pour "E" ou pour "ν".

— exemple où seul le coefficient de Poisson dépend d'une fonction nD

```
20000  nu= nu_fonction_nD: fct_nu
```

— exemple où seul le module d'Young dépend d'une fonction nD

```
E= E_fonction_nD: fct_E  0.3
```

51.2 Lois isotropes élastiques non-linéaires

Les lois isotropes élastiques non linéaires disponibles sont données dans la table (cf.116). 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 116 – liste des différentes lois isotropes élastiques non linéaires disponibles

identificateur	indications	ref du commentaire
ISO_ELAS_ESPO1D	1D : $\sigma = E(\varepsilon)\varepsilon = f(\lvert\varepsilon\rvert)E\varepsilon$	(51.2.1)
ISO_ELAS_ESPO3D	3D : $\sigma = E(\varepsilon)\varepsilon = f(\gamma)E\varepsilon$ avec $\gamma = \sqrt{2./3.\boldsymbol{\epsilon} : \boldsymbol{\epsilon}}$	(51.2.2)
ISO_ELAS_SE1D	1D : $\sigma = f(\epsilon)$	(51.2.3)

51.2.1 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. 308).

La loi nécessite donc la donnée du module d'Young et du coefficient de poisson. Ce dernier n'est pas utilisé pour le calcul des modules de compressibilité et de cisaillement nécessaire pour la mise à jour éventuelle de la section, et d'une fonction multiplicative $f(x)$: la table (117) donne un exemple de déclaration.

TABLE 117 – 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
```

51.2.2 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 : $\gamma = \sqrt{2./3.\boldsymbol{\epsilon} : \boldsymbol{\epsilon}}$. La loi est donc du type : $\sigma = E(\varepsilon)\varepsilon = f(\gamma)E\varepsilon$. La fonction f est générale, elle est choisie parmi la liste des fonctions 1D disponible (cf. 308).

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 (118) donne un exemple de déclaration.

TABLE 118 – 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
```

51.2.3 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. 308). 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 (119) et (120) donnent des exemples de déclaration.

Par défaut, la loi est construite pour une variation de section nulle. Néanmoins il est possible d'indiquer une variation des dimensions transversales d'une manière analogue aux autres lois 1D. Pour ce faire, on indique à la suite de la définition de la fonction, le mot clé " nu= " suivi de la valeur désirée. La table (121) donne un exemple d'utilisation. On

TABLE 119 – 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 120 – 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;
```

---

remarquera qu'il est possible d'indiquer directement une fonction avec le coef de Poisson ou alors d'utiliser une fonction déjà déclarée.

TABLE 121 – Exemple de déclaration de la loi élastique non linéaire ISO_ELAS_SE1D incluant un coefficient de Poisson non nul.

```
# ------------------------------------------------------------------------------------------"
# |......  loi de comportement isoelastique non lineaire 1D de type sigma = f(epsilon).......|"
# ------------------------------------------------------------------------------------------"
  loi_sangle3 ISO_ELAS_SE1D
#  f_coefficient CPL1D DdlP 0. 0.  1. 1.5e9  FdlP nu= 0.
f_coefficient courbe1 nu= 0.3
type_de_deformation DEFORMATION_LOGARITHMIQUE
```

51.3 Lois Hyper-élastiques

Les lois isotropes hyper-élastiques disponibles sont données dans la table (cf.122). Pour l'instant ces lois n'utilisent que la mesure de déformation d'Almansi. L'introduction d'une mesure logarithmique est en cours de développement.

TABLE 122 – liste des différentes lois isotropes hyper-élastiques disponibles

identificateur	indications	ref du commentaire
ISOHYPER3DFAVIER3	3D : potentiel proposé par Denis Favier	(51.3.1)
ISOHYPERBULK3	3D : potentiel pour la partie volumique seuil	(cf.51.3.2)
MOONEY_RIVLIN_1D	1D : loi classique de Mooney Rivlin	(51.3.4)
MOONEY_RIVLIN_3D	3D : loi classique de Mooney Rivlin	(51.3.5)
ISOHYPER3DORGEAS1	3D : potentiel proposé par Laurent Orgéas	(51.3.6)
ISOHYPER3DORGEAS2	3D : idem ISOHYPER3DORGEAS1 + amélioration des fonctions de dépendance à la phase	(cf.51.3.7)
POLY_HYPER3D	3D : loi polynomiale en invariants de même type que ceux de Mooney Rivlin	(51.3.8)
HART_SMITH3D	3D : loi de Hart Smith en 3D	(51.3.9)

51.3.1 ISOHYPER3DFAVIER3

Identificateur d'une loi 3D isotrope hyperélastique. Le potentiel est défini par l'expression suivante :

$$w = \frac{K}{6}\ln^2(V) + \frac{{Q_{or}}^2}{2\mu_0}\ln\left(\cosh\left(\frac{2\mu_0 Q_\varepsilon}{Q_{or}}\right)\right) + \mu_\infty {Q_\varepsilon}^2 \tag{52}$$

La variable "V" représente la variation relative de volume : $V = volume(t)/volume(t=0)$
Q_ε représente l'intensité (la norme) du déviateur des déformations.

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 (123) donne un exemple de déclaration.

Il est également possible d'introduire l'influence de la phase, selon la formule suivante :

$$\begin{aligned} Q_r &= \frac{Q_{or}}{(1+\gamma_Q cos(3\varphi_\varepsilon))^{n_Q}} \\ \mu_\infty &= \frac{\mu_{o\infty}}{(1+\gamma_\mu\ cos(3\varphi_\varepsilon))^{n_\mu}} \end{aligned} \tag{53}$$

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 :

TABLE 123 – 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
```

— 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 (124) donne un exemple de déclaration.

TABLE 124 – 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
```

Comme pour le potentiel ISOHYPER3DORGEAS1 Il est possible d'introduire un paramètre supplémentaire optionnel : " avec_regularisation_ ". On se reportera à 51.3.6 pour plus de détails.

Il est également possible de récupérer différentes grandeurs de travail, par exemple l'intensité du potentiel, mais ces grandeurs qui sont calculées au moment de la résolution ne sont pas en général stockées. Pour pouvoir les récupérer, il faut activer un mot clé lors

de la déclaration des paramètres de la loi de comportement. Plus précisément on indique comme **dernier paramètre** : le mot clé " sortie_post_ " suivi de 1. Ensuite au moment de la définition des grandeurs à sortir on aura une alimentation des variables suivantes :

— " POTENTIEL " : la valeur du potentiel

— " COS3PHI_EPS " : le cos de l'angle de phase du tenseur de déformation

— " Q_EPS " : l'intensité du déviateur de déformation ($\sqrt{(\bar{\varepsilon} : \bar{\varepsilon})}$)

— " V_vol " : la variation relative de volume (vol_t/vol_0)

Par défaut la variable " sortie_post_ " est false (=0), les grandeurs de travail seront nulles en sortie. Si ces grandeurs ne sont pas nécessaires, il est préférable de garder la valeur par défaut, ce qui permet de limiter la zone de stockage. Ce paramètre soit est le dernier paramètre sur la dernière ligne de définition des paramètres spécifiques de la loi (hors niveau de commentaire ou type de déformation par exemple).

51.3.2 ISOHYPERBULK3

Identificateur d'une loi 3D isotrope hyperélastique. Le potentiel est défini par l'expression suivante :

$$w = \frac{K}{6}\ln^2(V) \tag{54}$$

La variable "V" représente la variation relative de volume : $V = volume(t)/volume(t = 0)$

La loi nécessite la donnée de 1 paramètre, le coefficient de compressibilité volumique : K, équivalent au cas de Hooke à : $E/(1-2\nu)$

La table (125) 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 125 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 .

```
#-----------------------------------------------------
# Nom Materiau  |  Type loi       |   Potentiel     |
#-----------------------------------------------------
     MATE3        ISOHYPERBULK3

#-----------
#     K     |
#-----------
  270000
```

Il est également possible de définir un module de compressibilité qui dépend de la température etou de la variation de volume. La dépendance est multiplicative.

$$K(T, V) = f(T) \times g(V) \times K_0 \tag{55}$$

Les fonctions $f(T)$ et $g(V)$ peuvent être quelconques, en particulier elles peuvent être non-linéaires. La table 126 donne un exemple de thermodépendance. La table 127 donne un exemple de dépendance à la variation de volume. La table 128 donne un exemple dépendance à la variation de volume et à la température, via des fonctions 1D définies au niveau des courbes générales. Enfin la table 129 présente un exemple de déclaration via des courbes définies dans la loi.

TABLE 126 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 thermodépendante.

```
# un exemple de declaration: avec courbe1 le nom d'une courbe 1D
#----------------
#     K          |
#----------------
     160000    K_thermo_dependant_  courbe1
```

TABLE 127 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 dont le module de compressibilité dépend de la variation de volume.

```
# un exemple de declaration: avec courbe2 le nom d'une courbe 1D :
#----------------
#     K          |
#----------------
     160000    K_V_dependant_  courbe2
```

TABLE 128 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 avec une dépendance à la température et à la variation de volume. Les courbes "courbe1" et "courbe2" doivent avoir été définies au niveau des courbes générales.

```
#----------------
#     K          |
#----------------"
     160000    K_thermo_dependant_  courbe1  K_V_dependant_  courbe2
```

Comme pour le potentiel `ISOHYPER3DORGEAS1` Il est possible d'introduire un paramètre supplémentaire optionnel : " `avec_regularisation_` ". On se reportera à 51.3.6 pour plus de détails.

TABLE 129 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK3 avec une dépendance à la température et à la variation de volume, via des courbes définies dans la loi, ici il s'agit de courbes poly-linéaires.

```
#
# comme pour toutes les lois, la declaration de chaque courbe peut etre
# effectuee via un nom de courbe deja existante ou en declarant
# directement la courbe, dans ce dernier cas, ne pas oublier de finir
# chaque declaration de courbe avec un retour chariot (return)
#  un exemple de declaration:
        #----------------
        #     K         |
        #----------------
             160000    K_thermo_dependant_  CPL1D DdlP 0.  0.  1.  1.  FdlP
                       K_V_dependant_  CPL1D DdlP 0.  0.  1.  2.  FdlP
```

Comme pour le potentiel " ISOHYPER3DFAVIER3 " il est également possible de récupérer différentes grandeurs de travail via l'activation du mot clé " sortie_post_ ". On se reportera à 51.3.1 pour une description plus précise.

51.3.3 ISOHYPERBULK_GENE

Il s'agit ici d'une loi de comportement qui concerne uniquement le changement de volume en 3D isotrope.

Le potentiel est défini par l'expression suivante :

$$\omega = \omega(V) \tag{56}$$

Où "ω" est une fonction quelconque de la variable "V", qui représente la variation relative de volume : $V = volume(t)/volume(t = 0)$

Ce potentiel généralise le cas " ISOHYPERBULK3 " en évitant l'utilisation de la fonction "log". Dans le cas d'une évolution quelconque du module de compressibilité, l'identification de la fonction "F" est a priori plus simple via le potentiel " ISOHYPERBULK_GENE " que via le potentiel " ISOHYPERBULK3 ". C'est ce qui a motivé sa création. En particulier, la forme du potentiel $F(V)$ n'est soumise à aucune limite.

La loi nécessite donc la définition de la fonction "$\omega(V)$".

Remarque : On notera cependant que la détermination des contraintes nécessite le calcul de "$\omega(V)$" mais également celui de ses dérivées première et seconde (cf. [Rio, 2015]). Aussi soit on utilise une courbe dont on définie explicitement les dérivées première et seconde (cf. par exemple : 81.1.21) soit le logiciel calculera automatiquement par différences finis (par exemple) les dérivées. Dans ce dernier cas il faut veiller à ce que "$\omega(V)$" soit toujours correctement définie (par exemple pour des valeurs supérieures et inférieures à 1 qui est la valeur initiale de V).

La table (130) donne un exemple de déclaration, dans lequel on utilise une fonction préalablement définie au niveau des courbes 1D.

TABLE 130 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK_GENE, "courbe2" est le nom d'une fonction qui doit avoir été préalablement définie

```
# .......  loi de comportement 3D hyperelastique isotrope ISOHYPERBULK_GENE
#----------------------------------
#     omega    |  fonction     |
#----------------------------------
   omega_V=     courbe2
```

Il est également possible de définir la courbe dans la loi cf. La table (131). De manière optionnelle, le potentiel hyperélastique peut dépendre de la température de manière multiplicative.

$$\omega(T, V) = f(T) \times \omega(V) \tag{57}$$

La fonction $f(T)$ peut être quelconque. Sa présence est définie à l'aide du mot clé " omega_thermo_dependant_ " suivi de la courbe. La table 132 donne un exemple de thermodépendance à l'aide d'une courbe préalablement définie. La table 133 donne un exemple de thermodépendance à l'aide de la définition d'une fonction. La table 134

TABLE 131 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK_GENE via la définition d'une fonction

```
 # .......  loi de comportement 3D hyperelastique isotrope ISOHYPERBULK_GENE
 #-------------------------
 #     omega    |  fonction     |
 #-------------------------
   omega_V=   CPL1D DdlP  0.9  8000.  1.  6000.  1.1 5000.   FdlP
```

TABLE 132 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK_GENE avec thermodépendance multiplicative préalablement définie.

```
# .......  loi de comportement 3D hyperelastique isotrope ISOHYPERBULK_GENE
#---------------------------------
#     omega    |  fonction     |
#---------------------------------
   omega_V=   CPL1D DdlP  0.9  8000.  1.  6000.  1.1 5000.   FdlP
   omega_thermo_dependant_  courbe1
```

TABLE 133 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK_GENE avec thermodépendance via la définition d'une fonction.

```
# .......  loi de comportement 3D hyperelastique isotrope ISOHYPERBULK_GENE
#-------------------------
#     omega    |  fonction     |
#-------------------------
   omega_V=   CPL1D DdlP  0.9  8000.  1.  6000.  1.1 5000.   FdlP
   omega_thermo_dependant_  CPL1D DdlP  200  2.  273.  1. 300. 0.5   FdlP
```

donne un exemple de thermodépendance et de fonction "$\omega(V)$" utilisant des fonctions préalablement définies.

Comme pour le potentiel " ISOHYPER3DFAVIER3 " Il est également possible de récupérer différentes grandeurs de travail via l'activation du mot clé " sortie_post_ ". On se reportera à 51.3.1 pour une description plus précise.

TABLE 134 – Exemple de déclaration de la loi hyperélastique ISOHYPERBULK_GENE avec thermodépendance via la définition d'une fonction.

```
# .......  loi de comportement 3D hyperelastique isotrope ISOHYPERBULK_GENE
#---------------------------------
#     omega   |  fonction     |
#---------------------------------
   omega_V=   courbe1    omega_thermo_dependant_  courbe2
```

51.3.4 MOONEY_RIVLIN_1D

Identificateur d'une loi 1D isotrope de type Mooney Rivlin hyperélastique. Il s'agit du cas particulier d'un comportement 1D supposé incompressible.

Bien noter qu'il s'agit dans cas particulier, assez courant, mais qui ne couvre pas tous les cas 1D. Il est possible d'utiliser un comportement plus général, par exemple non incompressible, via la loi 3D de Mooney Rivlin (cf. 51.3.5) et un comportement de contrainte plane double (cf. 51.10).

Dans le cas ici particulier d'un comportement 1D supposé incompressible on obtient le comportement classique suivant, fonction de l'élongation $\lambda = \frac{L}{L_0}$:

$$\sigma = 2(\lambda^2 - \frac{1}{\lambda})(C_{10} + \frac{1}{\lambda}C_{01}) \tag{58}$$

Par défaut, la mesure de déformation utilisée par Herezh est celle d'Almansi et l'on a pour une interpolation linéaire :

$$\epsilon_1^1 = 0.5(1. - \frac{1}{\lambda^2}) \text{ ou encore } \lambda = \frac{1}{\sqrt{1. - 2.\epsilon_1^1}} \text{ et } \frac{1}{\lambda^2} = 1. - 2.\epsilon_1^1 \tag{59}$$

Cette équation permet de retranscrire la relation (58) dans Herezh de la manière suivante :

$$\sigma_1^1 = 2.C_{10}\left(\frac{1.}{(1. - 2.\epsilon_1^1)} - \sqrt{1. - 2.\epsilon_1^1}\right) + 2.C_{01}\left(\frac{1.}{\sqrt{1. - 2.\epsilon_1^1}} - 1. + 2.\epsilon_1^1\right) \tag{60}$$

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 (135) donne un exemple de déclaration.

TABLE 135 – 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) à l'aide de deux mots clés : `C01_thermo_dependant_` et `C10_thermo_dependant_` . La table (136) donne un exemple de déclaration dans le cas où les deux paramètres sont

TABLE 136 – 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_thermo_dependant_ C10_temperature_ courbe2
#   .. fin de la definition de la loi mooney rivlin
```

TABLE 137 – 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_thermo_dependant_ 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.
```

thermo-dépendants, la table (137) donne un exemple lorsque seul le second paramètre est thermo-dépendant.

Comme pour le potentiel ISOHYPER3DORGEAS1 Il est possible d'introduire un paramètre supplémentaire optionnel : " avec_regularisation_ ". On se reportera à 51.3.6 pour plus de détails.

Comme pour le potentiel " ISOHYPER3DFAVIER3 " il est également possible de récupérer différentes grandeurs de travail via l'activation du mot clé " sortie_post_ ". On se

reportera à 51.3.1 pour une description plus précise.

51.3.5 MOONEY_RIVLIN_3D

Identificateur d'une loi 3D isotrope de type Mooney Rivlin hyperélastique. Le potentiel est défini par l'expression suivante :

$$W_{Mooney-Rivlin} = (C_{10}\ (J_1 - 3) + C_{01}\ (J_2 - 3)) + \left(K\left[1 - \frac{1 + ln(\sqrt{I_3})}{\sqrt{I_3}}\right]\right) \quad (61)$$

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-mêmes calculées à partir du tenseur de Cauchy-Green gauche :$\boldsymbol{B} = {}^{t}_{0}\boldsymbol{G} = g^{ij}\ \hat{\vec{g}}_i \otimes \hat{\vec{g}}_j$.

On se réfèrera au document "hyper-elasticite.pdf" pour une explication précise de la signification des différents invariants utilisés dans le calcul du potentiel.

La table (138) donne un exemple de déclaration.

TABLE 138 – 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 (139) 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 disponibles :

$$\begin{aligned} W_{v1} &= K\left[1 - \frac{1 + ln(\sqrt{I_3})}{\sqrt{I_3}}\right] \\ W_{v2} &= \frac{K}{2}(V - 1) \\ W_{v3} &= \frac{K}{2}(log(V))^2 \\ W_{v4} &= \frac{K}{2}(V - 1)^2 \end{aligned} \quad (62)$$

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 paramètres de la loi, ce qui donne par exemple :

" C01= 0.0167 C10= 0.145 K= 3000 type_potvol_ 3 "

TABLE 139 – 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_thermo_dependant_ courbe2
  K= K_thermo_dependant_ courbe3
#   .. fin de la definition de la loi money rivlin
```

Après le type de variation volumique, on peut indiquer facultativement l'ajout au potentiel d'un terme permettant de raidir le comportement à partir d'un certain niveau de chargement. Cette partie additionnelle s'écrit :

$$W_{courbure} \overset{note}{=} W_c = \frac{1}{\sqrt{I_3}} \left(\left(\frac{(J_1 - 3)^{2r+1}}{a^{2r}} \right) \frac{1}{(2r+1)} \right) \tag{63}$$

Cette partie dépend donc de J_1 et $I_3 = J_3$.

Pour mettre en place ce comportement de raidissement, on indique le mot cle : " avec_courbure_ " puis on change de ligne. Le terme additionnel dépend de deux paramètres : "a" et "r", "a" positionne la valeur de J1 " à partir de laquelle il y a durcissement, "r" contrôle la courbure du changement de régime. La table 140 donne un exemple de déclaration.

Les deux paramètres peuvent "a" et "r", peuvent être l'un et/ou l'autre dépendant de la température. Dans un cas de dépendance, la déclaration de dépendance suit les règles habituelles.

La table 141 donne un exemple de déclaration dans le cas d'une dépendance à la température du terme de raidissement.

Il est possible de récupérer différentes grandeurs de travail, par exemple l'intensité du potentiel, mais ces grandeurs qui sont calculées au moment de la résolution ne sont pas en général stockées. Pour pouvoir les récupérer, il faut activer un mot clé lors de la déclaration des paramètres de la loi de comportement. Plus précisément on indique comme **dernier**

TABLE 140 – Exemple de déclaration de la loi de Mooney Rivlin en 3D avec un terme de raidissement.

```
#-------------------------------------
# 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   type_potvol_ 2  avec_courbure_
  a_courbure= 94 r_courbure= 1.24
```

TABLE 141 – Exemple de déclaration de la loi de Mooney Rivlin en 3D avec un terme de raidissement qui dépend de la température.

```
#-------------------------------------
# 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   type_potvol_ 2  avec_courbure_
  a_courbure= a_thermo_dependant_
  r_courbure= r_thermo_dependant_
```

paramètre : le mot clé " sortie_post_ " suivi de 1. Ensuite au moment de la définition des grandeurs à sortir on aura une alimentation des variables suivantes :

— "POTENTIEL" : la valeur du potentiel

Par défaut la variable " sortie_post_ " est fausse (=0), les grandeurs de travail seront nulles en sortie. Si ces grandeurs ne sont pas nécessaires, il est préférable de garder la valeur par défaut, ce qui permet de limiter la zone de stockage.

51.3.6 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 :

$$
\begin{aligned}
\omega_4 &= \frac{K_{rev}}{6}\ln^2(V) + Q_{\sigma_{rev}}Q_\epsilon + \mu_{2rev}Q_\epsilon^2 \\
&+ \frac{\mu_{1rev}}{2}\Bigg[Q_\epsilon\left(Q_\epsilon + 2Q_{\epsilon c}\right) - \left(Q_\epsilon + Q_{\epsilon c}\right)\left(\alpha_{1rev}^2 + \left(Q_\epsilon + Q_{\epsilon c}\right)^2\right)^{1/2} + Q_{\epsilon c}\left(\alpha_{1rev}^2 + Q_{\epsilon c}^2\right)^{1/2} \\
&- \alpha_{1rev}^2\left(\ln\left|Q_\epsilon + Q_{\epsilon c} + \left(\alpha_{1rev}^2 + \left(Q_\epsilon + Q_{\epsilon c}\right)^2\right)^{1/2}\right| - \ln\left|Q_{\epsilon c} + \left(\alpha_{1rev}^2 + Q_{\epsilon c}^2\right)^{1/2}\right)\right|\Bigg] \\
&+ \frac{\mu_{3rev}}{2}\Bigg[Q_\epsilon\left(Q_\epsilon - 2\left(Q_{\epsilon_{rev}}^2 + \alpha_{3rev}^2\right)^{1/2}\right) \\
&+ \alpha_{3rev}^2\left(\ln\left|Q_\epsilon - Q_{\epsilon_{rev}} + \left(\alpha_{3rev}^2 + \left(Q_\epsilon - Q_{\epsilon_{rev}}\right)^2\right)^{1/2}\right| - \ln\left|-Q_{\epsilon_{rev}} + \left(\alpha_{3rev}^2 + Q_{\epsilon_{rev}}^2\right)^{1/2}\right)\right| \\
&+ \left(Q_\epsilon - Q_{\epsilon_{rev}}\right)\left(\alpha_{3rev}^2 + \left(Q_\epsilon - Q_{\epsilon_{rev}}\right)^2\right)^{1/2} + Q_{\epsilon_{rev}}\left(\alpha_{3rev}^2 + Q_{\epsilon_{rev}}^2\right)^{1/2}\Bigg] \qquad (64)
\end{aligned}
$$

La table (142) 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 142 – 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 :

$$
\begin{aligned}
Q_{\sigma_{rev}} &= \frac{Q_{\sigma_{0rev}}}{\left(1+\gamma_{Q_{\sigma_{rev}}}\cos(3\varphi_\epsilon)\right)^{n_{Q_{\sigma_{rev}}}}} \\
Q_{\epsilon_{rev}} &= \frac{Q_{\epsilon_{0rev}}}{\left(1+\gamma_{Q_{\epsilon_{rev}}}\cos(3\varphi_\epsilon)\right)^{n_{Q_{\epsilon_{rev}}}}} \\
\mu_{1rev} &= \frac{\mu_{01rev}}{\left(1+\gamma_{\mu_{1rev}}\cos(3\varphi_\epsilon)\right)^{n_{\mu_{1rev}}}} \\
\mu_{2rev} &= \frac{\mu_{02rev}}{\left(1+\gamma_{\mu_{2rev}}\cos(3\varphi_\epsilon)\right)^{n_{\mu_{2rev}}}} \\
\mu_{3rev} &= \frac{\mu_{03rev}}{\left(1+\gamma_{\mu_{3rev}}\cos(3\varphi_\epsilon)\right)^{n_{\mu_{3rev}}}} \qquad (65)
\end{aligned}
$$

La table (143) donne un exemple de déclaration avec phase. $Q_{\sigma_{0rev}}$ est représenté par Qs, μ_{02rev} par Qe, 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 nQs et $gammaQs$, 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, $\mu 1$, $\mu 2$, $\mu 3$. Enfin il est également

TABLE 143 – Exemple de déclaration de la loi d'hyper-élasticité Orgéas 1 avec dépendance à la phase.

```
hyperAvecPhase       ISOHYPER3DORGEAS1
# .......  loi de comportement 3D hyperelastique isotrope Orgeas 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
```

possible de prendre en compte une dépendance à la température. Dans une première étape seule les paramètres 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 :

$$\begin{aligned} si \;\; T \geqslant T_c \;\; alors \;\; Q_s &= \frac{1}{2} G_r \left[(T - T_c + T_s) - \sqrt{(T - T_c + T_s)^2 + a_r^2} \right] + Q_{rmax} \\ sinon \quad Q_s &= \frac{1}{2} G_r \left[(T - T_c - T_s) + \sqrt{(T - T_c - T_s)^2 + a_r^2} \right] \end{aligned} \tag{66}$$

avec

$$\begin{aligned} T_c &= T_{0r} + \frac{Q_{rmax}}{2\ G_r} \\ T_s &= \frac{a_r^2 - (Q_{rmax/G_r})^2}{2\ (Q_{rmax/G_r})} \end{aligned} \tag{67}$$

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 :

$$Q_e(T) = Q_e(T_{0rQe}) + \frac{Q_s(T) - Q_s(T_{0rQe})}{3\ h_1\ (mu3 + mu2)} \tag{68}$$

Cette évolution est contrôlée par trois paramètres : une valeur à donner de $Q_e = Q_e(T_{0rQe})$ pour une température de référence T_{0rQe} à 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 (144) donne un exemple de loi dépendant de la phase et de la température selon l'évolution proposée par Laurent Orgéas dans sa thèse.

TABLE 144 – 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 (145) donne un exemple d'une telle dépendance. On observe que l'évolution de Q_{0s} est ici gérée par la courbe "courbe_evol_Q0s" qui doit donc avoir été définit auparavant. Cette courbe peut-être quelconque.

TABLE 145 – 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
```

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 (147) 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 146 – Exemple de déclaration de la loi d'hyper-élasticité Orgéas avec dépendance à la phase et une dépendance à la température des paramètres 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  \
  #  avec_regularisation_ 1.e-6
```

Il est possible d'introduire un paramètre supplémentaire optionnel : " avec_regularisation_ ". En fait à l'origine des déformations, le potentiel induit des indéterminations. Celles-ci sont "régularisées" via l'introduction d'un petit paramètre valant par défaut 1. 10^{-12}. Il est possible de donner une autre valeur à l'aide du mot clé " avec_regularisation_ " suivi d'une nouvelle valeur. Ce paramètre doit se situer après tous les autres spécifiques

à la loi, mais avant le paramètre sortie_post_ (cf. le paragraphe qui suit). La table (147) indique un exemple à condition de décommenter la ligne.

Comme pour le potentiel " ISOHYPER3DFAVIER3 " Il est également possible de récupérer différentes grandeurs de travail via l'activation du mot clé " sortie_post_ ". On se reportera à 51.3.1 pour une description plus précise.

51.3.7 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 situent 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 différente et la prise en compte de la phase est également légèrement différente. La table (147) donne un exemple d'utilisation. Dans cet 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 définie (comme habituellement pour les autres lois de comportement), ici c'est par exemple le cas pour les variables mu03_phi avec la courbe3 (nom de courbe devant être définie au début du fichier .info).

Une fois ces informations lues, la valeur des paramètres est multipliée par la fonction associée dépendant de la phase. Par exemple $mu2 = mu02.(gamma + alpha.(cos(3\varphi))^2)^n$.

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.

TABLE 147 – 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 Orgeas 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_
```

Comme pour le potentiel ISOHYPER3DORGEAS1 Il est possible d'introduire un paramètre supplémentaire optionnel : " avec_regularisation_ ". On se reportera à 51.3.6 pour plus de détails.

Comme pour le potentiel " ISOHYPER3DFAVIER3 " il est également possible de récupérer différentes grandeurs de travail via l'activation du mot clé " sortie_post_ ". On se reportera à 51.3.1 pour une description plus précise.

51.3.8 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 suivante :

$$W_{polynomiale} = \sum_{i+j=1}^{n} \left(C_{ij}\ (J_1-3)^i\ (J_2-3)^j\right) + \left(K\left[1-\frac{1+ln(\sqrt{I_3})}{\sqrt{I_3}}\right]\right) \quad (69)$$

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-mêmes calculées à partir du tenseur de Cauchy-Green gauche :$\boldsymbol{B} = {}^{t}_{0}\boldsymbol{G} = g^{ij}\ \ \hat{\vec{g}}_i \otimes \hat{\vec{g}}_j$.

On se réfèrera au document "hyper-elasticite.pdf" pour une explication précise de la signification des différents invariants utilisés dans le calcul du potentiel.

La table (148) 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.

TABLE 148 – 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
#-------------------------------------------------------------------------------
```

Tous les coefficients matériels peuvent être thermo-dépendants. La table (149) 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 149 – 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_thermo_dependant_ courbe2
                C02= C02_thermo_dependant_ courbe2
                C11= C10_thermo_dependant_ courbe2
                C20= 0.0167  K=  K_thermo_dependant_ 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 defaut il s'agit de : K(1-(1+log(V))/V) qui correspond au type 1
#   mais on peut choisir:                     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 defaut type_potvol_ a la valeur 1
#-------------------------------------------------------------------------------"
```

Comme pour le potentiel " MOONEY_RIVLIN_3D ", après le type de variation volumique on peut indiquer facultativement l'ajout au potentiel d'un terme permettant de raidir le comportement à partir d'un certain niveau de chargement. Cette partie additionnelle s'écrit :

$$W_{courbure} \overset{note}{=} W_c = \frac{1}{\sqrt{I_3}} \left(\left(\frac{(J_1 - 3)^{2r+1}}{a^{2r}} \right) \frac{1}{(2r+1)} \right) \tag{70}$$

Cette partie dépend donc de J_1 et $I_3 = J_3$. La syntaxe pour indiquer l'ajout d'un terme de raidissement est identique au cas du potentiel " MOONEY_RIVLIN_3D ", cf. 140 et 141 pour des exemples de déclaration.

Comme pour le potentiel " MOONEY_RIVLIN_3D " il est également possible de récupérer différentes grandeurs de travail via l'activation du mot clé " sortie_post_ ". On se reportera à 51.3.5 pour une description plus précise.

51.3.9 HART_SMITH3D

Identificateur d'une loi 3D isotrope de type Hart Smith hyperélastique. Le potentiel est défini par l'expression suivante :

$$W_{Hart_Smith} = C_1 \int_3^{J_1(finale)} exp[C_3\ (J_1-3)^2]\ dJ_1 + C_2\ log(\frac{J_2}{3}) + \left(K\left[1-\frac{1+ln(\sqrt{I_3})}{\sqrt{I_3}}\right]\right) \quad (71)$$

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-mêmes calculées à partir du tenseur de Cauchy-Green droit :$\boldsymbol{C} = {}^t_{0..}\boldsymbol{G} = \hat{g}_{ij}\ \ \vec{g}^i \otimes \vec{g}^j$.

La table (150) donne un exemple de déclaration.

TABLE 150 – 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
```

Comme dans le cas de Mooney Rivlin, les coefficients matériels peuvent être thermo-dépendants. La table (151) 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 au changement de volume. Quatre cas (idem Mooney-Rivlin) sont disponibles :

$$\begin{aligned} W_{v1} &= K\left[1-\frac{1+ln(\sqrt{I_3})}{\sqrt{I_3}}\right] \\ W_{v2} &= \frac{K}{2}(V-1) \\ W_{v3} &= \frac{K}{2}(log(V))^2 \\ W_{v4} &= \frac{K}{2}(V-1)^2 \end{aligned} \quad (72)$$

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 paramètres de la loi, ce qui donne par exemple :

```
C1= 1. C2= 0.145 C3= 1.e-4 K= 100 type_potvol_ 3
```

TABLE 151 – Exemple de déclaration de la loi de Hart Smith en 3D, dans le cas où les quatre coefficients matériaux sont thermo-dépendants.

```
          #-------------------------------------
          # Nom Materiau  |  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_thermo_dependant_ courbe3
  K= K_thermo_dependant_ 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.
#  par exemple supposons C1 et C2 fixes et K thermo-dependant, on ecrit:
#-------------------------------------------------------------------------------------
#  C1= 1.  C2= 0.145 C3= C3_thermo_dependant_ courbe4
#  K= K_thermo_dependant_ courbe4
#-------------------------------------------------------------------------------------
          #   .. fin de la definition 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 obtenu à l'aide d'un potentiel additionnel proposé par Moreau-Rio-Thuillier selon :

$$W_{courbure} = \frac{1}{\sqrt{I_3}} \left((J_1 - 3) \left(\frac{J_1 - 3}{a} \right)^{2r} \frac{1}{(2r+1)} \right) \tag{73}$$

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 [Moreau, 2000], 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 (152) donne un exemple de déclaration pour des paramètres constants.

TABLE 152 – 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 (153) donne un exemple de déclaration pour des paramètres dépendants de la température.

TABLE 153 – 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
```

Comme pour le potentiel " MOONEY_RIVLIN_3D " Il est également possible de récupérer différentes grandeurs de travail via l'activation du mot clé " sortie_post_ ". On se reportera à 51.3.5 pour une description plus précise.

51.3.10 MAHEO_HYPER

Identificateur d'une loi 3D isotrope hyperélatique développée dans le cadre de travaux sur les mousses en collaboration avec Laurent Mahéo du LimatB.

Le potentiel suit une évolution en tangente hyperbolique fonction du seul invariant J_1 est défini par l'expression suivante :

$$W_{Maheo\_Hyper} = \frac{Q^2_{\sigma_{rev}}}{\mu_{1rev}} \ln\left[\cosh\left(\frac{\mu_{1rev}}{Q_{\sigma_{rev}}}\sqrt{(J_1-3)}\right)\right] + \mu_{2rev}\,(J_1-3) \qquad (74)$$

On se réfèrera au document "hyper-elasticite.pdf" pour une explication précise de la signification de l'invariant J_1 utilisé dans le calcul du potentiel.

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-mêmes calculées à partir du tenseur de Cauchy-Green gauche :$\boldsymbol{B} = {}^{t}_{0}\boldsymbol{G} = g^{ij}\ \hat{\vec{g}}_i \otimes \hat{\vec{g}}_j$.

Les coefficients matériau sont μ_{1rev} qui représente la pente à l'origine, μ_{2rev} la pente à l'infini et $Q_{\sigma_{rev}}$ positionne le changement de pente.

La table (154) donne un exemple de déclaration. On remarque un paramètre supplémentaire qui en fait est optionnel : " avec_regularisation_ ". En fait à l'origine des déformations, le potentiel induit des indéterminations. Celles-ci sont "régularisées" via l'introduction d'un petit paramètre valant par défaut 1. 10^{-12}. Il est possible de donner une autre valeur à l'aide du mot clé " avec_regularisation_ " suivi d'une nouvelle valeur.

TABLE 154 – Exemple de déclaration de la MAHEO_HYPER.

```
MAHEO_HYPER
# ------------------------------------------------------------------------------
# |... loi de comportement hyper elastique 3D Maheo_hyper                  .. |
# ------------------------------------------------------------------------------
Qsig_rev= 0.430180937511839   mu1_rev= 10.4913717466094  mu2_rev= 0.460497314799597 \
avec_regularisation_  1.e-12
```

Tous les coefficients matériels peuvent être thermo-dépendants. La table (155) 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.

Pour l'introduction du paramètre supplémentaire optionnel : " avec_regularisation_ " on se reportera à 51.3.6 pour plus de détails.

Comme pour le potentiel " MOONEY_RIVLIN_3D " Il est également possible de récupérer différentes grandeurs de travail via l'activation du mot clé " sortie_post_ ". On se reportera à 51.3.5 pour une description plus précise.

TABLE 155 – Exemple de déclaration de la loi MAHEO_HYPER, dans le cas où les trois coefficients matériaux sont thermo-dépendants.

```
# ------------------------------------------------------------------------------
# |... loi de comportement hyper elastique 3D Maheo_hyper                    .. |
# ------------------------------------------------------------------------------
Qsig_rev= Qsig_rev_thermo_dependant_ courbe1
mu1_rev= mu1_rev_temperature_ courbe2
mu2_rev= mu2_rev_temperature_ courbe3
#  ou encore
#  Qsig_rev= Qsig_rev_temperature_ courbe2
#  mu1_rev= 0.0167  mu2_rev=  mu2_rev_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.
#-------------------------------------------------------------------------------
#  il est possible d'indiquer un facteur de regularisation qui permet d'eviter
#  de potentiels  problemes de NaN, de type division par 0 par exemple
#  1/a est remplace par 1/(a+fact_regularisation), par defaut fact_regularisation = 1.e-12
#  pour indiquer un facteur de regulation non nul on indique en dernier parametre
#  le mot cle avec_regularisation_ suivi du facteur voulu
#  ex:
#    avec_regularisation_  1.e-12
```

51.4 Lois élasto-plastiques.

Les lois élasto-plastiques disponibles sont données dans la table (cf.156).

TABLE 156 – liste des différentes lois isotropes élasto-plastiques disponibles

identificateur	indications	ref du commentaire
PRANDTL_REUSS1D	1D : Prandtl Reuss grandes déformations	(51.4.1)
PRANDTL_REUSS2D_D	2D : Prandtl Reuss grandes déformations déformations planes	(51.4.2)
PRANDTL_REUSS	3D : Prandtl Reuss grandes déformations	(51.4.3)

Les lois disponibles sont :

51.4.1 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 classiques (E et ν), la donnée d'un seuil d'écrouissage et d'une courbe d'écrouissage présentée sous forme d'une suite de points. La table (157) 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 segments joignant des points. Les points sont donnés entre deux mots clés : " Debut_des_coordonnees_des_points " et " Fin_des_coordonnees_des_points ". Chaque point est défini par également un mot clé : " Coordonnee ", ensuite on indique la la dimension puis les 2 coordonnées.

TABLE 157 – 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
```

51.4.2 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 classiques (E et ν), la donnée d'un seuil d'écrouissage et d'une courbe d'écrouissage présentée sous forme d'une suite de points. Les données sont identiques au cas Prandtl Reuss 1D , sauf le titre de la loi, ici " PRANDTL_REUSS2D_D ".

51.4.3 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 classiques (E et ν), la donnée d'un seuil d'écrouissage et d'une courbe d'écrouissage présentée sous forme d'une suite de points. Les données sont identiques au cas Prandtl Reuss 1D , sauf le titre de la loi, ici " PRANDTL_REUSS ".

51.5 Lois visco_élastiques isotropes.

Les lois visco_élastiques isotropes disponibles sont données dans la table (cf.158).

TABLE 158 – liste des différentes lois visco-élastiques isotropes disponibles

identificateur	indications	ref du commentaire
NEWTON1D	1D : $\boldsymbol{\sigma} = \mu \boldsymbol{D}$ ou $\boldsymbol{\sigma} = \mu (\boldsymbol{D} : \boldsymbol{D})^{n/2} \boldsymbol{D}$	(51.5.1)
NEWTON2D_D	2D : déformation plane $\boldsymbol{\sigma} = \mu \bar{\boldsymbol{D}}$ ou $\boldsymbol{\sigma} = \mu (\bar{\boldsymbol{D}} : \bar{\boldsymbol{D}})^{n/2} \bar{\boldsymbol{D}}$	(51.5.2)
NEWTON3D	3D : $\boldsymbol{\sigma} = \mu \bar{\boldsymbol{D}}$ ou $\boldsymbol{\sigma} = \mu (\bar{\boldsymbol{D}} : \bar{\boldsymbol{D}})^{n/2} \bar{\boldsymbol{D}}$	(51.5.3)
MAXWELL1D	1D : un ressort et un amortisseur en série	(51.5.4)
MAXWELL3D	3D : un ressort et un amortisseur en série en 3D	(51.5.6)

51.5.1 NEWTON1D

Identificateur d'une loi 1D isotrope viscoélastique de type Newton : $\boldsymbol{\sigma} = \mu \boldsymbol{D}$. 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 : $\boldsymbol{\sigma} = \mu \left((\boldsymbol{D} : \boldsymbol{D}) + D_0 \right)^{n/2} \boldsymbol{D}$ (ici la distinction déviatoire ou non n'a pas lieu d'être).

Le paramètre D_0 est une constante (par défaut = 0.01) qui permet d'éviter d'avoir une dérivée ∞ quand "n" est négatif (ce qui est courant). On utilise le mot clé : $Deps0 =$ pour changer sa valeur.

La table (159) donne un exemple de déclaration.

TABLE 159 – 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 | puis eventuellement          |"
# |                  |  pour une evolution non lineaire | une vitesse limite           |"
# | mu (obligatoire) |      xn (facultatif)             | inferieur (par defaut: 0.01) |"
#.................................................................................."

          mu=    47.262609     xn= -0.89   Deps0= 0.001

                          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 (160) donne un exemple de déclaration de coefficient thermo-dépendant.

TABLE 160 – Exemple de déclaration d'une loi de Newton en 1D dont le coefficient mu dépend de la température

```
 #-------- debut cas d'une loi de newton thermodependante --------
  polymere       NEWTON1D
# ........................  loi de comportement Newton 1D ........................."
# | viscosite        |  et eventuellement une puissance | puis eventuellement          |"
# |                  |  pour une evolution non lineaire | une vitesse limite           |"
# | mu (obligatoire) |      xn (facultatif)             | inferieur (par defaut: 0.01)|"
#.................................................................................."
    mu=  mu_thermo_dependant_ CPL1D DdlP 0.  15 100. 50  FdlP
    xn= -0.89 Deps0= 0.001
                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_ ". Remarquer qu'après la définition de la courbe de température on passe à la ligne suivante.

51.5.2 NEWTON2D_D

Identificateur d'une loi 2D isotrope viscoélastique de type Newton : $\boldsymbol{\sigma} = \mu \bar{\boldsymbol{D}}$, 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 : $\boldsymbol{\sigma} = \mu \left((\bar{\boldsymbol{D}} : \bar{\boldsymbol{D}}) + D_0 \right)^{n/2} \bar{\boldsymbol{D}}$ 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 contrainte avec le déviateur des déformations. La partie volumique est absente.

Pour les entrées des données, la syntaxe est identique au cas 1D, en remplaçant "1D" par "2D_D".

51.5.3 NEWTON3D

Identificateur d'une loi 3D isotrope viscoélastique de type Newton : $\boldsymbol{\sigma} = \mu \bar{\boldsymbol{D}}$. 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 : $\boldsymbol{\sigma} = \mu \left((\bar{\boldsymbol{D}} : \bar{\boldsymbol{D}}) + D_0 \right)^{n/2} \bar{\boldsymbol{D}}$

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 contrainte avec le déviateur des déformations. La partie volumique est absente.

Pour les entrées des données, la syntaxe est identique au cas 1D, en remplaçant "1D" par "3D".

51.5.4 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, ce qui conduit au temps caractéristique de relaxations $\tau = \mu/E$. 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 (161) donne un exemple de déclaration. Concernant le type de dérivée, nous avons :

- "type_derivee" = 1 − > dérivée deux fois contravariantes
- "type_derivee" = 0 − > dérivée deux fois covariante (valeur par défaut)
- "type_derivee" = -1 − > dérivée de Jauman

TABLE 161 – 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:
    E= 1.100000e+05     mu=      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 (162) donne un exemple de déclaration de coefficients thermo-dépendants.

Dans cet exemple toutes les fonctions 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 (163) donne un exemple de déclaration de coefficients thermo-dépendants au travers de courbes référencées.

Remarque À 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.

TABLE 162 – 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_  CPL1D DdlP 0. 100000. 100. 50000.0  FdlP
    mu=  mu_thermo_dependant_ CPL1D DdlP 0. 0.15 100. 0.05  FdlP
                fin_coeff_MAXWELL1D

#-------- fin cas d'une loi loi de maxwell thermodependante  --------
```

TABLE 163 – 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  --------
```

51.5.5 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 $\boldsymbol{II}_{\bar{\boldsymbol{D}}} = \boldsymbol{D} : \boldsymbol{D}$

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 cristalinité,

Si les fonctionnalités absentes sont recherchées, il est possible d'utiliser la loi de passage en contraintes planes (51.9) avec la loi 3D.

La table (164) donne un exemple de déclaration avec les commentaires inclus dans le Herezh.

TABLE 164 – 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 defaut),
# = 1 -> derivee deux fois contravariantes
# dans le cas ou l'on veut une valeur differente de la valeur par defaut il faut mettre le mot cle
# <type_deriveee> 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 thermodependence, il faut passer une ligne apres la descriptio
# 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
```

51.5.6 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 :

$$\boldsymbol{I}_\sigma = \frac{E}{(1-2\nu)}\boldsymbol{I}_\varepsilon \tag{75}$$

$\boldsymbol{I}_\sigma$ et $\boldsymbol{I}_\varepsilon$ étant les traces des tenseurs contraintes et déformations. Et dans le cas avec viscosité :

$$\boldsymbol{I}_D = \frac{1}{3K}\dot{\boldsymbol{I}}_\sigma + \frac{\boldsymbol{I}_\sigma}{\mu_p} \tag{76}$$

Avec $3K = E/(1-2\nu)$

Pour la partie déviatoire, on a :

$$\bar{\boldsymbol{D}} = \frac{1}{2G}\dot{\boldsymbol{S}} + \frac{\boldsymbol{S}}{\mu} \tag{77}$$

avec $\bar{\boldsymbol{D}}$ le déviateur du tenseur vitesse de déformation et $\dot{\boldsymbol{S}}$ 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. Ces coefficients conduisent au temps caractéristique de relaxations $\tau = \mu/E$. Trois types de dérivées matérielles sont implantées : Jauman (c'est-à-dire 1/2 de Lie en mixte dans les deux sens, valeur par défaut), de Lie deux fois covariante , et de Lie deux fois contravariante. 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 (166) donne un exemple de déclaration. Concernant le type de dérivée, nous avons :

- "type_derivee" = 1 − > dérivée deux fois contravariante (ou d'Oldroyd)
- "type_derivee" = 0 − > dérivée deux fois covariante (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 scission. Ceci s'effectue par la définition d'une fonction multiplicative $f(Q_{\bar{\boldsymbol{D}}})$ définie à la suite du mot clé " `fac_mu_cissionD=` ". La viscosité indiquée est alors multipliée par $f(Q_{\bar{\boldsymbol{D}}})$ calculée en fonction du taux de cisaillement en cours (cf.167) avec :

$$Q_{\bar{\boldsymbol{D}}} = \sqrt{\bar{\boldsymbol{D}} : \bar{\boldsymbol{D}}} \tag{78}$$

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 multiplié 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.165) 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.

TABLE 165 – 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
```

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. 166 et 168). 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 166 – 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 (162) et (163) 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 (168) donne un exemple d'entrée de données avec des viscosités thermo-dépendantes.

TABLE 167 – 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 defaut il faut mettre le mot cle
# <type_deriveee> 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 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
#   la derniere ligne doit contenir uniquement le mot cle:     fin_coeff_MAXWELL3D
```

TABLE 168 – Exemple de déclaration d'une loi de Maxwell en 3D avec les deux viscosités dépendantes d'une courbe de température "mu1dfT" définie 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
```

On introduit également la possibilité d'une viscosité dépendante de la cristalinité. La viscosité est obtenue à l'aide d'une modélisation proposée par Hieber, S.Han et K.K.

Wang. Nous avons :

$$\mu(\dot{\gamma}, T, p, \alpha) = \frac{\mu_0(T, p, \alpha)}{1 + \left(\mu_0 \frac{\dot{\gamma}}{\overset{*}{\tau}}\right)^{1-n}} \tag{79}$$

avec

$$\mu_0(T, p, \alpha) = \overset{*}{\mu}_0 (T, p) \ \exp(C_1 \alpha^2) \tag{80}$$

et

$$\overset{*}{\mu}_0 (T, p) = D_1 \exp\left(-\frac{A_1 \ (T - \overset{*}{T})}{A_2 + (T - \overset{*}{T})}\right) \text{ et } \overset{*}{T} = D_2 + D_3, \ A_2(p) = \overset{*}{A}_2 + D_3 \ p \tag{81}$$

La viscosité dépend ainsi de 8 paramètres matériels : n, $\overset{*}{\tau}$, D_1, D_2, D_3, A_1, $\overset{*}{A}_2$, $\overset{*}{C}_1$ On notera que le calcul de la viscosité dépend également de la température et de la pression.

TABLE 169 – Exemple de déclaration d'une loi de Maxwell en 3D avec une viscosité dépendante de la cristalinité

```
           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 (169) donne un exemple de définition de loi utilisant une viscosité dépendante de la cristalinité. 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 cristalinité, 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 cristalinité. Cette dernière peut-être accessible de deux manières. Par défaut elle est supposée connue au point d'intégration (c'est-à-dire au point où est calculée la loi de comportement). Dans ce cas, la cristalinité doit-être calculée au sein d'une loi thermo-physique (cf. 230) via un modèle de calcul de la cristalinité par exemple celui de Hoffman (cf.52.4). Une seconde possibilité est d'accéder à la cristalinité 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 cristalinité, via une lecture de champ de cristalinité par exemple.

51.6 Composition additive de différentes lois de base (ex : loi des mélanges).

Les compositions additives disponibles sont données dans la table (cf.170).

TABLE 170 – liste des différentes compositions additives disponibles de lois élémentaires

identificateur	indications	ref du commentaire
LOI_ADDITIVE_EN_SIGMA	1D, 2D, 3D : $\boldsymbol{\sigma} = \sum \boldsymbol{\sigma}_{(i)}$	(51.6.1)
LOI_DES_MELANGES_EN_SIGMA	1D, 2D, 3D : $\boldsymbol{\sigma} = (\alpha)\boldsymbol{\sigma}_{(1)} + (\alpha - 1)\boldsymbol{\sigma}_{(2)}$ ou $\Delta\boldsymbol{\sigma} = (\alpha)\Delta\boldsymbol{\sigma}_{(1)} + (\alpha - 1)\Delta\boldsymbol{\sigma}_{(2)}$	(51.6.2)

51.6.1 LOI_ADDITIVE_EN_SIGMA

Identificateur d'une loi additive en contrainte. Le principe est de déclarer une succession de 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. Ces informations sont sommées pour obtenir en finale la contrainte et le comportement tangent total. La table (171) donne un exemple de déclaration.

TABLE 171 – 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
```

Cas particuliers :

— Il est possible de ne retenir de la loi que les contraintes, ou que le comportement tangent. Ceci peut-être utile dans le cas d'un comportement tangent singulier par

exemple. Pour cela on indique après le nom de la loi, un des mots clés suivants : " sigma_seulement_ " ou " tangent_seulement_ ". Voici un exemple partiel de déclaration :

```
    ISOELAS  tangent_seulement_  # premiere loi isoelas 3D
                                 #dont on retient que le comportement tangent
           ....                  # déclaration de la loi élastique
    ISOELAS  tangent_seulement_  # seconde loi isoelas 3D
                                 #dont on retient que les contraintes
            ....                 # déclaration de la loi élastique
fin_liste_lois_elementaires   # ----- fin de loiAdditiveEnSigma
```

— Il est possible de pondérer chaque terme de la somme des contraintes. La contrainte finale sera alors :

$$\sigma = f_1 \times \sigma_1 + f_2 \times \sigma_2 + ...$$

Les facteurs de pondération sont des fonctions f_i de grandeurs disponibles pendant le calcul. Actuellement, les grandeurs disponibles sont :

- Un ddl quelconque de base, défini aux noeuds. Se référer à 80.11.4 ou 178, pour la liste exhaustive des ddl.
- La déformation cumulée équivalente : mot clé " def_equivalente "
- la déformation maximale rencontrée au sens de Mises : mot clé " def_duale_mises_maxi "
- la vitesse de déformation équivalente : mot clé " vitesse_def_equivalente "
- la déformation au sens de Mises : mot clé " def_duale_mises "
- la partie sphérique de la déformation : mot clé " Spherique_eps "
- la partie sphérique de la contrainte : mot clé " Spherique_sig "
- la proportion de cristallinité : mot clé " PROP_CRISTA "
- la valeur de la déformation "locale" EPS11 . Cette grandeur est intéressante uniquement par son signe, car sa valeur dépend de la taille de l'élément.

Pour ce faire, on indique sur la première ligne (après la ligne contenant le mot clé " LOI_ADDITIVE_EN_SIGMA) le mot clé : " avec_fonction_de_ponderation_ " , puis on passe à la ligne suivante. On définit alors successivement chaque loi de la manière suivante : avant le mot clé définissant la loi , on indique le mot clé " les_grandeur_ponderation= " suivi de m (m $<$ 11 maxi) couples (un nom de grandeur A_k suivi de l'une des deux chaines : " AuxNoeuds_ " ou " AuPti_ "), précisant où est défini la grandeur). L'ensemble des couples est terminé par le mot clé " fin_grandeur_ponderation_ ". On peut utiliser n'importe quelle grandeur définie (mais qui doit exister par ailleurs), aux noeuds ou aux points d'intégration, cf. documentation. Ensuite sur la ligne suivante on définit la fonction de pondération f, qui est le produit de fonctions 1D $g(A_k)$:

$$f = \Pi_{k=1}^{m}[g_k(A_k)]$$

On doit donc définir m fonctions 1D, d'où un mot clé : " deb_fonct_ponder= ", puis les noms des fonctions déjà définies, ou alors la définition directe de la fonction (après la définition directe d'une" fonction, on passe à la ligne suivante pour continuer) et enfin le mot clé : " fin_fonct_ponder_ " puis sur la ligne suivante : le nom de la loi. Optionnellement, après le nom de la loi, on peut indiquer (cf. explications précédentes) un des deux mots clé suivants : " sigma_seulement_ " ou " tangent_seulement_ " . Ensuite suivent les informations spécifiques à la loi. Sur la dernière ligne, on doit indiquer le mot clé : " fin_liste_lois_elementaires "

Exemple d'un somme pondéré de deux lois élastiques, chacune pondérée de fonctions dépendantes de la vitesse de déformation équivalente et de la température (noter que dans cet exemple il y a l'utilisation d'un caractère de continuation de ligne type Unix :
) :

```
metal  LOI_ADDITIVE_EN_SIGMA
   avec_fonction_de_ponderation_
    les_grandeur_ponderation= def_ equivalente  \
    AuPti_ TEMP AuxNoeuds_  fin_grandeur_ponderation_
    deb_fonct_ponder=   nom_fonc_1  nom_fonc_2  fin_fonct_ponder_
  ISOELAS
210000  0.3
   les_grandeur_ponderation= def_equivalente \
   AuPti_ TEMP  AuxNoeuds_ fin_grandeur_ponderation_
   deb_fonct_ponder=   nom_fonc_3  nom_fonc_4  fin_fonct_ponder_
 ISOELAS
300  0.1
```

Pondérations via des grandeurs spécifiques de certaines lois internes Il est possible de pondérer l'ensemble des fonctions à l'aide de certaines grandeurs disponibles uniquement en interne au niveau de certaines lois. C'est un cas spécifique et on se réfère à un exemple pour illustrer la méthode mise en place.

La table 172 donne un exemple d'utilisation. Dans cet exemple, il y a 4 lois, chacune pondérée par une fonction dépendante soit de la déformation équivalente de Mises (lois d'hystérésis) soit du maxi de cette déformation (lois hyperélastiques). Pour les lois d'hystérésis, est présente de plus une pondération qui dépend de 3 grandeurs qui ne sont disponibles (car calculées) "que" dans la loi d'hystérésis bulk " HYSTERESIS_BULK ".

1. PRESSION_HYST_T : qui correspond à la pression (= 1/3 de la trace de sigma) à l'instant t c'est-à-dire du début de l'incrément,
2. PRESSION_HYST_REF : qui correspond à la dernière pression de référence du modèle d'hystérésis bulk (cf.51.12.3). Cette grandeur est calculée dans la loi d'hystérésis bulk et est ensuite exportée au niveau de la loi additive. La grandeur utilisée au niveau de la loi additive, correspond à celle de l'instant initial = début de l'incrément.
3. PRESSION_HYST_REF_M : qui correspond à l'avant-dernière pression de référence du modèle d'hystérésis bulk (cf.51.12.3) : calcul et utilisation : idem PRESSION_HYST_REF .

La définition des grandeurs s'effectue entre les mots clés : " les_grandeurs_= " et " fin_grandeurs_ " . Ces deux mots clés doivent être précédés du mot clé " avec_ponder_grandeur_locale_ ". Ensuite l'utilisation de ces grandeurs s'effectue via une fonction nD définie par l'utilisation, ici de nom : " fct_dep_pression ". La table 173 donne la définition de la fonction nD qui se trouve dans le .info associé au calcul.

Le fonctionnement de la fonction additive est alors le suivant :

1. calcul des lois élémentaires : les deux lois élastiques et les deux lois d'hystérésis,
2. calcul des fonctions de pondérations classiques 1D
3. calcul de la fonction de pondération nD et multiplication du résultat avec celui des fonctions de pondérations 1D
4. calcul de la somme pondérée.

Actuellement, seules ces 3 pressions sont disponibles pour la fonction nD et le mécanisme de pondération fonctionne. L'intérêt de ce mécanisme est de pouvoir ajouter à la demande, de nouvelles grandeurs disponibles dans les lois internes.

Remarque :

1. Il est possible de définir qu'une seule fonction pondérée.
2. Les grandeurs de pondération, qui dépendent de la cinématique, sont calculées avant l'appel du calcul de la contrainte. Elles sont donc calculées au même moment que les contraintes. Par contre, les grandeurs de pondération qui dépendent elles-mêmes des contraintes sont calculées à partir des valeurs calculées au pas précédent, pour les calculs explicites, ou à l'itération précédente, pour les calculs implicites ou statiques. Ainsi, dans ce dernier cas, on peut certaine fois observé des ruptures de stabilité si les écarts sur un pas de temps ou sur une itération sont trop important. A priori, l'utilisation de grandeurs cinématique conduit à des calculs plus stables.

Opérateur tangent : L'opérateur tangent $\frac{\partial \sigma^{ij}}{\partial ddl}$ est par défaut, calculé directement dans chaque lois internes, puis l'opérateur final est obtenu en sommant l'ensemble des opérateurs élémentaires. Cette méthode est simple mais peut ne pas être la plus efficace. Une autre possibilité est de calculer l'opérateur tangent à l'aide de la décomposition suivante :

$$\frac{\partial \sigma^{ij}}{\partial ddl} = \frac{\partial \sigma^{ij}}{\partial \varepsilon_{kl}} \frac{\partial \varepsilon_{kl}}{\partial ddl} \tag{82}$$

Dans le cas d'une somme de "n" comportements élémentaires cela devient :

$$\frac{\partial \sigma^{ij}}{\partial ddl} = \sum_{e=1}^{n} \left(\frac{\partial \sigma^{ij}(e)}{\partial \varepsilon_{kl}} \right) \frac{\partial \varepsilon_{kl}}{\partial ddl} \tag{83}$$

Dans le cas où le nombre de ddl est plus important que le nombre de composantes de déformation (ce qui est très souvent le cas) cette dernière méthode est plus rapide. La table (174) donne un exemple d'utilisation. Le choix du type de calcul s'effectue à l'aide du mot clé : tangent_ddl_via_eps= . Par défaut celui-ci = 0

TABLE 172 – Exemple de déclaration d'une loi additive en contrainte

```
elastomere LOI_ADDITIVE_EN_SIGMA
avec_fonction_de_ponderation_
   #--------------------------------------
   # loi 1: hyperbulk gene
   #--------------------------------------
les_grandeur_ponderation= def_duale_mises_maxi  AuPti_  fin_grandeur_ponderation_
deb_fonct_ponder=   courbe_bulk    fin_fonct_ponder_
# .......  loi de comportement 3D hyperelastique isotrope ISOHYPERBULK_GENE
ISOHYPERBULK_GENE
#-------------------------"
#     K   |  fonction     |"
#-------------------------"
omega_V=   courbe_polynomiale sortie_post_ 1
   #--------------------------------------
   # loi 2: Orgeas sans partie sphérique
   #--------------------------------------
les_grandeur_ponderation= def_duale_mises_maxi  AuPti_  fin_grandeur_ponderation_
deb_fonct_ponder=   courbe_orgeas    fin_fonct_ponder_
ISOHYPER3DORGEAS1
0 10.0 0.1 1. 11.5 0.1 0.1 0.04 sortie_post_ 1
   #--------------------------------------
   # loi 3: Hystérésis déviatorique
   #--------------------------------------
les_grandeur_ponderation= def_duale_mises_maxi  AuPti_  fin_grandeur_ponderation_
deb_fonct_ponder=   courbe_hysteresis_3D    fin_fonct_ponder_
avec_ponder_grandeur_locale_
les_grandeurs_=  PRESSION_HYST_T  \
                            PRESSION_HYST_REF   \
                            PRESSION_HYST_REF_M1  \
fin_grandeurs_
deb_fonct_=   fct_dep_pression   fin_fonct_
HYSTERESIS_3D
np= 1.75       mu= 420    Qzero= 26   avec_parametres_de_reglage_
....
fin_parametres_reglage_Hysteresis_
#-------- fin hysteresis_3D déviatorique --------
   #--------------------------------------
   # loi 4: Hystérésis bulk
   #--------------------------------------
les_grandeur_ponderation= def_duale_mises_maxi  AuPti_  fin_grandeur_ponderation_
deb_fonct_ponder=   courbe_hysteresis_bulk    fin_fonct_ponder_
avec_ponder_grandeur_locale_
les_grandeurs_=  PRESSION_HYST_T  \
                            PRESSION_HYST_REF   \
                            PRESSION_HYST_REF_M1  \
fin_grandeurs_
deb_fonct_=   fct_dep_pression   fin_fonct_
HYSTERESIS_BULK #hystérésis sphérique
np= 1.75       mu= 1000    Qzero= 26   avec_parametres_de_reglage_
       ....
fin_parametres_reglage_Hysteresis_
#-------- fin hysteresis_bulk --------
fin_liste_lois_elementaires
```

TABLE 173 – Exemple de déclaration d'une fonction littérale qui dépend de 3 pressions

```
les_Fonctions_nD #------------
#  fonction Fonction_expression_litterale_nD
#                 f(P,PR,PRM1)
#  P   :  la pression,
#  PR  :  la dernière pression de référence
#  PRM1:  la pression de référence avant PR
fct_dep_pression FONCTION_EXPRESSION_LITTERALE_nD
un_argument= P un_argument= PR un_argument= PRM1

fct=  (P > 0.) ? \ # d'abord le cas positif
   ( ( (PR != 0.) && (P > PR) &&  (P <= (PR+PRM1)/2.) &&  ((PRM1 - PR) > 1.e-6) ) ?  \
       \ # première moitié de la remontée
      ( 0.4/(PR- (PR+PRM1)/2.) * P + (1. - PR*0.4/(PR- (PR+PRM1)/2.))) \
      : ( ( (PR != 0.) && (P > (PR+PRM1)/2.) && ((PRM1 - PR) > 1.e-6) ) ?  \
            \ #deuxième moitié de remontée
            ( 0.4/(PRM1 - (PR+PRM1)/2.) * P + (1. - 0.4 * PRM1/(PRM1 - (PR+PRM1)/2.))) \
            : 1. \  # tous les autres cas
        ) \
   )  \
   :  \  # sinon cas négatif
     ( ( (PR != 0.) && (P > PR) &&  (P <= (PR+PRM1)/2.) &&  ((PRM1 - PR) > 1.e-6) ) ?  \
       \ # première moitié de la remontée
        ( -0.4/(PR- (PR+PRM1)/2.) * P + (1. + PR*0.4/(PR- (PR+PRM1)/2.))) \
        : ( ( (PR != 0.) && (P > (PR+PRM1)/2.) && ((PRM1 - PR) > 1.e-6) ) ?  \
           \ #deuxième moitié de remontée
           ( -0.4/(PRM1 - (PR+PRM1)/2.) * P + (1. + 0.4 * PRM1/(PRM1 - (PR+PRM1)/2.))) \
           : 1. \  # tous les autres cas
          ) \
     )

  fin_parametres_fonction_expression_litterale_
```

TABLE 174 – Exemple de loi de comportement additive utilisant l'opérateur tangent par rapport à la déformation

```
# une loi pour tester l'opérateur tangent calculé à partir de dsig/deps
 acier_proj_3 LOI_ADDITIVE_EN_SIGMA
   tangent_ddl_via_eps= 1
   #----------
   PROJECTION_ANISOTROPE_3D
   # .......  loi de comportement Projection anisotrope 3D ........
    TYPE_PROJ_ANISO_  PROJ_ORTHO
    A1= 1. A2= 2 A3= 3. B12= 1.3 B13= 2.6 B23= 3.9
    nom_repere_associe_ repere1
            #-- loi interne
     ISOELAS
      1000  0.3
    fin_loi_interne # --- fin def loi interne"
    avec_parametres_de_reglage_
    sortie_post_       1
    type_transport_    0
    fin_parametres_reglage_
        permet_affichage_ 2
   fin_loi_projection_anisotrope # --- fin de la loi "
   #----------
fin_liste_lois_elementaires # ----- end of additive behaviors
```

51.6.2 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 :

$$\sigma_{final} = \alpha\sigma_1 + (1. - \alpha)\sigma_2 \tag{84}$$

Le même calcul est effectué pour la raideur.

