# L'interface `umat` aux lois de comportements mécaniques de `mfront`

**T. Helfer**

**2015**

## RÉSUMÉ

# SOMMAIRE


**1 INTRODUCTION . . . . . 3**

**2 DE LA BONNE UTILISATION DES LOIS DE COMPORTEMENT MFRONT DANS CAST3M . . . . . 3**

2.1 LOIS ÉCRITES EN PETITES DÉFORMATIONS . . . . . 3

*2.1.1 Incompatibilité avec les formulations hypoélastiques de Cast3M . . . . . 3*

*2.1.2 Comparaison avec les lois natives de Cast3M ou à des lois umat écrites en fortran . . 4*

2.2 LOIS ÉCRITES EN GRANDES TRANSFORMATIONS . . . . . 4

2.3 UTILISATION DES LOIS GÉNÉRÉES PAR MFRONT DANS CAST3M . . . . . 4

**3 FONCTIONNALITÉS DE L'INTERFACE UMAT . . . . . 5**

3.1 UTILISATION DES LOIS ÉCRITES EN PETITES DÉFORMATIONS DANS UN CONTEXTE DE TRANSFORMATIONS FINIES . . . . . 5

3.2 TRAITEMENTS DE MATÉRIAUX ORTHOTROPES . . . . . 5

3.3 GESTION GÉNÉRIQUE DES CONTRAINTES PLANES . . . . . 6

3.4 SOUS-DÉCOUPAGE AUTOMATIQUE DU PAS DE TEMPS . . . . . 6

*3.4.1 Description de l'algorithme . . . . . 7*

*3.4.2 Directives gérant le sous-découpage du pas de temps . . . . . 7*

3.5 GÉNÉRATION AUTOMATIQUE D'UN FICHIER mtest EN CAS D'ÉCHEC . . . . . 7

3.6 GESTION DES BORNES DE VALIDITÉ . . . . . 7

3.7 MODIFICATION DES PARAMÈTRES . . . . . 7

3.8 GÉNÉRATION DE LA MISE EN DONNÉE CAST3M . . . . . 8

**RÉFÉRENCES . . . . . 10**

**LISTE DES TABLEAUX . . . . . 11**

**LISTE DES FIGURES . . . . . 12**

**ANNEXE A DESCRIPTION DE L'INTERFACE . . . . . 13**

ANNEXE A.1 CHOIX DU TYPE DE CALCUL . . . . . 13

ANNEXE A.2 MÉTA-DONNÉES . . . . . 13

**ANNEXE B RAPPEL SUR LES FORMULATIONS HYPOÉLASTIQUES DE CAST3M . . . . . 13**

ANNEXE B.1 LIMITES INTRINSÈQUES DES FORMULATIONS HYPOÉLASTIQUES . . . . . 13

ANNEXE B.2 CHOIX DE LA FORMULATION HYPOÉLASTIQUE . . . . . 14

# 1 INTRODUCTION

Nous présentons dans cette note l'interface `umat` utilisée par le code aux éléments finis `Cast3M` pour l'utilisation de lois de comportement mécanique externes.

Cette note se décompose en deux parties :
- la première décrit comment utiliser correctement les lois de comportement générées par `mfront` dans `Cast3M`;
- la seconde décrit les fonctionnalités de l'interface `umat` ;

Cette note suppose que le lecteur soit familier avec l'écriture de lois de comportement mécanique à l'aide de `mfront`, écriture qui est décrite en détail par ailleurs [Helfer 13].

# 2 DE LA BONNE UTILISATION DES LOIS DE COMPORTEMENT MFRONT DANS CAST3M

Nous considérons dans cette section les conditions d'une utilisation correcte des lois de comportement `mfront` dans la procédure `PASAPAS` de `Cast3M` 2014. Pour les utilisateurs PLEIADES, ces remarques s'appliquent également aux procédures spécifiques `INCREPL`/`MEPL`.

Nous commençons par considérer les lois écrites en petites déformations, puis les lois écrites en grandes transformations.

## 2.1 LOIS ÉCRITES EN PETITES DÉFORMATIONS

Les lois de comportement `mfront` écrites en petites déformations sont utilisables dans `PASAPAS` si l'option `GRANDS_DEPLACEMENTS` est mise à `FAUX`.

