Le générateur de code `mfront` : écriture de lois de comportement mécanique

T. Helfer, É. Castelier, V. Blanc, J. Julien

Décembre 2013

RÉSUMÉ

L'une des missions de la plate-forme `pleiades` est de capitaliser les connaissances matériau utilisées pour la simulation des éléments combustibles et absorbants.

Cette note s'intéresse aux lois de comportement mécanique et plus particulièrement à leur écriture à l'aide du générateur de code `mfront`. Elle complète la présentation générale de `mfront` qui traite des propriétés matériau et des modèles physico-chimiques [Helfer 13d]. Dans le cadre de la plate-forme `pleiades`, ces lois de comportement peuvent être stockées dans la base de données `sirius`.

La variété des phénomènes traités fait que `mfront` propose différentes façons d'écrire des lois de comportement mécaniques.

Certaines sont spécifiques à des lois de comportement qui sont d'usage courant et pour lesquelles des algorithmes d'intégration performants existent (plasticité et viscoplasticité isotropes notamment).

Pour les lois les plus complexes, `mfront` permet d'utiliser des méthodes de RUNGE-KUTTA ou des méthodes implicites.

SOMMAIRE

1 INTRODUCTION 6

1.1 ANALYSEURS DISPONIBLES 7

1.2 PLAN DE LA NOTE 7

2 GÉNÉRALITÉS 8

2.1 QUELQUES DÉFINITIONS 8

2.2 CONSEILS SUR LE CHOIX DE L'ANALYSEUR À UTILISER 9

2.3 POINTS PARTICULIERS 10

2.3.1 Hypothèses de modélisation 10

2.3.2 Traitement des matériaux orthotropes 11

2.3.3 Traitement des dilatations libres 11

2.3.4 Gestion des bornes 13

2.4 INTERFACE AUX LOIS DE COMPORTEMENT 13

2.4.1 Rôle des interfaces aux lois de comportement 13

3 MATÉRIAUX ISOTROPES À ÉCOULEMENT PLASTIQUE OU VISCOPLASTIQUE INCOMPRESSIBLE 15

3.1 GÉNÉRALITÉS ET RÉSOLUTION IMPLICITE 15

3.1.1 Expression des lois de comportement 15

3.1.2 Méthode d'intégration numérique 18

3.1.3 Matrice tangente cohérente 21

3.1.4 Expression de la matrice tangente cohérente pour un mécanisme unique 23

3.1.5 Matrice tangente 24

3.1.6 Expression de la matrice tangente pour un mécanisme unique 24

3.2 UTILISATION DES ANALYSEURS SPÉCIFIQUES 24

3.2.1 La directive `@FlowRule` 25

3.2.2 Mise à jour des variables auxiliaires 25

3.2.3 Paramètres numériques automatiquement définis 26

3.2.4 Noms réservés 26

3.2.5 Comportement viscoplastique du SiC 26

4 INTÉGRATION DES LOIS DE COMPORTEMENT, INTÉGRATEUR PAR DÉFAUT 29

4.1 EXEMPLE DE L'ÉLASTICITÉ ORTHOTROPE 29

4.2 NOMS DES VARIABLES ET DES MÉTHODES C++ UTILISÉS EN INTERNE 29

4.3 LA DIRECTIVE @INTEGRATOR 29

5 INTÉGRATION DES LOIS DE COMPORTEMENT PAR UNE MÉTHODE EXPLICITE (ALGORITHME DE RUNGE-KUTTA) 31

5.1 LES ALGORITHMES DE RUNGE-KUTTA 31

5.2 ALGORITHMES DISPONIBLES 32

5.2.1 Algorithmes sans contrôle du pas de temps *32*
5.2.2 Algorithmes avec contrôle du pas de temps (correcteur/prédicteur) *33*
5.2.3 Gestion du pas de temps *34*
5.2.4 Critère d'arrêt *36*
5.3 LA DIRECTIVE @COMPUTESTRESS 36
5.4 LA DIRECTIVE @DERIVATIVE 36
5.5 MISE À JOUR DES VARIABLES AUXILIAIRES 37
5.6 PARAMÈTRES AUTOMATIQUEMENT DÉFINIS 37
5.7 NOTES SUR L'UTILISATION DES TABLEAUX DE VARIABLES INTERNES 37
5.8 INTÉGRATION D'UNE LOI D'ÉCOULEMENT VISCOPLASTIQUE ORTHOTROPE PAR UN ALGORITHME DE RUNGE-KUTTA 38
6 INTÉGRATION DES LOIS DE COMPORTEMENT PAR UNE MÉTHODE IMPLICITE 40
6.1 RÉSOLUTION D'UN SYSTÈME DIFFÉRENTIEL PAR UNE MÉTHODE IMPLICITE 40
6.1.1 Généralités *40*
6.2 MÉTHODE IMPLICITE UTILISÉE DANS MFRONT 42
6.2.1 Initialisation *43*
6.2.2 Critère d'arrêt *44*
6.2.3 Avantages et inconvénients des méthodes implicites *44*
6.3 MÉTHODE DE NEWTON-RAPHSON 45
6.4 MÉTHODES ALTERNATIVES 47
6.4.1 Les algorithmes de BROYDEN *47*
6.5 AMÉLIORATION DE LA ROBUSTESSE DES ALGORITHMES IMPLICITES 48
6.5.1 Méthode d'accélération utilisé par les algorithmes de résolution globaux de Cast3M . . *48*
6.5.2 Méthode de relaxation *49*
6.6 CALCUL DE LA MATRICE TANGENTE COHÉRENTE 49
6.6.1 Une façon générique de calculer de la matrice tangente cohérente *49*
6.6.2 Une méthode alternative de calcul de la matrice tangente cohérente *51*
6.7 LA DIRECTIVE @COMPUTESTRESS 52
6.8 LA DIRECTIVE @INTEGRATOR 52
6.9 MISE À JOUR DES VARIABLES AUXILIAIRES 53
6.10 ORDRE D'ÉVALUATION DES BLOCS DÉFINIS PAR L'UTILISATEUR 53
6.11 PARAMÈTRES AUTOMATIQUEMENT DÉFINIS 53
6.12 INTÉGRATION D'UNE LOI D'ÉCOULEMENT VISCOPLASTIQUE ORTHOTROPE PAR UN ALGORITHME IMPLICITE 53
7 CONCLUSIONS 57
7.1 UNE SOLUTION FLEXIBLE, MATURE ET PERFORMANTE 57
7.2 PERSPECTIVES 59

7.3 Développement 59
Références 60
Liste des tableaux 62
Liste des figures 63
Annexe A Résolution d'un problème mécanique quasi-statique par la méthode des éléments finis 64
Annexe A.1 Discrétisation 64
Annexe A.2 Principe des travaux virtuels 64
Annexe A.3 Principe de la méthode de Newton-Raphson 65
Annexe B Principales opérations tensorielles 67
Annexe B.1 Conventions de représentation des tenseurs 67
Annexe B.2 Opérations sur les tenseurs d'ordre 2 67
Annexe B.3 Opérations sur les tenseurs d'ordre 4 67
Annexe C Fonctions utiles 69
Annexe C.1 Manipulation des tenseurs d'ordre 2 symétriques 69
Annexe C.2 Calcul des coefficents de Lamé 69
Annexe C.3 Calcul du tenseur de Hill 70
Annexe D Description de l'algorithme Runge-Kutta-Cast3m 71
Annexe E Rappels sur les lois de comportement orthotropes et application aux tubes . 74
Annexe E.1 Directions d'orthotropie 74
Annexe E.2 Élasticité 74
Annexe E.3 Critère de Hill 75
Annexe E.4 Conventions de rangement des directions d'orthotropie pour les tubes en fonction de l'hypothèse de modélisation 76
E.4.1 Traitement des problèmes $1D$, $2D\,(r,z)$ et $3D$ 77
E.4.2 Traitement des problèmes plans non axisymétriques 78
E.4.3 Cas particulier 78
Annexe F Comparaisons numériques de différents algorithmes de résolution 79
Annexe F.1 Loi utilisée pour la comparaison 79
Annexe F.2 Résultats obtenus 79
Annexe G Noms des variables et des méthodes C++ utilisés en interne par les analyseurs de lois de comportement 82
Annexe G.1 Noms communs à tous les analyseurs de lois de comportement 82
Annexe G.2 Noms de variables réservés par les analyseurs spécifiques 83
Annexe G.3 Noms de variables réservés par l'analyseur Runge-Kutta 83
Annexe G.4 Noms de variables réservés par l'analyseur implicite 83
Index des fichiers d'entête fournis par la librairie tfel 85

INDEX DES CLASSES, DES MÉTHODES ET DES FONCTIONS FOURNIES PAR LA LIBRAIRIE TFEL 86
INDEX DES DIRECTIVES 87

1 INTRODUCTION

Ce document décrit comment écrire des lois de comportement mécanique avec le générateur de code `mfront`. `mfront` vise à garantir une gestion pérenne, robuste, efficace et évolutive des connaissances matériau dans la plate-forme `pleiades` [Michel 09, Helfer 11]. Il permet également à des utilisateurs non développeurs d'écrire leurs propres connaissances matériau [Helfer 10]. Cette note intègre des éléments d'une note antérieure décrivant un algorithme numérique particulier [Blanc 11].

Elle complète la présentation générale de `mfront` [Helfer 13d]. Deux autres notes traitent[1] :
- de l'interface `umat` utilisée par le codes aux éléments finis `Cast3M` [Helfer 13c] ;
- de l'interface utilisée par le codes aux éléments finis `Aster` [Helfer 13b] ;

Ces notes sont intégrées à la gestion de configuration de `tfel` et évoluent continûment avec les développements de `tfel`[2]. Le présent document a été généré à partir de la révision b62c3710a.

La lecture de ce document suppose que le lecteur est déjà familier de `mfront`. A minima, la lecture de la présentation générale de `mfront` semble nécessaire [Helfer 13d].

De multiples phénomènes Les matériaux solides réagissent aux sollicitations mécaniques par différents phénomènes : élasticité, viscoplasticité, plasticité, endommagement. Nous renvoyons aux ouvrages classiques pour la description physique de ces phénomènes [François 95, Chaboche 09, Besson 01].

Rôle de la loi de comportement Nous décrivons en annexe A un algorithme simplifié de recherche de l'équilibre mécanique statique non-linéaire qui précise la place de la loi de comportement mécanique. Le lecteur intéressé pourra se reporter aux documentations des codes éléments finis pour une description plus précise [Pascal 05, Abbas 13, Besson 01].

En résumé, connaissant l'état mécanique du matériau à un instant t, les lois de comportement doivent, en réponse à un incrément de sollicitation mécanique représentée par un incrément de déformations totales $\Delta\underline{\epsilon}^{to}$, fournir, en chaque point d'intégration :
- l'évolution microstructurelle du matériau, décrit par un ensemble de variables internes y_i sur le pas de temps Δt ;
- la valeur de la contrainte en fin de pas de temps (ou de manière équivalente l'incrément de contraintes $\Delta\underline{\sigma}$).

Si l'algorithme de recherche de l'équilibre mécanique le nécessite, la loi de comportement peut également fournir :
- la matrice tangente cohérente $\dfrac{\partial\Delta\underline{\sigma}}{\partial\Delta\underline{\epsilon}^{to}}$ ou une estimation de celle-ci.
- une matrice de prédiction en début de pas de temps permettant de fournir une première estimation de la solution en fin de pas de temps, avant de débuter la résolution.

Dans certains cas, les lois de comportement peuvent également donner des indications sur la qualité de la discrétisation temporelle du problème [Proix 11b].

1. Citons également qu'un guide de référence de la librairie `tfel` est également en cours de rédaction. Cette note étant assez ambitieuse, sa rédaction s'étalera dans le temps. Elle est cependant disponible en version projet dans les sources de la librairie. Elle contient en particulier le guide d'installation de la librairie.
2. La version la plus à jour est obtenue dans le répertoire de compilation des sources par la commande :

```
make doc-pdf
```

1.1 ANALYSEURS DISPONIBLES

L'écriture de lois de comportement nécessite de prendre à la fois en compte les mécanismes physiques décrits (certains algorithmes sont plus adaptés à certains phénomènes) et des besoins des codes éléments finis (matrice tangente cohérente). La variété des situations rencontrées en pratique expliquent la variété des analyseurs proposés par `mfront`, qui sont actuellement au nombre de 8[3]. La description de ces analyseurs est l'objet de ce document.

Plusieurs analyseurs dédiés aux lois de comportement sont actuellement disponibles :

- l'analyseur `DefaultParser` qui permet de traiter tous types de lois de comportements ;
- l'analyseur `IsotropicMisesCreep` qui gère exclusivement les lois de comportement mécanique viscoplastique incompressible sans écrouissage des matériaux isotropes ;
- l'analyseur `IsotropicStrainHardeningMisesCreep` qui gère exclusivement les lois de comportement mécanique viscoplastique incompressible avec écrouissage des matériaux isotropes ;
- l'analyseur `IsotropicPlasticMisesFlow` qui gère exclusivement les lois de comportement mécanique plastique incompressible des matériaux isotropes ;
- l'analyseur `MultipleIsotropicMisesFlows` qui gère une combinaison arbitraire d'écoulements des trois types précédents. Les différents écoulements sont supposés non couplés ;
- les analyseurs `Implicit` et `ImplicitII` qui simplifient la résolution d'une loi de comportement mécanique quelconque à l'aide d'une intégration implicite.

Ces différents analyseurs sont décrits dans les sections suivantes.

1.2 PLAN DE LA NOTE

La section 2 décrit des généralités sur les lois de comportements mécaniques et introduit certaines notions nécessaires à la suite.

Les analyseurs `IsotropicMisesCreep`, `IsotropicStrainHardeningMisesCreep`, `IsotropicPlasticMisesFlow` et `MultipleIsotropicMisesFlows` sont très proches et sont décrits en section 3.

Les analyseurs `DefaultParser`, `RungeKutta` et `Implicit` sont dits *génériques* car ils permettent de traiter n'importe quelle loi de comportement. Ils sont respectivement décrits dans les sections 4, 5 et 6.

3. La notion d'analyseur est décrite plus en détails dans la première partie de la documentation de `mfront` [Helfer 13d].

2 GÉNÉRALITÉS

Nous abordons dans cette section des points qui sont utiles pour la lecture de la suite. Nous commençons par préciser les définitions utilisées dans la suite. Nous donnons ensuite quelques conseils sur le choix de l'analyseur à utiliser. Nous traitons quelques points qui sont indépendants de l'analyseur utilisé. Enfin, nous précisons le rôle des interfaces aux lois de comportement.

2.1 QUELQUES DÉFINITIONS

Les lois de comportement peuvent être complexes. Pour les décrire, il nous faut introduire quelques définitions.

Propriétés matériau Afin de pouvoir adapter des lois de comportements à différents matériaux, celles-ci peuvent utiliser des propriétés matériau, qui sont définies, dans `mfront`, comme des fonctions des valeurs actuelles des variables d'état du matériau [Helfer 13d].

Les lois de comportement mécaniques peuvent :

- demander à ce qu'un certain nombre de propriétés matériau leur soient fournies par le code appelant [4]. Dans certains cas, le code `Cast3M` notamment [Helfer 13c], le code appelant impose que la loi de comportement utilise des propriétés matériau prédéfinies.
- utiliser des lois définies dans des fichiers `mfront` dédiés [5].

Les variables internes Les variables internes décrivent l'état mécanique local du matériau.

Pour `mfront`, les variables internes peuvent être soit des *scalaires* soit des *tenseurs* d'ordre 2 symétriques.

Certains analyseurs, dédiés à des lois de comportement spécifiques, ne permettent pas de déclarer de nouvelles variables internes.

Les variables internes auxiliaires Les variables internes auxiliaires désignent des variables internes qui ne sont pas nécessaires pour l'intégration de la loi de comportement. Ces variables ont des utilités diverses :

- elles peuvent désigner des variables qui peuvent être éliminées de l'intégration ;
- des variables uniquement destinées aux posttraitements.

Les variables internes auxiliaires sont mises à jour après l'intégration des variables internes. Elles peuvent être soit des *scalaires* soit des *tenseurs*.

Les variables externes Les variables externes désignent des variables dont l'évolution est donnée par ailleurs et connue sur le pas de temps. Ces variables peuvent ou être des variables d'état du matériau ou des paramètres externes (flux de neutrons, fluence, densité de fissions).

Parmi les variables externes, nous pouvons citer la température, la déformation totale du matériau (qui représente la sollicitation locale et dont la valeur est donnée par la résolution de l'équilibre global du matériau). Ces deux variables sont traitées de manière particulière par `mfront` et déclarées automatiquement.

Les variables locales Les variables locales permettent généralement de calculer des variables avant de débuter l'intégration afin d'éviter des calculs superflus.

4. Voir le mot clé `@MaterialProperty`.
5. Voir le mot clé `@MaterialLaw`.

Une utilisation typique de variable locale est de calculer avant l'intégration des termes de type ARRHENIUS (termes de la forme $\exp\left(-\frac{Q}{RT}\right)$) afin de ne pas les réévaluer au cours des itérations. Ces termes sont souvent évalués en milieu de pas de temps, ce qui est cohérent avec une intégration par une θ-méthode, et une approximation généralement suffisante pour les autres méthodes d'intégration.

Il n'y a pas de limite sur le type des variables locales.

La directive `@InitLocalVariables`[6] permet d'initialiser ces variables locales.

Tableaux de variables internes, de propriétés matériau et de variables externes Afin de pouvoir regrouper des équations dont le *formalisme* était similaire, nous avons introduit la possibilité de définir des tableaux de variables internes, de propriétés matériaux et de variables externes. Il faut noter que la taille de ces tableaux, c'est à dire le nombre de variables internes, de propriétés matériau et/ou de variables externes est fixés « en dur » dans le fichier d'entrée.

Il est alors possible d'écrire des boucles sur les variables internes constituant le tableau pour condenser l'écriture des lois.

La possibilité d'utiliser des tableaux de variables internes est particulièrement utile en homogénéisation. Les lois de comportement homogénéisées peuvent avoir un grand nombre de variables internes qui partagent des lois d'évolutions similaires. Un exemple de cela est donné par les lois issues de l'homogénéisation de poly-cristaux qui peuvent conduire à plusieurs milliers de variables internes [Proix 13a]. Grâce aux tableaux de variables internes, l'implantation de ce type de lois peut être très courte (une centaine de lignes).

2.2 CONSEILS SUR LE CHOIX DE L'ANALYSEUR À UTILISER

Pour pouvoir traiter tous les types de lois de comportement, différents analyseurs sont disponibles. Les analyseurs propres à des formes de lois de comportement sont dits spécifiques. Par opposition, les autres sont dits génériques.

Nous pouvons donner quelques conseils généraux sur le choix de l'analyseur à utiliser :

- si un intégrateur spécifique existe, il vaut mieux l'utiliser : il utilise un algorithme optimisé et robuste et le nombre d'informations à fournir est réduit.
- si l'on doit recourir à un intégrateur générique, il vaut mieux préférer l'intégration implicite, surtout s'il s'agit de lois indépendantes du temps (plasticité, endommagement) :
  - l'équation différentielle pour la plasticité ou l'endommagement doit être remplacée par la nullité du critère en fin de pas de temps ;
  - les temps de calculs sont souvent *très* avantageux ;
- il ne faut utiliser l'analyseur RUNGE-KUTTA que :
  - si *vraiment rien d'autre* n'est possible (impossibilité de calculer la jacobienne)
  - si le temps de développement est limité ;
  - si le temps d'intégration de la loi de comportement ne pose pas de problème de performance ;
  - si le nombre de variables internes est très grand (loi de comportement issus de l'homogénéisation de poly-cristaux [Proix 13a] notamment).

Insistons sur le fait que l'utilisation de l'analyseur RUNGE-KUTTA est fortement déconseillée. De nombreux développements ont été faits pour rendre l'utilisation des méthodes implicites plus simples (calcul automatique de la jacobienne par différentiation numérique, algorithmes de BROYDEN) et si l'expression de la loi reste encore un peu plus complexe, les gains en performance valent largement l'effort supplémentaire.

L'analyseur `DefaultParser` n'est utile que dans ces cas très particuliers, par exemple pour des lois dépendant explicitement des déformations (loi de MAZARS par exemple [**?**]).

6. La directive `@InitLocalVars` est synonyme de la directive `@InitLocalVariables`.

2.3 POINTS PARTICULIERS

2.3.1 Hypothèses de modélisation

Les lois de comportement sont implantées par des classes `template` paramétrée par l'hypothèse de modélisation. Ce choix permet de :

— proposer une implantation optimisée et fiable de la loi de comportement pour chacune des hypothèses de modélisation ;
— traiter les cas particuliers, les contraintes planes notamment ;
— de spécifier quelles sont les hypothèses valides. Par exemple, les lois issues de l'homogénéisation de poly-cristaux doivent nécessairement être intégrées en $3D$ [Proix 13a]

Le fichier d'entête `TFEL/MaterialLaw/ModellingHypothesis.hxx` définit une structure `ModellingHypothesis`. Dans cette structure, un objet de type énumération nommé `Hypothesis` définit les différentes hypothèses de modélisation ajourd'hui supportées :

— `AXISYMETRICALGENERALISEDPLANESTRAIN` qui désigne une modélisation $1D$ axisymétrique plan généralisée. Dans ce cas, le tenseur des contraintes a 3 composantes et est représenté par le vecteur suivant :

$$\underline{\sigma} = \begin{pmatrix} \sigma_{rr} \\ \sigma_{zz} \\ \sigma_{\theta\theta} \end{pmatrix}$$

— `AXISYMETRICAL` qui désigne une modélisation $2D$ axisymétrique. Dans ce cas, le tenseur des contraintes a 4 composantes et est représenté par le vecteur suivant :

$$\underline{\sigma} = \begin{pmatrix} \sigma_{rr} \\ \sigma_{zz} \\ \sigma_{\theta\theta} \\ \sqrt{2}\sigma_{rz} \end{pmatrix}$$

— `PLANESTRESS`, `PLANESTRAIN` et `GENERALISEDPLANESTRAIN` qui désigne différentes modélisations $2D$ qui se distinguent par le traitement de la direction axiale. Dans ces cas, le tenseur des contraintes a 4 composantes et est représenté par le vecteur suivant :

$$\underline{\sigma} = \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sqrt{2}\sigma_{xy} \end{pmatrix}$$

— `TRIDIMENSIONAL` qui désigne la modélisation la plus générale. Dans ce cas, le tenseur des contraintes a 6 composantes et est représenté par le vecteur suivant :

$$\underline{\sigma} = \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sqrt{2}\sigma_{xy} \\ \sqrt{2}\sigma_{xz} \\ \sqrt{2}\sigma_{yz} \end{pmatrix}$$

Les directives `@ModellingHypothesis` et `@ModellingHypotheses` Les directives `@ModellingHypothesis` et `@ModellingHypotheses` permettent de spécifier les hypothèses de modélisation valides. La première attend le nom d'une hypothèse, la seconde un tableau contenant une ou plusieurs hypothèses.

Les noms d'hypothèses valides sont donc les suivants :

— AxisymmetricalGeneralisedPlaneStrain ;

— Axisymmetrical ;
— PlaneStress ;
— PlaneStrain ;
— GeneralisedPlaneStrain ;
— Tridimensional.

Cas des contraintes planes Le cas des contraintes planes appelle certaines remarques. Le plus souvent, les lois de comportements ne peuvent être utilisées telles quelles en contraintes planes : une implantation spécifique doit être faite. Pour cette raison, les contraintes planes sont exclues des hypothèses de modélisation supportées par défaut.

Plusieurs stratégies peuvent être mises en place pour éviter d'avoir à écrire une implantation spécifique :
— le code aux éléments finis `Zebulon` introduit des éléments finis spécifiques, possédant un degré de liberté supplémentaire (la déformation axiale) dont la force associée (liée à la contrainte axiale) est nulle [Besson 98] ;
— le code aux éléments finis `Aster` modifie l'algorithme de résolution global pour converger vers une solution en contraintes planes [Proix 12d].

Pour le code aux éléments finis `Cast3M`, si une implantation en contraintes planes n'est pas disponible, l'interface `umat` fournie par `mfront` gère les contraintes planes en introduisant une variable interne supplémentaire (la déformation axiale) qu'elle ajuste (par appels successifs à la loi de comportement $2D$) pour trouver une contrainte axiale nulle [Helfer 13c].

2.3.2 Traitement des matériaux orthotropes

Ce paragraphe décrit des fonctionnalités qui ne sont pas accessibles aux analyseurs spécifiques (dédiés à des lois isotropes).

Écrire les lois de comportement orthotrope de manière indépendante de l'hypothèse de modélisation est rendue difficile par certaines restrictions imposées par les codes aux éléments finis `Cast3M` et `Aster` :
— il est nécessaire de gérer la rotation des contraintes dans le repère propre au matériau.
— il n'est pas possible de choisir de manière cohérente une convention sur l'ordonnancement des axes valable quelle que soit la dimension. Le cas des tubes est traité en détail dans l'annexe E.

2.3.3 Traitement des dilatations libres

`mfront` ne traite en général pas des dilatations libres, et plus particulièrement ne traite en général pas des dilatations thermiques :

- en petites transformations, Dans la plupart des codes aux éléments finis [CEA 13, EDF 13], les dilatations libres sont généralement traitées en amont de l'intégration de la loi de comportement. Cela se traduit par le fait que la déformation totale passée à la loi de comportement est en fait la déformation mécanique totale, c'est à dire la déformation totale à laquelle ont été retranchées toutes les dilatations libres.
- en grandes transformations, le traitement des dilatations libres dépend du formalisme choisi et en cadre général ne peut être fourni.

Il peut cependant être intéressant de traiter certains gonflements dans la loi de comportement. Donnons quatres exemples pratiques :

- certaines lois de comportement viscoplastique de gaine de réacteurs refroidis au sodium relient l'intensité de l'écoulement viscoplastique à la vitesse de gonflement ;
- certaines modélisations plastiques des aciers considèrent la température comme une variable interne : l'évolution de la température, supposée adiabatique le temps de l'expérience, est supposée due à la dissipation mécanique locale. Il peut paraître intéressant de coupler finement mécanique et thermique ;

- il peut être nécessaire de traiter des changements de phases au cours du calcul mécanique, en particulier la transition martensite-austénite de certains aciers qui dépend de l'état de contraintes [Milliard 14] ;
- certaine stratégies « grandes transformations » permettent d'utiliser un formalisme de lois en petites déformations, voir d'utiliser les lois identifiées en petites déformations (voir [Helfer e]). Dans ce cas, la dissymétrie de traitement par les codes appelant (notée plus haut) entre les petites déformations (traitement des dilatations libres) et les grandes déformations (non traitement des dilatations libres) apparaît d'autant plus gênante que l'expression usuelle des dilatations libres (voir équation (1) pour la dilatation thermique), convenablement post-traitée, peut être réutilisée.

Cas des lois en petites transformations, cas général En petites transformations, une solution simple pour prendre en compte une dilatation libre, est de calculer un incrément de dilatation et de le soustraire à l'incrément de dilatation totale dans la partie `@InitLocalVariables`. Cette stratégie n'est valide que dans le cas où la dilatation libre ne dépend pas du résultat de la mécanique

Il est possible de garder une trace de cette dilatation en définissant une variable auxiliaire associée. Il est cependant nécessaire de prendre garde à la manière dont sont mises à jour les variables auxiliaires par l'analyseur utilisé.

Cas des lois en petites transformations, dilatation thermique La dilatation thermique est un cas particulier par son coté systématique : il suffit de savoir calculer le coefficient de dilatation moyen $\alpha\left(T\right)$ que l'on suppose ne dépendre que de la température actuelle T. La dilatation thermique linéique s'écrit alors (voir [Helfer e]) :

$$\frac{\Delta\, l}{l^i}\left(T\right) = \frac{1}{1+\alpha\left(T^i\right)\left(T^i-T^\alpha\right)}\left[\alpha\left(T\right)\left(T-T^\alpha\right)-\alpha\left(T^i\right)\left(T^i-T^\alpha\right)\right] \tag{1}$$

où :
- T^α est la température de référence pour l'expérience de dilatométrie ayant servie à identifier le coefficient de dilatation thermique et qui correspond à une dilatation nulle ;
- T^i est la température à laquelle la géométrie du corps a été mesurée (température de début de calcul) ;
- $\Delta\, l$ est la variation de longueur par rapport à la température T^i.

La directive `@ComputeThermalExpansion` permet le calcul de la dilatation thermique par la formule (1) en amont de l'intégration de la loi de comportement.

Elle prend un nom de fichier `mfront` argument dans le cas isotrope, ou un tableau de trois noms de fichiers `mfront` dans le cas orthotrope (une dilatation pour chaque direction principale). Ces fichiers doivent décrire le coefficient de dilatation thermique moyen comme une propriété matériau.

La température T^α est supposée donnée dans le fichier définissant la dilatation par la définition d'une constante nommée `ReferenceTemperature` de la directive `@Constant`. Si cette température n'est pas fournie, `mfront` prendra la température ambiante ($293,15\,K$).

La température T^i est automatiquement déclarée comme un paramètre nommé `referenceTemperatureForThermalExpansion`[7]. Par défaut, sa valeur est de $293,15\,K$ (température ambiante).

Cas particuliers Si le formalisme « grandes transformations » choisi se base sur une écriture des lois écrites dans le formalisme des petites transformations (utilisation d'une hypothèse de « petites déformations, grandes rotations », déformations logarithmiques), il est souvent possible de proposer un traitement des dilatations libres systématique (voir [Helfer e]).

7. La notion de paramètres est décrite dans la notice générale de `mfront` [Helfer 13d].

Implantation Pour permettre aux interfaces de proposer des formalismes « grandes transformations » réutilisant le formalisme des « petites transformation », le calcul de la dilatation doit être fait en amont de la loi.

Il revient aux interfaces de modifier le chargement du point matériel de manière adéquate. Par exemple, en petites transformations, la déformation totale envoyée à la loi de comportement sera la déformation totale fournie par le code appelant à laquelle on aura retiré la dilatation.

2.3.4 Gestion des bornes

Deux types de bornes sont distinguées dans `mfront` :

- les bornes physiques[8], qui désignent les plages de valeurs acceptables pour une variable donnée. Par exemple, une température (en Kelvin) ne peut être négative, une porosité est positive et inférieure à 1 ;
- les bornes de validité[9], qui désignent les plages de valeurs sur lesquelles la loi de comportement a été identifiée.

Des bornes peuvent être posées sur :

- les valeurs des variables internes (en début et en fin de pas).
- les valeurs des contraintes.
- les valeurs des variables externes.

Dépassement des bornes de validité Le traitement d'une violation d'une borne de validité dépend de l'utilisateur qui peut choisir parmi trois « politiques » :

- ne rien faire ;
- afficher un message d'avertissement ;
- arrêter le calcul.

La façon de préciser cette politique dépend du code cible (voir paragraphe 2.4).

2.4 INTERFACE AUX LOIS DE COMPORTEMENT

Cette section décrit tout d'abord le rôle des interfaces puis chacune des deux interfaces disponibles actuellement :

- l'interface `umat` utilisée pour l'adhérence au code aux éléments finis `Cast3M` [CEA 13, Helfer 13c]. Cette interface est également utilisée par le code d'homogénéisation par transformées de FOURIER rapides `TMFFT` [Castelier 09, **?**] ;
- l'interface `Aster` utilisée pour l'adhérence au code aux éléments finis `Aster` [EDF 13, Helfer 13b].