La grandeur de proportion peut-être :

— soit une donnée fixe, au sens de grandeur imposée par l'utilisateur aux noeuds par exemple (ex : on impose une température à chaque noeud, mais cette température n'est pas un degré de liberté). La donnée est typée, cf. les noms de référence possibles. (voir 67.3 pour plus d'information)

— soit une ou plusieurs grandeur(s) variable(s) disponible(s), au sens de grandeur qui est calculée en fonction des résultats du calcul i.e. la valeur des ddls, des données etc.

— soit une ou plusieurs fonction (s) utilisateurs de grandeurs variables ou données.

Dans le cas d'une (ou plusieurs) grandeur(s) variable(s) disponible(s), elle peut être :

— définie à chaque noeud,

— directement accessible au point d'intégration (le point où la loi est calculée),

— le temps,

— une grandeur globale de type : "énergie, puissance, précision de l'équilibre globale, numéro d'incrément, numéro d'itération, etc. ", on se reportera à cf.88.1 pour une liste exhaustive,

— un mixte des précédentes grandeurs.

D'une manière générale, on se reportera à 89 pour une description exhaustive des pondérations possibles.

Quelques exemples de grandeurs locales :

— un identificateur d'un ddl défini au noeud, par exemple la température `TEMP` (cf. 178),

— " `PROP_CRISTA` " Le mot clé pour le choix entre grandeurs aux noeuds ou au point d'intégration est " `valeur_aux_noeuds=` " et la table (177) donne un exemple de déclaration.

— " `def_equivalente` ", " `def_duale_mises` ", " `Spherique_eps` " et " `Spherique_sig` " : le fonctionnement est identique au cas précédant. La table 179 montre un exemple (complexe) de loi de mélange pilotée par la pression, et faisant appelle à deux lois additives.

La liste cf.80.11.4 donne la liste des ddl possibles aux noeuds. Bien noter que l'existence d'un ddl dépend du problème traité.

D'une manière générale, on se reportera à 89 pour une description exhaustive des pondérations possibles.

La table (175) donne un exemple de déclaration. Dans cet 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 (178) donne un exemple (plus restreint) des grandeurs potentiellement disponibles 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 :

$$\Delta\sigma_{final} = \alpha\Delta\sigma_1 + (1. - \alpha)\Delta\sigma_2 \; et \; \sigma_{final}^{t+\Delta t} = \sigma_1^t + \sigma_2^t + \Delta\sigma_{final} \tag{85}$$

Dans ce dernier cas, la contrainte finale cumule ainsi toute l'histoire du facteur de pondération, ce qui peut être dans certains cas intéressants. 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 de mélange vaut 1 ce qui correspond au premier fonctionnement. La table (176) donne un exemple de fonctionnement.

La valeur finale de la contrainte peut-être multipliée par un coefficient "A" :

$$\sigma_{final} = A(\alpha\sigma_1 + (1. - \alpha)\sigma_2) \quad \text{ou} \quad \Delta\sigma_{final} = A(\alpha\Delta\sigma_1 + (1. - \alpha)\Delta\sigma_2) \tag{86}$$

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 179).

Il est donc possible également d'utiliser une fonction pour calculer α à partir d'une grandeur de pondération (par exemple on peut vouloir une dépendance à la température de α, c'est-à-dire une fonction $\alpha(TEMP)$). L'objet du paragraphe qui suit est de détailler la mise en données qui peut-être complexe.

Mise en donnée avec le cas particulier des pondérations

En fait de manière pratique, deux mises en données sont disponibles pour définir la loi avec les pondérations. La première est historique, elle n'évoluera pas dans les nouvelles versions et elle offre moins de possibilités par rapport à la seconde. Cependant elle est conservée pour garantir la compatibilité des nouvelles versions. La seconde (à partir de la version 6.784) offre de nombreuses possibilités. Elle est en phase de mise au point et susceptible de nouvelles évolutions...

1. **Première mise en données (historique)** : Sur la première ligne, on indique successivement :

(a) Obligatoirement : le type de grandeur (mot clé : " grandeur_proportion= ") qui règle la proportion entre les deux lois. On peut utiliser n'importe quelle grandeur définie (mais qui doit exister par ailleurs) aux noeuds.

(b) Optionnellement : le type de mélange (mot clé : " type_melange= ") (1 ou 2) cf. les explications données précédemment.

(c) Optionnellement : si la proportion provient des noeuds ou du point d'intégration de manière directe : (mot clé : " valeur_aux_noeuds= ") (1 ou 0) la valeur par défaut est 1 (en faite tout nombre différent de 0). Dans le cas de 0, les cas implantés sont relatifs à " PROP_CRISTA " et la liste indiquée dans cf.82.2 ". Exemple :

```
grandeur_proportion=   PROP_CRISTA  type_melange= 2 valeur_aux_noeuds= 0
```

(d) Optionnellement : l'extension à une somme pondérée étendue
(mot clé : " somme_pondere_etendue_ ") si oui, on passe une ligne et sur la ligne suivante on a le mot clé : " A= " suivi du coefficient multiplicatif de la somme pondérée (cf. l'explication déjà donnée précédemment).

(e) Optionnellement : une fonction permettant le calcul de la proportion en fonction de la grandeur qui règle la proportion
(mot clé : " avec_fonction_proportion_ "). Dans ce cas, on passe une ligne et la fonction est décrite avec le mot clé : " fonction_prop= " suivi du nom de la fonction (si elle existe déjà) ou de la définition de la fonction. Dans ce dernier cas, ne pas oublier de passer à la ligne après la description de la fonction (comme pour toutes les lois). La table 179 donne un exemple de fonction.

(f) Puis on passe une ligne pour décrire les 2 lois individuellement.

2. **Deuxième mise en données** Il est préférable d'utiliser cette nouvelle mise en données pour de nouveaux travaux. Elle offre un plus large choix de fonctions de pondération, par contre son utilisation est sans doute un peu plus complexe.

   La pondération finale peut-être un assemblage multiplicatif de :

   — une fonction nD de grandeurs globales (ex : précision d'équilibre globale, numéro d'itération, etc.). On se réfèrera à 82.4 pour une description précise.

   — un produit d'un ensemble de courbes 1D, chacune fonction d'un ddl étendu. On se réfèrera à 82.2 pour une description précise.

   — une courbe 1D fonction du temps. On se réfèrera à 82.1 pour une description précise.

   Sur la première ligne, on indique successivement :

   (a) le mot clé : " les_grandeurs_de_controle_= ",

   (b) ensuite on donne la liste des grandeurs que l'on va utiliser, rappelons que ces grandeurs peuvent être :

   — le temps (mot clé : "TEMPS"),

   — un ddl primaire (ex : U1 ou X1 au noeud) ou secondaire (SIG11 ou EPS11 au pti)

— un ddl étendu,

— une grandeur globale.

l'ensemble des grandeurs doit se terminer par le mot clé : " fin_grandeurs_ "

(c) ensuite sur la ligne suivante : on donne la liste des fonctions que l'on va utiliser : mot clé : " deb_fonct_= "
la liste doit suivre le même ordre que celui des grandeurs, cependant :

— dans le cas où on a indiqué au moins une grandeur globale autre que le temps ("TEMPS") , il y a une seule fonction pour toutes les grandeurs, c'est donc la place de la première grandeur globale qui est considérée. Les grandeurs globales réellement utilisées sont celles définies dans la fonction. Le fait de définir ici une grandeur globale, a pour seul objectif de déclencher la lecture d'une fonction nD. La grandeur globale indiquée doit appartenir à l'ensemble des grandeurs globales que la fonction gère. Par contre il n'est pas nécessaire de donner la liste exhaustive de toutes les grandeurs que la fonction nD gère, il suffit d'en donner une seule. En particulier, la fonction nD peut également dépendre du temps !

— pour les ddl et ddl étendues il faut indiquer après chaque fonction d'un ddl s'il s'agit :

— d'une grandeur au point d'intégration avec le mot clé " AuPti_ "

— d'une grandeur au noeud avec le mot clé " AuxNoeuds_ "

l'ensemble de la liste des fonctions doit se terminer par le mot clé : " fin_fonct_ "

(d) puis sur la ligne suivante, optionnellement : le type de mélange (mot clé : type_melange=) (1 ou 2) l'explication est identique au cas de la première mise en données

(e) puis on passe une ligne pour décrire les 2 lois individuellement.

La table 180 donne un exemple de cette nouvelle mise en données.

Bien noter que c'est le premier mot clé :
grandeur_proportion= ou les_grandeurs_de_controle_=
qui indique au programme le choix entre première et deuxième mises en données.

Suite de la mise en donnée concernant les deux lois internes.

À la suite du mot clé qui définit une loi interne, on peut choisir entre 4 types de fonctionnements. Pour simplifier et expliciter la présentation, on suppose que la première loi est une simple loi de Hooke. Les 4 types de fonctionnements sont alors les suivants :

1. ISOELAS → aucune particularité,
2. ISOELAS calcule_si_prop_non_nulle_ → le mot clé calcule_si_prop_non_nulle_ est utilisé pour indiquer que la loi ne doit-être calculée que si la proportion la concernant (c.-à-d. α pour la première loi, et $(1.-\alpha)$ pour la seconde loi) est non nulle. Si elle est nulle, la loi n'est pas calculée. Ce fonctionnement est intéressant, lorsque l'on veut par exemple augmenter la rapidité du calcul en évitant une opération inutile qu'est le calcul de la loi multiplié par 0.

3. ISOELAS demarre_a_prop_non_nulle_puis_strictement_décroissante_ → le fonctionnement est semblable au cas précédent avec en plus la particularité supplémentaire suivante : on interdit à la proportion d'augmenter. Ainsi, si entre deux appels consécutifs de la loi de comportement, la valeur de α tente d'augmenter au travers d'un calcul via une fonction de pondération, c'est la valeur la plus faible qui est retenue, et ceci tout au long du calcul. Ce comportement très particulier est construit en particulier pour être utilisé via une fonction de pondération qui dépend de la précision de l'équilibre globale. On force ainsi l'évolution d'α à être strictement décroissante à mesure que la précision augmente.
4. ISOELAS demarre_a_prop_non_nulle_puis_strictement_decroissante_sur_chaque_incre_ → le fonctionnement est un peu semblable au cas précédent, mais avec la particularité supplémentaire suivante : l'évolution est strictement décroissante sur un incrément. Par contre au début de chaque incrément, on initialise la valeur d'α via les fonctions de proportion.

La table 180 donne un exemple d'utilisation d'un de ces types de fonctionnements.

Bien noter que dans le cas où une loi n'est pas calculée, il n'y a pas accumulation de l'historique du comportement. Aussi c'est une option naturellement raisonnable pour une loi non incrémentale, dans le cas contraire, il faut avoir conscience de ce que l'on impose. Il peut-être certaine fois intéressant de ne pas accumuler l'historique... mais ce n'est pas courant.

TABLE 175 – Exemple de déclaration d'une loi des mélanges 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 entreles 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   # ----- fin de LoiDesMelangesEnSigma
```

TABLE 176 – Exemple de déclaration d'une loi des mélanges 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 177 – Exemple de déclaration d'une loi additive en contrainte, dans le cas où 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 178 – Liste des différentes grandeurs susceptibles d'être disponibles 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 cristalinité
   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 179 – 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 (expension) -----
  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
```

TABLE 180 – Exemple de loi des mélanges pilotée par la norme de convergence : cas de la nouvelle mise en données

```
toiles LOI_DES_MELANGES_EN_SIGMA

 les_grandeurs_de_controle_=  norme_de_convergence\
                 fin_grandeurs_

   deb_fonct_= fonction3   fin_fonct_

   # première loi élastique
   ISOELAS2D_C demarre_a_prop_non_nulle_puis_strictement_decroissante_sur_chaque_incre_
   #    module d'young :   coefficient de Poisson
   1.8e+09   0.3

   # seconde loi élastique beaucoup plus souple
   ISOELAS2D_C demarre_a_prop_non_nulle_puis_strictement_decroissante_sur_chaque_incre_
   #    module d'young :   coefficient de Poisson
   3e+06   0.3

fin_liste_lois_elementaires   # ----- fin de LoiDesMelangesEnSigma
```

51.7 Passage en déformations et contraintes planes pour une loi 3D quelconque : introduction

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 veiller à utiliser des conditions limites correctes sur les 3 dimensions (par exemple UZ).

51.8 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 raideur, internes.

En post-traitement, seules les coordonnées d'ordre 2 du tenseur de contrainte sont directement accessibles 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 choisi 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 181 donne un exemple de loi de déformation plane associée à une loi 3D additive en sigma.

51.9 LOI_CONTRAINTES_PLANES

Ce type de loi est particulièrement adapté aux membranes, plaques et coques. On considère que la direction normale à la surface est 3. Ainsi la condition s'énonce sous la forme d'une contrainte mathématique : $\sigma^{33} = 0$. On considère également que la direction

TABLE 181 – 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
```

3 est normée, la variance selon la direction 3 est donc sans importance d'où la condition finale :

$$\sigma^{33} = \sigma_{33} = \sigma_3^{.3} = \sigma_{.3}^3 = 0 \tag{87}$$

La membrane (plaque ou coque) est décrite via une interpolation dans le plan de la membrane. Les déformations naturellement disponibles via la cinématique, sont donc les déformations dans ce plan que l'on notera $\varepsilon_{\alpha\beta}$ avec α, β= 1, 2 . L'épaisseur constitue une inconnue supplémentaire du problème. Cette épaisseur est reliée avec la déformation ε_{33} de telle manière à satisfaire la relation 87 via par exemple $\sigma^{33}(\varepsilon_{33}) = 0$.

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.

51.9.1 Méthode par perturbation (explicite)

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 nouvelle épaisseur est prise 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. Cet 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 bases, 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'élément de manière classique (via les coordonnées entraînées).

— Le vecteur 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 raideur, internes.

En post-traitement, seules les coordonnées d'ordre 2 du tenseur des déformations sont directement accessibles 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 choisi pour la loi (nom donné par l'utilisateur) suivi du mot clé : " `LOI_CONTRAINTES_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 182 donne un exemple de loi avec condition de contrainte plane associée à une loi 3D additive en sigma.

51.9.2 Méthode de Newton (implicite)

L'idée est ici d'intégrer de manière implicite l'équation non linéaire qui correspond à la condition de contrainte plane. L'algorithme est alors itératif, mais dans la pratique la convergence observée est très rapide : 1 à 3 itérations en moyenne.

L'équation $\sigma^{33}(\varepsilon_{33}) = 0$ est résolue par une méthode de Newton ce qui conduit à déterminer la déformation d'épaisseur nécessaire pour la condition de contrainte plane. À la fin du processus, après convergence, on dispose :

— d'un champ de contraintes 3D qui satisfait la condition de contrainte plane

TABLE 182 – Exemple de loi de contraintes planes intégrant un comportement 3D et des conditions de contraintes planes. Dans cet exemple, la prise en compte de la condition de contraintes planes s'effectue par perturbation.

```
#-----------------------------------------
# Nom Materiau    |      Type loi         |
#-----------------------------------------
PPC    LOI_CONTRAINTES_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
```

— de la déformation d'épaisseur et de l'épaisseur correspondante.

L'algorithme intègre des paramètres de réglage :

— le nombre d'itérations maximum permis : exemple pour un nombre limité à 20 : `nb_iteration_maxi_ 20`

— le nombre de dichotomies maximum permis : exemple pour un nombre limité à 20 : `nb_dichotomie_maxi_ 20`. Dans le cas où la convergence n'a pas fonctionné avec " `nb_iteration_maxi_` " itérations, l'intervalle de déformation est divisé par 2 (première dichotomie), et l'algorithme est appliqué sur chaque demi-intervalle. Si non-convergence, le processus est répété. Le nombre maximum de dichotomies est limité par " `nb_dichotomie_maxi_` ".

— la tolérance absolue sur la condition $\sigma^{33} = 0$. Exemple : " `tolerance_residu_ 5.e-3`" signifie que la convergence est atteinte lorsque $|\sigma^{33}| < 5.e-3$. Il est également possible de piloter la tolérance à l'aide d'une fonction nD. Ainsi par exemple il est possible de relâcher la précision au début de la convergence générale et ensuite de tendre vers une tolérance voulue en fin de convergence globale. Pour cela on utilise la syntaxe suivante :
`tolerance_residu_ =fonction_nD:` suivi du nom d'une fonction nD préalablement définie.

— la tolérance relative sur la condition $\sigma^{33} = 0$. Exemple : " `tolerance_residu_rel_ 1.e-4`" signifie que la convergence est atteinte lorsque $|\sigma^{33}| < 1.e-4 \times (max(|\sigma^{\alpha\beta}|)$ avec $\alpha, \beta = 1, 2$. Comme pour la tolérance absolue, il est également possible de

piloter la tolérance à l'aide d'une fonction nD avec la syntaxe suivante :
`tolerance_residu_rel_ =fonction_nD:` suivi du nom d'une fonction nD préalablement définie.

- `mini_hsurh0_` suivi d'un réel = le minimum acceptable pour la variation suivant l'épaisseur : valeur par défaut = 0.001,
- `maxi_hsurh0_` suivi d'un réel = le maximum acceptable pour la variation suivant l'épaisseur : valeur par défaut = 1000,
- " `maxi_delta_var_eps_sur_iter_pour_Newton_` " : utilisé avec la méthode de Newton. Indique le maximum admissible sur $|\delta\varepsilon_{33}|$ à chaque itération de Newton. Lorsque l'algorithme calcule un nouvel incrément qui dépasse ce maximum, l'incrément est diminué suivant la formule : $\delta\varepsilon_{33,nouveau} = \delta\varepsilon_{33} \ . \ \delta\varepsilon_{max}/|\delta\varepsilon_{33}|$ où $\delta\varepsilon_{max}$ est la norme maxi que l'on a imposée.

Comme pour la méthode par perturbation, en post-traitement, seules les coordonnées d'ordre 2 du tenseur des déformations sont directement accessibles 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`).

Il est également possible de récupérer :

- le nombre d'incréments total (fonction du nombre de dichotomies) utilisé pour la convergence : mot clé proposé en interactif " `NB_INCRE_TOTAL_RESIDU` "
- le nombre d'itérations total (somme de toutes les itérations pour chaque incrément) associé : mot clé proposé en interactif " `NB_ITER_TOTAL_RESIDU` "

Ces grandeurs ne sont pas stockées normalement, il faut donc indiquer que l'on veut les stocker, avant de demander une récupération. Pour cela on indique le mot clé " `sortie_post_` " suivi de 1 , dans la liste des paramètres de réglage de l'algorithme.

Enfin, l'affichage des commentaires et erreurs éventuelles dépend par défaut du paramètre global " `niveau_commentaire` ". Il est possible d'indiquer un autre niveau localement (par exemple plus élevé) à l'aide du mot clé " `permet_affichage_` " suivi du niveau voulu localement.

La table 183 donne un exemple de loi avec condition de contrainte plane associée à une loi 3D additive en sigma avec une méthode de résolution de type Newton. On remarquera que l'ensemble des paramètres de réglage est encapsulé entre les mots clés : " `avec_parametres_de_reglage_` " et " `fin_parametres_reglage_Algo_Newton_` "

TABLE 183 – Exemple de loi de contraintes planes intégrant un comportement 3D et des conditions de contraintes planes. Dans cet exemple, la prise en compte de la condition de contraintes planes s'effectue par une méthode de Newton.

```
#-----------------------------------------
# Nom Materiau    |      Type loi         |
#-----------------------------------------
PPC    LOI_CONTRAINTES_PLANES
    NEWTON_LOCAL avec_parametres_de_reglage_
        nb_iteration_maxi_  20
        nb_dichotomie_maxi_  5
        tolerance_residu_ 1.e-4
        tolerance_residu_rel_  1.e-3
        sortie_post_      1
        permet_affichage_   5
   fin_parametres_reglage_Algo_Newton_

  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
```

51.10 LOI_CONTRAINTES_PLANES_DOUBLE

Ce type de loi est particulièrement adapté aux barres et poutres. On considère que les directions normales à la ligne de référence sont 2 et 3. Ainsi la condition s'énonce sous la forme d'une contrainte mathématique : $\sigma^{22} = 0$ et $\sigma^{33} = 0$. On considère également que les directions 2 et 3 sont normées, la variance selon les directions 2 et 3 est donc sans importance d'où la condition finale :

$$\sigma^{22} = \sigma_{22} = \sigma_2^2 = \sigma_{.2}^2 = 0 \quad \text{et} \quad \sigma^{33} = \sigma_{33} = \sigma_3^3 = \sigma_{.3}^3 = 0 \tag{88}$$

La barre ou poutre est décrite via une interpolation suivant la ligne de référence. Les déformations naturellement disponibles via la cinématique, sont donc la déformation le long de cette ligne que l'on notera ε_{11}. La largeur et l'épaisseur (directions 2 et 3) constituent deux inconnues supplémentaires du problème. Ces grandeurs sont reliées avec les déformations ε_{22} et ε_{33} de telle manière à satisfaire la relation 88 via par exemple $\sigma^{22} = 0$ et $\sigma^{33} = 0$.

Globalement, la technique de résolution suit la même méthodologie que pour les contraintes planes. En particulier on retrouve les mêmes algorithmes avec une implémentation très similaire.

51.10.1 Méthode par perturbation (explicite)

Comme pour la méthode de contrainte plane, l'idée de cette première méthode considérée est relativement triviale. Par exemple pour un pas de temps "t" à "$(t + \Delta t)$" la méthode est la suivante :

1. Calcul de la contrainte à $t + \Delta t$ à l'aide de la loi de comportement : $\sigma^{11} = f(\varepsilon_{11}(t + \Delta t), \varepsilon_{22}(t), \varepsilon_{33}(t))$
2. Comme on doit avoir $\sigma_2^2 = 0$ et $\sigma_3^3 = 0$, calcul de la trace prise en compte pour le calcul d'équilibre à savoir : σ_1^1,
3. Calcul de la variation de volume via la compressibilité, d'où les variations d'épaisseur et de largeur qui sont censées correspondre aux conditions $\sigma_2^2 = 0$ et $\sigma_3^3 = 0$. Pour un matériau isotrope, les deux variations relatives de dimensions sont supposées égales.
4. Mise à jour de l'épaisseur $h(t + \Delta t)$, de la déformation d'épaisseur : $\varepsilon_{33}(t + \Delta t)$, de la largeur $b(t + \Delta t)$, de la déformation de largeur : $\varepsilon_{22}(t + \Delta t)$. Pour un matériau isotrope, les deux déformations sont supposées égales.
5. Calcul de l'équilibre en utilisant $\sigma^{11}(t+\Delta t)$ précédemment calculées et des nouvelles épaisseur $h(t + \Delta t)$ et largeur $b(t + \Delta t)$.

En post-traitement, seules les coordonnées d'ordre 1 du tenseur des déformations sont directement accessibles via les grandeurs classiques. Cependant la déformation d'épaisseur et celle de largeur peuvent être récupérées via les grandeurs particulières associées à la loi de comportement (mots clés proposés en interactif : DEF_EPAISSEUR , DEF_LARGEUR).

Au niveau de l'utilisation, la méthodologie est simple.

— sur une première ligne, on indique le nom choisi pour la loi (nom donné par l'utilisateur) suivi du mot clé : " LOI_CONTRAINTES_PLANES_DOUBLE "

— puis sur la ligne qui suit, on indique le type de technique utilisée pour tenir compte de la condition de contraintes planes double,

— 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_double` "

La table 182 donne un exemple de loi avec condition de contrainte plane double associée à une loi 3D additive en sigma, résolue avec une méthode de perturbation.

TABLE 184 – Exemple de loi de contraintes planes intégrant un comportement 3D et des conditions de contraintes planes doubles. Dans cet exemple, la prise en compte de la condition de contraintes planes s'effectue par perturbation.

```
#----------------------------------------
# Nom Materiau    |      Type loi         |
#----------------------------------------
essai_traction_evolutive_CP LOI_CONTRAINTES_PLANES_DOUBLE
PERTURBATION  deformation_transversale

LOI_ADDITIVE_EN_SIGMA #

# ========================  hyperelastic part ==================================

 ISOHYPER3DORGEAS1
# .......  behavior 3D hyperelastic orgeas 1   ........
#-------------------------------------------------------------------------------------------
#     K      |     Qs     |   mu1     |   mu2  | mu3  | alpha1 |  alpha2 |    Qe    |
#-------------------------------------------------------------------------------------------
     270000         245        12000       200   10000    0.0005     0.005      0.064 avec_phase
    nQs= 0.9 gammaQs= 0.2  nQe= 0.9 gammaQe= -0.2  nMu1= 0.8 gammaMu1= 0.2 \
    avec_regularisation_ 0.001 sortie_post_ 1 # nMu2= 0.8 gammaMu2= 0.1
# ========================  hysteretis part ===================================
        HYSTERESIS_3D
# #-------------------------------------------------------------------
# .......  loi_de_comportement d'hysteresis 3D ........            |
#    para de prager    :         mu         : limite de plasticite |
# #-------------------------------------------------------------------
     np= 2    mu= 9000   Qzero= 100 avec_parametres_de_reglage_
#
#        avec_parametres_de_reglage_
#       ------------------------------------
#
#  #-- controle parameters
#
    type_de_resolution_  5  type_calcul_comportement_tangent_ 2
     cas_kutta_  5  erreurAbsolue_  1.e-4  erreurRelative_ 1.e-5
     nbMaxiAppel_ 1600

     tolerance_coincidence_  1.e-4
     nb_maxInvCoinSurUnPas_  -20
     mini_Qsig_pour_phase_sigma_Oi_tdt_ 10 #10
     mini_rayon_  1.e-12  min_angle_trajet_neutre_ 1.e-4

    fin_parametres_reglage_Hysteresis_

fin_liste_lois_elementaires # ----- end of additive behaviors

fin_loi_contrainte_plane_double
```

51.10.2 Méthode de Newton (implicite)

Comme pour la condition de contrainte plane, l'idée est ici d'intégrer de manière implicite l'équation non linéaire qui correspond à la condition de contrainte plane dans deux sens.

L'équation $< \sigma^{22}, \sigma^{33} > (\varepsilon_{22}, \varepsilon_{33}) =< 0, 0 >$ est résolue par une méthode de Newton ce qui conduit à déterminer les déformations d'épaisseur et de largeur nécessaires pour la condition de contrainte plane double. À la fin du processus, après convergence, on dispose :

- d'un champ de contraintes 3D qui satisfait la condition de contrainte plane double
- de la déformation d'épaisseur et de l'épaisseur correspondante,
- de la déformation de largeur et de la largeur correspondante.

L'algorithme intègre des paramètres de réglage, les mêmes que ceux de l'algorithme de contrainte plane :

- le nombre d'itérations maximum permis : exemple pour un nombre limité à 20 : `nb_iteration_maxi_ 20`
- le nombre de dichotomies maximum permis : exemple pour un nombre limité à 20 : `nb_dichotomie_maxi_ 20`. Dans le cas où la convergence n'a pas fonctionné avec " `nb_iteration_maxi_` " itérations, l'intervalle de déformation est divisé par 2 (première dichotomie), et l'algorithme est appliqué sur chaque demi-intervalle. Si non-convergence, le processus est répété. Le nombre maximum de dichotomies est limité par " `nb_dichotomie_maxi_` ".
- la tolérance absolue sur la condition $< \sigma^{22}, \sigma^{33} >=< 0, 0 >$. Exemple : " `tolerance_residu_ 5.e-3`" signifie que la convergence est atteinte lorsque $|| < \sigma^{22}, \sigma^{33} > || < 5.e-3$. Comme pour les contraintes planes, il est également possible de piloter la tolérance à l'aide d'une fonction nD. Ainsi par exemple il est possible de relâcher la précision au début de la convergence générale et ensuite de tendre vers une tolérance voulue en fin de convergence globale. Pour cela on utilise la syntaxe suivante : `tolerance_residu_ =fonction_nD:` suivi du nom d'une fonction nD préalablement définie. Comme pour la tolérance absolue, il est également possible de piloter la tolérance à l'aide d'une fonction nD avec la syntaxe suivante : `tolerance_residu_rel_ =fonction_nD:` suivi du nom d'une fonction nD préalablement définie.
- la tolérance relative sur la condition $< \sigma^{22}, \sigma^{33} >=< 0, 0 >$. Exemple : " `tolerance_residu_rel_ 1.e-4`" signifie que la convergence est atteinte lorsque $|| < \sigma^{22}, \sigma^{33} > || < 1.e-4 \times |\sigma^{11}|$
- `mini_hsurh0_` suivi d'un réel = le minimum acceptable pour la variation suivant l'épaisseur : valeur par défaut = 0.001,
- `maxi_hsurh0_` suivi d'un réel = le maximum acceptable pour la variation suivant l'épaisseur : valeur par défaut = 1000,
- `mini_bsurb0_` suivi d'un réel = le minimum acceptable pour la variation suivant la largeur : valeur par défaut = 0.001,
- `maxi_bsurb0_` suivi d'un réel = le maximum acceptable pour la variation suivant la largeur : valeur par défaut = 1000,

— calcul_en_3D_via_direction_quelconque_ Ce paramètre permet d'utiliser une nouvelle méthode pour laquelle l'axe de traction est différent du repère matériel de calcul. Cette option est surtout utile dans le cas de l'utilisation conjointe avec la loi plis. Par défaut ce paramètre n'est pas actif, et n'a pas d'intérêt pour une utilisation classique de loi 1D.

— " maxi_delta_var_eps_sur_iter_pour_Newton_ " : utilisé avec la méthode de Newton. Indique le maximum admissible sur $||\delta\varepsilon_{ii}||$ à chaque itération de Newton. Ici il s'agit de la norme du vecteur composé des deux composantes ε_{22} et ε_{33}. Lorsque l'algorithme calcule un nouvel incrément qui dépasse ce maximum, l'incrément est diminué suivant la formule :

$$< \delta\varepsilon_{22,nouveau}, \delta\varepsilon_{33,nouveau} >=< \delta\varepsilon_{22}, \delta\varepsilon_{33} > \ . \ \delta\varepsilon_{max}/||\delta\varepsilon_{ii}||$$

avec i=2,3 et où $\delta\varepsilon_{max}$ est la norme maxi que l'on a imposée.

Comme pour la méthode par perturbation, en post-traitement, seules les coordonnées d'ordre 1 du tenseur des déformations sont directement accessibles via les grandeurs classiques. Cependant la déformation d'épaisseur et la déformation de largeur peuvent être récupérées via les grandeurs particulières associées à la loi de comportement (mot clé proposé en interactif : DEF_EPAISSEUR et DEF_LARGEUR).

Il est également possible de récupérer :

— le nombre d'incréments total (fonction du nombre de dichotomies) utilisé pour la convergence : mot clé proposé en interactif " NB_INCRE_TOTAL_RESIDU "

— le nombre d'itérations total (somme de toutes les itérations pour chaque incrément) associé : mot clé proposé en interactif " NB_ITER_TOTAL_RESIDU "

Ces grandeurs ne sont pas stockées normalement, il faut donc indiquer que l'on veut les stocker, avant de demander une récupération. Pour cela on indique le mot clé " sortie_post_ " suivi de 1 , dans la liste des paramètres de réglage de l'algorithme.

Enfin, l'affichage des commentaires et erreurs éventuelles dépend par défaut du paramètre global " niveau_commentaire ". Il est possible d'indiquer un autre niveau localement (par exemple plus élevé) à l'aide du mot clé " permet_affichage_ " suivi du niveau voulu localement.

La table 185 donne un exemple de loi avec condition de contrainte plane associée à une loi 3D additive en sigma avec une méthode de résolution de type Newton. On remarquera que l'ensemble des paramètres de réglage est encapsulé entre les mots clés : " avec_parametres_de_reglage_ " et " fin_parametres_reglage_Algo_Newton_ "

TABLE 185 – Exemple de loi de contraintes planes intégrant un comportement 3D et des conditions de contraintes planes doubles. Dans cet exemple, la prise en compte de la condition de contraintes planes double s'effectue avec la méthode de Newton.

```
#----------------------------------------
# Nom Materiau    |      Type loi         |
#----------------------------------------
essai_traction_evolutive_CP LOI_CONTRAINTES_PLANES_DOUBLE
    NEWTON_LOCAL avec_parametres_de_reglage_
        nb_iteration_maxi_  20
        nb_dichotomie_maxi_  5
        tolerance_residu_ 1.e-4
        tolerance_residu_rel_  1.e-3
        sortie_post_      1
        permet_affichage_   5
    fin_parametres_reglage_Algo_Newton_

LOI_ADDITIVE_EN_SIGMA #

# ========================  hyperelastic part ==================================

 ISOHYPER3DORGEAS1
# .......  behavior 3D hyperelastic orgeas 1   ........
#-------------------------------------------------------------------------------------
#     K      |     Qs     |   mu1     |   mu2  | mu3  | alpha1 |  alpha2 |    Qe   |
#-------------------------------------------------------------------------------------
     270000         245        12000       200   10000    0.0005     0.005      0.064 avec_phase
    nQs= 0.9 gammaQs= 0.2  nQe= 0.9 gammaQe= -0.2  nMu1= 0.8 gammaMu1= 0.2 \
    avec_regularisation_ 0.001 sortie_post_ 1 # nMu2= 0.8 gammaMu2= 0.1
# ========================  hysteretis part ==================================
        HYSTERESIS_3D
# ---------------------------------------------------------------------
# .......  loi_de_comportement d'hysteresis 3D ........            |
#    para de prager    :         mu         : limite de plasticite |
# ---------------------------------------------------------------------
     np= 2    mu= 9000   Qzero= 100 avec_parametres_de_reglage_
# -- controle parameters
#
    type_de_resolution_  5  type_calcul_comportement_tangent_ 2
     cas_kutta_  5  erreurAbsolue_  1.e-4  erreurRelative_ 1.e-5
     nbMaxiAppel_ 1600

     tolerance_coincidence_  1.e-4
     nb_maxInvCoinSurUnPas_  -20
     mini_Qsig_pour_phase_sigma_Oi_tdt_ 10 #10
     mini_rayon_  1.e-12  min_angle_trajet_neutre_ 1.e-4

    fin_parametres_reglage_Hysteresis_

fin_liste_lois_elementaires # ----- end of additive behaviors

fin_loi_contrainte_plane_double
```

51.11 Prise en compte de critère sur une loi de comportement

On entend par critère, un comportement nouveau par rapport à la loi dite initiale, qui est susceptible d'apparaître et qui dans ce cas apporte une modification importante de comportement. De manière plus précise, les critères envisagés concernent :

— l'apparition de flambages locaux voir micro flambages, tels que l'apparition de plis dans une membrane en compression, ceci dans le cas ou on veut prendre en compte les conséquences de ce flambage local au niveau d'une loi qui est censée représenter le comportement homogénéisé d'un tissu avec plis.

L'implémentation des critères est envisagée de manière à pouvoir s'appliquer sur des classes quelconques de loi de comportement. Ainsi dans les développements qui suivent, la loi n'est pas précisée et on cherche à limiter les restrictions qui pourraient lui être imposées pour l'utilisation d'un critère donné.

51.11.1 Critère permettant de prendre en compte l'apparition de plis sur une membrane

La membrane ne doit pas pouvoir supporter des efforts de compression sinon il y a apparition de plis.

Le critère est mixte en contraintedéformation. Il fonctionne de la manière suivante :

— soit $\sigma_{I\ min} > 0$: $\Rightarrow$ la membrane est en tension dans toutes les directions de son plan médian

— soit $\varepsilon_{I\ max} < 0$: $\Rightarrow$ la membrane est relâchée dans toutes les directions de son plan médian

— sinon il y a un pli dans la direction de $\sigma_{I\ min}$

S'il y a pli : $\Rightarrow$ la raideur apparente du matériau et les contraintes et déformations résultantes sont modifiées. En particulier les déformations dans la direction des plis ne sont plus reliées avec la loi de comportement $\searrow$ de la stabilité de la structure

Le critère s'applique à une loi 2D contrainte plane qui doit s'appuyer sur un comportement 3D quelconque. L'application du critère entraîne une des 3 possibilités proposées :

— soit il n'y a pas de plis, l'application du critère n'entraîne aucune conséquence sur les contraintes et sur l'opérateur tangent,

— soit il y a un tel niveau de plis que l'état résultant de la membrane est totalement relâché, dans ce cas les contraintes et l'opérateur tangent résultant sont annulés

— soit il y a des plis dans une direction, mais il reste une direction tendue d'où une raideur et des contraintes résultantes dans cette direction. Un nouvel état d'équilibre de type contrainte plane double, est calculé dans cette direction d'où, par rapport aux résultats initiaux avant l'application du critère, de nouvelles valeurs pour les contraintes et l'opérateur tangent.

La table 186 présente un exemple d'utilisation d'une loi critère pour tenir compte de l'apparition de plissement de membrane. Dans cet exemple la loi 3D est l'élasticité linéaire. Elle peut être remplacée par n'importe quelle loi 3D.

En post-traitement il est possible de récupérer via le mot clé " DIRECTION_PLI " indiqué en interactif, deux vecteurs permettant d'identifier les directions des plis.

Lorsqu'il y a un pli dans une seule direction, le premier vecteur indique la direction selon laquelle la membrane est en traction. Son intensité est égale à la déformation cinématique dans le sens transverse c'est-à-dire la direction du plis. Cette déformation est différente de la déformation mécanique calculée par la loi 3D, c'est-à-dire issue de la contraction de la zone, due à la traction suivant la première direction. Cette déformation mécanique peut-être récupérée via la déformation " DEF_LARGEUR " associée à la loi de comportement. Le second vecteur est nul, ce qui permet de distinguer ce cas avec celui pour lequel il y a des plis dans les deux sens.

Lorsqu'il y a des plis dans deux directions, les deux vecteurs indiquent les directions principales d'apparition de plis. L'intensité des vecteurs est égale à la déformation cinématique dans la direction associée.

Lorsqu'il n'y a pas de plis, l'intensité des vecteurs est nulle.

TABLE 186 – Exemple de déclaration d'une loi critère dans le cas de plissement de membrane.

```
  plis_elastique         LOI_CRITERE
# .......  loi de comportement avec critere ........
    TYPE_DE_CRITERE_  PLISSEMENT_MEMBRANE
      LOI_CONTRAINTES_PLANES # loi de contrainte plane
          NEWTON_LOCAL  avec_parametres_de_reglage_
                    nb_iteration_maxi_  20
                    nb_dichotomie_maxi_  20
                    tolerance_residu_ 1.e-3
                fin_parametres_reglage_Algo_Newton_

     ISOELAS
       2000  0.3
     fin_loi_contrainte_plane  # --- fin de la loi de contrainte plane
    fin_loi_interne # --- fin des lois internes
  fin_loi_critere   # ----- fin de LOI_CRITERE
```

Il est possible et d'ailleurs souhaitable d'indiquer des paramètres de contrôles différents pour la partie contrainte plane et la partie contrainte doublement plane. La table 187 donne un exemple de déclaration.

Les paramètres de contrôle pour la loi doublement plane, sont délimités par deux mots clés tels que :

— parametres_controle_pour_contraintes_planes_double_

— ...

— les paramètres de contrôles (se référer à la loi : contraintes planes doubles pour la liste exhaustive)

— ...

— fin_parametres_controle_pour_contraintes_planes_double_

Bien noter que le critère plis s'applique en deux étapes :

1. calcul des contraintes en utilisant tout d'abord la loi de contrainte plane
2. analyse des contraintes et déformation : s'il y a des plis,
   (a) soit on recalcule les contraintes avec la loi en contraintes planes doubles si les plis existent dans une seule direction
   (b) soit on met les contraintes à 0 si les plis existent dans deux directions

   S'il n'y a pas de plis : on conserve les contraintes obtenues initialement en contrainte plane.

En conséquence, lorsque des plis existent, cela signifie que la membrane, qui est modélisée par une surface non plissée, va se contracter dans une direction, ou dans deux directions. Aussi, les déformations d'entrée de la loi en contrainte plane sont pour partie ou totalement négatives. On obtiendra donc a priori une augmentation d'épaisseur ce qui est normal à cette étape. On doit donc indiquer pour le paramètre

maxi_hsurh0_

une valeur supérieure à 1.

Ensuite, lorsqu'il y a une seule direction de plis, la loi en contraintes planes doubles est appelée. Les contraintes calculées correspondent à un étant de traction simple, ce qui doit généralement conduire à une diminution d'épaisseur et de largeur. On doit donc indiquer ici pour les contraintes planes doubles :

maxi_hsurh0_

= 1. et

maxi_bsurb0_

= 1.

De même on peut imaginer que les précisions des algorithmes de Newton, pour les contraintes planes et doublement planes, puissent être différentes. Cela dépend sans doute des applications.

Remarque Noter également dans l'exemple 187, la présence du mot clé

permet_affichage_

qui permet d'avoir des messages concernant le déroulement des algorithmes de Newton, en particulier s'il y a des problèmes de convergence.

TABLE 187 – Exemple de déclaration d'une loi mélange, comportant une partie loi critère de plissement de membrane avec une définition différente des paramètres de contrôle pour les contraintes planes et les contraintes doublement planes

```
 #-- Loi film
loi_film  LOI_DES_MELANGES_EN_SIGMA
  les_grandeurs_de_controle_= TEMPS  fin_grandeurs_
  deb_fonct_= f_ponderation fin_fonct_

    #-- première loi
    ISOELAS2D_C demarre_a_prop_non_nulle_puis_strictement_decroissante_
        2.500e+08  0.30

   #-- seconde loi
    LOI_CRITERE demarre_a_prop_non_nulle_puis_strictement_decroissante_
    TYPE_DE_CRITERE_  PLISSEMENT_MEMBRANE
        LOI_CONTRAINTES_PLANES
            NEWTON_LOCAL  avec_parametres_de_reglage_
                nb_iteration_maxi_  20
                nb_dichotomie_maxi_ 20
                tolerance_residu_ 1e1  # tolerance residu 10 Pa
                tolerance_residu_rel_ 1.e-4
                maxi_hsurh0_  10.
                mini_hsurh0_  0.01
                permet_affichage_ 0#5
            fin_parametres_reglage_Algo_Newton_

           ISOELAS
           2.500e+08  0.30

        fin_loi_contrainte_plane
    fin_loi_interne
    parametres_controle_pour_contraintes_planes_double_
      NEWTON_LOCAL avec_parametres_de_reglage_
                    nb_iteration_maxi_  20
                    nb_dichotomie_maxi_  20
                    tolerance_residu_ 1e1
                    tolerance_residu_rel_ 1.e-4
                    mini_hsurh0_ 1.e-2
                    mini_bsurb0_ 1.e-2
                    maxi_hsurh0_ 1.
                    maxi_bsurb0_ 1.
                    sortie_post_ 1
                    permet_affichage_ 0#4
                 fin_parametres_reglage_Algo_Newton_
    fin_parametres_controle_pour_contraintes_planes_double_
fin_loi_critere
fin_liste_lois_elementaires
#-------------------------------------------------------
```

51.11.2 Critère permettant de prendre en compte l'apparition de plis sur une biellette

On se place dans le cas d'une biellette qui est censée ne pas pouvoir supporter d'efforts de compression, ces derniers conduisant à des plis. Un exemple est le comportement d'un fil.

Le critère est en contrainte seule, la partie déformation n'apportant pas a priori d'information pertinente. Le fonctionnement est le suivant. On commence par calculer la contrainte due à la loi 1D (qui peut-être quelconque) associée à la biellette.

- soit $\sigma_I >= 0$: $\Rightarrow$ la biellette est en tension selon sa fibre médiane, il n'y a pas de pli,
- soit $\sigma_I < 0$: $\Rightarrow$ la biellette est en compression selon sa fibre médiane, on considère qu'il y a apparition de pli.

S'il y a pli : $\Rightarrow$ la raideur apparente du matériau et la contrainte et déformation résultante sont modifiées.

Plus précisément l'application du critère entraîne une des 2 possibilités proposées :

- soit il n'y a pas de plis, l'application du critère n'entraîne aucune conséquence sur la et sur l'opérateur tangent,
- soit il y a pli et l'état résultant de la biellette est totalement relâché, dans ce cas la contrainte et l'opérateur tangent résultant sont annulés

La table 188 présente un exemple d'utilisation d'une loi critère pour tenir compte de l'apparition de plissement de biellette. Dans cet exemple la loi 1D est l'élasticité linéaire. Elle peut être remplacée par n'importe quelle loi 1D.

En post-traitement il est possible de récupérer via le mot clé " DIRECTION_PLI " indiqué en interactif, un vecteur permettant d'identifier la direction des plis.

Le vecteur indique directement la direction de la biellette, donc la direction de la traction ou compression. Dans le cas sans pli, l'intensité du vecteur est nulle et dans le cas avec pli, l'intensité a pour valeur la déformation de compression.

TABLE 188 – Exemple de déclaration d'une loi critère dans le cas de plissement de biellette.

```
loi_avec_plis    LOI_CRITERE
 TYPE_DE_CRITERE_  PLISSEMENT_BIEL
   ISOELAS1D
     210000  0.3
   fin_loi_interne # --- fin des lois internes
 fin_loi_critere   # --- fin de la loi critere
```

51.11.3 Pilotage des lois critères

Par défaut la loi critère s'applique de manière identique du début du calcul à la fin. Il peut être intéressant de pouvoir gérer l'application du critère en fonction de grandeur(s). Par exemple pour un critère d'apparition de plis, au début de la recherche d'un équilibre, on pourrait ne pas appliquer de critère, donc effectuer un calcul avec une loi de contrainte plane seule, puis lorsque l'équilibre approche, basculer sur un comportement plus précis qui prend en compte l'application du critère.

L'objet de cette section est de présenter les possibilités existantes dans Herezh pour mettre en oeuvre ce type de calcul. Pour cela on s'appuie tout d'abord sur un exemple.

Supposons la mise en donnée suivante :

```
membrane          LOI_CRITERE
# .......  loi de comportement avec critere ........
TYPE_DE_CRITERE_  PLISSEMENT_MEMBRANE avecNiveauSigmaI_mini_pour_plis_
 les_grandeurs_de_controle_= TEMPS \
                compteur_increment_charge_algo_global\
                compteur_iteration_algo_global\
 fin_grandeurs_
 deb_fonct_= f_temps fonction1 fin_fonct_

     LOI_CONTRAINTES_PLANES # loi de contrainte plane
        NEWTON_LOCAL  avec_parametres_de_reglage_
          nb_iteration_maxi_  20
          nb_dichotomie_maxi_  20
          tolerance_residu_ 1.e-4
          tolerance_residu_rel_ 1.e-3
       fin_parametres_reglage_Algo_Newton_

       ISOELAS
         #    module d'young :   coefficient de poisson
         3500   0.31
     fin_loi_contrainte_plane  # --- fin de la loi de contrainte plane
   fin_loi_interne # --- fin des lois internes
fin_loi_critere   # ----- fin de LOI_CRITERE
```

Par rapport au cas sans pilotage, on observe tout d'abord le mot clé " avecNiveauSigmaI_mini_pour_plis_ " . Sa présence indique qu'un contrôle du niveau de la contrainte principale mini est prévu.

Ensuite la ligne suivante définit la liste des grandeurs possibles pour piloter le contrôle. Dans l'exemple, il s'agit du temps (" TEMPS ") en cours dans le calcul, le numéro de l'incrément en cours dans l'algorithme global (" compteur_increment_charge_algo_global ") et le numéro de l'itération en cours dans l'algorithme global (" compteur_iteration_algo_global "). Les grandeurs utilisées sont comprises entre deux mots clés : " les_grandeurs_de_contrôle_= " et " fin_grandeurs_ " .

La ligne suivante définit les fonctions qui vont utiliser les grandeurs. Dans l'exemple, deux fonctions sont définies. La première "f_temps" est une courbe 1D, qui doit utiliser la

première grandeur donc il s'agit dans l'exemple du temps. La seconde "fonction1" est une fonction multidimensionnelle, qui utilise toutes les grandeurs globales. Dans l'exemple la fonction doit être à deux variables. La définition des fonctions s'effectue entre les deux mots clés : " deb_fonct_= " et " fin_fonct_ "

Ensuite on retrouve la mise en donnée classique d'une loi critère.

Pour information, voici par exemple une mise en donnée de la courbe et de la fonction qui est cohérente avec la loi :

```
#-----------------------------
# Definition des courbes
#-----------------------------
   les_courbes_1D
  #------------------

 f_temps COURBEPOLYLINEAIRE_1_D
      Debut_des_coordonnees_des_points
         Coordonnee dim= 2 0.0  1.
         Coordonnee dim= 2 1.  1.
      Fin_des_coordonnees_des_points

   les_Fonctions_nD
   #-----------------------

#    f(i,j) = une expression de i et j
fonction1 FONCTION_EXPRESSION_LITTERALE_nD
  un_argument= i un_argument= j
  fct= ((i<2)?((j<1000)?-50.:0.):((j<400)?-10.: 0.))
fin_parametres_fonction_expression_litterale_
```

En fait 3 types de grandeurs peuvent être utilisées pour le contrôle :

- le temps : " TEMPS " , qui nécessite une fonction spécifique
- les grandeurs globales au nombre maxi <= 5, qui nécessite une seule fonction multidimensionnelle
- les degrés de liberté étendue, qui nécessite une fonction spécifique "par" ddl étendu.

Avant le calcul du critère, toutes les fonctions " f_i" sont évaluées et le choix entre contraintes plane (donc pas de critère) et l'application du critère s'effectue en fonction de :

$$si(\sigma^I_{mini} < \Pi_i \ f_i) \qquad \text{alors on applique la loi critère}$$
$$\text{sinon on applique la loi de contrainte plane}$$

Les grandeurs globales actuellement disponibles sont indiquées en 88.1.

Les degrés de liberté (ddl) étendue sont en fait de plusieurs types.

- Les ddl classiques qui sont définies aux noeuds, dont la liste est donnée en 80.11.4. Cependant, la liste contient des ddl qui ne sont pas toujours présents ! Par exemple si on indique "V1", dans le cas d'un calcul dynamique, le ddl sera présent, alors que dans un calcul statique il ne sera pas présent. Par contre on peut imposer la

présence d'une grandeur, par exemple la température, en définissant un champ de température aux noeuds. Dans ce cas le ddl " TEMP " sera présent. Etc. En résumé, la présence ou non d'un ddl dépends du calcul effectué.

— Mais cette liste intègre également des ddl possibles venant des points d'intégration. Comme la loi de comportement est en générale calculée au point d'intégration, il est préférable d'utiliser directement les grandeurs au point d'intégration si elles y sont disponibles : en général : contrainte, déformation, vitesse de déformation. Ceci étant, les composantes de ces tenseurs ne sont pas a priori disponibles (bien qu'il soit possible possible de les rajouter à la demande très facilement, car elles sont directement disponibles). Ces composantes ne sont pas intrinsèques aussi par défaut seules les grandeurs suivantes sont disponibles : (cf.80.11) :
  — la déformation équivalente, mot clé : " def_equivalente "
  — la déformation duale de Mises maxi, mot clé : " def_duale_mises_maxi "
  — la vitesse de déformation équivalente, mot clé : " vitesse_def_equivalente "
  — la déformation équivalente de Mises, mot clé : " def_duale_mises "
  — la partie sphérique de la déformation, mot clé : " Spherique_eps "
  — la partie sphérique de la contrainte, mot clé : " Spherique_sig "

Remarques

1. Ne pas oublier qu'il y a des vérifications d'existence, de taille, etc. qui sont faites avec la version non "fast", mais par contre qui ne sont pas faites avec la version "fast". Aussi, si l'on observe une erreur d'exécution, il est intéressant de reprendre le calcul avec la version lente pour une vérification plus précise. Par exemple l'accès aux grandeurs suppose que ces grandeurs existent. Dans le cas où une grandeur est absente, la version lente est censée donner un message explicite de l'erreur, par contre la version rapide peut éventuellement s'interrompre avec un message incompréhensible.
2. Si cela s'avère nécessaire, il est possible d'ajouter de nouvelles grandeurs dans les différentes listes !

En plus des informations données via l'exemple de présentation, la mise en donnée doit respecter les règles suivantes :

— Le nombre de grandeurs globales est pour l'instant limité à 5 qui correspond en fait à la limite du nombre d'arguments des fonctions multidimensionnelles.

— dans le cas de l'utilisation de ddl étendue, il faut indiquer dans la liste des fonctions associées : "la fonction" suivie d'un mot clé qui indique si la grandeur est définie au point d'intégration (mot clé : " AuPti_ ") ou aux noeuds qui entoure le point d'intégration (mot clé " AuxNoeuds_ ") ce qui oblige d'interpoler la grandeur pour la calculer au point d'intégration.

  Par exemple on pourrait avoir la mise en donnée partielle suivante :

```
membrane          LOI_CRITERE
# .......  loi de comportement avec critere ........
```

```
TYPE_DE_CRITERE_  PLISSEMENT_MEMBRANE avecNiveauSigmaI_mini_pour_plis_
 les_grandeurs_de_controle_= TEMPS \
                compteur_increment_charge_algo_global\
                compteur_iteration_algo_global\
                def_equivalente  X1 \
                 fin_grandeurs_

 deb_fonct_= f_temps fonction1 f_const AuPti_ f_const AuxNoeuds_ fin_fonct_
```

— au final il doit y avoir une fonction + un mot clé de localisation, pour chaque grandeur de type ddl étendue, une fonction pour le temps si ce dernier fait partie des grandeurs, et une seule fonction pour toutes les grandeurs globales.

51.11.4 Paramètres de contrôle généraux des lois critères

D'une manière générale, ces paramètres s'écrivent entre deux mots clés :
avec_parametres_controle_pour_loi_critere_
et fin_parametres_controle_pour_loi_critere_
qui doivent se trouver juste avant le mot clé fin_loi_critere

La liste actuelle de paramètres est :

1. permet_affichage_ suivi d'un entier indiquant le niveau local d'affichage de la partie spécifique à l'application du critère. Ce niveau est indépendant de celui des lois internes.
2. recalcul_dir_plis_ suivi de la définition d'une fonction nD qui doit renvoyer une valeur qui sera prise en compte comme un entier relatif. Par défaut la direction des plis est recalculée à chaque itération. Dans le cas du mot clé recalcul_dir_plis_ avec la définition d'une fonction nD, avant chaque calcul de la direction des plis, la fonction est appelée et en fonction de sa valeur de retour :
   — si = 1 : la direction des plis est recalculée.
   — si = 0 : la direction des plis n'est pas recalculée, on utilise la valeur calculée à **l'itération** précédente. Dans le cas où à l'itération précédente il n'y avait pas de plis, un premier calcul de direction est effectué. Au final, dans ce cas, seules les zones avec un pli ou 2 plis sont figées d'une itération à l'autre.
   — si = -1 : la direction des plis n'est pas recalculée, on utilise la valeur calculée à **l'incrément** précédent "t". Cependant, dans le cas où à l'incrément précédent il n'y avait pas de plis, un premier calcul de direction est effectué. Au final, dans ce cas, seules les zones avec un pli ou 2 plis sont figées d'un incrément à l'autre.
   — si = -2 : la direction des plis n'est pas recalculée, on utilise la valeur calculée à **l'incrément** précédent "t". Dans le cas où à l'incrément précédent il n'y avait pas de plis, on considère que cette situation perdure. Ainsi, les zones : sans plis, avec un pli ou avec 2 plis, sont complètement figées d'un incrément à l'autre.
   — si = -3 : la direction des plis n'est pas recalculée, on utilise la valeur calculée à **l'itération** précédente. Dans le cas où à l'itération précédente il n'y avait pas

de plis, on considère que cette situation perdure. Ainsi, les zones : sans plis, avec un pli ou avec 2 plis, sont complètement figées d'une itération à l'autre.

Au lieu d'utiliser une fonction nD déjà définie, il est possible de définir une nouvelle fonction localement, à la suite du mot clé `recalcul_dir_plis_` .

3. `choix_methode_cal_plis_memb_` suivi de "1" ou "2" , ce mot clé permet de définir le type de méthode a utiliser pour le critère plis. Par défaut c'est la méthode "1" qui est utilisée (première méthode implantée). La seconde méthode est plus générale et plus précise dans le cas d'une loi anisotrope, et/ou incrémentale. On se réfèrera à (cf. [RIO, 2019]) pour les détails théoriques. Les paramètres de contrôles sont globalement identiques pour les deux méthodes.

51.12 Lois d'hystérésis.

Les lois d'élasto-hystérésis disponibles sont données dans la table (cf.189).

TABLE 189 – liste des différentes lois d'élasto-hystérésis

identificateur	indications	ref du commentaire
HYSTERESIS_1D	1D : loi d'hystérésis classique (modèle de Guélin-Favier-Pégon)	(51.12.1)
HYSTERESIS_3D	3D : loi d'hystérésis classique modifiée (modèle de Guélin-Favier-Pégon-Bless-Rio)	(51.12.2)
HYSTERESIS_BULK	3D : loi d'hystérésis appliquée au comportement sphérique (Rio)	(51.12.3)

51.12.1 HYSTERESIS_1D

Identificateur d'une loi d'hystérésis 1D. La loi comporte d'une part une partie incrémentale dont l'équation est :

$$\dot{\boldsymbol{\sigma}} = 2\mu\bar{\boldsymbol{D}} + \beta\phi\boldsymbol{\Delta}^{\boldsymbol{t}}_{\boldsymbol{R}}\bar{\boldsymbol{\sigma}} \tag{89}$$

avec :

$$\begin{aligned} Q_{\Delta\sigma} &= \sqrt{trace(\Delta^t_R\boldsymbol{\sigma}.\Delta^t_R\boldsymbol{\sigma})} \\ \phi &= \Delta^t_R\boldsymbol{\sigma} : \bar{\boldsymbol{D}} \\ \beta &= \frac{-2\mu}{(w'Q_0)^{np}(Q_{\Delta\sigma})^{2-np}} \end{aligned} \tag{90}$$

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 [Guélin, 1980] [P. Pegon, 1987] [Pegon *et al.*, 1991] [Wack *et al.*, 1983], Manach[Favier *et al.*, 1997] [Manach *et al.*, 1996] [Rio *et al.*, 1995b], Orgeas, Tourabi,[Orgéas, 1997] [Orgéas et Favier, 1995] [Tourabi *et al.*, 1995] Couty[Couty, 1999], Bless[Bles, 2002], Rio, Laurent [Favier *et al.*, 2002] [Favier *et al.*, 1997] [Favier *et al.*, 2010a] [Favier *et al.*, 2010b] [Laurent *et al.*, 2007] [RIO, 2017] [RIO, 2019] [Rio *et al.*, 1995b] [Rio *et al.*, 1995a] [Rio *et al.*, 2009] [Rio *et al.*, 2008a] [Vandenbroucke *et al.*, 2010] [Zrida *et al.*, 2009] pour plus d'informations sur le modèle d'élasto-hystérésis.

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 (190) 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 (51.12.2). 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 (191), 2 indique que l'on veut une résolution explicite avec une méthode de Runge-Kutta imbriquée, voir l'exemple (192),
- " nb_iteration_maxi_ " : utilisé avec la méthode de linéarisation, indique le nombre d'itérations maxi permis 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 absolue sur le résidu à convergence de la méthode de Newton, par défaut $1.e-3$, l'erreur globale tolérée sera :" tolerance_residu_ + tolerance_residu_rel_ $\times$ résidu"
- " tolerance_residu_rel_ " : utilisé avec la méthode de linéarisation, indique la tolérance relative sur le résidu à convergence de la méthode de Newton, par défaut $1.e-3$, l'erreur globale tolérée sera :" tolerance_residu_ + tolerance_residu_rel_ $\times$ résidu"
- " maxi_delta_var_sig_sur_iter_pour_Newton_ " : utilisé avec la méthode de Newton. Indique le maximum admissible sur $|\delta\sigma|$ à chaque itération de Newton. Lorsque l'algorithme calcule un nouvel incrément qui dépasse ce maximum, l'incrément est diminué suivant la formule : $\delta\sigma_{nouveau} = \delta\sigma \ . \ \delta\sigma_{max}/|\delta\sigma|$ où $\delta\sigma_{max}$ est la norme maxi que l'on a imposée.
- " 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'appels de la fonction dérivée (sigma point ici), par défaut 120,
- " tolerance_coïncidence_ " indique la tolérance que l'on accepte sur les coïncidences.

TABLE 190 – Exemple de déclaration d'une loi d'hystérésis 1D

```
#-------- debut cas d'une loi d'hysteresis --------
#----------------------------------------
# Nom Materiau    |      Type loi         |
#----------------------------------------
  AMF         HYSTERESIS_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 --------
```

TABLE 191 – 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 192 – 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 --------
```

51.12.2 HYSTERESIS_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 :

$$\dot{\boldsymbol{\sigma}} = 2\mu\bar{\boldsymbol{D}} + \beta\phi\boldsymbol{\Delta}^{\boldsymbol{t}}_{\boldsymbol{R}}\bar{\boldsymbol{\sigma}} \tag{91}$$

avec :

$$\begin{aligned} Q_{\Delta\sigma} &= \sqrt{trace(\Delta^t_R\boldsymbol{\sigma}.\Delta^t_R\boldsymbol{\sigma})} \\ \phi &= \Delta^t_R\boldsymbol{\sigma} : \bar{\boldsymbol{D}} - \frac{(Q_{\Delta\sigma})^2}{2\mu}\frac{\dot{w'}}{w'} \\ \beta &= \frac{-2\mu}{(w'Q_0)^{np}(Q_{\Delta\sigma})^{2-np}} \\ w' &= w\ cos(\alpha) \end{aligned} \tag{92}$$

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 $\boldsymbol{\Delta}^{\boldsymbol{t}}_{\boldsymbol{R}}\bar{\boldsymbol{\sigma}}$ et $\boldsymbol{\Delta}^{\boldsymbol{R}}_{\boldsymbol{Oi}}\bar{\boldsymbol{\sigma}}$, Oi 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 Guelin, Pegon, Favier [Guélin, 1980, P. Pegon, 1987, Pegon *et al.*, 1991, Wack *et al.*, 1983], Manach[Favier *et al.*, 1997, Manach *et al.*, 1996, Rio *et al.*, 1995b], Orgeas, Tourabi,[Orgéas, 1997, Orgéas et Favier, 1995, Tourabi *et al.*, 1995] Couty[Couty, 1999], Bless[Bles, 2002], Rio, Laurent [Favier *et al.*, 2002, Favier *et al.*, 1997, Favier *et al.*, 2010a, Favier *et al.*, 2010b, Laurent *et al.*, 2007, RIO, 2017, RIO, 2019, Rio *et al.*, 1995b, Rio *et al.*, 1995a, Rio *et al.*, 2009, Rio *et al.*, 2008a, Vandenbroucke *et al.*, 2010, Zrida *et al.*, 2009] pour plus d'informations sur le modèle d'élasto-hystérésis.