Ces lois peuvent décrire des matériaux isotropes et orthotropes. Le cas orthotrope nécessite quelques précautions rappelées en section 3.2.

### 2.1.1 Incompatibilité avec les formulations hypoélastiques de `Cast3M`

Lorsque l'on active la gestion des grands déplacements, `Cast3M` propose différentes formulations hypoélastiques pour traiter les lois écrites dans le formalisme des petites transformations. Une description sommaire des formulations hypoélastiques est donnée en annexe B.

Le point important est que les lois de comportement écrites en `mfront` sont généralement incompatibles avec les formulations hypoélastiques de `Cast3M` (sauf cas particulier construit dans ce but).

L'une des manière de mettre ce point en évidence est que pour tenir compte de la variation des propriétés élastiques avec la température notamment, *il est d'usage dans `mfront` d'utiliser la déformation élastique comme variable interne*. Or une telle variable interne est prohibée pour les raisons décrites dans l'annexe B.

Cette incompatibilité est d'autant plus regrettable que :
- `Cast3M` impose le recours à l'option `GRANDS_DEPLACEMENTS` dès qu'il est nécessaire de décrire le contact entre différents corps ;
- comme nous l'avons souligné dans l'annexe B, tout semble fait pour que les lois de comportement écrites en petites déformations continue de fonctionner « comme si de rien n'était ».

En conséquence, de nombreux calculs sont aujourd'hui effectués en combinant de manière inappropriée des lois `mfront` aux formulations hypoélastiques de `Cast3M`.

Pour contrebalancer cette vision pessimiste, nous pouvons quand même espérer, dans les cas où les déformations et les rotations restent faibles, c'est à dire dans les cas où l'hypothèse des petites perturbations reste raisonnablement vérifiée, que les erreurs induites sur les résultats soient faibles. D'après notre expérience, le cas de chargement monotone semble également favorable.

Si l'on souhaite utiliser des modèles écrits avec le même formalisme qu'en petites transformations dans un contexte de grandes transformations, `mfront` propose différentes stratégies, exactes, décrites en section 3.1. Nous suggérons fortement d'utiliser ces stratégies plutôt que les formulations hypoélastiques de `Cast3M`.

### 2.1.2 Comparaison avec les lois natives de `Cast3M` ou à des lois `umat` écrites en fortran

Si l'on est amené à comparé les lois `mfront` aux lois natives de `Cast3M` ou à des lois `umat` écrites en fortran, il faut se prendre garde au fait que ces lois sont généralement écrites en vitesses pour être compatible avec les formulations hypoélastiques décrites plus haut.

*En particulier, pour les lois exposées, elles ne peuvent en général pas traiter de manière exacte le cas des propriétés élastiques variables dans le temps.* Nous avons donner les raisons de cette limite au paragraphe (B). Notons en particulier que l'élasticité, traitée de manière incrémentale pour pouvoir être utilisée avec une formulation hypoélastique, conduit à des résultats erronés dans ce cas.

## 2.2 Lois écrites en grandes transformations

Les lois `mfront` écrites en grandes transformations peuvent être utilisées dans `Cast3M`.

Il faut cependant prendre garde à explicitement préciser `'EPSILON' 'UTILISATEUR'` lors de la définition du modèle. Ce point est important car si l'on ne le fait pas, les contraintes seront modifiées comme dans les cas des formulations hypo-élastiques par l'opérateur `PICA`. Il n'y a malheureusement aucun garde-fous actuellement dans `Cast3M` pour éviter cette erreur.

## 2.3 Utilisation des lois générées par `mfront` dans `Cast3M`

Cette section suppose que l'utilisateur sait comment introduire une loi `umat` dans `Cast3M` et est familier avec la syntaxe `gibiane` associée.

```
* Création du modèle mécanique
ModM1 = 'MODELISER' s1 'MECANIQUE' 'ELASTIQUE'
'ISOTROPE' 'NON_LINEAIRE' 'UTILISATEUR'
'LIB_LOI' 'libMFrontCastemBehaviours.so'
'FCT_LOI' 'umatcompressibleneohookeanbehaviour'
'C_MATERIAU' CLOI
'C_VARINTER' VLOI
'PARA_LOI' PLOI;
```

**Figure 1 :** Appel d'une loi `mfront` dans la version `pleiades` de `Cast3M`.

L'appel, illustré en figure 1 présente deux mots clés intéressants :

- l'indice `LIB_LOI` désigne le nom de la librairie dynamique qui contient la loi de comportement. L'extension de la librairie dépend du système : `so` sous `unix`, `dll` sous `Windows`. Il est possible de donner le chemin, absolu ou relatif, vers cette librairie, mais cette manière de faire est déconseillée. Il est préférable de modifier la variable d'environnement `LD_LIBRARY_PATH` sous `unix`, ou `PATH` sous `Windows` ;
- l'indice `FCT_LOI` qui contient le nom de la fonction à appeler.