2.4.1 Rôle des interfaces aux lois de comportement

Les interfaces aux lois de comportement ont différentes fonctions :

- les lois de comportement choisissent l'une des implantations de la loi de comportement en fonction de l'hypothèse de modélisation[10] ;
- assurer la conversion entre la convention utilisée par le code appelant pour représenter les tenseurs et la convention utilisée dans `tfel` ;
- l'interface doit gérer les lois de comportement orthotropes. Par exemple, l'interface peut assurer la rotation des déformations totales et de leurs incréments dans le repère propre du matériau avant l'appel à la loi de comportement proprement dite et la rotation des contraintes dans le repère général après l'appel à la loi de comportement. ;

8. Voir le mot clé `@PhysicalBounds`
9. Voir le mot clé `@Bounds`
10. Les lois de comportements générées par `mfront` sont représentées par des classes `template` paramétrées par l'hypothèse de modélisation.

— l'interface doit fournir certains éléments nécessaires au calcul (matrice d'élasticité) à partir des informations fournies par le code ;
— l'interface doit capter les exceptions C++ et les traduire en message d'erreurs adaptés.

En fonction des fonctionnalités disponibles ou absentes du code cible, l'interface peut également :

— assurer le sous-découpage local (au niveau du point de GAUSS) du pas de temps en cas de non convergence ;
— gérer le cas des contraintes planes si les lois de comportement ne gèrent pas cette hypothèse, ce qui est le cas de la plupart des lois générées par `mfront` ;
— permettre l'utilisation des lois écrites pour les petites déformations dans un calcul en transformations finies. Pour cela, plusieurs pistes semblent intéressantes :
  — le cas des grandes rotations, petites déformations ;
  — l'utilisation des déformations logarithmiques ;

De manière optionnelle, il peut être utile de générer des exemples d'utilisation de la loi de comportement traitée. Ainsi, l'interface `umat` génère automatiquement un exemple de mise en données `gibiane` [Helfer 13c].

3 MATÉRIAUX ISOTROPES À ÉCOULEMENT PLASTIQUE OU VISCOPLASTIQUE INCOMPRESSIBLE

Nous nous intéressons dans cette section à une famille particulière de lois de comportement, très utilisée en mécanique, et qui représente la majorité des comportements de la plate-forme `pleiades`. Ces lois s'appliquent à des matériaux *isotropes* et décrivent un comportement *plastique* ou *viscoplastique*, avec ou sans *écrouissage*, dont les déformations résiduelles sont *isochores*. Ces particularités permettent d'optimiser les techniques d'intégration numérique.

Après une présentation des techniques d'intégration adaptées à cette famille de lois, nous détaillons la syntaxe des quatre analyseurs `mfront` qui leurs sont dédiés.

3.1 GÉNÉRALITÉS ET RÉSOLUTION IMPLICITE

Après une description des particularités des lois de comportement mécanique traitées dans cette section, les techniques d'intégration qui leurs sont appliquées dans `mfront` sont exposées. Celles-ci s'appuient sur une résolution implicite, une θ-méthode semblable à celles qui sont décrites en section 6.

Elles permettent également le calcul de la matrice tangente cohérente.

3.1.1 Expression des lois de comportement

Les lois de comportement traitées dans cette section se composent d'une partie élastique, et de plusieurs mécanismes d'écoulement plastique ou viscoplastique.

Partition des déformations Cette combinaison se traduit par la *partition des déformations* : la déformation totale $\underline{\epsilon}^{to}$ est la somme d'une déformation élastique $\underline{\epsilon}^{el}$ et d'une déformation inélastique $\underline{\epsilon}^{an}$:

(2a) $$\underline{\epsilon}^{to} = \underline{\epsilon}^{el} + \underline{\epsilon}^{an}.$$

Cette dernière se décompose à son tour en plusieurs déformations relatives aux différents mécanismes d'écoulement, indicés par i, et supposés indépendants :

(2b) $$\underline{\epsilon}^{an} = \sum_i \underline{\epsilon}_i^{an}.$$

Isotropie Les matériaux décrit ici sont supposés *isotropes*. Cette hypothèse sera utilisée pour formuler l'ensemble des mécanismes : élasticité, écoulements plastiques ou viscoplastiques.

Tenseurs Pour simplifier les expressions à venir, il est commode d'introduire des notations tensorielles, détaillées en annexe B :

- le produit tensoriel de deux tenseurs a, b : $a \otimes b$;
- le produit contracté de deux tenseurs a, b : $a : b$;
- le tenseur identité d'ordre 2 : $\underline{\mathbf{I}}$;
- le tenseur identité d'ordre 4 : $\underline{\underline{\mathbf{I}}}$.

Comportement élastique Les contraintes $\underline{\sigma}$ se déduisent des déformations élastiques $\underline{\epsilon}^{el}$ par la loi de HOOKE. Pour un matériau isotrope, cette relation peut s'écrire :

(3a) $$\underline{\sigma} = \lambda \,\mathrm{tr}\, \underline{\epsilon}^{el} \underline{\underline{\mathbf{I}}} + 2\,\mu\, \underline{\epsilon}^{el}$$

où tr $\underline{\epsilon}^{el}$ désigne la trace du tenseur $\underline{\epsilon}^{el}$ (somme des termes diagonaux). les coefficients de LAMÉ λ et μ du matériau se déduisent du module d'YOUNG et du coefficient de POISSON. Sous forme tensorielle cette loi s'écrit de manière plus compacte :

(3b) $$\underline{\sigma} = \underline{\underline{\mathbf{D}}} : \underline{\epsilon}^{el}, \quad \text{avec} \quad \underline{\underline{\mathbf{D}}} = \lambda\, \underline{\mathbf{I}} \otimes \underline{\mathbf{I}} + 2\,\mu\, \underline{\underline{\mathbf{I}}}.$$

La loi est alors résumée par le tenseur élastique $\underline{\underline{\mathbf{D}}}$.

Écoulements Les écoulements i, plastiques ou viscoplastiques, sont les mécanismes qui créent les déformations inélastiques $\underline{\epsilon}_i^{an}$ de la partition (2). Pour les lois décrites ici, la direction d'écoulement est supposée proportionnelle au déviateur des contraintes :

(4) $$\dot{\underline{\epsilon}}_i^{an} \propto \underline{\mathbf{s}},$$

défini par :

(5) $$\underline{\mathbf{s}} = \underline{\sigma} - \frac{1}{3}\,\mathrm{tr}\,\underline{\sigma}\,\underline{\mathbf{I}} = \underline{\underline{\mathbf{K}}} : \underline{\sigma}, \quad \text{avec} \quad \underline{\underline{\mathbf{K}}} = \underline{\underline{\mathbf{I}}} - \frac{1}{3}\underline{\mathbf{I}} \otimes \underline{\mathbf{I}}.$$

Avec cette hypothèse, les écoulements modélisés vérifient :

$$\mathrm{tr}\, \underline{\epsilon}_i^{an} = 0,$$

c'est-à-dire n'induisent pas de changement de volume : ils sont dits *isochores*.

Contraintes de VON MISES Les écoulements décrits ici sont supposés dépendre du tenseur des contraintes $\underline{\sigma}$ à travers la norme de son déviateur (5), appelée contrainte de VON MISES :

(6) $$\sigma_{eq} = \sqrt{\frac{3}{2}\,\underline{\mathbf{s}} : \underline{\mathbf{s}}}.$$

La contrainte de VON MISES est un des invariants isotropes de la contrainte. L'hypothèse assure donc le caractère isotrope de la loi d'écoulement.

Écrouissage Pour chaque écoulement i, la loi d'écoulement peut également dépendre de son *écrouissage*, ou *déformation inélastique cumulée* p_i, défini par l'équation différentielle :

(7a) $$\dot{p}_i = \sqrt{\frac{2}{3}\,\dot{\underline{\epsilon}}_i^{an} : \dot{\underline{\epsilon}}_i^{an}}.$$

Le facteur $3/2$, introduit par convention dans la contrainte de VON MISES (6), est compensé ici dans la définition de $\dot{p}_i$. En général, en début de calcul, au temps $t = t_0$, le matériau est supposé non écroui :

(7b) $$p_i(t_0) = 0.$$

Avec l'hypothèse sur la direction d'écoulement (4), la relation (7a) s'inverse en :

(8) $$\dot{\underline{\epsilon}}_i^{an} = \dot{p}_i\, \underline{\mathbf{n}}, \quad \text{avec} \quad \underline{\mathbf{n}} = \frac{3}{2}\frac{\underline{\mathbf{s}}}{\sigma_{eq}},$$

où le tenseur $\underline{\mathbf{n}}$ est appelé *normale* à l'écoulement. C'est un tenseur déviatorique de norme constante, qui vérifie :

(9) $$\underline{\underline{\mathbf{K}}} : \underline{\mathbf{n}} = \underline{\mathbf{n}}, \quad \text{et} \quad \underline{\mathbf{n}} : \underline{\mathbf{n}} = \frac{3}{2}.$$

Écoulements supportés Il est maintenant possible de formuler les trois types de mécanismes qui relèvent d'une intégration spécifique dans `mfront` :

— des écoulements viscoplastiques de la forme suivante, dont relève la loi de NORTON [Chaboche 09] :

$$\underline{\dot{\epsilon}}_i^{an} = f_i^{\mathsf{an}}\left(\sigma_{eq}\right)\,\underline{\mathbf{n}}, \quad \text{c'est-à-dire} \quad \dot{p}_i = f_i^{\mathsf{an}}\left(\sigma_{eq}\right); \tag{10a}$$

— des écoulements viscoplastiques avec écrouissage de la forme suivante, dont relève la loi LEMAITRE [Chaboche 09] :

$$\underline{\dot{\epsilon}}_i^{an} = f_i^{\mathsf{an}}\left(\sigma_{eq}, p_i\right)\,\underline{\mathbf{n}}, \quad \text{c'est-à-dire} \quad \dot{p}_i = f_i^{\mathsf{an}}\left(\sigma_{eq}, p_i\right). \tag{10b}$$

— et des écoulements plastiques qui satisfont une relation du type :

$$f_i^{\mathsf{an}}(\sigma_{eq}, p_i) \leq 0 \qquad \dot{p}_i \geq 0 \qquad f_i^{\mathsf{an}}(\sigma_{eq}, p_i)\,\dot{p}_i = 0 \tag{10c}$$

Il est classique que la fonction f_i^{an} de l'écoulement plastique ait la dimension d'une contrainte. `mfront` suppose que l'utilisateur a respecté cette convention, et divise de ce fait la fonction f_i^{an} par le module d'Young du matériau, pour que l'ensemble des inéquations (10c) aient la dimension d'une déformation.

La fonction f_i^{an}, qui définit l'écoulement, peut éventuellement faire intervenir des variables externes évoluant indépendamment de la mécanique (densité de fission, flux de neutrons rapides, taille de grain, *etc.*).

Remarques

1. Les écoulements viscoplastiques sans écrouissage (10a) sont un cas particulier des écoulements avec écrouissage (10b). Pour optimiser les temps de calculs, il est intéressant de maintenir leur distinction lors de l'implantation numérique. Pour les développements théoriques, il est préférable de regrouper ces écoulements sous leur forme commune (10b).
2. Avec l'hypothèse sur la direction de l'écoulement (8), les écoulements viscoplastiques (10b) comme les écoulements plastiques (10c) dépendent de l'écrouissage p_i au lieu de dépendre de la déformation $\underline{\epsilon}_i^{an}$. Cette particularité explique l'optimisation possible lors de l'intégration des lois (10) : il suffit d'intégrer des équations avec une inconnue scalaire (p_i) plutôt que tensorielle ($\underline{\epsilon}_i^{an}$).
3. L'évolution de la déformation viscoplastique (10b) est donnée par une équation différentielle, tandis que la formulation des lois plastiques est basée sur une équation (10c) : le respect de la surface de charge.

Système d'équations L'intégration de la loi de comportement proposée par `mfront` consiste à charger un point matériel avec la déformation totale $\underline{\epsilon}^{to}$ et à en déduire l'évolution des contraintes $\underline{\sigma}$. Pour cela, la loi s'appuie sur un jeu de variables internes, conservées en mémoire, et qui évoluent simultanément. Nous avons décidé de choisir comme variables internes :

— la déformation élastique $\underline{\epsilon}^{el}$;
— l'écrouissage p_i associé à chaque mécanisme.

Les équations qui permettent de calculer cette évolution sont rassemblées ici. Il s'agit de :

1. la partition des déformations (2)

$$\underline{\epsilon}^{to} = \underline{\epsilon}^{el} + \underline{\epsilon}^{an}, \quad \text{avec} \quad \underline{\epsilon}^{an} = \sum_i \underline{\epsilon}_i^{an}; \tag{11a}$$

2. la loi d'élasticité (3)

$$\underline{\sigma} = \underline{\underline{\mathbf{D}}} : \underline{\epsilon}^{el}; \tag{11b}$$

3. la décomposition du tenseur des contraintes $\underline{\sigma}$ en déviateur (5) et contrainte de VON MISES (6) :

$$\underline{\mathbf{s}} = \underline{\underline{\mathbf{K}}} : \underline{\sigma}, \quad \text{et} \quad \sigma_{eq} = \sqrt{\frac{3}{2}\,\underline{\mathbf{s}} : \underline{\mathbf{s}}}; \tag{11c}$$

4. la direction des écoulements, orientée suivant la normale à l'écoulement (8) :

$$\underline{\dot{\epsilon}}_i^{an} = \dot{p}_i\,\underline{\mathbf{n}}, \quad \text{avec} \quad \underline{\mathbf{n}} = \frac{3}{2}\frac{\underline{\mathbf{s}}}{\sigma_{eq}}; \tag{11d}$$

5. les écoulements viscoplastiques (10b) ou plastiques (10c) :

$$\dot{p}_i = f_i^{\mathsf{an}}\left(\sigma_{eq}, p_i\right), \tag{11e}$$

$$f_i^{\mathsf{an}}(\sigma_{eq}, p_i) \leq 0, \quad \dot{p}_i \geq 0, \quad f_i^{\mathsf{an}}(\sigma_{eq}, p_i)\,\dot{p}_i = 0. \tag{11f}$$

3.1.2 Méthode d'intégration numérique

Dans `mfront`, la loi de comportement (11) est intégrée par une méthode implicite, une θ-méthode, semblable à celle présentée au paragraphe 6, à ceci près que le système d'équations peut ici être réduit à une unique équation scalaire [Proix 12b].

Incrément de temps D'après ce qui précède, les variables internes sont la déformation élastique $\underline{\epsilon}^{el}$ et l'écrouissage p_i associé à chaque mécanisme. Elles sont connues au temps t, en début de pas de temps, et pour une déformation totale $\underline{\epsilon}^{to}|_t$. L'intégration consiste à calculer leurs nouvelles valeurs induites par un incrément de déformation totale $\Delta\underline{\epsilon}^{to}$ en fin d'un pas de temps Δt. Leurs incréments respectifs seront notés $\Delta\underline{\epsilon}^{el}$ et Δp_i.

La notion de θ-méthode, employée ici, consiste à s'intéresser également aux valeurs de certaines variables au temps intermédiaire $t + \theta\,\Delta t$, où le paramètre θ, déterminé à l'avance est compris entre 0 et 1. Une première approximation consiste à supposer les variables internes linéaires sur le pas de temps Δt, donc à poser :

$$\underline{\epsilon}^{el}\big|_{t+\theta\,\Delta\,t} \approx \underline{\epsilon}^{el}\big|_t + \theta\,\Delta\underline{\epsilon}^{el}, \tag{12a}$$

$$p_i|_{t+\theta\,\Delta\,t} \approx p_i|_t + \theta\,\Delta p_i. \tag{12b}$$

Contraintes D'après la loi d'élasticité (3) et sa propre définition (5), le déviateur des contraintes vaut :

$$\underline{\mathbf{s}} = 2\,\mu\,\underline{\underline{\mathbf{K}}}\colon \underline{\epsilon}^{el},$$

et d'après les approximations (12), sa valeur au temps intermédiaire vaut :

$$\underline{\mathbf{s}}|_{t+\theta\,\Delta\,t} = 2\,\mu\,\underline{\underline{\mathbf{K}}}\colon \left(\underline{\epsilon}^{el}\big|_t + \theta\,\Delta\underline{\epsilon}^{el}\right), \tag{13}$$

et la contrainte de VON MISES (6) et la normale à l'écoulement (8) pour cette même date s'en déduisent :

$$\sigma_{eq}|_{t+\theta\,\Delta\,t} = \sqrt{\frac{3}{2}\;\underline{\mathbf{s}}|_{t+\theta\,\Delta\,t} :\; \underline{\mathbf{s}}|_{t+\theta\,\Delta\,t}}, \quad \text{et} \quad \underline{\mathbf{n}}|_{t+\theta\,\Delta\,t} = \frac{3}{2}\frac{\underline{\mathbf{s}}|_{t+\theta\,\Delta\,t}}{\sigma_{eq}|_{t+\theta\,\Delta\,t}}. \tag{14}$$

Partition des déformations La partition des déformations (2), et la direction des écoulements (8) s'intègrent en :

$$\Delta\underline{\epsilon}^{to} = \Delta\underline{\epsilon}^{el} + \sum_i \int_t^{t+\Delta t} \dot{p}_i\,\underline{\mathbf{n}}\,\mathrm{d}t \approx \Delta\underline{\epsilon}^{el} + \underline{\mathbf{n}}|_{t+\theta\,\Delta\,t} \sum_i \Delta p_i. \tag{15}$$

Cette approximation s'appuie sur la valeur de la normale $\underline{\mathbf{n}}$ au temps intermédiaire. Avec les relations précédentes (13) et (14), et le type déviateur (9) de $\underline{\mathbf{n}}$, cela permet d'écrire :

$$\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t} = \frac{3\,\mu}{\sigma_{eq}|_{t+\theta\,\Delta\,t}}\left[\underline{\underline{\mathbf{K}}}\colon \left(\underline{\epsilon}^{el}\big|_t + \theta\,\Delta\underline{\epsilon}^{to}\right) - \underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}\;\theta\sum_i \Delta p_i\right],$$

c'est-à-dire :

$$\left(\sigma_{eq}|_{t+\theta\,\Delta\,t} + 3\mu\theta \sum_i \Delta p_i\right) \underline{\mathbf{n}}|_{t+\theta\,\Delta\,t} = 3\mu\underline{\mathbf{B}}, \quad \text{avec} \quad \underline{\mathbf{B}} = \underline{\underline{\mathbf{K}}} : \left(\underline{\epsilon}^{el}\big|_t + \theta\,\Delta\underline{\epsilon}^{to}\right). \tag{16}$$

La norme de cette expression (16), compte tenu de la norme (9) de $\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}$, donne finalement pour la contrainte équivalente :

$$\sigma_{eq}|_{t+\theta\,\Delta\,t} = \mu\sqrt{6\,\underline{\mathbf{B}} : \underline{\mathbf{B}}} - 3\,\mu\,\theta \sum_i \Delta p_i. \tag{17}$$

Cette valeur, réinjectée dans la relation (16), donne la normale :

$$\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t} = \frac{3}{\sqrt{6\,\underline{\mathbf{B}} : \underline{\mathbf{B}}}}\underline{\mathbf{B}}. \tag{18}$$

L'équation (17) va permettre de former, à partir des équations d'écoulements, un système non linéaire d'équations dont les inconnues sont les incréments Δp_i. Il faut maintenant construire ce système en intégrant les écoulements viscoplastiques, puis plastiques.

Écoulement viscoplastique L'intégrale sur le pas de temps Δt de l'écoulement viscoplastique (10b) s'écrit :

$$\Delta p_i = \int_t^{t+\Delta t} f_i^{\text{an}}\left(\sigma_{eq}, p_i\right) \mathrm{d}t \approx \Delta t\, f_i^{\text{an}}\left(\sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right),$$

où l'intégrale est approchée par la valeur de f_i^{an} au temps intermédiaire. Avec cette approximation, vérifier l'écoulement consiste à annuler une fonction à trois paramètres :

$$f_{p_i}\left(\Delta p_i, \sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right) = \Delta p_i - \Delta t\, f_i^{\text{an}}\left(\sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right). \tag{19a}$$

Écoulement plastique L'écoulement plastique consiste à annuler une fonction de même forme, dépendant des mêmes trois paramètres :

$$f_{p_i}\left(\Delta p_i, \sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right) = \begin{cases} f_i^{\text{an}}\left(\sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right) & \text{en cas de charge plastique,} \\ \Delta p_i & \text{sinon.} \end{cases} \tag{19b}$$

La condition de charge plastique, qui augmente l'écrouissage p_i, doit respecter les conditions d'écoulement plastique (10c). Dans la pratique, les incréments Δp_i, inconnues d'un système non linéaire, sont calculés de manière itérative. Chaque nouvelle itération propose de nouvelles valeurs de ces incréments. Pour ces nouvelles valeurs, la condition de charge plastique est activée si l'une des conditions suivante est vérifiée :

1. $f_i^{\text{an}}\left(\sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right) > \varepsilon$ et $\Delta p_i \geq 0$,
2. $\Delta p_i > \varepsilon$,

où ε est un paramètre numérique, qui stabilise les itérations.

Système non linéaire Les équations d'écoulement (19) :

$$f_{p_i}\left(\Delta p_i, \sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right) = 0, \quad \forall\, p_i,$$

modifiées avec les équations (12) et (17), forment un système non linéaire d'équations, dont les inconnues sont les incréments Δp_i :

$$f_{p_i}\left(\Delta p_i, \mu\sqrt{6\,\underline{\mathbf{B}}\colon\underline{\mathbf{B}}} - 3\,\mu\,\theta\sum_j \Delta p_j,\, p_i|_t + \theta\,\Delta p_i\right) = 0, \quad \forall\, p_i. \tag{20}$$

Pour le résoudre, une méthode de NEWTON-RAPHSON est utilisée. Cette méthode, discutée en détail au paragraphe 6.3, nécessite les dérivées partielles des fonctions f_{p_i} par rapport aux incréments Δp_i. Celles-ci s'obtiennent par une dérivation composée des f_{p_i} :

$$\frac{\partial f_{p_i}}{\partial \Delta p_j} = \frac{\partial f_{p_i}}{\partial \Delta p_i}\frac{\partial \Delta p_i}{\partial \Delta p_j} + \frac{\partial f_{p_i}}{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}\frac{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}{\partial \Delta p_j} + \frac{\partial f_{p_i}}{\partial\, p_i|_{t+\theta\,\Delta\,t}}\frac{\partial\, p_i|_{t+\theta\,\Delta\,t}}{\partial \Delta p_j},$$

avec, d'après les équations (12) et (17) :

$$\frac{\partial \Delta p_i}{\partial \Delta p_j} = \delta_{ij}, \quad \frac{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}{\partial \Delta p_j} = -3\,\mu\,\theta, \quad \text{et} \quad \frac{\partial\, p_i|_{t+\theta\,\Delta\,t}}{\partial \Delta p_j} = \theta\,\delta_{ij}, \quad \text{avec} \quad \delta_{ij} = \begin{cases} 1 & \text{si } i = j, \\ 0 & \text{sinon.} \end{cases}$$

Appliquée aux écoulements, ces relations différentielles donnent :

— pour les écoulement viscoplastiques (19a) :

$$\frac{\partial f_{p_i}}{\partial \Delta p_j} = \begin{cases} 1 - 3\,\mu\,\theta\,\Delta t\,\left[\dfrac{\partial f_i^{\text{an}}}{\partial \sigma_{eq}}\left(\sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right)\right] - \theta\,\Delta t\,\left[\dfrac{\partial f_i^{\text{an}}}{\partial p_i}\left(\sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right)\right] & \text{si } i = j, \\ -3\,\mu\,\theta\,\Delta t\,\left[\dfrac{\partial f_i^{\text{an}}}{\partial \sigma_{eq}}\left(\sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right)\right] & \text{si } i \neq j; \end{cases}$$

— pour les écoulement plastiques (19b), en cas de décharge plastique :

$$\frac{\partial f_{p_i}}{\partial \Delta p_j} = \begin{cases} \theta\,\left(3\,\mu\,\left[\dfrac{\partial f_i^{\text{an}}}{\partial \sigma_{eq}}\left(\sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right)\right] + \left[\dfrac{\partial f_i^{\text{an}}}{\partial p_i}\left(\sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right)\right]\right) & \text{si } i = j, \\ -3\,\mu\,\theta\,\dfrac{\partial f_i^{\text{an}}}{\partial \sigma_{eq}}\left(\sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right) & \text{si } i \neq j. \end{cases}$$

et en absence de charge plastique :

$$\frac{\partial f_{p_i}}{\partial \Delta p_j} = \begin{cases} 1 \text{ si } i = j, \\ 0 \text{ si } i \neq j. \end{cases}$$

Critère d'arrêt L'algorithme de NEWTON s'arrête quand la différence entre deux estimations des incréments des déformations viscoplastiques cumulées est inférieur à un certain critère ε :

$$\frac{1}{N}\sum_{i=1}^{N} |\Delta_n \Delta p_i| < \varepsilon \tag{21}$$

où$\Delta_n \Delta p_i$ désigne la différence entre l'estimation de l'incrément de la déformation viscoplastique cumulée $\Delta\, p_i$ à l'étape $n+1$ et de l'estimation à l'étape n.

L'algorithme échoue si le nombre d'itérations dépasse une borne maximale.

Étapes Finalement les différentes étapes de l'intégration de la loi de comportement mécanique par la θ-méthode sont les suivantes :

1. calcul du tenseur $\underline{\mathbf{B}}$ (16) :

$$\underline{\mathbf{B}} = \underline{\underline{\mathbf{K}}}\colon\, \left(\underline{\epsilon}^{el}\big|_t + \theta\,\Delta\underline{\epsilon}^{to}\right) \;;$$

2. calcul des incréments Δp_i, par résolution du système non linéaire (20) :

$$f_{p_i}\left(\Delta p_i, \mu\sqrt{6\,\underline{\mathbf{B}}\colon\underline{\mathbf{B}}} - 3\,\mu\,\theta\sum_j \Delta p_j,\, p_i|_t + \theta\,\Delta p_i\right) = 0, \quad \forall\, p_i,$$

en utilisant la méthode de NEWTON-RAPHSON, et calcul des nouvelles valeurs d'écrouissage :

$$p_i|_{t+\Delta\,t} = p_i|_t + \Delta p_i\ ;$$

3. calcul de l'incrément de déformation élastique, grâce à la partition des déformations (15), de la normale (18) :

$$\Delta\underline{\epsilon}^{el} = \Delta\underline{\epsilon}^{to} - \underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}\sum_i \Delta p_i, \quad \text{avec} \quad \underline{\mathbf{n}}|_{t+\theta\,\Delta\,t} = \frac{3}{\sqrt{6\,\underline{\mathbf{B}}\colon\underline{\mathbf{B}}}}\underline{\mathbf{B}},$$

et de la nouvelle valeur de déformation élastique :

$$\underline{\epsilon}^{el}\big|_{t+\Delta\,t} = \underline{\epsilon}^{el}\big|_t + \Delta\underline{\epsilon}^{el}\ ;$$

4. calcul de la contrainte en fin de pas de temps, par la loi d'élasticité (3) :

$$\underline{\sigma}|_{t+\Delta\,t} = \underline{\underline{\mathbf{D}}}\colon\ \underline{\epsilon}^{el}\big|_{t+\Delta\,t}\,.$$

3.1.3 Matrice tangente cohérente

La méthode de résolution implicite permet le calcul de la matrice tangente cohérente. Cette matrice tangente cohérente permet une convergence quadratique du calcul de structure, ce qui peut nettement accélérer les calculs.

Définition L'intégration numérique décrite au paragraphe précédent, permet de calculer l'évolution des contraintes $\Delta\underline{\sigma}$ induite par un incrément de déformation $\Delta\underline{\epsilon}^{to}$ sur un pas de temps Δt. La matrice tangente cohérente est définie comme le tenseur :

$$\underline{\underline{\mathbf{L}}}^{tc} = \frac{\partial\Delta\underline{\sigma}}{\partial\Delta\underline{\epsilon}^{to}}. \tag{22}$$

La loi d'élasticité (3) et la partition des déformations (15) permettent d'exprimer l'incrément de contrainte :

$$\Delta\underline{\sigma} = \underline{\underline{\mathbf{D}}}\colon\Delta\underline{\epsilon}^{el} = \underline{\underline{\mathbf{D}}}\colon\left(\Delta\underline{\epsilon}^{to} - \underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}\sum_i \Delta p_i\right)$$

et d'en déduire par dérivation (22) la matrice tangente cohérente :

$$\underline{\underline{\mathbf{L}}}^{tc} = \underline{\underline{\mathbf{D}}} - \underline{\underline{\mathbf{D}}}\colon\left(\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}\otimes\sum_i \frac{\partial\Delta p_i}{\partial\Delta\underline{\epsilon}^{to}} + \sum_i \Delta p_i\frac{\partial\,\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}}{\partial\Delta\underline{\epsilon}^{to}}\right). \tag{23}$$

Il faut maintenant calculer les dérivées de la normale $\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}$ et des incréments Δp_i pour chaque écoulement.

Normale La dérivée du tenseur (16) $\underline{\mathbf{B}}$ et de la normale (18) valent respectivement :

$$\frac{\partial\underline{\mathbf{B}}}{\partial\Delta\underline{\epsilon}^{to}} = \theta\,\underline{\underline{\mathbf{K}}}, \tag{24a}$$

$$\begin{aligned}\frac{\partial\,\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}}{\partial\Delta\underline{\epsilon}^{to}} &= \frac{3}{\sqrt{6\,\underline{\mathbf{B}}\colon\underline{\mathbf{B}}}}\left(\frac{\partial\underline{\mathbf{B}}}{\partial\Delta\underline{\epsilon}^{to}} - \frac{2}{3}\frac{\partial\underline{\mathbf{B}}}{\partial\Delta\underline{\epsilon}^{to}}\colon\ \underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}\otimes\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}\right),\\ &= \frac{3\,\theta}{\sqrt{6\,\underline{\mathbf{B}}\colon\underline{\mathbf{B}}}}\left(\underline{\underline{\mathbf{K}}} - \frac{2}{3}\,\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}\otimes\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}\right).\end{aligned} \tag{24b}$$

Ce calcul utilise la propriété (9) du tenseur $\underline{\mathbf{n}}|_{t+\theta\,\Delta\,t}$.