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 $\tau = Q0/\sqrt{(2)}$
- 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 (194) 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 (196) donne un exemple de déclaration avec 3 paramètres

thermo-dépendants, et la La table (197) donne un exemple de déclaration pour un premier paramètre fixe, les deux autres étant thermo-dépendants.

De manière à 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 couples parametre-grandeur sur chaque ligne, par contre la dernière ligne doit comporter uniquement le mot cle : " fin_parametres_reglage_Hysteresis_ ".

Les différents paramètres 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 résolue par une méthode itérative de Newton (avec sous-découpage de l'incrément de déformation, si la précision voulue n'est pas obtenue), voir l'exemple (191). 2 indique que l'on veut une intégration explicite de l'équation différentielle avec une méthode de Runge-Kutta imbriquée, voir l'exemple (192). 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 direct à un Runge-Kutta d'ordre 5 et l'erreur estimée est écrite (si le niveau d'impression est supérieur à 0). Différentes variantes sont également disponibles :

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 global 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 global 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 rebascule sur la méthode de Kutta imbriquée, 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 garantit 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'itérations maxi permis 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 absolue sur le résidu à convergence de la méthode de Newton. "Ex : tolerance_residu_ 1.10^{-5} ".

— " tolerance_residu_rel_ " : la tolérance relative sur le résidu. Par exemple : " tolerance_residu_rel_ 1.e-3 " signifie que la précision sera = " tolerance_residu_ $+ || {}^{t+\Delta t}\bar{\boldsymbol{\sigma}}||\times$ tolerance_residu_rel_ "

— " maxi_delta_var_sig_sur_iter_pour_Newton_ " : utilisé avec la méthode de Newton. Indique le maximum admissible sur $||\boldsymbol{\delta\sigma}||$ à chaque itération de Newton. Lorsque l'algorithme calcule un nouvel incrément qui dépasse ce maximum, l'incrément est diminué suivant la formule : $\boldsymbol{\delta\sigma}_{nouveau} = \boldsymbol{\delta\sigma} \ . \ (\delta\sigma_{max}/||\boldsymbol{\delta\sigma}||)$ où $\delta\sigma_{max}$ est la norme maxi que l'on a imposée.

— " 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'appels 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". Si le nombre indiqué est négatif, cela signifie que la tolérance sera relative a la valeur de la norme de la contrainte de référence : par exemple -0.1 signifie que la tolérance sera $MAX(0.1 \times || {}^{R}\bar{\boldsymbol{\sigma}}||, 0.1)$

— " tolerance_centre_ " : indique la tolérance sur le calcul des hyper-centres de trajet neutre. "Ex : tolerance_centre_ 1.e-3 ". Si le nombre indiqué est négatif, cela signifie que la tolérance sera relative à la valeur de la norme de la contrainte de référence : ex -0.1 signifie que la tolérance sera $MAX(0.1 \times || {}^{R}\bar{\boldsymbol{\sigma}}||, 0.1)$

— " mini_rayon_ " : le mini en dessous duquel on considère $\Delta^{R}_{0_i}\bar{\boldsymbol{\sigma}}$ et $\Delta^{t+\Delta t}_{R}\bar{\boldsymbol{\sigma}}$ 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 finale 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 $\Delta^R_{0_i}\bar{\boldsymbol{\sigma}}$ et $\Delta^{t+\Delta t}_R\bar{\boldsymbol{\sigma}}$. 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 : possibilite_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 indiquer le mot cle “ avecVarWprimeS_ ” suivi de 1 (0 étant la valeur par défaut). “Ex. : avecVarWprimeS_ 1 ”

— ” mini_Qsig_pour_phase_sigma_Oi_tdt_ ” : indique le minimum de Qsig en dessous duquel on considère que la phase pour $\Delta^{t+\Delta t}_{0_i}\bar{\boldsymbol{\sigma}}$ est nulle ”.
Ex ” mini_Qsig_pour_phase_sigma_Oi_tdt_ 1.5 ”.
Dans le cas ou ” mini_Qsig_pour_phase_sigma_Oi_tdt ” est négatif, cela devient un paramètre de régularisation.
Qsig est alors transformé en (Qsig- mini_Qsig_pour_phase_sigma_Oi_tdt) au niveau des divisions qui servent à calculer la phase. Dans ce cas l'angle de phase est toujours calculable, par contre pour les faibles valeurs de Qsig, on peut avoir angle erroné, mais ce n'est peut-être pas un problème !

— ” mini_QepsilonBH_ ” : indique le minimum de Q_ϵ en dessous duquel on considère que la phase pour la déformation totale est nulle. Ex ” mini_QepsilonBH_ 1.e-4 ”. Même comportement que pour ” mini_Qsig_pour_phase_sigma_Oi_tdt_ ”, dans le cas ou ” mini_QepsilonBH_ ” est négatif ce paramètre devient un paramètre de régularisation.

— ” force_phi_zero_si_negatif_ ” : Au niveau de l'équation constitutive, Φ doit-être toujours positif, cependant pendant le processus de convergence il peut être transitoirement négatif : on peut néanmoins forcer Φ à être toujours >= 0 avec le paramètre : ” force_phi_zero_si_negatif_ ”. Par défaut : force_phi_zero_si_negatif_ = 0 : on ne force pas la mise à 0. Si on indique ” force_phi_zero_si_negatif_ 1 ” : on force la mise à 0 lorsque Φ devient négatif.

— ” type_de_transport_memorisation_ ” : Il est également possible de préciser un type de transport des grandeurs mémorisées.
Dans ce cas, on met le mot clé : ” type_de_transport_memorisation_ ” suivi d'un entier signe 'i'.
  — i = 0 : transport historique en mixte (valeur par défaut actuellement)
  — i = -1 : transport incrémental (à la fin de chaque incrément) de manière cohérente avec la dérivée de Jauman.

— " depassement_QO_ " : Normalement l'intensité du déviateur des contraintes Qsig doit être inférieure a la saturation Q_0. Le paramètre " depassement_QO_ " donne la valeur tolérée de dépassement, au-dessus de laquelle la contrainte n'est plus recevable. Ex : " depassement_QO_ 2.2" signifie que l'on tolère 10% de déplacement.

— " permet_affichage_ " permet l'affichage des erreurs et des warnings 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 normal et de l'affichage particulier. "Ex : permet_affichage_ 5"

— " sortie_post_ " Accès aux indicateurs de la résolution : À chaque résolution, il est possible de stocker les indicateurs : nombre d'itérations, d'incrément, précision, etc. Les indicateurs sont renseignés en fonction du type de résolution. Le mot clé pour stocker les indicateurs est " sortie_post_ ". Par défaut il vaut 0, dans ce cas aucun indicateur n'est stocké. Sil est différent de 0, on peut accéder aux indicateurs en post-traitement. Seuls les indicateurs en cours sont disponibles, il n'y a pas de stockage sur plusieurs incréments. Il faut donc utiliser une "sortie au fil du calcul " pour visualiser une évolution des indicateurs au cours du chargement. Ex : " sortie_post_ 1 "

Le tableau (193) donne les valeurs par défaut des différents paramètres.

La table (195) 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 $\Delta \boldsymbol{S}_{O_i}^{t+\Delta t}$, O_i étant le centre de référence actuellement actif. Pour introduire la dépendance à la 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 contrôle de la dépendance. 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 après chaque courbe). La table (199) donne un exemple partiel de déclaration de dépendance à la phase, qui intègre également des paramètres de réglage de l'algorithme :

Dépendance à une déformation équivalente de manière à mieux maîtriser le passage du régime principalement d'élasticité au régime principalement de plasticité, qui est gérer 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}

$$\epsilon_{equi} = \int_R^t \frac{\phi}{\omega' \, Q_{\Delta \boldsymbol{\sigma}}} d\tau \tag{93}$$

TABLE 193 – 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}
tolerance_centre_	1.10^{-3}	1.10^{-3}
nb_maxInvCoinSurUnPas_	4	-10
min_angle_trajet_neutre_	0.008	0.008
avecVarWprimeS_	0	0
mini_Qsig_pour_phase_sigma_Oi_tdt	1.5	?
mini_QepsilonBH_	1.e-4	?
force_phi_zero_si_negatif_	0	0
type_de_transport_memorisation_	0	-1
depassement_Q0_	2.	?
permet_affichage_	0	0
sortie_post_	0	0

TABLE 194 – 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 195 – 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_
```

Cette dépendance a été introduite par Pierre Pégon [Pegon, 1988] (cf. document de thèse : annexe II.2 formule A.2.1), puis reprise par Laurent Orgéas [Orgéas, 1997](cf.

TABLE 196 – 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 197 – 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 198 – 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 --------
```

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 :

$$np = (2 - c1)\ f\left(-\epsilon_{equi}\ \left(\frac{2\ Q_0^2}{\omega'\ \mu}\right)\ Q_{\Delta\boldsymbol{\sigma}}\right) + c1 \tag{94}$$

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 (94). Suit un exemple de déclaration. Comme pour toutes les courbes on peut soit mettre

TABLE 199 – Exemple de déclaration d'une loi d'hystérésis 3D avec dépendance à la phase

```
# ---- 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_
```

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 correct 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 cet 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.

51.12.3 HYSTERESIS_BULK

Identificateur d'une loi d'hystérésis bulk.

Cette loi a pour objectif de décrire un comportement d'hystérésis (analogue à de la plasticité) sur la partie sphérique des contraintes. La loi est 3D et traite ainsi du comportement de la pression. Elle peut s'appliquer par exemple à des comportements de densification non réversible.

L'algorithme suit pour partie une logique analogue au cas 1D. Elle comporte d'une part une partie incrémentale dont l'équation est :

$$-\dot{\boldsymbol{P}} = 2\mu \boldsymbol{\Delta V} \left(1 - \frac{Q_{\Delta P}^{np}}{(wQ_0)^{np}}\right) \tag{95}$$

avec :

$$Q_{\Delta P} \quad = \quad |(\Delta_R^t \boldsymbol{P})| \tag{96}$$

w est le paramètre de Masing. 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. Cette gestion s'effectue à l'aide d'une fonction d'aide $\Delta_R^t W$

$$\Delta_R^t W = \int_R^t \frac{-\Delta_R^t \boldsymbol{P}\ \dot{V}}{dt} \tag{97}$$

On se reportera aux travaux théoriques de Guelin, Pegon, Favier [Guélin, 1980, P. Pegon, 1987, Pegon *et al.*, 1991, Wack *et al.*, 1983], Manach[Favier *et al.*, 1997, Manach *et al.*, 1996, Rio *et al.*, 1995b], Orgeas, Tourabi,[Orgéas, 1997, Orgéas et Favier, 1995, Tourabi *et al.*, 1995] Couty[Couty, 1999], Bless[Bles, 2002], Rio, Laurent [Favier *et al.*, 2002, Favier *et al.*, 1997, Favier *et al.*, 2010a, Favier *et al.*, 2010b, Laurent *et al.*, 2007, RIO, 2017, RIO, 2019, Rio *et al.*, 1995b, Rio *et al.*, 1995a, Rio *et al.*, 2009, Rio *et al.*, 2008a, Vandenbroucke *et al.*, 2010, Zrida *et al.*, 2009] pour plus d'informations sur l'algorithme de gestion via la fonction d'aide.

Les paramètres de la loi de comportement sont ainsi : μ qui représente la pente initiale, Q_0 qui représente le maxi des boucles en pression , np le paramètre de Prager qui contrôle le passage de la pente initiale au seuil maxi avec la limitation : $0 < np$.

La table (200) 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 (51.12.2). 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 (certain paramètre peuvent être absents) :

— " `type_de_resolution_` " : par défaut 1 c'est-à-dire par linéarisation de l'équation constitutive, 2 indique que l'on veut une résolution explicite avec une méthode de Runge-Kutta imbriquée,

— “ nb_iteration_maxi_ ” : utilisé avec la méthode de linéarisation, indique le nombre d'itérations maxi permis 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 absolue sur le résidu à convergence de la méthode de Newton, par défaut $1.e-3$, l'erreur globale tolérée sera :“ tolerance_residu_ + tolerance_residu_rel_ $\times$ résidu”

— “ tolerance_residu_rel_ ” : utilisé avec la méthode de linéarisation, indique la tolérance relative sur le résidu à convergence de la méthode de Newton, par défaut $1.e-3$, l'erreur globale tolérée sera :“ tolerance_residu_ + tolerance_residu_rel_ $\times$ résidu”

— ” maxi_delta_var_sig_sur_iter_pour_Newton_ ” : utilisé avec la méthode de Newton. Indique le maximum admissible sur $|\delta P|$ à chaque itération de Newton. Lorsque l'algorithme calcule un nouvel incrément qui dépasse ce maximum, l'incrément est diminué suivant la formule : $\delta P_{nouveau} = \delta P \ . \ \delta P_{max}/|\delta P|$ où δP_{max} est la norme maxi que l'on a imposée.

— “ 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'appels de la fonction dérivée (sigma point ici), par défaut 120,

— “ tolerance_coïncidence_ ” indique la tolérance que l'on accepte sur les coïncidences.

— ” nb_maxInvCoinSurUnPas_ ” : nombre maxi toléré d'inversions sur un pas Δt, 4 par défaut,

— ” depassement_Q0_ ” : tolérance de dépassement du maxi Q0 , par défaut 1.1,

— ” permet_affichage_ ” : permet de contrôler l'affichage local des messages de la loi, de manière plus précise que le paramètre global défini dans le .info, par défaut = 0,

— ” sortie_post_ ” : permet de stocker les indicateurs de la résolution de l'équation d'avancement (ex : nombre d'itérations, nombre de sous-pas ...) et ensuite d'y avoir accès en post-traitement

51.13 Lois hypo-élastiques

Les lois hypo-élastiques disponibles sont données dans la table (cf.201).

TABLE 200 – Exemple de déclaration d'une loi d'hystérésis bulk

```
#----------------------------------------
# Nom Materiau    |      Type loi         |
#----------------------------------------
 mousse        HYSTERESIS_BULK #hystérésis sphérique
      np= 1.75       mu= 1000    Qzero= 26   \
    avec_parametres_de_reglage_
        type_de_resolution_  2
        cas_kutta_  5
        erreurAbsolue_  1.e-5  erreurRelative_ 1.e-4
        nbMaxiAppel_  3600
        nb_iteration_maxi_  30  nb_dichotomie_maxi_  20
        tolerance_residu_ 1.e-3 tolerance_residu_rel_ 1.e-5
        depassement_Q0_ 1.1
        sortie_post_ 1
        permet_affichage_ 0
        tolerance_coincidence_ 1.e-10
        nb_maxInvCoinSurUnPas_ -20
        sortie_post_ 1
   fin_parametres_reglage_Hysteresis_
#-------- fin hysteresis_bulk --------
```

TABLE 201 – liste des différentes lois isotropes élastiques disponibles

identificateur	indications	ref du commentaire
HYPO_ELAS3D	3D : loi hypo-élastique linéaire $\dot{\boldsymbol{S}} = \mu\bar{\boldsymbol{D}}$ et $\dot{I}_\sigma = K_c I_{\boldsymbol{D}}$	(51.13.1)
HYPO_ELAS2D_C	2D : loi hypo-élastique contrainte plane $\dot{\boldsymbol{S}} = \mu\bar{\boldsymbol{D}}$ et $\dot{I}_\sigma = K_c I_{\boldsymbol{D}}$	(51.13.2)
HYPO_ELAS1D	1D : loi hypo-élastique $\dot{\boldsymbol{\sigma}} = f()\boldsymbol{D}$ et $\dot{I}_\sigma = K_c I_{\boldsymbol{D}}$	(51.13.3)

51.13.1 HYPO_ELAS3D

Identificateur d'une loi hypo-élastique 3D isotrope définie par les relations suivantes : $\dot{\boldsymbol{S}} = \mu\bar{\boldsymbol{D}}$ et $\dot{I}_\sigma = K_c I_{\boldsymbol{D}}$.

Remarque : Si on fait une analogie avec élasticité linéaire, on a pour la loi de Hooke : $K_c = E/(1-2\nu)$ et $\mu = E/(1+\nu)$ et notons que le coefficient de compressibilité vaut : $K = K_c/3$

Cette loi convient pour les éléments 3D. Elle nécessite la donnée d'un coefficient de compressibilité tangent $K_c/3$, 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 (202) donne un exemple de loi simple (pas de thermo-dépendance). La table (203) donne un exemple avec thermo-dépendance.

Il est également possible d'utiliser une compressibilité calculée 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 (204) 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 !

TABLE 202 – 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 defaut),
# = 1 -> derivee deux fois contravariantes
# dans le cas ou l'on veut une valeur differente de la valeur par defaut il faut mettre
# le mot cle <type_deriveee> 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 --------
```

51.13.2 HYPO_ELAS2D_C

Identificateur d'une loi hypo-élastique 2D en contraintes planes, isotrope définie par les relations suivantes : $\dot{\boldsymbol{S}} = \mu \bar{\boldsymbol{D}}$ et $\dot{I}_\sigma = K_c I_{\boldsymbol{D}}$. Cette loi convient pour les éléments 2D. Elle nécessite la donnée d'un coefficient de compressibilité tangent $K_c/3$, d'un module de cisaillement tangent μ et d'un type de dérivée matérielle pour l'intégration de la loi de comportement.

Remarque : Si on fait une analogie avec élasticité linéaire, on a pour la loi de Hooke : $K_c = E/(1-2\nu)$ et $\mu = E/(1+\nu)$ et notons que le coefficient de compressibilité vaut : $K = K_c/3$

L'entrée des données suit exactement la même logique que dans le cas 3D.

La table (205) donne un exemple de déclaration basique.

TABLE 203 – 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  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_thermo_dependant_   courbe3
      mu=  mu_thermo_dependant_   courbe2
      type_derivee  -1
                fin_loi_HYPO_ELAS3D

# le type de derivee est optionnel: = -1 -> derivee de jauman,
# = 0 -> derivee deux fois covariantes (valeur par defaut),
# = 1 -> derivee deux fois contravariantes
# dans le cas ou l'on veut une valeur differente de la valeur par defaut il faut mettre
# le mot cle <type_deriveee> suivi de la valeur -1 ou 0 ou 1"
# pour le module de compressibilite: Kc/3, il est possible d'indiquer
# que le module sera fourni
# par une loi thermophysique (celle associee au meme element),
# pour ce faire on indique uniquement:
# Kc=  Kc_avec_compressibilite_externe
# chaque parametre (a l'exclusion du cas Kc_avec_compressibilite_externe)
# 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 Kc:          Kc=   Kc_thermo_dependant_   courbe3
# exemple pour mu:          mu=   mu_thermo_dependant_   courbe2
# 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
# la derniere ligne doit contenir uniquement le mot cle:     fin_loi_HYPO_ELAS3D
#-------- fin cas d'une loi isoelas polymere --------
```

TABLE 204 – 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 205 – 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
```

51.13.3 HYPO_ELAS1D

Identificateur d'une loi hypo élastique 1D , isotrope définie par les relations suivantes :

$$\dot{\sigma}_1^{.1} = f()\ D_1^1 \quad \text{ou encore} \quad d\sigma_1^{.1} = f()\ d\varepsilon_1^{.1} \tag{98}$$

et

$$\frac{I_{\dot{\boldsymbol{\sigma}}}}{3} = \frac{\dot{\sigma}_1^{.1}}{3} = \frac{K_c}{3}\ I_{\boldsymbol{D}} = \frac{K_c}{3}\ D_i^{.i} \tag{99}$$

Cette loi convient pour les éléments finis 1D. Elle nécessite la donnée d'un coefficient de compressibilité tangent $K_c/3$, d'un module tangent $f()$ et d'un type de dérivée matérielle pour l'intégration de la loi de comportement.

Remarque : Si on fait une analogie avec élasticité linéaire, on a pour la loi de Hooke : $K_c = E/(1-2\nu)$ et $f() = E$ et notons que le coefficient de compressibilité vaut : $K = K_c/3$

L'entrée des données suit exactement la même logique que dans le cas 3D.

La table (206) donne un exemple de déclaration basique.

TABLE 206 – Exemple de déclaration d'une loi hypo élastique 1D

```
#----------------------------------------
# Nom Materiau    |      Type loi         |
#----------------------------------------
  essai_hypo         HYPO_ELAS1D
# ........................  loi de comportement HYPO_ELAS 1D ........................
# | coef compressibilite    | rapport contrainte | type de derivee objective utilisee |
# |      instantane         |      deformation   | pour le calcul de la contrainte    |
# |         Kc /3           |              f     |type_derivee  (facultatif)          |
#.....................................................................................
      Kc=     10     f=       0.15    type_derivee  -1
                fin_loi_HYPO_ELAS1D
```

Il est possible de modifier le comportement de base :

1. **Type de dérivée matériel** : le type de dérivée matérielle est par défaut celui de Jauman. Il est également possible de retenir une dérivée de Lie deux fois covariante ou une dérivée de Lie deux fois contravariante (se référer à la documentation théorique pour plus d'information sur ces dérivées). On utilise pour cela le mot clé optionnel : `type_deriveee` suivi de :
   - -1 : pour une dérivée de Jauman (valeur par défaut)
   - 0 = pour une dérivée deux fois covariante,
   - 1 = pour une dérivée deux fois contravariante.
2. **Kc calculé via une loi thermo physique** : il est possible d'indiquer que le module sera fourni par une loi thermo physique (celle associée au même élément

cf.lisloisthermophysiques). Pour ce faire on indique uniquement :
`Kc= Kc_avec_compressibilite_externe`
NB : aucune autre information concernant Kc ne doit être ensuite fournie.

3. **Dépendance explicite à la température** : chacun des 2 paramètres matériau Kc et f, peut-être remplacé par une fonction dépendante de la température (sauf si le cas `Kc_avec_compressibilite_externe` est actif). Pour ce faire on utilise un mot clé suivi d'un nom de courbe, ou la définition de la courbe directement comme avec les autres lois de comportement.
exemple pour Kc : `Kc= Kc_thermo_dependant_` courbe3
exemple pour f : `f= f_thermo_dependant_` courbe2
IMPORTANT : à chaque fois qu'il y a une thermo dépendance, il faut passer une ligne après la description de la grandeur thermo dépendante, mais pas de passage à la ligne si ce n'est pas thermo dépendant.

4. **Dépendance à une fonction nD** : Kc et f peuvent au choix dépendre d'une fonction nD. Dans ce cas Kc et f ne doivent pas avoir été déclarés thermo dépendant, mais on peut inclure une thermo dépendance dans la fonction nD. Exemple pour Kc :
`Kc= Kc_fonction_nD:` fonction1
Exemple pour f :
`f= f_fonction_nD:` fonction2

   La déclaration des fonctions suit la même logique que pour les courbes 1D

5. **Dépendance directe à $(\varepsilon : \varepsilon)$, la norme au carré de la déformation** :
   Il est possible de définir pour Kc un facteur multiplicatif fonction explicite du deuxième invariant au carré, de la déformation $= (\varepsilon : \varepsilon)$. Ceci permet d'obtenir une évolution finale non linéaire dont l'opérateur tangent prend en compte les variations de la déformation. Pour cela on indique le mot cle `Kc_IIeps_` suivi du nom d'une courbe" suivi d'un passage à la ligne.
   Exemple de déclaration d'un Kc thermo dépendant et fonction également du deuxième invariant :
   `Kc= Kc_thermo_dependant_` courbe2
   `Kc_IIeps_` courbe5

   Exemple de déclaration d'un Kc non thermo dépendant, mais fonction du deuxième invariant
   `Kc=` 10000 `Kc_IIeps_` courbe4

   NB :

   (a) on peut cumuler une dépendance à une fonction nD et une dépendance explicite multiplicative à $(\varepsilon : \varepsilon)$,

   (b) à la fin de la définition de `Kc_IIeps_` , il faut passer une ligne.

   D'une manière analogue on peut définir un facteur multiplicatif pour f fonction du deuxième invariant de la déformation $= (\varepsilon : \varepsilon)$ en indiquant le mot cle `f_IIeps_`

suivi du nom d'une courbe et d'un passage à la ligne.
Exemple de déclaration d'un f thermo dépendant et fonction également du deuxième invariant
`f= f_thermo_dependant_` courbe2
`f_IIeps_` courbe4

Exemple de déclaration d'un f non thermo dépendant, mais fonction du deuxième invariant :
`f=` 1200 `f_IIeps_` courbe4

NB :

(a) comme pour Kc, on peut cumuler une dépendance à une fonction nD et une dépendance explicite multiplicative à ($\boldsymbol{\varepsilon} : \boldsymbol{\varepsilon}$),

(b) à la fin de la définition de `f_IIeps_` , il faut passer une ligne.

NB : Pour toutes les définitions de courbe ou de fct nD il est aussi possible d'introduire directement la courbe à la place de son nom, comme pour toutes les autres lois.

6. **Restriction traction-compression** : À la suite du type de dérivée, on peut indiquer de manière optionnelle une restriction sur la contrainte calculée, via le mot clé :
`restriction_traction_compression_` suivi d'un scalaire tel que :
   (a) $= 0 -> $ pas de restriction (valeur par défaut)
   (b) = -1 : traction uniquement autorisée, la compression est mise à 0 ainsi que l'opérateur tangent,
   (c) = 1 : compression uniquement autorisée, la traction est mise à 0 ainsi que l'opérateur tangent.
7. **fin des paramètres** :
   La dernière ligne doit contenir uniquement le mot clé :
   `fin_loi_HYPO_ELAS1D`

51.14 Lois qui ne font rien !

Il s'agit d'une classe 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.207).

Remarque : Il faut noter que bien que la partie loi de comportement soit transparente pour le calcul de la raideur, tous les autres calculs associés : calcul des métriques et déformations, intégrales éventuelles ..., sont effectués. Pour que ces calculs soient correctement effectués, il faut que les informations autres que la loi de comportement, par exemple les données géométriques, soient correctement définies.

TABLE 207 – liste des différentes lois qui ne font rien, disponibles

identificateur	indications	ref du commentaire
LOI_RIEN1D	1D :	(51.14.1)
LOI_RIEN2D_C	2D :	(51.14.2)
LOI_RIEN2D_D	2D :	(51.14.3)
LOI_RIEN3D	3D :	(51.14.4)

51.14.1 LOI_RIEN1D

Identificateur d'une loi 1D qui ne fait rien. Cette loi ne nécessite aucun paramètre. La table (208) donne un exemple de cette loi.

TABLE 208 – 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 --------
```

51.14.2 LOI_RIEN2D_C

Identificateur d'une loi 2D en contrainte plane, qui ne fait rien. Cette loi ne nécessite aucun paramètre. La table (209) donne un exemple de cette loi.

TABLE 209 – Exemple de déclaration d'une loi 2D en contrainte plane ne faisant rien mécaniquement.

```
#-------- déclaration d'une loi qui ne fait rien mécaniquement  --------
 polymere       LOI_RIEN2D_C
#-------- fin de la loi --------
```

51.14.3 LOI_RIEN2D_D

Identificateur d'une loi 2D en déformation plane plane, qui ne fait rien. Cette loi ne nécessite aucun paramètre. La table (210) donne un exemple de cette loi.

TABLE 210 – Exemple de déclaration d'une loi 2D en déformation plane ne faisant rien mécaniquement.

```
#-------- déclaration d'une loi qui ne fait rien mécaniquement  --------
 polymere       LOI_RIEN2D_D
#-------- fin de la loi --------
```

51.14.4 LOI_RIEN3D

Identificateur d'une loi 3D qui ne fait rien. Cette loi ne nécessite aucun paramètre. La table (211) donne un exemple de cette loi.

TABLE 211 – 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 --------
```

51.15 Lois élastiques anisotropes

Remarque générale :
L'utilisation de lois anisotropes peut entraîner une non-symétrie importante de la raideur. L'utilisation d'un stockage générale symétrique (qui est le stockage par défaut) peut alors entraîner des difficultés de convergence (cf. https ://herezh.irdl.fr/issues/229). L'utilisation d'un stockage non symétrique est alors une solution à essayer : par exemple de type
```
BANDE_NON_SYMETRIQUE_LAPACK
```
avec les options
```
para_syteme_lineaire
TYPE_MATRICE BANDE_NON_SYMETRIQUE_LAPACK
SYMETRIE_MATRICE 0
```

(cf. 75).

51.15.1 ORTHOELA3D

Identificateur : `ORTHOELA3D`

Le comportement concerne une élasticité linéaire 3D de type Hooke, exprimée dans un repère particulier $\vec{O}'_a$. Avant déformation le repère est supposé orthonormé. Puis, le repère est entraîné par la matière. Les vecteurs $\vec{O}'_a$ restent normés par contre les angles entre vecteurs peuvent varier. On se reportera à la documentation théorique ([RIO, 2019]) pour une présentation détaillée des équations constitutives. On rappelle ici les éléments principaux qui permettent la compréhension de la mise en données.

Dans le repère $\vec{O}'_a$ (a= 1..3) les relations de comportement s'écrivent :

$$\begin{aligned} \varepsilon_{11} &= \frac{1}{E_1}(\sigma_{11} - \nu_{12}\sigma_{22} - \nu_{13}\sigma_{33}) \\ \varepsilon_{22} &= \frac{1}{E_2}(-\nu_{21}\sigma_{11} + \sigma_{22} - \nu_{23}\sigma_{33}) \\ \varepsilon_{33} &= \frac{1}{E_3}(-\nu_{31}\sigma_{11} - \nu_{32}\sigma_{22} + \sigma_{33}) \end{aligned} \tag{100}$$

et

$$\varepsilon_{ab} = \frac{1}{2.\ G_{ab}}\sigma_{ab} \quad \text{avec} \quad a \neq b \tag{101}$$

L'orthotropie classique nécessite les relations de symétrie :

$$\frac{-\nu_{12}}{E_1} = \frac{-\nu_{21}}{E_2}\ , \quad \frac{-\nu_{13}}{E_1} = \frac{-\nu_{31}}{E_3}\ , \quad \frac{-\nu_{23}}{E_2} = \frac{-\nu_{32}}{E_3} \tag{102}$$

Le comportement dépend donc de 9 paramètres et d'un repère particulier d'orthotropie. La table (212) donne un exemple de définition d'une loi basique à coefficients constants.

Remarques :

— L'ordre des paramètres doit être respecté,

— tous les modules E_a (a = 1..3) doivent-être strictement non nuls.

TABLE 212 – Exemple de déclaration d'une loi 3D orthotrope élastique entraînée. Cas basique utilisant des paramètres matériels constants.

```
  acier_ortho         ORTHOELA3D
# .......  loi de comportement orthotrope elastique 3D ........
#    9 parametres materiau: 3 modules, 3 coef de Poisson, 3 modules de cisaillement
E1= 100000 E2= 50000 E3= 20000 nu12= 0.2 nu13= 0.3 nu23= 0.1 \
G12= 20000 G13= 10000 G23= 2000
nom_repere_associe_ repere1
```

Pour que la loi puisse fonctionner, il faut que le repère "repere1" repéré par le mot clé `nom_repere_associe_` , ait été défini au préalable au niveau des éléments qui vont utiliser la loi d'orthotropie entraînée. On se reportera à (63) pour la mise en donnée de cette définition de repère.

Il est possible de remplacer un ou plusieurs coefficients par une fonction nD (cf. 81.2). On peut ainsi définir une dépendance des coefficients à des grandeurs locales telles que la position ou une autre variable interpolée comme la température si elle est par ailleurs définie. On peut également définir une dépendance à des grandeurs globales comme par exemple le temps ou toutes autres grandeurs existantes au moment de l'appel à la loi.

Pour définir par exemple une fonction nD pour le module E1, on utilisera la syntaxe suivante : " `E1= E1_fonction_nD:` " suivie soit du nom d'une fonction déjà définie soit de la définition directe d'une fonction nD. La méthodologie est identique pour tous les paramètres matériaux via la syntaxe : "nom du paramètre" + `_fonction_nD:` suivie soit du nom d'une fonction déjà définie soit de la définition directe d'une fonction nD. La table (213) donne deux exemples de paramètre défini à partir d'une fonction nD.

Remarques :

— Après chaque définition d'une fonction nD, on doit changer de ligne pour les prochains paramètres.

Après avoir défini les paramètres de la loi, il est possible de préciser et/ou modifier le fonctionnement de la loi via des paramètres de réglage qui sont encapsulés entre deux mots clés : `avec_parametres_de_reglage_` et `fin_parametres_reglage_` . L'ordre d'apparition des paramètres n'a pas d'importance ni le fait qu'ils soient sur une ou plusieurs lignes. La table (213) montre un exemple d'utilisation de paramètres de réglage. La liste des paramètres disponibles et leurs significations sont les suivantes :

— Par défaut, la base initiale d'orthotropie est transportée dans l'état courant via une méthode de "transport de type contravariant" (cf.[RIO, 2019]). Il est possible d'indiquer un transport de type covariant à l'aide du mot clé : `type_transport_` suivi de 0 ou 1 :

0 : pour un transport de type contravariant (valeur par défaut)

1 : pour un transport de type covariant.

— Il est possible de restreindre la loi de comportement au seul comportement sphérique ou au seul comportement déviatorique. Pour cela on utilise au choix un des mots clés :

TABLE 213 – Exemple de déclaration d'une loi 3D orthotrope élastique entraînée. Cas où certains paramètres matériels sont définis à partir de fonctions nD, avec utilisation de certains paramètres de réglage.

```
  acier_ortho         ORTHOELA3D
# .......  loi de comportement orthotrope elastique 3D ........
#    9 parametres materiau: 3 modules, 3 coef de Poisson, 3 modules de cisaillement
E1= 100000 E2= E2_fonction_nD: f2_temperature
E3= E3_fonction_nD: FONCTION_EXPRESSION_LITTERALE_nD
           deb_list_var_ TEMP  fin_list_var_  "
           fct= (100726-101325)/50*TEMP + 101325  "
        fin_parametres_fonction_expression_litterale_
nu12= 0.2 nu13= 0.3 nu23= 0.1 \
G12= 20000 G13= 10000 G23= 2000
nom_repere_associe_ repere1 avec_parametres_de_reglage_
    sortie_post_       1
    type_transport_    0
    permet_affichage_  5
    verification_convexite_ 1
  fin_parametres_reglage_
```

seule_deviatorique

seule_spherique

Par défaut l'ensemble sphérique + déviatorique est calculé. Lorsque que le comportement est restreint, le comportement tangent $\partial\sigma^{ij}/\partial ddl$ ou $\partial\sigma^{ij}/\partial\varepsilon_{kl}$, est également restreint.

— Il est souvent préférable d'utiliser un comportement tel que le potentiel élastique associé soit une fonction convexe des déformations. Mais ce n'est pas obligatoire, aussi par défaut un comportement non convexe est acceptable dans la mise en données. En cas de non-convexité, cela peut entraîner des difficultés de convergence dans un équilibre statique ou dans une formulation implicite en dynamique, lorsque l'on utilise un algorithme de type Newton. Cela peut également conduire à une solution qui n'est pas unique. Si on désire s'assurer que le potentiel est convexe pendant le calcul, en particulier lorsque l'on utilise des paramètres matériaux non constants, on peut indiquer le mot clé : verification_convexite_ suivi de 1 ou 0 indiquant que l'on veut ou non une vérification (cf.[RIO, 2019]). À noter que si une non-convexité est détectée, il y a seulement un affichage à l'écran d'un message. La modification éventuelle de l'évolution des paramètres matériaux reste à la charge de l'utilisateur.

— Par défaut le niveau d'affichage est géré par le paramètre global niveau_commentaire défini en début du fichier de mise en données .info . Il est possible de choisir un niveau différent de commentaire, localement pour la loi, via le mot clé permet_affichage_ suivi d'un entier entre 0 et 10, indiquant le niveau local voulu.

— La compressibilité est calculée à partir du log de la variation de volume constatée (cf. théorie Herezh++) et de la trace du tenseur des contraintes. Lorsque la variation

de volume est trop faible c'est la compressibilité initiale qui est utilisée. Lorsque la compressibilité est inférieure a 1/100 (valeur par défaut) de la compressibilité initiale, on considère que le calcul de cette compressibilité n'est pas correcte, et on utilise la compressibilité initiale. Le coefficient 1/100 est arbitraire, on peut le changer avec le mot clé : `ratio_inf_module_compressibilite_` <un reel> ce qui permet de moduler la borne inférieure de la compressibilité.

— par défaut, les grandeurs disponibles en post-traitement, relativement à la loi, sont la position courante du repère transporté. Il est possible également d'accéder d'une part aux coordonnées de la déformation courante exprimée dans le repère d'orthotropie transporté, et d'autre part aux coordonnées de la contrainte courante également exprimée dans ce repère. Cependant, ce n'est pas une option par défaut, étant donné que ce n'est pas forcément utile en exploitation courante, et que cela demande un espace de stockage supplémentaire. Aussi si l'on veut avoir accès à ces composantes il faut au préalable du calcul, indiquer le mot clé `sortie_post_` suivit de 1 (0 est la valeur par défaut).

51.15.2 Élastique orthotrope entraîné en contraintes planes

Identificateur : `ORTHOELA2D_C`

Le comportement concerne une élasticité linéaire de type Hooke dans un état de contrainte plane, exprimée dans un repère particulier $\vec{O}'_a$. Les deux premiers vecteurs du repère sont supposés être dans le plan de l'état de contrainte plane.

Avant déformation le repère est supposé orthonormé, en particulier la direction 3 est supposée être normale au plan de contrainte plane. Puis, le repère est entraîné par la matière. Les vecteurs $\vec{O}'_a$ restent normés par contre les angles entre les deux premiers vecteurs peuvent varier. On se reportera à la documentation théorique ([RIO, 2019]) pour une présentation détaillée des équations constitutives. On rappelle ici les éléments principaux qui permettent la compréhension de la mise en données.

Dans le repère $\vec{O}'_a$ (a= 1..3) les relations de comportement s'écrivent :

$$\begin{aligned} \varepsilon_{11} &= \frac{1}{E_1}(\sigma_{11} - \nu_{12}\sigma_{22}) \\ \varepsilon_{22} &= \frac{1}{E_2}(-\nu_{21}\sigma_{11} + \sigma_{22}) \\ \varepsilon_{33} &= \frac{1}{E_3}(-\nu_{31}\sigma_{11} - \nu_{32}\sigma_{22}) \end{aligned} \tag{103}$$

et

$$\varepsilon_{12} = \frac{1}{2.\ G_{12}}\sigma_{12} \tag{104}$$

L'orthotropie classique nécessite les relations de symétrie :

$$\frac{-\nu_{12}}{E_1} = \frac{-\nu_{21}}{E_2}\ ,\quad \frac{-\nu_{13}}{E_1} = \frac{-\nu_{31}}{E_3}\ ,\quad \frac{-\nu_{23}}{E_2} = \frac{-\nu_{32}}{E_3} \tag{105}$$

Le comportement dépend donc de 7 paramètres et d'un repère particulier d'orthotropie. La table (214) donne un exemple de définition d'une loi basique à coefficients constants.

Remarques :

— L'ordre des paramètres doit être respecté,

— tous les modules E_a (a = 1..3) doivent-être strictement non nuls.

TABLE 214 – Exemple de déclaration d'une loi 2D contrainte plane orthotrope élastique entraînée. Cas basique utilisant des paramètres matériels constants.

```
  acier_orthoCP         ORTHOELA2D_C
# .......  loi de comportement orthotrope elastique 2D CP ........
#    7 parametres materiau: 3 modules, 3 coef de Poisson, 1 module de cisaillement
E1= 100000 E2= 50000 E3= 20000 nu12= 0.2 nu13= 0.3 nu23= 0.1 \
G12= 20000
nom_repere_associe_ repere1
```

Pour que la loi puisse fonctionner, il faut que le repère "repere1" repéré par le mot clé `nom_repere_associe_` , ait été défini au préalable au niveau des éléments qui vont utiliser la loi d'orthotropie entraînée. On se reportera à (63) pour la mise en donnée de cette définition de repère.

Comme pour le cas 3D, il est possible de remplacer un ou plusieurs coefficients par une fonction nD (cf. 81.2). On peut ainsi définir une dépendance des coefficients à des grandeurs locales telles que la position ou une autre variable interpolée comme la température si elle est par ailleurs définie. On peut également définir une dépendance à des grandeurs globales comme par exemple le temps ou toutes autres grandeurs existantes au moment de l'appel à la loi.

Pour définir par exemple une fonction nD pour le module E1, on utilisera la syntaxe suivante : " `E1= E1_fonction_nD:` " suivie soit du nom d'une fonction déjà définie soit de la définition directe d'une fonction nD. La méthodologie est identique pour tous les paramètres matériaux via la syntaxe : "nom du paramètre" + `_fonction_nD:` suivie soit du nom d'une fonction déjà définie soit de la définition directe d'une fonction nD. La table (215) donne deux exemples de paramètre défini à partir d'une fonction nD.

Remarques :

— Après chaque définition d'une fonction nD, on doit changer de ligne pour les prochains paramètres.

Après avoir défini les paramètres de la loi, il est possible de préciser et/ou modifier le fonctionnement de la loi via des paramètres de réglage qui sont encapsulés entre deux mots clés : `avec_parametres_de_reglage_` et `fin_parametres_reglage_` . L'ordre d'apparition des paramètres n'a pas d'importance ni le fait qu'ils soient sur une ou plusieurs lignes. La table (213) montre un exemple d'utilisation de paramètres de réglage. La liste des paramètres disponibles et leurs significations sont les suivantes :

— Par défaut, la base initiale d'orthotropie est transportée dans l'état courant via une méthode de "transport de type contravariant" (cf.[RIO, 2019]). Il est possible d'indiquer un transport de type covariant à l'aide du mot clé : `type_transport_` suivi de 0 ou 1 :

0 : pour un transport de type contravariant (valeur par défaut)

TABLE 215 – Exemple de déclaration d'une loi 2D CP orthotrope élastique entraînée. Cas où certains paramètres matériels sont définis à partir de fonctions nD, avec utilisation de certains paramètres de réglage.

```
  acier_orthoCP         ORTHOELA2D_C
# .......  loi de comportement orthotrope elastique 2D CP ........
#    7 parametres materiau: 3 modules, 3 coef de Poisson, 1 module de cisaillement
E1= 100000 E2= E2_fonction_nD: f2_temperature
E3= E3_fonction_nD: FONCTION_EXPRESSION_LITTERALE_nD
           deb_list_var_ TEMP  fin_list_var_  "
           fct= (100726-101325)/50*TEMP + 101325  "
        fin_parametres_fonction_expression_litterale_
nu12= 0.2 nu13= 0.3 nu23= 0.1 \
G12= 20000
nom_repere_associe_ repere1 avec_parametres_de_reglage_
    sortie_post_       1
    type_transport_    0
    permet_affichage_  5
    verification_convexite_ 1
  fin_parametres_reglage_
```

1 : pour un transport de type covariant.

— Il est possible de restreindre la loi de comportement au seul comportement sphérique ou au seul comportement déviatorique. Pour cela on utilise au choix un des mots clés :

`seule_deviatorique`

`seule_spherique`

Par défaut l'ensemble sphérique + déviatorique est calculé. Lorsque que le comportement est restreint, le comportement tangent $\partial\sigma^{ij}/\partial ddl$ ou $\partial\sigma^{ij}/\partial\varepsilon_{kl}$, est également restreint.

— Il est souvent préférable d'utiliser un comportement tel que le potentiel élastique associé soit une fonction convexe des déformations. Mais ce n'est pas obligatoire, aussi par défaut un comportement non convexe est acceptable dans la mise en données. En cas de non-convexité, cela peut entraîner des difficultés de convergence dans un équilibre statique ou dans une formulation implicite en dynamique, lorsque l'on utilise un algorithme de type Newton. Cela peut également conduire à une solution qui n'est pas unique. Si on désire s'assurer que le potentiel est convexe pendant le calcul, en particulier lorsque l'on utilise des paramètres matériaux non constants, on peut indiquer le mot clé : `verification_convexite_` suivi de 1 ou 0 indiquant que l'on veut ou non une vérification (cf.[RIO, 2019]). À noter que si une non-convexité est détectée, il y a seulement un affichage à l'écran d'un message. La modification éventuelle de l'évolution des paramètres matériaux reste à la charge de l'utilisateur.

— Par défaut le niveau d'affichage est géré par le paramètre global `niveau_commentaire` défini en début du fichier de mise en données `.info` . Il est possible de choi-

sir un niveau différent de commentaire, localement pour la loi, via le mot clé permet_affichage_ suivi d'un entier entre 0 et 10, indiquant le niveau local voulu.

— par défaut, les grandeurs disponibles en post-traitement, relativement à la loi, sont la position courante du repère transporté. Il est possible également d'accéder d'une part aux coordonnées de la déformation courante exprimée dans le repère d'orthotropie transporté, et d'autre part aux coordonnées de la contrainte courante également exprimée dans ce repère. Cependant, ce n'est pas une option par défaut, étant donné que ce n'est pas forcément utile en exploitation courante, et que cela demande un espace de stockage supplémentaire. Aussi si l'on veut avoir accès à ces composantes il faut au préalable du calcul, indiquer le mot clé sortie_post_ suivit de 1 (0 est la valeur par défaut).

51.16 Lois hypo-élastiques anisotropes

51.16.1 HYPO_ORTHO3D

Identificateur : HYPO_ORTHO3D

La construction de la loi s'appuie sur les concepts :

— d'hypo-élasticité qui consiste à calculer la variation de la contrainte à partir de la variation de la déformation. Numériquement cela conduit sur un pas de temps, à calculer une contrainte finale à partir de la contrainte initiale et de son accroissement fonction de $\mathbf{\Delta}\varepsilon$ sur le pas de temps.

— de transport de grandeurs tensorielles : ici il s'agit du transport du tenseur contrainte calculé à t et devant être utilisé à $t + \Delta t$.

— d'anisotropie induite par un comportement initialement orthotrope dans un repère particulier, qui ensuite est entraîné par la matière.

La construction de la loi suit une méthodologie analogue à celle de la loi de "Hooke initialement orthotrope puis entraînée" .

On se reportera à la documentation ([RIO, 2019]) pour une présentation détaillée des aspects théoriques de la loi.

L'accroissement de contrainte sur un pas de temps, s'exprime dans un repère particulier $\vec{O}'_a$: le repère d'anisotropie. Avant déformation le repère est supposé orthonormé. Puis, le repère est entraîné par la matière. Les vecteurs $\vec{O}'_a$ restent normés par contre les angles entre vecteurs peuvent varier. On se reportera à la documentation théorique ([RIO, 2019]) pour une présentation détaillée des équations constitutives. On rappelle ici les éléments principaux qui permettent la compréhension de la mise en données.

Dans le repère $\vec{O}'_a$ (a= 1..3) les relations de comportement entre accroissements de contrainte et déformation, s'écrivent :

$$\begin{aligned}
\Delta_t^{t+\Delta t}\varepsilon^{11} &= \frac{1}{E_1}(\Delta_t^{t+\Delta t}\sigma^{11} - \nu_{12}\Delta_t^{t+\Delta t}\sigma^{22} - \nu_{13}\Delta_t^{t+\Delta t}\sigma^{33}) \\
\Delta_t^{t+\Delta t}\varepsilon^{22} &= \frac{1}{E_2}(-\nu_{21}\Delta_t^{t+\Delta t}\sigma^{11} + \Delta_t^{t+\Delta t}\sigma^{22} - \nu_{23}\Delta_t^{t+\Delta t}\sigma^{33}) \\
\Delta_t^{t+\Delta t}\varepsilon^{33} &= \frac{1}{E_3}(-\nu_{31}\Delta_t^{t+\Delta t}\sigma^{11} - \nu_{32}\Delta_t^{t+\Delta t}\sigma^{22} + \Delta_t^{t+\Delta t}\sigma^{33})
\end{aligned} \tag{106}$$

et

$$\Delta_t^{t+\Delta t}\varepsilon^{ab} = \frac{1}{2.\ G_{ab}}\Delta_t^{t+\Delta t}\sigma^{ab} \quad \text{avec} \quad a \neq b \tag{107}$$

En inversant ses relations on obtient ainsi l'accroissement $\Delta_t^{t+\Delta t}\sigma_{ab}$ en fonction de $\Delta_t^{t+\Delta t}\varepsilon^{cd}$, (a,b,c,d)= 1..3

Toutes ces composantes sont exprimées dans le repère d'anisotropie. Par changement de base, les composantes $\Delta_t^{t+\Delta t}\sigma^{ij}$ de l'accroissement des contraintes sont ensuite exprimées dans le repère de travail $\hat{\vec{g}}_i$ et la contrainte finale s'obtient alors via :

$$\sigma^{ij}(t+\Delta t) = \Delta_t^{t+\Delta t}\sigma^{ij} + transport_t^{t+\Delta t}\sigma^{ij}(t) \tag{108}$$

où $\sigma^{ij}(t)$ sont les contraintes calculées à la fin de l'incrément précédent.

Trois types de transport sont pris en compte dans Herezh++ : le transport deux fois covariant qui est cohérent avec la dérivée de Rivlin, le transport deux fois contravariants qui est cohérent avec la dérivée d'Oldroyd et le transport mixte qui est cohérent avec la dérivée de Jauman.

Remarques

— Les paramètres sont notés par analogie avec la loi de Hooke anisotrope. Néanmoins il faut noter qu'ils représentent ici un comportement tangent, contrairement au cas de la loi de Hooke où ils représentent un comportement sécant.

— Dans le cas où ces paramètres sont constants on obtient un comportement élastique réversible. Par contre lorsque ces paramètres varient, ce qui est un des intérêts d'une loi hypo-élastique, la réponse obtenue n'est a priori pas réversible. À l'aide de paramètres ainsi variables, il est possible de représenter des comportements très divers, la difficulté étant alors d'identifier correctement l'évolution de ces paramètres matériels.

— La variance des coordonnées n'a pas d'importance ici initialement, car le repère est orthonormé. Par contre, une fois la structure déformée il est tenu compte de l'évolution du repère.

Toujours par analogie avec la loi de Hooke anisotrope, on suppose que l'accroissement d'énergie de déformation sur un pas de temps, est convexe, ce qui conduit aux relations de symétrie :

$$\frac{-\nu_{12}}{E_1} = \frac{-\nu_{21}}{E_2}\ , \quad \frac{-\nu_{13}}{E_1} = \frac{-\nu_{31}}{E_3}\ , \quad \frac{-\nu_{23}}{E_2} = \frac{-\nu_{32}}{E_3} \tag{109}$$

Au final, le comportement dépend donc de 9 paramètres et d'un repère particulier d'orthotropie.

La table (216) donne un exemple de définition d'une loi basique à coefficients constants.

Remarques :

— L'ordre des paramètres doit être respecté,

— tous les modules E_a (a = 1..3) doivent-être strictement non nuls.

Pour que la loi puisse fonctionner, il faut que le repère "repere1" repéré par le mot clé `nom_repere_associe_` , ait été défini au préalable au niveau des éléments qui vont utiliser la loi hypo-elastique orthotrope entraînée. On se reportera à (63) pour la mise en donnée de cette définition de repère.

TABLE 216 – Exemple de déclaration d'une loi 3D hypo-élastique orthotrope entraînée. Cas basique utilisant des paramètres matériels constants.

```
  acier_hypo_ortho         HYPO_ORTHO3D
# .......  loi de comportement hypo elastique orthotrope 3D ........
#    9 parametres materiau: 3 modules tangent de traction, 3 coef type Poisson
#  , 3 modules type cisaillement tangent
#  et un nom de repere associe
    E1= 100000 E2= 50000 E3= 20000 \
    nu12= 0.2 nu13= 0.3 nu23= 0.1\
    G12= 20000 G13= 10000 G23= 2000
nom_repere_associe_ repere1
```

Il est possible de remplacer un ou plusieurs coefficients par une fonction nD (cf. 81.2). On peut ainsi définir une dépendance des coefficients à des grandeurs locales telles que la position ou une autre variable interpolée comme la température si elle est par ailleurs définie. On peut également définir une dépendance à des grandeurs globales comme par exemple le temps ou toutes autres grandeurs existantes au moment de l'appel à la loi.

Pour définir par exemple une fonction nD pour le module E1, on utilisera la syntaxe suivante : " `E1= E1_fonction_nD:` " suivie soit du nom d'une fonction déjà définie soit de la définition directe d'une fonction nD. La méthodologie est identique pour tous les paramètres matériaux via la syntaxe : "nom du paramètre" + `_fonction_nD:` suivie soit du nom d'une fonction déjà définie soit de la définition directe d'une fonction nD. La table (213) donne deux exemples de paramètre défini à partir d'une fonction nD.

Remarques :

— Après chaque définition d'une fonction nD, on doit changer de ligne pour les prochains paramètres.

Après avoir défini les paramètres de la loi, il est possible de préciser et/ou modifier le fonctionnement de la loi via des paramètres de réglage qui sont encapsulés entre deux mots clés : `avec_parametres_de_reglage_` et `fin_parametres_reglage_` . L'ordre d'apparition des paramètres n'a pas d'importance ni le fait qu'ils soient sur une ou plusieurs lignes. La table (217) montre un exemple d'utilisation de paramètres de réglage. La liste des paramètres disponibles et leurs significations sont les suivantes :

— Par défaut, la base initiale d'orthotropie est transportée dans l'état courant via une méthode de "transport de type contravariant" (cf.[RIO, 2019]). Il est possible d'indiquer un transport de type covariant à l'aide du mot clé : `type_transport_` suivi de 0 ou 1 :

  0 : pour un transport de type contravariant (valeur par défaut)

  1 : pour un transport de type covariant (pour l'instant, pas opérationnel : V 9.911).

— Il est possible de restreindre la loi de comportement au seul comportement sphérique ou au seul comportement déviatorique. Pour cela on utilise au choix un des mots clés :

TABLE 217 – Exemple de déclaration d'une loi 3D hypo-élastique orthotrope entraînée. Cas où certains paramètres matériels sont définis à partir de fonctions nD, avec utilisation de certains paramètres de réglage.

```
  acier_hypo_ortho         HYPO_ORTHO3D
# .......  loi de comportement hypo elastique orthotrope 3D ........
#    9 parametres materiau: 3 modules tangent de traction, 3 coef type Poisson
#  , 3 modules type cisaillement tangent
#  et un nom de repere associe
E1= 100000 E2= E2_fonction_nD: f2_temperature
E3= E3_fonction_nD: FONCTION_EXPRESSION_LITTERALE_nD
           deb_list_var_ TEMP  fin_list_var_  "
           fct= (100726-101325)/50*TEMP + 101325  "
        fin_parametres_fonction_expression_litterale_
nu12= 0.2 nu13= 0.3 nu23= 0.1 \
G12= 20000 G13= 10000 G23= 2000
nom_repere_associe_ repere1 avec_parametres_de_reglage_
    sortie_post_       1
    type_transport_    0
    permet_affichage_  3
  fin_parametres_reglage_
```

seule_deviatorique

seule_spherique

Par défaut l'ensemble sphérique + déviatorique est calculé. Lorsque que le comportement est restreint, le comportement tangent $\partial\sigma^{ij}/\partial ddl$ ou $\partial\sigma^{ij}/\partial\varepsilon_{kl}$, est également restreint.

— Par défaut le niveau d'affichage est géré par le paramètre global niveau_commentaire défini en début du fichier de mise en données .info . Il est possible de choisir un niveau différent de commentaire, localement pour la loi, via le mot clé permet_affichage_ suivi d'un entier entre 0 et 10, indiquant le niveau local voulu.

— par défaut, les grandeurs disponibles en post-traitement, relativement à la loi, sont la position courante du repère transporté. Il est possible également d'accéder d'une part aux coordonnées de la déformation courante et de l'incrément de déformation, exprimées dans le repère d'orthotropie transporté, et d'autre part aux coordonnées de la contrainte et de l'incrément de contrainte, courantes également exprimées dans ce repère. Cependant, ce n'est pas une option par défaut, étant donné que ce n'est pas forcément utile en exploitation courante, et que cela demande un espace de stockage supplémentaire. Aussi si l'on veut avoir accès à ces composantes il faut au préalable du calcul, indiquer le mot clé sortie_post_ suivi de 1 (0 est la valeur par défaut).

51.17 Lois anisotropes obtenues par une projection d'une loi existante

Consulter la remarque générale relative à la symétrie de la raideur résultante de lois anisotrope 51.15

51.17.1 PROJECTION_ANISOTROPE_3D

Identificateur : PROJECTION_ANISOTROPE_3D

On se reportera à la documentation théorique ([RIO, 2019]) pour une présentation détaillée des équations constitutives.

La loi s'appuie sur les ingrédients suivants :

— un type de projection

— un repère d'anisotropie

— une loi 3D isotrope déjà existante

— éventuellement des paramètres de réglage

Actuellement le type de projection implanté est PROJ_ORTHO . Ce type correspond à une projection dans l'espace des contraintes. On rappelle ici les éléments principaux qui permettent la compréhension de la mise en données.

On suppose défini un repère particulier initialement orthonormé $\vec{O'}_a$ qui représente le repère d'anisotropie. Dans ce repère est défini un tenseur d'ordre 4

$$\boldsymbol{H} = H^{ab}_{..cd}\, \vec{O'}_a \otimes \vec{O'}_b \otimes \vec{O'}^c \otimes \vec{O'}^d \tag{110}$$

Les composantes de $\boldsymbol{H}$ sont toutes nulles sauf les composantes : $H^{aa}_{..aa}$ a=1,2,3 et $H^{ab}_{..ab}$ $a \neq b$ et a,b=1,2,3. Pour des raisons de symétrie du tenseur final de contrainte on a : $H^{ab}_{..ab} = H^{ba}_{..ba} = H^{ab}_{..ba} = H^{ba}_{..ab}$. Les composantes de $\boldsymbol{H}$ dépendent donc uniquement de 6 scalaires.

Les directions principales du tenseur $\boldsymbol{H}$ dans l'espace tensoriel d'ordre 2, sont ainsi les tenseurs de base : $\vec{O'}_a \otimes \vec{O'}_b$ a,b= 1..3.

Soit une loi de comportement existante isotrope qui permet de calculer un tenseur de contrainte de référence $\boldsymbol{\sigma}_{(ref)}$. On en déduit le tenseur final des contraintes :

$$\boldsymbol{\sigma}(\boldsymbol{\varepsilon}, \boldsymbol{D}, ...) = \boldsymbol{H}..\boldsymbol{\sigma}_{(ref)}(\boldsymbol{\varepsilon}, \boldsymbol{D}, ...) \tag{111}$$

Dans le cadre d'une transformation finie, il faut adjoindre à cette équation constitutive une équation d'évolution pour $\boldsymbol{H}$. Dans une première étape l'idée est de considérer un transport matériel $\boldsymbol{H}$ qui s'effectue directement à partir du transport de la base $\vec{O'}_a$ avec les particularités suivantes :

— le transport est matériel via l'utilisation d'une base matérielle naturelle $\vec{g}_i$ et sa base duale,

— la direction des vecteurs $\vec{O'}_a$ ou $\vec{O'}^a$ suivant le choix de l'utilisateur) est convectée

— par contre les vecteurs de base transportés d'anisotropie, restent normés

Ainsi le tenseur $\boldsymbol{H}$ demeure constant dans un repère matériel entraîné dans le cas d'un mouvement solide. Dans le cas d'une déformation finie, il est sensible aux variations d'angles du repère $\vec{O}'_a$. Il n'est pas sensible aux variations de longueurs dans les directions entraînées de $\vec{O}'_a$.

Une conséquence pratique est que le repère d'anisotropie, initialement orthonormé, évolue vers un repère normé mais non orthogonal dans le cas d'une déformation finie quelconque. Le tenseur $\boldsymbol{H}$ intervient sous forme d'une amplification différenciée de chaque composante du tenseur de référence $\boldsymbol{\sigma}_{(ref)}$. La table (218) présente un exemple de mise en données. Dans cet exemple :

- les paramètres Aa , a=1..3 , correspondent aux composantes $H^{aa}_{..aa}$ et les paramètres Bab correspondent aux composantes $H^{ab}_{..ab}$, du tenseur $\boldsymbol{H}$ dans le repère anisotrope
- exprimé dans le repère $\vec{O}'_a$, les composantes du tenseur de contrainte sont dans cet exemple telles que les trois cisaillements demeurent identiques et les 3 contraintes de traction/compression sont dilatées selon $\sigma^{11} = 0.5\ \sigma^{11}_{(ref)}$, $\sigma^{22} = 2.\ \sigma^{2}_{(ref)}$, $\sigma^{33} = 3\ \sigma^{33}_{(ref)}$
- "repere1" est le repère d'anisotropie. Il doit avoir été défini préalablement (cf.63)

TABLE 218 – Exemple de déclaration d'une loi de projection anisotrope 3D qui s'appuie sur un comportement initialement élastique.

```
#----------------------------------------
# Nom Materiau    |      Type loi         |
#----------------------------------------
  acier_anisotrope         PROJECTION_ANISOTROPE_3D
# .......  loi de comportement Projection anisotrope 3D ........
    TYPE_PROJ_ANISO_  PROJ_ORTHO
    A1= 0.5 A2= 2. A3= 3. B12= 1. B13= 1. B23= 1.
    nom_repere_associe_ repere1
    ISOELAS
       2000  0.3
    fin_loi_interne # --- fin def loi interne
  fin_loi_projection_anisotrope # --- fin de la loi
```

Remarques : L'ordre des paramètres dans la mise en données, doit être respecté.

Pour que la loi puisse fonctionner, il faut que le repère "repere1" repéré par le mot clé `nom_repere_associe_` , ait été défini au préalable au niveau des éléments qui vont utiliser la loi. On se reportera à (63) pour la mise en donnée de cette définition de repère.

Il est possible de remplacer un ou plusieurs coefficients par une fonction nD (cf. 81.2). On peut ainsi définir une dépendance des coefficients à des grandeurs locales telles que la position ou une autre variable interpolée comme la température si elle est par ailleurs définie. On peut également définir une dépendance à des grandeurs globales comme par exemple le temps ou toutes autres grandeurs existantes au moment de l'appel à la loi. Bien noter que ces coefficients ainsi obtenus peuvent-être sensibles aux degrés de liberté du problème au travers des variables utilisées par les fonction nD définies par l'utilisateur.

Cette sensibilité ne peut pas être prise en compte dans le calcul de l'opérateur tangent final $\frac{\partial \sigma^{ij}}{\partial ddl}$ ou $\frac{\partial \sigma^{ij}}{\partial \varepsilon_{kl}}$. En conséquence, cela impactera la vitesse de convergence dans le cas d'évolution importante des coefficients d'anisotropie **sur un incrément de chargement**.

Pour définir par exemple une fonction nD pour le paramètre A1, on utilisera la syntaxe suivante : " `A1= A1_fonction_nD:` " suivie soit du nom d'une fonction déjà définie soit de la définition directe d'une fonction nD. La méthodologie est identique pour tous les paramètres matériaux via la syntaxe : "nom du paramètre" + `_fonction_nD:` suivie soit du nom d'une fonction déjà définie soit de la définition directe d'une fonction nD. La table (219) donne deux exemples de paramètre défini à partir d'une fonction nD.

Remarques :

— Après chaque définition d'une fonction nD, on doit changer de ligne pour les prochains paramètres.

TABLE 219 – Exemple de déclaration d'une loi de projection anisotrope 3D à coefficients variables. Cas où certains paramètres matériels sont définis à partir de fonctions nD, avec utilisation de certains paramètres de réglage.