Ces mots clés sont disponibles dans la version 2014 PLEIADES de Cast3M et seront standard dans la version 2015. Le site mfront décrit comment utiliser les lois mfront dans les versions officielles de Cast3M antérieures.

# 3 FONCTIONNALITÉS DE L'INTERFACE UMAT

## 3.1 UTILISATION DES LOIS ÉCRITES EN PETITES DÉFORMATIONS DANS UN CONTEXTE DE TRANSFORMATIONS FINIES

Les lois de comportement générées par mfront sont généralement écrites avec l'hypothèse des petites déformations et ne peuvent en l'état être utilisées dans des calculs en transformations finies. Nous avons vu au paragraphe B pourquoi l'utilisation des formulations hypoélastiques de Cast3M est insatisfaisant.

Il existe cependant différentes façons d'adapter des lois de comportements en petites déformations au contexte des transformations finies. Trois stratégies sont disponibles :

- un cas simple, en terme d'implantation informatique, est celui de cas des grandes rotations mais petites déformations [Proix 13]. Cette stratégie est intéressante car elle permet de réutiliser telle quelle des lois identifiées en petites déformations ;
- un cas plus complexe qui consiste à déduire du gradient de la transformation des déformations dites logarithmiques. Elle a été proposée par Miehe et est disponibles dans Aster et ZeBuLoN [Miehe 02, Bargellini 13] ;
- la dernière stratégie permet l'utilisation des déformations logarithmiques ci-dessus dans le cadre d'un code $1D$ écrit en petites déformations.

L'utilisateur peut choisir d'utiliser l'une et l'autre des ces stratégies par la directive @UMATFiniteStrainStrategy qui admet pour arguments FiniteRotationSmallStrain, MieheApelLambrechtLogarithmicStrain ou LogarithmicStrain1D.

Ces différentes stratégies sont exemptes des défauts des formulations hypoélastiques de Cast3M. Elles permettent de construire des lois objectives capables de décrire des matériaux isotropes ou orthotropes, de tenir compte de la variation des propriétés matériau avec la température. De plus, appliquées à une loi élastiques, elles conduisent à une loi hyperélastique.

## 3.2 TRAITEMENTS DE MATÉRIAUX ORTHOTROPES

Dans le cas des matériaux orthotropes, l'interface umat peut être amenée à gérer plusieurs aspects du calcul :

- dans le cas de matériaux orthotropes, l'interface umat se charge des changements de repère. Ils sont assurés par la classe UMATRotationMatrix.
- l'interface se charge du calcul de la matrice d'élasticité, si l'utilisateur le demande via la directive @RequireStiffnessTensor. Il est important de noter que l'interface umat passe cette matrice d'élasticité *après* l'appel au constructeur de la loi de comportement, ce qui signifie qu'elle ne peut être utilisée dans la phase d'initialisation introduite par le mot-clé InitLocalVariables.
- l'interface se charge du calcul du tenseur des coefficients de dilatation thermique, si l'utilisateur le demande via la directive @RequireThermalExpansionTensor. Il est important de noter que l'interface umat passe ce tenseur *après* l'appel au constructeur de la loi de comportement, ce qui signifie qu'il ne peut être utilisé dans la phase d'initialisation introduite par le mot-clé InitLocalVariables. Ce tenseur est calculé par la classe UMATComputeThermalExpansionTensor.