Incréments Les incréments d'écrouissage Δp_i sont issus de la résolution du système non linéaire, formé par les équations d'écoulement (19) :

$$f_{p_i}\left(\Delta p_i,\, \sigma_{eq}|_{t+\theta\,\Delta\,t}\,,\, p_i|_{t+\theta\,\Delta\,t}\right) = 0, \quad \forall\, p_i.$$

La dérivation de ces équations conduit à :

$$\frac{\partial f_{p_i}}{\partial \Delta p_i}\,\frac{\partial \Delta p_i}{\partial \underline{\underline{\epsilon}}^{to}} + \frac{\partial f_{p_i}}{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}\,\frac{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}{\partial \underline{\underline{\epsilon}}^{to}} + \frac{\partial f_{p_i}}{\partial\, p_i|_{t+\theta\,\Delta\,t}}\,\frac{\partial\, p_i|_{t+\theta\,\Delta\,t}}{\partial \underline{\underline{\epsilon}}^{to}} = 0, \quad \forall\, p_i. \tag{25}$$

La dérivation des équations (12) et (17), en utilisant les relations (9) et (24a) :

$$\begin{aligned}
\frac{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}{\partial \underline{\underline{\epsilon}}^{to}} &= 2\,\mu\,\theta\;\underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t} - 3\,\mu\,\theta\,\sum_j \frac{\partial \Delta p_j}{\partial \underline{\underline{\epsilon}}^{to}},\\
\frac{\partial\, p_i|_{t+\theta\,\Delta\,t}}{\partial \underline{\underline{\epsilon}}^{to}} &= \theta\,\frac{\partial \Delta p_i}{\partial \underline{\underline{\epsilon}}^{to}},
\end{aligned}$$

permet de modifier les équations (25) :

$$\left(\frac{\partial f_{p_i}}{\partial \Delta p_i} + \theta\,\frac{\partial f_{p_i}}{\partial\, p_i|_{t+\theta\,\Delta\,t}}\right)\frac{\partial \Delta p_i}{\partial \underline{\underline{\epsilon}}^{to}} - 3\,\mu\,\theta\,\frac{\partial f_{p_i}}{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}\,\sum_j \frac{\partial \Delta p_j}{\partial \underline{\underline{\epsilon}}^{to}} = -\frac{\partial f_{p_i}}{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}\,2\,\mu\,\theta\;\underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t}, \quad \forall\, p_i. \tag{26}$$

C'est un jeu d'équations linéaires, dont les inconnues sont les tenseurs dérivées recherchés $\dfrac{\partial \Delta p_i}{\partial \underline{\underline{\epsilon}}^{to}}$.

Solutions scalaires Le produit contracté des équations (26) par $\underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t}$ donne un système d'équations :

$$\left(\frac{\partial f_{p_i}}{\partial \Delta p_i} + \theta\,\frac{\partial f_{p_i}}{\partial\, p_i|_{t+\theta\,\Delta\,t}}\right) d_i - 3\,\mu\,\theta\,\frac{\partial f_{p_i}}{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}\,\sum_j d_j = -\frac{\partial f_{p_i}}{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}\,3\,\mu\,\theta, \quad \forall\, p_i. \tag{27a}$$

avec pour inconnues, les produits contractés :

$$d_i = \frac{\partial \Delta p_i}{\partial \underline{\underline{\epsilon}}^{to}} :\; \underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t}\,. \tag{27b}$$

Le produit contracté des équations (26) par tout tenseur $\underline{\underline{\mathbf{x}}}$ orthogonal à $\underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t}$ donne un système d'équations :

$$\left(\frac{\partial f_{p_i}}{\partial \Delta p_i} + \theta\,\frac{\partial f_{p_i}}{\partial\, p_i|_{t+\theta\,\Delta\,t}}\right)\frac{\partial \Delta p_i}{\partial \underline{\underline{\epsilon}}^{to}} : \underline{\underline{\mathbf{x}}} - 3\,\mu\,\theta\,\frac{\partial f_{p_i}}{\partial\, \sigma_{eq}|_{t+\theta\,\Delta\,t}}\,\sum_j \frac{\partial \Delta p_j}{\partial \underline{\underline{\epsilon}}^{to}} : \underline{\underline{\mathbf{x}}} = 0, \quad \forall\, p_i,$$

de second membre nul. Ses inconnues sont donc nulles :

$$\frac{\partial \Delta p_i}{\partial \underline{\underline{\epsilon}}^{to}} : \underline{\underline{\mathbf{x}}} = 0, \quad \forall\, p_i, \tag{28}$$

ce qui montre que les tenseurs $\dfrac{\partial \Delta p_i}{\partial \underline{\underline{\epsilon}}^{to}}$ sont tous colinéaires au tenseur $\underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t}$. Ils s'écrivent donc :

$$\frac{\partial \Delta p_i}{\partial \underline{\underline{\epsilon}}^{to}} = \frac{\dfrac{\partial \Delta p_i}{\partial \underline{\underline{\epsilon}}^{to}} : \underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t}}{\underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t} :\; \underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t}}\;\underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t} = \frac{2\,d_i}{3}\;\underline{\underline{\mathbf{n}}}|_{t+\theta\,\Delta\,t}\,, \tag{29}$$

où les composantes d_i sont les solutions du système (27).

Matrice tangente cohérente Finalement, la matrice tangente cohérente peut être reconstituée à partir des expressions (23), (24b) et (29), et en notant que :

$$\underline{\underline{\mathbf{D}}} : \underline{\underline{\mathbf{K}}} = 2\,\mu\,\underline{\underline{\mathbf{K}}}, \qquad \underline{\underline{\mathbf{D}}} : \left.\underline{\mathbf{n}}\right|_{t+\theta\,\Delta\,t} = 2\,\mu\,\left.\underline{\mathbf{n}}\right|_{t+\theta\,\Delta\,t}.$$

Elle vaut ainsi :

$$\underline{\underline{\mathbf{L}}}^{tc} = \underline{\underline{\mathbf{D}}} - 2\,\mu\,\left(\frac{3\,\theta\,\sum_i \Delta p_i}{\sqrt{6\,\underline{\mathbf{B}} : \underline{\mathbf{B}}}}\,\underline{\underline{\mathbf{K}}} + \left(\frac{2\,\sum_i d_i}{3} - \frac{2\,\theta\,\sum_i \Delta p_i}{\sqrt{6\,\underline{\mathbf{B}} : \underline{\mathbf{B}}}}\right)\,\left.\underline{\mathbf{n}}\right|_{t+\theta\,\Delta\,t} \otimes \left.\underline{\mathbf{n}}\right|_{t+\theta\,\Delta\,t}\right), \tag{30}$$

avec les composantes d_i solutions du système (27).

3.1.4 Expression de la matrice tangente cohérente pour un mécanisme unique

Dans le cas où le système se limite à un mécanisme unique, l'expression de la matrice tangente cohérente se simplifie.

Écoulement viscoplastique Dans le cas d'un seul écoulement viscoplastique (19a), d'incrément Δp, le système (27) se réduit à une équation d'inconnue d :

$$\left(1 - \Delta t\,\theta\,\frac{\partial f^{\text{an}}}{\partial p} + 3\,\mu\,\Delta t\,\theta\,\frac{\partial f^{\text{an}}}{\partial \sigma_{eq}}\right)\,d = \frac{\partial f^{\text{an}}}{\partial \sigma_{eq}}\,3\,\mu\,\Delta t\,\theta.$$

L'expression (30) de la matrice tangente cohérente se simplifie alors en :

$$\underline{\underline{\mathbf{L}}}^{tc} = \underline{\underline{\mathbf{D}}} - 2\,\mu\,\theta\,\left(\frac{3\,\Delta p}{\sqrt{6\,\underline{\mathbf{B}} : \underline{\mathbf{B}}}}\,\underline{\underline{\mathbf{K}}} + \left(\frac{2\,\mu\,\Delta t\,\dfrac{\partial f^{\text{an}}}{\partial \sigma_{eq}}}{1 + \theta\,\Delta t\,\left(3\,\mu\,\dfrac{\partial f^{\text{an}}}{\partial \sigma_{eq}} - \dfrac{\partial f^{\text{an}}}{\partial p}\right)} - \frac{2\,\Delta p}{\sqrt{6\,\underline{\mathbf{B}} : \underline{\mathbf{B}}}}\right)\,\left.\underline{\mathbf{n}}\right|_{t+\theta\,\Delta\,t} \otimes \left.\underline{\mathbf{n}}\right|_{t+\theta\,\Delta\,t}\right) \tag{31}$$

Écoulement plastique Dans le cas d'un seul écoulement viscoplastique, d'incrément Δp, le système (27) se réduit à une équation d'inconnue d. Si l'écoulement plastique est activé (19b), cette équation devient :

$$\left(\theta\,\frac{\partial f^{\text{an}}}{\partial p} - 3\,\mu\,\theta\,\frac{\partial f^{\text{an}}}{\partial \sigma_{eq}}\right)\,d = -\frac{\partial f^{\text{an}}}{\partial \sigma_{eq}}\,3\,\mu\,\theta.$$

L'expression (30) de la matrice tangente cohérente se simplifie alors en :

$$\underline{\underline{\mathbf{L}}}^{tc} = \underline{\underline{\mathbf{D}}} - 2\,\mu\,\theta\,\left(\frac{3\,\Delta p}{\sqrt{6\,\underline{\mathbf{B}} : \underline{\mathbf{B}}}}\,\underline{\underline{\mathbf{K}}} + \left(\frac{2\,\mu\,\dfrac{\partial f^{\text{an}}}{\partial \sigma_{eq}}}{3\,\mu\,\theta\,\dfrac{\partial f^{\text{an}}}{\partial \sigma_{eq}} - \theta\,\dfrac{\partial f^{\text{an}}}{\partial p}} - \frac{2\,\Delta p}{\sqrt{6\,\underline{\mathbf{B}} : \underline{\mathbf{B}}}}\right)\,\left.\underline{\mathbf{n}}\right|_{t+\theta\,\Delta\,t} \otimes \left.\underline{\mathbf{n}}\right|_{t+\theta\,\Delta\,t}\right) \tag{32}$$

Lorsque l'écoulement plastique n'est pas activé, l'équation d'écoulement (19b) et le système (27) se réduisent respectivement à :

$$\Delta p = 0, \quad \text{et} \quad d = 0,$$

et la matrice tangente cohérente au tenseur d'élasticité :

$$\underline{\underline{\mathbf{L}}}^{tc} = \underline{\underline{\mathbf{D}}}.$$

3.1.5 Matrice tangente

La matrice tangente définie par la relation en vitesse :

$$\underline{\dot{\sigma}} = \underline{\underline{\mathbf{L}}} : \underline{\dot{\epsilon}}^{to}$$

Sans chercher à justifier cette affirmation, la matrice tangente peut être calculée comme la limite de la matrice tangente cohérente lorsque le pas de temps Δt et l'incrément de déformation plastique cumulée Δp tendent vers 0 et en posant $\theta = 1$:

$$\underline{\underline{\mathbf{L}}} = \lim_{\substack{\Delta t \to 0 \\ \Delta p \to 0 \\ \theta = 1}} \underline{\underline{\mathbf{L}}}^{tc}$$

Les expressions précédentes peuvent donc être réutilisées en prenant garde à l'instant où l'on veut calculer la matrice tangente (en début ou fin de pas de temps).

3.1.6 Expression de la matrice tangente pour un mécanisme unique

Dans le cas où le système se limite à un mécanisme unique, l'expression de la matrice tangente cohérente se simplifie.

Écoulement viscoplastique Dans le cas d'un seul écoulement viscoplastique, l'expression (31) de la matrice tangente cohérente se réduit à la matrice d'élasticité.

Écoulement plastique Dans le cas d'un écoulement plastique, deux cas se présentent suivant que l'on soit en charge ou pas.

En cas de charge plastique, l'expression (32) de la matrice tangente cohérente conduit à :

$$\underline{\underline{\mathbf{L}}} = \underline{\underline{\mathbf{D}}} - \frac{4\,\mu^2 \dfrac{\partial f^{\mathsf{an}}}{\partial \sigma_{eq}}}{3\,\mu \dfrac{\partial f^{\mathsf{an}}}{\partial \sigma_{eq}} - \dfrac{\partial f^{\mathsf{an}}}{\partial p}} \underline{\mathbf{n}} \otimes \underline{\mathbf{n}}$$

Dans le domaine élastique ou en cas de décharge, la matrice tangente se réduit à la matrice d'élasticité.

3.2 UTILISATION DES ANALYSEURS SPÉCIFIQUES

Nous présentons dans ce paragraphe les analyseurs `mfront` dédiés aux lois de comportement plastique et viscoplastique incompressible des matériaux isotropes. Ils sont au nombre de 4 :

- l'analyseur `IsotropicMisesCreep` gère exclusivement les lois de comportement viscoplastique isotrope de la forme :
$$\dot{p} = f\left(\sigma_{eq}\right)$$
- l'analyseur `IsotropicStrainHardeningMisesCreep` gère exclusivement les lois de comportement viscoplastique isotrope de la forme :
$$\dot{p} = f\left(\sigma_{eq}, p\right)$$
- l'analyseur `IsotropicPlasticMisesFlow` gère exclusivement les lois de comportement plastique isotrope de la forme :
$$f(\sigma_{eq}, p) \leq 0 \qquad \dot{p} \geq 0 \qquad f(\sigma_{eq}, p)\,\dot{p} = 0$$

— l'analyseur `MultipleIsotropicMisesFlows` gère une combinaison arbitraire d'écoulements des trois types précédents. Les différents écoulements sont supposés non couplés.

3.2.1 La directive `@FlowRule`

La directive `@FlowRule` permet de définir un écoulement.

Cas des analyseurs `IsotropicMisesCreep`, `IsotropicStrainHardeningMisesCreep` et `IsotropicPlasticMisesFlow` Pour les analyseurs `IsotropicMisesCreep`, `IsotropicStrainHardeningMisesCreep` et `IsotropicPlasticMisesFlow`, une variable interne nommée p et représentant la déformation inélastique cumulée est automatiquement définie. Aucune autre variable interne ne peut être définie.

L'évolution de cette variable est introduite par la directive `@FlowRule`. Cette directive est suivie d'un bloc définissant l'écoulement. Ce bloc doit renseigner la valeur de la fonction `f`, dont la définition a été donnée plus haut en fonction de l'écoulement traité, et sa dérivée par rapport à la contrainte équivalente `df_dseq`. Pour les analyseurs `IsotropicStrainHardeningMisesCreep` et `IsotropicPlasticMisesFlow` il est également nécessaire de donner la dérivée de `f` par rapport à la déformation cumulée `df_dp`.

Dans le bloc suivant la directive `@FlowRule`, les variables `f`, `df_dseq` et éventuellement `df_dp` sont automatiquement définies. La contrainte équivalente actualisée en $t+\theta\,\Delta\,t$ est accessible par la variable `seq`. Pour les analyseurs `IsotropicStrainHardeningMisesCreep` et `IsotropicPlasticMisesFlow`. Si nécessaire, la déformation équivalente actualisée en $t+\theta\,\Delta\,t$ est accessible par la variable `p`.

Cas de l'analyseur `MultipleIsotropicMisesFlows` Plusieurs blocs `@FlowRule` peuvent être définis dans le cas de l'analyseur `MultipleIsotropicMisesFlows`. La directive `@FlowRule` est suivie d'un des trois types d'écoulement supportés, respectivement `Creep`, `StrainHardeningCreep` et `Plasticity`. Le bloc suivant décrit l'écoulement en suivant les règles données au paragraphe précédent. Notons que la déclaration d'un nouvel écoulement déclare automatiquement la déformation viscoplastique cumulée associée sous le nom `pi` où `i` est le nombre d'écoulement défini jusque là. Il n'est possible d'associer des noms de glossaire ou des bornes à ces variables qu'après la définition du bloc.

Transformation du code dans les blocs `@FlowRule` Le code dans les blocs suivant la directive `@FlowRule` est modifié ainsi :

— les variables externes sont remplacées par leurs valeurs en $t+\theta\,\Delta\,t$;
— la déformation inélastique cumulée p est remplacée par sa valeur en $t+\theta\,\Delta\,t$.

3.2.2 Mise à jour des variables auxiliaires

La directive `@UpdateAuxiliaryStateVariables` permet de mettre à jour les variables auxiliaires.

Déformations inélastiques Les déformations inélastiques $\underline{\epsilon}_i^{an}$ ne sont pas des variables internes. Pour y avoir accès (pour des posttraitements), il est possible de définir des variables internes auxiliaires.

Ces variables sont mises à jour après les variables internes et les contraintes, mais avant les variables externes (incluant la température) ou la déformation totale [11].

11. Les variables externes ne sont pas mises à jour car cela est du ressort de l'interface en cas de sous-découpage. Si aucun sous-découpage n'a lieu ou si l'on a réalisé le dernier pas de temps, on économise l'opération de mise à jour.

Exemple de la déformation inélastique totale Supposons la déformation inélastique totale représentée par une variable auxiliaire tensorielle nommée `evp` (déclarée par la directive `@AuxiliaryStateVariable`). Cette variable peut être calculée, dans le bloc suivant la directive `@UpdateAuxiliaryStateVariables` ainsi :

```
evp += deto-deel;
```

Une autre manière de calculer cette variable est :

```
evp = eto+deto-eel;
```

qui montre que la déformation élastique a été mise à jour et non la déformation totale.

3.2.3 Paramètres numériques automatiquement définis

Le paramètre θ de la θ-méthode vaut par défaut 1 pour l'analyseur `IsotropicPlasticMisesFlow` et $\frac{1}{2}$ pour les autres. Cette valeur par défaut peut être modifiée par la directive `@Theta`. Cette valeur est également un paramètre de la loi, nommé `theta`, qui peut être modifié à l'exécution.

La valeur du critère d'arrêt est par défaut de 10^{-8}. Cette valeur par défaut peut être modifiée par la directive `@Epsilon`. Cette valeur est également un paramètre de la loi, nommé `epsilon`, qui peut être modifié à l'exécution.

Le nombre maximum d'itération de l'algorithme est de 100 par défaut. Cette valeur par défaut peut être modifiée par la directive `@IterMax`. Cette valeur est également un paramètre de la loi, nommé `iterMax`, qui peut être modifié à l'exécution.

3.2.4 Noms réservés

Les noms réservés par cette analyseur sont décrits en annexe G.2.

3.2.5 Comportement viscoplastique du SiC

Nous voulons décrire le comportement viscoplastique du SiC Le SiC est supposé avoir un comportement viscoplastique isotrope donnée par :

$$\dot{p} = f\left(\sigma_{eq}\right)$$

où — p est la déformation viscoplastique équivalente ;
— σ_{eq} est la contrainte de VON MISES.

La fonction d'écoulement est donnée par :

(33) $$f\left(\sigma_{eq}\right) = \left(A\,\exp\left(-\frac{B}{T}\right) + a\,\phi\right)\,\sigma_{eq}$$

où : — A, B et a sont des coefficients ;
— T est la température ;
— ϕ est le flux de neutrons rapides ;

Cette loi de comportement dépend d'une variable externe, le flux de neutron rapide ϕ.

Pour implanter cette loi de comportement, nous utilisons l'analyseur `IsotropicMisesCreep`. Le code source est donné en figure 1.

```
@Parser IsotropicMisesCreep;
@Behaviour SiCCreep;
@Author    É. Brunon;
@Date      06/12/07;

@Description{
Matériaux RCG-T et RCG-R
Un point sur le carbure de Silicium
NT SESC/LIAC 02-024 ind 0 de décembre 2002
J.M. ESCLEINE
§8.4.2 Page 34
}

@ExternalStateVariable real flux;

@StaticVariable real A =  4.4e3;
@StaticVariable real B = 76.0e3;
@StaticVariable real a =  1.0e-37;

@LocalVariable real AF1;
@LocalVariable real AF3;

@InitLocalVariables{
  AF1   = A*exp(-B/(T+theta*dT));
  AF3   = a*(flux+theta*dflux);
}

@FlowRule{
  df_dseq  = AF1+AF3;
  f        = seq*df_dseq;
}
```

FIGURE 1 : Implantation de la loi de comportement viscoplastique du SiC en `mfront`.

Le mot clé `@Parser` La première ligne, commençant par le mot clé `@Parser`, décrit le type d'analyseur utilisé, ici `IsotropicMisesCreep`.

Le mot clé `@Behaviour` La seconde ligne, commençant par le mot clé `@Behaviour`, donne le nom de la loi de comportement.

Les mots clé `@Author` et `@Date` La troisième et la quatrième ligne renseignent respectivement l'auteur du fichier et la date de création à l'aide des mots clés `@Author` et `@Date`.

Le mot clé `@Description` Le mot clé `@Description` permet de donner les références bibliographiques d'où la loi est extraite.

Le mot clé `@ExternalStateVariable` Le mot clé `@ExternalStateVariable` définit une variable externe scalaire (`real`) nommée `flux`. Cette variable, et son incrément `dflux`, sont dès lors accessibles.

Le mot clé `@StaticVariable` Le mot clé `@StaticVariable` sert à définir les constantes utilisées par la loi. Ces constantes sont des scalaires (`real`).

Le mot clé `@LocalVariable` Le mot clé `@LocalVariable` sert à définir des variables de travail locales. Le rôle des variables `AF1` et `AF3` sera explicité dans la suite.

Le mot clé `@InitLocalVariables` Le mot clé `@InitLocalVars` permet d'écrire du code appelé avant tout calcul. Nous y initialisons les valeurs de coefficients de la loi. En effet, la méthode d'intégration utilisée évalue la fonction f au temps $t + \theta\,\Delta t$ (intégration implicite). Le coefficient dépendant de la température est donc connu et il est avantageux de l'évaluer ici plutôt qu'au cours des itérations de convergence de l'algorithme (nous économisons des appels à la fonction exponentielle). Nous utilisons ici le fait que la température au temps $t + \theta\,\Delta t$ est égale à $T + \theta\,\Delta T$. De même la valeur du flux de neutrons rapides ϕ est égal à $\phi + \theta\,\Delta\phi$.

Le mot clé `@FlowRule` Le mot clé `@FlowRule` permet de renseigner la fonction d'écoulement f et sa dérivée $\dfrac{\mathrm{d}f}{\mathrm{d}\sigma_{eq}}$.

4 INTÉGRATION DES LOIS DE COMPORTEMENT, INTÉGRATEUR PAR DÉFAUT

L'intégrateur par défaut est essentiellement utilisé à des fins de test ou pour certaines lois spécifiques (loi d'endommagement pilotée en déformations).

4.1 EXEMPLE DE L'ÉLASTICITÉ ORTHOTROPE

```
@Parser     DefaultParser;
@Behaviour OrthotropicElastic;
@Author     Helfer Thomas;
@Date       10/03/11;

@OrthotropicBehaviour;
@RequireStiffnessTensor;
@ProvidesSymmetricTangentOperator;

@Integrator{
  sig = D*(eto+deto);
  if(computeTangentOperator_){
    Dt = D;
  }
}
```

FIGURE 2 : Implantation d'une loi orthotrope élastique

La figure 2 décrit l'implantation d'une loi de comportement orthotrope élastique. Cette loi ne nécessitant pas d'algorithme d'intégration, l'analyseur par défaut est utilisé. La commande `@OrthotropicBehaviour` permet le support des lois orthotropes. La commande `@RequireStiffnessTensor` demande à ce que l'interface au code appelant mette à disposition une variable `D` contenant la matrice d'élasticité [12].

Le calcul des contraintes en fin de pas de temps (à l'instant $t+\Delta t$) s'écrit de manière similaire à son expression mathématique :

$$\underline{\sigma}_{t+\Delta t} = \underline{\underline{\mathbf{D}}} : \left(\underline{\epsilon}_t^{to} + \Delta \underline{\epsilon}^{to}\right)$$

Dans ce cas, le calcul de la matrice tangente cohérente est immédiat.

4.2 NOMS DES VARIABLES ET DES MÉTHODES C++ UTILISÉS EN INTERNE

Pour éviter les conflits avec les noms de variables de travail ou des méthodes C++ utilisés par les classes générées, certains noms de variables ne peuvent être utilisés. Leur liste est donnée en annexe G.1.

4.3 LA DIRECTIVE `@INTEGRATOR`

La directive `@Integrator` permet d'intégrer la loi de comportement.

12. Pour ce faire, l'interface déclare des propriétés matériaux supplémentaires. Leur nombre et l'ordre dans lequel il est nécessaire de les passer sont décrits dans les documentations des interfaces [Helfer 13c, Helfer 13b].

Conventions spécifiques Les conventions suivantes s'appliquent :

- `Dt` représente la matrice tangente cohérente qu'il faut calculer ;
- `sig` représente la contrainte qu'il faut calculer ;
- `eto` représente la déformation totale en début de pas ;
- `deto` représente l'incrément de déformation totale (constante sur le pas) ;
- `T` représente la valeur de la température en début de pas ;
- `dT` représente l'incrément de changement de température (constante sur le pas) ;
- pour toute variable interne `Y`, `Y` représente sa valeur en début de pas ;
- pour toute variable interne `Y`, `dY` représente l'incrément de cette variable sur le pas, incrément qu'il faut calculer ;
- pour toute variable externe `V`, `V` représente sa valeur en début de pas ;
- pour toute variable externe `V`, `dV` représente son incrément de variation sur le pas de temps (constante sur le pas).

Calcul d'une matrice de raideur La loi de comportement peut éventuellement fournir une matrice de raideur pour réaliser les itérations de la résolution globale. Pour que les interfaces puissent gérer cette possibilité, l'analyseur `DefaultParser` fournit deux mots clés :

- `@ProvidesTangentOperator` indique que la loi fournit une matrice tangente non symétrique ;
- `@ProvidesSymmetricTangentOperator` indique que la loi fournit une matrice tangente symétrique ;

Si le code appelant demande le calcul de la matrice tangente cohérente, la variable booléenne `computeTangentOperator_` est vraie.

Le type de matrice demandé est stocké dans la variable `smt` (`stiffness matrix type`). L'utilisateur doit tester sa valeur et effectuer le calcul le cas échéant. Sa valeur peut être :

- `ELASTIC`, pour la matrice d'élasticité (matrice d'élasticité) ;
- `SECANTOPERATOR`, pour la matrice sécante (matrice d'élasticité endommagée) ;
- `TANGENTOPERATOR`, pour la matrice tangente ;
- `CONSISTENTTANGENTOPERATOR`, pour la matrice tangente cohérente ;

Matrice de prédiction Il n'est aujourd'hui pas possible de préciser une matrice de prédiction avec cet analyseur.

5 INTÉGRATION DES LOIS DE COMPORTEMENT PAR UNE MÉTHODE EXPLICITE (ALGORITHME DE RUNGE-KUTTA)

Cette section décrit l'analyseur `RungeKutta` qui permet l'intégration des lois de comportement, formulées en vitesse, par une méthode explicite.

Contrairement aux analyseurs spécifiques décrits dans la section 3, aucune hypothèse n'est faite sur les lois de comportement présentées : l'analyseur `RungeKutta` est dit générique.

Nous supposons que l'utilisateur a réussi à exprimer la loi de comportement sous la forme d'un système différentiel :

$$\dot{Y} = G\left(Y, t\right) \tag{34}$$

où G est une fonction *a priori* non linéaire et que nous supposerons *a minima* continûment dérivable. Cette fonction se construit en rassemblant les équations régissant les différentes variables d'états.

Y un vecteur regroupant les différentes variables internes :

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix}$$

Cette écriture est *symbolique* et chaque terme y_j peut représenter une variable interne qui peut être soit un scalaire soit un tenseur symétrique d'ordre 2. L'analyseur `RungeKutta` impose que le premier terme de ce vecteur soit la déformation élastique, dont la variable associée est `eel`, qui est automatiquement déclarée. Le nom de glossaire de cette variable est `ElasticStrain`.

La dépendance en temps de la fonction G qui apparaît dans l'équation (34) désigne en réalité une dépendance à la variation de certaines variables *externes* qui influencent la loi de comportement. Des exemples de telles variables externes sont :

- le taux de combustion ;
- la taille de grain ;
- la densité de fission ;
- etc...

Nous détaillons les méthodes d'intégration proposées par l'analyseur `RungeKutta`. Nous donnons en même temps les directives `mfront` qui permettent de programmer le système différentiel et de modifier le comportement des algorithmes.

5.1 LES ALGORITHMES DE RUNGE-KUTTA

Les méthodes de RUNGE-KUTTA désignent une famille d'algorithmes telles que :

$$Y|_{t+\Delta\, t} = Y|_t + \sum_{i=1}^{n} b_i k_i$$

où :
$$\begin{cases} k_1 = \Delta\, t\, G\left(Y|_t, t\right) \\ k_2 = \Delta\, t\, G\left(Y|_t + a_{21}\, k_1, t + c_2\, \Delta\, t\right) \\ k_3 = \Delta\, t\, G\left(Y|_t + a_{31}\, k_1 + a_{32}\, k_2, t + c_3\, \Delta\, t\right) \\ \quad \vdots \\ k_n = \Delta\, t\, G\left(Y|_t + a_{n1}\, k_1 + \ldots + a_{n,n-1}\, k_{n-1}, t + c_n\, \Delta\, t\right) \end{cases}$$

Une méthode particulière est caractérisée par la donnée du nombre d'étapes n et des différents coefficients a_{ij}, b_i et c_i.

L'un des intérêts des méthodes de RUNGE-KUTTA est que les coefficients k_i d'une méthode d'ordre élevée peuvent parfois être utilisés pour construire une méthode d'ordre moins élevée. On obtient ainsi deux estimations de la valeur de fin de pas qui peuvent être comparées : si la comparaison entre les deux estimations n'est pas satisfaisante, on en déduit que le pas de temps utilisé est trop grand et il est possible de mettre en place une stratégie de redécoupage du pas de temps.

Pas de temps minimal En cas de convergence forcée [Pascal 05], le code aux éléments finis `Cast3M` envoie à la loi de comportement un pas de temps nul qui ne permet pas d'écrire les lois de comportement sous la forme d'un système différentiel. Il est nécessaire de préciser un pas de temps minimal par la directive `@MinimalTimeStep` pour gérer ce cas. À l'exécution, la valeur choisie peut être modifiée à l'aide du paramètre `dtmin`.

5.2 ALGORITHMES DISPONIBLES

Argument	Algorithme associé
`Euler`	algorithme d'EULER
`rk2`	algorithme de RUNGE-KUTTA d'ordre 2
`rk4`	algorithme de RUNGE-KUTTA d'ordre 4
`rk42`	algorithme avec contrôle de l'erreur et adaptation du pas de temps basé sur des méthodes de RUNGE-KUTTA d'ordre 4 et d'ordre 2
`rk54`	algorithme avec contrôle de l'erreur et adaptation du pas de temps basé sur des méthodes de RUNGE-KUTTA d'ordre 5 et d'ordre 4
`rkCastem`	algorithme avec contrôle de l'erreur et adaptation du pas de temps extrait du code `Cast3M`

TABLEAU 1 : Arguments possibles de la directive `@Algorithm`.

Différents algorithmes de RUNGE-KUTTA sont disponibles. L'utilisateur peut choisir l'algorithme utilisé par la directive `@Algorithm`. Les arguments possibles de cette directive sont regroupés au tableau 1.

5.2.1 Algorithmes sans contrôle du pas de temps

`mfront` permet d'utiliser un certain nombre d'algorithmes de RUNGE-KUTTA sans contrôle du pas de temps. L'utilisation de ces algorithmes n'est pas recommandée car leurs prédictions sont généralement peu satisfaisantes : il est préférable d'utiliser les algorithmes avec contrôle de l'erreur et pas de temps adaptatif.

Méthode d'EULER explicite L'estimation la plus simple consiste à remplacer la dérivée $\dot{Y}$ par la différence finie $\frac{Y|_{t+\Delta t} - Y|_t}{\Delta t}$, ce qui conduit à :

$$Y|_{t+\Delta t} = Y|_t + \Delta t\, G\left(Y|_t, t\right) \tag{35}$$

Cette estimation, dite *explicite*, est connue pour être à la fois peu précise et peu robuste. L'erreur de cette méthode est (asymptotiquement) proportionnelle au pas de temps : elle est dite d'ordre 1.