```
#-----------------------------------------
# Nom Materiau    |      Type loi         |
#-----------------------------------------
  acier_anisotrope         PROJECTION_ANISOTROPE_3D
# .......  loi de comportement Projection anisotrope 3D ........
    TYPE_PROJ_ANISO_  PROJ_ORTHO
    A1= fonction_nD: A1_temperature
    A2= fonction_nD: A2_temperature
    A3= 3. B12= 1. B13= 1. B23= 1.
    nom_repere_associe_ repere1
    ISOELAS
       2000  0.3
       permet_affichage_ 5 # = le niveau utilise pour la loi elastique
    fin_loi_interne # --- fin def loi interne
    avec_parametres_de_reglage_
       sortie_post_       1
       type_transport_    0
    fin_parametres_reglage_
     permet_affichage_ 3  # = le niveau utilise pour la loi de projection
  fin_loi_projection_anisotrope # --- fin de la loi
```

Après avoir défini les paramètres de la loi, il est possible de préciser et/ou modifier le fonctionnement de la loi via des paramètres de réglage qui sont encapsulés entre deux mots clés : `avec_parametres_de_reglage_` et `fin_parametres_reglage_` . L'ordre d'apparition des paramètres n'a pas d'importance ni le fait qu'ils soient sur une ou plusieurs lignes. La table (219) montre un exemple d'utilisation de paramètres de réglage. La liste des paramètres disponibles et leurs significations sont les suivantes :

— Par défaut, la base initiale d'anisotropie est transportée dans l'état courant via une méthode de "transport de type contravariant" (cf.[RIO, 2019]). Il est possible

d'indiquer un transport de type covariant à l'aide du mot clé : `type_transport_` suivi de 0 ou 1 :

0 : pour un transport de type contravariant (valeur par défaut)

1 : pour un transport de type covariant.

cependant dans la version 6.901 seule le transport par défaut est opérationnel

— Il est possible de restreindre la loi de comportement au seul comportement sphérique ou au seul comportement déviatorique. Pour cela on utilise au choix un des mots clés :

`seule_deviatorique`

`seule_spherique`

Par défaut l'ensemble sphérique + déviatorique est calculé. Lorsque le comportement est restreint, le comportement tangent $\partial\sigma^{ij}/\partial ddl$ ou $\partial\sigma^{ij}/\partial\varepsilon_{kl}$, est également restreint.

— Par défaut le niveau d'affichage est géré par le paramètre global `niveau_commentaire` défini en début du fichier de mise en données `.info` . Il est possible de choisir un niveau différent de commentaire, localement pour la loi, via le mot clé `permet_affichage_` suivi d'un entier entre 0 et 10, indiquant le niveau local voulu.

— par défaut, les grandeurs disponibles en post-traitement, relativement à la loi, sont la position courante du repère transporté. Il est possible également d'accéder aux coordonnées de la contrainte courante également exprimée dans ce repère. Cependant, ce n'est pas une option par défaut, étant donné que ce n'est pas forcément utile en exploitation courante, et que cela demande un espace de stockage supplémentaire. Aussi si l'on veut avoir accès à ces composantes il faut au préalable du calcul, indiquer le mot clé `sortie_post_` suivi de 1 (0 est la valeur par défaut).

51.18 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 (17) pour avoir plus d'information concernant la mise en route d'un processus Herezh fournissant la loi Umat, et également pour l'introduction d'une dépendance à la température. La loi peut également être défini par un programme externe en C ou autre langage qui puisse appeler des routines C (exemple : scilab, matlab, etc.).

La table (220) donne un exemple de déclaration d'utilisation de loi externe par Herezh non thermodépendante. Dans cet exemple le nom de la loi est "acier" la catégorie par le mot clé " `CAT_MECANIQUE` " puis on indique la dimension. La déclaration est terminée par le mot clé : " `fin_loi_Umat` ". Le nom "acier" sera utilisé pour vérifier la cohérence du nom du matériau entre le programme appelant (celui qui utilise une Umat externe) et le programme qui produit l'Umat.

Dans le cas d'une loi thermodépendante, il faut remplacer le mot clé " `CAT_MECANIQUE` " par " `CAT_THERMO_MECANIQUE` ", ceci permet d'indiquer à Herezh que la loi utilisera la température. (cette méthodologie est susceptible d'évoluer dans l'avenir)

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 (221).

Vient 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). 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 (222) 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 quelconque, 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 les variables utilisées par ce programme. Les tables suivantes présentent la structuration des données, qui est utilisée pour la sérialisation (transformation dans les pipes en octets banalisés) en entrée et sortie des programmes C qui gèrent le dialogue . 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.*, 2008b] pour plus d'information sur le fonctionnement d'ensemble.

TABLE 220 – 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 221 – 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 222 – 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 223 – Définition du stockage des informations d'entrée-sortie pour la création d'une Umat externe au programme Herezh

```
 /*************************************************************************
*  UNIVERSITE DE BRETAGNE SUD (UBS) --- ENSIBS DE LORIENT                *
*************************************************************************
*           IRDL - Equipe DE GENIE MECANIQUE ET MATERIAUX (EG2M)        *
* 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)     *
*           accés directe aux informations.                             *
*                                                                       *
*                                                                  $     *
*     ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,     *
*                                                                     *
*     VERIFICATION:                                                     *
*                                                                       *
*     !  date  !   auteur   !       but                          !     *
*     ------------------------------------------------------------     *
*     !        !            !                                    !     *
*                                                                  $     *
*     ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,     *
*     MODIFICATIONS:                                                    *
*     !  date  !   auteur   !       but                          !     *
*       5 avril 2016 : G Rio : prise en compte des éléments axi         *
*       17 janvier 2018 : G Rio : prise en compte des contraintes       *
*                                 planes et déformation planes.         *
*     ------------------------------------------------------------     *
*                                                                  $     *
*************************************************************************/

#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*/
```

TABLE 224 – procédure d'écriture sur le pipe, dans le cas de la création d'une Umat externe à Herezh

```
/* 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_DDSDDE =   &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_DROT  =    &t_car_x_n.x[80]; /* 9 */
double* u_herezh_CELENT  =  &t_car_x_n.x[89]; /* 1 */
double* u_herezh_DFGRD0  =  &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 225 – procédure d'écriture sur le pipe, dans le cas de la création d'une Umat externe à Herezh

```
/* ---------------------- 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 DFGRD0,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 DFGRD0(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 ------------- */

    /* procedure de lecture de recuperation sur le pipe des infos calculees */
    void LectureDonneesUmat(double* STRESS,double*  DDSDDE
                            ,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,O_RDONLY);
     /* lecture dans le tampon  */
     read (tub,t_car_x_n.tampon,480);
     close (tub); /* fermeture du tampon */
```

TABLE 226 – procédure d'écriture sur le pipe, dans le cas de la création d'une Umat externe à Herezh

```
      /* transformation de l'information
     a priori *NDI + *NSHR = *NTENS, mais abaqus garde les 3 valeurs donc on se refere uniquement sur l
     premieres donnees -> d'ou un test */
    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;
        DDSDDE[r] = u_herezh_DDSDDE[r];
       };
     };
    }
    else if ((*NDI == 2) && (*NSHR == 1))
     { /* cas des contraintes planes */
       int ij;
       for (ij=0;ij<3;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<3;kl++)
            {int r=ij*3+kl;
             DDSDDE[r] = u_herezh_DDSDDE[r];
            };
        };
     }
    else if ((*NDI == 3) && (*NSHR == 1))
    {/* cas d'elements axisymetrique 3D */
     int ij;
     for (ij=0;ij<4;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<4;kl++)
       {int r=ij*4+kl;
        DDSDDE[r] = u_herezh_DDSDDE[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 227 – 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'ecriture 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)
{

      /* a priori *NDI + *NSHR = *NTENS, mais abaqus garde les 3 valeurs donc on se refere uniquement su
      premieres donnees -> d'ou un test (que l'on pourrait sans doute eviter ) */
   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];
      };
    }
   else if ((*NDI == 2) && (*NSHR == 1))
    { /* cas des contraintes planes */
      int ij;
      for (ij=0;ij<3;ij++)
       {u_herezh_STRESS[ij] = STRESS[ij];
        u_herezh_STRAN[ij] = STRAN[ij];
        u_herezh_DSTRAN[ij] = DSTRAN[ij];
       };
    }
   else if ((*NDI == 3) && (*NSHR == 1))
    { /* cas d'éléments axisymétrique 3D */
      int ij;
      for (ij=0;ij<4;ij++)
       {u_herezh_STRESS[ij] = STRESS[ij];
        u_herezh_STRAN[ij] = STRAN[ij];
        u_herezh_DSTRAN[ij] = DSTRAN[ij];
       };
    };
```

TABLE 228 – procédure d'écriture sur le pipe, dans le cas de la création d'une Umat externe à Herezh

```
   *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 gradiant_t "; for (int i=1;i<9;i++) cout << " ("<<i<<")=" << u_herezh_DFGRD0[i];
      cout << "\n gradiant_tdt "; for (int i=1;i<9;i++) cout << " ("<<i<<")=" << u_herezh_DFGRD1[i];
  cout << endl; */
  /*   Creation d'un processus de pour l'ecriture des donnees */
  /* ouverture du tube nomme en ecriture */
  printf("\n on ouvre le tube pour l'ecriture: sp essai");
  int tub = open(reception,O_WRONLY);
  /* ecriture dans le tampon  */
  /* en fait l'objectif est d'ecrire 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 */
  printf("\n fermeture du tampon: sp essai");
  return;
};
```

51.18.1 Cas d'un état de contraintes planes

La table (229) donne un exemple d'utilisation d'une Umat pour un état de contrainte plane. Par rapport au cas classique 3D, les différences sont :

— Le type de loi de comportement à utiliser doit-être `LOI_VIA_UMAT_CP` au lieu de `LOI_VIA_UMAT`

— Il faut indiquer " `dim_loi= 2` " au lieu de 3,

Remarque On notera que l'Umat ne fournit pas d'information en retour sur la variation d'épaisseur contrairement au cas d'une loi de contrainte plane classique définie dans Herezh.

TABLE 229 – Cas d'un état de contraintes planes : exemple d'utilisation par Herezh d'une Umat extérieure

```
       dimension   3  # toujours en 3D pour une Umat
       niveau_commentaire   3
      TYPE_DE_CALCUL
        non_dynamique
#-------------------------------------
       < rect10x10.her
 #-------------------------------------
        choix_materiaux
          E_tout acier
        materiaux
         acier         LOI_VIA_UMAT_CP
# ------------------------------------------------------------
# |     exemple de loi de comportement defini en Umat      |
# |      via un processus qui dialogue avec herezh++       |
# ------------------------------------------------------------
     nom_de_la_loi= acier_interne_CP    categorie=   CAT_MECANIQUE   dim_loi=  2
     nom_pipe_envoi= Umat_envoi_Hz nom_pipe_reception= Umat_reception_Hz
     permet_affichage_ 0
    fin_loi_Umat
#    --- divers stockages (1) -------
    epaisseurs   #--------#
    E_tout  1
    masse_volumique   #--------#
    E_tout  1

     charges #------------#
     blocages #------------#
     N_S  UY
     N_O  UX
     N_NE  'UY= 10'
     N_tout UZ
      resultats  pas_de_sortie_finale_
      COPIE 0
  _fin_point_info_
```

52 Liste de lois thermo physiques disponibles

52.1 Lois isotropes thermiques

Les lois disponibles sont données par la table (cf. 230).

TABLE 230 – liste des lois de comportement thermophysique

nom	commentaire simplifié	référence
LOI_ISO_THERMO	loi simple isotrope 1D 2D 3D	(52.2)
LOI_DE_TAIT	loi de Tait 1D 2D 3D isotrope	(52.3)

52.2 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 (231) donne un exemple de déclaration de loi à coefficients constants.

TABLE 231 – 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 (232) 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" associé aux points d'intégration (dans le format maple par exemple).

Comme pour la loi de Tait (cf. 52.3), il est également possible de calculer un taux de cristalinité en même temps que les autres grandeurs thermo physiques. Cette prise en compte s'effectue de la manière suivante :

TABLE 232 – 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 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'Hoffmann (cf. 52.4) :

" type_de_calcul_= 2 Cristalinite_= HOFFMAN' '

On se reportera à la table (235) pour un exemple d'application.

52.3 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 (233) donne un exemple de déclaration de ce type de loi.

On remarquera que les coefficients lambda et cp peuvent être dépendants 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++ définisse 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 234).

Actuellement, les grandeurs disponibles (pour le post-traitement) sont : la température de transition, le volume spécifique, le coefficient de dilatation, la conductivité, la capacité

TABLE 233 – 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
```

calorifique, et enfin le module de compressibilité. Ces grandeurs sont accessibles via le menu "grandeurs particulières" associé aux points d'intégration (dans le format maple par exemple). Dans le cas du calcul de la cristalinité, on a également accès à la valeur du

taux de cristalinité.

TABLE 234 – Exemple de déclaration d'une loi thermo physique de Tait avec préparation pour le postraitement 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
```

Il est également possible de calculer un taux de cristalinité en même temps que les autres grandeurs thermo physiques. Cette prise en compte s'effectue de la manière suivante :

- tout d'abord un indicateur (facultatif) indiquant le type de calcul voulu, par défaut = 0
  - `type_de_calcul_ = 1` $\longrightarrow$ calcul du taux de cristalinité, mais pas de dépendance des variables Thermo-physiques (α, λ, la compressibilité) avec la cristalinité
  - `type_de_calcul_ = 2` $\longrightarrow$ 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'Hoffmann (cf. 52.4) :

“ type_de_calcul_= 2 Cristalinite_= HOFFMAN1 ”

On se reportera à la table (235) pour un exemple d'application.

TABLE 235 – Exemple de déclaration d'une loi thermo physique de Tait avec calcul du taux de cristalinité 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 defaut -> 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 (faculatatif) indiquant le type de calcul voulu, par defaut = 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 cristalinité, avec dépendance 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_
```

52.4 Loi d'Hoffman d'évolution de la cristalinité

Dans l'implantation actuelle, ce type de loi est a utilisée conjointement avec une loi thermo physique de type par exemple loi de Tait (52.3) ou la loi LOI_ISO_THERMO (52.2). 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 :

$$K(T,P) = G_0 \left(\left(\frac{4}{3} \pi N_0 \right)^{1/3} \exp\left(- \frac{\overset{*}{U}}{(R\ (T_{corr} - T_\infty))} \right) \exp\left(- \frac{Kg}{(T_{corr}\ (T_m - T_{corr})} \right) \right) \tag{112}$$

avec les données suivantes :

- $DT_{corrPres} = P\ (\alpha_P + P\ \beta_P)$
- $T_{corr} = T - DT_{corrPres} + 273.$
- $N_0 = \exp\left(N_{01}\ (T_m - T_{corr}) + N_{02}\right)$
- 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}, $\overset{*}{U}$, T_∞, Kg, T_m, α_P, β_P
Le taux de cristalinité $c(t)$ se calcul à l'aide de l'expression suivante :

$$c(t) = X_\infty \left\{ 1 - \exp\left(- \left[\int_0^t K(t')\ dt' \right]^n \right) \right\} \tag{113}$$

X_∞ correspond au taux maximal de cristalinité atteignable pour les conditions données de pression et de température.
La table (236) donne un exemple de description de la loi d'Hoffman.

52.5 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*Tcorr/(Tcorr + Tm)

$$K(T,P) = (log(2))^{1/n}\ \frac{1}{T_{1/2}} \left(\exp(- \frac{\overset{*}{U}}{R\ (T_{corr} - T_{inf}} \right) \exp\left(- \frac{Kg}{(T_{corr}\ (T_m - T_{corr}) * f)} \right) \tag{114}$$

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, $T_{1/2}$, $\overset{*}{U}$, T_∞, Kg, T_m, α_P, β_P

TABLE 236 – Exemple de déclaration d'une loi d'Hoffman permettant le calcul de la cristalinité 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_ "
```

Comme pour Hoffman1, le taux de cristalinité $c(t)$ se calcul à l'aide de l'expression suivante :

$$c(t) = X_\infty \left\{ 1 - \exp\left(- \left[\int_0^t K(t')\, dt' \right]^n \right) \right\} \tag{115}$$

X_∞ correspond au taux maximal de cristalinité atteignable pour les conditions données de pression et de température.

La table (237) donne un exemple de description de la loi d'Hoffman.

53 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 constitués 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 "cet" élément fini.

Les éléments de contact se créent et disparaissent au gré de l'apparition ou non de contact.

TABLE 237 – Exemple de déclaration d'une loi d'Hoffman permettant le calcul de la cristalinité 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,invt012,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) * invt012 * 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         invt012= 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_ "
```

Les lois disponibles sont données par la table (cf. 238).

TABLE 238 – liste des lois disponibles de frottement lié au contact

nom	commentaire simplifié	référence
LOI_COULOMB	loi de coulomb 1D 2D 3D	(53.1)

53.1 Loi de Coulomb

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 régimes 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 :

$$\vec{F}_T = -\mu ||\vec{F}_N|| . \frac{\vec{V}_{rt}}{||\vec{V}_{rt}||} \tag{116}$$

$\vec{F}_T$ 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 :

$$\mu(v) = \mu_S + (\mu_S - \mu_C).e^{-c.v} \tag{117}$$

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 (239) donne un exemple de description d'une loi de Coulomb classique.

TABLE 239 – 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 (240) donne un exemple de description d'une loi de Coulomb dont le coefficient de frottement dépend de la vitesse tangentielle selon la loi (117).

TABLE 240 – 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 cinematique
#  x_c= regle le passage du frottement statique au frottement cinematique
#  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 régularisé, 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, dû 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 :

$$\vec{F_T} = -\mu.\varphi(v)||\vec{F_N}||.\frac{\vec{V_{rt}}}{||\vec{V_{rt}}||} \tag{118}$$

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).

$$\varphi_1(v) = \frac{v}{\sqrt{(v^2+\varepsilon^2)}} \tag{119}$$

$$\varphi_1(v) = tanh\left(\frac{v}{\varepsilon}\right) \tag{120}$$

$$\varphi_1(v) = \begin{cases} -1 & if \quad v < -\varepsilon \\ \frac{v}{\varepsilon} & if \quad -\varepsilon \leq v \geq \varepsilon \\ 1 & if \quad > \varepsilon \end{cases} \tag{121}$$

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 (241) 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 241 – Exemple de déclaration d'une loi de frottement de Coulomb régularisée.

```
# ----------------------------------------------------------------------
# |......  loi de comportement de frottement de coulomb  .......    |
# ----------------------------------------------------------------------
    mu_statique= 0.3
    regularisation= 3   epsil= 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))
#  epsil=  est le parametre de reglage 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

Divers stockages

54 Généralités

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 définir 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 associés automatiquement l'épaisseur indiquée.

La syntaxe est donnée par l'exemple de la table (243) dans le cas d'un seul maillage.

TABLE 242 – exemple de définition d'épaisseur et de section.

```
     epaisseurs ------------
EMAT0001        240.

        sections----------
 EMAT0002       30000.
```

Cet exemple permet d'introduire une référence d'épaisseurs. Tous les éléments de la référence d'éléments "EMAT0001" auront l'épaisseur 240. De la même manière, il est possible d'introduire également une section 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 définie, les éléments sont considérés comme incomplets, et le calcul ne peut se faire. En général, le programme indique un message d'erreur en ce sens.

La syntaxe est donnée par l'exemple de la table (243) dans le cas d'un seul maillage.

TABLE 243 – 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 maillages, 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 (244) dans le cas de plusieurs maillages.

TABLE 244 – 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.
```

À noter qu'au niveau du fichier .info, l'introduction des stockages particuliers s'effectue dans deux endroits :

1. soit à la suite des lois de comportement, avant la définition des charges (indiqué par 1 dans le tableau exhaustif 245)
2. soit à la suite des conditions d'initialisation avant les paramètres de contrôle (indiqué par 2 dans le tableau exhaustif 245).

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 maillages, il est obligatoire d'ajouter la définition du nom de maillage.

55 Liste exhaustive des stockages divers

TABLE 245 – Liste exhaustive des stockages divers

grandeurs	position dans le fichier .info	ref
epaisseurs	1	56
sections	1	57
variation_section	1	58
largeurs	1	59
masse_volumique	1	60
dilatation_thermique	1	64
hourglass_gestion_	1	42
stabilisation_transvers_membrane_biel_	1	43
integrale_sur_volume_	1	61
integrale_sur_volume_et_temps_	1	61
masse_addi	2	62

56 Définition de l'épaisseur d'un groupe d'éléments

L'exemple précédent (cf. 243) donne un exemple de l'entrée d'une épaisseur.

NB : Bien noter que tous les éléments qui ont une épaisseur doivent avoir une épaisseur définie. Il n'y a pas d'épaisseur par défaut. Par contre tous les éléments n'ont pas une épaisseur.

Il est également possible de définir une épaisseur qui varie en fonction de l'application d'une fonction nD. Pour cela on indique le mot clé `une_fonction_nD_` à la suite de la référence d'éléments puis le nom (une chaîne de caractère) d'une fonction nD qui doit exister. La table (246) donne un exemple de déclaration.

TABLE 246 – exemple de définition d'une épaisseur variable en fonction d'une fonction nD.

```
     epaisseurs  # ------------
E_tout        une_fonction_nD_  fepaiss
```

57 Définition de la section d'un groupe d'éléments

L'exemple (cf. 248) donne un exemple de l'entrée d'une section.

NB : Bien noter que tous les éléments qui ont une section, doivent avoir une section définie. Il n'y a pas de section par défaut. Par contre tous les éléments n'ont pas une section.

Il est possible de définir une section qui varie en fonction de l'application d'une fonction nD. Pour cela on indique le mot clé `une_fonction_nD_` à la suite de la référence d'éléments puis le nom (une chaîne de caractère) d'une fonction nD qui doit exister. La table (247) donne un exemple de déclaration.

Néanmoins, actuellement la seule dépendance possible est vis-vis des coordonnées initiales c'est-à-dire les variables libellées : `X1_t0` , `X2_t0` et `X3_t0`

TABLE 247 – exemple de définition d'une épaisseur variable en fonction d'une fonction nD.

```
     sections   # ------------
E_tout        une_fonction_nD_  f_section
```

58 Définition éventuelle du type de variation de la section d'un groupe d'éléments

L'exemple (cf. 248) donne un exemple de définition du type de variation d'une section. Si l'on indique 0, cela signifie que l'on ne veut pas de variation de section. Elle reste donc fixe pendant le calcul. Si l'on indique 1, ce qui est la valeur par défaut, la section peut varier pendant le calcul. Dans un calcul mécanique, elle est mise à jour en fonction de l'équilibre locale via la loi de comportement et la cinématique imposée.

TABLE 248 – exemple de définition de la section et de la variation de section d'un groupe d'éléments. Ici on ne veut pas de variation de section.

```
#    --- divers stockages (1) -------
 sections   #--------#
E_tout  4
 masse_volumique   #--------#
E_tout  2
 variation_section
E_tout  0
```

59 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. 249).

TABLE 249 – 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 définie. Il n'y a pas de largeur par défaut. Par contre tous les éléments n'ont pas une largeur.

Il est possible de définir une largeur qui varie en fonction de l'application d'une fonction nD. Pour cela on indique le mot clé `une_fonction_nD_` à la suite de la référence d'éléments puis le nom (une chaîne de caractère) d'une fonction nD qui doit exister. La table (250) donne un exemple de déclaration.

TABLE 250 – exemple de définition d'une épaisseur variable en fonction d'une fonction nD.

```
     largeurs   # ------------
E_tout        une_fonction_nD_  f_largeur
```

TABLE 251 – exemple de définition de la masse volumique d'un groupe d'éléments.

```
     masse_volumique #------------
EMAT0001        24.
```

60 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. 251).

NB : Bien noter que tous les éléments doivent avoir une masse volumique définie. Il n'y a pas de masse volumique par défaut.

Il est possible de définir une masse volumique qui varie en fonction de l'application d'une fonction nD. Pour cela on indique le mot clé une_fonction_nD_ à la suite de la référence d'éléments puis le nom (une chaîne de caractère) d'une fonction nD qui doit exister. La table (252) donne un exemple de déclaration.

TABLE 252 – exemple de définition d'une masse volumique variable en fonction d'une fonction nD.

```
     masse_volumique   # ------------
E_tout        une_fonction_nD_  f_masse_vol
```

61 Définition d'une intégration sur un volume

On se reportera à 84 pour une information plus précise sur l'utilisation d'une intégration de volume. L'objectif ici est de préciser la syntaxe à utiliser pour mettre en place une intégrale de volume particulière. Celle-ci s'effectue via 4 paramètres :

1. un mot clé choisit parmi :
   (a) soit "

       integrale_sur_volume_

   " ce qui indique que l'on désire une intégrale sur le volume,

TABLE 253 – Exemple de définition d'intégrale sur un groupe d'éléments.

```
       integrale_sur_volume_  #-----
# exemple de l'intégrale de volume sur E_tout de la coordonnées X3
E_tout        un_ddl_etendu_     X3
# exemple de l'intégrale de volume sur E_tout de la fonction f1
#  f1 doit avoir été définie auparavant comme fonction nD
E_tout        une_fonction_nD_    f1
      integrale_sur_vol_et_temps_
# exemple du cumul en temps de l'intégrale de volume sur E_tout de D11
E_tout        un_ddl_etendu_     D11
```

(b) soit "

integrale_sur_vol_et_temps_

" ce qui indique que l'on désire une intégrale sur le volume et le temps,

2. une référence d'élément
3. un mot clé choisit parmi 2 propositions :
   (a) soit "

   un_ddl_etendu_

   " qui indique alors que l'on désire une intégrale d'un ddl de base (cf. 80.11.4) ou un ddl étendu (cf. 88.3),
   (b) soit "

   une_fonction_nD_

   " qui indique alors que l'on désire une intégrale d'une fonction nD (cf. 81.2)
4. puis :
   (a) soit le nom du ddl ou ddl étendu choisit parmi 80.11.4 et 88.3, dans le cas d'une intégrale d'un ddl,
   (b) soit le nom d'une fonction nD déjà définie par ailleurs dans la mise en donnée.

La table 253 donne des exemples de la syntaxe.

62 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. 251). 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.

TABLE 254 – exemple de définition de masse additionnelle sur un groupe de noeud.

```
        masse_addi ------------
N_tout          0.03.
```

63 Définition d'un repère d'anisotropie

Il est possible d'associer un repère initial d'anisotropie aux éléments. Ce type de repère peut-être utilisé par une loi de comportement anisotrope qui en général nécessite l'utilisation d'un repère particulier pour s'exprimer.

En pratique un repère initial d'anisotropie est souvent fonction de la géométrie de la pièce modélisée d'où l'idée de permettre une flexibilité dans la définition du repère via l'utilisation de fonction nD.

La définition s'effectue via :

1. le mot clé `repere_anisotropie_`
2. une référence d'élément
3. un identificateur de repère = une chaîne de caractère au choix de l'utilisateur. Cet identificateur permet ensuite un choix entre plusieurs repères éventuellement existants pour un même élément. Par exemple si on utilise une loi anisotrope, l'identificateur est utilisé pour associer un repère à une loi (ex : 51.15.1 et 212).
4. un mot clé qui indique le type de repère que l'on souhaite. Actuellement, un seul type est accessible : `orthonorme_`
5. un mot clé qui indique comment le repère est défini. Actuellement, deux méthodes de définition sont accessibles :
   (a) `Par_projection_et_normale_au_plan_` : cette méthode est utilisable pour les éléments 2D (membranes, coques, plaques). La fonction nD fournit un premier vecteur qui, projeté sur la surface de l'élément et normé, donne la première direction du repère. La normale à la surface de l'élément donne la deuxième direction du repère. Le produit vectoriel de ces deux premiers vecteurs donne la troisième direction.
   (b) `par_fonction_nD_` : cette méthode est utilisable pour tout type d'élément. La fonction nD doit-être une fonction vectorielle à 6 composantes, telles que les 3 premières valeurs correspondent à la première direction, les 3 suivantes à la deuxième direction, le dernier vecteur est obtenu par produit vectoriel des deux premiers. La table (255) donne un exemple de définition.
6. le nom d'une fonction nD qui est associé à la méthode de définition du repère.

TABLE 255 – exemple de définition d'un repère d'anisotropie sur un groupe d'éléments.

```
 repere_anisotropie_   #--------#
#-------------------------------------------------------
 E_tout  repere_1 orthonorme_   par_fonction_nD_  une_fonction_nD_ fct_repere1
```

64 Déclaration de la prise en compte d'une dilatation thermique

On se reportera à 108 pour un exemple complet de déclaration. Bien noter que la définition d'une méthode de prise en compte d'une dilatation thermique via une loi thermophysique n'est pas suffisant pour que la dilatation soit prise en compte. Si l'on veut activer la dilatation, il faut définir les éléments sur lesquels on veut une prise en compte d'une dilatation. Une des conséquences est que l'on peut ainsi simuler des dilatations activées uniquement dans certaines régions géométriques.

Huitième partie

Conditions de contact

65 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.(§77).

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.(65.1). On peut également prendre en compte un auto-contact entre les noeuds et éléments d'un même maillage cf.(65.2). 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.(65.3).

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 fournie 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.(§77).

65.1 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.(§28). 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.

Ainsi par défaut tous les noeuds du maillage esclave sont susceptibles d'entrer en contact avec les faces frontières 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 avant 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, dans ce cas le maillage est maître et esclave.
- Il est également possible de définir plusieurs maillages qui jouent le rôle de maître et esclave. Ceci s'effectue via le mot clé " `auto_contact` " cf. 65.2. Ensuite on peut particulariser éventuellement les zones du maillage où l'on souhaite que les noeuds soient esclaves et les zones où l'on veut des frontières maîtres cf.(65.3).
- par défaut tous les couples possibles "maillage esclave" $\leftrightarrow$ "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.(65.3).
- Il n'y a pas de limitation sur le nombre de maillages possibles.

65.2 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. Mais c'est aussi la technique dans Herezh++ pour spécifier qu'un maillage est en même temps maître et esclave. 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.(65.3).
- 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.(65.3).

Exemple :

```
...
 epaisseurs   #--------#
nom_mail= exterieur E_tout  2.
```

```
#    --- les contacts -------
  auto_contact
    2

  zone_contact
#-----------------------------------------------------
# ref noeuds            |      ref face             |
#-----------------------------------------------------
  nom_mail= exterieur  N_LIGNE_ext_sup  nom_mail= cube  F_liaison

       charges #------------#
 ...
```

65.3 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.(§37). Voici un exemple de déclaration :

```
     zone_contact -------------
#-------------------------------------------------
# liste des références ou le contact est recherche |
#-------------------------------------------------
nom_mail= tole_25 N_deb   nom_mail= outil  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 toutes les autres frontières !
- Les références de toutes les frontières et de tous les noeuds peuvent-être automatiquement générées à la lecture du maillage cf.(37.1).
- Dans le cas où il n'y a pas de zone de contact explicitement définit, Herezh utilise les frontières générées automatiquement (cf. 22.2). Dans certains cas (par exemple

dans le cas d'éléments volontairement superposés dans un même maillage) certaines frontières ne sont plus accessibles (car considérées comme non frontière par Herezh cf. 22.2). Si l'on veut maintenir néanmoins un ensemble de frontières, il faut déclarer explicitement des zones de contact collant ou pas, ce qui permettra d'inclure ces zones dans les frontières et les noeuds esclaves, candidats au contact.

65.4 Introduction d'un "collage" entre maillage à l'aide d'un contact

Il est possible d'utiliser les zones de contact pour indiquer un contact "collant". La technique peut-être utile pour indiquer une liaison entre maillages contigus mais non compatibles. C'est le cas par exemple lorsque au niveau de la liaison, les noeuds des deux maillages ne sont pas coïncidant et/ou que l'interpolation n'est pas identique.

Suivant la méthode de contact utilisée, la condition de collage introduit différentes continuités au niveau de la liaison.

— Pour un contact avec pénalisation, il y a une continuité des efforts. Au niveau cinématique, la continuité dépend de la pénalisation. Dans le cas d'Herezh 2 pénalisations sont utilisées. La première gère la pénétration, le gap de pénétration constaté au niveau de chaque noeud esclave représente la discontinuité cinématique dans la direction normale à la facette de contact. La seconde pénalisation gère le déplacement sur la surface de contact. Le gap de déplacement mesure la discontinuité cinématique sur la surface de la facette de contact. Dans le cas où la seconde pénalisation est nulle, le noeud esclave est autorisé à se déplacer sur la facette sans contrainte.

— Pour un contact via une condition cinématique, il y a continuité des efforts et continuité cinématique. Pour l'instant ce contact pour une liaison collante, nécessite dans Herezh des développements ...

Pour indiquer un contact collant il faut utiliser des zones de contact avec le mot clé `glue_contact` en lieu et place du mot clé `zone_contact` . À noter qu'il est possible d'avoir plusieurs types de zone de contact.

Lorsque l'on obtient des maillages issues par exemple d'une CAO, via un modeleur, il n'est pas rare d'avoir une légère différence de position entre les points et faces définies au niveau de la zone de collage. Ceci provient pas exemple du fait que l'interpolation utilisée dans la CAO est différente (et en générale plus riche) que celle utilisée en éléments finis. La conséquence est que dès le début du calcul, les conditions de contact peuvent imposer des déplacements visant à supprimer ces différences de positions. Ceci peut conduire à des efforts internes éventuellement important et non physique, car résultants d'erreurs initiales de positions.

Une technique est proposée dans Herezh pour palier à ces difficultées dans le cas d'un contact collant. À la place du mot clé `glue_contact` l'utilisation du mot clé " `glue_contact_init_gap_zero` " indique à Herezh de supprimer le gap initial avant de commencer le calcul. La méthodologie mise en oeuvre est la suivane :

1. Avant la phase de calcul, une phase d'initialisation de contact est effectuée (dans tous les cas) et permet de définir précisément les zones de contact collant.

2. Chaque noeud esclave appartenant à une zone de contact type : glue_contact_init_gap_zero est alors projeté perpendiculairement sur la frontière la plus proche. Cette projection n'est valide que si la distance entre le noeud et le projeté est inférieure à une limite controlée par le mot clé : DISTANCE_MAXI_AU_PT_PROJETE cf.77.
3. La position initiale des noeuds esclaves, pour les cas où le projeté est valide, devient alors celle du projeté.

Remarques :

1. Les éléments de contact collant ne sont définit qu'à l'initialisation.
2. Les éléments de contact collant, sont conservés tout au long du calcul. En particulier lors d'un restart, les éléments collants initiaux sont conservés : il s'agit des éléments définis à l'initialisation. L'élément de contact est constitué d'un noeud esclave et d'une facette maître en contact. S'il y a un glissement important, la facette en contact peut changer en cours de calcul.
3. La suppression du gap initial n'est effectuée qu'à l'initialisation, avant le calcul.

65.5 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.(§77). On se reportera à cette section pour plus d'information. De plus dans le cas particulier de l'utilisation de l'algorithme de relaxation dynamique, un paramètre de contrôle supplémentaire est disponible cf. (20.3.13) pour plus d'informations.

65.6 Remarques sur le post-traitement

Le contact introduit des comportements qui peuvent-être difficile à analyser. Pour faciliter l'analyse de l'évolution des calculs il est possible de au fil du calcul (et par exemple en mode debug) un certain nombre d'informations sur le contact. Le choix des sorties (par exemple en .maple ou en .pos) est contrôlé via un mot clé. On donne ci-dessous des indications sur la signification des sorties en fonction du mot clé choisi.

— NOEUD_PROJECTILE_EN_CONTACT : chaque noeud projectile reçoit la valeur 100 dans le cas où il appartient à un contact actif, -1 s'il appartient à un contact inactif. Dans le cas où le noeud appartient à plusieurs éléments de contact, la valeur est cumulée. Ainsi, à condition qu'un noeud n'appartienne pas à plus de 99 éléments de contact il est possible de déterminer les noeuds en contact, actifs ou pas.

— NOEUD_FACETTE_EN_CONTACT : chaque noeud d'une facette en contact reçoit la valeur 100 ou -1 selon que l'élément de contact est actif ou pas. Si le noeud appartient à plusieurs facettes, l'opération est cumulée, et on suppose qu'un noeud n'appartient pas à plus de 99 facettes.

— PENETRATION_CONTACT : chaque noeud projectile reçoit un vecteur $= \vec{M}_{tdt} - \vec{M}_{impact}$, $\vec{M}_{tdt}$ la position du noeud, $\vec{M}_{impact}$ la position du point d' impact sur la facette.

— GLISSEMENT_CONTACT : chaque noeud projectile reçoit le vecteur déplacement parallèlement à la facette de contact. Il y a cumul si le noeud appartient à plusieurs facettes.

— NORMALE_CONTACT : chaque noeud projectile reçoit un vecteur = la normale de contact. Il y a cumul si le noeud appartient à plusieurs facettes.

— FORCE_CONTACT : chaque noeud projectile reçoit un vecteur = la force de contact. Il y a cumul si le noeud appartient à plusieurs facettes.

— CONTACT_PENALISATION_N : chaque noeud projectile reçoit un scalaire = la valeur du facteur de pénalisation dans la direction normale à la facette de contact. Il y a cumul si le noeud appartient à plusieurs facettes.

— CONTACT_PENALISATION_T : chaque noeud projectile reçoit un scalaire = la valeur du facteur de pénalisation dans la direction tangentielle à la facette de contact. Il y a cumul si le noeud appartient à plusieurs facettes.

— CONTACT_ENERG_PENAL : chaque noeud projectile reçoit un scalaire = l'énergie liée à l'application de pénalisation normale. Il y a cumul si le noeud appartient à plusieurs facettes.

— CONTACT_ENERG_GLISSE_ELAS : chaque noeud projectile reçoit un scalaire = l'énergie élastique donc réversible, liée à l'application de pénalisation tangentielle. Il y a cumul si le noeud appartient à plusieurs facettes.

— CONTACT_ENERG_GLISSE_PLAS : chaque noeud projectile reçoit un scalaire = l'énergie plastique (donc non réversible et non visqueuse)liée à l'application de pénalisation tangentielle. Il y a cumul si le noeud appartient à plusieurs facettes.

— CONTACT_ENERG_GLISSE_VISQ : chaque noeud projectile reçoit un scalaire = l'énergie visqueuse liée à l'application de pénalisation tangentielle. Il y a cumul si le noeud appartient à plusieurs facettes.

— CONTACT_NB_DECOL : chaque noeud projectile reçoit un scalaire = le nombre de situations en décollement actuellement stocké. Il y a cumul si le noeud appartient à plusieurs facettes.

— CONTACT_NB_PENET : chaque noeud projectile reçoit un scalaire = le nombre de situations en pénétration actuellement stocké. Il y a cumul si le noeud appartient à plusieurs facettes.

— CONTACT_CAS_SOLIDE : chaque noeud projectile reçoit un scalaire = :

  — =0 il s'agit d'un contact bi-déformable,

  — =1 le noeud est libre et la frontière est bloqué (solide)

  — = 2 le noeud est bloqué (solide) la frontière est libre

  — = 3 tout est solide

  Il n'y a pas cumul si le noeud appartient à plusieurs facettes, c'est le dernier élément de contact auquel appartient le noeud projectile, qui est utilisé pour la sortie.

— CONTACT_COLLANT : chaque noeud projectile reçoit un scalaire = : 1 si le noeud esclave est collant , 0 sinon. Il y a cumul si le noeud appartient à plusieurs facettes.

— NUM_ZONE_CONTACT : chaque noeud projectile reçoit un scalaire = : le numéro de la zone de contact du noeud esclave (0 s'il n'y a pas de définition de zone de contact particulière). Le numéro correspond au numéro d'ordre de déclaration des zones de contact, dans le fichier de mise en données. Si aucune zone n'est déclarée il est nul. Il y a cumul si le noeud appartient à plusieurs facettes.

Neuvième partie

Types d'efforts

66 Conditions de chargement imposé

66.1 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 (256) montre comment implanter un chargement ponctuel lorsqu'il n'y a qu'un seul maillage.

TABLE 256 – 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 ponctuel sur tous les noeuds de la liste de nom "NI". Le vecteur charge a deux coordonnées : O. 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 gène pas la lecture des données, seulement il faut se rappeler, que seules les n premières composantes, n étant la dimension, seront prise en compte pour le calcul.

Dans le cas où il y a plusieurs maillages, 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 (257) montre comment implanter un chargement ponctuel lorsqu'il y a plusieurs maillages.

TABLE 257 – 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 maillages.

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 maillages, il est obligatoire d'ajouter la définition du nom de maillage*

— *comme pour tous les chargements, il est possible de compléter le chargement par une courbe de charge, une échelle, un temps mini et un temps maxi. Ne pas oublier la possibilité d'utiliser le signe*

\

pour couper une ligne en deux, si celle-ci est trop longue (cf.1.3).

66.1.1 Cas d'un chargement via un champ de valeurs

Lorsque l'on veut charger un ensemble volumineux de noeuds il peut-être intéressant d'utiliser une variante sous la forme d'un chargement de type "champ". La table 258 donne un exemple de syntaxe.

Dans cet exemple :

— "N_list_noe" : représente une référence de "n" noeuds,

— " `PONCTUELLE` " : le mot clé qui indique que le chargement est de type force ponctuelle,

— " `champ_de_valeurs:` " : ce mot clé indique qu'il s'agit d'un champ de valeur

— "<charge_champ_neige.force" : < indique qu'il s'agit d'une inclusion d'un fichier. Ainsi ici toutes les prochaines infos sont à lire dans le fichier de nom "charge_champ_neige.force" qui doit donc contenir le champ de valeurs. Le fichier doit contenir "n" vecteurs (un vecteur par noeud de la référence "N_list_noe"). On peut indiquer un ou plusieurs vecteurs par ligne. Le ième vecteur correspond à la force qui doit-être appliquée au ième noeud de la référence.

— " `fin_champ_de_valeurs:` " : le mot clé qui indique la fin du champ de vecteurs.

L'utilisation de la notion de champ est surtout une facilité pour prendre en compte un grand nombre de valeurs, générées souvent automatiquement. Par exemple en utilisant le fichier .reac généré automatiquement par Herezh en fin de calcul, on obtient ainsi un champ de vecteurs ici le champ de réactions, qui peut alors lui-même être utilisé comme données d'entrée pour un autre calcul.

NB :

— *Il n'est pas obligatoire d'utiliser un fichier intermédiaire, les informations peuvent être évidemment directement incluses dans le fichier .info,*

— *il faut veiller à ce qu'il y ait le même nombre de vecteurs dans le champ que de noeuds indiqués dans la référence. Si ce n'est pas le cas, au moment de l'exécution, Herezh indiquera une erreur qui stoppera l'exécution.*

— *comme pour tous les chargements, à la suite du champ, il est possible de compléter le chargement par une courbe de charge, une échelle, un temps mini et un temps maxi. Ne pas oublier la possibilité d'utiliser le signe "ṗour couper une ligne en deux, si celle-ci est trop longue (cf.1.3).*

TABLE 258 – 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 |
 #-------------------------------------------------------------
N_list_noe  PONCTUELLE champ_de_valeurs:
                                < charge_champ_neige.force
                            fin_champ_de_valeurs:
```

66.1.2 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 prescrite 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 (260) indique que l'intensité de la force ponctuelle suit la courbe de charge qui pour nom "courbe1" (voir 46).

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.

66.1.3 Chargement particulier suivant une fonction nD

De manière analogue à l'utilisation d'une courbe de charge 1D, il est possible d'utiliser une fonction nD. Dans ce cas le chargement peut dépendre de toutes les variables globales qui existent pendant le calcul, et également des grandeurs disponibles aux points où on calcule le chargement.

La syntaxe est identique au cas 66.1.2 avec comme différence le mot clé `COURBE_CHARGE:` remplacé par le mot clé `Fonction_nD_CHARGE:` . La table 259 donne un exemple de déclaration pour un chargement ponctuel.

Dans le cas d'un chargement volumique, les points d'intégration utilisés pour imposer et calculer les efforts imposés, sont les mêmes points d'intégration que ceux utilisés pour le calcul d'équilibre. Aussi, dans la définition de la fonction nD, il est possible de choisir comme paramètre, toutes les grandeurs qui sont calculées classiquement au point d'intégration à savoir : déformation, contrainte, vitesse de déformation, etc. et également toutes les grandeurs qui en découlent comme les invariants, la contrainte de Mises, etc. On se reportera à 88.3 pour une description exhaustive des grandeurs habituellement disponibles. On notera qu'il est en particulier possible d'utiliser comme variable, la position

TABLE 259 – Exemple le mise en place d'un chargement ponctuel avec un contrôle via une fonction nD, 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   Fonction_nD_CHARGE: fcharge ECHELLE: 1.2 TEMPS_MINI= 0. TEMPS_MAXI= 3.
```

actuelle , la position au pas de temps précédant, la position initiale, pour modéliser un chargement qui dépend de la géométrie.

Dans le cas d'un chargement sur une frontière (ex : pression sur une surface), par défaut, les points d'intégrations des efforts externes ne sont pas forcément coïncidant avec les points d'intégration pour le calcul de l'équilibre. Aussi, les grandeurs disponibles sont essentiellement celles disponibles par interpolation de valeurs aux noeuds :

- les grandeurs liées à la cinématique : position actuelle et/ou à t et/ou initiale, déplacement, vitesse et accélération s'il s'agit d'un calcul dynamique,
- les grandeurs obtenues par interpolation de données imposées ou connues aux noeuds : par exemple la température si elle est donnée par la mise en données. D'une manière générale, toutes les grandeurs que l'utilisateur décide d'introduire aux noeuds via sa mise en données, sont utilisables pour les fonctions nD.

Il est possible également d'utiliser les grandeurs globales disponibles pendant le calcul, et en particulier le temps. Ainsi la fonction nD généralise la fonction de charge qui ne dépend que du temps. On se reportera à 88.1 pour une liste de grandeurs habituellements disponibles. Mais il peut y avoir d'autres grandeurs globales qui apparaissent pour un calcul particulier : par exemple à chaque fois que l'on introduit des intégrales à calculer. On se reportera à 61 et 84 pour une information plus détaillée. Enfin lorsque l'on effectue un calcul avec " `plus visualisation` " cf. 80.3, on peut obtenir en interactif, la liste des grandeurs globales disponibles, pour le calcul en cours.

Remarque importante Lorsqu'une fonction nD est de type scalaire (i.e. ramène une seule valeur), c'est l'ensemble du chargement qui est multiplié par le scalaire. Lorsque la fonction nD est de type vectorielle (cf. 81.2.5), et qu'elle est utilisée dans un chargement qui utilise un vecteur, il faut que la dimension du vecteur de charge et de la fonction nD soit la même, de façon qu'au final chaque composante de chargement est égale à la composante du vecteur de charge initiale multiplié par la composante correspondante de la fonction nD.

Par exemple, supposons que la fonction nD "fcharge" de l'exemple (259) soit définit de manière suivante :

```
fcharge  FONCTION_EXPRESSION_LITTERALE_nD
 un_argument= temps_courant
 fct=  10*temps_courant, 20*temps_courant, temps_courant
 fin_parametres_fonction_expression_litterale_
```

TABLE 260 – 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 ECHELLE: 1.2 TEMPS_MINI= 0. TEMPS_MAXI= 3.
```

Dans ce cas la force ponctuelle appliquée sera au final le vecteur :
< 0 * (10* `temps_courant`) , -20340* (20* `temps_courant`) , 0* (`temps_courant`) >
On voit aussi dans cet exemple qu'il n'est pas pertinent d'avoir des "0" dans la définition du vecteur initial, mais c'est simplement un exemple.

66.1.4 Mise en place d'un temps mini et/ou d'un temps maxi

L'exemple de la table (260) 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 indiqué. 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 exclue de l'intervalle, à l'inverse du temps maxi qui lui est inclus dans l'intervalle.

Le paragraphe(66.2) indique les différents types de chargement que l'on peut introduire.

66.1.5 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 nulle. 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.

66.2 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 attachées à dimensionner le chargement.

La liste exhaustive des identificateurs disponibles est donnée dans la table (261).

Il est possible évidemment d'avoir plusieurs types de chargement en même temps. Il suffit dans ce cas d'indiquer successivement les différents chargements. 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 261 – liste des différents types de chargement

identificateur	ref du commentaire
PONCTUELLE	(66.1)
VOLUMIQUE	(66.2.1) et pseuso_massique (66.2.2)
LINEIQUE	(66.2.3)
LINEIC_SUIVEUSE	(66.2.4)
UNIFORME	(66.2.5)
PRESSION	(66.2.6)
PRESSDIR	(66.2.7)
PHYDRO	(66.2.8)
TORSEUR_PONCT	(66.2.10)

66.2.1 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 : (262).

TABLE 262 – Exemple de déclaration d'un chargement volumique

```
            charges ------------
#-----------------------------------------
# Ref face | Type de charge   |    valeurs |
#-----------------------------------------
E_25      VOLUMIQUE             0. 0.1  5.
```

Comme pour le chargement de type " PONCTUELLE " il est possible d'utiliser un champ de vecteur lorsque l'on veut grouper un ensemble de charges différentes. La syntaxe pour le champ, est identique au cas du chargement PONCTUELLE (cf.258).

NB : Ne pas oublier que l'on peut rajouter si on le souhaite, à la suite de la description du chargement : une courbe de charge globale (cf. 66.1.2) ou une fonction nD (cf. 66.1.3), un facteur d'échelle, un temps mini et un temps maxi.

66.2.2 Chargement pseudo massique

En fait ce chargement est une déclinaison du chargement volumique. On l'indique dans une section à part de manière à plus facilement le repérer lors d'une recherche dans la table des matières de la documentation.

Le chargement volumique est par défaut un chargement par rapport au volume final. Ce qui est souvent logique pour un vrai chargement volumique. Par contre si l'on désire un chargement par rapport à la masse, il suffit de charger la pièce via un chargement volumique se rapportant au volume initial, ceci via la masse volumique initiale. Pour ce faire on ajoute le mot clé " `sur_volume_initial_` " au chargement volumique classique. Ce mot clé doit être précédé du mot clé : " `ATTRIBUT_` " .

Exemple de syntaxe : (263).

TABLE 263 – Exemple de déclaration d'un chargement volumique se rapportant au volume initial

```
            charges ------------
#-----------------------------------------
# Ref face | Type de charge   |    valeurs |
#-----------------------------------------
E_25      VOLUMIQUE             0. 0.1  5.  ATTRIBUT_  sur_volume_initial_
```

Le reste du fonctionnement suit les mêmes règles que le cas volumique classique cf.66.2.1.

66.2.3 Chargement linéique

Identificateur " `LINEIQUE` " : identificateur de charge linéique. Nécessite une référence d'arêtes 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'arête, elle varie donc pendant le calcul. Exemple de syntaxe : (264).

TABLE 264 – Exemple de déclaration d'un chargement linéique

```
            charges ------------
#-----------------------------------------
# Ref face | Type de charge   |    valeurs |
#-----------------------------------------
   A_34      LINEIQUE             0. 0.  2.
```

Comme pour le chargement de type " PONCTUELLE " il est possible d'utiliser un champ de vecteur lorsque l'on veut grouper un ensemble de charges différentes. La syntaxe pour le champ, est identique au cas du chargement PONCTUELLE (cf.258).

NB : Ne pas oublier que l'on peut rajouter si on le souhaite, à la suite de la description du chargement : une courbe de charge globale (cf. 66.1.2) ou une fonction nD (cf. 66.1.3), un facteur d'échelle, un temps mini et un temps maxi.

66.2.4 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 : (265).

TABLE 265 – Exemple de déclaration d'un chargement linéique suiveur

```
            charges ------------
#-----------------------------------------
# Ref face | Type de charge   |    valeurs |
#-----------------------------------------
   A_34      LINEIC_SUIVEUSE             0. 0.2
```

Comme pour le chargement de type " PONCTUELLE " il est possible d'utiliser un champ de vecteur lorsque l'on veut grouper un ensemble de charges différentes. La syntaxe pour le champ, est identique au cas du chargement PONCTUELLE (cf.258).

NB : Ne pas oublier que l'on peut rajouter si on le souhaite, à la suite de la description du chargement : une courbe de charge globale (cf. 66.1.2) ou une fonction nD (cf. 66.1.3), un facteur d'échelle, un temps mini et un temps maxi.

66.2.5 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 : (266).

Comme pour le chargement de type " PONCTUELLE " il est possible d'utiliser un champ de vecteur lorsque l'on veut grouper un ensemble de charges différentes. La syntaxe pour le champ, est identique au cas du chargement PONCTUELLE (cf.258).

NB : Ne pas oublier que l'on peut rajouter si on le souhaite, à la suite de la description du chargement : une courbe de charge globale (cf. 66.1.2) ou une fonction nD (cf. 66.1.3),

TABLE 266 – Exemple de déclaration d'un chargement uniforme surfacique

```
            charges ------------
 #-----------------------------------------
 # Ref face | Type de charge   |    valeurs |
 #-----------------------------------------
F_02      UNIFORME             0. 0.  2.
```

un facteur d'échelle, un temps mini et un temps maxi.

66.2.6 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 : (267).

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

$$P = -\boldsymbol{I_\sigma}/dim$$

avec dim la dimension de l'espace de travail.

TABLE 267 – Exemple de déclaration d'un chargement pression

```
            charges ------------
 #-----------------------------------------
 # Ref face | Type de charge   |    valeurs |
 #-----------------------------------------
    F_02      PRESSION            8.
```

Comme pour le chargement de type " PONCTUELLE " il est possible d'utiliser un champ de scalaires lorsque l'on veut grouper un ensemble de pressions différentes. La syntaxe pour le champ, est identique au cas du chargement PONCTUELLE (cf.258), par contre il faut noter ici que ce ne sont pas des vecteurs que l'on indique, mais des pressions, c'est donc un champ de scalaires que l'on indique.

NB : Ne pas oublier que l'on peut rajouter si on le souhaite, à la suite de la description du chargement : une courbe de charge globale (cf. 66.1.2) ou une fonction nD (cf. 66.1.3), un facteur d'échelle, un temps mini et un temps maxi.

66.2.7 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 initiales 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 : (268).

TABLE 268 – Exemple de déclaration d'un chargement de type surfacique directionnelle

```
            charges ------------
#-----------------------------------------
# Ref face | Type de charge   |    valeurs |
#-----------------------------------------
   F_02      PRESSDIR            1. 1. 0.
```

Comme pour le chargement de type " PONCTUELLE " il est possible d'utiliser un champ de vecteur lorsque l'on veut grouper un ensemble de charges différentes. La syntaxe pour le champ, est identique au cas du chargement PONCTUELLE (cf.258).

NB : Ne pas oublier que l'on peut rajouter si on le souhaite, à la suite de la description du chargement : une courbe de charge globale (cf. 66.1.2) ou une fonction nD (cf. 66.1.3), un facteur d'échelle, un temps mini et un temps maxi.

66.2.8 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'un plan de normale $\vec{n}$, 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 A de la surface libre du liquide, le poids volumique du liquide (ρg du liquide). La pression en un point M se détermine à partir de la distance de M au plan de référence : $x_n = A\vec{M}.\vec{n}$, selon l'expression : $p(x_n) = -x_n \times \rho g$ si $x_n < 0$ et $p(x_n) = 0$ si $x_n \geqslant 0$. Exemple de syntaxe : (270). La pression calculée n'est prise en compte que pour $x_n < 0$.

En fait d'une manière plus générale, le chargement de type PHYDRO peut représenter tout chargement en pression, qui évolue de manière linéaire en fonction d'une distance perpendiculairement à un plan. Il est parfois intéressant de ne pas limiter la pression prise en compte aux x_n négatifs. Si on veut supprimer cette limitation, on indique le mot clé : " sans_limitation_ " à la suite d'un autre mot clé " ATTRIBUT_ ".

Il est également possible d'indiquer des fonctions d'évolutions pour les composantes de la normale ou du point libre. De même on peut indiquer une courbe de charge pour le

TABLE 269 – 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.
```

TABLE 270 – Exemple de déclaration d'un chargement en pression avec une évolution linéaire, sans limitation de position (noter le caractère de continuation l'antislash, pour l'écriture sur 2 lignes)

```
                    charges ------------
#-------------------------------------------------------------------------------
# Ref face | Type de charge | direction normale | point surf libre | coef hydro |
#-------------------------------------------------------------------------------
   F_02        PHYDRO          0. 0. 1.           0. 0. 0.           4. \
   ATTRIBUT_  sans_limitation_
```

poids volumique. La table (271) présente un exemple d'utilisation. Dans cet exemple, la coordonnée x de la normale est une fonction de nom "c_Nx", qui doit avoir été définie dans la liste des courbes, de même la coordonnée x du point de la surface libre via la fonction "c_Ax". En fait chaque coordonnée peut-être remplacée par une fonction selon la syntaxe : le mot clé " nom_Xi_ " suivi d'un nom de courbe 1D. Au moment de l'exécution, les fonctions seront appelées et évaluées en fonction du paramètre "temps". Dans le cas où la normale n'est pas normée, elle est normée pendant l'exécution.

Concernant le coefficient hydro (= le poids volumique), le fonctionnement est un peu différent. La valeur utilisée par le programme est le produit du coefficient fixe lu (dans l'exemple 0.01) et de la fonction de charge évaluée en fonction du paramètre "temps". Par exemple si le coefficient fixe vaut 1, la valeur utilisée dans le programme sera directement la valeur de la courbe de charge.

NB : Ne pas oublier que l'on peut rajouter si on le souhaite, à la suite de la description du chargement : une courbe de charge globale (cf. 66.1.2) ou une fonction nD (cf. 66.1.3), un facteur d'échelle, un temps mini et un temps maxi.

66.2.9 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 :

TABLE 271 – Exemple de déclaration d'un chargement en pression avec une évolution linéaire, sans limitation de position et avec l'utilisation de fonctions dépendantes du temps

```
    charges
#-------------------------------------------------------------------------------
# Ref face | Type de charge | direction normale | point surf libre | coef hydro |
#-------------------------------------------------------------------------------
F_haut     PHYDRO           nom_Xi_ c_Nx    0.     0.    \
                            nom_Xi_  c_Ax    0.     0.    \
                            0.01 ATTRIBUT_  sans_limitation_ \
                            COURBE_CHARGE: courbe1
```

— les forces dues à un frottement fluide visqueux,

— les forces aérodynamiques.

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
$$\vec{F_n} = \omega f_n(V) S(\vec{n}.\vec{u})\vec{u}$$
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
$$\vec{F_t} = \omega f_t(V) S(\vec{n}.\vec{u})\vec{w}$$
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 $\vec{V_t} = \vec{V} - (\vec{V}.\vec{n})\vec{n}$ 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 (272) 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 272 – 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
```

NB : Ne pas oublier que l'on peut rajouter si on le souhaite, à la suite de la description du chargement : une courbe de charge globale (cf. 66.1.2) ou une fonction nD (cf. 66.1.3), un facteur d'échelle, un temps mini et un temps maxi.

66.2.10 torseur d'efforts via une répartition de charges ponctuelles

L'idée est d'imposer un torseur d'effort via des forces ponctuelles réparties sur un ensemble de noeuds. Le mot clé correspondant est TORSEUR_PONCT et différentes possibilités sont disponibles.

1. On commence par définir une référence de noeuds qui supporteront les charges ponctuelles calculées pour imposer le torseur.
2. Puis on indique le mot clé TORSEUR_PONCT ,
3. On définit la résultante du torseur via le mot clé Re= suivi :
   - soit des coordonnées de la résultante
   - soit du mot clé Fonction_nD_Re: puis d'un nom de fonction nD déjà définie qui doit renvoyer un vecteur. Au cours du calcul, la fonction nD sera utilisée pour déterminer la valeur de la résultante à imposer.
4. On définit le moment du torseur via le mot clé Mo= suivi :
   - soit des coordonnées du moment
   - soit du mot clé Fonction_nD_Mo: puis d'un nom de fonction nD déjà définie qui doit renvoyer un vecteur. Au cours du calcul, la fonction nD sera utilisée pour déterminer la valeur du moment à imposer.
5. On définit le point "O' " par rapport auquel le torseur est donné, via le mot clé centre_= suivi :
   - du mot clé centre_noeud_= qui indique que le point "O' " est coïncidant avec une position de noeud. Le mot clé est suivi :
     - soit d'un numéro de noeud s'il y a qu'un seul maillage

— soit, s'il y a plusieurs maillages, du mot clé `nom_mail=` puis un nom de maillage puis un numéro de noeud

Puis on indique le temps pour lequel on retient les coordonnées du noeud via un des 3 mots clés suivant :

— `TEMPS_0`

— `TEMPS_t`

— `TEMPS_tdt`

— soit du mot clé `centre_gravite_ref_=` . Là le centre sera le centre de gravité des noeuds d'une référence qui peut-être différente de la ref sur laquelle s'applique le torseur. Pour ce faire on indique :

— soit le nom d'une référence de noeuds s'il y a qu'un seul maillage

— soit, s'il y a plusieurs maillages, du mot clé `nom_mail=` puis un nom de maillage puis le nom d'une référence de noeuds.

Puis on indique le temps pour lequel on retient les coordonnées des noeuds via un des 3 mots clés suivant :

— `TEMPS_0`

— `TEMPS_t`

— `TEMPS_tdt`

— soit du mot clé `Fonction_nD_centre_:` puis d'un nom de fonction nD déjà définie qui doit renvoyer un vecteur. Au cours du calcul, la fonction nD sera utilisée pour déterminer les coordonnées de "O' ", avant chaque imposition du torseur.

— soit des coordonnées du point.

6. On définit éventuellement la fonction de pondération avec le mot clé `Fonction_nD_Poids:` suivi du nom d'une fonction nD qui doit renvoyer un scalaire. Dans le cas où le mot clé n'existe pas, tous les poids sont égaux à 1.

   En appelant Ce le centre de référence du torseur, et M_i la position géométrique du noeud sur lequel va s'appliquer l'effort ponctuel, on calcule le vecteur $C_e\vec{M}_i$ et sa norme $r = ||C_e\vec{M}_i||$. La fonction nD reçoit comme argument :

   — la norme "r"

   — successivement les composantes du vecteur $C_e\vec{M}_i$

   Ainsi en dimension 3, la fonction reçoit 4 arguments scalaires.

   Si dim est la dimension de l'espace de travail, la fonction nD doit posséder 1+dim arguments. Par contre dans la définition de la fonction, ces 1+dim arguments peuvent avoir une dénomination locale arbitraire. La table 273 donne un exemple de déclaration de fonction poids.

   Ainsi on peut définir par exemple une pondération qui augmente (ou diminue, ou varie de manière donnée) en fonction de ses arguments.

L'application du torseur d'effort s'effectue via une répartition d'efforts ponctuelles qui est calculée à chaque fois que le torseur doit être appliqué.

On se reportera à la documentation théorique d'Herezh++, pour avoir plus de détail sur la méthode utilisée pour calculer cette répartition d'efforts ponctuelles.

TABLE 273 – Exemple de déclaration d'une fonction nD de pondération pour un torseur d'effort

```
  fpoids  FONCTION_EXPRESSION_LITTERALE_nD
   # en plus des arguments specifiques, on utilise une variable globale
   deb_list_var_   r x y z temps_courant fin_list_var_
   fct= cos(y*3.1416/20.) * temps_courant # ici r n'est pas utilise
   fin_parametres_fonction_expression_litterale_
```

TABLE 274 – Exemple de déclaration simple d'un torseur de charge correspondant à l'application d'une résultante excentrée par rapport au centre du torseur

```
       charges #------------#
N_gauche TORSEUR_PONCT  Re= 2 0 0  Mo= 0 0 0  centre_= 0 0 0
```

TABLE 275 – Exemple de déclaration d'un torseur de charge dont la résultante dépend d'une fonction nD et qui utilise une fonction de pondération de poids de type sinusoïdal

```
        fpoids  FONCTION_EXPRESSION_LITTERALE_nD
   deb_list_var_   r x y z temps_courant fin_list_var_
   fct= cos(y*3.1416/20.) * temps_courant # ici r n'est pas utilise
   fin_parametres_fonction_expression_litterale_
 .....
       charges #------------#
N_gauche TORSEUR_PONCT  Re= Fonction_nD_Re: fRe  Mo= 0 0 0 \
 centre_= centre_gravite_ref_= N_gauche TEMPS_t \
 Fonction_nD_Poids: fpoids
```

TABLE 276 – Exemple de déclaration d'un torseur de charge dont le moment dépend d'une fonction nD

```
  fMo  FONCTION_EXPRESSION_LITTERALE_nD
   un_argument= temps_courant
   fct= temps_courant * 1.e6 , 0, 0
   fin_parametres_fonction_expression_litterale_
 .....
       charges #------------#
N_gauche TORSEUR_PONCT  Re= 0 0 0  Mo= Fonction_nD_Mo: fMo \
 centre_= centre_gravite_ref_= N_gauche TEMPS_t
```

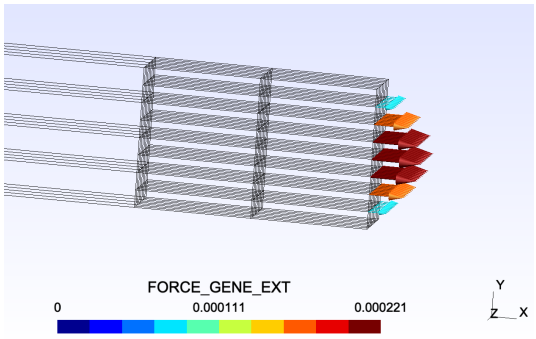


FIGURE 1 – exemple d'une résultante de répartition sinusoïdale dans l'épaisseur d'une poutre (cf. 275)

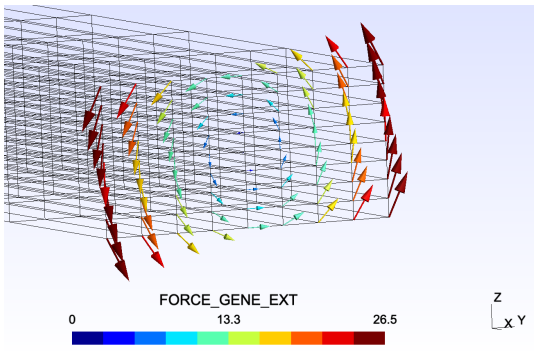


FIGURE 2 – exemple d'un moment dépendant d'une fonction nD (cf. 276)

66.2.11 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 suiveuse (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 : $\int_{ligne} q_l \ 2 \ \Pi \ rdl$
- Il est possible d'indique 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 : $\int_{surface} q_s \ 2 \ \Pi \ rds$
- 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 :

$$\text{résidu}(\overset{*}{V}_{ar}) = \sum_{i=1}^{i=npti} \vec{F}.\vec{I}_a \varphi_r \sqrt{g(i)} poids_i \tag{122}$$

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 dû 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 calcule. 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 : $ds = dr.r.2.\pi$ 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 points d'intégration 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

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

Les conditions limites sur les degrés de liberté consistent à imposer soit une valeur numérique aux ddl concernés, soit une fonction de charge. Ceci s'effectue à l'aide du mot clé "blocage", cf. l'exemple (277).

TABLE 277 – Exemple de déclaration des blocages des degrés de liberté dans le cas d'un seul maillage

```
             blocages ------------
#-------------------------------------------------------------
# Ref noeud |  Blocages
#-------------------------------------------------------------
   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 liberté 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 liberté 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 spécifiée entre cotes ex : ' `UX=` 2.'.