Les lois orthotropes posent le problème d'une définition cohérente des axes d'orthotropie. Cast3M (ni aucun autre code aux éléments finis à notre connaissance) ne permet pas une définition cohérente pour l'ensemble des hypothèses de modélisations. Nous recommandons de suivre la démarche que nous avons proposée pour les tubes dans la note générale sur l'écriture des lois de comportement mécanique [Helfer 13]

## 3.3 GESTION GÉNÉRIQUE DES CONTRAINTES PLANES

L'hypothèse de contraintes planes constituent un cas particulier :

— la loi de comportement doit assurer en interne le respect de la condition de contraintes planes ;
— la troisième composante de la déformation axiale n'est pas renseignée. En effet, celle-ci ne peut être déduite du champs de déplacement et `Cast3M` lui affecte par défaut une valeur nulle.

Dans cette hypothèse, la déformation axiale est une inconnue supplémentaire du problème.

Pour tenir compte de ces particularités, deux voies sont possibles :

— écrire un algorithme de résolution spécifique qui tient explicitement compte de la condition de contraintes planes. La note `mfront` décrivant le support des contraintes planes montre comment cela peut être réalisé [Helfer 14]. Il s'agit de la voie la plus efficace ;
— réutiliser l'intégration en *déformations planes* au sein d'un algorithme de résolution qui se charge de trouver une déformation axiale telle que la contrainte axiale soit nulle. Cette dernière solution est particulièrement intéressante car elle fait l'économie de ce développement spécifique. Il est néanmoins certain que cette méthode a un coût puisque l'intégration en déformations planes sera faites plusieurs fois [Proix 12].

Si un analyseur ne gère pas les contraintes planes, l'interface `umat` utilise l'algorithme suivant :

— l'incrément de déformation axiale $\Delta\,\epsilon_z^{to\,(0)}$ est initialisé par une estimation élastique. Dans le cas d'un comportement élastique isotrope, cela se traduit par :

$$\Delta\,\epsilon_z^{to\,(0)} = -\frac{1}{E}\,\sigma_z{}^{(0)} - \nu\left(\Delta\,\epsilon_x^{to} + \Delta\,\epsilon_y^{to}\right)$$

La présence du terme $\frac{1}{E}\sigma_z{}^{(0)}$ permet de prendre en corriger le fait que la condition de contraintes planes n'est pas vérifiée de manière exacte. La loi de comportement en déformations planes est alors appelée.

— si la contrainte axiale obtenue n'est pas nulle, au critère près, une nouvelle correction élastique est effectuée :

$$\Delta\,\epsilon_z^{to\,(1)} = \Delta\,\epsilon_z^{to\,(0)} - \frac{1}{E}\,\frac{1}{E}\sigma_z{}^{(1)}$$

La loi de comportement en déformations planes est appelée une seconde fois.

— si la contrainte axiale obtenue n'est pas nulle, la méthode itérative de la sécante est utilisée pour estimer un incrément convenable.

Le critère choisi pour tester la condition de contraintes planes est, dans le cas isotrope :

$$\left|\sigma_z{}^{(n)}\right| < E\,\varepsilon$$

où $E$ est le module d'YOUNG du matériau et où $\varepsilon$ est choisi égal à $10^{-12}$ [1].

## 3.4 SOUS-DÉCOUPAGE AUTOMATIQUE DU PAS DE TEMPS

L'intégration d'une loi de comportement peut échouer. La loi de comportement peut également indiquer que les résultats obtenus sont imprécis.

Pour gérer ces cas, l'interface `umat` permet de sous-découper localement, au niveau des points de GAUSS, le pas de temps.

**Note** Il est important de réaliser que l'intégration peut échouer ou indiquer que les résultats sont imprécis pour deux raisons :

1. Cette valeur n'est aujourd'hui pas modifiable

— l'intégration locale peut effectivement être très difficile ;
— le pas de temps utilisé globalement pour la recherche d'équilibre est trop grand. En particulier, ce pas de temps peut ne pas respecter l'hypothèse de radialité du chargement [Proix 11].

Le défault de méthode présentée est de ne pas distinguer ces deux cas. En particulier, elle peut permetre la convergence de l'intégration locale en masquant le fait que la recherche de l'équilibre global est fondamentalement faussée par un pas de temps trop grand.

### 3.4.1 Description de l'algorithme

Pour décrire l'algorithme utilisé, notons $\Delta t$ le pas de temps utilisé par l'algorithme de résolution global, $\delta t$ un pas de temps de courant et $n$ le nombre d'étapes d'intégration qu'il reste pour réaliser l'intégration sur l'intégralité du pas de temps.