Méthode de RUNGE-KUTTA d'ordre 2 Les méthodes de RUNGE-KUTTA permettent d'améliorer la précision du schéma d'EULER. Pour obtenir une méthode d'ordre 2, la valeur $Y|_{t+\Delta\,t}$ est recherchée de la forme :

$$Y|_{t+\Delta\,t} = Y|_t + a_1 \Delta\,t\,G\left(Y|_t, t\right) + a_2 \Delta\,t\,G\left(Y|_t + a_4 \Delta\,t, t + a_3 \Delta\,t\right)$$

Les quatre coefficients $a_i|_{i\in[0:4]}$ sont choisis pour que l'expression précédente coïncide avec le développement de TAYLOR à l'ordre 2 de la fonction G. Cette contrainte conduisant à un système de 3 équations à 4 inconnues, plusieurs choix sont possibles.

Le choix fait dans `mfront` est dite « du point milieu ». Ce choix correspond à la solution $a_1 = 0$, $a_2 = 1$, $a_3 = 1/2$ et $a_4 = G\left(Y_i, t\right)/2$. Cette méthode s'écrit ainsi :

$$Y|_{t+\Delta\,t} = Y|_t + \Delta\,t\,G\left(Y|_t + \frac{\Delta\,t}{2} G\left(Y|_t, t\right), t + \frac{\Delta\,t}{2}\right) \tag{36}$$

Méthode de RUNGE-KUTTA d'ordre 4 La méthode de RUNGE-KUTTA d'ordre 4 retenue dans `mfront` est donnée par :

$$Y|_{t+\Delta\,t} = Y|_t + \frac{1}{6}\left(k_1 + 2\,k_2 + 2\,k_3 + k_4\right) \tag{37}$$

où :
$$\begin{cases} k_1 = \Delta\,t\,G\left(Y|_t, t\right) \\ k_2 = \Delta\,t\,G\left(Y|_t + \frac{1}{2}\,k_1, t + \frac{1}{2}\,\Delta\,t\right) \\ k_3 = \Delta\,t\,G\left(Y|_t + \frac{1}{2}\,k_2, t + \frac{1}{2}\,\Delta\,t\right) \\ k_4 = \Delta\,t\,G\left(Y|_t + k_3, t + \Delta\,t\right) \end{cases}$$

5.2.2 Algorithmes avec contrôle du pas de temps (correcteur/prédicteur)

`mfront` propose trois algorithmes de RUNGE-KUTTA avec sous-découpage du pas de temps pour garantir la qualité de la solution trouvée. Pour cela, on introduit un temps local t^l, compris entre$t$ et $t + \Delta\,t$, et un pas de temps local $\delta\,t^l$ variable permettant d'augmenter t^l.

Méthode de RUNGE-KUTTA d'ordre $4/2$ Les algorithmes (36) et (37) partagent les mêmes coefficients k_1 et k_2 : si l'on utilise une méthode d'ordre 4, on a « gratuitement » une estimation d'ordre 2. La différence entre ces deux estimations est en $O(3)$ et peut servir à contrôler le pas de temps.

Méthode de RUNGE-KUTTA d'ordre $5/4$ (méthode de FEHLBERG) Pour décrire cette méthode, utilisée par défaut, nous reprenons la présentation d'A. FORTIN [Fortin 01]. Il est utile d'introduire les 6 constantes sui-

vantes :

$$
\begin{aligned}
k_1 &= \Delta t \cdot G\left(Y_i|_t, t\right) \\
k_2 &= \Delta t \cdot G\left(Y_i|_t + \frac{1}{4}\,k_1, t + \frac{1}{4}\,\Delta t\right) \\
k_3 &= \Delta t \cdot G\left(Y_i|_t + \frac{3}{32}\,k_1 + \frac{9}{32}\,k_2, t + \frac{3}{8}\,\Delta t\right) \\
k_4 &= \Delta t \cdot G\left(Y_i|_t + \frac{1932}{2197}\,k_1 - \frac{7200}{2197}\,k_2 + \frac{7296}{2197}\,k_3, t + \frac{12}{13}\,\Delta t\right) \\
k_5 &= \Delta t \cdot G\left(Y_i|_t + \frac{439}{216}\,k_1 - 8\,k_2 + \frac{3680}{513}\,k_3 - \frac{845}{4104}\,k_4, t + \Delta t\right) \\
k_6 &= \Delta t \cdot G\left(Y_i|_t - \frac{8}{27}\,k_1 + 2\,k_2 - \frac{3544}{2565}\,k_3 + \frac{1859}{4104}\,k_4 - \frac{11}{40}\,k_5, t + \frac{1}{2}\Delta t\right)
\end{aligned}
$$

Ces coefficients permettent de construire une méthode de RUNGE-KUTTA d'ordre 4 :

$$Y_i|_{t+\Delta t} = Y_i|_t + \left(\frac{25}{216}\,k_1 + \frac{1408}{2565}\,k_3 + \frac{2197}{4104}\,k_4 - \frac{1}{5}\,k_5\right)$$

et une autre d'ordre 5 :

$$Y_i|_{t+\Delta t} = Y_i|_t + \left(\frac{16}{135}\,k_1 + \frac{6656}{12\,825}\,k_3 + \frac{28\,561}{56\,430}\,k_4 - \frac{9}{50}\,k_5 + \frac{2}{55}\,k_6\right)$$

La différence entre ces deux estimations est en $O(5)$ et peut servir à contrôler le pas de temps.

Méthode `Cast3M` Une méthode de RUNGE-KUTTA particulière est disponible dans `Cast3M`. Cette méthode se distingue par sa gestion du pas de temps et son critère d'arrêt qui sont uniquement basés sur la différence des contraintes entre deux ordres d'approximations (et non sur les variables internes). Cette méthode est décrite en annexe D.

5.2.3 Gestion du pas de temps

Les méthodes de RUNGE-KUTTA d'ordre $4/2$ et d'ordre $5/4$ permettent de comparer les estimations fournies par deux méthodes d'ordre différents. La différence entre ces deux estimations mesure l'erreur d'intégration et est comparée à une certaine tolérance ε :

$$\left|\left|Y^{(a)} - Y^{(b)}\right|\right|_{\text{RK}} < \varepsilon \quad \text{avec} \tag{38}$$

Deux mesures de l'erreur sont utilisées. Pour un nombre restreint de variables internes[13], la mesure utilisée est la suivante :

$$\left|\left|Y^{(a)} - Y^{(b)}\right|\right|_{\text{RK}} < \varepsilon \quad \text{avec} \quad \left|\left|Y^{(a)} - Y^{(b)}\right|\right|_{\text{RK}} = \frac{1}{\sum_{i=1}^{N} N_{y_i}} \sum_{i=1}^{N} \frac{\left|y_i^{(a)} - y_i^{(b)}\right|}{E_{y_i}} \tag{39}$$

Pour des systèmes différentiels de plus grandes tailles, la norme suivante est utilisée :

$$\left|\left|Y^{(a)} - Y^{(b)}\right|\right|_{\text{RK}} = \frac{1}{\sum_{i=1}^{N} N_{y_i}} \sum_{i=1}^{N} \frac{\left|y_i^{(a)} - y_i^{(b)}\right|}{E_{y_i}} \tag{40}$$

13. Cette mesure est utilisée si la taille du vecteur Y est inférieure à 20 en $1D$.

Dans ces deux expressions, nous avons noté :

- N est le nombre de variables internes ;
- N_{y_i} le nombre de composantes de la variable interne y_i ;
- $y_i^{(a)}$ est l'estimation de la variable interne en fin de pas de temps obtenue par la première méthode ;
- $y_i^{(b)}$ est l'estimation de la variable interne en fin de pas de temps obtenue par la seconde méthode ;
- E_{y_i} est un paramètre de normalisation associé à la variable E_{y_i}, si l'utilisateur en a définit une par la méthode `setErrorNormalisationFactor`. Ce paramètre permet de normaliser toutes les variables à quelque chose de l'ordre des déformations. Par exemple, si la variable y_i est de l'ordre des contraintes, il est possible de la normaliser ainsi :

  ```
  yi.setErrorNormalisationFactor(young);
  ```

  L'argument de la méthode `setErrorNormalisationFactor` peut être soit une valeur scalaire soit une propriété matériau.

Si l'inégalité (38) n'est pas respectée, le pas de temps $\delta\, t^l$ est sous-découpé automatiquement. Inversement, si elle est respectée, le pas de temps $\delta\, t^l$ est augmenté.

Supposons que l'erreur soit en $O(n)$, un facteur β d'augmentation ou de réduction du pas de temps peut être proposé ainsi :

$$\beta = \left(\frac{\varepsilon}{\left|\left| Y^{(a)} - Y^{(b)} \right|\right|_{\text{RK}}} \right)^{\frac{1}{n}}$$

L'augmentation ou la réduction du pas de temps local est limité à un facteur 10. Naturellement, le pas de temps local $\delta\, t^l$ ne peut être supérieur à $t + \Delta\, t - t^l$.

Gestion par l'utilisateur Si le code suivant la directive `@Derivative`, présenté dans la suite et qui sert à programmer le système différentiel, renvoie la valeur booléenne `false`, le pas de temps est automatiquement divisé par 2.

Cette possibilité est en pratique assez importante, en particulier pour des comportements très raides. Il est par exemple fréquent de rencontrer des lois viscoplastique de type NORTON dont l'exposant est grand. Dans ces cas, les premières estimations des solutions conduisent à des contraintes équivalentes assez importantes qui conduisent à des vitesses d'écoulement infinies [14]. Informatiquement, ces vitesses se traduisent par l'apparition de la valeur flottante `NaN` [15]. Les algorithmes de RUNGE-KUTTA captent ce cas et sous-découpent alors automatiquement le pas de temps, mais cette solution présente l'inconvénient de ne pas être compatible avec certaines techniques de déverminage extrêmement utiles. Il est dès lors préférable, et également plus efficace puisque des opérations lourdes (calculs de puissance) peuvent être évitées, que l'utilisateur introduise des tests appropriés.

Choix de la tolérance La tolérance ε est fixée par défaut à 10^{-8}. L'utilisateur peut modifier cette valeur par la directive `@Epsilon`. À l'exécution, l'utilisateur peut modifier cette valeur à l'aide du paramètre `epsilon`.

14. Ceci est particulièrement vrai quand on utilise le système international où les contraintes s'expriment en Pascal. Une autre manière de résoudre le problème est de réécrire la loi de NORTON différemment. Au lieu d'écrire $\dot{p} = A\, \sigma_{eq}^n$, il faudrait mieux écrire $\dot{p} = \dot{p}_0 \left(\frac{\sigma_{eq}}{\sigma_0} \right)^n$ avec $\dot{p}_0 = A\, \sigma_{eq}^0$.

15. Acronyme de « Not a Number »

5.2.4 Critère d'arrêt

L'intégration s'arrête quand l'une des conditions suivantes est vérifiée :

- le temps local devient trop petit, ce qui marque l'échec de l'algorithme. Le pas de temps minimal admissible est égal au pas de temps total Δt multiplié par 100 fois la précision machine du type réel considéré ;
- le temps local est égal au temps de fin de pas, ce qui marque la réussite de l'intégration. Ceci est vérifié par :

$$\left|t + \Delta t - t^l\right| < \frac{\delta t^l}{2}$$

5.3 LA DIRECTIVE @COMPUTESTRESS

La directive `@ComputeStress` permet de calculer les contraintes aux différentes étapes de la méthode et en fin de calcul. Les contraintes sont supposées fonction des valeurs actuelles des variables internes.

Conventions spécifiques aux différentes étapes de la méthode Dans la méthode `@ComputeStress`, les conventions suivantes s'appliquent :

- `sig` représente la contrainte qu'il faut calculer ;
- `eto` représente la déformation totale actualisée ;
- `deto` représente la vitesse de déformation totale (constante sur le pas) ;
- `T` représente la valeur actualisée de température ;
- `dT` représente la vitesse de changement de température (constante sur le pas) ;
- pour toute variable interne `Y`, `Y` représente sa valeur actualisée ;
- pour toute variable interne auxiliaire `Y`, `Y` représente sa valeur en *début* de pas de temps ;
- pour toute variable externe `V`, `V` représente sa valeur actualisée ;
- pour toute variable externe `V`, `dV` représente sa vitesse de variation sur le pas de temps (constante sur le pas).

Conventions spécifiques en fin d'intégration Dans la méthode `@ComputeStress`, les conventions suivantes s'appliquent :

- `sig` représente la contrainte qu'il faut calculer ;
- `eto` représente la déformation totale en fin de pas ;
- `deto` représente la vitesse de déformation totale (constante sur le pas) ;
- `T` représente la valeur de la température en fin de pas ;
- `dT` représente la vitesse de changement de température (constante sur le pas) ;
- pour toute variable interne `Y`, `Y` représente sa valeur en fin de pas ;
- pour toute variable interne auxiliaire `Y`, `Y` représente sa valeur en *début* de pas de temps ;
- pour toute variable externe `V`, `V` représente sa valeur en fin de pas ;
- pour toute variable externe `V`, `dV` représente sa vitesse de variation sur le pas de temps (constante sur le pas).

5.4 LA DIRECTIVE @DERIVATIVE

La directive `@Derivative` permet de préciser les équations différentielles associées aux différentes variables internes.

Le code fourni est appelé après celui fourni par la méthode `@ComputeStress`.

Conventions spécifiques Dans la méthode `@Derivative`, les conventions suivantes s'appliquent :

- `sig` représente la valeur actualisée de la contrainte ;
- `eto` représente la déformation totale actualisée ;
- `deto` représente la vitesse de déformation totale (constante sur le pas) ;
- `T` représente la valeur actualisée de température ;
- `dT` représente la vitesse de changement de température (constante sur le pas) ;
- pour toute variable interne `Y`, `Y` représente sa valeur actualisée ;
- pour toute variable interne auxiliaire `Y`, `Y` représente sa valeur en *début* de pas de temps ;
- pour toute variable externe `V`, `V` représente sa valeur actualisée ;
- pour toute variable externe `V`, `dV` représente sa vitesse de variation sur le pas de temps (constante sur le pas).

Pour toute variable interne `Y`, `dY` représente sa vitesse de variation : c'est cette variable que l'utilisateur doit renseigner.

5.5 Mise à jour des variables auxiliaires

Les variables auxiliaires sont mises à jour à chaque fois qu'un pas est jugé valide au sens du critère d'arrêt (38) après la mise à jour des variables internes.

Cette méthode peut donc être appelée plusieurs fois si un algorithme à pas de temps adaptatif est utilisé. Une variable spécifique, nommée `dt_`, permet de récupérer l'incrément de temps effectivement utilisé par l'algorithme. Le pas de temps total est donné par la variable `dt`.

5.6 Paramètres automatiquement définis

L'analyseur implicite définit automatiquement différents paramètres :

- `epsilon`, la valeur du critère d'arrêt de l'algorithme de Runge-Kutta. La valeur par défaut de ce paramètre est de 10^{-8} et peut être modifiée par la directive `@Epsilon`. Ce paramètre doit avoir une valeur strictement positive ;
- `dtmin`, la valeur minimale du pas de temps utilisé pour l'intégration [16]. Par défaut, ce paramètre n'existe que si la directive `@MinimalTimeStep` est utilisée. Ce paramètre doit avoir une valeur strictement positive.

5.7 Notes sur l'utilisation des tableaux de variables internes

L'utilisation des tableaux de variables internes dans le cas des algorithmes de Runge-Kutta mérite quelques précisions.

Pour réduire le temps de développement de cette fonctionnalité et surtout réduire les sources d'erreurs, nous avons utilisé des classes livrés avec TFEL (la librairie mathématique sur laquelle se base `mfront`). En fonction de la taille des tableaux, deux types de vecteurs sont utilisés :

- pour des tableaux de taille inférieure à 10, des vecteurs de taille finie, alloués sur la pile, sont utilisés ;
- pour des tableaux plus grands, des vecteurs alloués dynamiquement ont été utilisés.

Dans les deux cas, les opérations mathématiques usuelles ont été définies : l'addition d'un vecteur de tenseur est possible et optimisée.

Ainsi, la génération des différents algorithmes de Runge-Kutta est restée inchangée, ce qui est satisfaisant.

16. Ce paramètre a été introduit essentiellement pour contourner les difficultés liées à l'algorithme de convergence forcée du code `Cast3M` qui envoie, lorsqu'il s'active, un incrément de temps nul à la loi de comportement.

5.8 INTÉGRATION D'UNE LOI D'ÉCOULEMENT VISCOPLASTIQUE ORTHOTROPE PAR UN ALGORITHME DE RUNGE-KUTTA

Nous nous intéressons maintenant à l'écriture d'une loi de comportement viscoplastique orthotrope. La forme retenue est proche d'une loi de NORTON, la vitesse de l'écoulement étant donnée par :

$$\underline{\dot{\epsilon}}^{vis} = A\ (\sigma_H)^E\ \underline{\mathbf{n}}$$

où $sigmaH$ désigne la contrainte de HILL et $\underline{\mathbf{n}}$ le tenseur normal. Ils sont définis en annexe au paragraphe E.3.

Il est d'usage d'introduire une déformation viscoplastique cumulée p dont la vitesse est donnée par :

$$\dot{p} = A\,(\sigma_H)^E$$

Le système d'équations complet est donc :

$$\begin{aligned} \dot{p} &= A\,(\sigma_H)^E \\ \underline{\dot{\epsilon}}^{vis} &= \dot{p}\,\underline{\mathbf{n}} \\ \underline{\dot{\epsilon}}^{el} &= \underline{\dot{\epsilon}}^{to} - \underline{\dot{\epsilon}}^{vis} \end{aligned} \tag{41}$$

Implantation La figure 3 présente l'implantation de ce système. Le calcul du tenseur de HILL a été présenté au paragraphe C.3. Rappelons simplement qu'il est nécessaire d'inclure le fichier d'entête `TFEL/MaterialLaw/Hill.hxx` dans la directive `@Includes`. Le calcul du tenseur normal est immédiat d'après les formules données au paragraphe E.3.

Deux remarques peuvent être faites :

- les intégrations de type RUNGE-KUTTA conduisent souvent à des prédictions initiales élevées des contraintes. Nous testons ici si la contrainte de HILL est supérieure à $10^9\,MPa$: dans ce cas, nous renvoyons la valeur `false` qui entraîne un découpage du pas de temps ;
- cette implantation tient compte des limitations du code aux éléments finis `Cast3M` qui ne permet pas d'avoir une définition homogène des directions d'orthotropie quelque soit l'hypothèse de modélisation. Les conventions utilisées ici sont conformes à celles décrites en annexe E pour les tubes.

```
@Parser     RungeKutta;
@Behaviour OrthotropicCreep;
@Author     Helfer Thomas;
@Algorithm rk54;
@Date       8/03/11;

@OrthotropicBehaviour;
@RequireStiffnessTensor;

@StateVariable Stensor evp; /* Viscoplastic strain            */
evp.setGlossaryName("ViscoplasticStrain");
@StateVariable real p;      /* Equivalent viscoplastic strain */
p.setGlossaryName("EquivalentViscoplasticStrain");

@Includes{
#include<TFEL/Material/Lame.hxx>
#include<TFEL/Material/Hill.hxx>
}

@ComputeStress{
  sig = D*eel;
}

@Derivative{
  const real H_rr = 0.371;
  const real H_tt = 1-H_rr;
  const real H_zz = 4.052;
  const real H_rt = 1.5;
  const real H_rz = 1.5;
  const real H_tz = 1.5;
  st2tost2<N,real> H;
  if((getModellingHypothesis()==ModellingHypothesis::PLANESTRESS)||
     (getModellingHypothesis()==ModellingHypothesis::PLANESTRAIN)||
     (getModellingHypothesis()==ModellingHypothesis::GENERALISEDPLANESTRAIN)){
    H = hillTensor<N,real>(Hzz,Hrr,Htt,
      Hrt,Hrz,Htz);
  } else {
    H = hillTensor<N,real>(Htt,Hrr,Hzz,
      Hrz,Hrt,Htz);
  }
  const real sigeq = sqrt(sig|H*sig);
  if(sigeq>1e9){
    return false;
  }
  Stensor  n(0.);
  if(sigeq > 10.e-7){
    n    = H*sig/sigeq;
  }
  dp   = 8.e-67*pow(sigeq,8.2);
  devp = dp*n;
  deel = deto - devp;
}
```

FIGURE 3 : Implantation d'une loi d'écoulement viscoplastique orthotrope par une méthode de RUNGE-KUTTA

6 INTÉGRATION DES LOIS DE COMPORTEMENT PAR UNE MÉTHODE IMPLICITE

Nous décrivons dans cette section l'intégration des lois de comportement par une méthode implicite. Ces méthodes sont très efficaces, stables et peuvent être utilisées pour intégrer toutes les lois de comportement.

Elles présentent de nombreux avantages par rapport aux autres méthodes. Les plus importants sont que :
- ces méthodes introduisent naturellement la matrice tangente cohérente. De plus, dans la plupart des cas, cette matrice est pratiquement « gratuite » (voir paragraphe 6.6.1) ;
- elles permettent de traiter naturellement les lois de comportement indépendantes du temps. Un exemple est donné par le traitement de la plasticité dans les analyseurs spécifiques (voir paragraphe 3).

Ces méthodes supposent que l'on soit capable de résumer l'intégration de la loi de comportement à la résolution d'un système non-linéaire. Nous avons détaillé comment procéder pour les analyseurs spécifiques (voir paragraphe 3). Nous montrons dans le paragraphe suivant comment procéder pour transformer un système différentiel en système d'équations non linéaires et nous en profitons pour introduire les méthodes implicites en se basant sur les méthodes de RUNGE-KUTTA.

Deux analyseurs, nommés `Implicit` et `ImplicitII`, sont disponibles pour intégrer les lois de comportement par une méthode implicite. Ils se distinguent par le fait que l'analyseur `Implicit` déclare automatiquement la déformation élastique comme première variable interne [17].

Différentes méthodes pour résoudre le système non-linéaire introduit par la méthode implicite sont disponibles :
- `NewtonRaphson`, l'algorithme classique de NEWTON-RAPHSON ;
- `NewtonRaphson_NumericalJocabian`, l'algorithme classique de NEWTON-RAPHSON avec une approximation numérique du jacobien ;
- `Broyden`, premier algorithme de BROYDEN ;
- `BroydenII`, second algorithme de BROYDEN.

L'utilisateur choisi l'algorithme par la directive `@Algorithm`.

Outre la présentation des bases mathématiques de ces méthodes, nous profitons de ce paragraphe pour justifier certaines initialisations implicites faites dans `mfront` qui rendent l'écriture des lois de comportement plus concise.

6.1 RÉSOLUTION D'UN SYSTÈME DIFFÉRENTIEL PAR UNE MÉTHODE IMPLICITE

6.1.1 Généralités

Soit à résoudre le système différentiel suivant :

$$\dot{Y} = G\left(Y, t\right) \tag{42}$$

où G est une fonction *a priori* non linéaire et que nous supposerons *a minima* continûment dérivable.

La dépendance en temps évoquée ici dans la fonction G désigne en réalité une dépendance à la variation de certaines variables *externes* qui influencent la loi de comportement. Des exemple de telles variables externes sont :
- le taux de combustion ;
- la taille de grain ;
- la densité de fission ;
- etc...

17. Nous verrons plus loin une justification de ce choix : il permet un calcul quasi-automatique de la matrice tangente cohérente.

Y un vecteur regroupant les différentes variables internes :

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix}$$

Cette écriture est *symbolique* et chaque terme y_j peut représenter une variable interne qui peut être soit un scalaire soit un tenseur symétrique d'ordre 2.

L'analyseur `Implicit` impose que le premier terme de ce vecteur soit la déformation élastique, dont la variable associée est `eel`, qui est automatiquement déclarée[18].

La résolution numérique de cette équation consiste à estimer, à partir de la valeur $Y|_t$ à un instant t, la valeur $Y|_{t+\Delta t}$ à un instant $t + \Delta t$.

Méthode d'EULER L'estimation la plus simple consiste à remplacer la dérivée $\dot{Y}$ par la différence finie $\dfrac{Y|_{t+\Delta t} - Y|_t}{\Delta t}$, ce qui conduit à :

$$Y|_{t+\Delta t} = Y|_t + \Delta t\, G\left(Y|_t, t\right) \tag{43}$$

Cette estimation, dite *explicite*, est connue pour être à la fois peu précise et peu robuste. L'erreur de cette méthode est (asymptotiquement) proportionnelle au pas de temps : elle est dite d'ordre 1.

Méthode de RUNGE-KUTTA (explicite) d'ordre 2 Les méthodes de RUNGE-KUTTA permettent d'améliorer la précision du schéma d'EULER. Pour obtenir une méthode d'ordre 2, la valeur $Y|_{t+\Delta t}$ est recherchée de la forme :

$$Y|_{t+\Delta t} = Y|_t + a_1 \Delta t\, G\left(Y|_t, t\right) + a_2 \Delta t\, G\left(Y|_t + a_4 \Delta t, t + a_3 \Delta t\right)$$

Les quatre coefficients $a_i|_{i\in[0:4]}$ sont choisis pour que l'expression précédente coïncide avec le développement de TAYLOR à l'ordre 2 de la fonction G. Cette contrainte conduisant à un système de 3 équations à 4 inconnues, plusieurs choix sont possibles.

Méthode du point milieu Ce choix correspond à la solution $a_1 = 0$, $a_2 = 1$, $a_3 = {}^1\!/_2$ et $a_4 = {}^{G\left(Y_i, t\right)}\!/_2$. Cette méthode s'écrit ainsi :

$$Y|_{t+\Delta t} = Y|_t + \Delta t\, G\left(Y|_t + \frac{\Delta t}{2} G\left(Y|_t, t\right), t + \frac{\Delta t}{2}\right) \tag{44}$$

Méthode d'EULER modifiée Ce choix correspond à la solution $a_1 = a_2 = {}^1\!/_2$, $a_3 = 1$ et $a_4 = G\left(Y_i, t\right)$. Cette méthode s'écrit ainsi :

$$Y|_{t+\Delta t} = Y|_t + \frac{\Delta t}{2}\left[G\left(Y|_t, t\right) + G\left(Y|_t + \Delta t\, G\left(Y|_t, t\right), t + \Delta t\right)\right] \tag{45}$$

18. Le nom de glossaire de cette variable est `ElasticStrain`. La notion de nom de glossaire est décrite dans la première partie de la documentation de `mfront` [Helfer 13d].

Estimation implicite Des estimations dites *implicites* de la solution consistent à introduire, à la place de l'estimation $Y|_t + \Delta t\, G\left(Y|_t, t\right)$, la valeur recherchée $Y|_{t+\Delta t}$. Deux schémas implicites, se déduisent alors des schémas de RUNGE-KUTTA précédent.

À partir de la méthode du point milieu (44), nous obtenons :

$$Y|_{t+\Delta t} = Y|_t + \Delta t\, G\left(\frac{1}{2}\left(Y|_t + Y|_{t+\Delta t}\right), t + \frac{\Delta t}{2}\right)$$

À partir de la méthode d'EULER modifiée (45), nous obtenons :

$$Y|_{t+\Delta t} = Y|_t + \frac{\Delta t}{2}\left[G\left(Y|_t, t\right) + G\left(Y|_{t+\Delta t}, t + \Delta t\right)\right]$$

θ-méthodes Les schémas implicites précédents sont généralement présentés en introduisant un paramètre θ, compris entre 0 et 1, dans les expressions précédentes afin de rendre les apports des termes $Y|_t$ et $Y|_{t+\Delta t}$ dissymétriques.

À partir de la méthode du point milieu (44), nous obtenons :

$$Y|_{t+\Delta t} = Y|_t + \Delta t\, G\left(\left(1-\theta\right) Y|_t + \theta\ Y|_{t+\Delta t}, t + \theta\, \Delta t\right) \tag{46}$$

La seconde forme, obtenue à partir de la méthode d'EULER modifiée (45), nous obtenons :

$$Y|_{t+\Delta t} = Y|_t + \Delta t\ \left[\left(1-\theta\right) G\left(Y|_t, t\right) + \theta\, G\left(Y|_{t+\Delta t}, t + \Delta t\right)\right]$$

Cette seconde forme d'équation implicite est rarement utilisée, au moins pour l'intégration des lois de comportement mécaniques.

6.2 MÉTHODE IMPLICITE UTILISÉE DANS MFRONT

`mfront` tend, suivant l'usage, à supporter naturellement la formulation (46), mais rien n'interdit d'utiliser la première. Pour simplifier la présentation, nous ne décrirons le principe de la résolution que pour la première forme.

L'équation (46) peut s'écrire :

$$F\left(\Delta Y\right) = \Delta Y - \Delta t\, G\left(Y|_t + \theta \Delta Y, t + \theta\, \Delta t\right) = 0 \tag{47}$$

où nous avons introduit l'incrément ΔY de la variable Y au cours du pas de temps Δt, variable qui apparaît comme l'inconnue « naturelle » de cette équation. Nous avons donc ramené la résolution du système différentiel (42) à la recherche d'un zéro de la fonction $F\left(\Delta Y\right)$.

Convention d'écriture Il est d'usage d'introduire des notations (et des abus de langage) pratiques.

Si dans le système différentiel initial, une fonction f des variables internes $f\left(Y\right)$ est présente, alors l'expression $f\left(Y + \theta\, \Delta Y\right)$ apparaîtra dans le système d'équations.

Par abus de langage, nous dirons que $Y + \theta\, \Delta Y$ est la valeur en milieu de pas de Y, que nous noterons $Y|_{t+\theta\, \Delta t}$:

$$Y|_{t+\theta\, \Delta t} = Y + \theta\, \Delta Y$$

Cette notation s'étend naturellement aux composantes de Y :

$$y_i|_{t+\theta\, \Delta t} = y_i + \theta\, \Delta y_i$$

En particulier, ces notations donnent un sens à l'expression $\frac{\partial\, y_i|_{t+\theta\,\Delta\,t}}{\partial \Delta\, y_i}$ qui est égale à θ.

De même, nous dirons que $f\left(Y+\theta\,\Delta\,Y\right)$ est la valeur en milieu de pas de f et nous la noterons $f|_{t+\theta\,\Delta\,t}$:

$$f|_{t+\theta\,\Delta\,t} = f\left(Y+\theta\,\Delta\,Y\right)$$

Décomposition du vecteur F L'écriture générique (42) résulte de la concaténation des équations différentielles régissant chaque variable interne Y. En pratique, le vecteur F est décomposé en termes associés à chaque variable interne :

$$F = \begin{pmatrix} f_{y_1} \\ \vdots \\ f_{y_i} \\ \vdots \\ f_{y_n} \end{pmatrix}$$

Cette écriture est là aussi *symbolique* et chaque terme fy_j peut être soit un scalaire soit un tenseur symétrique d'ordre 2.

`mfront` définit automatiquement les différents termes f_{y_j}. Ces termes se manipulent comme un scalaire si y_i est un scalaire, et comme une tenseur d'ordre 2 si y_i l'est.

Pour des raisons de performances, la variable f_{y_j} est l'image d'une portion du vecteur F et toute modification de cette variable modifie le vecteur F

Si y_i désigne un tableau de variables internes, f_{y_j} est une méthode qui prend en argument un indice entier.

Choix du paramètre θ L'optimum en terme d'efficacité numérique est obtenu pour $\theta = {}^1\!/_2$ (schéma dit « semi-implicite »), ce qui est le choix par défaut dans `mfront`. Le traitement de mécanismes indépendants du temps (plasticité ou endommagement) nécessite, pour être physiquement satisfaisant, d'utiliser la valeur 1 (schéma dit « purement implicite »). La valeur par défaut peut être modifiée par la directive `@Theta`. À l'exécution, elle peut être modifiée par le paramètre `theta`.