L'application d'un blocage non nul sans courbe de charge s'effectue en pratique, suivant la courbe de chargement globale (cf.70). L'application du chargement s'effectue par incrément, qui s'ajoute 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 "$t + \Delta t$" 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 279).

Remarque : Attention ! , le blocage relatif ne fonctionne que dans le cas d'utilisation de courbe de charge individuelle. Donc, ne fonctionne pas avec les valeurs fixes ! !

NB : Sur certain clavier il y a deux types de simples cotes, ' et ‘ . Pour trouver le bon symbole, il n'y a que l'essai !

Dans le cas où il y a plusieurs maillages, il est nécessaire (obligatoire) d'indiquer avant la référence, le nom du maillage auquel cette référence est attachée (cf.(278)

TABLE 278 – 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 279 – Exemple de déclaration de degrés de liberté imposés 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. BLOCAGE_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 maillages, il est obligatoire d'ajouter la définition du nom de maillage

67.1 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. 46). 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 définitive 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.70) n'intervient pas dans le calcul, par contre le coefficient multiplicatif global (mot clé `MULTIPLICATEUR` cf.296) lui intervient.

La table 277 donne un exemple d'utilisation de courbe de charge.

67.1.1 Chargement particulier suivant une fonction nD

De manière analogue à l'utilisation d'une courbe de charge 1D pour piloter la condition cinématique, il est possible d'utiliser une fonction nD. Dans ce cas la valeur du ddl imposé peut dépendre de toutes les variables globales qui existent pendant le calcul, et également des grandeurs disponibles au noeud où on calcule le chargement.

La syntaxe est identique au cas 67.1 avec comme différence le mot clé `COURBE_CHARGE:` remplacé par le mot clé `Fonction_nD_CHARGE:` . La table 280 donne un exemple de déclaration.

Dans la définition de la fonction nD, il est possible de choisir comme paramètre, les grandeurs existantes au noeud au moment du calcul :

- les degrés de liberté (ex : `UX UY UZ`
- les positions (ex : `X1 X2 X3`)
- en dynamique les vitesses et accélérations,
- les données qui ont été introduites au noeud via la mise en données,

et également toutes les grandeurs globales disponibles au moment de l'utilisation de la fonction nD.

On se reportera à 88.1 pour une liste de grandeurs habituellement disponibles. Mais il peut y avoir d'autres grandeurs globales qui apparaissent pour un calcul particulier : par exemple à chaque fois que l'on introduit des intégrales à calculer. On se reportera à 61 et 84 pour une information plus détaillée. Enfin lorsque l'on effectue un calcul avec " `plus visualisation` " cf. 80.3, on peut obtenir en interactif, la liste des grandeurs globales disponibles, pour le calcul en cours.

TABLE 280 – Exemple de déclaration des blocages des degrés de liberté à l'aide d'une fonction nD

```
             blocages ------------
#-------------------------------------------------------------
#  nom du maillage | Ref noeud |  Blocages
#-------------------------------------------------------------
nom_mail= marteau      N_0  'UX= Fonction_nD_CHARGE: fct1 ECHELLE: 1.2' TEMPS_MINI= 0. TEMPS_MAXI= 3.
```

Remarque : Il n'est pas possible d'utiliser un blocage relatif avec seulement une fonction nD. Si l'on veut un blocage relatif avec une fonction nD, il est nécessaire d'utiliser conjointement avec la fonction nD, une courbe 1D. Le résultat de la condition imposée est alors le produit de la courbe 1D avec la fonction nD. Pour que la condition soit correcte, il ne faut pas que la fonction nD utilise la variable temps (i.e. dépende du temps). **Attention il n'y aura pas de message d'erreur si la fonction nD est néanmoins dépendante du temps** . La table 281 donne un exemple d'utilisation.

TABLE 281 – Exemple de déclaration des blocages des degrés de liberté à l'aide d'une fonction nD

```
             blocages ------------
#-------------------------------------------------------------
#  nom du maillage | Ref noeud |  Blocages
#-------------------------------------------------------------
 N_O  'UX= Fonction_nD_CHARGE: fct1 COURBE_CHARGE: c1 ECHELLE: 1.2' TEMPS_MINI= 0. TEMPS_MAXI= 3.
```

67.2 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 cet intervalle, le noeud sera considéré libre de conditions limites. Par défaut cet 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=` suivit de la valeur du temps maxi. En dehors de cet intervalle, le degré de liberté est libre. Bien noter que la borne inférieure est exclue 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 signale l'erreur en warning. Cette erreur peut survenir par exemple lorsque l'on indique plusieurs conditions 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 munie du coefficient d'échelle et du coefficient multiplicateur global. Dans le cas on l'on indique également une condition initiale sur le noeud (cf.69), il faut veiller à ce qu'elle soit cohérente avec la condition d'évolution pour éviter des à-coups.

67.3 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 grandeur, 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 utilisés, ces noeuds sont ignorés.
2. soit au moment de l'initialisation certains objets (éléments finis, lois de comportement ...) indiquent qu'ils vont utiliser les grandeurs, dans ce cas les grandeurs sont

mises en service.

Ainsi durant le déroulement du calcul, seules les grandeurs en service sont utilisées. 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.

67.4 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 (282). Dans cet exemple N_tout est une 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 noeuds. 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 282 – Exemple de déclaration des blocages des degrés de liberté dans le cas d'un champ de valeurs fixes

```
#------------------------------------------------------------
# Ref noeud |  Blocages
#------------------------------------------------------------
   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 général de mise en place du chargement.

Il est également possible de définir un champ de valeurs qui utilisent chacune une courbe de charge. Le tableau (283) donne un exemple de ce type de déclaration. Dans cet exemple les courbes de nom c_temp_1 et c_temp_2 doivent au préalable avoir été évidemment définies.

67.5 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

TABLE 283 – Exemple de déclaration des blocages des degrés de liberté dans le cas d'un champ de valeurs déterminées par des courbes de charge

```
#-------------------------------------------------------------
# Ref noeud |  Blocages
#-------------------------------------------------------------
   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.
```

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 se 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 (69.1) 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 :

$$(200°C + t.100°C) - (200°C) = t.100°C$$

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!!

67.6 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 (284) donne un exemple de mouvement solide global. La syntaxe suit la même logique que pour les mouvements initiaux appliqués aux maillages (cf. 29), on s'y reportera pour plus d'information sur la syntaxe et les différentes possibilités. 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_` "

Il est également possible de définir une courbe de charge globale et/ou une fonction nD, qui agira sur l'amplitude des différentes opérations que l'on a définies. 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

TABLE 284 – Exemple de déclaration d'un déplacement imposé par mouvements solides

```
#-------------------------------------------------------------
# Ref noeud |  Blocages
#-------------------------------------------------------------
 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_
```

le mot clé " fin_mouvement_solide_ " selon la procédure habituelle et en respectant l'ordre. La table (285) donne un exemple complet.

TABLE 285 – Exemple de déclaration d'un déplacement imposé par mouvements solides

```
#-------------------------------------------------------------
# Ref noeud |  Blocages
#-------------------------------------------------------------
   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.
```

67.7 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 encastrées plus une condition de direction imposée à la tangente de surface. Ces conditions peuvent également être imposées via les conditions initiales (cf.69), 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.

67.7.1 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 (286) 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 286 – Exemple de déclaration d'une direction de tangente imposée à une surface moyenne d'une coque SFE

```
#-------------------------------------------------------------
# Ref aretes |  mot cle     | Blocages
#-------------------------------------------------------------
  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.

NB : Actuellement, les conditions de tangence ne sont opérationnelles que pour les éléments `SFE3` . Pour les autres éléments SFE, il faut définir une bande d'éléments supplémentaires, au niveau de la frontière où l'on veut imposer une tangence. La bande d'éléments supplémentaires est positionnée pour définir le plan de tangence voulu.

67.7.2 Encastrement imposé

Comme il a été indiqué 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 !

67.8 Cas particulier de géométries et chargements axisymétrique

Dans le cas de géométries et chargements axisymétriques, la dimension d'espace a 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.

68 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 liberté existants. 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 leurs 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 liberté 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 groupées 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 conditions limites classiques et les conditions initiales.

68.1 Déplacement ou positionnement dans un plan (3D) ou sur une droite (2D)

Pour fixer les idées, prenons un exemple (287).

TABLE 287 – 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 maillages 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 évoluent 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 obligatoires " 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 (288).

TABLE 288 – 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_
```

Pour la première condition, on a choisi 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.

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. À part ces deux particularités, le reste du fonctionnement est identique au cas 3D.

Par comparaison au cas des conditions linéaires générales (cf.68.2), 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.68.2), voir l'exemple : (289),
- Il n'est pas possible d'indiquer que la condition est relative (comme dans le cas général).
- Il n'est pas possible d'indiquer des références secondaires (comme dans le cas général), car cela n'a aucune signification ici.

— Il n'est pas possible d'indiquer une condition supplémentaire (ddl bloqué ou CLL supplémentaire) pour un noeud soumis à une condition de projection. Pourtant, a priori ce type de surcharge de blocage est certaines fois possible. Si on désire superposer plusieurs conditions, il faut alors utiliser les conditions linéaires générales qui elles permettent la surcharge. En résumé, on retiendra donc que les CLL générales sont moins pratiques pour représenter des projections, mais par contre elles sont plus versatiles.

TABLE 289 – 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.

68.2 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 290 – 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 (290) 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 toutes les trois de la famille des déplacements. À noter qu'une condition linéaire sur " UX " est différente d'une condition linéaire sur " X1 ", la première concerne des déplacements, la seconde des positions.

Vient 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 associée 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 seul 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 précédé 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 précédée 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 coefficients est égal au nombre de références c'est-à-dire le nombre de références associées + 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(du noeud "i" de "N_deb") + 1*UX(du noeud "i" de "N_fin") = 0 , ceci pour i=1 au nombre total de noeuds de chaque référence

Comme dans le cas des conditions limites (68.1) 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 après le mot clé " fin_list_coefficients_ "on indique le mot clé " AvecFonctionsDeCharges_ " (cf. 291 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 encadrée 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 ième 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 prévisible a priori. Par contre les coefficients, fixes ou dépendants de fonctions de charge, sont connus à l'avance, donc pour ces dernières il n'y a aucun intérêt de prévoir une procédure particulière à l'option "relatif".

TABLE 291 – 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(noeud 1) + UX(noeud 2) = 2." avec un facteur d'échelle unitaire (valeur par défaut), cette relation sera valide, quelsque soient 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 ce 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 coefficients dépendent d'une fonction de charge, ils dépendent donc du temps (ou du paramètre d'avancement) de chargement. Que ce soit une condition absolue (situation par défaut) ou relative, la valeur des coefficients 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 calculée à l'aide des paramètres de la condition. Cet accroissement s'ajoute alors à la valeur à t du ddl.

Il est possible de superposer un grand nombre de conditions 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, modélisée par un élément linéaire à 2 noeuds, et soumise à une force de traction au noeud 2. Supposons de plus la condition linaire : "U1(noeud 1) - U1(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 "U1(noeud 1) + U1(noeud 2) = 0" avec comme noeud principal 1 conduit à un problème bien posé et ne pose aucun problème de résolution.

68.2.1 Particularités liées aux noms de maillage

Lorsqu'il n'y a qu'un seul 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 secondaire. 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. À 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 utilisé.

La table (292) donne un exemple d'utilisation de nom de maillage.

TABLE 292 – 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_
```

68.3 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++ calcule 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 optimale, 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. 7) supérieur ou égal 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 accélérations (la seule pour l'instant mise au point et testée). Si la matrice de masse demandée est de type diagonal, 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. À 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.

On se reportera pour plus d'information sur les conséquences sur le stockage matriciel aux remarques en fin de chapitre de : 75 et 73.

69 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 déplacements (ou autre degré de liberté) imposés. Le mot clé à utiliser est : " initialisation ".

L'exemple de la table (293) montre comment implanter des conditions initiales lorsqu'il n'y a qu'un seul maillage.

TABLE 293 – Exemple de mise en place de conditions initiales dans le cas d'un seul maillage

```
  initialisation -----------
#-------------------------------------------------------------
# Ref noeud |  Blocages
#-------------------------------------------------------------
   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 (69.1) 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 maillages, 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 (294) montre comment implanter des conditions initiales lorsqu'il y a plusieurs maillages.

TABLE 294 – Exemple de mise en place de conditions initiales dans le cas de plusieurs maillages

```
  initialisation -----------
#-------------------------------------------------------------------
#     nom du maillage  |  Ref noeud |          Blocages              |
#-------------------------------------------------------------------
   nom_mail= piece    N_tout           'V1=1000.'  # vitesse initiale
   nom_mail= outil    N_2            'X1=41.'  # position initiale
```

Dans le cas de la dynamique on se reportera à (18) pour des précisions supplémentaires.

69.1 Particularité de l'initialisation des positions par rapport aux autres degrés de liberté

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 Xi, il y a en fait initialisation des Xi 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 liberté, 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 (67.5) pour plus d'information sur le fonctionnement).

Dans le cas de la dynamique on se reportera à (18) pour des précisions supplémentaires.

69.2 Conditions de symétrie ou d'encastrement initial

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 (67.7) pour sa description précise.

Onzième partie

Chargement global

70 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 à convergé le temps de calcul est incrémenté sinon on diminue le temps en espérant ainsi faciliter le calcul.

Une première utilitée 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
```

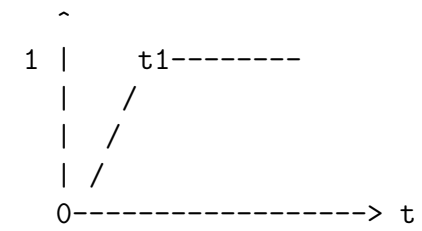
La liste exhaustive des algorithmes de chargement disponibles est donné dans la table (295).

TABLE 295 – 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 :

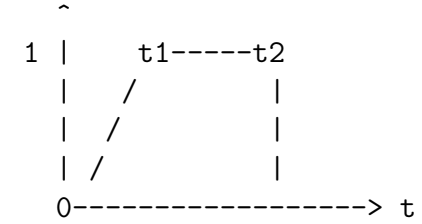
1. TYPE1 : le chargement correspond au schéma suivant :



Du temps t=0 à t=t1, le chargement évolue de manière linéaire pour atteindre la valeur finale au temps t1, en ordonnée est indiqué le facteur multiplicatif global du chargement. A partir du temps t1, 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 :

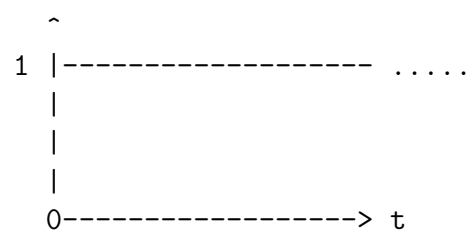


Du temps t=0 à t=t1, le chargement évolue de manière linéaire pour atteindre la valeur finale au temps t1, en ordonnée est indiqué le facteur multiplicatif global du chargement. A partir du temps t1, le chargement demeure constant jusqu'au temps t2, puis tout le chargement est suprimé de manière instantanée, 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 t1 grand. De t1 à t2 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. 66.1.5). 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.308). 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.308). 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 représentes 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

71 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 (72) : liste exhaustive (296) ,
- les paramètres dédiés à la dynamique (73) : liste exhaustive (297) ,
- les paramètres de contrôle liés aux calculs des énergies (74) : liste exhaustive (298) ,
- les paramètres de gestion de la résolution du système linéaire (75) : liste exhaustive (299) ,
- les paramètres de contrôle du pilotage de la résolution globale (76) : liste exhaustive (300) ,
- les paramètres liés au pilotage du contact (77) : liste exhaustive (301) ,
- les paramètres liès à l'affichage des résultats (78) : liste exhaustive (302) ,
- les paramètres liès à certains calculs géométriques (79) : liste exhaustive (303) .

La plupart des paramètres sont optionnels, les valeurs par défaut essayant de répondre aux cas les plus courants.

Remarques :

- Tout paramètre réel, entier ou booléen peut être remplacé par le nom d'une constante utilisateur (cf.4). C'est la valeur de la constante utilisateur au moment de la lecture du paramètre de contrôle, qui est utilisée pour renseigner le paramètre de contrôle.
- Les groupes de paramètres de contrôle peuvent également être encapsulés entre les deux mots clés :
  `==PARA_SPECIFIQUE_SOUS_ALGO==`
  et
  `==FIN_PARA_SPECIFIQUE_SOUS_ALGO_==`
  Ces 2 mots clés permettent de définir des paramètres de contrôle spécifiques pour un sous-algorithme de l'algorithme `combiner` (cf.24). Ils doivent être indiqués après la définition des paramètres de contrôle généraux (qui sont les paramètres par défaut de tous les sous algorithmes). Il s'agit ici d'une déclaration de paramètres, particulière à l'algorithme `combiner` et qui provoquera une erreur dans le cas de l'utilisation d'un autre algorithme générale de calcul.

72 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             0.1
#MULTIPLICATEUR     10000.
ITERATIONS        200
PRECISION          1e-3
#RESTART            1
```

La liste exhaustive des sous-mots clés disponible est donnée dans la table (296).

TABLE 296 – liste des sous-mots clés associés aux paramètres de contrôle

sous mot clé	valeur par défaut	ref du commentaire
SAUVEGARDE	1	(1)
DELTAt	1.	(2)
DELTAtMAXI	1.	(3)
DELTAtMINI	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)
MAX_ESSAI_INCRE	MAXINCRE	(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)

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 transformé 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'incréments 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 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'itérations 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.

Il est également possible de combiner une sauvegarde " INTER_TEMPS " ou " INTER_INCREMENT " et " DERNIER_CALCUL ". Dans ce cas on indique les deux mots clés. Par exemple pour une sauvegarde tous les 20 incréments et une sauvegarde finale on indiquera :

```
SAUVEGARDE   INTER_INCREMENT 20 DERNIER_CALCUL
```

Pour une sauvegarde tous les 0.1 seconde et une sauvegarde à la fin du calcul on indiquera :

```
SAUVEGARDE   INTER_TEMPS 0.1 DERNIER_CALCUL
```

2. DELTAt <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 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 : DELTAt on indique COEF_PASCRITIQUE suivi de la valeur du coefficient α. Dans ce cas le pas de temps *Deltat* 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 donnée 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 72). 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_PASCRITIQUE_DFC à la place de COEF_PASCRITIQUE . L'avantage de ce pas de temps critique est qui sera identique, quelque soit l'algorithme d'avancement temporel utilisé.

3. DELTAtMAXI <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 DELTAtMAXI . 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 : DELTAtMAXI on indique COEF_PASCRITIQUE suivi de la valeur du coefficient α. Dans ce cas le pas de temps $Deltat_{maxi}$ est évalué en début d'algorithme puis mis à jour à chaque changement de pas de temps critique. De manière identique au paramètre DELTAt , on peut également utiliser le pas de temps critique de la méthode DFC en indiquant COEF_PASCRITIQUE_DFC à la place de COEF_PASCRITIQUE .
4. DELTAtMINI <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 DELTAtMINI . 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 : DELTAtMINI on indique COEF_PASCRITIQUE suivi de la valeur du coefficient α. Dans ce cas le pas de temps $Deltat_{mini}$ est évalué en début d'algorithme puis mis à jour à chaque changement de pas de temps critique. De manière identique au paramètre DELTAt , on peut également utiliser le pas de temps critique de la méthode DFC en indiquant COEF_PASCRITIQUE_DFC à la place de COEF_PASCRITIQUE .
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'itérations 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. Il est possible de moduler la précision voulue à l'aide d'une fonction nD. On se reportera à (20.3.12) pour plus de précision de cette option.
9. NORME <une chaine de caractère > : les choix possibles sont :
   - Residu : correspond à une norme infinie 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.

— `min(Res,Res/MaX(Reaction_et_PVExterne))` : 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 et des puissances virtuelles externes c'est-à-dire les forces externes.

— `E_cinetique/E_statique_ET_ResSurReact` : norme utilisable **uniquement** avec un algorithme dynamique. Correspond au maxi d'une part de : l'énergie cinétique divisée par l'énergie interne plus l'énergie externe, et d'autre part : du résidu divisé par le maxi des réactions.

— `E_cinetique/E_statique_ET_Res/Reac_et_Fext` : norme utilisable **uniquement** avec un algorithme dynamique. Correspond au maxi d'une part de : l'énergie cinétique divisée par l'énergie interne plus l'énergie externe, et d'autre part : du résidu divisé par le maxi d'une part des réactions et d'autre part des puissances externes et internes. Cette norme est intéressante, lorsqu'il n'y a pas de réaction aux appuis. Par exemple, dans le cas d'une structure soumise à des forces internes et simplement posée, sans poids.

— `E_cin/E_stat_ET_min(Res,Res/Reac_et_Fext`) : norme utilisable **uniquement** avec un algorithme dynamique. Correspond au maxi d'une part de : l'énergie cinétique divisée par l'énergie interne plus l'énergie externe, et d'autre part : du minimum du résidu et du résidu divisé par le maxi d'une part des réactions et d'autre part des puissances externes et internes. Cette norme est très proche de la précédente. Elle permet de plus, lorsque les forces externes et les réactions sont très faibles (lors de la fin d'un déchargement par exemple), de retenir seulement le résidu.

— `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.

— `fonction_nD:` suivi du nom d'une fonction nD < fct_norme > : permet de définir une norme particulière dépendant de variables globales (cf. 88.1) à travers l'utilisation d'une fonction nD (cf. 81.2). La fonction nD doit ramener un réelle = fct_norme() qui constituera la norme à atteindre pour valider l'équilibre vis-à-vis du résidu global de l'équation d'équilibre. La fonction est appelée avant chaque vérification de l'équilibre (lorsqu'il y en a une car en dynamique explicite classique, il n'y a pas de vérification d'équilibre car l'équilibre est toujours satisfait via le calcul d'une accélération ad hoc). Le type de calcul et la valeur

de la norme d'équilibre peut ainsi évoluer au cours du calcul d'équilibre au gré de l'objectif de l'utilisateur.

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. 76) 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 réactions 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 rajoutée 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 a 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.

Doublement du contrôle du résidu à convergence : Lorsqu'on utilise des grandeurs qui évoluent pendant les itérations de convergence, et qu'on veut s'assurer qu'une fois la convergence obtenue, l'ensemble des grandeurs qui pilotent l'évolution ont bien la valeur nécessaire à convergence, une solution est de revérifier la norme de convergence en effectuant une itération suplémentaire. Pour cela on peut rajouter la chaîne : `et_verification_ddouble` . Par exemple : supposons qu'on utilise une loi des mélanges qui est pilotée par le niveau de précision obtenu à chaque itération, il est utile de prévoir un doublement du calcul du résidu final pour vérifier que l'équilibre utilise bien une précision suffisante au niveau du pilotage.

Remarque : Il est possible de cumuler tous les contrôles et cela dans n'importe quel ordre. Par exemple les deux mots clés suivants sont corrects :
" `Residu_et_VarRes_et_miniVarDdl_et_maxiVarDdl` " ,
" `Residu/Reaction_et_maxiVarDdl_et_VarRes_et_miniVarDdl` ".

Les contrôles sur les minis et maxi des variations de ddl dépendent 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 (76) et plus particulièrement au tableau (300) 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. 76) 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 autres d'éviter qu'un calcul ne s'arrête jamais. Le paramètre MAXINCRE définit le maximum d'incrément ayant fonctionné correctement. Un deuxième paramètre MAX_ESSAI_INCRE , qui par défaut est égal à MAXINCRE , permet de spécifier le nombre maximum d'essais d'incréments autorisés. Typiquement, MAX_ESSAI_INCRE est à définir avec une valeur a priori >= MAXINCRE . Au final le calcul s'arrêtera si une des deux conditions est remplie à savoir : soit le nombre d'incréments de calcul ayant fonctionné correctement est supérieur à MAXINCRE ou soit le nombre de tentatives de calculs d'incrément est supérieur à MAX_ESSAI_INCRE . Le nombre de tentatives déjà effectué est sauvegardé dans le .BI. Il est donc récupéré lors d'une opération de restart. Il est possible de changer la valeur des deux paramètres MAXINCRE et MAX_ESSAI_INCRE dans le .info pour que ces nouvelles valeurs soient prise en compte au cours d'un restart.
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 72).
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 active.
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 recalcul 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 retient ce pas de temps, sinon, il recalcul 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 dû 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'interventions.

73 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 donnée dans la table (297). 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 chaine de caratère> : 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 chaque noeud, 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 correct. L'expression définissant les termes de la matrice masse élémentaire est de la forme suivante :

$$m_i = \frac{1}{n} \int_D \rho dv \tag{123}$$

TABLE 297 – 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)

"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 noeuds de l'élément.

— MASSE_CONSISTANTE : Dans ce cas la matrice masse est calculée suivant la formule théorique

$$M(ar, bs) = \int_D \rho \varphi_r \varphi_s \delta^{ab} \sqrt{g} dv \tag{124}$$

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 continu. Conviens bien aux éléments linéaires, peut poser des problèmes dans le cas de polynômes de degrés plus élevés incomplets : par exemple dans le cas de l'utilisation d'éléments quadratiques incomplets. Par contre si l'élément est complet il n'y a a priori pas de problème.

— 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 :

$$m_i = \alpha \int_D \varphi_i^2 dv \ , \quad avec \quad \alpha = \frac{\int_D \rho dv}{\sum_{j=1}^{n} \int_D \rho \varphi_j^2 dv} \tag{125}$$

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 noeuds 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 dela 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 DELTAt et DELTAtMAXI . 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 : $\boldsymbol{\sigma}_{finale} = \boldsymbol{\sigma} - P\ \boldsymbol{I} = \boldsymbol{\sigma} - \rho l(C_1 l I_D^2 - C_2 c I_D)\ \boldsymbol{I}$. 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).

Remarque

— Par défaut, l'utilisation d'un type de calcul diagonal de la matrice masse conduit automatiquement à un choix d'une matrice de stockage de type diagonal. Dans le cas de la présence de conditions linéaires (ou non) cinématiques, le choix de la matrice est automatiquement modifié de la manière suivante :

— si la condition est prise en compte sous forme d'une modification de repères locaux, et que la condition ne concerne que les ddl d'un seul noeud → le nouveau stockage est de même type que celui indiqué pour la raideur (cf. 75) avec comme nombre de colonnes (quand cela a un sens), le nombre de ddl par noeud. Par défaut c'est un stockage bande.

— si la condition est prise en compte sous forme d'une modification de repères locaux, et que la condition concerne les ddl de plusieurs noeuds → le nouveau stockage est de même type que celui indiqué pour la raideur (cf. 75) avec comme nombre de colonnes (quand cela a un sens) une largeur de bande qui dépend de la numérotation des noeuds de la condition linéaire et de leur nombre. En général cela conduit très rapidement à une largeur de bande très importante ! Par défaut c'est un stockage bande symétrique qui est utilisée.

— voir également : 68.3 et 75.

74 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 généralisées. 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'intégration (grandeurs a post-traité comme les contraintes et les déformations). On remarquera que la somme de ces trois énergies est différente 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 :

$$energie_{interne} = \frac{(R_{int(t)} + R_{int(t+\Delta t)})}{2}.\Delta X \tag{126}$$

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 certain nombre 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 donnée dans la table (298). Bien noter que ces paramètres sont facultatifs,le groupe de paramètre est lui-même également facultatif.

Avec les commentaires suivants :

TABLE 298 – 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)

1. NB_INCR_CAL_ENERG <un entier> : indique à partie 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 booleen> : indique si de plus on veut l'affichage des différentes énergies sur l'incrément (indépendamment des énergies cumulées).

75 Résolution des systèmes linéaires

Ce jeu 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 disponibles est donnée dans la table (299).

TABLE 299 – 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)
MATRICE_S_SECONDAIRE_S_		(8)
TYPE_RESOLUTION_S_SECONDAIRE_S_		(9)
OPTIMISATION_POINTEURS_ASSEMBLAGE	0	(10)

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 colonnes = nombre de lignes = nombre de degrés de liberté du problème. Ce mode de stockage n'est pas en général économique en élément fini. De plus les méthodes de résolution du système linéaire ne sont pas optimisées dans Herezh. Il faut donc réserver ce mode de stockage pour des structures ayant un faible nombre de degré de liberté, ou pour des cas particuliers : des tests par exemple.
   - RECTANGLE : D'une manière théorique, ce stockage permet de différencier le nombre de colonnes et le nombre de lignes. 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 : $\frac{(M(i,j)+M(j,i))}{2} -> M(i,j)$.
   - BANDE_NON_SYMETRIQUE_LAPACK : la matrice de raideur sera stockée en bande complète. 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 certains cas de couplages ou de frottement par exemple.

L'intérêt ici est d'utiliser la bibliothèque Lapack associée avec la bibliothèque BLASS, 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.

— BANDE_SYMETRIQUE_LAPACK : le stockage est identique au cas BANDE_SYMETRIQUE , par contre ici on peut utiliser la parallélisation de la bibliothèque BLASS associée à LAPACK (à condition d'utiliser des bibliothèques parallélisées et optimisées).

— CARREE_LAPACK et RECTANGLE_LAPACK : sont deux modes de stockage équivalent à CARREE et RECTANGLE , mais ici optimisée avec LAPACK. En particulier quand la matrice de raideur est naturellement pleine, cela peut-être un choix intéressant.

— CARREE_SYMETRIQUE_LAPACK : matrice carrée symétrique utilisant LAPACK. Ici le stockage est réduit par rapport au CARREE_LAPACK .

— 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, seuls 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 chaine de caractères > : les choix possibles sont soit la méthode directe générale de Gauss, soit pour les matrices symétriques la méthode de Cholesky, soit les autres méthodes qui sont itératives de type gradient conjugué. Mais toutes les combinaisons ne sont pas possibles !
   — CHOLESKY : utilisable uniquement avec les matrices symétriques (stockées en symétrique). Corresponds 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. Donc les matrices concernées sont :
     — BANDE_SYMETRIQUE
     — BANDE_SYMETRIQUE_LAPACK
     — CARREE_SYMETRIQUE_LAPACK
     — CARREE en n'oubliant pas de mettre le paramètre SYMETRIE_MATRICE à 1

— GAUSS : utilisable uniquement avec certaines matrices LAPACK , il s'agit de la méthode classique "pivot de Gauss" et factorisation LU. Les matrices concernées sont :

— BANDE_NON_SYMETRIQUE_LAPACK

— CARREE_LAPACK

— DIAGONALE : stockage par défaut pour la matrice masse. A priori ne convient que pour ce cas qui conduit à une création automatique d'une matrice diagonale, quelque soit le type demandé, à condition que la matrice masse soit effectivement diagonale (donc hors CLL linéaires prise en compte par changement de repère local). C'est donc un cas particulier.

— Les méthodes itératives de type gradient conjugué. Théoriquement utilisable avec un grand nombre de types de matrices symétriques ou non. En fait dans la pratique, la difficulté et la performance de la méthode réside dans l'opération de préconditionnement qui précède l'opération de résolution. Pour l'instant seules quelques méthodes de préconditionnement sont implantées dans Herezh, cf. le paramètre TYPE_PRECONDITIONNEMENT . Les méthodes implantées proviennent de la bibliothèque SparseLib++ [**?**] et ont été étendues aux différents fonctionnements d'Herezh.

Les méthodes de type gradient conjugué disponibles sont :

— BI_CONJUG : Méthode du bi-gradient conjugué. Utilisable avec tous les types de matrices symétriques ou non.

— BI_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 Richarson, 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 chaine de caractères > : les méthodes itératives de type gradient conjugué nécessites 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 basique diagonal classique. En général l'efficacité est correcte sans plus ! Est disponible pour toutes les matrices.

— ICP : cas de la factorisation de cholesky incomplète. Ce préconditionnement est très efficace. Par contre pour l'instant il n'est disponible que pour les matrices CREUSE_COMPRESSEE_COLONNE

— ILU : cas de la factorisation LU : Ce préconditionnement est comme le précédent très efficace. Pour l'instant disponible uniquement pour les matrices CREUSE_COMPRESSEE_COLONNE .

5. NB_ITER_NONDIRECTE < un entier > : s'utilise avec les méthodes itératives, spécifie le nombre d'itérations 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'itérations 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_VECT_RESTART < un entier> : s'utilise avec la méthode GENE_MINI_RESIDUAL , spécifie le nombre de vecteurs maxi de restart que l'on accepte de stocker.
8. MATRICE_S_SECONDAIRE_S_ < une liste de type de matrice puis le mot clé > FIN_TYPE_MAT_SECONDAIRES_ :

   Permet de définir un ou plusieurs types de matrice de stockage secondaire. Dans le cas où la résolution du sytème linéaire à l'aide de la matrice principale, ne fonctionne pas (erreur de pivot par exemple, perte de positivité ...), automatiquement la matrice principale est remplacée par la première matrice secondaire. Si le calcul de nouveau échoue, c'est la seconde matrice secondaire qui est prise en compte etc. jusqu'à épuisement de la liste des matrices. Chaque substitution de matrice s'effectue sans recalculer les matrices locales, ceci pour optimiser le temps de calcul. Pour ce faire, les matrices secondaires sont remplies par la matrice principale avant la résolution ce qui suppose que ces matrices soient effectivement instanciées au moment de l'exécution. Ainsi on gagne en vitesse, en robustesse mais cela se fait au détriment de la place mémoire car il faut également stocker les matrices secondaires.

   Un exemple d'utilisation :

   En matrice principal l'utilisation de BANDE_SYMETRIQUE_LAPACK qui implique une résolution de type cholesky ce qui suppose une matrice définie positive. Si la positivité cesse, la résolution échoue. Dans ce cas en utilisant une matrice secondaire de type par défaut, on peut en général continuer le calcul. Un exemple de déclaration :

```
para_syteme_lineaire
#-----------------------------------------------
```

```
# PARAMETRE | VALEUR |
#-------------------------------------------
TYPE_MATRICE BANDE_SYMETRIQUE_LAPACK
MATRICE_S_SECONDAIRE_S_   BANDE_SYMETRIQUE FIN_TYPE_MAT_SECONDAIRES_
TYPE_RESOLUTION_S_SECONDAIRE_S_  CHOLESKY FIN_TYPE_RESOLUTION_SECONDAIRES_
```

9. TYPE_RESOLUTION_S_SECONDAIRE_S_ < une liste de type de matrice puis le mot clé > FIN_TYPE_RESOLUTION_SECONDAIRES_ :
   Permet de définir le type de résolution des matrices secondaires (cf. 8). Par défaut c'est le type de la matrice principale qui est utilisé si cela a un sens.
10. OPTIMISATION_POINTEURS_ASSEMBLAGE < un entier > : si =1 : permet de demander une renumérotation globale des pointeurs d'assemblages des différents degrés de liberté au niveau du stockage matriciel. Chaque degré de liberté possède un numéro global unique qui correspond à sa position dans le vecteur résidu global et la matrice de raideur globale (quand elle est utilisée). Par défaut ce numéro est construit à partir du numéro de noeud (pour un ddl dépendant d'un noeud) et du numéro du maillage (= la position du maillage dans le fichier de mise en données : i.e. .info). Cette numérotation n'est pas toujours optimale, en particulier pour le cas où on utilise plusieurs maillages, et conduit à des matrices pouvant être très creuses (i.e. incluant beaucoup de zéros). Pour optimiser le stockage matriciel, il est possible de demander une renumérotation globale de tous les pointeurs d'assemblage.
    Remarque :
    (a) La renumérotation globale des pointeurs d'assemblages n'est possible que si les maillages possèdent des relations entre eux : par exemple via des conditions linéaires. Dans le cas où les maillages sont indépendants, la renumérotation globale n'apportera pas d'optimisation par rapport à une renumérotation de noeuds (cf. 31)
    (b) Dans le cas du contact, entre plusieurs maillages on crée de ce fait des relations entre les maillages avec la particularité que ces relations peuvent aparaîtres ou disparaîtres pendant le calcul. Il est alors possible de demander une renumérotation pendant le calcul via le mot clé OPTIMISATION_NUMEROTATION indiqué au niveau des paramètres de contact voir 77.

Remarque :

1. 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 quelque 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.

   Cependant, pendant le calcul, les différents algorithmes nécessaires à l'ensemble du calcul, peuvent nécessiter de modifier la matrice masse en taille. Par exemple la

mise en place de condition de contact, ou de conditions linéaires, peuvent demander d'augmenter la zone de stockage. Dans ce cas, même si le stockage initial était de type diagonal, il bascule automatiquement vers le type de celui de la matrice raideur. On se reportera à 73 pour plus de précision.

2. Pour la raideur la taille initiale peut également varier au cours du calcul, par exemple dans le cas de la présence de conditions linéaires (ou non) cinématiques, et ou de contact en implicite.
   - si la condition est prise en compte sous forme d'une modification de repères locaux, et que la condition ne concerne que les ddl d'un seul noeud → cela n'entraîne pas de modification.
   - si la condition est prise en compte sous forme d'une modification de repères locaux, et que la condition concerne les ddl de plusieurs noeuds → le nombre de colonnes (quand cela a un sens) ou la largeur de bande est modifié en fonction de la numérotation des noeuds de la condition linéaire et de leur nombre. En général cela conduit très rapidement à une largeur de bande très importante !

   À noter qu'en explicite, le contact pris en compte par pénalisation ne modifie pas la largeur de bande. Il en est de même pour le contact solide-déformable en implicite.

   Par contre, en implicite, le contact déformable-déformable pris en compte par pénalisation augmente la largeur de bande ! ! et cela de manière aussi impactante que le cas de conditions linéaires entre noeuds différent. Dans ce cas il est sans doute plus judicieux d'utiliser un contact par multiplicateur de Lagrange ce qui va également agrandir la largeur de bande, mais seulement en fonction du maximum de différences de numéros entre les noeuds en contact, contrairement au cas par pénalisation qui va augmenter la largeur en fonction de la "somme totale" des différences entre de numéros des éléments en contact. Cependant les multiplicateurs entraînent d'autres difficultés : des ddl en plus, un conditionnement moins bon, le caractère défini positif n'est plus garanti, etc. donc ce n'est pas non plus la panacée.
3. voir également : 68.3 et 73.

76 Pilotage de la résolution globale

Ce jeu 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 donnée dans la table (300).

TABLE 300 – liste des sous-mots clés associés aux paramètres de contrôle du pilotage

sous 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)
NORME_MAXI_X_INCREMENT	nombre très grand	(11)
NORME_MAXI_V_INCREMENT	nombre très grand	(12)
MAXI_VARIATION_DDL_POUR_CONVERGENCE	nombre très grand	(13)
MINI_VARIATION_DDL_POUR_CONVERGENCE	0.	(14)
NB_CYCLE_CONTROLE_RESIDU	3	(15)
PLAGE_CONTROLE_RESIDU	3	(16)
INIT_INCRE_AVEC_PAS_PREC	1	(17)
CAS_JACOBIEN_NEGATIF	1	(18)
VAR_MAXI_JACOBIEN	0.	(19)
CAS_PB_FCTND_CHARGE	0	(20)

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 . 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 résidu(n+α) > résidu(n), n étant le nombre d'itérations, 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 diminution.
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'itérations 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'itérations inférieur ou égal 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'itérations 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 $\boldsymbol{\sigma} = \boldsymbol{T} \ \boldsymbol{\varepsilon}$ 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 déterminée 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 supérieure au facteur, l'ensemble du vecteur ddl est modifié selon la formule : $(ddl) - > facteur * (ddl)/||(ddl)||$. 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. NORME_MAXI_X_INCREMENT <un nombre réel > : Dans le cas d'un calcul dynamique implicite avec la méthode de Newmark, et pour l'algorithme de relaxation

dynamique, il est possible de limiter le maximum de l'incrément de déplacement. Lorsque le nombre indiqué est positif, c'est l'ensemble du vecteur déplacement qui est modifié sous forme d'un facteur d'homotétie de telle manière que la norme du vecteur incrément final = la valeur imposée. Lorsque le nombre indiqué est négatif, le fonctionnement est différent : seules les composantes qui ont une intensité (valeur absolue) supérieur à la valeur imposée, sont réduites à la valeur imposée. Il s'agit alors d'un écrêtement des composantes maxi.

12. NORME_MAXI_V_INCREMENT <un nombre réel > : Le fonctionnement est identique au cas " NORME_MAXI_X_INCREMENT " mais ici pour le vecteur incrément de vitesse.
13. MAXI_VARIATION_DDL_POUR_CONVERGENCE <un nombre réel > : défini 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.
14. MINI_VARIATION_DDL_POUR_CONVERGENCE <un nombre réel > : défini 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.
15. NB_CYCLE_CONTROLE_RESIDU < un entier >, et PLAGE_CONTROLE_RESIDU < un entier > : définissent les paramètres de réglage du contrôle de la variation de résidu. Ce paramètre est à utiliser avec le type de pilotage PILOT_GRADIENT : voir (voir 1) pour l'utilisation de ces paramètres.
16. PLAGE_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 .
17. 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 précédant. Cette initialisation ne fonctionne que pour les algorithmes globaux statiques (pour l'instant, Newmark en est exclu).
18. 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.
19. 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 initial 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 imposée, on considère qu'il y a divergence. La suite dépend du type de pilotage retenu.

20. CAS_PB_FCTND_CHARGE < un entier > : permet d'indiquer ce que l'on fait lorsque l'on rencontre le cas d'un chargement utilisant une fonction nD, et que celle-ci rencontre un pb : typiquement une valeur de retour infinie.
    - = 0 : par défaut le chargement est neutralisé, le calcul des efforts extérieurs est stoppé et la suite dépend du type de pilotage
    - = 1 : le chargement est neutralisé, le calcul des efforts extérieurs continus sans la prise en compte du chargement neutralisé (il y a un message d'erreur si le niveau de commentaire n'est pas nul)

77 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 donnée dans la table (301).

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. 79, en particulier les paramètres : POINT_INTERNE_DELTA_THETAI_MAXI et POINT_INTERNE_PREC_THETAI_INTERNE .
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).
    - = 0 : le contact n'est pas actif,
    - = 1 : il s'agit d'une méthode sans multiplicateur de Lagrange ni pénalisation, utilisable en implicite avec l'algorithme particulier : " non_dyna_contact " cf. (III),

TABLE 301 – liste des sous-mots clés associés aux paramètres de contrôle liés au contact - frottement

sous 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_PENALISATION_TANGENTIELLE	2	(6)
TYPE_DE_DECOLLEMENT	0	(7)
NB_DECOLLEMENT_MAXI	1	(8)
PRECISION_BOITE_PRELOCALISATION	1.05	(9)
MINI_EXTRA_BOITE_PRELOCALISATION	0.001	(10)
AJOUT_EXTRA_BOITE_PRELOCALISATION	0.	(11)
PENETRATION_CONTACT_MAXI	0.1	(12)
TANGENTIELLE_CONTACT_MAXI	0.1	(13)
PENETRATION_BORNE_REGULARISATION	un nombre très grand	(15)
TANGENTIELLE_BORNE_REGULARISATION	un nombre très grand	(6)
FORCE_CONTACT_NOEUD_MAXI	un nombre très grand	(16)
FORCE_TANGENTIELLE_NOEUD_MAXI	un nombre très grand	(17)
DISTANCE_MAXI_AU_PT_PROJETE	un nombre très grand	(18)
FACT_POUR_RAYON_ACCOSTAGE	1	(19)
NIVEAU_COMMENTAIRE_CONTACT	0	(20)
OPTIMISATION_NUMEROTATION	0	(21)
FCT_ND_BASCUL_CONTACT_TYPE_4	"_"	(22)
PRECISION_PT_SUR_FRONTIERE	1.e-4	(23)
NB_BOUCLE_NEWTON_SUR_POSITION_FRONTIERE	10	(24)

— = 2 : il s'agit d'une méthode avec pénalisation utilisable avec la plupart des algorithmes (sauf indication contraire dans la description de l'algorithme),

— = 4 : Il s'agit d'une méthode avec pénalisation, dont le facteur est dimensionné de manière automatique en fonction de la rigidité constaté des matériaux en contact. Le facteur de pénalisation est tout d'abord dimensionné au cours d'une première phase d'équilibre global, puis sa valeur est utilisé dans la seconde et dernière phase de calcul. La transition entre ces deux phases est pilotée à l'aide du paramètre FCT_ND_BASCUL_CONTACT_TYPE_4 (cf.22). Dans la seconde phase, tout ce passe comme si le contact était de type 2, à part le fait que le facteur de pénalisation garde la valeur identifiée dans la première phase.

Dans le cas d'un calcul avec un algorithme global implicite et lorsqu'aucune fonction de basculement n'est définie (cf.22), le basculement s'effectue automatiquement pour chaque incrément, à partir de la 2ième itération.

Dans le cas d'un calcul avec un algorithme global explicite, la définition d'une fonction particulière de basculement est impérative (voir 81.2 pour plus de

détails). Par exemple on peut décider d'une mise à jour périodique du facteur de pénalisation en fonction du nombre d'incrément.

Durant la première phase, les noeuds esclaves en contact sont systématiquement ramenés sur la surface maître selon un déplacement normal à la surface $\Delta\vec{M} = \Delta M.\vec{N}$, avant le calcul des puissances internes (= efforts généralisés correspondant à la déformation des structures). On satisfait ainsi de manière exacte la condition cinématique de contact. Après calcul des efforts internes, ceux relatifs aux noeuds en contact correspondent en fait aux forces $\vec{F}_{contact}$ exercées par le contact pour ramener le noeud esclave "M" sur la surface suivant $\Delta M.\vec{N}$. On en déduit le facteur de pénalisation α tel que :

$$\vec{F}_{contact} = \alpha|\Delta M|_{maxi}\vec{N} \text{ c'est-à-dire } \alpha = \frac{||\vec{F}_{contact}||}{|\Delta M|_{maxi}} \tag{127}$$

Avec $|\Delta M|_{maxi} =$ PENETRATION_CONTACT_MAXI .
Ainsi le paramètre PENETRATION_CONTACT_MAXI définit la pénétration maximale que l'on souhaite.

Remarque Il faut noter qu'une valeur trop faible du paramètre PENETRATION_CONTACT_MAXI peut conduire à un facteur de pénalisation pouvant être très grand par rapport aux termes de la matrice de raideur correspondante aux efforts internes. Dans ce cas, le conditionnement de la matrice de raideur globale finale se détériore pouvant être responsable d'une divergence au niveau de la recherche de l'équilibre globale.

Enfin, dans le cas par exemple d'un calcul dynamique avec impact, les forces de contact peuvent fortement osciller. Pour minimiser les conséquences de ces oscillations sur le calcul de la pénalisation, on adopte une moyenne glissante. Par défaut (version 6.896) la moyenne glissante s'effectue sur 10 valeurs successives du facteur de pénalisation. Il est possible de modifier ce nombre à l'aide de la fonction nD définie par le paramètre FCT_ND_BASCUL_CONTACT_TYPE_4 . Pour ce faire il faut définir une fonction vectorielle qui renvoit deux composantes, dont la seconde = le nombre d'éléments que l'on désire dans la moyenne glissante. La mise en données (partielle) suivante donne un exemple de définition pour lequel on bascule après 100 itérations et on utilise une moyenne glissante sur 4 éléments.

```
 les_fonctions_nD #------------
 f_ct4  FONCTION_EXPRESSION_LITTERALE_nD
    un_argument= compteur_iteration_algo_global
      fct= (compteur_iteration_algo_global > 100 )? 1 : 0 , 4
    fin_parametres_fonction_expression_litterale_
    .
    .
para_contact ------------
#----------------------------------------
# PARAMETRE | VALEUR |
#----------------------------------------
```

```
CONTACT_TYPE  4
TYPE_PENALISATION_PENETRATION     8
PENETRATION_CONTACT_MAXI                1.e-2
FCT_ND_BASCUL_CONTACT_TYPE_4    f_ct4
```

3. PENALISATION_PENETRATION <un réel $\alpha >$ indique un des éléments permettant le calcul du facteur de pénalisation, sauf pour le contact type 4 pour lequel le facteur de pénalisation est calculée automatiquement (voir CONTACT_TYPE = 4).

   Le facteur $\alpha >$ est employé pour définir la force de réaction due à une pénétration $\Delta\vec{X}$ telle 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 .

   Il est possible de piloter la valeur de α via une fonction nD. On se reportera à (77.1) pour la disponibilité des paramètres de passage. La fonction doit ramener un scalaire, qui est donc utilisé comme paramètre α. La syntaxe de mise en donnée est la suivante :
   PENALISATION_PENETRATION FCT_ND_PENALISATION_PENETRATION <nom_d'une_fonction_nD>
   où :

   — FCT_ND_PENALISATION_PENETRATION : est le mot clé qui indique que l'on veut utiliser une fonction nD

   — <nom_d'une_fonction_nD> : est le nom d'une fonction nD qui doit avoir été préalablement déclarée dans la mise en données.

4. TYPE_PENALISATION_PENETRATION <un entier "indic" > qui indique la méthode utilisée pour calculer le facteur β.

   Indique également la méthode prise en compte pour la suppression du contact ou de la force de contact.

   Pour la suite on appelle $e = P\hat{\vec{M}}.\vec{N}$ le gap de pénétration avec : "P" le point de contact sur la surface maître, $\hat{M}$ la position du noeud esclave au temps "$t + \Delta t$", $\vec{N}$ le vecteur normal à la surface maître au point "P" dont le sens va vers l'extérieur . Le point "P" est déterminé initialement par l'intersection de la trajectoire du noeud esclave avec la surface maître. Ensuite, lorsque le contact est actif, la trajectoire du noeud esclave est approximativement collinéaire avec la surface maître aussi le point "P" est égal à la projection du noeud esclave sur la facette maître.

   En fonction de la valeur de "indic" on a :

   = 0 : dans ce cas très particulier, le facteur de pénalisation = 0. Ce cas peut-être utile lorsque l'on veut activer le contact tout en ne générant aucune force de contact, ceci pour une certaine phase de calcul.

   =1 : dans ce cas $\beta = 1$, ce qui veut dire que α est directement le facteur de pénalisation. Lorsque "$e > 0$.", la force de contact est mise à zéro.

   =2 : Comme pour le cas 1, lorsque "$e > 0$.", la force de contact est mise à zéro. Si "$e < 0$.", deux cas sont à distinguer : soit le noeud esclave impact un élément solide, soit il impacts 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 :

$$\beta = \frac{K_e \ A_e}{max(distance\ entre\ les\ noeuds)}$$

=3 : même fonctionnement que le cas 2, mais avec l'algorithme supplémentaire suivant. On sauvegarde d'un pas sur l'autre la pénétration. À 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 certains 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 : $-e_{regul} \leqslant e \leqslant e_{regul}$ alors : $\beta = \beta_{base} \times 0.25 \times (e - e_{regul})^2/(e_{regul})^2$, fonction qui varie de 1. (pour e = $-e_{regul}$) à 0. (pour e=e_{regul})
- si $e > e_{regul}$ dans ce cas $\beta = 0$, la force de réaction sera donc nulle.
- si $e < -e_{regul}$ dans ce cas $\beta = \beta_{base}$

La grandeur e_{regul} est donnée par le paramètre
PENETRATION_BORNE_REGULARISATION (cf.15).

=4 : même algorithme que le cas 5 avec la différence suivante : lorsque le pas de calcul est validé, en supposant une raideur de contact proportionnelle avec la pénétration, et dans le cas où la pénétration aurait dépassé un seuil maxi "e_{maxi}" (c'est-à-dire $e(t) < -e_{maxi}$), on calcul un facteur supplémentaire sur β pour tenter de revenir au seuil maxi. On aurait donc :

$$\beta = \beta_{base} \times \frac{e(t)}{e_{maxi}}$$

Bien noter que la grandeur $e(t)$ est la pénétration du pas précédent qui a été validé. Donc cette grandeur peut-être différente de la pénétration actuelle $e(t + \Delta t)$. Cependant, on utilise la grandeur $e(t)$ pour éviter des oscillations, si on retient les valeurs actuelles cela cause des instabilités de convergence dramatiques.

Dans la pratique, on applique une moyenne glissante sur 2 calculs successifs. C'est-à-dire d'un pas à l'autre (d'une itération à l'autre en implicite par

exemple, ou sur deux pas de temps en explicite) on utilise la moyenne des deux valeurs successives que prend le facteur : $\frac{e(t)}{e_{maxi}}$. L'idée est ainsi encore de réduire les oscillations de la grandeur $\frac{e(t)}{e_{maxi}}$.
La grandeur e_{maxi} est donnée par le paramètre PENETRATION_CONTACT_MAXI (cf.12).

=6 : même algorithme que le cas 5 mais avec une fonction différente :

— si la pénétration "e" est telle que : $-e_{regul} \leqslant e \leqslant e_{regul}$ alors :

$$\beta = -\beta_{base} \times \frac{\tanh(4.e/e_{regul}) - 1.}{2}$$

, fonction qui varie de ≈ 0.99966 (pour e $= -e_{regul}$) à $\approx 3.3535e-04$ (pour e$=e_{regul}$)

— si $e > e_{regul}$ dans ce cas $\beta = 0$, la force de réaction sera donc nulle.

— si $e < -e_{regul}$ dans ce cas $\beta = \beta_{base}$

=7 : même algorithme que le cas 2 mais avec le fait que la force de contact n'est jamais neutralisée. Néanmoins elle peut devenir nulle si $e > e_{regul}$, donc dans la pratique ce type n'est pas utile !

=8 : même algorithme que le cas 4, avec le fait que la force de contact n'est jamais mise à 0 : quelque soit le sens de la pénétration, celle-ci est considérée négative.

Il s'agit donc d'un cas particulier de contact toujours collant, une fois le contact détecté.

Remarques, cas particulier où le module de compressibilité volumique n'existe pas : dans le cas où la facette appartient à un élément qui n'a pas de compressibilité volumique définit, par exemple si aucune loi de comportement ne lui est associée, il n'est pas possible de calculer un facteur β. Dans ce cas, le facteur β est mis arbitrairement à 1.

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 actuellement que dans 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. Cependant si la pénalisation tangentielle est nulle, le déplacement tangentiel du noeud est totalement libre. En fait la notion de "non glissement" est relative au fait que l'on n'est pas dans un régime de glissement avec une loi de frottement. Le contact collant correspond à la première partie d'une loi de Coulomb dans laquelle on se trouve dans le "cône" de la loi.

Remarque Cette partie sera complétée lors de la mise en oeuvre des lois de frottement.

Il est possible de piloter la valeur de α_t via une fonction nD. On se reportera à (77.1) pour la disponibilité des paramètres de passage. La fonction doit ramener un scalaire, qui est donc utilisé comme paramètre α_t. La syntaxe de mise en donnée est

la suivante :
PENALISATION_TANGENTIELLE FCT_ND_PENALISATION_TANGENTIELLE <nom_d'une_fonction_nD>
où :

- FCT_ND_PENALISATION_TANGENTIELLE : est le mot clé qui indique que l'on veut utiliser une fonction nD
- <nom_d'une_fonction_nD> : est le nom d'une fonction nD qui doit avoir été préalablement déclarée dans la mise en données.

6. TYPE_PENALISATION_TANGENTIELLE <un entier "m" > qui indique la méthode utilisée. Le fonctionnement suit le même principe que dans le cas TYPE_PENALISATION_PENETRATION ceci à l'aide des paramètres propres TANGENTIELLE_CONTACT_MAXI et TANGENTIELLE_BORNE_REGULARISATION , les autres paramètres étant identiques au cas de la pénétration.
7. 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. 15)

   NB Dans le cas où TYPE_PENALISATION_PENETRATION = 7, le contact une fois détecté, n'est jamais supprimé.
8. 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. **NB** Dans le cas où TYPE_PENALISATION_PENETRATION = 7, le contact une fois détecté, n'est jamais supprimé.
9. 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 $\alpha' >= 1$.. Sans raison particulière, il est préférable de laisser le paramètre par défaut.
   - puis la boite est agrandie dans toutes les directions d'une même valeur : $\beta' \times MAX_{i=1}^{dim} \mid X_{max}^i - X_{min}^i \mid$. Là aussi il est préférable de laisser la paramètre β' (cf.10) par défaut.
   - enfin la boite est de nouveau agrandie dans toutes les directions de la valeur éventuellement indiquée par l'utilisateur : γ' (cf.11) par défaut = 0.

Ces paramètres (α' β' γ') sont également utilisés pour définir la boite d'encombrement de l'élément géométrique attaché à la facette. Cette boite sert pour la détection des noeuds internes (à la matière) au début du calcul, et également pendant certaines phases de la détection de contact.

10. MINI_EXTRA_BOITE_PRELOCALISATION <un réel > = β'. (cf.9)
11. AJOUT_EXTRA_BOITE_PRELOCALISATION <un réel > = γ'. (cf.9)
12. PENETRATION_CONTACT_MAXI <un réel e_{max} > : qui indique une borne maxi de pénétration. Si la pénétration excède cette valeur et que
" TYPE_PENALISATION_PENETRATION = 4", le facteur de pénalisation est modifié pour essayer de limiter la pénétration à la valeur indiquée e_{max}.
Il est possible de piloter la valeur de e_{max} via une fonction nD. On se reportera à (77.1) pour la disponibilité des paramètres de passage. La fonction doit ramener un scalaire, qui est donc utilisé comme paramètre e_{max}. La syntaxe de mise en donnée est la suivante :
PENETRATION_CONTACT_MAXI FCT_ND_PENETRATION_CONTACT_MAXI <nom_d'une_fonction_nD>
où :
    - FCT_ND_PENETRATION_CONTACT_MAXI : est le mot clé qui indique que l'on veut utiliser une fonction nD
    - <nom_d'une_fonction_nD> : est le nom d'une fonction nD qui doit avoir été préalablement déclarée dans la mise en données.
13. TANGENTIELLE_CONTACT_MAXI <un réel tc_{max} > : qui indique une borne maxi de déplacement sur la surface frontière. Dans le principe, l'utilisation est identique au cas de PENETRATION_CONTACT_MAXI , mais ici relativement au déplacement sur la frontière.
Il est possible de piloter la valeur de tc_{max} via une fonction nD. On se reportera à (77.1) pour la disponibilité des paramètres de passage. La fonction doit ramener un scalaire, qui est donc utilisé comme paramètre tc_{max}. La syntaxe de mise en donnée est la suivante :
TANGENTIELLE_CONTACT_MAXI FCT_ND_TANGENTIELLE_CONTACT_MAXI <nom_d'une_fonction_nD>
où :
    - FCT_ND_TANGENTIELLE_CONTACT_MAXI : est le mot clé qui indique que l'on veut utiliser une fonction nD
    - <nom_d'une_fonction_nD> : est le nom d'une fonction nD qui doit avoir été préalablement déclarée dans la mise en données.
14. PENETRATION_BORNE_REGULARISATION <un réel e_{regul} > : qui indique une borne de régularisation sur la pénétration (dont l'utilisation dépend de l'algorithme qui gère la pénalisation (cf. 4 et 7).
Il est possible de piloter la valeur de e_{regul} via une fonction nD. On se reportera à (77.1) pour la disponibilité des paramètres de passage. La fonction doit ramener un scalaire, qui est donc utilisé comme paramètre e_{regul}. La syntaxe de mise en donnée est la suivante :
PENETRATION_BORNE_REGULARISATION FCT_ND_PENETRATION_BORNE_REGULARISATION <nom_d'une_fonction_nD>
où :

— FCT_ND_PENETRATION_BORNE_REGULARISATION : est le mot clé qui indique que l'on veut utiliser une fonction nD

— <nom_d'une_fonction_nD> : est le nom d'une fonction nD qui doit avoir été préalablement déclarée dans la mise en données.

15. TANGENTIELLE_BORNE_REGULARISATION <un réel et_{regul} > : qui indique une borne de régularisation sur le déplacement tangentiel (dont l'utilisation dépend de l'algorithme qui gère la pénalisation) .

    Il est possible de piloter la valeur de et_{regul} via une fonction nD. On se reportera à (77.1) pour la disponibilité des paramètres de passage. La fonction doit ramener un scalaire, qui est donc utilisé comme paramètre et_{regul}. La syntaxe de mise en donnée est la suivante :

    TANGENTIELLE_BORNE_REGULARISATION FCT_ND_TANGENTIELLE_BORNE_REGULARISATION <nom_d'une_fonction_nD>

    où :

    — FCT_ND_TANGENTIELLE_BORNE_REGULARISATION : est le mot clé qui indique que l'on veut utiliser une fonction nD

    — <nom_d'une_fonction_nD> : est le nom d'une fonction nD qui doit avoir été préalablement déclarée dans la mise en données.

16. FORCE_CONTACT_NOEUD_MAXI <un réel f_{max} > : qui indique une force maxi pour la force de réaction normale (de pénétration) par pénalisation. Si la force calculée est supérieure, elle est réduite arbitrairement à la valeur f_{max}.

    Il est possible de piloter la valeur de f_{max} via une fonction nD. On se reportera à (77.1) pour la disponibilité des paramètres de passage. La fonction doit ramener un scalaire, qui est donc utilisé comme paramètre f_{max}. La syntaxe de mise en donnée est la suivante :

    FORCE_CONTACT_NOEUD_MAXI FCT_ND_FORCE_CONTACT_NOEUD_MAXI <nom_d'une_fonction_nD>

    où :

    — FCT_ND_FORCE_CONTACT_NOEUD_MAXI : est le mot clé qui indique que l'on veut utiliser une fonction nD

    — <nom_d'une_fonction_nD> : est le nom d'une fonction nD qui doit avoir été préalablement déclarée dans la mise en données.

17. FORCE_TANGENTIELLE_NOEUD_MAXI <un réel ta_{max} > : qui indique une force maxi pour la force de réaction tangentielle par pénalisation. Si la force calculée est supérieure, elle est réduite arbitrairement à la valeur ta_{max}.

    Il est possible de piloter la valeur de ta_{max} via une fonction nD. On se reportera à (77.1) pour la disponibilité des paramètres de passage. La fonction doit ramener un scalaire, qui est donc utilisé comme paramètre ta_{max}. La syntaxe de mise en donnée est la suivante :

    FORCE_TANGENTIELLE_NOEUD_MAXI FCT_ND_FORCE_TANGENTIELLE_NOEUD_MAXI <nom_d'une_fonction_nD>

    où :

    — FCT_ND_FORCE_TANGENTIELLE_NOEUD_MAXI : est le mot clé qui indique que l'on veut utiliser une fonction nD

    — <nom_d'une_fonction_nD> : est le nom d'une fonction nD qui doit avoir été préalablement déclarée dans la mise en données.

18. 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 $\min(d_{max}, 2.\alpha\ ||\Delta X||_\infty)$ où $\alpha\ ||\Delta X||_\infty$ est détaillé en 19.
19. 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 inférieure à $\alpha\ ||\Delta X||_\infty$ pour être licite, ΔX étant la variation des coordonnées. $\alpha >$ est utilisé conjointement avec d_{max} (cf.18). La distance maxi admissible entre le noeud et sa projection étant : $\min(d_{max}, 2.\alpha\ ||\Delta X||_\infty)$.
20. NIVEAU_COMMENTAIRE_CONTACT <un entier "i" > : permet de spécifier un niveau de commentaires différents du niveau général. Utile si l'on veut plus de commentaire spécifiquement sur le contact.
21. OPTIMISATION_NUMEROTATION <un entier "i" > s'il vaut 1, cela signifie, dans le cas d'un calcul statique ou dynamique implicite avec matrice de raideur, que l'on veut une optimisation du stockage de la raideur tout au long du calcul, qui tienne compte de la présence des éléments de contact. La méthodologie mise en place est la suivante à chaque fois qu'il y a une modification des lieux de contact c'est-à-dire des éléments de contact (nouvel élément, et/ou suppression d'élément et/ou changement de frontière)
    (a) renumérotation globable de l'ensemble des maillages en tenant compte des connexions introduites par les éléments de contact de manière à obtenir une largeur de bande théorique minimale,
    (b) modification des pointeurs d'assemblages de tous les degrés de liberté actifs en tenant compte de la nouvelle répartition des numérotations dans les maillages,
    (c) modification et ajustement de la taille des matrices pour tenant compte des nouveaux pointeurs d'assemblage.

    Remarque À noter que la numérotation des noeuds dans les maillages est indépendantes de celle des pointeurs d'assemblages des degrés de liberté des noeuds, ceci pour que l'optimisation ne soit pas liée au découpage par maillage.
22. FCT_ND_BASCUL_CONTACT_TYPE_4 < une chaine de caractères > : indique le nom d'une fonction nD, qui permet pour un contact de type 4, de piloter le basculement entre la première phase de calcul, dédié au dimensionnement du facteur de pénalisation, et la seconde phase dédié à l'utilisation du facteur de pénalisation (cf.2). On se reportera à 81.2 pour plus de détail sur la définition des fonctions nD. La fonction doit exister et dépendre uniquement de grandeurs globales (cf.88.1). Par exemple est peut dépendre du numéro d'incrément en cours et/ou du numéro d'itération (pour un calcul implicite).
23. PRECISION_PT_SUR_FRONTIERE < un réel > : qui donne la précision de positionnement des points de contact recherchés sur les frontières.
24. NB_BOUCLE_NEWTON_SUR_POSITION_FRONTIERE < un entier> : qui donne le nombre maxi de boucle de Newton pour la recherche du point de projection de contact sur une surface : a priori courbe donc correspond à une recherche non linéaire.

77.1 Pilotage d'un paramètre via une fonction nD :

La fonction nD peut utiliser toutes les grandeurs globales disponibles. De plus elle peut utiliser les grandeurs suivantes :

- tous les `Ddl_enum_etendu` (cf. 88.3) qui sont définies au niveau du noeud en contact. Cette liste est spécifique à chaque noeud et dépend du type de problème traité. En particulier on dispose pour le noeud en contact de :
  - `X1` : coordonnée x actuel,
  - `X2` : coordonnée y actuel,
  - `X3` : coordonnée z actuel,
  - `reaction_normale` : l'intensité de la force de réaction normale,
  - `reaction_tangentielle` : l'intensité de la force de réaction tangentielle,
  - `X1_t` : coordonnée x à t,
  - `X2_t` : coordonnée y à t,
  - `X3_t` : coordonnée z à t,
  - `X1_t0` : coordonnée x initiale,
  - `X2_t0` : coordonnée y initiale,
  - `X3_t0` : coordonnée z initiale,
- également un ensemble de grandeurs qualifiées de "quelconques" dans Herezh, qui sont définies au niveau de l'élément de contact ou du noeud en contact. Elles correspondent en général, aux grandeurs disponibles en post-traitement. En particulier on notera la disponibilité des grandeurs suivantes :
  - `CONTACT_NB_DECOL` : correspond au nombre de décollements (donc de réaction de collages) qui ont été enregistrées pour le noeud en contact,
  - `CONTACT_PENALISATION_N` : la valeur du facteur de pénalisation complet pour la pénétration,
  - `CONTACT_PENALISATION_T` : la valeur du facteur de pénalisation complet pour le déplacement tangentiel,
  - `CONTACT_ENERG_GLISSE_ELAS` : l'énergie élastique (réversible) liée au déplacement tangentiel,
  - `CONTACT_ENERG_GLISSE_PLAS` : l'énergie plastique (non-réversible et non-visqueuse) liée au déplacement tangentiel,
  - `CONTACT_ENERG_GLISSE_VISQ` : l'énergie visqueuse (non-réversible d'origine visqueuse) liée au déplacement tangentiel,
  - `CONTACT_ENERG_PENAL` : l'énergie réversible liée au déplacement de pénétration,
  - `NUM_NOEUD` : le numéro du noeud en contact, qui correspond au numéro initiale dans le maillage,
  - `NUM_MAIL_NOEUD` : le numéro du maillage qui contient le numéro du noeud en contact. Les maillages sont numérotées dans l'ordre de lecture de la mise en données,

— NOEUD_PROJECTILE_EN_CONTACT : indique si le noeud en contact est actuellement actif (=100) ou non (= -0.1)

— CONTACT_CAS_SOLIDE : indique :

  — =0 contact bi-déformable

  — =1 le noeud est libre et la frontière est bloquée (solide)

  — = 2 le noeud est bloqué (solide) la frontière est libre

  — = 3 tout est solide

— CONTACT_COLLANT : indique si le noeud esclave est collant (= 1) ou non (=0)

— NUM_ZONE_CONTACT : indique éventuellement (si différent de 0) le numéro de la zone de contact du noeud esclave. Le numéro correspond au numéro d'ordre de déclaration des zones de contact, dans le fichier de mise en données. Si aucune zone n'est déclarée il est nul.

78 Affichage des résultats

Ce jeu de paramètres est facultatif. L'objectif de ce groupe de paramètre est de piloter l'affichage en général à l'écran. Dans certains 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'incréments, 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'itérations affiché.

Deux paramètres sont donc introduits 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 Mac OS. 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 spécifique 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 différente de la sortie des données au fil de calcul. Par exemple on peut vouloir un affichage continu au fil du calcul (tous les incréments par exemple), alors que la sauvegarde est plus espacée (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 donnée dans la table (302).

TABLE 302 – liste des sous-mots clés associés aux paramètres de contrôle du pilotage

sous 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. 296, le paramètre SAUVEGARDE) dans le cas où la fréquence de sauvegarde est supérieure à 0. Dans le cas où la fréquence de sauvegarde est inférieure 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 inférieure ou égale 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 (80.9) pour plus de précision à se sujet.
4. NB_CHIFFRE_POUR_DOUBLE_CALCUL <un nombre entier positif m > : donne le nombre de chiffres significatifs 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 chiffres significatif utilisé par le programme pour l'affichage des nombres en double précision utilisée pour les sorties graphiques.

79 Calculs géométriques

Ce jeu de paramètres est facultatif. L'objectif de ce groupe de paramètre est de piloter des calculs géométriques, qui sont relativement indépendants 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 (303) donne une liste exhaustive des différents paramètres et de leur signification.