En début d'intégration, le pas de temps courant $\delta t$ est initialisé à $\Delta t$ et $n$ est initialisé à $1$.

En cas de sous-découpage, la stratégie adoptée est simpliste : le pas de temps courant $\delta t$ est divisé par $2$ et $n$ est multiplié par $2$.

Quand l'intégration réussit pour le pas de temps courant $\delta t$, $n$ est diminué de $1$. Le travail d'intégration est fini quand $n$ vaut $0$.

Le nombre maximum de sous-découpage est limité. Si cette limite est atteinte, le code retourne une execption.

### 3.4.2 Directives gérant le sous-découpage du pas de temps

Le mot clé `@UMATUseTimeSubStepping` demande à ce que l'interface mette en place une stratégie de sous-découpage.

Le mot clé `@UMATMaximumSubStepping` permet de préciser le nombre maximum de sous-découpages autorisés. Cette information doit être explicitement fournie par l'utilisateur.

## 3.5 GÉNÉRATION AUTOMATIQUE D'UN FICHIER `mtest` EN CAS D'ÉCHEC

Le mot clé `@UMATGenerateMTestFileOnFailure` est suivi d'une valeur booléenne. Si cette valeur est vraie, un fichier `mtest` sera généré automatiquement en cas d'échec de l'intégration.

## 3.6 GESTION DES BORNES DE VALIDITÉ

Le traitement d'une violation des bornes de validité expérimentale dépendant de la politique définie par l'utilisateur à l'aide de la variable d'environnement `CASTEM_OUT_OF_BOUND_POLICY`. Trois politiques sont sont possibles :

— `STRICT`, qui traite un dépassement comme une erreur, avec un traitement similaire à ce qui est fait pour les bornes physiques ;
— `WARNING`, qui conduit à afficher un message d'avertissement sans génération d'erreur ;
— `NONE`, qui ignore le dépassement ;

Si la variable `CASTEM_OUT_OF_BOUND_POLICY` n'est pas définie, les dépassements sont ignorés.

## 3.7 MODIFICATION DES PARAMÈTRES

Les valeurs des paramètres peuvent être lus dans un fichier texte. Ce fichier texte doit :

1. se nommer `XX-parameters.txt` où `XX` est le nom de la loi de comportement ;
2. se trouver dans le répertoire courant.

Chaque ligne de ce fichier doit commencer par le nom d'appel du paramètre (c'est à dire le nom de glossaire si il est défini, le nom de la variable associé, etc...), suivi du nom de la valeur à associer.

## 3.8 GÉNÉRATION DE LA MISE EN DONNÉE CAST3M

Dans le cas de l'interface `umat`, `mfront` génère automatiquement un exemple de mise en donnée `Cast3M` pour l'appel de la loi générée. Plusieurs remarques sont nécessaires :

- la loi générée ne peut être appelée directement, il est nécessaire de créer à la main un chapeau `umat` qui appellera la loi générée et de recompiler `Cast3M` pour intégrer ce chapeau.
- la syntaxe varie suivant la dimension du problème. En effet, `Cast3M` ne définit que des variables internes scalaires. Il est donc nécessaire de définir les variables internes tensorielles composante par composante et le nombre de composante varie avec la dimension.

```
*
* \file SiCCreep.dgibi
* \brief example of how to use the SiCCreep behaviour law
* in the Cast3M finite element solver
* \author Helfer Thomas
* \date 06 / 12 / 07
*

** 3D Example

coel3D  =  'MOTS'  'YOUN'  'NU'  'ALPH';
stav3D  =  'MOTS'  'EEXX'  'EEYY'  'EEZZ'  'EEXY'  'EEXZ'  'EEYZ'    'P';
para3D  =  'MOTS'  'T'  'FLUX';

MODL3D  =  'MODELISER'  v3D  'MECANIQUE'  'ELASTIQUE'
'NON_LINEAIRE'  'UTILISATEUR'  'NUME_LOI'  3
'C_MATERIAU'  coel3D
'C_VARINTER'  stav3D
'PARA_LOI'     para3D
'CONS'  M;

MATR3D  =  'MATERIAU'  MODL3D
'YOUN'  xyoun  'NU'  xnu  'ALPH'  xalph;

VAR3D  =  'MANUEL'  'CHML'  MODL3D
'EEXX'  0.  'EEYY'  0.  'EEZZ'  0.  'EEXY'  0.  'EEXZ'  0.  'EEYZ'  0.  'P'  0.
TYPE  'VARIABLES_INTERNES';
```

**FIGURE 2 :** Appel à la loi de comportement générée depuis `Cast3M`.

La figure 2 montre un extrait de l'exemple de code `gibiane` généré par `mfront` pour la loi traitée.