Méthodes mixtes Le choix du paramètre θ dépend du phénomène traité. Pour des lois mêlant différents types de phénomènes, certains auteurs choisissent de traiter les écoulements viscoplastiques par une méthode semi-implicite et les phénomènes indépendants du temps par une méthode purement implicite. Il est possible de faire de même dans `mfront` en prenant certaines précautions sur le calcul des contraintes.

Initialisation du vecteur F L'équation (47) montre que la fonction f_{y_i} comporte comme premier terme l'incrément $\Delta\, y_i$. `mfront` initialise donc le vecteur F à cette valeur avant chaque itération de la méthode de résolution.

6.2.1 Initialisation

Les méthodes de résolution convergent si l'initialisation de la recherche $\Delta\, Y^0$ est « suffisamment proche » de la solution. Dans le cas présent, l'équation (43) conduit à supposer que $\Delta\, Y$ ne sera qu'une faible correction de Y_t (pour des pas de temps suffisamment petits), de sorte qu'une estimation initiale nulle est généralement satisfaisante. Il s'agit du choix fait par défaut par `mfront` qui peut être modifié par l'utilisateur dans un bloc `@Predictor`. Un choix classique est d'utiliser l'estimation explicite de la solution :

$$\Delta\, Y^0 = \Delta\, t\, G\left(Y|_t, t\right)$$

6.2.2 Critère d'arrêt

Les algorithmes présentés s'arrêtent quand la différence entre deux estimations des incréments des déformations viscoplastiques cumulées est inférieure à un certain critère ε :

$$\frac{1}{\sum_{i=1}^{N} N_i} \sum_{i=1}^{N} |\Delta_n \Delta\, y_i| < \varepsilon$$

où $\Delta_n \Delta\, y_i$ désigne la différence entre l'estimation de l'incrément de la i^{e} variable à l'étape $n+1$ et de son estimation à l'étape n et N_i est le nombre de composantes de la variable interne y_i [19].

Notes Cette écriture du critère d'arrêt montre qu'il n'est valide que si les différentes variables internes sont du même ordre de grandeur. Ceci est également nécessaire pour le bon comportement numérique de la méthode de NEWTON. Dans le cas contraire, il est possible de normaliser les variables par la méthode `setNormalisationFactor`.

6.2.3 Avantages et inconvénients des méthodes implicites

Les méthodes implicites sont généralement opposées aux méthodes d'intégration plus ou moins basées sur des algorithmes de type RUNGE-KUTTA. Ces dernières méthodes présentent plusieurs avantages :

- un ordre de convergence qui peut être élevé ;
- une maîtrise de la précision des résultats par « correction/prédiction ». Cette maîtrise permet de sous-découper localement le pas de temps ;
- une facilité d'implantation par rapport aux méthodes implicites, ces méthodes ne nécessitant pas le calcul de matrice jacobienne notamment.

`mfront` propose un analyseur générique basé sur les algorithmes de type RUNGE-KUTTA (par défaut un correcteur-prédicteur $5/4$ à pas de temps adaptatif qui allie un ordre de convergence élevé à une efficacité numérique acceptable) décrit à la section 5.

Le principal défaut de ces méthodes est le nombre d'appels nécessaires à la fonction G du système (42), en particulier si une stratégie à pas de temps adaptatif est utilisée.

En comparaison, les méthodes implicites sont réputées beaucoup plus stables numériquement et ne nécessitent qu'un nombre faible d'appels à la fonction G. De plus, les méthodes implicites sont particulièrement adaptées aux lois de comportement indépendantes du temps (endommagement ou plasticité) où le système différentiel peut être remplacé par la nullité du critère plastique en fin de pas de temps, ce qui est précisément la condition *physique* à vérifier [20]. Cette particularité explique pourquoi les méthodes implicites sont toujours à préférer aux méthodes de type RUNGE-KUTTA dans le cas de lois de comportement indépendantes du temps, même si elles présentent sur le papier des ordres de convergence moins élevés.

L'application des méthodes implicites aux lois de comportement dépendantes du temps (écoulements viscoplastiques notamment) est plus délicate en raison justement de ces ordres de convergences plus faibles (que les méthodes de type RUNGE-KUTTA) [21].

En pratique, nous avons *toujours* constaté des résultats extrêmement satisfaisants tant en termes de précision qu'en termes de performance numérique avec les méthodes implicites, quel que soit le type de loi de comportement.

19. Pour les tenseurs, $|s|$ représente la somme des valeurs absolues des termes du tenseur (les composantes extra-diagonales ne sont comptées qu'une fois).

20. Nous avons traité le cas de la plasticité isotrope au paragraphe 3.1.2.

21. Il ne faut pas oublier qu'il ne sert généralement à rien d'utiliser des pas de temps trop grands. En effet, l'algorithme de résolution global (en quasi-statique) utilise une résolution implicite : sur un pas de temps, il faut que l'hypothèse qui permet la linéarisation de l'évolution mécanique de la structure soit vérifiée.

Méthode d'intégration	Temps CPU	Nombre d'itérations
Implicite	5 m 26 s	1817 itérations
RUNGE-KUTTA $5/4$	32 m 21 s	17493 itérations

TABLEAU 2 : Temps de calculs obtenus sur une éprouvette entaillée en traction modélisée en $2D\,(r,z)$. Calcul réalisé par J.-M. PROIX sur la version 11 de code `Aster`. La loi de comportement locale est la loi de HAYHURST [Proix 12a]. Cette comparaison a été fournie par J.-M. Proix (EDF/AMA).

Enfin, les méthodes implicites introduisent naturellement la matrice tangente cohérente, ce qui permet de réduire considérablement le nombre d'itérations nécessaires à la méthode globale. Ceci est illustré sur la figure 2.

Ceci explique que la plupart des analyseurs syntaxiques disponibles dans `mfront` soient basés sur ces méthodes.

Les deux principales faiblesses des méthodes implicites restent :
- la nécessité de calculer la matrice jacobienne ou d'utiliser des méthodes alternatives moins efficaces ;
- l'absence de contrôle de l'erreur.

6.3 MÉTHODE DE NEWTON-RAPHSON

Une méthode généralement utilisée pour trouver un zéro d'une fonction est la méthode itérative de NEWTON-RAPHSON.

Connaissant une estimation ΔY^n de la solution, la fonction F peut être approximée dans un voisinage ΔY^n par :

$$F\left(\Delta Y\right) = F\left(\Delta Y^n\right) + \left.\frac{\mathrm{d}F}{\mathrm{d}\Delta Y}\right|_{\Delta Y=\Delta Y^n}\left(\Delta Y - \Delta Y^n\right)$$

où $\dfrac{\mathrm{d}F}{\mathrm{d}\Delta Y}$ est la matrice jacobienne de la fonction F, qui sera notée J dans la suite.

Une nouvelle estimation ΔY^{n+1} est choisie en annulant cette approximation de F, ce qui conduit à l'expression suivante :

$$\Delta Y^{n+1} = \Delta Y^n - \left[J^{-1}\left(\Delta Y^n\right)\right]\, F\left(Y^n\right) \tag{48}$$

Dans cette équation, l'écriture $\left[J^{-1}\left(\Delta Y^n\right)\right]\, F\left(Y^n\right)$ représente la solution $\Delta^2 Y$ du système linéaire :

$$\left[J\left(\Delta Y^n\right)\right]\, \Delta^2 Y = F\left(Y^n\right)$$

Matrice jacobienne La matrice jacobienne s'exprime ainsi :

$$J = \frac{\mathrm{d}F}{\mathrm{d}\Delta Y} = I - \theta\Delta t\,\frac{\mathrm{d}G}{\mathrm{d}Y}\left(Y|_t + \theta\Delta Y, t + \theta\,\Delta t\right) \tag{49}$$

où I est la matrice identité.

Le calcul de la matrice jacobienne est généralement le point le plus difficile de la méthode implicite.

Calcul de la matrice jacobienne par blocs En pratique, la matrice J est calculée par blocs et prend la forme suivante :

$$J = \frac{\partial F}{\partial \Delta Y} = \begin{pmatrix} \frac{\partial f_{y_1}}{\partial \Delta y_1} & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \frac{\partial f_{y_i}}{\partial \Delta y_j} & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \frac{\partial f_{y_N}}{\partial \Delta y_N} \end{pmatrix}$$

Les différents termes $\frac{\partial f_{y_i}}{\partial \Delta y_j}$ sont automatiquement définies par `mfront`. La nature mathématique de ces termes dépend de la nature des variables y_i et y_j. Elles peuvent être manipulées par les opérations mathématiques usuelles. Pour des raisons de performances, ces variables sont des images de la matrice jacobienne : en les manipulant, on modifie directement la matrice jacobienne.

Ainsi, la dérivée d'une variable `a` par rapport à la variable `b` sera accessible par `dfa_ddb`.

Dans le cas particulier des tableaux de variables internes, `dfa_ddb` est une méthode qui prend :
- un unique argument entier si soit `a` ou soit `b` désigne un tableau de variables internes, mais pas les deux en mêmes temps ;
- deux arguments entiers si `a` et `b` désignent deux tableaux de variables internes ;

Ces méthodes renvoient des objets spéciaux, qui agissent comme des scalaires ou des tenseurs d'ordre 2 ou 4 suivant les cas. Ces objets permettent un accès aux valeurs de la jacobienne qui peut être, en fonction des capacités du compilateur utilisé, moins performant que dans le cas des variables ordinaires.

Comparaison de la matrice jacobienne fournie à une approximation numérique Le calcul de la matrice jacobienne est, à juste titre, jugé délicat et une erreur peut avoir un impact significatif sur les performances ou la robustesse de l'intégration et dans le meilleur des cas une non convergence [22].

Pour détecter les erreurs dans le calcul du jacobien, nous avons introduit le mot clé `@CompareToNumerical-Jacobian`. Ce mot clé déclenche un processus de vérification par comparaison du jacobien fournit par l'utilisateur à un jacobien évalué numériquement par perturbations. Cette vérification se fait par étapes :
- la première étape est le calcul d'un jacobien numérique par perturbations. Pour gagner en précision, une différence finie centrée est utilisée [23] ;
- la seconde étape est de comparer les termes du jacobien. La comparaison se fait bloc par bloc : la différence entre la dérivée $\frac{\partial f_{y_i}}{\partial \Delta y_j}$ calculée par l'utilisateur et son approximation numérique $\frac{\Delta f_{y_i}}{\Delta \Delta y_j}$ est évaluée ainsi :

  $$\frac{1}{N_{y_i} N_{y_j}} \left|\left| \frac{\partial f_{y_i}}{\partial \Delta y_j} - \frac{\Delta f_{y_i}}{\Delta \Delta y_j} \right|\right|$$

  où N_{y_i} est le nombre de composantes de la variables y_i et où $||a||$ est la somme des valeurs absolues des composantes de a. Cette différence est comparée à un certain critère. Ce critère est par défaut égal au critère de convergence de l'algorithme implicite, mais l'utilisateur peut en préciser en autre en utilisant le mot clé `@JacobianComparisonCriterium`. L'utilisateur peut également utiliser le paramètre

22. En effet, dans ce cas, l'erreur apparaît alors que sinon le seul symptôme est une convergence lente qui peut passer inaperçue si l'on y prend garde.

23. En $1D$, la dérivée numérique d'une fonction $f(x)$ au point x est évaluée ainsi :

$$\frac{f(x+\epsilon) - f(x-\epsilon)}{2\,\epsilon}$$

La perturbation est choisie égale à la valeur du critère d'arrêt de l'algorithme implicite.

	Nombre de cycles	Rapport à la référence
Référence	4 776 120.	1
Calcul du jacobien numérique et comparaison	26 944 564	5,65

TABLEAU 3 : Impact du calcul de jacobien numérique et de sa comparaison à celui fourni par l'utilisateur pour une loi de comportement simple.

`jacobianComparisonCriterium` pour modifier ce critère à l'exécution. Si la différence est supérieure au critère, un message donnant la valeur de la différence, le bloc considéré et son approximation numérique est affiché sur la sortie standard.

Le coût du calcul du jacobien numérique est souvent important. Nous pouvons le constater au tableau 3 où nous avons analysé le nombre de cycles processeur pour réaliser l'intégration d'une loi de NORTON lors d'un essai de traction uniaxial [24]. Pour le cas traité, le coût de l'intégration est multiplié par $5,65$. Il faut donc prendre garde à retirer cette option pour les versions de production des lois de comportement.

Initialisation de la matrice jacobienne L'expression (49) montre que le premier terme de la matrice jacobienne est la matrice identité. Cette écriture nous a conduit, dans `mfront` à initialiser, à chaque itération, la matrice jacobienne à la matrice identité.

6.4 MÉTHODES ALTERNATIVES

Insistons, le calcul de la matrice jacobienne est la principale difficulté de la méthode de NEWTON-RAPHSON.

Afin de dispenser l'utilisateur de fournir la matrice jacobienne, nous avons envisagé d'approximer le jacobien par différentiation numérique. Un jacobien correctement évalué (par une différence finie d'ordre 2) conduit à une bonne convergence de l'algorithme, mais a un coût numérique important [25]. Cette méthode est accessible par l'algorithme `NewtonRaphson_NumericalJacobian`.

Ce constat est général et de nombreux auteurs ont proposé des algorithmes qui ne se basent pas sur une différentiation numérique (sauf éventuellement pour une estimation initiale ou en cas de non convergence).

L'idée de ces méthodes est de modifier l'algorithme (48) en substituant à la matrice jacobienne une matrice $\underset{\sim}{J}$ bien choisie :

$$Y_n = Y_{n-1} - \underset{\sim}{J}^{-1} . F\left(Y_{n-1}\right) \tag{50}$$

Parmi ces méthodes, les algorithmes de BROYDEN sont les plus utilisés.

6.4.1 Les algorithmes de BROYDEN

Nous décrivons dans ce paragraphe deux algorithmes dû à BROYDEN.

Premier algorithme de BROYDEN Un des algorithmes les plus utilisés est celui de BROYDEN qui met à jour au cours des itérations une approximation $\underset{\sim}{J}_n$ de la matrice jacobienne à partir de son expression $\underset{\sim}{J}_{n-1}$ à l'itération précédente :

$$\underset{\sim}{J}_n = \underset{\sim}{J}_{n-1} + \frac{\Delta F_n - \underset{\sim}{J}_{n-1} . \Delta Y_n}{\left|\left|\Delta Y_n\right|\right|^2} \otimes \Delta Y_n \tag{51}$$

24. Cet essai est décrit plus en détails en annexe F.
25. Voir, en annexe F, les tableaux 8 et 9

où : — $\Delta Y_n = Y_n - Y_{n-1}$;
— $\Delta F_n = F(Y_n) - F(Y_{n-1})$;
— $v_1 \otimes v_2$ est le produit tensoriel de deux vecteurs ;
— $||v||$ est la norme euclidienne du vecteur v.

Calcul exact de certains blocs Il est intéressant de remarquer qu'il est toujours possible de modifier la matrice jacobienne par blocs : si certains termes sont connus et/ou faciles à calculer, il peut être intéressant de les injecter dans la matrice jacobienne en écrasant les modifications fournies par l'algorithme de BROYDEN.

Initialisation du jacobien Il est nécessaire de fournir une estimation initiale $\underset{\sim 0}{J}$ du jacobien. Deux choix ont été regardés dans cette note :

— $\underset{\sim 0}{J}$ est choisi égal à l'identité. L'expression (49) montre que pour des pas de temps raisonnablement petits, l'identité constitue une bonne approximation de la matrice jacobienne ;
— $\underset{\sim 0}{J}$ est calculé par perturbations du système initial.

Second algorithme de BROYDEN Une variante de l'algorithme de BROYDEN met à jour une approximation $\underset{\sim n}{J^{-1}}$ de l'inverse de la matrice jacobienne par la relation suivante :

$$\underset{\sim n}{J^{-1}} = \underset{\sim n-1}{J^{-1}} + \frac{\Delta Y_n - \underset{\sim n-1}{J^{-1}} . \Delta F_n}{\Delta Y_n \mid \underset{\sim n-1}{J^{-1}} . \Delta F_n} \otimes \left(\Delta Y_n \mid \underset{\sim n-1}{J^{-1}} \right) \tag{52}$$

Cette variante est disponible dans `mfront` sous le nom de `Broyden2`.

Dans ce cas, il n'est pas possible de fournir des blocs du jacobien.

6.5 AMÉLIORATION DE LA ROBUSTESSE DES ALGORITHMES IMPLICITES

Différentes méthodes permettant d'améliorer la robustesse des algorithmes ont été testées. Nous les décrivons maintenant bien que leur utilisation conduise à des résultats mitigés.

Pour des raisons d'efficacité du code généré, il a été décidé que le support d'une méthode d'accélération était une décision prise au moment de la génération du code. Les méthodes présentées présentent des paramètres permettant de fixer un nombre minimal d'itérations avant que la méthode s'active. En donnant une valeur suffisamment grande à ce paramètre, la méthode ne s'activera jamais, ce qui ne veut pas dire qu'elle n'aura aucun coût : par exemple, les méthodes présentées utilisent des valeurs des itérées précédentes qui sont toujours stockées.

Enfin, les paramètres associés à ces méthode ne seront disponibles que si elles sont activées.

6.5.1 Méthode d'accélération utilisé par les algorithmes de résolution globaux de `Cast3M`

L'idée de cette méthode d'accélération est de trouver une nouvelle estimation de la solution à partir des itérations précédentes. Soient F_{n-2}, F_{n-1} et F_n les résidus associés aux trois dernières estimations ΔY_{n-2}, ΔY_{n-1} et ΔY_n de la solution.

Il est possible de construire à partir de ces trois résidus un hyperplan. L'idée est d'exprimer la projection du vecteur nul (qui correspond à la solution recherchée) sur cet hyperplan comme une combinaison linéaire de

ces trois résidus et de déduire une nouvelle estimation de la solution en appliquant la même combinaison linéaire aux estimations ΔY_{n-2}, ΔY_{n-1} et ΔY_n.

Dans certains cas difficiles, cette méthode peut stabiliser les calculs, mais elle empêche la convergence quadratique de la méthode de NEWTON. Son intérêt apparaît aujourd'hui très limité.

La directive `@UseAcceleration` est utilisée pour activer cette méthode d'accélération. Elle est suivie soit du mot clé `true` soit du mot clé `false` et d'un point-virgule.

La directive `@AccelerationTrigger` permet de préciser à partir de quelle itération la méthode d'accélération est utilisée. La valeur fournie doit être supérieure à 3 pour que la méthode ait un sens. La valeur fournie peut être modifiée par le paramètre `accelerationTrigger`.

La directive `@AccelerationPeriod` permet de préciser le nombre d'itérations séparant deux emplois successifs de cette méthode. La valeur fournie peut être modifiée par le paramètre `accelerationPeriod`.

6.5.2 Méthode de relaxation

Une méthode de relaxation a été introduite pour améliorer la convergence de certains cas difficiles où des oscillations lors de la recherche de solution peuvent apparaître. Soient ΔY_{n-1} et ΔY_n les deux dernières estimations de la solution. La relaxation consiste à prendre comme nouvelle estimation :

$$(1-w)\ \Delta Y_{n-1} + w\,\Delta Y_n$$

où w est un coefficient de relaxation.

La directive `@RelaxationTrigger` permet de préciser à partir de quelle itération la méthode d'accélération est utilisée. La valeur fournie doit être supérieure à 3 pour que la méthode ait un sens. La valeur fournie peut être modifiée par le paramètre `relaxationTrigger`.

La directive `@RelaxationCoefficient` permet de préciser le coefficient w. La valeur fournie peut être modifiée à l'exécution par le paramètre `relaxationCoefficient`.

6.6 CALCUL DE LA MATRICE TANGENTE COHÉRENTE

L'utilisateur peut calculer la matrice tangente cohérente dans un bloc introduit par la directive `@TangentOperator`.

Cette directive est appelée si l'intégration de la loi a été un succès, avant la mise à jour des variables internes et la mise à jour de contraintes.

Symétrie de la matrice tangente Par défaut, la matrice tangente est supposée non symétrique. L'utilisateur peut explicitement dire que la matrice est symétrique par la directive `@IsTangentOperatorSymmetric` qui est suivie d'une des valeurs booléennes `true` ou `false`[26].

6.6.1 Une façon générique de calculer de la matrice tangente cohérente

La matrice tangente cohérente est la dérivée d'incrément de contraintes $\underline{\sigma}$ par rapport à l'incrément de déformations totales $\underline{\epsilon}^{to}$. Cet incrément était considéré jusqu'à présent comme un paramètre du système d'équations implicites. Il est considéré dans ce paragraphe comme la variable principale.

26. Si l'opérateur tangent est symétrique, l'interface `Aster` fait l'économie d'une transposition de la matrice (cette transposition est nécessaire pour la compatibilité avec le `fortran`.

Pour commencer, nous supposerons que le tenseur d'élasticité est une des variables internes de la loi. Plus loin, nous imposerons que cette variable soit la première dans le système implicite. Ces conditions sont imposées si on utilise l'analyseur `Implicit`.

Nous supposerons ensuite que la contrainte ne dépende que du tenseur d'élasticité (éventuellement de manière non linéaire). Nous reviendrons sur cette hypothèse plus tard. Nous en tirons que :

$$\frac{\partial \Delta \underline{\sigma}}{\partial \Delta \underline{\epsilon}^{to}} = \left.\frac{\partial \underline{\sigma}}{\partial \underline{\epsilon}^{el}}\right|_{\underline{\epsilon}^{el}+\Delta \underline{\epsilon}^{el}} : \frac{\partial \Delta \underline{\epsilon}^{el}}{\partial \Delta \underline{\epsilon}^{to}}$$

Le problème du calcul de la matrice tangente cohérente se ramène donc au calcul de la dérivée $\frac{\partial \Delta \underline{\epsilon}^{el}}{\partial \Delta \underline{\epsilon}^{to}}$.

Nous montrons dans ce paragraphe que si la matrice jacobienne du système est connue, alors il existe une façon générique de calculer ce terme dans de très nombreux cas.

Rappels mathématiques Avant de rentrer dans les détails de la méthode, il est utile de faire quelques rappels mathématiques. Les incréments des variables internes ΔY sont reliées à l'incrément de déformations totales par la relation implicite :

$$F\left(\Delta Y\left(\Delta \underline{\epsilon}^{to}\right), \Delta \underline{\epsilon}^{to}\right) = 0$$

Si $\Delta \underline{\epsilon}^{to}$ varie, cette relation reste vraie. Par différentiation, nous obtenons :

$$\mathrm{d}\, F = \frac{\partial F}{\partial \Delta Y}\, \mathrm{d}\, \Delta Y + \frac{\partial F}{\partial \Delta \underline{\epsilon}^{to}}\, \mathrm{d}\, \Delta \underline{\epsilon}^{to} = 0$$

Le terme $\frac{\partial F}{\partial \Delta Y}$ est la jacobienne J du système qui est connue après la résolution.

Pour déduire $\frac{\partial \Delta \underline{\sigma}}{\partial \Delta \underline{\epsilon}^{to}}$ de cette équation, nous allons faire une hypothèse qui permet de calculer explicitement $\frac{\partial F}{\partial \Delta \underline{\epsilon}^{to}}$.

Hypothèse Nous supposons que l'incrément de déformation totales $\Delta \underline{\epsilon}^{to}$ n'apparaît que dans l'équation relative aux déformations élastiques par la relation de partition des déformations, qui, une fois discrétisée, donne :

$$\Delta \underline{\epsilon}^{el} + \sum_i \Delta \underline{\epsilon}_i^{an} - \Delta \underline{\epsilon}^{to} = 0$$

La dérivée de $\frac{\partial F}{\partial \Delta \underline{\epsilon}^{to}}$ est alors évidente et le vecteur $\frac{\partial F}{\partial \Delta \underline{\epsilon}^{to}}\, \mathrm{d}\, \Delta \underline{\epsilon}^{to}$ est égal à :

$$\frac{\partial F}{\partial \Delta \underline{\epsilon}^{to}}\, \mathrm{d}\, \Delta \underline{\epsilon}^{to} = -\begin{pmatrix} \mathrm{d}\, \Delta \underline{\epsilon}^{to} \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Nous en déduisons que :

$$\mathrm{d}\, \Delta \underline{\epsilon}^{el} = J^{-1}_{\underline{\epsilon}^{el}} :\ \mathrm{d}\, \Delta \underline{\epsilon}^{to}$$

où $J^{-1}_{\underline{\epsilon}^{el}}$ est la partie supérieure gauche de l'inverse de la jacobienne.

Finalement, nous obtenons :

$$\frac{\partial \Delta \underline{\sigma}}{\partial \Delta \underline{\epsilon}^{to}} = \left.\frac{\partial \underline{\sigma}}{\partial \underline{\epsilon}^{el}}\right|_{\underline{\epsilon}^{el}+\Delta \underline{\epsilon}^{el}} : J_{\underline{\epsilon}^{el}}^{-1}$$

La matrice $J_{\underline{\epsilon}^{el}}^{-1}$ peut être calculée par la méthode `getPartialJacobianInvert` à qui l'on doit passer un tenseur d'ordre 4. Cette méthode ne doit être utilisée que dans le bloc `@TangentOperator`.

Ce cas explique pourquoi la déformation élastique est toujours définie comme la première variable interne dans le cas de l'analyseur `Implicit`.

Généralisation La méthode précédente se généralise dans le cas où la contrainte dépend de plusieurs variables internes et éventuellement de la déformation totale. En effet, il suffit d'écrire :

$$\frac{\partial \Delta \underline{\sigma}}{\partial \Delta \underline{\epsilon}^{to}} = \sum_i \left.\frac{\partial \underline{\sigma}}{\partial y_i}\right|_{y_i+\Delta y_i} : \frac{\partial \Delta y_i}{\partial \Delta \underline{\epsilon}^{to}} + \left.\frac{\partial \underline{\sigma}}{\partial \underline{\epsilon}^{to}}\right|_{\underline{\epsilon}^{to}+\Delta \underline{\epsilon}^{to}}$$

Les mêmes considérations que précédemment montrent que les différentes dérivées $\dfrac{\partial \Delta y_i}{\partial \Delta \underline{\epsilon}^{to}}$ se lisent sur les $N_{\underline{\epsilon}^{to}}$ premières colonnes de l'inverse de la matrice jacobienne, où $N_{\underline{\epsilon}^{to}}$ est le nombre de composantes du tenseur $\underline{\epsilon}^{to}$.

Il est possible de récupérer ces dérivées par la méthode `getPartialJacobianInvert` à qui l'on doit passer successivement des variables qui vont contenir les dérivées des variables d'intérêt dans *l'ordre de déclaration des variables*. Ces variables doivent être des tenseurs d'ordre 2 dans le cas de variables scalaires et des tenseurs d'ordre 4 dans le cas de variables tensorielles.

Dans le cas des tableaux de variables internes, il faut passer des vecteurs de tailles finies du type adéquat [27].

Insistons : si l'on a besoin que des dérivées de deux premières variables internes (par exemple la déformation élastique et la variable d'endommagement) alors il suffit de passer deux arguments à la méthode `getPartialJacobianInvert`, on ne calculera donc pas les autres dérivées pour rien. À l'utilisateur de déclarer les variables dans le bon ordre !

6.6.2 Une méthode alternative de calcul de la matrice tangente cohérente

Il existe une autre méthode de calcul de la matrice tangente cohérente basée sur un partitionnement de la jacobienne et non de son inverse [Proix 12c].

Pour cela, notons z_i les variables internes autres que les déformations élastiques. Nous avons, avec les mêmes hypothèses que précédemment :

$$J \,.\, \begin{pmatrix} \mathsf{d}\,\Delta \underline{\epsilon}^{el} \\ \mathsf{d}\,\Delta Z \end{pmatrix} - \begin{pmatrix} \mathsf{d}\,\Delta \underline{\epsilon}^{to} \\ 0 \end{pmatrix} = 0$$

En découpant la matrice jacobienne en quatres blocs :

$$J = \begin{pmatrix} J_0 & J_1 \\ J_2 & J_3 \end{pmatrix}$$

on obtient deux systèmes :

$$\begin{cases} J_0\, \mathsf{d}\,\Delta \underline{\epsilon}^{el} + J_1\, \mathsf{d}\,\Delta Z = \mathsf{d}\,\Delta \underline{\epsilon}^{to} \\ J_2\, \mathsf{d}\,\Delta \underline{\epsilon}^{el} + J_3\, \mathsf{d}\,\Delta Z = 0 \end{cases}$$

Le second système conduit à :

$$\mathsf{d}\,\Delta Z = -J_3^{-1}\, J_2\, \mathsf{d}\,\Delta \underline{\epsilon}^{el}$$

27. On utilisera la classe `tvector` de `tfel` qui paramétrée par sa taille et le type d'objet contenu.

Par report dans le premier système, nous obtenons :

$$\mathrm{d}\,\Delta\,\underline{\epsilon}^{el} = \left(J_0 - J_1\,J_3^{-1}\,J_2\right)^{-1}\,\mathrm{d}\,\Delta\underline{\epsilon}^{to}$$

et finalement :

$$\frac{\partial\Delta\,\underline{\epsilon}^{el}}{\partial\Delta\underline{\epsilon}^{to}} = \left(J_0 - J_1\,J_3^{-1}\,J_2\right)^{-1}$$

Cette méthode peut être avantageuse si le nombre de variables internes est petit.

6.7 LA DIRECTIVE @COMPUTESTRESS

La directive `@ComputeStress` permet de calculer les contraintes aux différentes itérations de la méthode et en fin de calcul. Les contraintes sont supposées fonction des valeurs actuelles des variables internes.

Conventions spécifiques aux différentes itérations de la méthode Dans la méthode `@ComputeStress`, les conventions suivantes s'appliquent :
- `sig` représente la contrainte qu'il faut calculer ;
- `eto` représente la déformation totale en milieu de pas (en $t + \theta\,\Delta\,t$) ;
- `deto` représente l'incrément de la déformation totale (constante sur le pas) ;
- `T` représente la valeur de la température en milieu de pas (en $t + \theta\,\Delta\,t$) ;
- `dT` représente la variation de température (constante sur le pas) ;
- pour toute variable interne `Y`, `Y` représente son estimation actuelle en milieu de pas (en $t + \theta\,\Delta\,t$) ;
- pour toute variable interne auxiliaire `Y`, `Y` représente sa valeur en *début* de pas de temps ;
- pour toute variable externe `V`, `V` représente sa valeur en milieu de pas (en $t + \theta\,\Delta\,t$) ;
- pour toute variable externe `V`, `dV` représente son incrément sur le pas de temps (constante sur le pas).

Conventions spécifiques en fin d'intégration Dans la méthode `@ComputeStress`, les conventions suivantes s'appliquent :
- `sig` représente la contrainte qu'il faut calculer ;
- `eto` représente la déformation totale en fin de pas ;
- `deto` représente la vitesse de déformation totale (constante sur le pas) ;
- `T` représente la valeur de la température en fin de pas ;
- `dT` représente la vitesse de changement de température (constante sur le pas) ;
- pour toute variable interne `Y`, `Y` représente sa valeur en fin de pas ;
- pour toute variable interne auxiliaire `Y`, `Y` représente sa valeur en *début* de pas de temps ;
- pour toute variable externe `V`, `V` représente sa valeur en fin de pas ;
- pour toute variable externe `V`, `dV` représente sa vitesse de variation sur le pas de temps (constante sur le pas).