```
            para_calculs_geometriques ------------
#---------------------------
# PARAMETRE    |  VALEUR   |
#---------------------------
POINT_INTERNE_DELTA_THETAI_MAXI    1.e-4
POINT_INTERNE_PREC_THETAI_INTERNE  1.e-6
POINT_INTERNE_NB_BOUCLE_SUR_DELTA_THETAI  10
POINT_INTERNE_NB_EXTERNE                  4
```

TABLE 303 – liste des sous-mots clés associés aux paramètres de contrôle du pilotage

sous mot clé	valeur par défaut	ref du commentaire
POINT_INTERNE_DELTA_THETAI_MAXI	1.e-4	(1)
POINT_INTERNE_PREC_THETAI_INTERNE	1.e-6	(2)
POINT_INTERNE_NB_BOUCLE_SUR_DELTA_THETAI	10	(3)
POINT_INTERNE_NB_EXTERNEL	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_THETAI_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 $|\theta^i_{maxi}|$ sont en général de l'ordre de 1 pour la plupart des éléments.

2. POINT_INTERNE_PREC_THETAI_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 $||\theta^i(M) - \theta^i(pointfrontiere)|| < e$ 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_THETAI_INTERNE éventuellement plus faible que celle POINT_INTERNE_DELTA_
(?)

3. POINT_INTERNE_NB_BOUCLE_SUR_DELTA_THETAI <un nombre entier positif m > : nombre de boucles 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 lequel 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 boucles 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 a 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'erreurs d'arrondis, lorsque le conditionnement des matrices des composantes des tenseurs est mal conditionné.

Treizième partie

Sortie des résultats

80 Sortie des résultats

80.1 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 multicolonnes 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.*

80.2 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 :

1. 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. Le fichier contient successivement :

- les réactions individuelles pour chaque ddl bloqué,
- les torseurs de réaction globaux, un par référence bloquée
- les réactions dues aux conditions linéaires
- (version > 6.728) les efforts de blocage sous forme d'un champ de vecteur, avec en commentaire en préambule, une référence sur l'ensemble des ddl bloqués.

2. 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. Dans le cas où on demande de plus une sortie au fil du calcul (cf.80.9) au format gmsh, toutes les grandeurs de sortie doivent être transférées aux noeuds. Dans ce cas on récupère dans le fichier .ddl toutes les dernières grandeurs calculées au dernier incrément et transférées aux noeuds. On peut ainsi retrouver des contraintes, des déformations, etc. , en faite toutes les grandeurs que l'on a demandé pour les sorties au fil du calcul.

3. suffixe .res : ce fichier contient un certain nombre de grandeurs classiques calculées aux points d'intégration. 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 "

4. suffixe " _cab.isoe " : ce fichier contient uniquement les contraintes calculées aux points d'intégration. En fait le suffixe est précédé d'un numéro, " i_cab.isoe " où "i" est un entier, indiquant qu'il s'agit des informations pour le maillage numéro "i". Le maillage numéro "i" correspond au ième maillage lu dans le ".info". On a donc en sortie, un fichier par maillage. Le format des informations est le suivant :
   . ligne 1 : le chiffre "0" suivi du nombre d'éléments
   . ligne 2 : la ou les chaînes de caractères indiquant les composantes :
      — en 1D : " Sig11 "
      — en 2D : " Sig11 Sig22 Sig12 "
      — en 3D : " Sig11 Sig22 Sig33 Sig12 Sig23 Sig13"
   . lignes suivantes :
   Pour chaque élément :
      * Pour chaque point d'intégration, les valeurs sur une ligne de σ^{ij} dans l'ordre indiqué dans l'entête

TABLE 304 – Exemple de fichier de sortie de contraintes : suffixe "_cab.isoe ", ici cas 1D

```
         0    5
          Sig11
-1           2068.91
-1           2068.91
-1           2068.91
-1           2068.91
-1           2068.91
```

5. suffixe " _dpl.points " : le fichier contient uniquement les déplacements des noeuds. En fait le suffixe est précédé d'un numéro, " i_dpl.points " où "i" est un entier, indiquant qu'il s'agit des informations pour le maillage numéro "i". Le maillage numéro "i" correspond au ième maillage lu dans le ".info". On a donc en sortie, un fichier par maillage. Le format des informations est le suivant :

. ligne 1 : le nombre de noeuds

. ligne 2 et suivantes Pour chaque noeud :

  * les valeurs sur une ligne des 3 composantes de déplacement, que ce soit en 1D ou 2D ou 3D, puis le chiffre 0

TABLE 305 – Exemple de fichier de sortie de contraintes : suffixe "_dpl.points", ici seule les x varient car il s'agit d'un cas 1D

```
6
                0                0                0                0
  0.2000000180774109                0                0                0
  0.4000000361548217                0                0                0
  0.600000054232197                 0                0                0
  0.8000000723096434                0                0                0
  0.9999999999999432                0                0                0
```

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.

80.2.1 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'intégration. 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).

80.3 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 sous forme d'un ou plusieurs sous-types de calcul au début du fichier .info. On se reportera à (26) pour les indications générales relatives aux sous-types. Pour le cas spécifique des sorties graphiques, 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 [306].

TABLE 306 – liste des différentes options de post-traitement graphique, à 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

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
```

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)
 changement de fichier de commande .CVisu          ?  (rep 7)
 visualisation au format Gmsh                      ?  (rep 8)
 nom grandeurs actuelles accessibles globalement   ?  (rep 9)
 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.

80.3.1 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 autocommenté, 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 defaut 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 il a été prévu dans le fichier .info une sauvegarde des paramètres avec le mot clé "sauveCommandesVisu" cf.(306), soit il y a utilisation de 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.(306). Ce mot clé peut également être associé au mot clé "sauveCommandesVisu" cf.(306), 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 defaut 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 defaut 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 defaut 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

#########################################################################################################
```

80.4 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 defaut ? : 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 deformees
 parametre par defaut ? : affichage en filaire, couleur cyan,
                           affichage des facettes, couleur jaune,
                          pas d'affichage de numéros,
                          amplification du deplacement = 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 defaut): 1
parametre par defaut ? 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 définit 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.296). 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 certain pas de temps. Dans ce dernier cas, au moment de la visualisation, seuls ces pas de temps sauvegardé 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 maillages disponibles :  de 1 a 1
parametre par defaut ? 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).

80.5 Format tableaux Maple

La sortie des grandeurs sous forme de tableau a été initialement développée pour être exploitée avec le logiciel "maple", aussi le fichier généré par Herezh s'écrit avec un suffixe ".maple".

80.5.1 introduction

En fait, il s'agit ici d'une sortie littérale (format texte) de toutes les grandeurs accessibles en post-traitement. Les grandeurs obtenues représentent celles utilisées ou calculées pendant l'exécution d'Herezh++. C'est la sortie qu'il faut utiliser pour être le plus proche des grandeurs du calcul. Par exemple, si l'on utilise une sortie au format "**gmsh**", une opération de transfert aux noeuds via une extrapolation, est effectuée sur les grandeurs post-traitées. Les grandeurs obtenues sont en général lissées, et peuvent ainsi être différentes de celles initialement calculées pendant la résolution.

Le fichier de sortie est formaté sous forme d'un tableau multi-colonnes auto-documenté, ce qui permet de l'utiliser sans l'aide du programme de calcul.

En particulier, cela permet de tracer aisément 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 (par exemple avec le logiciel **gnuplot**). 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 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. En particulier, la signification de chaque colonne est spécifiée dans le préambule commenté.

L'exemple classique 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 defaut : 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 défaut): 0 1
 parametre par defaut ? le dernier increment    (rep 'o') pour accepter sinon autre
```

auquel on choisira a priori la réponse par défaut.

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
moyenne, maxi, mini etc. sur ref N    -> rep : smN
moyenne, maxi, mini etc. sur ref E    -> rep : smE
moyenne, maxi, mini etc. ref face E   -> rep : smFE
moyenne, maxi, mini etc. ref arete E -> rep : smAE
ddl aux noeuds                        -> rep : noe ?
ddl etendu aux noeuds                 -> rep : net ?
grandeur particuliere aux noeuds      -> rep : nop ?
grandeurs generique aux elements      -> rep : ele
grandeurs particulieres aux elements -> rep : elp
grandeurs tensorielles aux elements   -> rep : elt
grandeurs aux faces d'elements        -> rep : elF
grandeurs aux aretes d'elements       -> rep : elA
style de sortie                       -> rep : sty
 pour accepter la valeur par defaut   -> rep : o
 pour arreter les questions           -> rep : fin (ou f)
reponse ?
```

80.5.2 Cas des grandeurs aux noeuds

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 (307)

TABLE 307 – 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 gererales: 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 accolees 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'idendificateur 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 corresponds 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_externe
#                                                  [4]energie_elastique [5]energie_hourglass
#---------------------------------  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, chaques 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 defaut) : (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
```

80.5.3 Cas des grandeurs de type degré de liberté aux éléments

Après avoir choisi la réponse "ele" 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 coordonnées 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 ?
```

80.5.4 Cas des grandeurs de type spécifique, aux éléments

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 : efref)
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 (80.5.3) ;

80.5.5 Cas des grandeurs tensorielles, aux éléments

Il s'agit de récupérer la forme tensorielle, donc l'ensemble des composantes, des grandeurs. Par exemple tous les composantes du tenseur des contraintes, des déformations ...

Le fonctionnement suit une logique semblable à celles présentées précédemment.

80.5.6 Cas de grandeurs quelconques définies aux faces et arâtes d'éléments

Les chargements effectués sur les faces et-ou arêtes d'élément sont calculés via une intégrale de surface ou curviligne, qui utilise des points d'intégration (pti), typiquement des points de Gauss. Ces points d'intégration sont différents des points de l'élément. La valeur de la charge utilisée est celle définie exactement aux pti de surface ou d'arête. Il peut-être intéressant de pouvoir récupérer ces valeurs, notamment lorsque la charge est variable via par exemple une fonction nD.

On dispose pour cela de deux menus particulier, un pour les grandeurs reliées aux pti de face (choix "elF" dans le menu principal) et un pour les grandeurs reliées au pti d'arête (choix "elA").

Le fonctionnement des menus qui sont ensuite présentés suit une logique très semblable à celle du choix de grandeurs aux pti d'éléments.

On notera qu'il est possible d'utiliser des références de pti de faces ou des références de pti d'arêtes. Ces références correspondent à une suite de triplets d'entiers : 1) une numéro d'élément, 2) un numéro de face ou d'arête, 3) un numéro de pti.

80.5.7 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 (307)).

80.5.8 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. 80.5.1) 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.68). 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 généré 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 blo-

cage) 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.

80.5.9 Cas des moyennes, maxi, mini etc. sur des références de noeuds

Cette sortie permet de récupérer des grandeurs globalisées sur un ensemble de noeuds. Pour cela on doit choisir :

- une ou plusieurs références de noeuds,
- une suite de grandeurs qui peuvent être en fonction de leur existence : soit :
  - un degré de liberté de base (ex : `X1`)
  - un degré de liberté étendu au noeud (ex : `SIG11`)
  - une grandeur particulière (ex : `FORCE_GENE_EXT`)

À partir de ces informations Herezh calcule et affiche dans le .maple pour chaque référence :

- la somme des valeurs obtenues à chaque noeud,
- la moyenne des valeurs,
- pour les degrés de liberté de base et de liberté étendues :
  - le maximum des valeurs relatives avec le numéro du noeud correspondant,
  - le minimum des valeurs relatives avec le numéro du noeud correspondant,
  - le maximum des valeurs absolues avec le numéro du noeud correspondant,
  - le minimum des valeurs absolues avec le numéro du noeud correspondant.

Cette sortie peut par exemple être utilisée pour déterminer le déplacement maxi suivant chaque axe.

80.5.10 Cas des moyennes, maxi, mini etc. sur des références d'éléments

Cette sortie permet de récupérer des grandeurs globalisées sur un ensemble d'éléments. Pour cela on doit choisir :

- une ou plusieurs références d'éléments avec un ou plusieurs points d'intégration,
- ou une ou plusieurs références de points d'intégration,
- une suite de grandeurs qui peuvent être en fonction de leur existence : soit :
  - une grandeur de base définie au point d'intégration (ex : `SIG11`)
  - une grandeur particulière aux éléments (ex : `VOLUME_PTI`)
  - une grandeur évoluée (ex : `CONTRAINTE_COURANTE`)

À partir de ces informations Herezh calcule et affiche dans le .maple pour chaque référence :

- la somme des valeurs obtenues à chaque élément,
- la moyenne des valeurs,
- pour les grandeurs de base :
  - le maximum des valeurs relatives avec le numéro de l'élément et le numéro du point d'intégration correspondant,
  - le minimum des valeurs relatives avec le numéro de l'élément et le numéro du point d'intégration correspondant,
  - le maximum des valeurs absolues avec le numéro de l'élément et le numéro du point d'intégration correspondant,
  - le minimum des valeurs absolues avec le numéro de l'élément et le numéro du point d'intégration correspondant.

Cette sortie peut être utilisée par exemple pour déterminer les contraintes maxi et mini avec leur position.

Un autre exemple d'utilisation est l'obtention du volume (surface, longueur) d'une sélection d'éléments.

80.5.11 Cas des moyennes, maxi, mini etc. sur des références de faces ou d'arêtes d'élément

Ces moyennes concernent des références de faces et-ou d'arêtes d'éléments ainsi que des références de points d'intégration de faces et-ou d'arêtes d'élément. Elles suivent une logique semblable à celle exposée pour les noeuds et les points d'intégration d'éléments.

80.5.12 Remarques concernant l'ordre de sortie des composantes des grandeurs tensorielles

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 : $A(1,1)$ $A(2,1)$ $A(2,2)$ $A(3,1)$ $A(3,2)$ $A(3,3)$.

80.5.13 Cas particulier de l'ordre de sortie des grandeurs particulières pour les lois de comportement complexes

Supposons par exemple que l'on souhaite obtenir les contraintes particulières pour chaque lois appartenant à une loi complexe (ex : loi additive, ou de mélange, ou en def ou contraintes planes etc.)

80.6 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.

80.7 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 defaut : 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.

80.8 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 présentent, 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 etou 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.

80.9 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'incréments 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.80.3.1). 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 lies à l'affichage des résultats (cf. 78) 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.

80.10 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 proches 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 noeuds, 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 complète) 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.

80.11 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.

80.11.1 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-Δt 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-Δt 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" $= trace(\boldsymbol{\epsilon})/3.$)

— l'intensité du déviateur de déformation : "Q_eps" $= \sqrt{\overline{\boldsymbol{\epsilon}} : \overline{\boldsymbol{\epsilon}}}$

— 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" $= \sqrt{2/3 \times \overline{\boldsymbol{\epsilon}} : \overline{\boldsymbol{\epsilon}}}$,

— la déformation cumulée appelée certaine fois déformation équivalente : "def_equivalente" $= \int_0^t \sqrt{2./3. \times \overline{\boldsymbol{D}} : \overline{\boldsymbol{D}}}\, dt$,

— 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" $= \sqrt{2./3. \times \overline{\boldsymbol{D}} : \overline{\boldsymbol{D}}}$

80.11.2 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" $= trace(\boldsymbol{\sigma})/3.$)
— l'intensité du déviateur des contraintes : "Q_sig" $= \sqrt{\overline{\boldsymbol{\sigma}} : \overline{\boldsymbol{\sigma}}}$
— 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" ,

80.11.3 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" .

80.11.4 Liste des ddl possibles aux noeuds

On donne ici la liste exhaustive des ddl possibles aux noeuds (version V. 6.784) : .

X1 , X2 , X3, EPAIS , TEMP , UX, UY, UZ , V1 , V2 , V3, PR,
GAMMA1, GAMMA2, GAMMA3, SIG11,SIG22,SIG33,SIG12,SIG23,SIG13,
ERREUR, EPS11,EPS22,EPS33,EPS12,EPS23,EPS13,
DEPS11,DEPS22,DEPS33,DEPS12,DEPS23,DEPS13, PROP_CRISTA,
DELTA_TEMP,FLUXD1,FLUXD2,FLUXD3,R_TEMP,
GRADT1,GRADT2,GRADT3,DGRADT1,DGRADT2,DGRADT3,
R_X1,R_X2,R_X3,R_EPAIS,R_V1,R_V2,R_V3,R_GAMMA1,R_GAMMA2,R_GAMMA3

Remarques L'existence d'un ddl dépend du problème en cours !

Avec les explications suivantes :
— Xi : coordonnées du points,
— `EPAIS` : épaisseur,
— `TEMP` : température,
— `UX,UY,UZ` : déplacements,
— Vi : coordonnées de la vitesse,
— `PR` : pression,
— GAMMAi : coordonnées de l'accélération,
— SIGij : composantes du tenseur contrainte,
— `ERREUR` : erreur
— EPSij : composantes du tenseur de déformation,

— DEPSij : composantes du tenseur de vitesse de déformation,
— PROP_CRISTA : proportion de cristalinité,
— DELTA_TEMP : Δ température,
— FLUXDi : coordonnées du flux thermique,
— R_TEMP : réaction due à une température fixée,
— GRADT i : coordonnées du gradient de température,
— DGRADT i : coordonnées de l'incrément du gradient de température,
— R_Xi : coordonnées de la réaction due à un déplacement bloqué,
— R_EPAIS : réaction due à une épaisseur bloquée,
— R_Vi : coordonnées de la réaction due à une vitesse bloquée,
— R_GAMMAi : coordonnées de la réaction due à une accélération bloquée.

80.12 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 orthonormé, 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 !

80.13 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 calcule ce volume et en particulier son évolution dans le temps. Pour prendre en compte le problème des volumes partiels, 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

81 Utilitaires

81.1 Fonctions 1D

Pour certaines parties de la mise en données telles que pour les lois de comportement (fonction de charge) ou plus habituellement pour imposer des conditions limites qui varient pendant le chargement , il est possible d'utiliser des fonctions 1D. Différents 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 (308) donne la liste des fonctions disponibles.

TABLE 308 – liste des fonctions 1D

nom	commentaire simplifié	référence
COURBEPOLYLINEAIRE_1_D	ensemble de point reliés par des segments	81.1.1
CPL1D	ensemble de point reliés par des segments	81.1.2
COURBE_EXPOAFF	$f(x) = \gamma + \alpha(\|x\|^n)$	81.1.3
COURBE_UN_MOINS_COS	$f(x) = c/2(1 - cos((x-a)\Pi/(b-a)))$	81.1.4
COURBEPOLYNOMIALE	$f(x) = \sum_{i=1}^{n} a_i x^i$	81.1.5
F1_ROND_F2	$f(x) = f1 \ \ o \ \ f2(x)$	81.1.6
F1_PLUS_F2	$f(x) = f1(x) \quad + \quad f2(x)$	81.1.7
F_CYCLIQUE	courbe qui se répète à intervalles réguliers avec un facteur d'amplification multiplicatif	81.1.8
F_CYCLE_ADD	courbe qui se répète à intervalles réguliers avec un facteur d'amplification additif	81.1.9
F_UNION_1D	courbe qui rassemble plusieurs, chacune défini sur un intervalle différent	81.1.10
COURBE_TRIPODECOS3PHI	$f(x) = (1. + \gamma \ cos(3 \ x))^{(-n)}$	81.1.11
COURBE_SIXPODECOS3PHI	$f(x) = (1. + \gamma \ (cos(3 \ x))^2)^{(-n)}$	81.1.12
COURBE_EXPO_N	$f(x) = (\gamma + \alpha \ x)^n$	81.1.13
COURBE_EXPO2_N	$f(x) = (\gamma + \alpha \ x^2)^n$	81.1.14
COURBE_RELAX_EXPO	$f(x) = (a - b) \ exp(-c \ x) + b)$	81.1.15
COURBE_COS	$f(x) = cos(x)$	81.1.16
COURBE_SIN	$f(x) = sin(x)$	81.1.17
COURBE_TANH	$f(x) = a + b \ tanh((x-c)/d)$	81.1.18
COURBEPOLYHERMITE_1_D	Interpolation de type Hermite par morceau	81.1.19
COURBE_EXPRESSION_LITTERALE_1D	définition littérale d'une courbe (analytique)	81.1.20
COURBE_EXPRESSION_LITTERALE_AVEC_DERIVEE_1D	définition littérale d'une courbe (analytique) avec ses dérivées première et seconde	81.1.21

81.1.1 Fonction polylinéaire

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 cet exemple, x=0., y=100. .La table (309) donne un exemple de déclaration d'une courbe polylinéaire pour une fonction d'écrouissage plastique de mot clé loi_ecrouissage (cf.51.4.3)

TABLE 309 – Exemple de déclaration d'une fonction d'écrouissage plastique polylinéaire

```
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 initial 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.

Durant l'utilisation de la fonction, lorsque x excède l'abscisse maximale des points enregistrés, le résultat de la fonction est extrapolé à l'aide de la ligne qui passe par les deux derniers points. De la même manière, lorsque x est inférieur à l'abscisse du premier point enregistré, le résultat est extrapolé à l'aide de la ligne qui passe par les deux premiers points indiqués.

Remarque : si l'on veut un fonctionnement tel que la fonction ne change pas de valeur après le dernier point, il faut entrer deux points d'abscisse différentes mais ayant le même y. Même chose avant le premier point.

La table (310) donne un exemple de déclaration de courbes polylinéaires avec translations initiales.

81.1.2 Fonction polylineaire-simple

Fonction identique à celle définie par le mot clé " COURBEPOLYLINEAIRE_1_D ". En fait c'est une version simplifiée qui permet de définir une courbe polylinéaire d'une manière compacte, mais le fonctionnement est identique au cas " COURBEPOLYLINEAIRE_1_D ". La table (311) 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 pairs de x et y, n étant quelconque, terminé par le mot clé : " FdlP ".

TABLE 310 – Exemple de déclaration d'une courbe polylinéaire avec translations initiales

```
courbe_2    COURBEPOLYLINEAIRE_1_D
decalageX_= 10. decalageY_= 3.
# def des points constituants la courbe
    Debut_des_coordonnees_des_points
     Coordonnee dim= 2 0. 0.
     Coordonnee dim= 2 1. 0.5
    Fin_des_coordonnees_des_points
```

TABLE 311 – Exemple de déclaration d'une courbe polylinéaire simplifiée

```
courbe1 CPL1D DdlP 0. 100000. 100. 50000.0  FdlP
```

81.1.3 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 (312) donne un exemple de déclaration de la fonction.

TABLE 312 – 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
```

81.1.4 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 (313) donne un exemple de déclaration de la fonction.

81.1.5 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 (314) donne un exemple de déclaration de la fonction. Le degré est quelconque.

TABLE 313 – 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.
```

TABLE 314 – 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
```

81.1.6 Fonctions composées : $f(x) = f1 \ \ o \ \ f2(x)$

Fonctions composées : $f(x) = f1 \ \ o \ \ f2(x)$. Les tables (315) et (316) donnent deux exemples de déclaration de fonctions composées. Les 2 fonctions à composer peuvent être définis soit explicitement cf.(316) soit via des noms de fonctions déjà définis cf.(315).

TABLE 315 – 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.

```
courbe_exemple1    F1_ROND_F2     # nom de la courbe
   courbe1= fonction_temperature   # def  de la premiere courbe par un nom de  referenc
   courbe2= transfo1               # def  de la seconde courbe par un nom de reference
```

TABLE 316 – Exemple de déclaration d'une fonction composée $f(x) = f1of2(x)$. Les 2 fonctions sont définies ici 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
```

81.1.7 Fonctions composées : $f(x) = f1(x) \quad + \quad f2(x)$

Fonctions composées : $f(x) = f1(x) \quad + \quad f2(x)$. 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 (317) donne un exemple de déclaration de fonction somme. Comme pour la fonction F1_ROND_F2 , les 2 fonctions à composer peuvent être définis soit explicitement cf.(316) soit via des noms de fonctions déjà définis cf.(315).

TABLE 317 – Exemple de déclaration d'une fonction composée $f(x) = f1(x) + f2(x)$. Les 2 fonctions sont définies par un nom qui est une référence à des fonctions définies 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
```

81.1.8 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, définie 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 $0 \leqslant (x - x_0) \leqslant cy \quad g(x) = y_0 + f(x - x_0)$
- pour $cy \leqslant (x - x_0) \leqslant 2\ cy \quad g(x) = y_0 + f(cy) + a\ f(x - cy - x_0)$
- pour $2\ cy \leqslant (x - x_0) \leqslant 3\ cy \quad g(x) = y_0 + (1 + a)\ f(cy) + a^2\ f(x - 2\ cy - x_0))$
- ...
- pour $n\ cy \leqslant (x - x_0) \leqslant (n + 1)\ cy \quad g(x) = y_0 + (1 + a + a^2 + ... + a^{n-1}) f(cy) + a^n\ f(x - n\ cy - x_0) = y_0 + (a^n - 1)/(a - 1) f(cy) + a^n\ f(x - n\ cy - x_0)$

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.(318) donne deux exemples d'utilisation d'une telle fonction. x_0 et y_0 représentent deux offsets possibles selon x et y. Par défaut

ces offsets sont nuls, mais leurs utilisations permettent par exemple de joindre plusieurs courbes.

TABLE 318 – 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 defaut
#   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.
```

81.1.9 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 (81.1.8) 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, définie 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 $0 \leqslant (x - x_0) \leqslant cy \quad g(x) = y_0 + a\ f(x - x_0)$
- n=2 : pour $cy \leqslant (x - x_0) \leqslant 2\ cy \quad g(x) = y_0 + f(cy) + 2\ a\ f(x - cy - x_0) = y_0 + a\ (f(cy) + 2\ f(x - cy - x_0))$
- n=3 : pour $2\ cy \leqslant (x-x_0) \leqslant 3\ cy \quad g(x) = y_0+a\ (1+2)\ f(cy)+3\ a\ f(x-2\ cy-x_0)) = y_0 + a\ ((1 + 2)f(cy) + 2\ f(x - cy - x_0))$
- ...
- cycle n : pour $(n-1)\ cy \leqslant (x-x_0) \leqslant n\ cy \quad g(x) = y_0+a\ (1+2+3+...+(n-1))f(cy)+ n\ a\ f(x-n\ cy-x_0) = y_0 + a\ (((n-1)^2 + n - 1)/2\ f(cy) + n\ f(x - (n-1)\ cy - x_0))$

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.(319) donne deux exemples d'utilisation d'une telle fonction. x_0 et y_0 représentent deux offsets possibles selon x et y. Par défaut ces offsets sont nuls, mais leurs utilisations permet par exemple de joindre plusieurs courbes.

81.1.10 Fonctions réunion de domaine

Cette fonction, que l'on appellera $g(x)$, permet de globaliser le comportement de n fonctions, que l'on appellera $fi(x)$, chacune de ces fonctions membres étant définit sur un intervalle propre $[x_i, x_{i+1}[$ différent des autres fonctions. Cependant l'ensemble des intervalles doit être continu de manière à représenter un grand intervalle plein (sans trou). La table (320) donne un exemple d'utilisation d'une telle fonction. Le nombre de fonctions n'est pas limité. La borne mini et la borne maxi sont facultatives (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 inférieur 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.

81.1.11 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 (321) donne un exemple de déclaration de la fonction.

TABLE 319 – 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 defaut
#   ainsi on doit indiquer la longueur du cycle, mais le facteur d'amplitude est
#   facultatif (par defaut = 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.
```

TABLE 320 – Exemple de déclaration d'une fonction union permettant de globaliser plusieurs fonctions déjà définis.

```
#.......................................
#   il s'agit d'une fonction qui aglomere plusieurs fonctions deja existantes
#   chaque fonction est definie sur un interval disjoint des autres
#   Les intervalles doivent se suivre
#   Xmin= precise la borne inferieure 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'intervalle 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'intervalle, on applique la premiere
#    (x<Xmin) ou la derniere fonction (x>Xmax)
```

TABLE 321 – Exemple de déclaration d'une fonction de type $f(x) = (1. + \gamma\ cos(3\ x))^{(-n)}$

```
f_coefficient COURBE_TRIPODECOS3PHI
# def des coeff de la courbe tripode
gamma= 0.9  n= 0.1
```

81.1.12 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 paire 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 (322) donne un exemple de déclaration de la fonction.

TABLE 322 – Exemple de déclaration d'une fonction de type $f(x) = (1.+\gamma\ (cos(3\ x))^2)^{(-n)}$

```
f_coefficient COURBE_SIXPODECOS3PHI
# def des coeff de la courbe tripode
gamma= 0.9  n= 0.1
```

81.1.13 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 (323) donne un exemple de déclaration de la fonction.

TABLE 323 – 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
```

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

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, permettent d'avoir des valeurs toujours positives et une fonction paire. 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 (324) donne un exemple de déclaration de la fonction.

81.1.15 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=∞, sous forme d'une tangence. 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 (325) donne un exemple de déclaration de la fonction.

TABLE 324 – 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
```

TABLE 325 – 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
```

81.1.16 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 $a \leqslant x \leqslant b$, $f(x) = e\ cos(\alpha x + \beta)$
- 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 (326) donne un exemple de déclaration de la fonction.

81.1.17 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

TABLE 326 – 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 defaut = -l'infini et + l'infini
     a= 0. b= 1. ampli= 2. alph= 3. bet= 0.5 en_degre_
```

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 $a \leqslant x \leqslant b$, $f(x) = e\ sin(\alpha x + \beta)$
- 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 (327) donne un exemple de déclaration de la fonction.

TABLE 327 – 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 defaut = -l'infini et + l'infini
     a= 0. b= 1. ampli= 2. alph= 3. bet= 0.5 en_degre_
```

81.1.18 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 (328) donne un exemple de déclaration de la fonction.

TABLE 328 – 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.
```

81.1.19 Fonction de type "interpolation d'Hermite" par morceau

Fonction de type "interpolation d'Hermite" par morceau et de mot clé " COURBEPOLYHERMITE_1_D " : construite à partir d'un ensemble de points relié par des polynômes cubiques, de telle manière que la fonction résultante est continue C1.

La fonction nécessite la donnée d'un ensemble de points et dérivées associées. 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 et dérivées, un enregistrement par ligne. Un point est constitué du mot clé " Coordonnee " suivi du mot clé " dim= " puis la dimension ici 3, l'abscisse, l'ordonnée, la dérivée, ainsi un point est constitué d'un x, d'un y et d'un y' (la dérivée). Un exemple de ligne représentant un point est :
Coordonnee dim= 3 0. 100. 2.
dans cette exemple, x=0., y=100. et $\frac{dy}{dx} = 2.$ .La table (329) donne un exemple de déclaration d'une courbe poly-Hermite 1D, avec une dérivée initiale et finale nulle, utilisables par exemple pour un chargement.

TABLE 329 – Exemple de déclaration d'une fonction de charge

```
fonction-de-charge COURBEPOLYHERMITE_1_D
  Debut_des_coordonnees_des_points
     Coordonnee dim= 3  0.   0.  0.
     Coordonnee dim= 3  1.   1.  0.
  Fin_des_coordonnees_des_points
```

Il est également possible d'introduire un décalage initial 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 (330) donne un exemple de déclaration de courbes poly-Hermite avec translations initiales.

81.1.20 Fonction analytique littérale

Il s'agit ici d'utiliser une expression littérale construite à partir de fonctions mathématiques classiques. Herezh++ utilise la bibliothèque muParser, pour "parser" l'expression lue et

TABLE 330 – Exemple de déclaration d'une courbe poly-Hermite avec translations initiale

```
courbe_2    COURBEPOLYHERMITE_1_D
decalageX_= 10. decalageY_= 3.
# def des points constituants la courbe
    Debut_des_coordonnees_des_points
     Coordonnee dim= 3 0. 0.  0.
     Coordonnee dim= 3 1. 0.5  0.
    Fin_des_coordonnees_des_points
```

ensuite utiliser la fonction demandée. Par exemple la table 331 donne un exemple de déclaration. Dans cette exemple on remarquera la fonction qui est précédée par le mot clé " `f(x)=` ". La variable de la fonction est toujours "x".

La formule doit terminer une ligne (il ne faut pas d'autre information entre la fin de la formule et la fin de la ligne).

L'expression qui suit s'appuie sur des fonctions et des opérations déjà existantes. La liste des fonctions existantes (cf. la documentation de muParser) est donnée par le tableau 332. De même on trouvera dans le tableau 334 la liste des opérations binaires possibles (cf. la documentation de muParser) et dans le tableau 334 la liste des opérations ternaires.

Outre la définition littérale de la fonction, plusieurs paramètres optionnels sont disponibles. Ces paramètres peuvent être définis dans un ordre quelconque. On peut ainsi indiquer :

1. " `a=` " suivi d'une valeur $\rightarrow$ pour x $\leqslant$ a $\Longrightarrow$ f(x) = f(a), par défaut a= -∞
2. " `b=` " suivi d'une valeur $\rightarrow$ pour x $\geqslant$ a $\Longrightarrow$ f(x) = f(b), par défaut b= +∞
3. " `delta_xSur_x=` " suivi d'une valeur numérique que l'on nommera "e" pour la suite, qui est utilisée pour le calcul des dérivées. Celles ci sont évaluées numériquement à l'aide des formules à trois points de troncature à l'ordre 2 (formules des différences finies centrées) :
$$f'(x) \approx \frac{f(x+\delta x) - f(x-\delta x)}{2.\delta x}$$
et
$$f"(x) \approx \frac{(f(x+\delta x) - 2f(x) + f(x-\delta x))}{(\delta x)^2}$$
avec $\delta x = max(|e.x|, e)$. Par défaut la valeur utilisée est calculée à partir du plus petit nombre numérique représentable exactement sur la machine.

La description doit se terminer par le mot clé : " `fin_parametres_courbe_expression_litterale_` "

Remarque : Lorsque la fonction littérale contient des signes < la chaîne de caractère qui suit n'est pas interprétée comme un nom de fichier contrairement aux entrées standards d'Herezh.

TABLE 331 – Exemple de déclaration d'une courbe définie sous forme d'une expression littérale

```
dep_impose    COURBE_EXPRESSION_LITTERALE_1D
# a= 0. b= 4. f(x)=  0.01 * cos(x) + 0.01 *ln( x+0.1 )  # exemple avec bornes
   f(x)=  0.01 * cos(x) + 0.01 *ln( x+0.1 )  # exemple sans bornes
 fin_parametres_courbe_expression_litterale_
```

TABLE 332 – liste des fonctions mathématiques disponibles

Nom	nombre d'arguments	Explications
sin	1	sine function
cos	1	cosine function
tan	1	tangens function
asin	1	arcus sine function
acos	1	arcus cosine function
atan	1	arcus tangens function
sinh	1	hyperbolic sine function
cosh	1	hyperbolic cosine
tanh	1	hyperbolic tangens function
asinh	1	hyperbolic arcus sine function
acosh	1	hyperbolic arcus tangens function
atanh	1	hyperbolic arcur tangens function
log2	1	logarithm to the base 2
log10	1	logarithm to the base 10
log	1	logarithm to the base 10
ln	1	logarithm to base e (2.71828...)
exp	1	e raised to the power of x
sqrt	1	square root of a value
sign	1	sign function -1 if x<0 ; 1 if x>0
rint	1	round to nearest integer
abs	1	absolute value
min	var.	min of all arguments
max	var.	max of all arguments
sum	var.	sum of all arguments
avg	var.	mean value of all arguments

81.1.21 Fonction avec dérivées premières et seconde analytique littérale

Il s'agit ici d'utiliser pour la fonction, ses dérivées première et seconde, une expression littérale construite à partir de fonctions mathématiques classiques. Herezh++ utilise la bibliothèque muParser, pour "parser" l'expression lue et ensuite utiliser la fonction demandée. Par exemple la table 335 donne un exemple de déclaration. Dans cet exemple on

TABLE 333 – liste des opérations mathématiques binaires disponibles. **NB** L'opération = est particulière, il s'agit de transférer le continue de droite dans la variable de gauche, donc ne peut s'appliquer qu'à des variables.

Opération	description	priorité
=	assignement	-1
&&	logical and	1
||	logical or	2
<=	less or equal	4
>=	greater or equal	4
! =	not equal	4
==	equal	4
>	greater than	4
<	less than	4
+	addition	5
−	subtraction	5
*	multiplication	6
/	division	6
^	raise x to the power of y	7

TABLE 334 – liste des opérations mathématiques ternaires disponibles

Opération	description	remarque
? :	if then else operator	C++ style syntax

remarquera la fonction qui est précédée par le mot clé " `f(x)=` ". La variable de la fonction est toujours " `x` ". L'expression qui suit s'appuie sur des fonctions et des opérations déjà existantes. De même on doit indiquer la formule permettant de calculer la dérivée première précédée du mot clé : " `f'(x)=` ", et la formule donnant la dérivée seconde précédée du mot clé : " `f''(x)=` ".

Chaque formule avec son mot clé doit terminer une ligne. Le plus simple est donc de mettre une formule par ligne !.

La liste des fonctions existantes (cf. la documentation de muParser) est donnée par le tableau 332. De même on trouvera dans le tableau 334 la liste des opérations binaires possibles (cf. la documentation de muParser) et dans le tableau 334 la liste des opérations ternaires.

Outre la définition littérale de la fonction et de ses dérivées première et seconde, plusieurs paramètres optionnels sont disponibles. Ces paramètres peuvent être définis dans un ordre quelconque. On peut ainsi indiquer successivement :

1. "a=" suivi d'une valeur → pour x $\leqslant$ a $\Longrightarrow$ f(x) = f(a), par défaut a= -∞
2. "b=" suivi d'une valeur → pour x $\geqslant$ a $\Longrightarrow$ f(x) = f(b), par défaut b= +∞

La description doit se terminer par le mot clé : " `fin_parametres_courbe_expression_litterale_` "

Remarque : Lorsque la fonction littérale contient des signes, < la chaîne de caractère qui suit n'est pas interprétée comme un nom de fichier contrairement aux entrées standards

TABLE 335 – Exemple de déclaration d'une courbe définie sous forme d'une expression littérale

```
dep_impose    COURBE_EXPRESSION_LITTERALE_AVEC_DERIVEE_1D
        # 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 defaut = -l'infini et + l'infini
        a= 0. b= 1.
        f(x)= (x^2+3.)+cos(3.*x)
        f'(x)= 2.*x -3.*sin(x)
        f"(x)= 2.-9.*cos(x)
   fin_parametres_courbe_expression_litterale_
```

d'Herezh.

81.2 Fonctions nD

De manière analogue aux courbes 1D, cette partie est dédiée à la définition de fonctions multi-dimensionnelles.

L'ensemble des fonctions nD commence par le mot clé : `les_fonctions_nD` mis sur une seule ligne. Ensuite, sur les lignes qui suivent, 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 (336) donne la liste des fonctions disponibles.

TABLE 336 – liste des fonctions nD

nom	commentaire simplifié	référence
FONCTION_EXPRESSION_LITTERALE_nD	définition littérale d'une fonction (analytique)	81.2.2
FONCTION_COURBE1D	fonction utilisant une courbe 1D	81.2.3
FONC_SCAL_COMBINEES_ND	combinaison analytique de fonctions de base	81.2.4
FONCTION_EXTERNE_ND	fonction définie à l'extérieur d'Herezh++	81.2.6

81.2.1 Remarques sur l'utilisation de variables globales

Toutes les fonctions nD peuvent avoir pour arguments des variables globales (cf.88.1). Il faut cependant faire attention à la présence ou nom de la variable globale utilisée qui peut éventuellement ne pas être calculée ou existé pour certain algorithme (ou type de problème).

81.2.2 Fonction analytique littérale à n variables

Il s'agit ici d'utiliser une expression littérale construite à partir de fonctions mathématiques classiques. Herezh++ utilise la bibliothèque muParser, pour "parser" l'expression lue et ensuite utiliser la fonction demandée.

Syntaxe : On commence par définir les variables.

La liste de variable (5 au maximum) peut-être définie de deux manières :

1. soit les variables sont définies une à une via le mot clé : " `un_argument=` " suivi du nom de la variables,
2. soit toutes les variables sont définies sous forme d'une liste de nom bornée de part et d'autre par deux balises selon la syntaxe :

```
deb_list_var_   nom1 nom2 ... fin_list_var_
```

où " `deb_list_var_` " et " `fin_list_var_` " sont les balises et "nom1, nom2 etc. sont les noms des variables.

Ensuite on définit la formule analytique, qui doit commencer par le mot clé " `fct=` ". La formule doit terminer une ligne (il ne faut pas d'autre information entre la fin de la formule et la fin de la ligne).

L'expression s'appuie sur des fonctions et des opérations déjà existantes. La liste des fonctions existantes (cf. la documentation de muParser) est donnée par le tableau 332. De

même on trouvera dans le tableau 334 la liste des opérations binaires possibles (cf. la documentation de muParser) et dans le tableau 334 la liste des opérations ternaires. La description doit se terminer par le mot clé : " fin_parametres_fonction_expression_litterale_ "

La table 337 donne un exemple de déclaration. Dans cet exemple on remarquera la fonction qui est précédée par le mot clé "fct=".

TABLE 337 – Exemple de déclaration d'une liste de fonctions nD.

```
les_fonctions_nD   #------------

# exemple de definition d'une fonction Fonction_expression_litterale_nD
#                 f(i,j) = une expression de i et j
 fonction1  FONCTION_EXPRESSION_LITTERALE_nD
   un_argument= i un_argument= j
   fct=  (i < 2) ? ((j < 1000) ? -50. : 0. ): ((j < 2000) ? -10. : 0. )
   fin_parametres_fonction_expression_litterale_
```

Il est possible d'indiquer un niveau d'affichage spécifique à l'exécution de la fonction via le paramètre permet_affichage_ suivi d'une valeur comprise entre 0 et 10. En particulier pour un niveau > 6 on peut voir afficher les valeurs des arguments et du retour de la fonction à chaque appel de la fonction.

Le mot clé permet_affichage_ et le niveau correspondant doivent-être indiqués avant le mot clé fin_parametres_fonction_expression_litterale_ sur une ligne particulière.

Remarque : Lorsque la fonction littérale contient des signes, $<$ la chaîne de caractère qui suit n'est pas interprétée comme un nom de fichier contrairement aux entrées standards d'Herezh.

81.2.3 Fonction utilisant une courbe 1D

L'objectif de cette fonction est de pouvoir disposer au sein des fonctions nD de l'ensemble des courbes 1D déjà définies par ailleurs. Ensuite, il est possible de mixer ces courbes sous une forme quelconque via une fonction " FONC_SCAL_COMBINEES_ND " (cf.FONCTSCALCOMBINEES).

La déclaration de la fonction est très simple. La table 338 présente deux exemples de déclaration. Dans le premier exemple, la déclaration est réduite au minimum :

1. déclaration du nom de la fonction suivi du mot clé " FONCTION_COURBE1D "
2. mot clé "courbe= " suivi du nom d'une courbe 1D qui existe par ailleurs
3. mot clé " un_argument= " suivi du nom de l'argument, qui peut être une variable globale,
4. mot clé de fin de déclaration " fin_fonction_courbe1D_ "

Il est par ailleurs possible également de déclarer une courbe 1D à la place de son nom, suivant la syntaxe propre à la courbe que l'on veut déclarer. Le second exemple illustre cette possibilité.

TABLE 338 – Deux exemples de déclaration d'une fonctions nD qui utilise une courbe 1D déjà existante

```
#.........................................
#   il s'agit d'une fonction qui utilise une courbe 1D
#   cette dernière peut soit deja exister, soit être définie a la suite
#   exemple 1: cas d'une courbe qui existe déjà sous le nom: cf1
    fct1_exemple    FONCTION_COURBE1D
       courbe= cf1
       un_argument= x
     fin_fonction_courbe1D_
#
#   exemple 2: cas d'une courbe qui est définit à la suite
    fct2_exemple    FONCTION_COURBE1D
       un_argument= x
       courbe= COURBE_EXPRESSION_LITTERALE_1D
          a= 0. b= 1. f(x)= (x^2+3.)/cos(x)+log((1.+x)/(1.+x^3))*23.-8.
        fin_parametres_courbe_expression_litterale_
     fin_fonction_courbe1D_
#
```

81.2.4 Fonction analytique scalaire, combinant plusieurs fonctions de base nD

L'objectif est de pouvoir combiner le résultat de plusieurs fonctions de base sous la forme d'une expression analytique quelconque qui mélange des appels aux fonctions de base, des variables et des constantes.

L'idée est en particulier de permettre :

— la réutilisation de fonctions de bases déjà définies par ailleurs,

— de simplifier une expression complexe en sous ensembles plus simple. Chaque sous ensembles sera alors représenté par une fonction de base.

La définition d'une fonction " FONC_SCAL_COMBINEES_ND " s'effectue selon la logique suivante :

1. Tout d'abord on définit les fonctions de base : deux cas pour chaque fonction de base :
   (a) soit la fonction existe déjà, i.e. a déjà été définit par ailleurs,
   (b) soit on veut définir une fonction interne à la fonction combinée.

   Remarque : dans le cas a) la fonction existante peut être utilisée plusieurs fois par exemple pour la définition de plusieurs fonctions combinées. Dans le cas b) la fonction définie en interne n'est pas accessible en dehors de la fonction combinée qui est en cours de définition.
   Syntaxe cas a) :

   ```
   fct_base= nom_fonction_existante
   ```

   où " fct_base= " est le mot clé et "nom_fonction_existante" est le nom de la fonction qui existe par ailleurs.
   Syntaxe cas b) :

   ```
   fct_base= nom_type_fonction ident_interne_ nom_identificateur
   ```

   où où " fct_base= " est le mot clé, "nom_type_fonction" est une chaine de caractères donnant le type de la fonction que l'on veut définir, " ident_interne_ " est le mot clé qui indique que c'est une fonction interne, "nom_identificateur" est le nom interne qu'on souhaite pour la fonction et qui sera ensuite utilisé dans l'expression analytique globale de combinaison.
   La liste des fonctions de base doit se terminer par le mot clé (sur une ligne seule) :

   ```
   fin_fcts_interne_fonction_combinee_
   ```

2. On définit ensuite la liste des variables de la fonction globale. Cette liste inclus par défaut tout d'abord :
   (a) les variables internes de chaque fonction de base, dans l'ordre ou les fonctions sont déclarées. On n'a donc pas à les re-déclarer.
   (b) de nouvelles variables propres à la fonction globale, avec les 2 syntaxes possibles :

i. soit une variable par ligne selon :

```
un_argument= nom_argument
```

où " un_argument= " est le mot clé , "nom_argument" est le nom de la variable

ii. soit une liste de variables selon la syntaxe

```
deb_list_var_   nom1 nom2 ... fin_list_var_
```

où " deb_list_var_ " et " fin_list_var_ " sont les balises encadrant la liste et "nom1 nom2 ..." sont les nom des variables

Remarque : les variables peuvent être des grandeurs locales ou globales (cf.88.1).

3. on définit l'expression de la fonction globale. L'expression s'écrit une une ligne (qui peut-être fractionnée via le caractère cf. doc). Elle doit commencer par le mot clé " fct= " suivi d'une expression analytique quelconque (cf. doc fonction analytique). L'expression peut contenir :

   — les noms de variables,

   — le nom des fonctions ou de leur identificateur.

   La liste des variables et la définition de l'expression générale doit se terminer par le mot clé (sur une ligne seule) :

```
fin_parametres_fonction_combinee_
```

Les tables 339 et 340 présentent trois exemples de déclaration.

Il est possible d'indiquer un niveau d'affichage spécifique à l'exécution de la fonction via le paramètre permet_affichage_ suivi d'une valeur comprise entre 0 et 10. En particulier pour un niveau > 6 on peut voir afficher les valeurs des arguments et du retour de la fonction à chaque appel de la fonction.

Le mot clé permet_affichage_ et le niveau correspondant doivent-être indiqués avant le mot clé fin_parametres_fonction_expression_litterale_ sur une ligne particulière.

81.2.5 Fonction nD vectorielle

Cette partie concerne les fonctions nD dans lesquelles le retour de la fonction est définie au final par une expression analytique qui suit le mot clé : fct=

Par défaut, la fonction retourne une seule valeur, celle correspondante au calcul de l'expression qui suit le mot clé fct= . Il s'agit donc d'une fonction scalaire.

Il est possible de définir une fonction vectorielle. Pour cela, à la suite du mot clé fct= on indique successivement plusieurs expressions analytiques, chacune séparée des autres par une virgule. Il s'agit alors d'une fonction vectorielle.

Par exemple pour une fonction qui ramène 3 composantes on aura :

fct= expression1 , expression2 , expression3

Ce type de fonction peut-être utilisée par exemple, pour imposer des contions limites ou encore un repère d'anisotropie.

TABLE 339 – Deux exemples de déclaration d'une fonctions nD FONCTION_EXPRESSION_LITTERALE_nD ? . La première utilise deux fonctions existantes par ailleurs. La seconde utilise deux fonctions définies in situe.

```
#   exemple 1: def de la fonction addition_simple :
#              somme de deux fonctions de base
#
      addition_simple  FONC_SCAL_COMBINEES_ND
         fct_base= montee # fct 1
         fct_base= relax # fct 2
       fin_fcts_interne_fonction_combinee_
       fct= montee + relax
      fin_parametres_fonction_combinee_

#  NB: ici les fonctions montee et relax doivent exister par ailleurs
#      les parametre de addition_simple sont ceux de montee + ceux de
#       relax
#
#   exemple 2: def de la fonction addition_pondere :
#              somme ponderee de deux fonctions de base definies en interne
#
      addition_pondere  FONC_SCAL_COMBINEES_ND
         fct_base= FONCTION_EXPRESSION_LITTERALE_nD  ident_interne_ F1
           un_argument= x
           fct=  sin(x)
          fin_parametres_fonction_expression_litterale_
         fct_base= FONCTION_EXPRESSION_LITTERALE_nD  ident_interne_ F2
           un_argument= y
           fct=  cos(y)
          fin_parametres_fonction_expression_litterale_
       fin_fcts_interne_fonction_combinee_
       deb_list_var_   e ENERGIE_CINETIQUE fin_list_var_
       fct= (1.-e^2)*(ENERGIE_CINETIQUE-1.)*F1 + e^2*F2
      fin_parametres_fonction_combinee_

#  NB: . ici les fonctions de base F1(x) et F2(y) sont definies en interne
#      . les parametre de addition_pondere sont dans l'ordre:
#       x, y, e, ENERGIE_CINETIQUE
#      . le parametre ENERGIE_CINETIQUE est une grandeur globale
```

TABLE 340 – Un exemple de déclaration d'une fonction qui combine des fonctions existantes et l'utilisation d'une variable globale ENERGIE_CINETIQUE

```
#
#   exemple 3: def de choix_selon_energie_cinetique  :
#     -> choix entre deux fonctions suivant la valeur de l'energie cinetique
#
      choix_selon_energie_cinetique  FONC_SCAL_COMBINEES_ND
         fct_base= F1 # fct 1
         fct_base= F2 # fct 2
       fin_fcts_interne_fonction_combinee_
           un_argument= ENERGIE_CINETIQUE
       fct=  (ENERGIE_CINETIQUE < 2) ? F1 : F2
      fin_parametres_fonction_combinee_
#
#  NB: ici les fonctions F1 et F2 doivent exister par ailleurs
#      les parametre  sont ceux de F1 + ceux de F2 + ENERGIE_CINETIQUE
#      si ENERGIE_CINETIQUE < 2 alors F1 sinon F2
```

81.2.6 Fonction à n variables définie à l'extérieur d'Herezh++ et dialoguant via des pipes nommés

Il s'agit ici de permettre à l'utilisateur d'agir directement sur le fonctionnement d'Herezh via l'utilisation de fonction nD spécifiques définies en dehors d'Herezh++.

En effet, les fonctions nD permettent d'agir sur une grande partie des étapes de calcul. Elles peuvent piloter par exemple :

- certains paramètres des algorithmes
- certains paramètres des lois de comportement
- l'intensité et la direction éventuelle des chargements
- l'application des conditions limites
- ...

La méthode la plus simple pour définir une fonction nD, est de la décrire dans la mise en donnée via une des méthodes proposées (cf. 81.2).

Dans le cas où les méthodes proposées ne sont pas suffisantes pour l'utilisateur (averti !) il est possible de définir une fonction nD à l'extérieur d'Herezh++ .

Les principaux éléments de la définition d'une fonction externe sont les suivants :

- on indique dans la mise en données (i.e. dans le fichier .info) le nom et les paramètres de la fonction externe
- on indique également dans quelle partie du calcul, la fonction nD est utilisée
- on définit un programme externe, construit via un langage quelconque : C++, C, perl, python, octave..., qui doit être capable de lire et d'écrire dans un pipe nommé (= tube Unix ou linux). En fait un pipe nommé correspond à un fichier en mémoire vive ce qui permet d'avoir des temps d'écriture et de lecture très rapides.
- la mise en route d'Herezh++ conduit à la construction automatique de 2 pipes nommés : un pipe d'écriture et un pipe de lecture. Puis au moment de l'utilisation de la fonction externe, Herezh écrit sur le pipe d'écriture, les variables que l'utilisateur a indiquées dans la mise en données. Ensuite l'exécution d'Herezh s'arrête en attendant les résultats de la fonction externe sur le pipe de lecture.
- le programme externe est alors sensé :
  - lire la valeur des variables sur le pipe d'écriture crée par Herezh
  - puis écrire les résultats attendus par Herezh, sur le pipe de lecture.
- l'exécution d'Herezh redémarre alors après lecture des résultats sur le pipe de lecture, ceci jusqu'à un nouvel appel de la fonction externe.

La syntaxe de la mise en donnée est très similaire à celle de la définition d'une fonction nD analytique (cf. 81.2.2)

La table 341 donne un exemple de déclaration.

- le mot clé `FONCTION_EXTERNE_ND` indique qu'il s'agit d'une fonction externe
- la définition d'un ou plusieurs arguments, suit la même syntaxe que celle des fonctions internes (cf. par exemple 81.2.2)

TABLE 341 – Exemple de déclaration d'une fonction externe nD.

```
# exemple de definition d'une fonction Fonction_externe_nD qui renvoi une seule valeur (un reel
 cont_precision  FONCTION_EXTERNE_ND
   un_argument= norme_de_convergence
   un_argument= algo_global_actuel
   un_argument= compteur_iteration_algo_global
   nb_double_ret_  1
   permet_affichage_ 8
   fin_parametres_fonction_externe_
```

— le mot clé `nb_double_ret_` , obligatoire, indique le nombre de réels calculés et ramenés par la fonction externe,

— `permet_affichage_` permet de gérer le niveau de commentaire de la fonction (écriture, lecture).

Remarques

— Ici il n'y a pas de limitation sur le nombre de variables,

— le nombre de résultats doit-être cohérent avec son utilisation par Herezh,

— toutes les grandeurs, en entrée et en sortie sont des réels.

La suite concerne un exemple de définition de fonction externe réalisé en C++.

La fonction est définie à travers une classe "F_externe_nD" qui comporte comme méthodes principales :

— deux méthodes publiques : "InitialisationPipesNommes()" qui initialise les pipes , "Execution()" qui calcule la fonction,

— deux méthodes protégées : "LecturePipe()" qui lit des données dans le pipe d'écriture d'Herezh, "EcriturePipe()" qui écrit les résultats dans le pipe de lecture d'Herezh.

TABLE 342 – Exemple de programme principal externe .