6.8 LA DIRECTIVE @INTEGRATOR

La directive `@Integrator` permet de construire le système implicite à résoudre.

Conventions spécifiques Les conventions suivantes s'appliquent :
- `sig` représente la contrainte actualisée (calculée par le code fourni après la directive `@ComputeStress`) ;
- `eto` représente la déformation totale en début de pas ;
- `deto` représente la vitesse de déformation totale (constante sur le pas) ;

— `T` représente la valeur de la température en début de pas ;
— `dT` représente la vitesse de changement de température (constante sur le pas) ;
— pour toute variable interne `Y`, `Y` représente sa valeur en début de pas ;
— pour toute variable interne `Y`, `Y` représente l'estimation courante de l'incrément de cette variable sur le pas ;
— pour toute variable interne `Y`, `fY` représente l'équation implicite associée à cette variable ;
— pour toute variable interne auxiliaire `Y`, `Y` représente sa valeur en début de pas ;
— pour toute variable externe `V`, `V` représente sa valeur en début de pas ;
— pour toute variable externe `V`, `dV` représente sa vitesse de variation sur le pas de temps (constante sur le pas).

Si l'algorithme de NEWTON ou le premier algorithme de BROYDEN est utilisé, pour tout couple de variables internes `Yi` et `Yj`, `dfYi_ddYj` représente le terme du jacobien $\frac{\partial f_{Y_i}}{\partial Y_j}$;

6.9 MISE À JOUR DES VARIABLES AUXILIAIRES

Les variables auxiliaires sont mises à jour après la mise à jour des variables internes et la mise à jour des contraintes, une fois l'algorithme convergé.

À ce moment, les valeurs externes (déformations totales et températures incluses) n'ont pas été mises à jour.

6.10 ORDRE D'ÉVALUATION DES BLOCS DÉFINIS PAR L'UTILISATEUR

La figure 4 montre l'ordre d'appel des différents blocs de code définis par l'utilisateur.

6.11 PARAMÈTRES AUTOMATIQUEMENT DÉFINIS

L'analyseur implicite définit automatiquement différents paramètres :
— `epsilon`, la valeur du critère d'arrêt de l'algorithme implicite. La valeur par défaut de ce paramètre est de 10^{-8} et peut être modifiée par la directive `@Epsilon`. Ce paramètre doit avoir une valeur strictement positive ;
— `theta`, la valeur de θ utilisé par l'algorithme implicite. La valeur par défaut de ce paramètre est de 0.5 et peut être modifiée par la directive `@Theta`. Les valeurs admissibles sont comprises entre 0 et 1 ;
— `iterMax`, le nombre maximum d'itérations autorisées.

Si la méthode d'accélération de `Cast3M` est utilisée, les paramètres suivants sont déclarés :
— `accelerationTrigger`, le nombre d'itérations minimum pour utiliser la méthode d'accélération ;
— `accelerationPeriod`, le nombre d'itérations entre deux utilisations de la méthode d'accélération ;

Si la méthode de relaxation de `Cast3M` est utilisée, les paramètres suivants sont déclarés :
— `relaxationTrigger`, le nombre d'itérations minimum pour utiliser la méthode de relaxation ;
— `relaxationCoefficient`, le coefficient de relaxation.

6.12 INTÉGRATION D'UNE LOI D'ÉCOULEMENT VISCOPLASTIQUE ORTHOTROPE PAR UN ALGORITHME IMPLICITE

Nous reprenons ici l'exemple précédent en modifiant l'algorithme d'intégration utilisé par un algorithme implicite. Nous traitons ici la généralisation de la loi de NORTON au cas orthotrope.

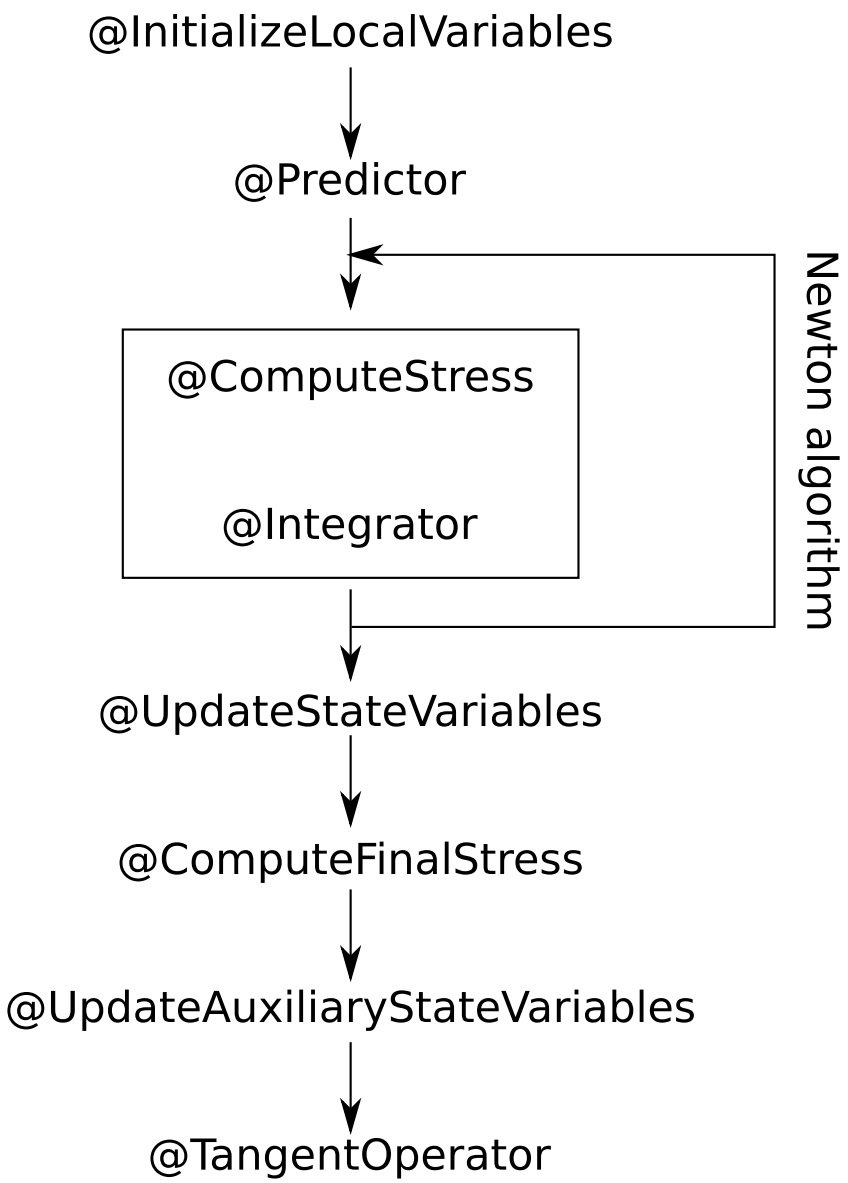


FIGURE 4 : Ordre d'évaluation des différents blocs de code fournis par l'utilisateur dans le cas d'une intégration implicite.

Les algorithmes implicites (θ-méthodes) conduisent à réécrire le système différentiel (41) ainsi :

$$\begin{aligned} \Delta\,\underline{\epsilon}^{el} + \Delta\,p\,\underline{\mathbf{n}} - \Delta\,\underline{\epsilon}^{to} &= 0 \\ \Delta p - A\,\left(\sigma_H\right)^E\,\Delta\,t &= 0 \end{aligned} \tag{53}$$

où :
- $\Delta\,\underline{\epsilon}^{to}$ est l'incrément sur le pas de temps de la déformation totale (imposée) ;
- $\Delta\,\underline{\epsilon}^{el}$ est l'incrément sur le pas de temps de la déformation élastique ;
- $\Delta\,p$ est l'incrément sur le pas de temps de la déformation viscoplastique équivalente ;
- $\underline{\mathbf{n}}$ est la normale à la surface de charge évaluée à l'instant $t + \theta\,\Delta\,t$;
- σ_H est la contrainte équivalente de HILL évaluée à l'instant $t + \theta\,\Delta\,t$;
- $\Delta\,t$ est l'incrément de temps.
- θ est un paramètre d'intégration compris entre 0 et 1.

Ce système non-linéaire d'inconnues $\Delta\,\underline{\epsilon}^{el}$ et $\Delta\,p$ se décompose naturellement en deux équations $f_{\underline{\epsilon}^{el}}$ et f_p.

Ce système est résolu par une méthode de NEWTON-RAPHSON, ce qui nécessite de calculer sa matrice jacobienne. Cette matrice se décompose en quatre parties notées respectivement $\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta\,\underline{\epsilon}^{el}}$, $\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta\,p}$, $\frac{\partial f_p}{\partial \Delta\,\underline{\epsilon}^{el}}$ et $\frac{\partial f_p}{\partial \Delta\,p}$.

Ces différents termes se calculent ainsi :

$$\left\{\begin{aligned} \frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta\,\underline{\epsilon}^{el}} &= \underline{\underline{\mathbf{I}}} + \Delta\,p \frac{\partial n}{\partial \underline{\sigma}} \frac{\partial \underline{\sigma}}{\partial \Delta\,\underline{\epsilon}^{el}} \\ &= \underline{\underline{\mathbf{I}}} + \Delta\,p \frac{\partial n}{\partial \underline{\sigma}} \frac{\partial \underline{\sigma}}{\partial\,\underline{\epsilon}^{el}\big|_{t+\theta\,\Delta\,t}} \frac{\partial\,\underline{\epsilon}^{el}\big|_{t+\theta\,\Delta\,t}}{\partial \Delta\,\underline{\epsilon}^{el}} \\ &= \underline{\underline{\mathbf{I}}} + 2\,\mu\,\theta\,\Delta\,p\,\frac{\partial n}{\partial \underline{\sigma}} \\ &= \underline{\underline{\mathbf{I}}} + \frac{2\,\mu\,\theta\,\Delta\,p}{\sigma_{eq}}\,\left(\underline{\underline{\mathbf{H}}} - \underline{\mathbf{n}} \otimes \underline{\mathbf{n}}\right) \\ \frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta\,p} &= \underline{\mathbf{n}} \\ \frac{\partial f_p}{\partial \Delta\,\underline{\epsilon}^{el}} &= -A\,E\,\Delta\,t\,\left(\sigma_H\right)^{E-1}\,\frac{\partial \sigma_H}{\partial \Delta\,\underline{\epsilon}^{el}} \\ &= -A\,E\,\theta\,\Delta\,t\,\left(\sigma_H\right)^{E-1}\,\underline{\mathbf{n}} : \underline{\underline{\mathbf{D}}} \\ \frac{\partial f_p}{\partial \Delta\,p} &= 1 \end{aligned}\right.$$

où $\underline{\underline{\mathbf{H}}}$ est le tenseur de HILL du matériau déjà rencontré au paragraphe 5.8.

Implantation La figure 5 présente l'implantation de cette loi. La définition du système implicite se fait dans la partie `@Integrator`. Les variables `feel`, `fp`, `dfeel_ddeel`, `dfeel_ddp`, `dfp_ddeel`, `dfp_ddp` sont automatiquement définies. Les termes du système sont initialisés à la valeur des incréments des variables correspondantes (`feel` est initialisé à `deel`) et la matrice jacobienne est initialisée à l'identité.

Cette implantation tient compte des limitations du code aux éléments finis `Cast3M` qui ne permet pas d'avoir une définition homogène des directions d'orthotropie quelle que soit l'hypothèse de modélisation. Les conventions utilisées ici sont conformes à celles décrites en annexe E pour les tubes.

```
@Parser     Implicit;
@Behaviour ImplicitOrthotropicCreep;
@Author     Helfer Thomas;
@Date       15/03/11;

@OrthotropicBehaviour;
@RequireStiffnessTensor;

@Theta 1.;

@StateVariable real p;       /* Equivalent viscoplastic strain */
p.setGlossaryName("EquivalentViscoplasticStrain");

@Includes{
#include<TFEL/Material/Lame.hxx>
#include<TFEL/Material/Hill.hxx>
}

@ComputeStress{
  sig = D*eel;
}

@Integrator{
  const real H_rr = 0.371;
  const real H_tt = 1-H_rr;
  const real H_zz = 4.052;
  const real H_rt = 1.5;
  const real H_rz = 1.5;
  const real H_tz = 1.5;
  const real A = 8.e-67;
  const real E = 8.2;
  st2tost2<N,real> H;
  if((getModellingHypothesis()==ModellingHypothesis::PLANESTRESS)||
     (getModellingHypothesis()==ModellingHypothesis::PLANESTRAIN)||
     (getModellingHypothesis()==ModellingHypothesis::GENERALISEDPLANESTRAIN)){
    H = hillTensor<N,real>(Hzz,Hrr,Htt,
      Hrt,Hrz,Htz);
  } else {
    H = hillTensor<N,real>(Htt,Hrr,Hzz,
      Hrz,Hrt,Htz);
  }
  const real sigeq = sqrt(sig|H*sig);
  const real tmp = A*pow(sigeq,E-1.);
  real inv_sigeq(0);
  Stensor  n(0.);
  if(sigeq > 1.){
    inv_sigeq = 1/sigeq;
  }
  n  = (H*sig)*inv_sigeq;
  feel += dp*n-deto;
  fp   -= tmp*sigeq*dt;
  dfeel_ddeel += theta*dp*(H-(n^n))*D*inv_sigeq;
  dfeel_ddp    = n;
  dfp_ddeel    = -theta*tmp*E*dt*(n|D);
}
```

FIGURE 5 : Implantation d'une loi d'écoulement viscoplastique orthotrope par un algorithme implicite

7 CONCLUSIONS

Cette note a décrit les possibilités de `mfront` dans le domaine de l'intégration des lois de comportement mécanique. Elle complète la présentation générale de `mfront` [Helfer 13d].

Pour des raisons pédagogiques, nous avons choisi de ne présenter dans cette note que des exemples élémentaires. Des exemples plus complexes peuvent être trouvés dans les sources de la librairie `tfel`. Pour les applications de la plate-forme `pleiades`, des lois de comportement des matériaux usuels du nucléaire peuvent être trouvés dans la base de données `sirius`. Les utilisateurs CEA de l'application `licos` peuvent également accéder à un corpus de connaissances matériau précompilées via le projet `MFrontMaterials` [Helfer 13a].

7.1 UNE SOLUTION FLEXIBLE, MATURE ET PERFORMANTE

`mfront` constitue aujourd'hui une solution flexible et mature au cœur de la gestion des matériaux de la plate-forme `pleiades`. Il a été évalué en interne ou par les partenaires industriels du CEA [Olagnon 13].

L'évaluation par l'équipe `Aster` Pour illustrer ce point, nous pouvons nous appuyer sur l'évaluation faite par l'équipe de code `Aster` qui est particulièrement complète [Proix 13b].

Différentes classes de lois de comportement ont été testées :
- loi de MAZARS décrivant l'endommagement fragile du béton [**?**] ;
- l'endommagement sous fluage tertiaire de structure en acier par une loi dite de HAYHURST [Proix 12a] ;
- lois plastiques et viscoplastiques de type CHABOCHE avec écrouissage cinématique (linéaire ou non linéaire) ;
- comportement mono-cristallin et poly-cristallins pour des structures CFC.

De plus, l'équipe `Aster` a reversé de nombreux tests de loi de comportement dans la librairie `tfel`. Ces tests ont été écrits à l'aide de l'utilitaire `mtest` [**?**].

	`mfront` implicite (19 équations)	`VISC_CIN1_CHAB` implicite aster optimisé (1 équation)
Nombre de pas de temps	76	76
Nombre d'itérations de NEWTON	160	151
Temps CPU	4,67s	4,19s

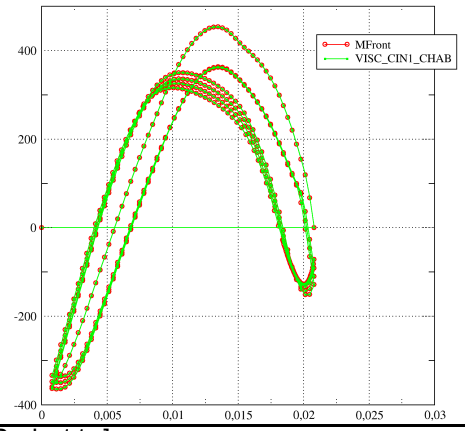


TABLEAU 4 : Résultat du test `HSNV125` [Proix 11a].

Test de robustesse Le test `HSNV125` est extrait de la base de cas test de `Aster` [Proix 11a]. Il s'agit d'un test particulièrement sollicitant pour les algorithmes d'intégration des loi de comportement. Un exemple de résultat de ce test, qui permet d'apprécier la complexité du chargement imposé, est donné au tableau 4 pour

une loi de type CHABOCHE avec deux variables cinématiques non linéaires. Dans cet exemple, la loi `mfront` est légèrement moins efficace que la loi optimisée d'`Aster`[28].

	MFRONT Implicite	MFRONT implicite avec jacobienne numérique	Aster avec jacobienne numérique
Nombre de pas de temps	11	11	11
Nombre d'itérations de NEWTON	37	36	44
Temps CPU	8mn11s	7mn58s	17mn43s
Temps d'intégration de la loi de comportement	2mn26s	2mn21s	10mn51s
Temps d'inversion de la matrice	5mn11s	5mn5s	6mn11s

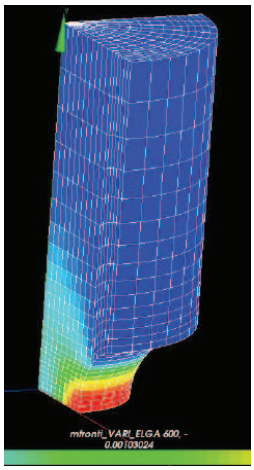


TABLEAU 5 : Résultat des test de fluage sur éprouvette entaillée $3D$, loi de HAYHURST [Proix 13b].

Tests de performances Plusieurs comparaisons avec les lois implantées dans `Aster` ont été menées. Dans l'ensemble, les lois générées par `mfront` se sont avérées efficaces, autant que les lois `Aster` et parfois meilleures.

Un premier exemple est donnée par le tableau 5 où l'on reporte les résultats obtenus sur une éprouvette entaillée subissant un essai de fluage. Le calcul est effectué en 3D en grandes déformations grâce aux formalisme des déformations logarithmiques qui permet d'utiliser des lois écrites en petites déformations [Bargellini 13, Miehe 02].

	`mfront` (RK42)	`Aster`
Temps CPU	90s	136s

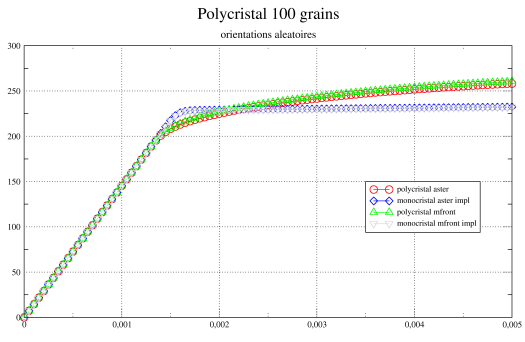


TABLEAU 6 : Comparaison des temps de calculs pour une loi issue de l'homogénéisation d'un poly-cristal constitué de 100 grains.

Un second test, basé sur une loi d'homogénéisation de lois de plasticité cristalline, démontre les capacités de `mfront` à traiter un grand nombre de variables internes (plus de $4\,200$ ici) en conservant des performances numériques intéressantes, reportées au tableau 6. Le fichier `mfront` fait une centaine de lignes.

28. Une version non optimisée de la loi est également disponible dans `Aster` (21 équations résolues par une méthode implicite). Cette version conduit à une divergence du calcul pour le test proposé.

7.2 PERSPECTIVES

L'utilisation de `mfront` hors de la plate-forme `pleiades`, au DMN et à EDF notamment, offre des possibilités de qualification de l'outil extrêmement intéressantes et des voies de collaboration extrêmement riches et enrichissantes.

7.3 DÉVELOPPEMENT

Dans le cadre `pleiades` différentes perspectives ont été identifiées :

- un meilleur support des contraintes planes ;
- la possibilité d'écrire des lois de comportements poutre ;
- l'amélioration de la robustesse de algorithmes implicites, qui peuvent diverger si l'estimation initiale est mauvaise. La méthode de POWELL semble particulièrement intéressante [Chen 81].

Pour une utilisation dépassant le cadre combustible, différents développements sont envisagés. Ces développements se résument généralement à la possibilité d'utiliser des lois de comportement qui ne soient pas limitées au cas volumique petites déformations. Ceci inclut :

- le cas des poutres qui utilisent des variables généralisées (on retrouve le besoin combustible) ;
- les grandes transformations ;
- le cas des modèles de zones cohésives qui travaillent en traction/saut de déplacement ;
- les lois à gradients (de déformations ou à variables internes).

À moyen terme, ces développements intéressent également nos applications combustibles :

- les grandes transformations semblent intéressantes dans des contextes accidentels (RIA, APRP) ;
- les modèles de zones cohésives sont utilisés au DMN/SEMI pour la modélisation de la propagation d'une fissure par corrosion sous contrainte dans la gaine ;
- les lois à gradients permettraient une meilleure représentation des phénomènes d'endommagement dans le cadre des travaux des thésards du SESC (modélisation de l'essai de flexion des barreaux UO_2).

La structure de `mfront` fait que tous ces développements sont informatiquement connexes et peuvent être regroupés dans une même action.

RÉFÉRENCES

[Abbas 13] ABBAS MICKAEL. *Algorithme non linéaire quasi-statique STAT_NON_LINE*. Référence du Code Aster R5.03.01 révision : 10290, EDF-R&D/AMA, Janvier 2013.

[Bargellini 13] BARGELLINI R. *Modélisation élasto(visco)plastique avec écrouissage isotrope en grandes déformations*. Référence du Code Aster R5.03.21 révision : 11537, EDF-R&D/AMA, Septembre 2013.

[Besson 98] BESSON JACQUES, LERICHE RODOLPHE, FOERCH RONALD et CAILLETAUD GEORGES. *Object-oriented programming applied to the finite element method Part II. Application to Material Behaviors*. Revue Européenne des Éléments, 1998, vol 7, n° 5, p 567–588.

[Besson 01] BESSON JACQUES, CAILLETAUD GEORGES et CHABOCHE JEAN-LOUIS. *Mécanique non linéaire des matériaux*. Hermès, Paris, 2001.

[Blanc 11] BLANC VICTOR et JULIEN JÉRÔME. *Intégration d'un algorithme de résolution au générateur de code MFront - application au modèle couplé fluage-fissuration*. Rapport technique 11 - 026, CEA DEN/DEC/SESC/LSC, 2011.

[Castelier 09] CASTELIER ÉTIENNE, HELFER THOMAS et PACULL JULIEN. *Spécifications d'un code de thermomécanique avec résolution par transformées de Fourier rapides*. Note technique 09 - 045, DEC/SESC/LSC, 2009.

[CEA 13] CEA . *Site Cast3M*, 2013.

[Chaboche 09] CHABOCHE JEAN-LOUIS, LEMAÎTRE JEAN, BENALLAL AHMED et DESMORAT RODRIGUE. *Mécanique des matériaux solides*. Dunod, Paris, 2009.

[Chen 81] CHEN HERN-SHANN et STADTHERR MARK A. *A modification of Powell's dogleg method for solving systems of nonlinear equations*. Computers & Chemical Engineering, 1981, vol 5, n° 3, p 143 – 150.

[EDF 13] EDF . *Site du Code_Aster*, 2013.

[Fortin 01] FORTIN ANDRÉ. *Analyse numérique pour ingénieurs*. Presses internationales Polytechnique, [Montréal], 2001.

[François 95] FRANÇOIS DOMINIQUE, PINEAU ANDRÉ et ZAOUI ANDRÉ. *Comportement mécanique des matériaux II - viscoplasticité, endommagement, mécanique de la rupture, mécanique du contact*. Hermès, Paris, 1995.

[Helfer 10] HELFER THOMAS, CASTELIER ÉTIENNE, BRUNO ÉRIC, GOHIER ÉRIC et BONHOMME CHRISTIAN. *Ajout de connaissances matériau (propriétes matériau, lois de comportement et modèles - au code Celaeno à l'aide générateur code MFront*. Rapport technique 10-002, DEC/SESC/LSC, 2010.

[Helfer 11] HELFER THOMAS, MICHEL BRUNO, BOUINEAU VINCENT et COSTOMIRIS AUDREY. *Bilan de l'optimisation de l'intégration des lois de comportement mécanique dans Alcyone et Germinal V2*. Note technique 11 - 005, CEA DEN/DEC/SESC/LSC, Février 2011.

[Helfer 13a] HELFER THOMAS. *Connaissances matériaux disponibles dans l'application Licos version 1.1*. Note technique, CEA DEN/DEC/SESC/LSC, 2013. À paraître.

[Helfer 13b] HELFER THOMAS. *L'interface aster aux lois de comportement mécanique de MFront*. Note technique, CEA DEN/DEC/SESC/LSC, 2013. En cours de rédaction.

[Helfer 13c] HELFER THOMAS. *L'interface umat aux lois de comportement mécanique de MFront*. Note technique, CEA DEN/DEC/SESC/LSC, 2013. En cours de rédaction.

[Helfer 13d] HELFER THOMAS et CASTELIER ÉTIENNE. *Le générateur de code mfront : présentation générale et application aux propriétés matériau et aux modèles*. Note technique 13-019, CEA DEN/DEC/SESC/LSC, Juin 2013.

[Helfer e] HELFER THOMAS. *Écriture de lois de comportement mécanique en grandes transformations avec le générateur de code MFront*. Note technique, CEA DEN/DEC/SESC/LSC, 2014 (à paraître).

[Michel 09] MICHEL BRUNO. *Étude de faisabilité de la génération automatique des lois de comportement mécanique non linéaires dans la plate-forme pleiades*. Note technique 09-028, DEC/SESC/LSC, Novembre 2009.

[Miehe 02] MIEHE C., APEL N. et LAMBRECHT M. *Anisotropic additive plasticity in the logarithmic strain space : modular kinematic formulation and implementation based on incremental minimization principles for standard materials*. Computer Methods in Applied Mechanics and Engineering, Novembre 2002, vol 191, n° 47–48, p 5383–5425.

[Milliard 14] MILLIARD FRANCK, COURCELLE ARNAUD, GAVOILLE PIERRE et FALESKOG JONAS. *Mechanical behavior of hexagonal tubes during accidental situation*, Février 2014.

[Olagnon 13] OLAGNON JULIEN et GARNIER CHRISTOPHE. *Analysis of a new integrator for finite element code for the the calculation of fuel rods thermal-mechanical beahviour : mfront*. Rapport technique FS1-0010103, Areva-NP, 2013.

[Pascal 05] PASCAL SERGE. *Notice d'utilisation du composant mécanique de PLEIADES : INCREPL*. Rapport technique RT/05-011/A PLE 05-007, SEMT/LM2S, Mars 2005.

[Proix 11a] PROIX JEAN-MICHEL. *HSNV125 - Élément de volume en traction / cisaillement et température variables*. Référence du Code Aster V4.22.125 révision : 7982, EDF-R&D/AMA, 2011.

[Proix 11b] PROIX JEAN-MICHEL. *Étude d'un critère de radialité*. Rapport technique CR-AMA-11.303, EDF-R&D/AMA, 2011.

[Proix 12a] PROIX JEAN-MICHEL. *Comportement viscoplastique avec endommagement de Hayhurst*. Référence du Code Aster R5.03.13 révision : 8886, EDF-R&D/AMA, 2012.

[Proix 12b] PROIX JEAN-MICHEL. *Intégration des relations de comportement élasto-plastique de Von Mises*. Référence du Code Aster R5.03.92 révision : 8597, EDF-R&D/AMA, Mars 2012.

[Proix 12c] PROIX JEAN-MICHEL. *Intégration implicite et explicite des relations de comportements non linéaires*. Référence du Code Aster R5.03.14 révision : 9045, EDF-R&D/AMA, 2012.

[Proix 12d] PROIX JEAN-MICHEL. *Prise en compte de l'hypothèse des contraintes planes dans les comportements non linéaires*. Référence du Code Aster R5.03.03 révision 10101, EDF-R&D/AMA, 2012.

[Proix 13a] PROIX JEAN-MICHEL. *Comportements élastoviscoplastiques mono et polycristallins*. Référence du Code Aster R5.03.11 révision : 10623, EDF-R&D/AMA, 2013.

[Proix 13b] PROIX JEAN-MICHEL. *Intégration des lois de comportement à l'aide de MFront : bilan des tests réalisés pour l'utilisation avec Code Aster*. Rapport technique H-T64-2013-00922-FR, EDF-R&D/AMA, 2013.

LISTE DES TABLEAUX

TABLEAU 1 Arguments possibles de la directive `@Algorithm`. 32

TABLEAU 2 Temps de calculs obtenus sur une éprouvette entaillée en traction modélisée en $2D\,(r,z)$. Calcul réalisé par J.-M. PROIX sur la version 11 de code `Aster`. La loi de comportement locale est la loi de HAYHURST [Proix 12a]. Cette comparaison a été fournie par J.-M. Proix (EDF/AMA). 45

TABLEAU 3 Impact du calcul de jacobien numérique et de sa comparaison à celui fourni par l'utilisateur pour une loi de comportement simple. 47

TABLEAU 4 Résultat du test `HSNV125` [Proix 11a]. 57

TABLEAU 5 Résultat des test de fluage sur éprouvette entaillée $3D$, loi de HAYHURST [Proix 13b]. 58

TABLEAU 6 Comparaison des temps de calculs pour une loi issue de l'homogénéisation d'un poly-cristal constitué de 100 grains. 58

TABLEAU 7 Temps calculs obtenus avec l'algorithme de NEWTON-RAPHSON. 79

TABLEAU 8 Temps calculs obtenus avec l'algorithme de quasi-NEWTON et un calcul numérique du jacobien par une différence finie (approximation d'ordre 1). 80

TABLEAU 9 Temps calculs obtenus avec l'algorithme de quasi-NEWTON et un calcul numérique du jacobien par une différence finie centrée (approximation d'ordre 2). 80

TABLEAU 10 Temps calculs obtenus avec le premier algorithme de BROYDEN et jacobien initial égal à l'identité. 81

TABLEAU 11 Temps calculs obtenus avec le premier algorithme de BROYDEN et jacobien initial approximé numériquement. 81

TABLEAU 12 Temps calculs obtenus avec le second algorithme de BROYDEN et jacobien initial égal à l'identité. 81

LISTE DES FIGURES

FIGURE 1 Implantation de la loi de comportement viscoplastique du SiC en `mfront`. 27
FIGURE 2 Implantation d'une loi orthotrope élastique 29
FIGURE 3 Implantation d'une loi d'écoulement viscoplastique orthotrope par une méthode de RUNGE-KUTTA 39
FIGURE 4 Ordre d'évaluation des différents blocs de code fournis par l'utilisateur dans le cas d'une intégration implicite. 54
FIGURE 5 Implantation d'une loi d'écoulement viscoplastique orthotrope par un algorithme implicite 56
FIGURE 6 Description de l'algorithme pour déterminer le sous-découpage du pas de temps 73
FIGURE 7 Conventions adoptées pour la définition des axes d'orthotropies 77
FIGURE 8 Contraintes équivalentes obtenues respectivement par une résolution par un algorithme de NEWTON-RAPHSON ou par un algorithme de BROYDEN. 80

ANNEXE A RÉSOLUTION D'UN PROBLÈME MÉCANIQUE QUASI-STATIQUE PAR LA MÉTHODE DES ÉLÉMENTS FINIS

Nous décrivons succinctement la résolution par la méthode des éléments finis utilisées pour décrire l'évolution quasi-statique d'un système mécanique. Ceci nous permet de bien préciser la place de la loi de comportement et de faire apparaître la notion de matrice tangente cohérente.