```
    //
//  main.cpp
//  essai_fct_externe
//
//  Created by gerard rio on 04/03/2020.
//  Copyright © 2020 gerard rio. All rights reserved.
//

#include <iostream>
// --- fichier pour le tube ------
#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
#include "F_externe_nD.h"

int main(int argc, const char * argv[]) {

 // on définit une instance de fonction externe
 F_externe_nD fct;
 // init des pipes
 fct.InitialisationPipesNommes();
 fct.ChangeNiveauAffichage(4);
 // on exécute
 int toto=1;
 while (toto) // boucle infinie
  fct.Execution();

 return 0;
}
```

TABLE 343 – Exemple d'une classe pour fonction externe : partie .h

```
// fichier : F_externe_nD.h
// classe  : F_externe_nD
/************************************************************************
*           UNIVERSITE DE BRETAGNE SUD (UBS) --- LORIENT               *
*************************************************************************
*          IRDL - Equipe DE GENIE MECANIQUE ET MATERIAUX (EG2M)        *
* Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex         *
* tel. 02.97.32.82.47                 http://IRDL.fr                   *
*************************************************************************
 *     DATE:        4/03/2020                                          *
 *                                                                $     *
 *     AUTEUR:      G RIO   (mailto:gerard.rio@univ-ubs.fr)            *
 *                  phone  0297874573, fax : 0297874588                *
 *                                                                $     *
 *     PROJET:      Herezh++                                           *
 *                                                                $     *
 ************************************************************************
 *     BUT:    exemple de fct nd externe                              *
 *             , définie par l'utilisateur, via 2 pipes nommés.       *
 *                                                                $     *
 ************************************************************************/
 #ifndef FONCTION_EXTERNE_ND_H
#define FONCTION_EXTERNE_ND_H
# include <iostream>
#include <fstream>
using namespace std;  //introduces namespace std
#include <string.h>
#include <string>
#include <vector>
// définition d'une union qui lie les réels, les entiers et les caractères
union Tab_car_double_int_1
  {  char    tampon[928];
     double  x[116];
     int     n[232];
  } ;
class F_externe_nD
{
  public :
    // CONSTRUCTEURS :
    F_externe_nD();
    // DESTRUCTEUR :
    ~F_externe_nD();
    // METHODES PUBLIQUES :
    // changement de l'affichage
    void ChangeNiveauAffichage(int niveau)
      {permet_affichage=niveau;};
     // initialisation des pipes
    void InitialisationPipesNommes();
    // exécution de la Fonction
    void Execution() ;
  protected :
    // VARIABLES PROTEGEES :
    // -- variables pour les tubes nommés ----------
    string nom;
    // nom du tube nommé pour l'envoi des données
    string envoi;
    // nom du tube nommé pour la réception des données
    string reception;
    Tab_car_double_int_1 t_car_x_n; // tableau de caractères, réels et entiers
    int permet_affichage;
    int nb_variable,nb_retour;
     // METHODES PROTEGEES :
    // lecture des données sur le pipe
    void LecturePipe();
    // écriture des résultats sur le pipe
    void EcriturePipe();
 };
#endif
```

TABLE 344 – Exemple d'une classe pour fonction externe : partie 1 .cc

```
#include "F_externe_nD.h"
using namespace std;  //introduces namespace std

// --- fichier pour les tubes ------
#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
#include <math.h>

    // CONSTRUCTEURS :
F_externe_nD::F_externe_nD() :
 reception("cont_precision_envoi_Hz.FIFO"),envoi("cont_precision_reception_Hz.FIFO")  // a modifier en fonction du nom donné par
 // il suffit de lancer Herezh avec un niveau d'affichage pour la fonction externe > 0
 // Herezh écrit alors sur cout le nom des pipes nommés
 ,nom("cont_precision")
 ,permet_affichage(1)
 ,nb_variable(0),nb_retour(0)
  {};
    // DESTRUCTEUR :
F_externe_nD::~F_externe_nD()
  {};

// METHODES PUBLIQUES :
// exécution de la Fonction
void F_externe_nD::Execution()
  {
    // lecture des données sur le pipe
    LecturePipe();
    // exemple de fonction: ici on calcul le sinus de la première variable
    t_car_x_n.x[1] = sin(t_car_x_n.x[1]);
    // on ramène une seule valeur
    t_car_x_n.n[1] = nb_retour = 1;
    // écriture des résultats sur le pipe
    EcriturePipe();
  };
```

TABLE 345 – Exemple d'une classe pour fonction externe : partie 2 .cc

```
// initialisation des pipes
void F_externe_nD::InitialisationPipesNommes()
 { try
    { // pipe de sortie
      // on test l'existence du pipe
      struct stat buf;
      int tub=0; // init
      int etat = stat(envoi.c_str(), &buf);
       int tube_a_cree = 0;
      if (etat == -1) // cela veut dire que la fonction stat n'a pas fonctionné
        // on part sur le principe que les tubes n'existent pas
        {tube_a_cree=1;}
      else if (!S_ISFIFO(buf.st_mode))
        {tube_a_cree=1;};
      if (tube_a_cree ) // cas où le tube n'existe pas
       {tub = mkfifo (envoi.c_str(), S_IRWXU);
        if (tub == -1 ) // gestion d'erreur éventuelle
         {std::cout << "\n *** erreur en creation du tube nomme: " << envoi << ", de sortie  pour la "
               << " fonction externe "<< nom
               << "\n F_externe_nD::InitialisationPipesNommes() "<<flush;
         }
        else
         { cout << "\n creation du pipe nomme: "<< envoi << " pour la "
                << " fonction externe "<< nom << flush;
         } ;
        close (tub); // fermeture du tampon
       }
     else
       { cout << "\n utilisation du pipe nomme: "<< reception << " pour la "
              << " fonction externe "<< nom << flush;
       } ;
     tub = 0;         // réinit
     tube_a_cree = 0; // réinit

     // ouverture du tube nommé en lecture
     etat = stat(reception.c_str(), &buf);
     if (etat == -1) // cela veut dire que la fonction stat n'a pas fonctionné
       // on part sur le principe que les tubes n'existent pas
       {tube_a_cree=1;}
     else if (!S_ISFIFO(buf.st_mode))
       {tube_a_cree=1;};
     if (tube_a_cree ) // cas où le tube n'existe pas
      {tub = mkfifo (reception.c_str(), S_IRWXU);
       if (tub == -1 ) // gestion d'erreur éventuelle
        {cout << "\n *** erreur en creation du tobe nomme: " << reception << ", de lecture d'herezh pour la "
              << " fonction externe "<< nom
              << "\n F_externe_nD::InitialisationPipesNommes() "<<flush;
         exit(1);
        }
       else
        { cout << "\n creation du pipe nomme: "<< reception << " pour la "
              << " fonction externe "<< nom << flush;
        } ;
       close (tub); // fermeture du tampon
      }
     else
      { cout << "\n utilisation du pipe nomme: "<< reception << " pour la "
             << " fonction externe "<< nom << flush;
      } ;
    }
   catch(...)
    {cout << "\n *** erreur inconnue lors de l'initialisation des pipes I/O pour la "
          << " fonction externe "<< nom
          << "\n F_externe_nD::InitialisationPipesNommes() "<<flush;
     exit(1);
    };
 };
```

TABLE 346 – Exemple d'une classe pour fonction externe : partie 3 .cc

```
// lecture des données
void F_externe_nD::LecturePipe()
{
 try
  {//  Creation d'un processus de reception des données
   // en fait l'objectif est de réceptionner que les variables d'entrées ou
   // d'entrée sortie
   if (permet_affichage )
     cout << "\n :  ouverture du tube en lecture, lecture sur le tube"
          << reception << endl;
    // le premier élément est le nombre de grandeurs  à lire
    // le second élément est le nombre de grandeurs que l'on attend en retour
    int nb_carac_a_lire = 2*4 ;
    int tub = open(reception.c_str(),O_RDONLY);
    char* tampon_reception =  &(t_car_x_n.tampon[0]);
    read (tub,tampon_reception,nb_carac_a_lire);
    if (permet_affichage )
        {cout << "\n lecture de " << t_car_x_n.n[0] << " reels ";};
    // puis lecture des variables de passage
    nb_variable=t_car_x_n.n[0];
    nb_carac_a_lire = t_car_x_n.n[0] * 8;
    tampon_reception =  &(t_car_x_n.tampon[8]);
    read (tub,tampon_reception,nb_carac_a_lire);
    close (tub); // fermeture du tampon
    if (permet_affichage )
        {if (permet_affichage > 3)
          {for (int i=1;i <= nb_variable; i++)
            cout << "\n variable "<< i<< "= " << t_car_x_n.x[i];
          };

         cout << "\n : fermeture du tube en lecture"
              << reception << endl;
        };
  }
 catch(...)
  {cout << "\n *** erreur en lecture des resultats de la fonction externe "
        << nom ;
   cout << flush;
   exit(1);
  };
};
```

TABLE 347 – Exemple d'une classe pour fonction externe : partie 4 .cc

```
// écriture des données Umat
void F_externe_nD::EcriturePipe()
{try
  {//   Creation d'un processus d'ecriture des variables
   // en fait l'objectif est d'ecrire que les variables qui évoluent
   if (permet_affichage > 2)
        cout << "\n : ouverture du tube en ecriture, ecriture  sur le tube "
             << envoi << endl;
   // le premier élément est le nombre de variables
   // le deuxième élément est le nombre de réel à écrire
   // on considère qu'il a déjà été défini lors du calcul de la fonction
   // ouverture du tube nomme en ecriture
   int tub = open(envoi.c_str(),O_WRONLY);
   int nb_carac_a_ecrire = nb_retour * 8 + 2*4 ;
   write (tub,t_car_x_n.tampon,nb_carac_a_ecrire);
   close (tub); // fermeture du tampon
   if (permet_affichage)
        {cout << "\n ecriture de " << nb_retour << " reels ";
         if (permet_affichage > 3)
          {for (int i=1;i <= nb_retour; i++)
            cout << "\n resultat "<< i<< "= " << t_car_x_n.x[i];
          };

         cout << "\n : fermeture du tube en ecriture"
              << envoi << endl;
        };
   }
  catch(...)
   {cout << "\n *** erreur en ecriture des parametre de la fonction externe "
         << nom ;
    cout << flush;
    exit(1);
   };
 };
```

82 Pondérations par fonctions

Les fonctions de pondérations peuvent apparaitre en particulier :

— dans les algorithmes,

— dans la définition de loi de comportement (ex : loi des mélanges...),

4 types de pondérations sont implantées dans Herezh. Suivant l'endroit où elles interviennent, on peut disposer soit de l'ensemble des versions soit uniquement d'une partie, il faut se référer à la documentation spécifique (cf. la liste précédente).

82.1 Pondération via une fonction du temps

C'est la pondération la plus simple. Elle nécessite la définition d'une simple fonction scalaire de type courbe 1D. On se reportera à (81.1) pour les choix disponibles.

82.2 Pondération via une ou plusieurs fonctions de grandeurs locales type ddl étendus

La pondération finale est obtenue par multiplication de fonctions scalaires de type courbe 1D (cf. 81.1). L'argument de chaque fonction scalaire peut-être une grandeur appelée dans Herezh++ `Ddl_enum_etendu` . Parmi l'ensemble des `Ddl_enum_etendu` manipulés par Herezh (il y en a 133 en V. 6.784) ceux qui sont actuellement accessibles comme argument sont :

1. la trace/3 du tenseur de déformation : " `Spherique_eps` " $= trace(\boldsymbol{\epsilon})/3.$)
2. l'intensité du déviateur de déformation : " `Q_eps` " $= \sqrt{\overline{\boldsymbol{\epsilon}} : \overline{\boldsymbol{\epsilon}}}$
3. le cosinus de trois fois l'angle de Lode (de phase) du déviateur de déformation : " `Cos3phi_eps` "
4. la déformation duale de Mises : " `def_duale_mises` " $= \sqrt{2/3 \times \overline{\boldsymbol{\epsilon}} : \overline{\boldsymbol{\epsilon}}}$
5. la déformation cumulée appelée certaine fois déformation équivalente : " `def_equivalente` " $= \int_0^t \sqrt{2./3. \times \overline{\boldsymbol{D}} : \overline{\boldsymbol{D}}}\, dt$
6. la vitesse de déformation équivalente : " `vitesse_def_equivalente` " $= \sqrt{2./3. \times \overline{\boldsymbol{D}} : \overline{\boldsymbol{D}}}$
7. la déformation au sens duale de Mises, maximale obtenue entre 0 et t : " `def_duale_mises_maxi` "
8. la trace/3 du tenseur de contrainte : " `Spherique_sig` " $= trace(\boldsymbol{\sigma})/3.$)
9. l'intensité du déviateur des contraintes : " `Q_eps` " $= \sqrt{\overline{\boldsymbol{\sigma}} : \overline{\boldsymbol{\sigma}}}$
10. le cosinus de trois fois l'angle de Lode (de phase) du déviateur des contraintes : " `Cos3phi_sig` "
11. la contrainte de Mises : " `contrainte_mises` "

On remarquera qu'il s'agit de grandeurs indépendantes du système de coordonnées.

82.3 Pondération via une fonction de grandeurs quelconques

Il s'agit ici d'une fonction nD (cf.81.2) de grandeurs quelconques (cf. 88.4). L'existence d'une grandeur quelconque est totalement dépendante du lieu d'appel.

82.4 Pondération via une fonction de grandeurs globales

Il s'agit ici d'une fonction nD (cf.81.2) de grandeurs globales (cf. 88.1). En général, la plupart des grandeurs globales sont disponibles néanmoins suivant les calculs effectués, certaines grandeurs peuvent ne pas exister.

Quinzième partie

Estimation d'erreur

83 Estimation d'erreur après calcul

Il est possible de calculer une répartition d'un estimateur d'erreur, fondé sur la méthode proposée par [O. C. Zienkiewicz, 1989]. Pour ce faire, après un calcul qui a permis de trouver une solution d'équilibre à un problème mécanique, le programme évalue un champ de contrainte C^0 le plus proche au sens des moindres carrés, de celui obtenu dans la résolution d'équilibre. Puis sur chaque élément, il évalue l'intégrale de la différence au carré, entre ces deux champs de contraintes. Cette différence constitue alors un estimateur de l'erreur due à la discrétisation géométrique. En fait de manière à obtenir une grandeur homogène à une différence de contrainte le programme calcule sur chaque élément :

$$\mathcal{F}_{\mathcal{V}_1} = \int_{\partial D_{V_e}} (\boldsymbol{\sigma}_{(v)} - \boldsymbol{\sigma}) : (\boldsymbol{\sigma}_{(v)} - \boldsymbol{\sigma}) dv_e \tag{128}$$

Puis la grandeur :

$$\delta\sigma_{moy} = \sqrt{\frac{\mathcal{F}_{\mathcal{V}_1}}{\int_{\partial D_{V_e}} dv_e}} \tag{129}$$

Dans ces expressions :

- $\boldsymbol{\sigma}$ représente le tenseur des contraintes issue du calcul initial d'équilibre mécanique,
- $\boldsymbol{\sigma}_{(v)}$ représente le nouveau champ de contraintes issue d'une approximation aux moindres carrés du champ initial $\boldsymbol{\sigma}$
- $\delta\sigma_{moy}$ correspond ainsi à une certaine moyenne de la différence qui existe entre les expressions continues et discontinues du tenseur des contraintes.

La table 348 donne un exemple de mise en place d'un calcul d'erreur.

- " `avec plus avec_remontee_et_calcul_d'erreur` " signifie qu'à la suite du calcul mécanique non dynamique, le programme effectue une "remontée" aux contraintes c'est-à-dire qu'il détermine le champ de contrainte continue entre les éléments, qui soit le plus proche possible au sens des moindres carrés, de la solution obtenue par éléments finis lors de la résolution de l'équilibre mécanique.
- " `plus visualisation` " permet d'obtenir après le calcul d'erreur, le menu interactif permettant de visualiser les nouvelles informations. En particulier il est possible de visualiser les contraintes aux noeuds. Là il ne s'agit pas des grandeurs aux points d'intégrations, transférées aux noeuds, mais directement des composantes absolues du champs de contraintes considérées comme degrés de libertés du problème de remontée aux contraintes.

Par exemple, dans le .CVisu de 349 on observe :

- `debut_tableau_ddl_aux_noeuds SIG11 SIG22 SIG12 fin_tableau_ddl_aux_noeuds` :
  cette ligne indique que l'on demande de tracer les isovaleurs des degrés de libertés SIGIJ définies aux noeuds, il s'agit des coordonnées absolues du tenseur de contraintes en 2D (ici en contrainte plane),

TABLE 348 – Exemple de déclaration permettant d'activer un calcul d'erreur

```
TYPE_DE_CALCUL
#-----------------------------------
# probleme d'equilibre non dynamique
#-----------------------------------
non_dynamique avec plus avec_remontee_et_calcul_d'erreur plus visualisation
```

— `debut_tableau_ddl_aux_elements ERREUR SIG11 SIG22 SIG12 fin_tableau_ddl_aux_elements`

: cette ligne indique que l'on demande de tracer les isovaleurs des contraintes calculées aux points d'intégrations (aux éléments) et transférées aux noeuds par une opération de moyenne. Il s'agit donc ici des contraintes déterminées par approximation aux moindres carrées des contraintes calculées initialement aux points d'intégrations.

TABLE 349 – Exemple de .CVisu contenant une visualisation d'erreur et de contraintes continues aux noeuds

```
#  ======================================================
#  ||       *****         demande d'une visualisation Gmsh:       *****           ||
#  ======================================================
....
#  ---------- definition des parametres pour les isovaleurs : -----
debut_isovaleur_Gmsh     # mot cle de debut des parametres pour les isovaleurs
....
 #   tableau des ddl  aux noeuds a visualiser, un par maillage
debut_tableau_ddl_aux_noeuds  SIG11  SIG22 SIG12  fin_tableau_ddl_aux_noeuds
#   tableau des choix_var aux noeuds a visualiser, un par maillage
#   choix_var (=1 ou 0)  indique si oui ou non il s'agit de la variation
debut_tableau_choix_var_ddl_aux_noeuds 1 1 1 fin_tableau_choix_var_ddl_aux_noeuds
....
 #   tableau de ddl aux elements a visualiser, un par maillage
debut_tableau_ddl_aux_elements  ERREUR SIG11 SIG22 SIG12  fin_tableau_ddl_aux_elements
....
fin_visualisation_Gmsh
#  ================================================
#  ||         fin de la  visualisation Gmsh                        ||
#  ================================================
```

Remarque : bien noter que le calcul et la sortie en post-traitement de la grandeur " ERREUR ", ne s'effectuent qu'après la fin du calcul mécanique. Si on utilise une sortie au fil du calcul, donc pendant le calcul, les sorties correspondantes à " ERREUR " seront nulles.

Il faut donc soit :

— en interactif, après le calcul, redemander une nouvelle sortie, typiquement : " visualisation automatique ? (rep 2)"

— soit en automatique, utiliser le mot clé : `lectureCommandesVisu` à la place de `visualisation` au niveau de la définition de l'algorithme dans le .info

84 Calcul d'intégrales de volumes

Au travers d'un calcul d'équilibre par exemple, il est possible de demander un calcul d'intégrale de volume d'une grandeur particulière.

Considérons par exemple un calcul qui a pour objectif de déterminer un champ de contrainte et de déformation sur un volume représentatif d'un matériau hétérogène pour ensuite en déduire une contrainte et une déformation moyenne via une moyenne volumique.

$$\boldsymbol{\sigma}_{moyen} = \frac{\int_{volume} \boldsymbol{\sigma} dv}{volume} \quad \text{et} \quad \boldsymbol{\varepsilon}_{moyen} = \frac{\int_{volume} \boldsymbol{\varepsilon} dv}{volume} \tag{130}$$

Herezh fournit la possibilité de calculer directement ces grandeurs pendant la résolution. Elles sont ensuite disponibles en post-traitement.

D'une manière générale, il est possible de demander deux types d'intégrales :

1. une intégrale sur un volume particulier délimité par une référence d'éléments.
2. le cumul en temps d'une intégrale sur un volume particulier délimité comme précédemment, par une référence d'éléments.

Deux choix de grandeurs sont proposés pour l'intégration :

1. soit un degré de liberté de base (cf. 80.11.4) ou étendu (cf. 88.3 pour la liste exhaustive), il s'agit ici de l'intégrale d'un scalaire.
2. soit une fonction nD (cf. 81.2) fonction de grandeurs globales et/ou de degré de liberté de base ou étendu. Il peut s'agir ici d'une intégrale complexe, dont le résultat est un scalaire ou un vecteur selon la définition de la fonction nD.

La mise en donnée de la demande du calcul d'une intégrale est précisée au chapitre (61).

Le programme calcule l'intégrale sur chaque élément qui constitue le volume d'intégration, puis effectue la somme sur l'ensemble des éléments du domaine demandé. Enfin le résultat est transféré au niveau des grandeurs globales et peut donc être utilisé (a priori) au même type que toutes les grandeurs globales, par exemple comme argument à d'autres fonctions nD.

À noter que pour un calcul incrémental explicite par exemple, à l'incrément n, on disposera comme grandeur globale, du résultat à l'incrément précédent. Si le calcul est itératif et incrémental, à l'itération n on disposera comme grandeur globale, du résultat de l'itération n-1 (itération précédente) ainsi que de la grandeur à l'incrément précédent.

En sortie de résultat, on disposera :

— des grandeurs globales à chaque incrément, disponible en sortie tableau (format maple)

— des intégrales sur chaque élément, disponible en isovaleurs ou en sortie tableau (format maple).

Seizième partie

Interruption et multistep

85 Gestion des interruptions prévues ou non

85.1 Interruptions prévues initialement

Il s'agit ici des interruptions initialement prévues avant le démarrage du programme.

Une fois une première lecture effectuée du fichier principal, c'est-à-dire d'une lecture de toutes les grandeurs jusqu'à la partie relative au mot clé " resultats ", il est possible d'indiquer le mot clé : " _pause_point_info_ " Dans ce cas, le programme s'arrête et un menu apparaît, permettant d'intervenir sur l'exécution future du programme. Les ordres alors disponibles sont les mêmes que ceux relatifs au cas d'une interruption non prévue initialement. Il faut se reporter à la table 350 pour plus d'information.

Exemple d'utilisation du mot clé " _pause_point_info_ " :

```
.....
      para_affichage #------------
#--------------------------
# PARAMETRE    |  VALEUR   |
#--------------------------
 FREQUENCE_SORTIE_FIL_DU_CALCUL 1

#  ------------------------------------------------------------------------------------
      resultats  pas_de_sortie_finale_
      COPIE 0
#

  _pause_point_info_
  _suite_point_info_

      controle #------------
#--------------------------
# PARAMETRE    |  VALEUR   |
#--------------------------
 SAUVEGARDE 1
 DELTAt 0.1 FORCE_DELTAT_DU_.INFO
 TEMPSFIN 0.5
   _fin_point_info_
```

85.2 Interruptions non prévues initialement

Cette partie concerne les différentes gestions d'interruptions, utilisées avec Herezh++ . Tout d'abord il existe classiquement les interruptions générées par le système, dues à la génération d'erreurs de calcul, type division par 0, racine carrée d'un nombre négatif ... , ou encore un accès à une zone mémoire protégée via un pointeur illicite. La plupart du temps, l'erreur est prise en compte par le programme, et en fonction des cas, un message "plus ou moins" explicite est produit. Néanmoins, ce message est plus explicite lorsque l'on utilise la version peu rapide d'Herezh++, dans laquelle de nombreux tests sont effectués. Avec la version rapide, les tests sont réduits au minimum. De plus, le niveau de commentaire demandé joue également.

Il existe un autre type d'interruption, c'est celles demandées par l'utilisateur en cours de calcul. À tout moment il est possible de générer une interruption via un ordre "kill" suivi d'un nombre qui indique le type d'interruption que l'on veut générer puis le numéro du processus auquel on veut transmettre l'interruption. Dans le cas d'Herezh actuellement (à partir de la version 6.676) l'interruption prise en compte est le type "15", ce qui a pour effet d'être immédiatement redirigé vers un menu particulier.

De manière pratique, on peut par exemple récupérer dans une fenêtre terminal, les processus qui fonctionnent via l'ordre unix : " ps - al ". Comme en général il y en a beaucoup, on peut filtrer pour récupérer uniquement les processus dont le nom contient "HZPP" via l'ordre : "ps -al — grep HZpp" ce qui permet d'obtenir le numéro du processus = PID. Une autre solution est de lancer le programme "top" qui indique dans le terminal, tous les processus en cours avec leur PID.

Ensuite pour transmettre l'interruption au programme on indique : "kill -15 <le numéro du PID> "

On obtient alors un menu dans lequel on trouve les options données dans la table 350.

TABLE 350 – Liste des options disponibles après une interruption via un contrôle c

Ordre	**Commentaire**	**ref**
(0 ou f) (fin execution)	demande explicite d'arrêt du programme	arrêt immédiat
(1 ou c) continue l'execution	demande explicite de continuer l'exécution	
(2) lecture de donnees secondaires dans le .info	permet de dire au programme de continuer la lecture dans le .info après sa modification	mise en place d'un drapeau interne (85.2)
(3) sortie de la recherche d'equilibre global	arrêt dès que possible des itérations et incréments d'équilibre	mise en place d'un drapeau interne
(4) sauvegarde de l'etat actuel dans le .BI	sauvegarde dès que possible de l'état actuel dans le .BI (n'est effectuée qu'après un équilibre validé)	mise en place e d'un drapeau interne
(5) sauvegarde post-traitement (CVisu) actuel	sortie dès que possible d'une visualisation (n'est effectuée qu'à la fin de l'itération en cours ou incrément)	mise en place d'un drapeau interne
(6) affiche nom grandeurs actuelles accessibles globalement	affichage de toutes les grandeurs globales actuellement disponible dans Herezh	
(7) modification d'une constante globale utilisateur	permet de modifier une constante utilisateur, l'effet est immédiat pour les fonctions nD utilisant ces constantes	voir (4) pour plus de détails sur les constantes utilisateurs

— Un premier exemple d'utilisation est la séquence suivante : après un "kill -15 <le numéro du PID> ", on modifie le fichier .info et ensuite on désire que le programme lise ces modifications. Ce qui se passe : le programme update le fichier .info qu'il est en train de lire, seules les données secondaires seront prises en compte dans la suite du calcul. Par exemple supposons qu'initialement, le fichier .info se termine de la façon suivante

```
.....
      controle #-----------
#--------------------------
# PARAMETRE    |  VALEUR   |
#--------------------------
 SAUVEGARDE 1
```

```
 DELTAt 0.1
 TEMPSFIN 0.2

#  ------------------------------------------------------------
      resultats  pas_de_sortie_finale_
      COPIE 0
                      _fin_point_info_
```

et que l'on génère une interruption pendant le calcul via "kill -15 <le numéro du PID> ", puis que l'on choisisse " (2) lecture de donnees secondaires dans le .info" puis on modifie le .info de la manière suivante :

```
.....
      controle #------------
#--------------------------
# PARAMETRE    |  VALEUR   |
#--------------------------
 SAUVEGARDE 1
 DELTAt 0.1
 TEMPSFIN 0.2

#  ------------------------------------------------------------
      resultats  pas_de_sortie_finale_
      COPIE 0
#
  _suite_point_info_

      controle #------------
#--------------------------
# PARAMETRE    |  VALEUR   |
#--------------------------
 SAUVEGARDE 1
 DELTAt 0.2 FORCE_DELTAT_DU_.INFO
 TEMPSFIN 0.5

         _fin_point_info_
```

Aussitôt que le programme a atteint le temps fin "t=0.2", il y a lecture des nouveaux paramètres après le mot clé (obligatoire) " _suite_point_info_ " et les nouveaux paramètres sont adoptés. Par exemple le programme redémarre le calcul jusqu'au nouveau temps fin "t=0.5".

— Un deuxième exemple d'utilisation concerne la modification d'une constante utilisateur via l'option (7) du menu. Après avoir modifié une constante, et avoir relancé le calcul via l'option (1) du menu, toutes les fonctions nD qui utilisent cette constante, vont utiliser sa nouvelle valeur. On peut ainsi piloter en cours de calcul, un changement de précision, un algorithme, une variation de chargement etc. ou tout simplement forcer une convergence pour récupérer l'état actuel du calcul.

86 Multistep

Il est possible pour certains algorithmes d'enchaîner plusieurs calculs (ou étapes ou step en anglais) dans un même fichier de commande .info. C'est en fait une utilisation particulière du mot clé " _suite_point_info_ ".

Dans l'exemple suivant, le calcul s'effectue en 3 step : le premier de 0.1 à 0.2s, le deuxième de 0.2s à 0.5s et enfin le troisième de 0.5s à 1.5s . Le calcul commence à 0.1s car l'utilisateur a indiqué un restart à l'incrément 1, qui correspond donc à la sauvegarde (d'un précédent calcul) au premier incrément donc à 0.1s

Au début du deuxième step, l'utilisateur impose un nouveau pas de temps, une nouvelle fréquence de sauvegarde, etc. On peut ainsi changer tous les paramètres de contrôle d'un step à l'autre. Par exemple, l'utilisateur ici change également un paramètre de contact : " PENETRATION_BORNE_REGULARISATION ".

Remarque importante : À chaque nouveau step, seuls les paramètres qui sont indiqués sont modifiés. Lorsqu'on commence un calcul avec un restart non nul (comme dans l'exemple) le numéro de restart va rester actif pour tous les steps. Ainsi dans l'exemple, supposons que l'on n'indique pas de nouvelle valeur de restart, la première valeur va être utilisée à chaque step. Le programme effectuera donc un nouveau restart à l'incrément 1 à chaque nouveau step, ce qui n'est pas l'objectif visé (en général). Pour éviter cela, il est donc nécessaire d'annuler le restart initial en indiquant "RESTART 0" comme cela est indiqué dans l'exemple. Dans ce cas seul le restart initial sera effectué une seule fois ce qui est en général l'objectif voulu.

```
.....
      controle #------------
#--------------------------
# PARAMETRE    |  VALEUR   |
#--------------------------
 SAUVEGARDE 1
 DELTAt 0.1
 TEMPSFIN 0.2
 RESTART  1

#  ------------------------------------------------------------
      resultats  pas_de_sortie_finale_
      COPIE 0
#

# $$$$$ step 2 $$$$$

_suite_point_info_

      controle #------------
#--------------------------
# PARAMETRE    |  VALEUR   |
#--------------------------
DELTAt 0.01
```

```
SAUVEGARDE 10
DELTAtMAXI  1.
TEMPSFIN 0.5
RESTART   0

para_contact
PENETRATION_BORNE_REGULARISATION 1e-9

# $$$$$ step 3 $$$$$

_suite_point_info_

      controle #------------
#--------------------------
# PARAMETRE    |  VALEUR   |
#--------------------------
DELTAt 0.1
TEMPSFIN 1.5.
RESTART   0

_fin_point_info_
```

Pour l'instant, ce fonctionnement multistep dans un même fichier .info , est opérationnel pour les algorithmes (cf 3) :

— équilibre quasi statique (non linéaire) : " `non_dynamique` "

— relaxation dynamique : " `dynamique_relaxation_dynam` "

Pour les autres algorithmes, si l'on veut enchaîner plusieurs step, il est toujours possible de faire une mise en données avec plusieurs .info, et à partir du deuxième .info, on effectue un restart sur le dernier incrément enregistré. Par rapport à la solution d'un seul fichier .info, on introduit un temps CPU supplémentaire nécessaire à la lecture/écriture de l'incrément de restart.

Dix-septième partie

Gestion des temps CPU

87 Gestion des temps CPU

87.1 Temps globaux

L'introduction des temps CPU a pour objectif de fournir des indicateurs quantitatifs sur certains gros blocs de calcul comme : la partie loi de comportement, ou la partie résolution du système linéaire ...

La collecte des temps CPU est systématique et est sauvegardée dans le .BI . Aussi lors d'un restart, le programme commence par récupérer les précédents temps de l'incrément de restart, puis les temps sont cumulés. A priori les temps sont sauvegardés en micro secondes, tandis que dans herezh les temps utilisés sont en nanoseconde.

À la fin du calcul, le bilan de tous les temps globaux est également sauvegardé dans un fichier spécifique dont le nom est construit à partir du nom du fichier .info, concaténé avec le suffixe : "_temps.cpu" . À noter que ce fichier n'est pas créé si le calcul n'a pas été mené à terme à cause d'une erreur de déroulement non gérée. Dans le fichier, la présentation des temps fait également apparaitre le pourcentage de chaque temps relativement au temps global.

Les temps disponibles en post-traitement, par exemple via le format maple, sont :

— "tpsU_InitAlgo" : temps consacré à l'initialisation de l'algorithme global,

— "tpsU_MiseAJourAlgo" : dans le cas où l'on change d'algorithme global, ce temps correspond à la mise à jour de l'initialisation lors du changement,

— "tpsU_CalEquilibre" : temps globalement consacré au calcul de l'équilibre, comprends en général tous les temps intermédiaires sauf la sauvegarde et les sorties au fil du calcul.

— "tpsU_MatSmLoc" : temps globalement consacré aux calculs des matrices locales et des vecteurs locaux correspondants aux puissances internes virtuelles c'est-à-dire en général aux forces internes généralisées, ceci dans le cas d'un calcul statique, transitoire ou dynamique implicite. Par contre ne contient pas les forces internes dans les cas d'algorithmes explicites ou de type équivalent.

— "tpsU_SmLoc" : idem le cas précédent, mais pour les algorithmes explicites ou de type équivalent.

— "tpsU_ResSystLineaire" : corresponds au cumul du temps consacré à la résolution du système linéaire global.

— "tpsU_CL" : temps consacré à l'application des conditions fixées sur les degrés de liberté : ex un déplacement imposé (conditions de Dirichlet).

— "tpsU_CLL" : temps consacré à l'application des conditions linéaires imposées sur les degrés de liberté. Ce temps ne comprend pas les répercution des CLL sur la résolution globale comme par exemple l'augmentation des largeurs de bandes

— "tpsU_chargement" : temps consacré à l'application des efforts ou flux (conditions de Neuman).

— 'tpsU_contactMatSmLoc' : temps relatif au calcul des forces et raideurs associées aux contacts, ceci dans le cas d'un calcul statique, transitoire ou dynamique implicite.

— "tpsU_contactSmLoc" : idem le cas précédent, mais pour les algorithmes explicites ou de type équivalent.

— "tpsU_rech_contact" : corresponds à la recherche et mise à jour des éléments de contact c'est-à-dire des surfaces de contact (ne comprend pas les temps liés aux forces internes et raideurs éventuellement associés).

— "tpsU_Metriques" : corresponds aux temps liés aux calculs géométriques nécessaires pour calculer les efforts généralisés internes et raideurs associées éventuelles. Comprends par exemple les calculs relatifs : au jacobien, à la déformation, la vitesse de déformation, les différents gradients, etc. À noter que ce temps est inclus dans le temps relatif à la loi de comportement.

— "tpsU_Lois_comp" : globalise tous les temps relatifs aux calculs de lois de comportement, inclus également "tpsU_Metriques".

— "tpsU_Sauvegarde" : correspond au temps de sauvegarde sur le .BI. À noter que le temps indiqué ne comprend pas la dernière sauvegarde, car il est mis à jour après la sauvegarde. Il y a donc toujours un décalage d'une sauvegarde.

— "tpsU_SortieFilCalcul" : globalise tous les temps des sorties au fil du calcul (ne comprends pas les sauvegardes sur le .BI). Comme pour le temps de sauvegarde, ce temps ne comprend pas la dernière sortie au fil du calcul, car il est mis à jour après la sortie.

Remarque Il est possible d'avoir accès directement à tous les temps via le fichier .BI, les temps sont indiqués au début de la sauvegarde de chaque incrément.

87.2 Temps locaux

Il est possible également de récupérer les répartitions spatiales (un temps par point d'intégration) des temps relatifs :

— au calcul de la loi de comportement,

— au calcul associé à la métrique. Comme pour les temps globaux, ce temps correspond aux calculs géométriques nécessaires pour calculer les efforts généralisés internes et raideurs associées éventuelles. Comprends par exemple les calculs relatifs : au jacobien, à la déformation, la vitesse de déformation, les différents gradients, etc. À noter que ce temps est inclus dans le temps relatif à la loi de comportement.

On peut par exemple visualiser cette répartition via des isovaleurs par exemple avec gmsh.

Dix-huitième partie

Grandeurs

88 Grandeurs manipulées

88.1 Grandeurs globales

Au cours de la mise en donnée, il est possible d'utiliser des grandeurs dites globales, pour alimenter par exemple des fonctions. Ces grandeurs sont mises à jour en fonction des calculs demandés. Dans certains cas elles peuvent soit ne pas avoir de sens (car elles ne sont pas mises à jour) soit éventuellement ne pas exister (car aucun objet n'a alimenté la grandeur). La pertinence des valeurs consultées est donc fonction des calculs effectués.

La liste des grandeurs globales est amenée à s'enrichir en fonction des évolutions d'Herezh.

Les grandeurs globales actuellement disponibles sont :

- les constantes introduites par l'utilisateur. Les 3 premiers caractères doivent obligatoirement être " `C__` "
- le temps courant, mot clé : " `temps_courant` "
- l'énergie cinétique du système, mot clé : " `energie_cinetique` "
- l'énergie interne du système, mot clé : " `energie_interne` "
- l'énergie générée par les forces externes, mot clé : " `energie_externe` "
- la puissance d'accélération du système, mot clé : " `puissance_acceleration` "
- la puissance interne du système, mot clé : " `puissance_interne` "
- la puissance des forces externes, mot clé : " `puissance_externe` "
- le maximum de puissance extérieure : `maxpuissext` . Par défaut il s'agit de la puissance virtuelle pour une vitesse de 1, ce qui conduit en fait aux forces extérieures $\vec{V}_{ext}$. Ainsi `maxpuissext` représente le maximum des forces extérieures en jeux pendant le calcul.
  `maxpuissext` $= ||\vec{V}_{ext}||_\infty$
  Ces forces correspondent aux forces imposées et elles incluent les forces de contact ! par contre elles n'incluent pas les réactions dues aux cinématiques imposées.
- le maximum de puissance intérieure : `maxpuissint` . Comme pour la puissance extérieure, il s'agit par défaut de la puissance virtuelle intérieure pour une vitesse de 1, ce qui conduit en fait aux forces généralisées intérieures $\vec{V}_{int}$. `maxpuissint` représente ainsi le maximum des forces généralisées intérieures, en jeux pendant le calcul.
  `maxpuissint` $= ||\vec{V}_{int}||_\infty$
  Les forces intérieures sont dues au comportement interne du matériau, elles sont donc directement pilotées par les lois de comportement utilisées dans le calcul.
- le maximum des réactions :
  `maxreaction` $= ||\vec{R}eac||_\infty$
  dues aux cinématiques imposées.
- le maximum du résidu global d'équilibre :
  `maxresiduglobal` $= ||\vec{R}||_\infty$

— le maximum de l'incrément de déplacement entre les instants "t" et "t+Δt" :
maxdeltax $= ||\Delta_t^{t+\Delta t}\vec{X}||_\infty$.
Cette grandeur est mise à jour à chaque itération et chaque incrément.

— le maximum de variation de l'incrément de déplacement entre 2 itérations en implicite et entre 2 incréments en explicite.
maxvardeltax $= ||\delta_i^{i+1}\left(\Delta_t^{t+\Delta t}\vec{X}\right)||_\infty$

— le maximum de l'incrément de degré de liberté entre les instants "t" et "t+Δt" :
maxvarddl $= ||\Delta_t^{t+\Delta t}d\vec{d}l||_\infty$.
Cette grandeur est mise à jour à chaque itération et chaque incrément. Ici il s'agit des ddl primaires correspondant à l'équation d'équilibre global et du type de problème.

- En dynamique explicite, il s'agit des accélérations entre "t" et "t+Δt",
- en dynamique implicite, il s'agit de la variation des accélérations entre l'itération "i" et "i+1",
- en mécanique statique et transitoire, il s'agit de la variation des déplacements entre l'itération "i" et "i+1",
- en thermique statique et transitoire il s'agit de la variation des températures entre l'itération "i" et "i+1".

— concernant les algorithmes

- la valeur de la norme utilisée pour la convergence globale, mot clé : " norme_de_convergence "
- le numéro des itérations d'équilibre en cours dans l'algorithme global, mot clé : " compteur_iteration_algo_global "
- le numéro d'incrément de chargement en cours dans l'algorithme global, mot clé : " compteur_increment_charge_algo_global "
- le numéro interne de définition de l'algorithme en cours, mot clé : algo_global_actuel
  On se reportera à (9) dans la partie "Remarques", pour plus d'information concernant la signification de ce numéro.

— concernant les lois de comportement :

- l'énergie élastique, mot clé : " energie_elastique "
- l'énergie plastique, mot clé : " energie_plastique "
- l'énergie visqueuse, mot clé : " energie_visqueuse "

— concernant les énergies "numériques :

- l'énergie visqueuse numérique, mot clé : " energie_visco_numerique "
- l'énergie du bulk viscosity, mot clé : " energie_bulk_viscosity "
- l'énergie d'hourglass, mot clé : " energie_hourglass "

— concernant les géométries :

- le volume total de matière : " volume_total_matiere "
- dans le cas de structures membranaires, le volume entre la membrane et le plan yz : " vol_total2D_avec_plan_yz "

— idem avec le plan xz : " vol_total2D_avec_plan_xz "

— idem avec le plan xy : " vol_total2D_avec_plan_xy "

Remarque importante Pour pouvoir utiliser les volumes entre la membrane et les plans il faut spécifier leurs calculs à l'aide du paramètre
" CAL_VOL_TOTAL_ENTRE_SURFACE_ET_PLANS_REF " cf. 79. Sinon, ces grandeurs globales ne sont pas alimentées.

88.2 Degrés de liberté : ddl

Cette série de grandeurs concerne les degrés de liberté primaires (Ddl) correspondant au problème traité : ex : positions ou déplacements dans un problème de mécanique. La liste s'étoffe à mesure de l'évolution des demandes. On reportera à 80.11.4 pour une description des ddl possibles.

88.3 Ddl_enum_etendu

Cette série de grandeurs concerne les degrés de liberté dits étendues (Ddl_enum_etendu). Ils correspondent en général à toutes les grandeurs classiques qui sont calculées à partir des ddl primaires. La liste s'étoffe à mesure de l'évolution des demandes.

La liste des Ddl_enum_etendu (version V. 6.896) :

Green-Lagrange11, Green-Lagrange22, Green-Lagrange33
Green-Lagrange12, Green-Lagrange13, Green-Lagrange23
Almansi11, Almansi22, Almansi33
Almansi12, Almansi13, Almansi23
Cauchy_local11, Cauchy_local22, Cauchy_local33
Cauchy_local12, Cauchy_local13, Cauchy_local23
Almansi_local11, Almansi_local22, Almansi_local33
Almansi_local12, Almansi_local13, Almansi_local23
Def_principaleI, Def_principaleII, Def_principaleIII
Sigma_principaleI, Sigma_principaleII, Sigma_principaleIII
contrainte_mises, contrainte_tresca, def_plastique_cumulee
Vit_def11, Vit_def22, Vit_def33
Vit_def12, Vit_def13, Vit_def23
Vit_principaleI, Vit_principaleII, Vit_principaleIII
Delta_def11, Delta_def22, Delta_def33
Delta_def12, Delta_def13, Delta_def23
logarithmique11, logarithmique22, logarithmique33
logarithmique12, logarithmique13, logarithmique23
Almansi_totale11, Almansi_totale22, Almansi_totale33
Almansi_totale12, Almansi_totale13, Almansi_totale23
Green_Lagrange_totale11, Green_Lagrange_totale22, Green_Lagrange_totale33
Green_Lagrange_totale12, Green_Lagrange_totale13, Green_Lagrange_totale23
logarithmique_totale11, logarithmique_totale22, logarithmique_totale33
logarithmique_totale12, logarithmique_totale13, logarithmique_totale23

energie_elastique, dissipation_plastique, dissipation_visqueuse
masse_relax_dyn
def_duale_mises
Spherique_eps, Q_eps, Cos3phi_eps
Spherique_sig, Q_sig, Cos3phi_sig
Spherique_Deps, Q_Deps, Cos3phi_Deps
def_equivalente , def_duale_mises_maxi, vitesse_def_equivalente
reaction_normale, reaction_tangentielle
force_gene_ext, force_gene_int, pression_ext
N_11, N_22, N_33, N_12, N_13, N_23
M_11, M_22, M_33, M_12, M_13, M_23
norme_gradT, norme_DgradT, norme_dens_flux
contact_actif
DeltagradT1, DeltagradT2, DeltagradT3
N_surf_1, N_surf_2, N_surf_3
N_surf_1_t, N_surf_2_t, N_surf_3_t
N_surf_1_t0, N_surf_2_t0, N_surf_3_t0
X1_t, X2_t, X3_t
X1_t0, X2_t0, X3_t0
Masse_diago_noeud, comp_tors_reaction

Quelques explications succinctes :

— Green-Lagrange : pour les composantes de la déformation mécanique (sans la thermique) de Green Lagrange en absolue

— Almansi : pour les composantes de la déformation mécanique (sans la thermique) d'Almansi en absolue

— Cauchy_local : pour les composantes des contraintes de Cauchy dans le repère local ! $(\vec{g}_i)$

— Almansi_local : pour les composantes des contraintes de Cauchy dans le repère local ! $(\vec{g}^i)$

— Def_principaleI, Def_principaleII, Def_principaleIII : les trois déformations principales, la première la plus grande, la dernière la plus petite

— Sigma_principaleI, Sigma_principaleII, Sigma_principaleIII : idem pour les contraintes de Cauchy

— contrainte_mises, contrainte_tresca, def_plastique_cumulee : correspond au nom

— Vit_def : pour les composantes de la vitesse de déformation en absolue

— Vit_principaleI, Vit_principaleII, Vit_principaleIII : les trois vitesses de déformations principales, la première la plus grande, la dernière la plus petite

— Delta_def : pour les composantes de l'accroissement de déformation en absolue

— logarithmique : pour les composantes de la déformation mécanique (sans la thermique) logarithmique en absolue

— Almansi_totale : pour les composantes de la déformation géométrique totale (avec la thermique) d'Almansi en absolue

— Green_Lagrange_totale : pour les composantes de la déformation géométrique totale (avec la thermique) de Green Lagrange en absolue

— logarithmique_totale : idem pour la déformation logarithmique

— `energie_elastique, dissipation_plastique, dissipation_visqueuse` : comme le nom l'indique

— `masse_relax_dyn` : valeur au noeud de la pseudo-masse utilisée pour la relaxation dynamique

— `def_duale_mises` : comme le nom l'indique

— `Spherique_eps, Q_eps, Cos3phi_eps, Spherique_sig, Q_sig, Cos3phi_sig Spherique_Deps, Q_Deps, Cos3phi_Deps , def_equivalente` et `def_duale_mises_maxi, vitesse_def_equivalente` : on se reportera à 80.11 pour une explication de ces grandeurs

— `reaction_normale, reaction_tangentielle` : comme le nom l'indique

— `force_gene_ext, force_gene_int` : correspond aux vecteurs "force généralisée externe et interne"

— `pression_ext` : comme le nom l'indique

— N_ij et M_ij : correspondent aux composantes de la résultante et du moment de la densité de torseur associé aux efforts généralisés dans le cas des coques et plaques

— `norme_gradT, norme_DgradT, norme_dens_flux` : comme le nom l'indique, sachant que "gradT" correspond au vecteur gradient thermique, et "flux" le flux associé

— `DeltagradT1, DeltagradT2, DeltagradT3` : les composantes du gradient thermique

— `contact_actif` : indique pour un noeud s'il est en contact ou non

— `N_surf_1, N_surf_2, N_surf_3` : composantes de la normale actuelle à une surface

— `N_surf_1_t, N_surf_2_t, N_surf_3_t` : composantes de la normale au temps t (= le dernier temps où le calcul a convergé), à une surface

— `N_surf_1_t0, N_surf_2_t0, N_surf_3_t0` : composantes de la normale initiale à la surface

— `X1_t, X2_t, X3_t` : composantes d'un point au temps t (= le dernier temps où le calcul a convergé)

— `X1_t0, X2_t0, X3_t0` : composantes initiales d'un point

— `Masse_diago_noeud` : la masse ponctuelle définie au noeud

— `comp_tors_reaction` : les composantes de torseur de réaction

88.4 Grandeurs dites quelconques

Ces grandeurs sont appelées "grandeurs quelconques" dans Herezh. La liste est grande et s'étoffe à mesure de l'évolution des demandes. Voici pour information, version V. 6.896, la liste enregistrée :

```
SIGMA_BARRE_BH_T
,CONTRAINTE_INDIVIDUELLE_A_CHAQUE_LOI_A_T
,CONTRAINTE_INDIVIDUELLE_A_CHAQUE_LOI_A_T_SANS_PROPORTION
,CONTRAINTE_COURANTE,DEFORMATION_COURANTE,VITESSE_DEFORMATION_COURANTE
,ALMANSI,GREEN_LAGRANGE,LOGARITHMIQUE,DELTA_DEF
,ALMANSI_TOTAL,GREEN_LAGRANGE_TOTAL,LOGARITHMIQUE_TOTALE
,DEF_PRINCIPALES,SIGMA_PRINCIPALES,VIT_PRINCIPALES
,DEF_DUALE_MISES,DEF_DUALE_MISES_MAXI
,CONTRAINTE_MISES,CONTRAINTE_TRESCA,ERREUR_Q,DEF_PLASTIQUE_CUMULEE
,ERREUR_SIG_RELATIVE
,TEMPERATURE_LOI_THERMO_PHYSIQUE, PRESSION_LOI_THERMO_PHYSIQUE
,TEMPERATURE_TRANSITION, VOLUME_SPECIFIQUE
,FLUXD, GRADT, DGRADT,DELTAGRADT
,COEFF_DILATATION_LINEAIRE, CONDUCTIVITE, CAPACITE_CALORIFIQUE
,MODULE_COMPRESSIBILITE,MODULE_CISAILLEMENT,COEFF_COMPRESSIBILITE
,MODULE_COMPRESSIBILITE_TOTAL,MODULE_CISAILLEMENT_TOTAL
,E_YOUNG,NU_YOUNG,MU_VISCO,MU_VISCO_SPHERIQUE
,NB_INVERSION,HYPER_CENTRE_HYSTERESIS,SIGMA_REF
,Q_SIG_HYST_Oi_A_R,Q_SIG_HYST_R_A_T
,Q_DELTA_SIG_HYST,COS_ALPHA_HYSTERESIS,COS3PHI_SIG_HYSTERESIS
,COS3PHI_DELTA_SIG_HYSTERESIS
,FCT_AIDE
,NB_ITER_TOTAL_RESIDU,NB_INCRE_TOTAL_RESIDU,NB_APPEL_FCT,NB_STEP,ERREUR_RK
,PRESSION_HYST_REF,PRESSION_HYST,PRESSION_HYST_REF_M1,PRESSION_HYST_T
,UN_DDL_ENUM_ETENDUE,ENERGIE_ELASTIQUE_INDIVIDUELLE_A_CHAQUE_LOI_A_T
,ENERGIE_PLASTIQUE_INDIVIDUELLE_A_CHAQUE_LOI_A_T
,ENERGIE_VISQUEUSE_INDIVIDUELLE_A_CHAQUE_LOI_A_T
,PROPORTION_LOI_MELANGE, FONC_PONDERATION
,POSITION_GEOMETRIQUE,POSITION_GEOMETRIQUE_t,POSITION_GEOMETRIQUE_t0
,CRISTALINITE
,VOLUME_ELEMENT,VOLUME_PTI,EPAISSEUR_MOY_INITIALE, EPAISSEUR_MOY_FINALE
,SECTION_MOY_INITIALE, SECTION_MOY_FINALE
,EPAISSEUR_INITIALE, EPAISSEUR_FINALE,SECTION_INITIALE, SECTION_FINALE
,VOL_ELEM_AVEC_PLAN_REF,INTEG_SUR_VOLUME,INTEG_SUR_VOLUME_ET_TEMPS
,ENERGIE_HOURGLASS,PUISSANCE_BULK,ENERGIE_BULK
,ENERGIE_STABMEMB_BIEL,FORCE_STABMEMB_BIEL
,TENSEUR_COURBURE, COURBURES_PRINCIPALES, DIRECTIONS_PRINC_COURBURE
,DIRECTIONS_PRINC_SIGMA,DIRECTIONS_PRINC_DEF,DIRECTIONS_PRINC_D
,REPERE_LOCAL_ORTHO, REPERE_LOCAL_H, REPERE_LOCAL_B
,REPERE_D_ANISOTROPIE,EPS_TRANSPORTEE_ANISO,SIGMA_DANS_ANISO
```