La présentation faite est suffisamment générale pour notre propos mais nous avons passé sous silence de nombreux points (transformations finies, gestion du contact, etc...). Chaque code aux éléments finis présentera sa propre variation, plus au moins sophistiquée [Besson 01, Pascal 05, Abbas 13].

ANNEXE A.1 DISCRÉTISATION

La méthode des éléments finis repose sur une approximation de l'espace des solutions par un espace de fonctions de dimension finie.

Maillage de la discrétisation Un espace d'approximation adapté à de tels espaces est construit en approchant le volume Ω d'intérêt par un volume Ω^h appelé le *maillage* obtenu par agrégation d'un ensemble de polygones convexes appelés *mailles* ou *éléments*. Ces polygones définissent :

— un ensemble de points particuliers incluant en particulier leur sommet où sont estimées les inconnues ;
— des fonctions d'interpolation qui permettent des construire les solutions approchées.

Notion de champ nodal La discrétisation éléments finis permet de définir la notion de *champ nodal* : à chaque nœud est associé des valeurs qui permettent de construire une approximation d'une fonction, dont la solution du problème mécanique, le champ de déplacements.

ANNEXE A.2 PRINCIPE DES TRAVAUX VIRTUELS

La méthode présentée repose sur un découpage temporel. Connaissant le champ de déplacement et l'état mécanique des matériaux, caractérisé par des variables internes y_i, à l'instant t, la méthode des éléments finis propose de rechercher sur une géométrie discrétisée, un champ nodal $\Delta\vec{\mathbb{U}}$ telle que la fonction associée $\Delta\vec{u}^h$ vérifie une approximation du principe des travaux virtuels :

$$\forall \vec{v}^h \ C.A.^h, \quad \underbrace{\int_{\Omega^h} \underline{\sigma}_{t+\Delta t}\left(\Delta\underline{\epsilon}^{to}, t, \Delta t\right) : \underline{\epsilon}^{to}\left(\vec{v}^h\right) \ \mathrm{d}V}_{\text{Travaux intérieurs}} = \underbrace{\int_{\Omega^h} \vec{f}_{t+\Delta t}.\vec{v}^h \ \mathrm{d}V + \int_{\partial\Omega^h_{\vec{T}}} \vec{T}_{t+\Delta t}.\vec{v}^h \ \mathrm{d}S}_{\text{Travaux extérieurs}}$$

où nous avons noté $\Delta\underline{\epsilon}^{to}$ la déformation totale associée à la fonction $\Delta\vec{u}^h$ $\left(\Delta\underline{\epsilon}^{to} = \underline{\epsilon}^{to}\left(\Delta\vec{u}^h\right)\right)$. La dépendance au t symbolise la dépendance de la réponse du matériau à des variables externes (fluence, flux neutronique, etc...). L'espace $C.A.^h$ est l'ensemble des fonctions $\vec{v}^h$ correspondant aux champs nodaux vérifiant les conditions aux limites cinématiques. L'espace $C.A.^h$ étant fini, il existe deux champs nodaux $\vec{\mathbb{F}}_i\left(\Delta\vec{\mathbb{U}}\right)$ et $\vec{\mathbb{F}}_e$ tels que :

$$\begin{aligned} \forall \vec{v}^h \ C.A.^h, \quad \vec{\mathbb{F}}_i\left(\Delta\vec{\mathbb{U}}\right).\vec{\mathbb{V}}^h &= \int_{\Omega^h} \underline{\sigma}_{t+\Delta t}\left(\Delta\underline{\epsilon}^{to}, \Delta t\right) : \underline{\epsilon}^{to}\left(\vec{v}\right) \ \mathrm{d}V \\ \vec{\mathbb{F}}_e.\vec{\mathbb{V}}^h &= \int_{\Omega^h} \vec{f}_{t+\Delta t}.\vec{v}^h \ \mathrm{d}V + \int_{\partial\Omega^h_{\vec{T}}} \vec{T}_{t+\Delta t}.\vec{v}^h \ \mathrm{d}S \end{aligned}$$

Les deux champs nodaux $\vec{\mathbb{F}}_i\left(\Delta\vec{\mathbb{U}}\right)$ et $\vec{\mathbb{F}}_e$ sont appelés respectivement les *forces nodales intérieures* et *extérieures*. Le principe des travaux virtuels est alors équivalent à l'égalité des forces nodales intérieures et extérieures :

(54)
$$\boxed{\vec{\mathbb{F}}_i\left(\Delta\vec{\mathbb{U}}\right) = \vec{\mathbb{F}}_e}$$

Calcul des forces intérieures Les forces nodales intérieures peuvent être calculées à partir de la contribution de chacun des éléments finis. La reconstruction du vecteur $\vec{\mathbb{F}}_i$ à partir des contributions élémentaires $\vec{\mathbb{F}}_i^e$ est l'étape d'*assemblage*. Sur chaque élément fini, cette contribution élémentaire est calculée par intégration du champ de contraintes :

$$\vec{\mathbb{F}}_i^e = \int_{V^e} \underline{\sigma}_{t+\Delta t}\left(\Delta\underline{\epsilon}^{to}, \Delta t\right) : \underline{\mathbf{B}}\, \mathrm{d}V$$

où V^e est le volume de l'élément fini et $\underline{\mathbf{B}}$ une matrice propre à l'élément fini permettant de définir la déformation de la fonction interpolée à partir des valeurs du champ nodal aux nœuds de l'élément. La matrice $\underline{\mathbf{B}}$ est directement reliée aux gradients des fonctions d'interpolation.

Intégration numérique, points de GAUSS Numériquement, l'intégrale $\int_{v^e} \underline{\sigma}_{t+\Delta t} : \underline{\mathbf{B}}\ \mathrm{d}V$ est évaluée par quadrature : la fonction $\underline{\sigma}_{t+\Delta t} : \underline{\mathbf{B}}$ est évaluée en un certain nombre de points choisis pour minimiser l'erreur commise. Ces points particuliers sont les *points de* GAUSS de l'élément. La formule de quadrature s'écrit alors :

(55)
$$\vec{\mathbb{F}}_i^e = \sum_{i=1}^{N^G} \left(\underline{\sigma}_{t+\Delta t}\left(\Delta\underline{\epsilon}^{to}\left(\vec{\eta}_i\right), \Delta t\right) : \underline{\underline{\mathbf{B}}}\left(\vec{\eta}_i\right)\right) w_i$$

où N^G est le nombre de points de GAUSS de l'élément, $\vec{\eta}_i$ leurs coordonnées et w_i un poids associé au $i^{\text{ième}}$ point de GAUSS. Les conséquences de cette formule sont importantes : les contraintes et, par conséquent, les variables d'états z_i, n'ont besoin d'être connues qu'aux points de GAUSS.

L'étape d'*intégration locale*, c'est à dire la détermination de la contrainte en fin de pas de temps, est donc effectuée aux points de GAUSS.

Notion de résidu L'équation d'équilibre discrétisée (54) est écrite classiquement en introduisant le *résidu* $\vec{\mathbb{R}}\left(\Delta\vec{\mathbb{U}}\right)$:

(56)
$$\vec{\mathbb{R}}\left(\Delta\vec{\mathbb{U}}\right) = \vec{\mathbb{O}} \quad \text{avec} \quad \vec{\mathbb{R}}\left(\Delta\vec{\mathbb{U}}\right) = \vec{\mathbb{F}}_i\left(\Delta\vec{\mathbb{U}}\right) - \vec{\mathbb{F}}_e$$

La résolution de l'équilibre de la structure est équivalente à la recherche d'un champ nodal annulant le résidu.

ANNEXE A.3 PRINCIPE DE LA MÉTHODE DE NEWTON-RAPHSON

La *méthode de* NEWTON-RAPHSON propose de rechercher de manière itérative la solution à l'équation (56). Une estimation $\Delta\vec{\mathbb{U}}^{n+1}$ de cette solution est construite à partir de l'estimation $\Delta\vec{\mathbb{U}}^{n}$ à l'étape précédente. La relation de récurrence entre $\Delta\vec{\mathbb{U}}^{n+1}$ et $\Delta\vec{\mathbb{U}}^{n}$ s'obtient en écrivant le développement limité à l'ordre 1 du résidu $\vec{\mathbb{R}}\left(\Delta\vec{\mathbb{U}}^{n+1}\right)$ en supposant que les estimations $\Delta\vec{\mathbb{U}}^{n+1}$ et $\Delta\vec{\mathbb{U}}^{n}$ proches :

$$\vec{\mathbb{R}}\left(\Delta\vec{\mathbb{U}}^{n+1}\right) \approx \vec{\mathbb{R}}\left(\Delta\vec{\mathbb{U}}^{n}\right) + \left.\frac{\partial\vec{\mathbb{R}}}{\partial\Delta\vec{\mathbb{U}}}\right|_{\Delta\vec{\mathbb{U}}^n} . \left(\Delta\vec{\mathbb{U}}^{n+1} - \Delta\vec{\mathbb{U}}^{n}\right)$$

En écrivant que cette approximation est solution de l'équation (56) et en supposant $\left.\frac{\partial \vec{\mathbb{R}}}{\partial \Delta \vec{\mathbb{U}}}\right|_{\Delta \vec{\mathbb{U}}^n}$ inversible, nous obtenons la relation de récurrence suivante :

$$\Delta \vec{\mathbb{U}}^{n+1} = \Delta \vec{\mathbb{U}}^n - \left(\left.\frac{\partial \vec{\mathbb{R}}}{\partial \Delta \vec{\mathbb{U}}}\right|_{\Delta \vec{\mathbb{U}}^n} \right)^{-1} . \vec{\mathbb{R}} \left(\Delta \vec{\mathbb{U}}^n \right) \tag{57}$$

Nous sommes donc amenés à résoudre le problème (56) par une *succession de problèmes linéaires*.

Matrice de raideur Le calcul de la dérivée $\frac{\partial \vec{\mathbb{R}}}{\partial \Delta \vec{\mathbb{U}}}$ se fait en supposant les forces extérieures constantes au cours de la résolution :

$$\left.\frac{\partial \vec{\mathbb{R}}}{\partial \Delta \vec{\mathbb{U}}}\right|_{\Delta \vec{\mathbb{U}}^n} = \underline{\underline{\mathbb{K}}} \quad \text{avec} \quad \underline{\underline{\mathbb{K}}} = \left.\frac{\partial \vec{\mathbb{F}}_i}{\partial \Delta \vec{\mathbb{U}}}\right|_{\Delta \vec{\mathbb{U}}^n}$$

La matrice $\underline{\underline{\mathbb{K}}}$ est appelée *matrice de raideur* de la structure.

Calcul de la matrice de raideur La méthode des éléments finis permet de calculer la matrice de raideur globale par *assemblage* de *matrices de raideur élémentaires* $\underline{\underline{\mathbb{K}}}^e$. Pour chaque élément, l'expression de cette matrice de raideur élémentaire est :

$$\underline{\underline{\mathbb{K}}}^e = \sum_{i=1}^{N^G} {}^t\underline{\underline{\mathbf{B}}}(\vec{\eta}_i) : \frac{\partial \Delta \underline{\sigma}}{\partial \Delta \underline{\epsilon}^{to}}(\vec{\eta}_i) : \underline{\underline{\mathbf{B}}}(\vec{\eta}_i)\, w_i \tag{58}$$

où apparaît la *matrice tangente cohérente* $\frac{\partial \Delta \underline{\sigma}}{\partial \Delta \underline{\epsilon}^{to}}$.

Matrice de raideur tangente cohérente L'utilisation de la matrice tangente cohérente ou de la matrice sécante est la solution la plus performante en théorie car elle conduit à une convergence quadratique de la méthode de NEWTON-RAPHSON. Le coût de la réactualisation de la matrice de raideur du système à chaque itération peut cependant être important.

ANNEXE B PRINCIPALES OPÉRATIONS TENSORIELLES

Nous décrivons dans ce paragraphe les opérations tensorielles fournies par `tfel` pour les tenseurs d'ordre 2 et d'ordre 4.

ANNEXE B.1 CONVENTIONS DE REPRÉSENTATION DES TENSEURS

Les tenseurs d'ordre 2 symétriques peuvent être représentés par des vecteurs et « tenseurs d'ordre 4 », peuvent être représentés par des matrices.

En $3D$, les tenseurs d'ordre 2 symétriques ont 6 composantes. Un tenseur $\underline{\mathbf{a}}$ est représenté par le vecteur suivant :

$$\begin{pmatrix} \mathcal{A}_0 \\ \mathcal{A}_1 \\ \mathcal{A}_2 \\ \mathcal{A}_3 \\ \mathcal{A}_4 \\ \mathcal{A}_5 \end{pmatrix} = \begin{pmatrix} a_{xx} \\ a_{yy} \\ a_{zz} \\ \sqrt{2}a_{xy} \\ \sqrt{2}a_{xz} \\ \sqrt{2}a_{yz} \end{pmatrix}$$

Le racine de 2 sur les termes extradiagonaux sont là pour assurer que le produit doublement contracté de deux tenseurs d'ordre 2 symétriques $\underline{\mathbf{a}}$ et $\underline{\mathbf{b}}$ est égal au produit scalaire de leurs représentations vectorielles.

ANNEXE B.2 OPÉRATIONS SUR LES TENSEURS D'ORDRE 2

Pour les tenseurs d'ordre 2, `tfel` propose :

- la négation d'un tenseur ;
- la somme de deux tenseurs ;
- la différence de deux tenseurs ;
- la multiplication à droite ou à gauche d'un tenseur par un scalaire ;
- la division à droite d'un tenseur par un scalaire ;
- le produit tensoriel de deux tenseurs grâce à l'opérateur `^`. Le résultat est un tenseur d'ordre 4 « tenseurs d'ordre 4 » $\underline{\mathbf{a}} \otimes \underline{\mathbf{b}}$ dont la représentation matricielle vérifie a pour élément

$$\underline{\mathbf{c}} = \underline{\mathbf{a}} \otimes \underline{\mathbf{b}} \quad \Rightarrow \quad \underline{\underline{\mathbf{C}}}_{ij} = \mathcal{A}_i \, \mathcal{B}_j$$

- le produit contracté de deux tenseurs grâce à l'opérateur `|`. Comme indiqué plus haut, ce produit contracté est égal au produit scalaire de leurs représentations tensorielles :

$$\underline{\mathbf{a}} : \underline{\mathbf{b}} = \sum \mathcal{A}_i \, \mathcal{B}_i$$

En C++, ces opérations respectent l'ordre de priorité usuel sauf pour les deux derniers qui peuvent nécessiter l'emploi de parenthèses.

ANNEXE B.3 OPÉRATIONS SUR LES TENSEURS D'ORDRE 4

Pour les tenseurs d'ordre 4, `tfel` propose :

- la négation d'un tenseur ;
- la somme de deux tenseurs ;
- la différence de deux tenseurs ;
- la multiplication à droite ou à gauche d'un tenseur par un scalaire ;

— la division à droite d'un tenseur par un scalaire ;
— la multiplication de deux tenseurs ;
— l'application d'un tenseur d'ordre 4 à un tenseur d'ordre 2 par l'opérateur de multiplication $\star$. La représentation vectorielle du résultat de cette opération est égal à la multiplication vectorielle de la représentation matricielle du tenseur d'ordre 4 par la représentation vectorielle du tenseur d'ordre 2 :

$$\underline{\mathbf{c}} = \underline{\underline{\mathbf{b}}} : \underline{\mathbf{a}} \quad \Rightarrow \quad \mathcal{C}_i = \sum_j \mathcal{B}_{ij} A_j$$

— le produit contracté à gauche d'un tenseur d'ordre 4 par un tenseur d'ordre 2 grâce à l'opérateur $|$:

$$\underline{\mathbf{c}} = \underline{\mathbf{a}} : \underline{\underline{\mathbf{B}}} \quad \Rightarrow \quad \mathcal{C}_i = \sum_j \mathcal{A}_j \mathcal{B}_{ji}$$

Ces opérations respectent l'ordre de priorité usuel sauf pour le dernier qui peuvent nécessiter l'emploi de parenthèses.

ANNEXE C FONCTIONS UTILES

Cette section décrit quelques fonctions utiles à la création de lois de comportement mécanique fournies par la librairie `tfel`.

ANNEXE C.1 MANIPULATION DES TENSEURS D'ORDRE 2 SYMÉTRIQUES

Le fichier d'entête `TFEL/Math/stensor.hxx`, inclut par toutes les lois de comportement mécanique, définit la classe `stensor` représentant les tenseurs d'ordre 2 symétriques.

La méthode `computeEigenValues` La méthode `computeEigenValues` calcule les valeurs propres du tenseur d'ordre 2 symétriques. Ses valeurs propres doivent être passées en référence.

La méthode `computeEigenVectors` La méthode `computeEigenVectors` calcule les valeurs et vecteurs propres du tenseur d'ordre 2 symétriques. Son premier argument doit être un objet de type `tvector`[29] et contiendra les différentes valeurs propres. Son second argument doit être un objet de type `tmatrix` contenant les vecteurs propres[30]. Cet objet peut être passé directement à la méthode `changeBasis` .

La méthode `changeBasis` La méthode `changeBasis` effectue le changement de repère d'un tenseur d'ordre 2 symétrique. Ce changement de repère est défini par un objet de type `tmatrix`.

La fonction `trace` La fonction `trace` renvoie la trace d'un tenseur d'ordre 2 symétrique.

La fonction `sigmaeq` La fonction `sigmaeq` renvoie la contrainte équivalente au sens de VON MISES d'un tenseur d'ordre 2 symétrique.

La fonction `abs` La fonction `abs` renvoie la somme des valeurs absolues des composantes d'un tenseur d'ordre 2 symétrique, les termes extradiagonaux sont comptés une fois et sont affectés d'un facteur $\sqrt{2}$.

ANNEXE C.2 CALCUL DES COEFFICENTS DE LAMÉ

Le fichier d'entête `TFEL/MaterialLaw/Lame.hxx`, qu'il est nécessaire d'inclure au niveau du fichier `mfront` en utilisant la directive `@Includes`, définit différentes fonctions utilitaires :

- `computeLambda` permet de calculer le premier coefficent de LAMÉ. Elle prend en argument le module d'YOUNG et le coefficient de POISSON ;
- `computeMu` permet de calculer le second coefficent de LAMÉ, c'est à dire le module de cisaillement. Elle prend en argument le module d'YOUNG et le coefficient de POISSON

Ce fichier d'entête propose également une classe `computeElasticStiffness`, paramétré par la dimension d'espace et le type numérique utilisé qui permet de calculer la matrice d'élasticité à partir des coefficients de LAMÉ. Cette classe propose en méthode statique nommée `exe` qui prend en arguments :

- une référence un objet de type `st2tost2<N,T>` qui représente la matrice d'élasticité.
- le premier coefficient de LAMÉ ;

29. Les objets de type `tvector` sont définis dans le fichier d'entête `TFEL/Math/tvector.hxx`.
30. Les objets de type `tmatrix` sont définis dans le fichier d'entête `TFEL/Math/tmatrix.hxx`.

— le second coefficient de LAMÉ.

ANNEXE C.3 CALCUL DU TENSEUR DE HILL

La librairie `TFEL` fournit la fonction `hillTensor`. Cette fonction est déclarée dans le fichier d'entête `TFEL/MaterialLaw/Hill.hxx` qu'il est nécessaire d'inclure au niveau du fichier `mfront` en utilisant la directive `@Includes`.

Cette fonction est paramétrée par la dimension d'espace, notée `N` dans le fichier `mfront`, et par le type numérique utilisé, noté T. Ce type est appelé `real` dans le fichier `mfront`. Elle prend en argument les 6 coefficients F, G, H, L, M et N présentés aux paragraphes E.3.

Elle retourne un objet de type `st2tost2` qui représente une forme linéaire sur les tenseurs d'ordre 2 symétriques et qui se manipule comme une matrice[31].

Utilisation dans le cas des tubes Avec les conventions décrites au paragraphe E.4 et E.4.2 et les identifications faites aux paragraphes E.3 et E.4.2, cette fonction s'utilise ainsi :

```
st2tost2<N,real> H;
if((getModellingHypothesis()==ModellingHypothesis::PLANESTRESS)||
   (getModellingHypothesis()==ModellingHypothesis::PLANESTRAIN)||
   (getModellingHypothesis()==ModellingHypothesis::GENERALISEDPLANESTRAIN)){
   H = hillTensor<N,real>(Hzz,Hrr,Htt,
                          Hrt,Hrz,Htz);
} else {
   H = hillTensor<N,real>(Htt,Hrr,Hzz,
                          Hrz,Hrt,Htz);
}
```

31. Les objets de type `st2tost2` sont définis dans le fichier d'entête `TFEL/Math/st2tost2.hxx`.

ANNEXE D DESCRIPTION DE L'ALGORITHME RUNGE-KUTTA-CAST3M

G représente une fonction a priori non linéaire et que nous supposerons a minima continûment dérivable. On cherche à résoudre le système différentiel suivant :

$$\dot{Y} = G\left(Y, t\right) \tag{59}$$

La résolution numérique de cette équation consiste à estimer, à partir de la valeur $Y|_t$ à un instant t, la valeur $Y|_{t+\Delta t}$ à un instant $t + \Delta t$.

Voici une description de l'algorithme RUNGE-KUTTA-`Cast3M` :

— première estimation d'une solution à $\Delta t/2$:

$$\dot{Y}_1 = G\left(Y|_t, t\right) \tag{60a}$$

$$Y_1|_{t+\frac{\Delta t}{2}} = Y|_t + \frac{\Delta t}{2}\dot{Y}_1 \tag{60b}$$

— deuxième estimation d'une solution à $\Delta t/2$:

$$\dot{Y}_2 = G\left(Y_1, t\right) \tag{61a}$$

$$Y_{12}|_{t+\frac{\Delta t}{2}} = Y|_t + \frac{\Delta t}{2}\left(\frac{\dot{Y}_1 + \dot{Y}_2}{2}\right) \tag{61b}$$

— première estimation d'une solution à Δt :

$$\dot{Y}_3 = G\left(Y_{12}, t\right) \tag{62a}$$

$$Y_{13}|_{t+\Delta t} = Y_{12}|_{t+\frac{\Delta t}{2}} + \frac{\Delta t}{2}\dot{Y}_3 \tag{62b}$$

— deuxième estimation d'une solution à Δt :

$$\dot{Y}_4 = G\left(Y_{13}, t\right) \tag{63a}$$

$$Y_f|_{t+\Delta t} = Y_{12}|_{t+\frac{\Delta t}{2}} + \frac{\Delta t}{2}\left(\frac{\dot{Y}_3 + \dot{Y}_4}{2}\right) \tag{63b}$$

— troisième estimation d'une solution à Δt :

$$\dot{Y}_5 = G\left(Y_f, t\right) \tag{64a}$$

$$Y_5|_{t+\Delta t} = Y|_t + \Delta t\left(\frac{\dot{Y}_1 + 4\dot{Y}_3 + \dot{Y}_5}{6}\right) \tag{64b}$$

Dans le cas où le calcul d'une de ces estimations ne pourraient être calculés, un sous-découpage du pas de temps est effectué comme le montre l'algorithme décrivant le sous-découpage du pas temps. Ceci est illustré en figure 6.

Une fois toutes ces estimations calculées, on compare les contraintes obtenues lors des estimations $Y_f|_{t+\Delta t}$ ((63b)) et $Y_5|_{t+\Delta t}$ ((64b)), notées respectivement $\underline{\sigma}|_{t+\Delta t}$ et $\underline{\sigma}_5$.

Pour cela, on se donne une erreur dont l'amplitude est basée sur la valeur des contraintes en début de pas $\underline{\sigma}|_t$ ou sur la valeur du module d'YOUNG E :

$$errabs = \begin{cases} \sqrt{\underline{\sigma}|_t : \underline{\sigma}|_t} \times 10^{-5} \text{ si } \sqrt{\underline{\sigma}|_t : \underline{\sigma}|_t} > E \times 10^{-3} \\ E \times 10^{-8} \text{ si } \sqrt{\underline{\sigma}|_t : \underline{\sigma}|_t} < E \times 10^{-3} \end{cases}$$

On définit deux variables représentant l'écart des contraintes ra et $sqra$ définies de la manière suivante :

$$ra = \frac{\sqrt{\left(\underline{\sigma}|_{t+\Delta t} - \underline{\sigma}_5\right) : \left(\underline{\sigma}|_{t+\Delta t} - \underline{\sigma}_5\right)}}{errabs} \tag{65}$$

$$sqra = \sqrt{ra} \tag{66}$$

(67)

En testant ces deux paramètres avec des critères (fixés), on établit la convergence ou la non-convergence du calcul ainsi que le nouvel incrément de temps Δt qui sera utilisé. Dans le code de calcul, cet incrément de temps Δt est noté `dt_`. L'algorithme déterminant le nouvel incrément de temps est présenté à la figure 6. Dans cet algorithme, le temps est initial est noté `t` tandis que le temps final est noté `dt`. La variable `dtprec` est définie comme une valeur minimale du sous-découpage `dt_` en deçà de laquelle on renvoie une erreur.

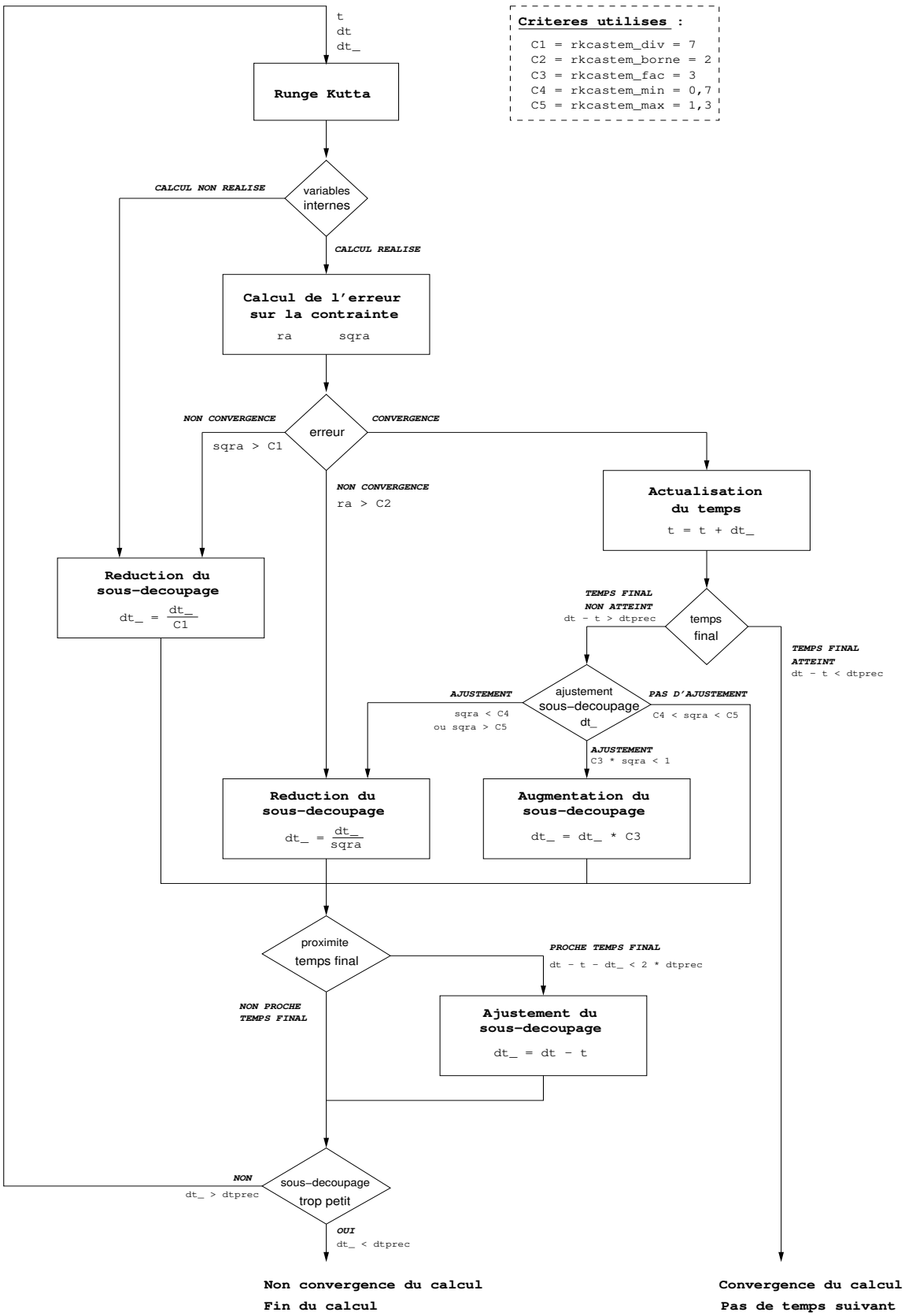


FIGURE 6 : Description de l'algorithme pour déterminer le sous-découpage du pas de temps

ANNEXE E RAPPELS SUR LES LOIS DE COMPORTEMENT ORTHOTROPES ET APPLICATION AUX TUBES

Un matériau (mécaniquement) orthotrope possède, en chaque point d'espace, trois plans de symétrie orthogonaux vis à vis de son comportement mécanique. Ces trois plans définissent trois directions perpendiculaires appelées directions d'orthotropie.

Un repère d'orthotropie est un repère orthonormé formé par ces trois directions. Le problème traité amène souvent à privilégier un de ces repères pour décrire le comportement mécanique du matériau. Le cas des tubes sera par exemple traité en détails au paragraphe E.4. Le repère est alors appelé le repère d'orthotropie du matériau. Les trois directions associées à ce repère sont simplement notées 1, 2 et 3. *Dans la suite, l'expression de la loi de comportement mécanique, de la matrice de souplesse et des différents tenseurs, se fera toujours dans ce repère.*

ANNEXE E.1 DIRECTIONS D'ORTHOTROPIE

Le tenseur des contraintes sera dans la suite représenté sous forme vectorielle. En $3D$, ce tenseur s'écrira donc :

(68)
$$\underline{\sigma} = \begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sqrt{2}\,\sigma_{12} \\ \sqrt{2}\,\sigma_{13} \\ \sqrt{2}\,\sigma_{23} \end{pmatrix}$$

De même, le tenseur des déformations totales sera représenté par le vecteur suivant :

(69)
$$\underline{\epsilon}^{to} = \begin{pmatrix} \epsilon_{11}^{to} \\ \epsilon_{22}^{to} \\ \epsilon_{33}^{to} \\ \sqrt{2}\,\epsilon_{12}^{to} \\ \sqrt{2}\,\epsilon_{13}^{to} \\ \sqrt{2}\,\epsilon_{23}^{to} \end{pmatrix}$$

Contrairement aux notations de VOIGT utilisées par le code aux éléments finis `Cast3M`, ces notations permettent de symétriser le rôle des contraintes et des déformations et de substituer au produit doublement contracté de deux tenseurs le produit scalaire de leurs représentations vectorielles. Cette symétrie des tenseurs des contraintes et de déformations est particulièrement utile quand des calculs tensoriels complexes sont nécessaires.

Notons que l'ordre des composantes de cisaillement n'est pas celui des notations de VOIGT, mais reprend celui utilisé par le code aux éléments finis `Cast3M`.

ANNEXE E.2 ÉLASTICITÉ

Dans le cadre de l'élasticité linéaire, les contraintes $\underline{\sigma}$ sont liées aux déformations élastiques $\underline{\epsilon}^{el}$ par la **matrice de rigidité**, appelée également **matrice d'élasticité** dans la suite et notée $\underline{\underline{\mathbf{D}}}$, ou par son inverse, la **matrice de souplesse**, notée $\underline{\underline{\mathbf{S}}}$, par la relation :

$$\underline{\sigma} = \underline{\underline{\mathbf{D}}} : \underline{\epsilon}^{el}, \quad \text{et} \quad \underline{\epsilon}^{el} = \underline{\underline{\mathbf{S}}} : \underline{\sigma}.$$

La matrice de souplesse d'un matériau orthotrope s'écrit, dans le repère d'orthotropie, sous la forme :

(70)
$$\underline{\underline{\mathbf{S}}} = \begin{pmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_1} & -\frac{\nu_{13}}{E_1} & 0 & 0 & 0 \\ -\frac{\nu_{21}}{E_2} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{31}}{E_3} & -\frac{\nu_{32}}{E_3} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2\,G_{12}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2\,G_{13}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2\,G_{23}} \end{pmatrix}$$

Elle est donc caractérisée par 9 coefficients indépendants :
- 3 modules d'YOUNG E_1, E_2 et E_3 ;
- 3 coefficients de POISSON ν_{12}, ν_{13} et ν_{23} ;
- 3 modules de cisaillement G_{12}, G_{13} et G_{23}.

Le facteur 2 apparaissant sur les modules de cisaillement est dû à la convention de représentation des tenseurs utilisée dans cette note (paragraphe E.1) et est introduit pour que la définition des modules d'élasticité soit cohérente avec les conventions `Cast3M` qui utilise une notation proche de celle de VOIGT. En particulier, les termes de cisaillement s'écrivent :

$$\sigma_{12} = 2\,G_{12}\epsilon^{el}_{12}$$

Les trois coefficients de Poisson, ν_{21}, ν_{31} et ν_{32} vérifient les relations imposées par la symétrie de la matrice $\underline{\underline{\mathbf{S}}}$:

$$\nu_{21} = \frac{E_2}{E_1}\,\nu_{12}, \quad \nu_{31} = \frac{E_3}{E_1}\,\nu_{13} \quad \text{et} \quad \nu_{32} = \frac{E_3}{E_2}\,\nu_{23},$$

ANNEXE E.3 CRITÈRE DE HILL

Par analogie avec l'usage courant pour les matériaux isotropes et en accord avec l'expérience, la plupart des écoulements plastiques ou viscoplastiques sont décrits en introduisant une variable scalaire de l'état des contraintes, la contrainte équivalente de HILL, notée dans la suite σ_H [Chaboche 09, Besson 01]. L'intensité de l'écoulement est fonction de cette contrainte équivalente et les composantes du tenseur des contraintes n'interviennent pas explicitement.

Bien que cela ne soit pas nécessaire, nous ferons dans la suite l'hypothèse de normalité de l'écoulement : la direction d'écoulement est donnée par la normale aux iso-surfaces de la contrainte équivalente.

Tenseur de Hill Avant de donner l'expression de la contrainte équivalente, il est pratique d'introduire un tenseur d'ordre 4, nommé dans la suite tenseur de HILL et noté $\underline{\underline{\mathbf{H}}}$:

(71)
$$\underline{\underline{\mathbf{H}}} = \begin{pmatrix} F+H & -F & -H & 0 & 0 & 0 \\ -F & G+F & -G & 0 & 0 & 0 \\ -H & -G & H+G & 0 & 0 & 0 \\ 0 & 0 & 0 & L & 0 & 0 \\ 0 & 0 & 0 & 0 & M & 0 \\ 0 & 0 & 0 & 0 & 0 & N \end{pmatrix}$$

où les 6 coefficients F, G, H, L, M et N doivent être identifiés expérimentalement.

Contrainte équivalente au sens de HILL La contrainte équivalente au sens de HILL s'écrit classiquement :

$$\sigma_H = \sqrt{\underline{\sigma} : \underline{\underline{\mathbf{H}}} : \underline{\sigma}}$$

En développant cette expression avec les conventions d'écriture du tenseur des contraintes décrites par l'équation (68), nous obtenons :

$$\sigma_H = \sqrt{(\sigma_{22}-\sigma_{11})^2\, F + (\sigma_{33}-\sigma_{22})^2\, G + (\sigma_{11}-\sigma_{33})^2\, H + 2\,\sigma_{12}^2\, L + 2\,\sigma_{13}^2\, M + 2\,\sigma_{23}^2\, N} \tag{72}$$

Tenseur normal Le tenseur normal $\underline{\mathbf{n}}$ est défini comme la normale aux iso-surfaces de la contrainte équivalente. Il est donc égal à :

$$\underline{\mathbf{n}} = \frac{\partial \sigma_H}{\partial \underline{\sigma}} = \frac{1}{2\sqrt{\underline{\sigma} : \underline{\underline{\mathbf{H}}} : \underline{\sigma}}} \frac{\partial}{\partial \underline{\sigma}} \left(\underline{\sigma} : \underline{\underline{\mathbf{H}}} : \underline{\sigma}\right) = \frac{1}{\sigma_H} \underline{\underline{\mathbf{H}}} : \underline{\sigma} \tag{73}$$

Incompressibilité Il est aisé de vérifier que le tenseur $\underline{\underline{\mathbf{H}}} : \underline{\sigma}$, et donc le tenseur normal $\underline{\mathbf{n}}$ sont déviatoriques (de trace nulle). L'écoulement associé est donc *incompressible*.

Lien avec l'isotropie Les lois d'écoulement isotropes dépendent généralement de la contrainte équivalente de VON MISES qui est donnée par la relation :

$$\sigma_{eq} = \sqrt{\underline{\sigma} : \underline{\underline{\mathbf{M}}} : \underline{\sigma}}$$

où $\underline{\underline{\mathbf{M}}}$ est le tenseur d'ordre 4 égal à $\frac{3}{2}\left(\underline{\underline{\mathbf{I}}} - \frac{1}{3}\underline{\mathbf{I}} \otimes \underline{\mathbf{I}}\right)$. Le tenseur $\underline{\underline{\mathbf{I}}} - \frac{1}{3}\underline{\mathbf{I}} \otimes \underline{\mathbf{I}}$, généralement noté $\underline{\underline{\mathbf{K}}}$, projette un tenseur d'ordre 2 sur l'espace des tenseurs déviatoriques (de trace nulle).

Le tenseur $\underline{\underline{\mathbf{M}}}$ est un cas particulier du tenseur $\underline{\underline{\mathbf{H}}}$ obtenu pour les coefficients :

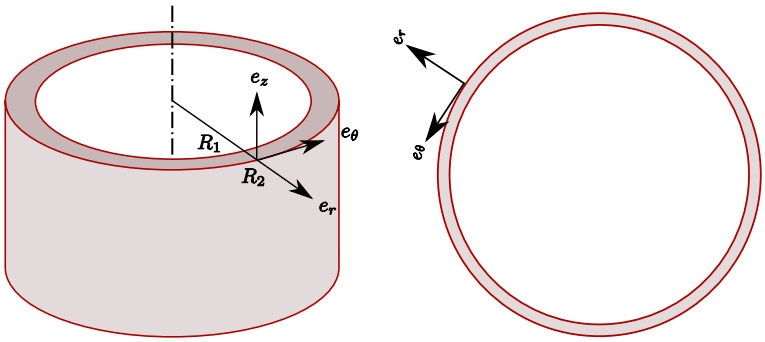
$$\begin{cases} F = G & = H = \dfrac{1}{2} \\ L = M & = N = \dfrac{3}{2} \end{cases}$$

Les termes L, M et N sont difficiles à évaluer expérimentalement (et inaccessibles dans des essais sur tube chargé en pression interne). Ils sont souvent supposés égaux à la valeur qu'ils auraient pour un matériau isotrope.

ANNEXE E.4 CONVENTIONS DE RANGEMENT DES DIRECTIONS D'ORTHOTROPIE POUR LES TUBES EN FONCTION DE L'HYPOTHÈSE DE MODÉLISATION

L'écriture des lois de comportements mécaniques sous forme tensorielle garantie théoriquement que l'écriture de la loi est indépendante de l'hypothèse de modélisation utilisée ($1D$, $2D\,(r,\theta)$, $2D\,(r,z)$ ou $3D$). Les restrictions imposées par le code aux éléments finis `Cast3M` sur la définition des directions d'orthotropie rendent cependant impossible une telle écriture. Ce paragraphe traite cette question en détails dans le cas des tubes et explicite les choix faits pour l'écriture des lois de comportements orthotropes.

Le code aux éléments finis `Cast3M` ne permettant pas à l'utilisateur de modifier le rangement des axes en $1D$, cette hypothèse de modélisation impose que :



(a) $1D$,$2D\,(r,z)$ et $3D$ (b) $2D\,(r,\theta)$

FIGURE 7 : Conventions adoptées pour la définition des axes d'orthotropies

— le premier axe d'orthotropie soit la direction radiale r ;
— le second axe d'orthotropie soit la direction axiale z ;
— la troisième direction soit la direction circonférentielle θ.

Cette convention, illustrée en figure 7(a), s'applique sans difficulté aux hypothèses de modélisations $1D$, $2D\,(r,z)$ et $3D$ mais ne peut s'appliquer aux autres hypothèses de modélisation dans le plan ($2D\,(r,\theta)$ notamment). Dans ce cas, le code aux éléments finis `Cast3M` ne permet de spécifier que la première direction d'orthotropie qui doit être dans le plan, la deuxième étant nécessairement la direction perpendiculaire dans ce plan, c'est-à-dire la direction circonférentielle, ce qui est incohérent avec le choix présenté plus haut. Le traitement des hypothèses de modélisation dans le plan autre que $2D\,(r,z)$ nécessite donc des précautions particulières que nous détaillerons plus loin.

E.4.1 Traitement des problèmes $1D$, $2D\,(r,z)$ et $3D$

Nous adoptons le rangement des axes imposé par l'hypothèse de modélisation $1D$ pour traiter les problèmes $1D$, $2D\,(r,z)$ et $3D$. Le tenseur des contraintes sera dans la suite représenté sous forme vectorielle. En $3D$, les composantes du tenseur des contraintes s'écrivent :

(74)
$$\sigma = \begin{pmatrix} \sigma_r \\ \sigma_z \\ \sigma_\theta \\ \sqrt{2}\,\sigma_{rz} \\ \sqrt{2}\,\sigma_{r\theta} \\ \sqrt{2}\,\sigma_{\theta z} \end{pmatrix}$$

Identification des coefficients de HILL Les essais sur tube permettent ainsi d'identifier des coefficients d'orthotropie généralement notés H_r, H_θ, H_z, $H_{r\theta}$, H_{rz} et $H_{\theta z}$ qui sont définis par la relation suivante :

$$\sigma_H = \sqrt{(\sigma_z-\sigma_r)^2\,H_\theta + (\sigma_\theta-\sigma_z)^2\,H_r + (\sigma_r-\sigma_\theta)^2\,H_z + 2\,\sigma_{rz}^2\,H_{rz} + 2\,\sigma_{r\theta}^2\,H_{r\theta} + 2\,\sigma_{\theta z}^2\,H_{\theta z}}$$

Cette équation permet d'identifier terme à terme les 6 coefficients F, G, H, L, M, N aux 6 coefficients H_r, H_θ, H_z, H_{rz}, $H_{r\theta}$, $H_{\theta z}$ par comparaison à l'équation (72) :

$$\begin{array}{llllll} H_r = & G & H_t = & F & H_z = & H \\ H_{rz} = & L & H_{r\theta} = & M & H_{\theta z} = & N \end{array}$$

E.4.2 Traitement des problèmes plans non axisymétriques

Nous avons vu au paragraphe E.4 que les problèmes plans ne peuvent être traités avec les mêmes conventions que les problèmes $1D$, $2D\,(r,z)$ et $3D$. Pour ces problèmes, la première direction d'orthotropie, dans le cas d'un tube est naturellement la direction radiale (r). La deuxième est la direction circonférentielle (θ) et la troisième la direction axiale (z). Ce choix est illustré en figure 7(b).

Le tenseur des contraintes s'écrit alors :

(75)
$$\sigma = \begin{pmatrix} \sigma_r \\ \sigma_\theta \\ \sigma_z \\ \sqrt{2}\,\sigma_{r\theta} \\ \sqrt{2}\,\sigma_{rz} \\ \sqrt{2}\,\sigma_{\theta z} \end{pmatrix}$$

Identification des coefficients de HILL L'expression de la contrainte équivalente σ_H est :

(76)
$$\sigma_H = \sqrt{(\sigma_\theta - \sigma_r)^2\ F + (\sigma_z - \sigma_\theta)^2\ G + (\sigma_r - \sigma_z)^2\ H + 2\,\sigma_{r\theta}^2\,L + 2\,\sigma_{rz}^2\,M + 2\,\sigma_{\theta z}^2\,N}$$

ce qui conduit à l'identification :

$$\begin{array}{lll} H_r = G & H_t = H & H_z = F \\ H_{rz} = M & H_{r\theta} = L & H_{\theta z} = N \end{array}$$

E.4.3 Cas particulier

Certains matériaux orthotropes présentent ou sont supposés présenter des propriétés équivalentes dans les directions θ et z. L'ensemble des lois de comportement des composites en carbure de silicium identifiées sur tube présentent cette spécificité.

Pour ces matériaux, il est possible d'avoir une définition unique quelque soit l'hypothèse de modélisation.

Variante	Nombre de cycles	Ratio par rapport à l'algorithme de NEWTON
J exact	$4\,707\,335$	1
J égal à l'identité	pas de convergence	
$\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta\, \underline{\epsilon}^{el}}$ et $\frac{\partial f_p}{\partial \Delta\, p}$ exacts	pas de convergence	
$\frac{\partial f_p}{\partial \Delta\, \underline{\epsilon}^{el}}$ et $\frac{\partial f_p}{\partial \Delta\, p}$ exacts, et $\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta\, \underline{\epsilon}^{el}}$ égal à l'identité	pas de convergence	
$\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta\, p}$ et $\frac{\partial f_p}{\partial \Delta\, p}$ exacts, et $\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta\, \underline{\epsilon}^{el}}$ égal à l'identité	$152\,721\,841$	32.44

TABLEAU 7 : Temps calculs obtenus avec l'algorithme de NEWTON-RAPHSON.

ANNEXE F COMPARAISONS NUMÉRIQUES DE DIFFÉRENTS ALGORITHMES DE RÉSOLUTION

ANNEXE F.1 LOI UTILISÉE POUR LA COMPARAISON

Pour ces premières évaluations, nous avons utilisés un écoulement viscoplastique de NORTON. Il s'agit d'une loi extrêmement simple convergent en peu d'itérations. Le coût intrinsèque des différentes méthodes est donc exacerbé. Cette loi présente deux variables internes ont été retenues, la déformation élastique $\underline{\epsilon}^{el}$ et la déformation viscoplastique cumulée p.

ANNEXE F.2 RÉSULTATS OBTENUS

Nous comparons, dans le cas d'un essai en traction uniaxiale, l' algorithme de NEWTON à l'algorithme de BROYDEN.

Pour l'algorithme de NEWTON, nous avons introduit une variante dans laquelle le jacobien est estimé numériquement (avec une différence finie d'ordre 1 ou d'ordre 2) et réactualisé à différentes périodes (tableaux 8 et 9).

Pour le premier algorithme de BROYDEN, nous avons testé différentes variantes suivantes (tableaux 10 et 11) :
- — que le jacobien initial était pris égal à l'identité ou évalué numériquement par une différence finie d'ordre 2 ;
- — que des termes du jacobien étaient calculés explicitement.

Pour le second algorithme de BROYDEN, nous avons distingué deux variantes suivant que le jacobien inital était pris égal à l'identité ou évalué numériquement par une différence finie d'ordre 2.

Quand ils convergent, les algorithmes donnent toujours la même solution, ce qui est illustré en figure 8. Les coûts des différentes méthodes, en termes de nombre cycles processeurs [32], sont fournis dans les tableaux 7, 8, 9, 10, 11 et 12.

Le test effectué étant extrêmement frustre, il serait mal avisé d'être conclusif.

Le tableau 7 montre à quel point l'algorithme de NEWTON est sensible à la qualité de la matrice jacobienne.

Les tableaux 8 et 9 montrent que les évalutions numériques des jacobiens sont assez coûteuses. On peut

32. Il s'agit d'une mesure fiable du coût d'une fonction, indépendante de la charge du système. Nous avons utilisé l'outil `valgrind` pour avoir accès à cette mesure.

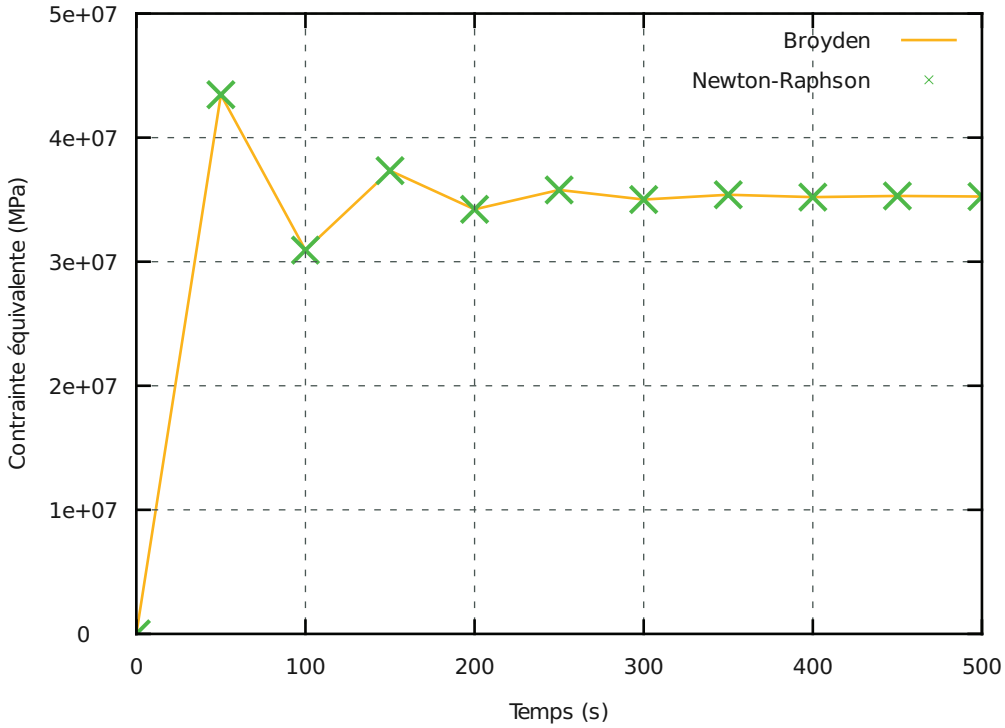


FIGURE 8 : Contraintes équivalentes obtenues respectivement par une résolution par un algorithme de NEWTON-RAPHSON ou par un algorithme de BROYDEN.

Variante	Nombre de cycles	Ratio par rapport à l'algorithme de NEWTON
$\underset{\sim}{J}$ réactualisé à chaque itération	$9\,900\,064$	$2,10$
$\underset{\sim}{J}$ réactualisé toutes les 2 itérations	$8\,674\,464$	$1,84$
$\underset{\sim}{J}$ réactualisé toutes les 3 itérations	$9\,159\,968$	$1,94$

TABLEAU 8 : Temps calculs obtenus avec l'algorithme de quasi-NEWTON et un calcul numérique du jacobien par une différence finie (approximation d'ordre 1).

Variante	Nombre de cycles	Ratio par rapport à l'algorithme de NEWTON
$\underset{\sim}{J}$ réactualisé à chaque itération	$14\,916\,316$	$3,16$
$\underset{\sim}{J}$ réactualisé toutes les 2 itérations	$11\,773\,880$	$2,5$
$\underset{\sim}{J}$ réactualisé toutes les 3 itérations	$13\,330\,216$	$2,83$

TABLEAU 9 : Temps calculs obtenus avec l'algorithme de quasi-NEWTON et un calcul numérique du jacobien par une différence finie centrée (approximation d'ordre 2).

Variante	Nombre de cycles	Ratio par rapport à l'algorithme de NEWTON
Défaut	$20\,197\,423$	$4,29$
J exact	$6\,120\,862$	$1,3$
$\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta \underline{\epsilon}^{el}}$ exact	$36\,821\,766$	$7,82$
$\frac{\partial f_p}{\partial \Delta p}$ exact	$33\,826\,368$	$7,186$
$\frac{\partial f_p}{\partial \Delta \underline{\epsilon}^{el}}$ exact	$19\,577\,698$	$4,15$
$\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta p}$ exact	$12\,132\,956$	$2,58$
$\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta p}$ et $\frac{\partial f_p}{\partial \Delta \underline{\epsilon}^{el}}$ exacts	$4\,686\,228$	$0,995$

TABLEAU 10 : Temps calculs obtenus avec le premier algorithme de BROYDEN et jacobien initial égal à l'identité.

Variante	Nombre de cycles	Ratio par rapport à l'algorithme de NEWTON
Défaut	$9\,535\,347$	$2,02$
$\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta p}$ et $\frac{\partial f_p}{\partial \Delta \underline{\epsilon}^{el}}$ exacts	$8\,178\,547$	$1,73$

TABLEAU 11 : Temps calculs obtenus avec le premier algorithme de BROYDEN et jacobien initial approximé numériquement.

également noté que si l'on gagne du temps à n'évaluer la matrice que toutes les deux itérations, on en perd à ne l'évaluer toutes les trois itérations car la convergence est dégradée. Il n'y a pas de gain à utiliser une différence finie d'ordre 2.

Il est intéressant de noter que les algorithmes de BROYDEN convergent toujours.

Le cas où tous les termes de la jacobienne sont fournis permet de comparer le coût intrinsèque plus élevé du premier algorithme de BROYDEN (tableau 11).

Si aucun terme du jacobien n'est fourni, le premier algorithme de BROYDEN peut être relativement efficace, à condition d'estimer numériquement la jacobienne initiale (tableau 11).

De plus, il est intéressant de s'intéresser au terme $\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta \underline{\epsilon}^{el}}$ de la matrice jacobienne. Il s'agit du terme le plus complexe (et le plus coûteux) à calculer. Nous pouvons noter que si l'on ne calcule pas ce terme dans l'algorithme de BROYDEN, mais que l'on calcule les autres, l'algorithme converge et est légèrement meilleur que l'algorithme de NEWTON.

Variante	Nombre de cycles	Ratio par rapport à l'algorithme de NEWTON
Défaut	$19\,530\,077$	$4,15$

TABLEAU 12 : Temps calculs obtenus avec le second algorithme de BROYDEN et jacobien initial égal à l'identité.

ANNEXE G NOMS DES VARIABLES ET DES MÉTHODES C++ UTILISÉS EN INTERNE PAR LES ANALYSEURS DE LOIS DE COMPORTEMENT

ANNEXE G.1 NOMS COMMUNS À TOUS LES ANALYSEURS DE LOIS DE COMPORTEMENT

Certain variables sont utilisées en interne et ne peuvent être utilisées par l'utilisateur.

Certaines des variables ont des sens physiques :

- `D` désigne le tenseur d'élasticité. Ce tenseur est calculé à la demande, voir à ce sujet la documentation de la directive `@RequireStiffnessTensor`.
- `Dt` désigne la matrice tangente, éventuellement cohérente ;
- `sig` désigne les contraintes ;
- `F0` désigne le gradient de la transformation en début de pas ;
- `F1` désigne le gradient de la transformation en fin de pas ;
- `eto` désigne les déformations totales ;
- `deto` désigne l'incrément des déformations totales ou leurs vitesses ou sa vitesse (voir les conventions spécifiques de chaque analyseur) ;
- `T` est la température, généralement en début de pas (voir les conventions spécifiques de chaque analyseur) ;
- `dT` est l'incrément de la température ou sa vitesse de variation (voir les conventions spécifiques de chaque analyseur) ;
- `dt` est l'incrément de temps ;
- `N` est la dimension spatiale associée à l'hypothèse de modélisation considérée (1, 2 ou 3) ;

Certaines noms sont utilisés en interne :

- `Type` est le nom du type numérique utilisé pour les calculs ;
- `use_qt` est un booléen. Si il est vrai, les variables utilisées sont affectées d'unités et la cohérence des calculs, au sens du respect des unités, est assurée par le compilateur ;
- `std`, `tfel`, `math`, `material` et `utilities` qui désignent des espaces de noms des librairies C++ utilisées pour l'intégration des lois ;
- `real` qui est un alias du type réel utilisé par la routine d'intégration ;
- `policy` et `policy_value` qui déterminent la politique à suivre en cas de dépassement des bornes ;
- `src1`, `src2` sont des noms de variables utilisés par les constructeurs des classes générés ;
- `integrate` est le principal point d'entrée méthode d'intégration ;
- `computeStress` qui recalcule la valeur du tenseur des contraintes au cours de l'intégration ;
- `computeFinalStress` qui calcule la valeur du tenseur des contraintes en fin de pas ;
- `computeFdF` qui calcule la valeur de la fonction à annuler et de sa dérivée pour les méthodes implicites ;
- `updateStateVars` qui met à jour les valeurs des variables d'état, après un pas d'intégration réussi ;
- `updateAuxiliaryStateVars` qui met à jour les valeurs des variables d'état auxiliaires, après un pas d'intégration réussi ;
- `getModellingHypothesis` qui permet de récupérer l'actuelle hypothèse de modélisation ;
- `getTimeStepScalingFactor` qui retourne un estimation du pas de temps à utiliser (pour le pas de temps courant ou pour le pas de temps suivant ;
- `getTangentOperator` qui retourne la matrice tangente, éventuellement cohérente, si celle-ci est disponible ;
- `hypothesis`, qui est une variable contenant l'hypothèse de modélisation ;
- `hypothesis_`, qui est utilisé dans les constructeurs pour initialiser la valeur de la variable `hypothesis`.

ANNEXE G.2 NOMS DE VARIABLES RÉSERVÉS PAR LES ANALYSEURS SPÉCIFIQUES

Les variables suivantes sont utilisées en interne par les analyseurs spécifiques et leurs noms ne peuvent être réutilisés par l'utilisateur :

- `NewtonIntegration` est le nom de la méthode réalisant les itérations de la méthode de NEWTON ;
- `theta` est le nom du paramètre de la méthode implicite ;
- `epsilon` est le nom de la valeur du critère utilisé pour arrêter les itérations de la méthode de NEWTON ;
- `iterMax` est le nom du nombre d'itérations autorisées pour la méthode de NEWTON ;
- `p` est la déformation inélastique cumulée est le nom de la déformation plastique cumulée ;
- `n` est le nom de la direction d'écoulement, donnée par la normale aux isovaleurs de contrainte équivalente.

ANNEXE G.3 NOMS DE VARIABLES RÉSERVÉS PAR L'ANALYSEUR RUNGE-KUTTA

- `epsilon`
- `dtmin`
- `eel`
- `deel`
- `t`
- `T_`
- `eto_`
- `deto_`
- `dt_`
- `corrector`
- `dtprec`
- `converged`
- `error`
- `failed`

ANNEXE G.4 NOMS DE VARIABLES RÉSERVÉS PAR L'ANALYSEUR IMPLICITE

- `theta`
- `epsilon`
- `iterMax`
- `jacobianComparisonCriterium`
- `relaxationTrigger`
- `accelerationTrigger`
- `accelerationPeriod`
- `relaxationCoefficient`
- `eel`
- `deel`
- `previous_zeros`
- `zeros`
- `tzeros`
- `zeros_1`
- `fzeros`
- `tfzeros`
- `zeros2`
- `fzeros2`
- `Dzeros`
- `Dfzeros`

— jacobian
— tjacobian
— njacobian
— jacobian2
— t
— dt_
— error
— idx
— idx2
— schmidt
— computeNumericalJacobian
— accelerate
— accelerate_k0
— accelerate_k1
— accelerate_k2
— accelerate_c0
— accelerate_c1
— accelerate_re0
— accelerate_re1
— accelerate_r0
— accelerate_r1
— accelerate_r2
— iter
— converge
— broyden_inv

INDEX DES FICHIERS D'ENTÊTE FOURNIS PAR LA LIBRAIRIE TFEL

T

TFEL/MaterialLaw/Hill.hxx 70
TFEL/MaterialLaw/Lame.hxx 69
TFEL/MaterialLaw/ModellingHypothesis.hxx 10
TFEL/Math/st2tost2.hxx 70
TFEL/Math/stensor.hxx 69
TFEL/Math/tmatrix.hxx 69
TFEL/Math/tvector.hxx 69

INDEX DES CLASSES, DES MÉTHODES ET DES FONCTIONS FOURNIES PAR LA LIBRAIRIE TFEL

A

abs 69

C

compute Lambda 69
compute Mu 69
computeElasticStiffness 69
computeElasticStiffness : : exe 69

H

hillTensor 70

M

ModellingHypothesis 10
ModellingHypothesis : : Hypothesis 10

S

sigmaeq 69
st2tost2<N,T> 69, 70
stensor<N,T,Storage> 69
stensor<N,T,Storage> : : changeBasis 69
stensor<N,T,Storage> : : computeEigenValues 69
stensor<N,T,Storage> : : computeEigenVectors 69

T

tmatrix 69
trace 69
tvector 51, 69

INDEX DES DIRECTIVES

A

@AccelerationPeriod . . . 49
@AccelerationTrigger . . . 49
@Algorithm . . . 32, 40
@Author . . . 28
@AuxiliaryStateVariable . . . 26

B

@Behaviour . . . 28
@Bounds . . . 13

C

@Compare To Numerical Jacobian . . . 46
@ComputeStress . . . 36, 52
@ComputeThermalExpansion . . . 12
@Constant . . . 12

D

@Date . . . 28
@Derivative . . . 35–37
@Description . . . 28

E

@Epsilon . . . 26, 35, 37, 53
@External State Variable . . . 28

F

@FlowRule . . . 25, 28

I

@Includes . . . 69, 70
@InitLocalVariables . . . 9, 12, 28
@InitLocalVars . . . 9, 28
@Integrator . . . 29, 52, 55
@IsTangentOperatorSymmetric . . . 49
@IterMax . . . 26

L

@Local Variable . . . 28
@LocalVariable . . . 28

M

@MaterialLaw . . . 8
@MaterialProperty . . . 8
@MinimalTimeStep . . . 32, 37
@ModellingHypotheses . . . 10
@ModellingHypothesis . . . 10

O

@Orthotropic Behaviour . . . 29

P

@Parser 28
@PhysicalBounds 13
@Predictor 43

R

@RelaxationCoefficient 49
@RelaxationTrigger 49
@Require Stiff ness Tensor 29
@RequireStiffnessTensor 82

S

@Static Variable 28
@StaticVariable 28

T

@TangentOperator 49, 51
@Theta 26, 43, 53

U

@UpdateAuxiliaryStateVariables 25, 26
@UseAcceleration 49