```
,PARA_ORTHO
,SPHERIQUE_EPS,Q_EPS,COS3PHI_EPS,SPHERIQUE_SIG,Q_SIG,COS3PHI_SIG
,SPHERIQUE_DEPS,V_vol,Q_DEPS,COS3PHI_DEPS,POTENTIEL
,DEF_EQUIVALENTE,DEF_EPAISSEUR,D_EPAISSEUR,DEF_LARGEUR,D_LARGEUR
 ,DEF_MECANIQUE,DEF_P_DANS_V_A
 ,SIG_EPAISSEUR,SIG_LARGEUR
 ,FORCE_GENE_EXT,FORCE_GENE_INT,FORCE_GENE_TOT,RESIDU_GLOBAL
 ,VECT_PRESSION,VECT_FORCE_VOLUM
 ,VECT_DIR_FIXE,VECT_SURF_SUIV,VECT_HYDRODYNA_Fn
 ,VECT_HYDRODYNA_Ft,VECT_HYDRODYNA_T
 ,VECT_LINE,VECT_LINE_SUIV,VECT_REAC,VECT_REAC_N
 ,NN_11,NN_22,NN_33,NN_12,NN_13,NN_23,MM_11,MM_22,MM_33,MM_12,MM_13,MM_23
 ,DIRECTION_PLI,DIRECTION_PLI_NORMEE,INDIC_CAL_PLIS
 ,NN_SURF,NN_SURF_t,NN_SURF_t0
 ,NOEUD_PROJECTILE_EN_CONTACT,NOEUD_FACETTE_EN_CONTACT
 ,GLISSEMENT_CONTACT,PENETRATION_CONTACT,FORCE_CONTACT,CONTACT_NB_PENET
 ,CONTACT_NB_DECOL,CONTACT_CAS_SOLIDE
 ,CONTACT_ENERG_PENAL
 ,CONTACT_ENERG_GLISSE_ELAS,CONTACT_ENERG_GLISSE_PLAS,CONTACT_ENERG_GLISSE_VISQ
 ,CONTACT_PENALISATION_N,CONTACT_PENALISATION_T
 ,NORMALE_CONTACT
 ,TEMPS_CPU_USER,TEMPS_CPU_LOI_COMP,TEMPS_CPU_METRIQUE
 ,GENERIQUE_UNE_GRANDEUR_GLOBALE,DEPLACEMENT, VITESSE
 ,DELTA_XI,MASSE_RELAX_DYN
 ,COMP_TORSEUR_REACTION
```

Ne pas oublier que la pertinence et l'existence de ces grandeurs dépendent du lieu où ces grandeurs sont demandées. En particulier, on remarquera que certaines grandeurs paraissent en double par rapport aux `Ddl_enum_etendu` . En fait, les grandeurs quelconques ont été créées pour la sortie des informations de préférence quelconque. Dans ce cadre, elles peuvent contenir un `Ddl_enum_etendu` , d'où deux identificateurs qui paraissent équivalents. Mais dans Herezh, ils n'apparaissent pas aux mêmes endroits ce qui justifie ce double référencement qui correspond à des conteneurs différents.

89 Grandeurs utilisées pour les pondérations

Il s'agit ici de lister ou rappeler, les différentes grandeurs utilisables avec les fonctions de pondérations. L'existence de chaque grandeur dépend du lieu où elle est demandée. Les listes qui suivent ont pour objectifs :

— d'indiquer la syntaxe de la grandeur,

— de montrer les grandeurs possibles.

Les grandeurs peuvent apparaitre en particulier :

— dans les algorithmes,

— dans la définition de loi de comportement (ex : loi des mélanges...),

— pour les chargements,

— pour les conditions limites cinématiques,

— dans les pilotages et les entrées/sorties.

De manière systématique voici les différentes grandeurs qui peuvent apparaître.

1. La grandeur la plus simple est le temps. Sans précision supplémentaire, c'est la grandeur par défaut qui alimente les pondérations. Dans le cas où il est nécessaire de préciser la grandeur temps, le mot clé est : "TEMPS".
2. La deuxième série de grandeurs concerne les degrés de liberté primaires (Ddl) ou étendus (`Ddl_enum_etendu`). Les premiers correspondent aux ddl primaires du problème traité : ex : positions ou déplacements dans un problème de mécanique, les seconds correspondent en général à toutes les grandeurs classiques qui sont calculées à partir des ddl primaires. La liste s'étoffe à mesure de l'évolution des demandes. On reportera à 80.11.4 pour une description des ddl possibles, et à 88.3 pour la liste des ddl étendus.
3. La troisième série de grandeurs concerne les grandeurs appelées "quelconques" dans Herezh. On se reportera à 88.4 pour une information détaillée.
4. La 4ième série de grandeurs concerne les grandeurs globales. On se réfèrera à 88.1 pour une description des grandeurs globales possibles.

Dix-neuvième partie

Chronologie et historique des mises à jour

90 Introduction

Il s'agit ici de retracer l'historique des mises à jour, vues du coté de l'utilisateur. Cette partie débute en 2004, alors que le projet existe alors depuis 6 ans, ce qui explique que l'on démarre à la version 5.

91 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'un 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 hytérésis avec abaqus et avec Herez++. 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.

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.

Versions 5.87 26 septembre 2005 : Mise en place d'une thermo-dépendance possible des paramètres des lois de Mooney-Rivlin 1D et 3D.

Version 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.

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.

Version 5.92 Correction des principaux bugs sur la loi de Mooney-Rivlin 3D y compris la version Umat.

Version 5.93 Mise en place du calcul des puissances réelles en sortie globale.

Version 5.94 13 octobre 2005, modification des sorties : pour Gid, la sortie du maillage initiale devient implicite, pour les numéros d'incréments 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.

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.

Version 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.

Version 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 !).

Version 5.99 15 janvier 2006. Mise en place d'éléments axisymétriques quadrangulaires : linéaires, quadratiques incomplets, quadratiques complets.

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.

Version 5.993 28 janvier 2006. Correction d'un bug sur les sorties d'énergies élastiques internes en explicite.

Version 5.994 31 janvier 2006. Introduction de loi 1D 2D et 3D qui ne font rien mécaniquement ! .

Version 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.

Version 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.

Version 6.00 22 février 2006. Debuggage de la version 1D hystérétis 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 ! !.

Version 6.01 16 mars 2006. Debuggage relaxation dynamique et hystérésis. Informations sur la sortie des grandeurs tensorielles en format maple.

Version 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.

Version 6.03 Première mise en place du calcul des énergies élastiques et plastiques en élasto-hystérésis 3D.

Version 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).

Version 6.032 Correction de bug en lecture de données lorsque l'on passe d'un calcul implicite à un calcul explicite.

Version 6.033 26 mars 2006 : Correction de bug sur le calcul de charge volumique en 3D.

Version 6.034 2 avril 2006 : Correction de bug sur la loi visco-élastique de maxwell 3D.

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.

Version 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).

Version 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.

Version 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.

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.

Version 6.066 4 juin 2006 : Correction d'un Bug sur le fonctionnement de la relaxation dynamique en implicite.

Version 6.067 5 juin 2006 : Correction d'un Bug sur le fonctionnement du comportement des forces hydrodynamique en 3D.

Version 6.07 27 juin 2006 : Mise en place d'un algorithme global de type Runge-Kutta pour résoudre l'équilibre dynamique de la structure.

Version 6.08 10 aout 2006 : Première mise en place d'un algorithme de Galerkin discontinu : celui de Bonelli.

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.

Version 6.083 1 sept 2006 : Introduction de la possibilité d'utiliser le pas de temps critique de la méthode DFC pour paramètrer les pas de temps courant, mini et maxi.

Version 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.

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.

Version 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.

Version 6.090 8 oct 2006 : Modification et amélioration du contrôle d'avancement temporel pour l'algorithme de Runge-Kutta global (équilibre de la structure).

Version 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 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 suiveuses 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 définit sur un seul intervale. 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 SFE1 sur petit, moyen et grand maillage. Modification du post-traitement pour l'adapter aux sfe1. Pour l'instant uniquement la déformée est opérationnelle. Mise à jour de la doc d'utilisation, au niveau des sfe1 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 arrêtes. 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 theta1 et theta2.

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 $\dot{w}'$ 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 cristalinité 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 cristalinité 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 (cristalinité 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 cristalinité.

139 **Version 6.419** 7 avril 2008 : Mise en place des courbes : COURBE_TRIPODECOS3PHI COURBE_SIXPODECOS3PHI COURBE_EXPO_N 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 dévaut.

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 mu1 dans la loi de orgéas1.

152 **Version 6.450** 25 janvier 2009 : fin intégration de la dépendance à la phase de mu1 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 sfe.

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 !

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 débug. Introduction de la possibilité d'un critère d'arrêt uniquement sur le résidu hors viscosité numérique.

Version 6.540-43 20 septembre 2011 : Correction du petits bugs, dont un sur l'utilisation de matrice masse consistante.

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.

Version 6.545 28 septembre 2011 : Correction d'un bug sur le contact avec présence d'interpolation quadratique.

Version 6.546 10 octobre 2011. Introduction du contact entre membrane et membrane, puis première implantation de l'auto-contact.

Version 6.547-8 17 octobre 2011. Correction de deux bugs.

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 !

Version 6.566 15 mars 2012 : Introduction des déformations planes utilisable pour toutes les lois 3D implantés.

Version 6.576 25 avril 2012 : Correction de bugs. Ajout de la possibilité d'utiliser des matrices bandes non-symétriques, de la bibliothéque Lapack (avec accélération sur mac)

Version 6.585 25 septembre 2012 : Mise en place d'un algorithme d'orientation automatique pour un maillage de membrane-plaque-coque.

Version 6.586 9 octobre 2012 : Mise en place d'une méthode pour fusionner des noeuds très voisins.

Version 6.587-9 6 novembre 2012 : Correction de bug.

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.

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.

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.

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.

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.

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.

Version 6.602 23 janvier 2013 : Correction d'un bug sur les sorties .maple.

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.

Version 6.604 2 février 2013 : rep Bug sur le calcul de la def equi, lorsque très petite. Mise en place du calcul des énergies pour maxwell 2D contraintes planes.

Version 6.605 7 février 2013 : Introduction d'un facteur de régularisation dans la loi de Newton non linéaire : 1D, 2D_D et 3D.

Version 6.606-6.609 16 mars 2013 : Amélioration de la loi d'élastohystérésis 3D, début de l'introduction de la thermique, correction de bugs, introduction d'une nouvelle technique de blocage des modes d'hourglass.

Version 6.610-6.611 20 mars 2013 : correction de bugs, et introduction d'un élément pentaèdre quadratique incomplet à 15 noeuds et 3 pti.

Version 6.612-6.613 24 mars 2013 : fin d'une première mouture permettant de créer de manière interactive, un fichier .info complet. Ajout dans les conditions de chargement PHYDRO, de la possibilité d'avoir un calcul de pression quelque soit la position du point.

Version 6.614-6.623 8 juillet 2013 : correction de bugs, première mise en place de la gestion d'interruption. Mise en place d'un comportement de Bulk dépendant de la variation de volume.

Version 6.623-6.635 14 novembre 2013 : correction de bugs. Introduction d'éléments tétraédriques quadratiques complets sous-intégrés.

Version 6.636-6.642 5 décembre 2013 : correction de bugs sur le contact. Introduction d'un pilotage des charges hydro via des fonctions pour chaque composante. Introduction d'éléments SFE1 à 5 points d'intégration dans l'épaisseur.

Version 6.643-6.646 11 février 2014 : correction de bugs, amélioration du contact.

Version 6.647-6.649 14 avril 2014 : correction de bugs, en particulier sur l'hystérésis : algo de basculement RG sur linéarisation .

Version 6.650 14 avril 2014 : Introduction d'un nouveau type de courbe : type poly-Hermite .

Version 6.651-53 6 mai 2014 : Introduction d'un nouveau potentiel hyperélastique "ISOHYPERBULK_GENE".

Version 6.654-72 2 octobre 2014 : intro mat lapack bande symétrique et non symétrique, intro mat lapack carré symétrique, correction de bugs et amélioration sur : les conditions limites linéaires, la prise en compte simultané de plusieurs maillages, l'optimisation de la numérotation, la loi d'hystérésis.

Version 6.673-78 27 novembre 2014 : intro de l'estimation d'erreur, commandes interactives pour la construction des sous types de calcul, prise en compte de la lecture sur des fichiers externes de contraintes aux pti et de déplacements aux noeuds, mise à jour doc et commandes interactives de définition.

Version 6.679-85 6 janvier 2015 : gestion exacte des fins de chargement, correction bug hystérésis, intégration de la régularisation dans l'hystérésis pour certains paramètres (se référer à la doc dans l'exécutable pour plus d'info), integ prec relative dans newton et application hystérésis, cor bug numérotation d'increments, modif cosmétique sur l'affichage temps fin, test arrêt et sauvegarde combinée.

Version 6.685-88 2 février 2015 : Intro des volumes au pti, intro contact elem axi avec possibilité d'utiliser pour le contact un coef calculé automatiquement comme pour les éléments volumiques classiques.

Version 6.689-91 10 février 2015 : correction bug sur l'avancement temporel, modif para contrôle hystérésis (cas phi négatif), correction bug sur passage quadratique incomplet en quadratique complet pour des pentaèdres.

Version 6.692-6.703 30 avril 2015 : Corrections et améliorations diverses. Intro test inclusion sur hysteresis, intro fct aide en 3D hyste + visu, intro loi Maheo_hyper, cor bug sur calcul epaisseur quadrangles, triangles, mise à jour Maxwell 2D_D, 2C_C, 3D, 3D-¿2D_C3, intro contraintes planes pour hyper_grenoble, et tests en 2D_CP pour l'hystérésis, amélioration de la convergence en CP. Amélioration de l'hystérésis 3D, ajout para contact dans -n. Chgt indic visu hyster et CP. Cor bug Hyster, amélioration CP en explicite, idem résol globale. Intro pour le post traitement des grandeurs polaires pour les lois hyper grenobloise. Ajout dsig_deps3D en curviligne pour les lois de types Mooney-Rivlin d'où possibilité d'utiliser ces lois en CP.

Version 6.704-6.712 11 juillet 2015 : intro d'un nouvel algo d'orientation de surface. Intro d'une loi critère sur les plis dans les membranes. Introduction en statique transitoire d'un algo Newton modifié, et/ou raideur moyenne. Changement de la prise en compte du calcul des épaisseurs avec les éléments SFE. Ajout de la loi CP double avec validations simples.

232 **Version 6.713-6.722** Intégration de la possibilité d'utiliser successivement plusieurs types de matrices et de résolution, dans le cas où l'un ou l'une bug. Introduction d'une loi d'hystérésis sur la variation de volume. Introduction de sortie systématique de grandeurs de travail, pour les lois hyperélastiques 3D. Par exemple, il est possible en post-traitement, d'avoir accès à l'intensité du potentiel, sous forme de valeur aux pti, ou sous forme d'isovaleurs.Correction d'un bug sur le chaînage de tenseur. Correction d'un bug sur le Runge Kutta. Intro de la variable MAX_ESSAI_INCRE au niveau des paramètres globaux, correction de bug sur la gestion des temps fins, correction d'un bug sur la loi Maheo_Hyper

233 **version 6.724** Concernant l'hystérésis 3D : modification des différents algorithmes de gestion des coïncidences et inversion, correction de bugs.

234 **Version 6.725** Lorsqu'il y avait une erreur au niveau de la loi de comportement, cela générait un affichage de l'erreur. Cette information est maintenant complètée par l'affichage du numéro de l'élément et du point d'intégration où est calculée la loi.

235 **version 6.726** Correction d'un bug sur l'énuméré des coque-poutre et plaque qui faisait que les membranes passaient certaines fois sous les procédures des coques, poutres et plaques, dans le cas des sorties en post-traitement. Optimisation des méthodes d'initialisation du post-traitement, appelées lors d'une sortie au fil du calcul d'où un gain de temps sur les sorties au fil du calcul. Introduction de nouvelles grandeurs en sortie concernant le contact : possibilité de visualiser les noeuds en contact, les éléments en contact, les forces de contact aux noeuds esclaves, les forces aux noeuds des facettes maîtres, etc. (cf. les différentes possibilités offertes en interactif après une étude avec contact). Au niveau des mouvements solides initiaux, possibilités d'effectuer avant le calcul, une opération d'homothétie, avec éventuellement des rapports différents suivant x, y et z.

236 **version 6.727-8** Amélioration de la lecture : normalement maintenant on peut avoir un fichier qui s'arrête à la fin d'une ligne de donnée sans ligne vide qui suit. Sortie dans le .reac final de l'ensemble des réactions des ddl bloqués, sous forme de vecteur (et non de composantes) avec en commentaire une référence associée qui globalise tous les noeuds bloqués. Mise en place de la possibilité d'introduire des chargements sous forme de champ de valeurs scalaires ou vectorielles (suivant les types de chargement).

237 **version 6.729** Modification du contact :1)mise à jour de la documentation au niveau des paramètres de contact, 2) mise en place d'un cas 4 (TYPE_PENALISATION_PENETRATION 4) qui permet d'ajuster la pénalisation pour satisfaire une pénétration maxi fixée. 3) petites améliorations du fonctionnement des différents algos au niveau du calcul automatique du facteur de pénalisation en fonction des raideurs matériaux.

238 **version 6.730** Correction d'un bug sur le post-traitement de calcul d'erreur et de remonté au contrainte dans le cas d'un maillage quadrangle.

239 **version 6.732** Mise en place de la possibilité d'utiliser plusieurs matrices de raideurs dans le cas d'un calcul dynamique implicite (comme c'était déjà le cas en statique). L'objectif visé est d'utiliser un type de matrice principal qui est performant. Si jamais la résolution du système linéaire global échoue, Herezh switch alors sur un type de matrice secondaire éventuellement moins performant mais par exemple plus robuste. Le switch s'effectue sans recalculer les raideurs locales ce qui fait que l'impact en temps de calcul est minimisé.

240 **version 6.733**Amélioration de la prise en compte des conditions linéaires lorsque celles-ci concernent soit les ddl d'un noeud, soit le contact solide-déformable par pénalisation. En particulier cela concerne les conditions linéaires de type "Déplacement ou positionnement dans un plan (3D) ou sur une droite (2D)". Les plans et droites pouvant par ailleurs évoluer pendant le calcul. exemple de gain de calcul : ancienne méthode sur un maillage triangles de 346 noeuds, pour un calcul d'équilibre (loi élastique sur membrane) en relaxation dynamique avec 146 conditions linéaires : $->$ environ 44 minutes sur ma machine (2 incréments, 22000 itérations) nouvelle méthode : $->$ environ 5 minutes sur ma machine (2 incréments, 22000 itérations) a priori avec les mêmes résultats ... mais reste à confirmer.

241 **version 6.734-36** Introduction de la mesure de certains temps cpu de calcul. Un point important est qu'à partir de cette version il est nécessaire d'avoir installé la bibliothèque boost qui est disponible sur toutes les plateformes : linux, window et osx Dans tous les cas il est préférable d'installer "toute la bibliothèque" car dans les futures versions, l'utilisation de nouveaux éléments de la bibliothèque est prévue en particulier pour le multi-streads Correction d'un bug sur le contact en axisymétrique. La surface associée au calcul du facteur de pénalisation était erronée. Correction d'un bug sur la sortie des forces : en axi, seules les directions x et y ont des composantes non nulles Mise à jour de la documentation.

242 **version 6.737** Introduction de la sortie systématique des temps cpu dans un fichier ¡nom du .info¿ _temps.cpu.

243 **version 6.738-44** - Corr bug sortie en fichier des temps cpu. - Modification de la sortie des incréments (sortie au fil du calcul). Pour les grandeurs autres que celles calculées aux points d'intégration, il y avait une sortie pour l'incrément 0. Cette sortie n'était pas cohérente avec les autres, elle a donc été supprimée. Ceci étant, lorsque l'on veut avoir l'état des grandeurs pour t=0 la solution est d'effectuer un premier incrément de temps avec un chargement nul. - Introduction de la possibilité de créer interactivement une condition linéaire de type projection sur plan ou droite, ou de type général. Cette possibilité était jusqu'à maintenant absente dans le menu de création interactif d'un nouveau fichier de commande .info - Versions associées à une gestion complète des dépendances entre les sources ¿ refonte complète des makefiles. Devrait (?) permettre par la suite de générer rapidement les nouvelles versions Linux (au même titre que celles du système osX) correction d'un petit bug sur la génération interactive (via la création d'un nouveau .info) de l'entête correspondant à l'algorithme "information", nécessaire par exemple pour générer de nouvelles références. - Modification du code pour la sortie interactive de référence : une étonnante différence de fonctionnement entre osX et linux ? ? - amélioration de l'algorithme de contact pour le type 4 : le facteur multiplicatif (qui dépend de la pénétration maxi imposée) est lissé au cours des incréments, sous forme d'une moyenne pondérée sur 2 pas -¿ meilleure stabilité - Dans le cas de l'utilisation d'une UMAT via un dialogue entre Herezh et Herezh ou entre Herezh et Abaqus il faut que le nom de la loi de comportement soit cohérente entre les deux programmes. La version 6.744 intègre un message d'erreur dans le cas où les noms ne sont pas identiques.

244 **version 6.745-58** - dans le cas de l'utilisation d'une loi externe à Herezh via une Umat (cf. doc), extension du cas 3D actuel à la prise en compte du cas des calculs en axisymétrique. L'extension est également valable lorsque Herezh sert pour calculer la loi de comportement en tant qu'Umat extérieur. En fait, dans le cas d'un dialogue Herezh–Herezh, il n'y avait pas de pb, par contre dans le cas d'un dialogue Herezh–Abaqus, il y avait un pb à cause du fait qu'Abaqus ne fournit que 4 composantes des tenseurs (les composantes non nulles). La version est pour l'instant en test. Le programme c (ou c++) permettant de relier Herezh avec un programme externe et l'exemple d'une liaison avec Abaqus est également accessible sur le site. - passage à l'utilisation de la librairie boost version 1.58

245 **version 6.746** ajouts dans les utilitaires Herezh, de la possibilité de supprimer des éléments à 2 noeuds très proches : au lieu de supprimer les noeuds dans les références, les anciens numéros sont remplacés par les numéros des noeuds restants correspondants amélioration de la mise à jour des références de noeuds après le soudage de noeuds très proches - amélioration de la mise à jour des références d'éléments après la suppression d'éléments superposés : au lieu de supprimer les éléments dans les références, les anciens numéros sont remplacés par les numéros des noeuds restants correspondants - intégration de la température dans les lois de type UMAT, en interne et en externe.

246 **version 6.747** - suite à la demande #98, modification de la signification (et du fonctionnement interne) de la variable température passée en paramètre. - suite à la demande #94, modification de la signification (et du fonctionnement interne) des variables temps - mise à jour de la documentation au niveau des chargements : on précise que l'on peut utiliser en option une courbe de charge + échelle + temps mini et maxi

247 **version 6.748** - ajout de la possibilité de générer automatiquement à la lecture des maillages d'une référence sur les noeuds non référencés par des éléments. cf. la documentation chap : 2.1 - correction bugs sur la prise en compte de la viscosité pour les lois Umat (dépendance au temps)

248 **version 6.749** - correction de bugs et amélioration de la loi de comportement HYSTERESIS_BULK ... en espérant que tout est ok maintenant !

249 **version 6.750** - introduction d'une nouvelle classe de fonctions 1D quelconques qui peuvent être définies par une expression littérale. L'implantation utilise la bibliothèque muParser. La mise à jour de la documentation présente les différentes possibilités offertes dans l'écriture des expressions littérales.

250 **version 6.751** - introduction de la résolution d'un problème thermique via des éléments spécifiques thermiques qui ont pour vocation de pouvoir être couplés faiblement et fortement avec des éléments mécaniques. L'ensemble des classes génériques sont implantées et un premier élément "biellette thermique" fonctionne pour un cas simple de condition de température bloquée en statique. La

partie post-traitement fonctionne également, en format maple et en post traitement graphique : grandeurs disponibles pour l'instant : température, gradient de température, flux thermique, vitesse de gradient thermique. bon... maintenant il faut étoffer ! La doc sera mise à jour dès que possible.

251 **version 6.752** - introduction d'un chargement volumique dit "pseudo-massique" correspondant en fait à un chargement volumique relativement au volume initial.

252 **version 6.753-54** - modification et amélioration de la sortie des réactions de contact (mais le travail n'est pas terminé) - introduction du critère plis sur les biellettes. Ce critère vient compléter ce qui existe déjà avec des membranes ce qui permet de coupler des maillages de membranes et de biellettes tout en garantissant que chaque élément satisfera aux critères plis respectifs.

253 **version 6.755-58** Introduction du pilotage des critères. Introduction de variables globales accessibles de partout, avec un cycle de temps d'accès et/modification rapide. Première utilisation des fonctions multidimensionnelles. Qq modifs sur le calcul des masses virtuelles en relaxation dynamique.

254 **version 6.759** Ajout dans les grandeurs globales consultables de partout de : concernant les lois de comportement : l'énergie élastique, l'énergie plastique, l'énergie visqueuse, concernant les énergies "numériques" : l'énergie visqueuse numérique, l'énergie du bulk viscosity, l'énergie d'hourglass.

255 **version 6.760-66** Correction d'un bug relatif à l'utilisation d'une loi additive à l'intérieur d'une loi critère. Correction d'un bug sur le pilotage de la loi critère. Correction d'un bug sur le calcul de valeurs propres sur les tenseurs d'ordre 2. Introduction des variables globales volume matière et volumes avec des plans. Pour l'instant ce n'est pas testé. Introduction du critère mixte : relaxation dynamique avec amortissement cinétique puis visqueux (type 4). Fonctionnement par défaut via la précision $->$ testé ok, $->$ mise à jour de la documentation. Introduction pour le type 4 de la relaxation dynamique, de l'utilisation d'un pilotage avec une fonction dépendant du temps et/ou une fonction nD dépendant des variables globales. Implantation terminée. Pour l'instant ce n'est pas testé et il faudra mettre à jour la doc. Introduction pour tous les types de chargement classiques de l'utilisation d'une fonction nD. Implantation terminée. Pour l'instant ce n'est pas testé et il faudra mettre à jour la doc. Correction de bugs : loi additive, loi contrainte plane. Inversion de l'orientation de l'effort produit par un chargement hydrostatique. Introduction dans les lois ISO_ELAS_SE1D et ISO_ELAS_ESPO1D du calcul du module de compressibilité et de cisaillement, utilisées en particulier pour la mise à jour éventuelle des sections.

256 **version 6.767-74** Intégration de la méthode "Calcul_dsigma_deps" pour la loi Iso_elas_expo3D ce qui permet d'utiliser les routines CP et CP2. Correction d'un bug sur la sortie des grandeurs à la suite de l'utilisation de suite_point_info. Correction d'un bug sur la sortie de la déformation d'épaisseur pour les lois contraintes planes de doublement plane, il manquait une sortie. Cependant ce qui était sortie en non nul, était correct. Amélioration du restart en particulier pour éléments 1D et 2D. Intégration du re-calcul de la métrique à l'instant t. Cela impact le cas des variations d'épaisseur pour les éléments 2D et les variations de section pour les éléments 1D. Introduction de l'opérateur tangent d_sig/d_eps pour l'hypo-élasticité 3D. Mise à jour de la documentation théorique associée. Modification du test de validation en conséquence : on obtient maintenant des résultats quasi identiques au cas 2D contraintes planes lorsque l'on utilise la loi 3D. La différence résiduelle est a priori du au fait que le calcul de l'épaisseur est différent suivant que l'on utilise une loi de contrainte plane hypo ou que l'on utilise une loi 3D avec un critère de contrainte plane (cf. doc théorique sur le calcul de l'épaisseur). Loi additive : introduction d'une fonction nD de pondération fonction de grandeurs locales à une des lois membres, correction bug de restart lorsque celui-ci commençait à la fin d'un .BI. Bug étrange mais une solution a été trouvée ; correction d'un bug d'initialisation de la déformation thermique sur loi additive + loi des mélanges. Mis à jour de la documentation pour le comportement d'hystérésis bulk et pour l'utilisation d'une pondération nD de grandeurs locales avec la loi additive.

257 **version 6.775-78** Correction bug sur matrices secondaires avec contact : les matrices peuvent changer de dimensions, correction bug init def thermique sur toutes les lois combinées : l'ordre d'initialisation était erronée, amélioration de l'option suite_point_info_ pour 2 algos : relaxation dynamique et statique. Il sera possible de modifier également tous les autres algos, mais j'attends qu'il y ait des demandes !, correction d'un bug sur les éléments points pour Umat, correction d'un bug sur le contact axisymétrique, amélioration du restart dans le cas de contact et de l'utilisation du mot clé _suite_point_info_, mise à jour de la doc avec introduction du chapitre Multistep.

258 **version 6.779-84** Cor bug contact 1D, amélioration des éléments axi, du contact en axi(cor bug), restart en axi, amélioration du contact axi avec triangles esclaves quadratiques, intro d'une nouvelle répartition de pti pour trai axi quadracomp, chgt des messages warning et erreur, intro loi des mélanges avec pilotage via des grandeurs globales, intro d'une nouvelle mise en données et de nouvelles fonctionnalités (voir doc), cor bug lors du passage d'incrément sur la loi CP2, introduction de bornes mini et maxi pour les variations de h/h0 et de b/b0, ceci pour les lois CP et CP2, prise en compte des conditions limites cinématiques suivant l'axe 3, par neutralisation, en axisymétrique et en dynamique. Mise à jour de la doc pour la loi des mélanges, "Les grandeurs", "Pondérations par fonctions".

259 **version 6.785** Cor bug sur plusieurs bornes TEMPS_MIN= ET TEMPS_MAX= qui se suivent, dans les conditions limites.. Introduction des fonctions nD "FONCTION_COURBE1D" et "FONC_SCAL_COMBINEES_ND". Dans les fonctions nD, amélioration de la prise en compte mixte des variables globales et locales. Amélioration de l'utilitaire de création d'un .info : validation par retour chariot des paramètres par défaut à définir avant l'arrivé au menu principal. Ajout de la variable globale "temps_courant".

260 **versions 6.786**

- validation avec correction de bug du fonctionnement des fonctions combinées nD
- correction d'un bug sur les chargement avec fonction nD sans courbe de charge.
- correction d'un bug sur les sorties .maple : dans le cas où une variable demandée n'existait pas, Herezh++ dans la version fast, sortait une valeur incontrôlée contrairement à la version lente ou c'était ok.

261 **versions 6.787**

- correction de plusieurs bugs sur la sortie des directions principales (ticket #128)
- mise à jour doc a propos du mot clé `les_fonctions_nD` qui remplace les_Fonctions_nD dans le .info (ticket #129)

262 **versions 6.788-91**

- fin cor bug sur la sortie des directions principales
- introduction d'une nouvelle mise à jour et calcul du facteur lambda dans l'algorithme de relaxation dynamique
- introduction de la vérification automatique des noeuds non référencés par des éléments. Le fait d'avoir des noeuds sans raideur venant d'élément n'est pas interdit, par contre lorsque ces noeuds ne sont pas bloqués cela peut engendrer des erreurs difficiles à voir. Et lorsqu'ils sont bloqués et coïncidant avec d'autres noeuds, on peut être étonné des comportements obtenus.
- cor bug pression hydro : correction d'une erreur d'orientation (ticket #136)
- cor bug sur les conditions linéaires dans le cas d'une géométrie axisymétrique

263 **versions 6.792-93**

- introduction du calcul d'intégrales de volumes (mais non validé pour l'instant)
- correction loi Mooney Rivlin 1D (ticket #140)

264 **version 6.794 :**

- introduction d'une initialisation de température pour le calcul du temps critique de stabilité, utilisé par les algos de type explicite (ticket #141)

265 **version 6.795 :**

- introduction d'un nouvel utilitaire permettant la création d'un nouveau maillage à partir d'un maillage déjà existant et d'une référence d'éléments $->$ le nouveau "sous-maillage" contient également les références réduites.

266 **version 6.796 :**

- correction d'un bug sur l'utilisation de variable globale par les fonctions nD existantes,
- amélioration de l'algorithme d'amortissement cinétique : intro d'un sous-cas qui "semble" bien améliorer la convergence globale, lors d'une variation pendant un même pas de temps, du chargement et/ou de la loi de comportement... à valider plus en détail par retour d'expérience.

version 6.797 :

- prise en compte automatique des référence i_tout (i= N, F, A, E, G, P) du premier maillage de manière à récupérer ces références dans le maillage final,
- introduction d'une nouvelle option : " `calcul_geometrique` ", pour l'algorithme " `informations` " : − > un premier calcul : distance initiale entre deux noeuds

version 6.798-6.801 :

- corr bug cas init certains contacts déformables rigides
- modification du pilotage de lambda dans algo relax, amélioration création SFE3 : prise en compte de la distorsion du maillage pour un choix automatique des SFE1 dans le cas où les SFE3 sont mis en défaut
- extension du pilotage des différents chargements à l'aide d'une fonction nD pouvant dépendre de la position du point d'application pour les forces ponctuelles, du point d'intégration de la surface chargée pour les chargements de surface, du point d'intégration de la ligne pour les chargements linéiques
- validation sur des cas d'école, des intégrales de volume (ou de surface, ou linéique suivant le type d'élément fini support) de fonction nD (utilisateur), idem pour une intégration en volume + en temps,
- introduction des grandeurs intégrées, au niveau global, au temps courant "et" au dernier pas de temps qui a convergé
- validation sur des cas d'école, de l'utilisation des grandeurs globales issues des intégrales de volume et en temps, dans des fonctions nD permettant ainsi de piloter tout ce qui est pilotable par des fonctions nD, en particulier les chargements et certaines parties des lois de comportement combiné.
- sortie dans le .BI de la valeur des intégrales de volume et en temps
  ==> (beaucoup de changement de code => demandera peut-être des ajustements à l'usage)

version 6.802 :

- implantation de 3 nouveaux éléments : . élément biel quadratique, . élément biel linéaire axisymétrique, . élément biel quadratique axisymétrique, Il faut maintenant les tester,
- modification de la loi `Maheo_hyper` pour qu'elle fonctionne en dynamique, et en relaxation dynamique

version 6.803

- modification relaxation dynamique : jusqu'à présent le temps fin de calcul en relaxation dynamique n'était pas respecté exactement (d'une manière analogue à l'explicite). C'est maintenant modifié pour garantir des temps de fin strictement satisfaits.
- correction d'un bug sur l'utilisation des fonctions nD dépendantes du point pour les différents chargements.

version 6.804

- correction d'un bug qui apparaissait uniquement à la fin de l'exécution, lorsque l'on faisait une succession d'opérations sur les maillages avec création et suppressions d'éléments et de maillages. La surcharge de l'opérateur de copie des frontières de l'élément était incomplète d'où un pb au moment de la destruction.

version 6.805 - 6

- modification du statut des données imposées c-a-d lorsque on met en place une condition limite (ou une initialisation) sur une grandeur qui n'est pas directement un ddl primaire (= un déplacement pour un problème mécanique) cette grandeur prend automatiquement le statut de donnée en service (auparavant elle avait le statut de donnée hors service). Ensuite, la donnée est "lisible libre" si elle n'est pas active (le temps est inf au temps mini ou sup au temps maxi de la condition), et bloquée sinon. Auparavant il fallait qu'un objet particulier (exemple : une loi de comportement) active la donnée pour la rendre lisible sinon elle restait HS. A priori ce fct n'est pas suffisamment souple d'où la modif. Normalement cela ne doit rien changer au fct actuel, mais... à voir

version 6.807-8

- amélioration des éléments biel axi : les éléments biel axi linéaires et quadratiques étaient implantés, mais n'avaient pas été testés donc plusieurs bugs au rendez-vous qui ont été réparés. . les éléments ont été testés sur un cas basique de stabilité d'un ballon sphérique soumis à une pression uniforme, dont on a une solution théorique. On retrouve la solution théorique avec néanmoins une petite déviation au niveau des conditions limites... c'est peut-être normal compte tenu de la discrétisation ? . les éléments ont également été testés sur une forme isotensoïde proposée par Anne-Sophie Lectez et on obtient quelque chose qui parait correct ...(à compléter néanmoins)
- amélioration de l'hystérésis bulk : dans le cas de boucles il peut arriver que sur un même pas de temps on ait une inversion + une coïncidence. Ce phénomène peut apparaître particulièrement dans le cas de petites boucles et au niveau du début du chargement. Ce cas n'était pas pris en compte d'où des erreurs observées par Émilie Vieville dans ses calculs de thèses. Normalement maintenant c'est ok.

version 6.809

- nouvelle modification de l'avancement temporel pour l'hystérésis Bulk, avec amélioration du filtrage des dépassements de la saturation
- amélioration du contact : le paramètre `PENETRATION_CONTACT_MAXI` qui contrôle le maximum de pénétration autorisée dans le cas d'un " `TYPE_PENALISATION_PENETRATION` 4 " est maintenant satisfait de manière plus rigoureuse.

version 6.810 et 11

- correction bug 1D contact + améliorations commentaires contact
- correction bug sortie des déformations logarithmiques

version 6.812

- amélioration du contrôle de la loi critère plis : introduction de la possibilité de prendre en compte des paramètres différents pour les lois en contraintes planes (CP) et contraintes doublement planes (CP2)
- modification des initialisations de déformations transversales lors du passage CP à CP2 et vis-versa
- modification de la prise en compte de la non-convergence des algo de Newton en CP et CP2 : un appel direct à la loi 3D, est fait avec les variations de déformations transversales inchangées. On suppose qu'à convergence, l'algo de Newton ne sera pas mis en défaut (ce qui semble le cas dans les tests effectués).

version 6.813

- amélioration du calcul des épaisseurs et du post-traitement pour le critère plis
- ajout chapitre 12.3 sur la doc théorique, explication des variations d'épaisseur calculées avec le critère plis

version 6.814-16

- mise en place d'un nouveau fonctionnement des fct nD, via des grandeurs sous forme scalaires, mais également vectorielles/tensorielles. En pratique pour l'utilisateur cela se traduit par la possibilité de piloter les chargements via une fonction nD qui peut dépendre : . de grandeurs globales : soit celles existantes systématiquement comme le volume, l'énergie..., soit de nouvelles grandeurs qui apparaissent au cours du calcul sous forme d'intégrales sur un domaine particulier (cf. définition d'intégrale de fonction nD) . de grandeurs cinématiques : positions, delta de positions, vitesse et accélérations s'il s'agit d'un calcul dynamique . de données interpolées, mises en place par la mise en données : Température par exemple
- Dans le cas des chargements volumiques, il est possible d'utiliser toutes les grandeurs existantes aux points d'intégration. Pour les autres chargements, les grandeurs disponibles sont restreintes à celles existantes aux noeuds.

version 6.817-18

- cor d'un bug sur l'utilisation de fonction nD dans le chargement

— mise en place de la possibilité d'utiliser des fonctions nD conjointement avec des courbes de charges pour appliquer des conditions cinématiques : pour des ddl fixés directement par les fonctions ou pour des mouvements solides. Herezh est également à jour pour créer la mise en données correspondantes en interactif (cf. option -n au lancement d'herezh)

version 6.819 : (version test)

— prise en compte pour le calcul des temps cpu, d'une sortie inopinée : les temps cpu était erronés, car le comptage n'était pas correctement finalisé.

— modification de la prise en compte du cas type de contact = 1, dans le dimensionnement de la matrice de masse en relaxation dynamique. Dans le cas où les surfaces maîtres ont une cinématique imposée, les conditions linéaires associées au contact type 1, sont réduites aux ddl de chaque noeud indépendamment . En conséquence, la matrice masse est transformée par défaut en une matrice bande tridiagonale. Mais on peut également indiquer un autre type de stockage via la mise en donnée habituelle. − > concernant la demande #115, la comparaison sur les 1000 premiers incréments entre les versions 6.812 (qui comportait un bug qui permettait l'utilisation d'une matrice strictement diagonale) et la version 6.819 avec une bande tridiagonale ne présente pas de différence en temps cpu (la partie résolution du système linéaire est négligeable / aux calculs des forces généralisées)

version 6.820 : (version test))

— correction bug fonction nD (ticket #156)

version 6.821-25 : (versions tests)

— corrections de bugs sur les fonctions nD (cf. ticket #156)

— optimisation de l'accès en interne aux grandeurs nécessaires pour la sortie de résultats et pour l'alimentation des fonctions nD (possibilités de bug de jeunesse, car un nombre important de modifs)

— introduction de la possibilité dans les fonctions nD, d'utiliser les coordonnées à t=0 et à t à l'aide des noms : `X1_t` , `X2_t, X3_t, X1_t0, X2_t0, X3_t0`

— introduction de la possibilité dans les fonctions nD, d'utiliser les normales à t=0 et à t à l'aide des noms : `N_surf_1_t, N_surf_2_t, N_surf_3_t, N_surf_1_t0, N_surf_2_t0, N_surf_3_t0`

Version 6.826-27 : (versions tests) Mise en place d'un restart pour des fonctions nD et des intégrales associées d'où plusieurs ajouts dont :

(a) Concernant les intégrations

— le stockage (.BI) au niveau des éléments, des intégrations locales

— le stockage (.BI) des intégrations globales

— la possibilité de geler une intégration après restart

(b) au niveau des fonctions nD :

— un mécanisme récursif de sortie d'erreur lors de l'utilisation de fonctions emboitées, avec une information détaillée sur les paramètres d'appel à chaque niveau,

— une amélioration des tests d'erreurs sur le cas de manipulation de grandeurs très grandes

— la possibilité d'introduire un niveau d'impression particulier, sur chaque fonction de manière séparée : en particulier on peut ainsi afficher la valeur des paramètres d'appel et celles de la fonction

Version 6.828 : (versions tests)

— correction bug #166

— ajout dans la loi Maxwell 3D d'un niveau local de sortie d'info

— validation de l'utilisation d'une umat interne (sans utilisation de pipe) en contrainte plane : la suite via les pipes est en cours.

Version 6.829-33

— mise en place d'une première version opérationnelle d'Umat CP

— corrections bug sur algo Newmark,

— correction d'un bug sur la sortie des .her et .lis lorsque l'on demande une sortie sans calcul (via " plus " mais sans le "avec" après la définition de l'algorithme). Auparavant sortait des positions nulles ! !

— au niveau de la définition de nouvelles références via Herezh :

— ajout de la possibilité de choisir des numéros de noeuds au lieu de positions pour la définition des plans, des droites, des points dans le cas des créations de références

— Correction d'un bug sur l'affichage des frontières lors de la sortie d'informations pour l'utilisateur

— Mise en place des 3 nouvelles méthodes `entre_plan_dist` , `entre_droite_dist` , `entre_point_dist` , par exemple `entre_plan_dist` propose une condition entre plans qui est définie à partir d'un plan médian et d'une distance +- via la normale du plan.

— modifications relatives au contact :

— Mise en place d'une nouvelle gestion des zones de contact plus précise, optimisée −− > beaucoup de changements et d'essais d'améliorations (?)

— prise en compte de contacts multiples − > premier test sur le contact collant ok ... correction des autres bugs :

— Mise en place d'un contact collant : mise en oeuvre d'une pénalisation autonome (− > paramètres supplémentaires) sur le frottement, validation sur un cas test

— Mise en place de la possibilité de suppression des gaps de contact avant le début du calcul en déplaçant les noeuds esclaves sur les surfaces présumées de contact.

— Mise en place de la possibilité de renuméroter à la volée en fonction des contacts mise en jeux. Application à la possibilité d'optimiser en continu les matrices de raideurs.

Version 6.834

— ajout d'un message d'erreur supplémentaire dans le cas de la définition de zones de contact qui sont incomplètes ou inutilisables

— ajout dans la loi de Maxwell 3D, dans l'opérateur tangent dsigma/deps , et dans le cas de l'utilisation de fonction d'évolution de E et/ou des mu en fonction de la déformation de mises et/ou de `Q_Deps` , du cas de repère curviligne (cas normal dans Herezh fonctionnant seul) le cas auparavant implanté était uniquement dédié aux repères particuliers orthonormés : cas d'un fonctionnement d'Herezh en Umat par exemple.

— correction d'un bug (en fait il manquait une partie) sur la définition d'un plan de ref à partir d'un noeud et d'une normale dans la partie : définition interactive de référence

Version 6.835-6.841

— corr bug `Suppression_gap_pour_noeud_collant` + d'autres bugs de jeunesse vis-à-vis des nouveaux ajouts

— introduction de la prise en compte de repères d'anisotropie au niveau des éléments

— travaux concernant la mise en place d'une loi élastique orthotrope entraînée. Pour l'instant, pas encore opérationnelle, la mise en données est lue et exploitée correctement (paramètres fixes de la loi + repère d'orthotropie) ... à suivre.

Version 6.842 (version de développement)

— mise à jour de la documentation d'utilisation de la loi d'orthotropie élastique entraînée

— modif dans l'insertion d'un repère d'anisotropie,

- intro de la sortie d'info en post-traitement pour la loi d'orthotropie élastique entraînée : ajout d'une grandeur quelconque type repère
- .. mise au point de la loi d'orthotropie élastique entraînée : à suivre ...

289 **Version 6.843 (version de développement)**

- mise à jour et amélioration des sorties de commentaires dans loi critère + plis, dispo pour la version non fast et non actif pour la version fast
- correction de bugs, et amélioration du fct critère plis
- amélioration de la prise en compte des masses nulles
- doc utilisateur complété sur paramètre algo relaxation dynamique et présentation des mots clés pour les algos
- doc théorique complété sur des détails

290 **Version 6.844 (version de développement)**

- pour la relaxation dynamique, introduction de la variable globale `AMOR_CINET_VISQUEUX` qui permet de savoir si on est en amortissement cinétique ou visqueux
- correction d'un bug d'écriture temporaire sur le comptage du temps cpu dans les lois de comp
- introduction dans l'écriture des temps cpu, du type de version : rapide ou non
- doc mise à jour (utilisateur et dans Herezh) sur les paramètres spécifiques d'amortissement mixte en relaxation dynamique

291 **Version 6.847**

- correction de bugs
- améliorations de la loi maxwell 3D
- ajout de la possibilité de moduler la précision d'équilibre global à l'aide d'une fonction nD qui utilise uniquement des variables globales
- amélioration de la loi critère : l'idée est de figer la direction des plis pendant plusieurs itérations (voir sur un incrément ?) $->$ introduction d'un paramètre qui permet d'activer le recalcul de la direction : par défaut le recalcul s'effectue à chaque itération : maintenant il peut-être piloté par une fonction nD
- amélioration des messages d'erreur dans la méthode interne de Newton, utilisée par exemple dans les lois complexes

292 **Version 6.848**

- mise en place d'une dépendance possible des paramètres d'une loi isotrope élastique, à une fonction nD
  $->$ possibilité d'obtenir en sortie les valeurs des paramètres matériau réellement calculés à chaque point d'intégration
- mise en place pour la loi de Maxwell 3D de la possibilité d'obtenir en sortie les valeurs des paramètres matériau (`E, nu, mu, mu_p`) réellement calculés à chaque point d'intégration

293 **Version 6.849**

- premières validations de la loi d'orthotropie entraînée :
  (a) cas d'un cube et d'un hexaèdre dont les cotés sont alignés avec les directions initiales d'orthotropie :
    - vérification de la convergence en implicite via l'opérateur tangent d_sigma/d_ddl,
    - vérification du bon calcul de chacune des contraintes (tractions, cisaillements)

  (b) cas d'un hexaèdre avec plusieurs éléments et des directions initiales d'orthotropie, à 45° des côtés de l'hexaèdre :
    - vérification du calcul du repère via les fonctions nD et de son transfert dans la loi de comportement aux points d'intégration,
    - vérification de la bonne convergence du calcul en implicite

  (c) cas d'une poutre en flexion maillée en éléments quadratiques avec intégration complète, et un repère d'orthotropie à 45°
    - vérification de la bonne convergence du calcul en implicite

  NB : pour l'instant l'opérateur tangent d_sigma/d_eps n'est pas encore opérationnel, la mise au point de l'opérateur d_sigma/d_ddl a mis en évidence des modifications à effectuer qui sont en cours.

294 **Version 6.850**

- mise en place de l'opérateur tangent d_sig/d_eps pour la loi de comportement d'orthotropie entraînée $->$ possibilité d'utiliser un comportement ce contrainte plane.
  - mise à jour de la Documentation :
    .. partie théorique : Loi de Hooke initialement ortotrope puis matériellement entraînée. theorie_herezh++.pdf : version 6.850 : chapitre 4 : page 35 - 46
    .. manuel d'utilisation : utilisation_herezh++.pdf : version 6.849 : chapitre 36.15.1 : page 290 - 292
- Tests effectués : en 3D volumique, en 2D contraintes planes
  - sollicitations basiques : traction, cisaillement : comparaison analytique,
  - convergence sur un problème plus complexe :
    . flexion d'une poutre avec des éléments complets quadratiques, algorithme Newton-Raphson en quasi-statique
    . gonflement d'une membrane via une géométrie de coussin carré (sans la prise en compte de plis) : algorithme de Relaxation dynamique
- validation de la version 6.850 vis-à-vis des tests actuellement enregistrés pour Herezh

295 **Version 6.851**

- modif de l'algo utilitaire de renumérotation lorsqu'il y a plusieurs maillages avec éventuellement des CLL : cf. #175 (il y a encore de nouvelles modifs prévues pour automatiser les choix par défaut)
- modif de la sortie des contraintes et déformations locales dans le cas d'une loi orthotrope entraînée (les contraintes et déformations exprimées dans le repère d'orthotropie)
- cor d'un bug sur les sorties en format maple des moyennes de grandeurs quelconques et particulières

296 **Version 6.852-53 (version de développement)**

- complément dans la loi orthotrope entraînée concernant le transport des grandeurs associées à chaque point d'intégration
- ajout de la prise en compte des raideurs de contact dans le calcul de la stabilité pour l'algorithme de relaxation dynamique
- mise en place de plusieurs grandeurs disponibles en post-traitement du contact, accessibles avec l'option debug et au fil du calcul (force de contact, pénétration, glissement, force de glissement, énergies, statut ... voir doc chap : 50.6 Remarques sur le post-traitement)
- modification exploratoire du type 8 de contact... à suivre

297 **Version 6.854 (version de développement)**

- amélioration de l'algorithme de calcul d'erreur et sur la sortie des résultats d'erreur

— correction d'un bug sur le dimensionnement du conteneur des volumes déplacés en 2D et 1D, calcul qui n'existe pas en 2D et 1D.

Version 6.855 (version de développement)

— correction erreur sur la sortie des vecteurs propres pour les tenseurs (il y avait inversion des directions)
— introduction d'un addimentionnement au niveau du calcul des inverses de matrices de dimension 1,2, 3
— modification de la loi ortho vis-à-vis de son utilisation en CP2 + plis

Version 6.856 (version de développement)

— correction erreur sur l'utilisation de force linéique avec des éléments pentaédriques
— correction erreur sur la mise en données d'un calcul 1D

Version 6.857 (version de développement)

— correction d'un bug un peu gênant sur le constructeur de copie de noeuds : qui induisait certaine fois un crash lors de la fusion de maillages

Version 6.858-6.864 (versions de développement)

— corr d'un bug de vérification
— amélioration loi CP2
— modif recalcul dir plis
— ajout de nouvelle sortie (vecteur résidu avant résolution)
— amélioration sortie grandeurs particulières loi plis, et de l'affichage d'infos
— modif affichage incrément sur l'algo global pour loi Umat
— ajout de la prise en compte de la température au noeud pour un élément point (utilisé pour la loi Umat)
— correction d'un bug sur le constructeur de copie de l'hystérésis 3D (utilisé pour un critère plis)

Version 6.865

— introduction dans l'algorithme de relaxation dynamique du paramètre " `et_recalcul_masse_a_la_transition_` " qui pour un amortissement mixte (type 4) permet de recalculer ou non, la matrice masse juste au moment de la transition : Hugo Le Meitour a montré que le recalcul de la masse améliore la convergence (pour certains cas, sans cette option, pas de convergence)
— introduction dans l'algorithme de relaxation dynamique du paramètre " `opt_visqueux_recal_mass=` " qui permet d'avoir conjointement deux types de re-calcul de masse, un type pour l'amortissement visqueux et un type pour l'amortissement cinétique

Version 6.866-69

— correction de plusieurs bugs d'initialisation
. initialisation des algos utilitaires, lorsqu'il y a des fonctions nD
. initialisation de tous les algos, dans le cas de l'utilisation d'une loi ortho entraînées avec une en particulier une sortie d'info (la création du CVisu plantait !)
— introduction d'un message explicite d'erreur en cas de taille de matrice de raideur trop importante pour l'ordinateur utilisé, ceci dans le cas d'un stockage matriciel par défaut (stockage bande symétrique)
— introduction de la possibilité de piloter le paramètre `inter_nb_entre_relax` (cf. amortissement cinétique) via une fonction nD
— introduction de la possibilité de piloter le paramètre de recalcul de la masse via une fonction nD (voir la documentation)
— modif de l'affichage des erreurs de mise en données, lorsqu'il n'y a une loi absente de stabilisation d'hourglass
— transformation du mot clé " `et_recalcul_masse_a_la_transition_` " en " `et_pas_recalcul_masse_a_la_transition_` " , le recalcul de matrice masse à la transition devient la version par défaut et non l'inverse car les tests de Hugo montrent que c'est toujours plus performant

Version 6.870-72

— ajout de possibilité d'un niveau de commentaire particulier pour toutes les lois de comp, en cohérence avec l'existant
— refonte de l'algo de stabilisation de membrane et ajout d'une technique dérivée
— ajout de 4 nouveaux cas de calcul de la pseudo-masse en relaxation dynamique
— affichage des numéros d'incrément dès le premier pour le fonctionnement en UMAT

Version 6.870-73

— mise en place d'une première version fonctionnelle, intégrant le nouvel algorithme de calcul de contrainte doublement plane. Sera en particulier, utilisée avec la nouvelle loi plis (qui est à venir). (cf. doc théorique)
— mise en place d'une méthode de régularisation (au sens mathématique) de la singularité pour un chargement nul, des lois hyperélastiques (ex : Favier, Orgeas...) utilisant les invariants : V, Qeps et phase (= angle de Lode). Voir doc

Version 6.874

— correction bug #193 : une erreur de directive de précompilation sur le changement de base de tenseur du 4ieme ordre de dimension 2
— modification de l'algo de gestion d'Umat (reste à confirmer)

Version 6.875

— correction d'un mauvais choix sur l'amortissement cinétique, qui pénalisait (et même donnait une non convergence sur un test) la convergence et donc la vitesse en relaxation dynamique (trouvé grâce à une vieille non-régression passée sous les radars : en fait qui a eu lieu entre les version 6.858 et 6.859 !)
— modification d'initialisation lors des procédures de restart entre algo différents
— fin de la première implémentation de la loi plis deuxième mouture $->$ qu'il faut maintenant mettre au point !

Version 6.876-79

— correction pb #195 (sauvegarde conteneurs pour les intégrales volumique)
— mise à jour du module de compressibilité tangent Kt pour la loi isoelastique 3D
— sortie affichage écran suivant le niveau d'affichage spécifique à la loi, de la valeur de la proportion, dans une loi de mélange, pour les versions non fast
— correction bug sur les courbes 1D union dans le cas où on fait l'union de plus de 2 courbes
— validations élémentaires de la loi plis version 2 : cas de cisaillement en loi élastique et avec des maillages 10x10 $->$ on retrouve des résultats identiques avec la première version... à suivre

Version 6.880-81

— correction d'un bug sur le pilotage du calcul (et recalcule éventuel) des directions de plis dans la loi critère.
— introduction d'un test lors de la création d'un nouveau fichier .info : si le fichier existe déjà, on demande à l'utilisateur s'il veut vraiment écraser le fichier existant.

— première mise en place des torseurs de forces ponctuelles

Version 6.882

— fin de la mise en place des torseurs de forces ponctuelles : tests de validations des différentes options (cf. documentation d'utilisation)

— introduction d'un pilotage possible via une fonction nD des temps min et max sur les conditions limites cinématiques et de charge

Version 6.883-85

— intro de la possibilité de définir des masses ponctuelles dont la valeur dépend d'une fonction nD : non validé pour l'instant

— amélioration du contact axisymétrique

— mise en place de la possibilité d'activer ou non le contact entre plusieurs `suite_point_info`

— corr bug #199 : le cas relaxation dynamique avec amortissement visqueux suivi d'un calcul implicite puis retour à de la relaxation dynamique avec amortissement visqueux (via des restarts) fonctionne maintenant.

— introduction de la possibilité de définir une épaisseur moyenne initiale via une fonction nD dépendant le la position, ceci pour des éléments membranes triangulaires. La méthode a été validée sur un cas test et la mise en données interactive a également été mise à jour.

Version 6.886-91

— extension de la possibilité de définir une épaisseur initiale via une fonction nD dépendante de la position, ceci pour les élément quadrangulaires : validation sur un cas test,

— mise en place d'un stockage particulier au niveau du conteneur relatif à la loi de comportement au point d'intégration > dans les messages d'erreur sur la loi de comportement, on peut ainsi indiquer les numéros : pti, élément, maillage amélioration de la renumérotation en axisymétrique : on ne travaille plus que sur les ddl dans le plan, pour la résolution globale

— correction d'un bug sur les efforts linéiques en axisymétrique

— mise en place de la possibilité d'optimiser la numérotation des pointeurs d'assemblages, indépendamment de la numérotation des noeuds > intéressant pour le cas où on a plusieurs maillages avec des relations entres ces maillages, réduit considérablement la taille des matrices globales mise en place et validation sur un cas test, de la nouvelle méthode de calcul automatique du facteur de pénalisation pour des éléments 2D.

— correction d'un manque sur la sortie des vecteurs donnant la direction des plis pour la loi plis, dans le cas des vecteurs ortho pour une zone totalement relâchée

Version 6.892-93

— correction bug de jeunesse sur sortie plis,

— correction bug sur la suppression des éléments ayant des noeuds très proches

— correction bug sur une initialisation de la loi pli 2

Version 6.894-95

— mise en place d'un pilotage possible via des fonctions nD, des tolérances utilisées dans les lois contrainte plane et contrainte doublement plane.

— amélioration loi plis2 et CP2 en hors axes, mais il reste des pb de fct lors de l'utilisation de lois orthotropes ... la convergence n'est pas bonne dans ces cas là.

— correction bug #200 #201

Version 6.896

— correction d'une erreur d'affichage du nombre d'éléments dans les refs définies automatiquement lors de la réorientation de facette

— ajout de la possibilité d'une norme pilotée par une fonction nD

— ajout des grandeurs globales : `MAXPUISSEXT` , `MAXPUISSINT` , `MAXREACTION` , `MAXRESIDUGLOBAL` , `MAXdeltaX` , `MAXvarDeltaX` , `MAXvarDdl` introduction du calcul systématique de ces grandeurs dans tous les algo

Version 6.896

— correction loi CP2 hors axes + validation ortho

— modification du fonctionnement du contact type 4 (cf. doc)

Version 6.897 - 6.906

— modification loi CP2 : inversion matrice carrée non symétrique

— essai correction entrées clavier successives pour la définition de références

— introduction de la loi de projection anisotrope 3D

— correction d'un bug sur les sorties relatives aux lois d'orthotropie entraînée et de projection anisotrope : concerne en particulier (mais pas seulement) la visualisation des repères d'anisotropie

— modification du calcul des repères ad hoc (concerne toutes les sorties)

— correction d'un bug de sortie en repère ad hoc

— passage ok de tous les tests actuels de validation − > version validée

— corr bug ele axi 1D lin et quad en explicit : le nombre de pti était bloqué à 1, alors qu'il faut au moins 2 pti

— chgt nb pti : 2 -¿ 4 pour ele 1D quadra axi

— insertion de la gestion des modes d'hourglass pour tous les éléments 1D. Actuellement les éléments implantés ne présentent pas d'hourglass. Mais par la suite il sera possible d'utiliser des éléments sous-intégrés.

Version 6.906 - 6.908

— corr bug sur le calcul des torseurs de réaction dans le cas de l'algo de relaxation dynamique

— algo de relaxation dynamique : modification du critère de convergence lorsque l'on utilise `ARRET_A_EQUILIBRE_STATIQUE_` 2 Dans certains cas, le critère en déplacement n'était pas utilisé. D'autre part lorsque l'énergie cinétique reste nulle sur 2 itérations consécutives, on considère que le critère en déplacement est satisfait.

— mise en place de la possibilité de définir des "constantes utilisateurs" au début du fichier .info et de les modifier après un " `_suite_point_info_` " (voir documentation). Les constantes sont gérées comme des grandeurs globales. Elles sont donc utilisables par les fonctions nD. De plus tout paramètre de contrôle de type réel, entier ou booléen, peut être définie à l'aide d'une constante globale qui est évaluée au moment de la lecture du paramètre de contrôle.

Version 6.909 - 6.910

— corr bug init poly-hyper, modif init RD (cf. #209)

— corr bug construction de repère ad hoc pour les éléments 1D, utilisés en post-traitement en absolu (cf. #211)

— modification en lien avec le contact : la méthodologie retenue est la suivante : a) la présence d'1 ou plusieurs maillages esclaves est "nécessaire" pour initialiser un contact potentiel. Le contact peut très bien ne pas être actif. Par contre si aucun maillage esclave n'est déclaré, l'activation du contact conduit à une erreur. b) il n'est pas possible d'introduire des maillages esclaves et éventuellement des zones particulières de contact au moment d'un restart ou un `suite_point_info_` C) il est toujours possible de modifier l'activité du contact (0 ou autres), les types, et en fait tous les paramètres de contact au moment d'un restart ou après un `suite_point_info_`

320 **Version 6.911**

— introduction de la loi hypo-élastique anisotrope entraînée : les tests élémentaires de traction, cisaillement, et un test de flexion, fonctionnent.

— correction d'une "belle erreur" due à un cast de double vers int dans le cas de la sauvegarde des grandeurs globales (via *void) : volumes par rapport aux plans, et toutes les énergies − > en particulier il est probable que les fonctions nD qui utilisaient ces grandeurs devaient fonctionner a priori incorrectement (?)

321 **Version 6.912**

— correction d'un bug sur le post-traitement : en fait sur le stockage des grandeurs quelconques aux noeuds. Il y avait un dimensionnement par rapport au nombre maxi de grandeurs quelconques or cette dimensions peut augmenter pendant le calcul... ce n'était pas pris en compte, c'est ok maintenant

— ajout de garde-fous numériques sur le calcul d'épaisseurs en fonction des valeurs du module de compressibilité et de la trace des contraintes, ceci pour les éléments membranes triangulaires et quadrangulaire, et les éléments coques SFE

322 **Version 6.913-15**

— correction sur le calcul de Ks sur les lois `IsoHyperBulk3` et `IsoHyperBulk_gene` pour une déformation nulle (le calcul s'effectue par différence finie)

— correction sur un post-traitement

— introduction d'un nouveau paramètre de pilotage sur les algo. utilisant la méthode de Newton de recherche de 0. On peut maintenant spécifier une limite maximale sur la variation de la norme des degrés de liberté du problème, ceci d'une itération à l'autre. Application aux lois : CP, CP2, Hystérésis. Par défaut, pas de changement pour les lois d'hystérésis : la variation des contraintes n'est pas bornée (mais l'utilisateur peut le faire). Pour les lois CP et CP2, la variation des déformations est limitée par défaut à 10% sur chaque itération

— introduction d'une gestion du cas particulier, où l'utilisateur oublie de terminer son fichier .info par un mot clé

323 **Version 6.916-25**

— corr bug #215, modif fct nD

— introduction de l'algorithme combiné

— intro sortie conditionnelle d'info dans `isoelas2D_C` cf #217

— intro des paramètres spécifiques pour les sous algo

— intro doc interactive sur l'algo combiné (cf. -n)

— algo combiner : integ restart, integ de tous les sous-algo existants

— ajout dans le critère plis du cas : `recalcul_dir_plis_` == -2 et -3

324 **Version 6.926-35**

— correction d'une erreur (de base ! !) sur la résolution d'un système linéaire à 2 inconnues via Cramer ...

— correction #222, premier test de la loi ortho 2D_C

— modif lecture de la loi de projection anisotrope, ajout de la possibilité d'une dépendance à la contrainte et def de Mises pour les fct nD (cf. #220)

— modifs sur le contact et algo RD (cf. #223)

— amélioration de l'algo combiner (pour Hugo)

— modif des forces de stabilisation de membrane et des répercussions dans l'algo implicite

— implantation de la possibilité d'utiliser des fonctions nD externes, développées dans un langage quelconque, et dialoguant avec Herezh via des pipes nommés

— implantation de la possibilité de modifier des constantes utilisateurs, en temps réel via une interruption (kill -15)

325 **Version 6.936-44**

— correction bugs : lecture anisotropie, dans la def de constantes simples, lecture nb iter resolution iterative, contact

— ajout de l'affichage des noms de variables pour les fct nD externes

— introduction de la possibilité de sortie en post-traitement de grandeurs aux pti de faces et d'arêtes d'élément

— introduction de la possibilité de définir des frontières "utilisateur" pour le contact, modification de l'initialisation du contact

— introduction de la possibilité de définir des variables utilisateurs utilisables globalement via les fct nD. Ces variables sont alimentées par les résultats de calcul . Elles peuvent représenter a priori toutes les grandeurs définies aux noeuds, aux pti d'élément, aux pti de face et arêtes d'élément. Pour l'instant seules les grandeurs définies aux noeuds sont validés sur des cas simples. Le reste est implanté mais sans validation... à suivre

— correction de l'opérateur tangent en déformation, sur la loi de projection (cf. #233)

326 **Version 6.945-46**

— mise ne place de nouvelles sorties aux noeuds pour le format Gid (sorties déjà présentent pour les formats maple et gmsh). − > on peut obtenir maintenant en plus des ddl classiques, les ddl étendus disponibles, et les grandeurs quelconques : ex : les vecteurs réaction, forces généralisées internes, forces généralisées externes ... et les grandeurs spécifiques au pb traité

327 **Version 6.947**

— correction d'un bug sur la sortie du contact pour un niveau > 5 dans `LesContacts::Suppression_gap_pour_noeud_collant` ()

— contact : modif message si raideur du maître n'existe pas

— contact : si type de contact = 0 en tangentiel et/ou en pénétration > le facteur de pénalisation 0 contact : ajout pour les variables de contact,
− > de la sortie de `cas_collant` , et pour l'utilisation dans un fctnD de pilotage du contact, mise à jour de la doc
− > idem pour le numéro de la zone de contact

328 **Version 6.948-49**

— correction pb sur l'incrémentation du numéro d'incrément, lors d'un restart, on sautait un numéro !

— correction d'un pb sur la stabilisation qui utilise les coordonnées de l'itération 0. Cela fonctionnait en implicite mais pas en explicite et RD,

— intro de la possibilité de changer d'algo (combiné vers non combiné ou inverse) au moment d'un restart, quelque soit les algos impliqués.

— introduction d'une loi hypoélastique isotrope 1D. Voir la documentation théorique et utilisateur pour les détails.

— introduction des algo :
`NON_DYNA` , `RELAX_DYNA` , `DYNA_IMP` , `DYNA_EXP_CHUNG_LEE` , `DYNA_EXP_TCHAMWA` , `DYNA_EXP` , `DYNA_EXP_ZHAI` , `DYNA_EXP_BONELLI`
au niveau de la possibilité de faire un `suite_point_info_`

Références

[Bles, 2002] Bles, G. (2002). *Bases thermomécaniques de la modélisation du comportement des matériaux tissés et des polymères solides.* Thèse de doctorat, University UJF Grenoble 1.

[Couty, 1999] Couty, N. (1999). *Étude expérimentale de phénomènes de choc : modélisation en dynamique incluant un comportement d'élasto-visco-hystérésis volumique et coque. Application à l'impact de plaques.* Thèse de doctorat, Université de Bretagne Sud.

[Favier *et al.*, 2002] Favier, D., Liu, Y., Orgeas, L. et Rio, G. (2002). Mechanical instability of niti in tension, compression and shear. *In IUTAM Symposium on Mechanics of Martensitic Phase Transformation in Solids*, pages 205–212. Springer.

[Favier *et al.*, 2010a] Favier, D., Orgéas, L., Rio, G. et Liu, Y. (2010a). Elastohysteresis modelling of ferroelastic and pseudoelastic behaviour. *In S(3)T 2010 Smart Structural Systems Technologies*, Porto, Portugal.

[Favier *et al.*, 2010b] Favier, D., Orgéas, L., Rio, G. et Liu, Y. (2010b). Elastohysteresis modelling of ferroelastic and pseudoelastic behaviour. *In Workshop S3T EUROCORES, Porto, Portugal - S(3)T2010 School and Symposium on Smart Structural Systems Technologies*, numéro ISBN 978-989-96697-0-3, pages 249–261.

[Favier *et al.*, 1997] Favier, D., Rio, G. et Manach, P.-Y. (1997). Grandes déformations et loi d'élastohystérésis appliquées au calcul par éléments finis d'élastomères. *In* G. Coupard éditeur, Apollor, N., éditeur : *Génie Mécanique des caoutchoucs et des élastomères thermoplastiques*, pages 327–330. Apollor.

[Guélin, 1980] Guélin, P. (1980). Remarques sur l'hystérésis mécanique : les bases d'un schéma thermo-mécanique à structure héréditaire. *Journal de Mécanique*, 19(2):217–247.

[Hulbert et Chung, 1996] Hulbert, G. et Chung, J. (1996). Explicit time integration algorithms for structural dynamics with optimal numerical dissipation. *Comp. Meth. Appl. Mech. Engng.*, 137:175–188. PII : S0045-7825(96)01036-5.

[Laurent *et al.*, 2007] Laurent, H., Vandenbroucke, A., Couedo, S., Rio, G. et Hocine, N. (2007). An hyper-visco-hysteretic model for elastomeric behaviour under low and high temperature : experimental and numerical investigations. constitutive models for rubber. *In V, Eds Boukamel Laiarinandrasana, Méo and Verron*, pages 47–52, Paris, France. ECCMR.

[Manach *et al.*, 1996] Manach, P. Y., Favier, D. et Rio, G. (1996). Finite element simulations of internal stresses generated during the pseudoelastic deformation of niti bodies. *J. de Physique*, C1(6):244–253.

[Moreau, 2000] Moreau, C. (2000). *Etude expérimentale et Numérique de l'Hyperélasticité avec prise en compte de l'incompressibilité. Identification paramètrique inverse. Application aux élastomères compacts : polychloroprène, mélanges à base de caoutchouc naturel.* Thèse de doctorat, Université de Bretagne Sud.

[O. C. Zienkiewicz, 1989] O. C. Zienkiewicz, R. L. T. (1989). *The Finite Element Mehtod, vol 1 : Basic Formulation and Linear Problems*, volume 1. McGraw-Hill Book Company. ISBN : 0-07-084174-8 (v.1).

[Orgéas, 1997] Orgéas, L. (1997). *Etude expérimentale et numérique du comportement thermo-mécanique d'un alliage à mémoire de forme industriel NiTi*. Thèse de doctorat, Université Joseph Fourier - Grenoble I, http ://www.theses.fr/1997GRE10256.

[Orgéas et Favier, 1995] Orgéas, L. et Favier, D. (1995). Non symmetric tension compression behaviour of niti alloy. *J. Phys. IV*, C8(605-10).

[P. Pegon, 1987] P. Pegon, P. G. (1987). On thermomechanical zaremba schemes of hysteresis. *Res. Mech. Letters*, pages 21–34.

[Pegon, 1988] Pegon, P. (1988). *Contribution à l'étude de l'hystérésis élastoplastique*. Thèse d'état, Institut National Polytechnique, Université Joseph Fourier, Grenoble.

[Pegon *et al.*, 1991] Pegon, P., Guelin, P., Favier, D., Wack, B. et Nowacki, W. (1991). Constitutive scheme of discrete memory form for granular materials. *Archives of Mechanics*, 43(1):3–27.

[Rio, 2015] Rio, G. (2015). Hyperélasticité, rapport interne sur le développement de potentiels hyperélastiques en formulation entraînée. Rapport technique, Université de Bretagne Sud, http ://kleger.univ-ubs.fr/Herezh/projects/herezh/documents.

[RIO, 2017] RIO, G. (2017). Logiciel herezh++, application à la simulation du comportement mécanique de ballons et structures souples. *In Rencontre de Technologies Spatiales : Mécanique déformable des ballons et des structures souples Centre National des Etudes Spatiales - Toulouse (8 juin)*.

[RIO, 2019] RIO, G. (2019). Herezh++, éléments théoriques (theory manual in french), university of "bretagne sud", iddn.fr.010.0106078.000.r.p.2006.035.20600 (http ://kleger.univ-ubs.fr/herezh/projects/herezh/documents).

[Rio *et al.*, 1995a] Rio, G., Favier, D. et Desplats, H. (1995a). Finite element simulation of mechanical behaviour of shape memory alloys coupled with a non-stationary thermal field. *J. Phys. IV*, C8(215-220).

[Rio *et al.*, 2008a] Rio, G., Favier, D. et Liu, Y. (2008a). Elastohysteresis : an accurate phenomenological model for pseudoelasticity and ferroelasticity 3d simulation of shape memory alloys under complex loadings. *In International Conference on Shape Memory and Superelastic Technologies (SMST)*. Stresa, Italy.

[Rio *et al.*, 2009] Rio, G., Favier, D. et Liu, Y. (2009). Elastohysteresis model implemented in the finite element sofware herezh++. *In* Sittner, P., Heller, L. et Paidar, V., éditeurs : *ESOMAT 2009 The 8th European Symposium on Martensitic Transformations*, numéro 08005, DOI :10.1051/esomat/200908005. EDP Sciences (www.esomat.org).

[Rio *et al.*, 2008b] Rio, G., Laurent, H. et Bles, G. (2008b). Asynchronous interface between a finite element commercial software abaqus and an academic research code herezh++. *Advances in Engineering Software*, 39(12):1010–1022. (ISSN 0965-9978).

[Rio *et al.*, 1995b] Rio, G., Manach, P. Y. et Favier, D. (1995b). Finite element simulation of 3d mechanical behaviour of niti shape memory alloys. *Archives of Mechanics*, 47(3):537–556.

[Tourabi *et al.*, 1995] Tourabi, A., Guélin, P. et Favier, D. (1995). Towards modelling of deformable ferromagnets and ferroelectrics. *Archives of Mechanics*, 47(3):237–483.

[Vandenbroucke *et al.*, 2010] Vandenbroucke, A., Laurent, H., Hocine, N. et Rio, G. (2010). A hyperelasto-visco-hysteresis model for an elastomeric behaviour : Experimental and numerical investigations. *Computational Materials Science*, 48(doi :10.1016/j.commatsci.2010.02.012):495–503.

[Wack *et al.*, 1983] Wack, B., Terriez, J. M. et Guelin, P. (1983). A hereditary type, discrete memory, constitutive equation with applications to simple geometries. *acta mechanica*, 30:9–37.

[ZHAI, 1996] ZHAI, W.-M. (1996). Two simple fast integration methods for large-scale dynamic problems in engineering. *International Journal for Numerical Methods in Engineering*, 39(24):4199–4214.

[Zrida *et al.*, 2009] Zrida, M., Laurent, H., Rio, G., Masmoudi, N. et Bradai, C. (2009). Identification d'un modèle de comportement d'hyper-visco-hystérésis pour un matériau polypropylène. *In 22ème congrès sur la déformation des polymères solides DEPOS22, La Colle sur Loup, Alpes-Maritimes, France.*