Cette exemple contient un certain nombre de manques et de valeurs arbitraires :

- nous avons définit le modèle mécanique sur un maillage nommé v3D ;
- nous avons attribué la valeur 3 au numéro de la loi (mot clé `NUME_LOI`). Cette valeur est en fait attribuée au moment de la construction du chapeau `umat` ;
- les valeurs des coefficients matériau `xyoun`, `xnu` et `xalph` ne sont pas précisés ;
- la variable externe `FLUX` doit être définie par un chargement externe si l'on utilise la procédure mécanique `PASAPAS`.

La nécessité de passer par la construction d'un chapeau `umat` et de recompiler `Cast3M` rend cette façon de faire pénible et source d'erreur.

```
*
* \file SiCCreep.dgibi
* \brief example of how to use the SiCCreep behaviour law
* in the Cast3M finite element solver
* \author Helfer Thomas
* \date 06 / 12 / 07
*

** 3D Example

tab  =  'TABLE';
tab.'MODELE'      =  'umatSiCCreep';
tab.'LIBRAIRIE'  =  'libExternalUmat.so';

coel3D  =  'MOTS'  'YOUN'  'NU'  'ALPH';
stav3D  =  'MOTS'  'EEXX'  'EEYY'  'EEZZ'  'EEXY'  'EEXZ'  'EEYZ'    'P';
para3D  =  'MOTS'  'T'  'FLUX';

MODL3D  =  'MODELISER'  v3D  'MECANIQUE'  'ELASTIQUE'
'NON_LINEAIRE'  'UTILISATEUR'  'DESC_LOI'  tab
'C_MATERIAU'  coel3D
'C_VARINTER'  stav3D
'PARA_LOI'      para3D
'CONS'  M;

MATR3D  =  'MATERIAU'  MODL3D
'YOUN'  xyoun  'NU'  xnu  'ALPH'  xalph;

VAR3D  =  'MANUEL'  'CHML'  MODL3D
'EEXX'  0.  'EEYY'  0.  'EEZZ'  0.  'EEXY'  0.  'EEXZ'  0.  'EEYZ'  0.  'P'  0.
TYPE  'VARIABLES_INTERNES';
```

**FIGURE 3 :** Appel direct à la loi de comportement générée depuis la version `pleiades` de `Cast3M`.

# RÉFÉRENCES


[Bargellini 13] BARGELLINI R. *Modèles de grandes déformations GDEF_LOG et GDEF_HYPO_ELAS*. Référence du Code Aster R5.03.24 révision : 10464, EDF-R&D/AMA, 2013.

[Forest 13] FOREST SAMUEL, AMESTOY MICHEL, DAMAMME GILLES, KRUCH SERGE, MAUREL VINCENT et MAZIÈRE MATTHIEU. *Mécanique des milieux continus*, 2013.

[Helfer 13] HELFER THOMAS, CASTELIER ÉTIENNE, BLANC VICTOR et JULIEN JÉRÔME. *Le générateur de code mfront : écriture de lois de comportement mécanique*. Note technique 13-020, CEA DEN/DEC/SESC/LSC, 2013.

[Helfer 14] HELFER THOMAS. *Prise en compte des hypothèses de contraintes planes dans les lois générées par le générateur de code MFront : application à Cyrano3*. Note technique 14-013, CEA DEN/DEC/SESC/LSC, 2014.

[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.

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

[Proix 12] 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 13] PROIX JEAN-MICHEL. *Loi de comportement en grandes rotations et petites déformations*. Référence du Code Aster R5.03.22 révision : 11536, EDF-R&D/AMA, Septembre 2013.

[Sidoroff 81] SIDOROFF FRANÇOIS. *Formulations élasto-plastiques en grandes déformations*. Rapport GRECO 29/1981, CNRS, 1981.

[Sidoroff 82] SIDOROFF FRANÇOIS. *Cours sur les grandes transformations*. Rapport GRECO 51/1982, CNRS, Sophia-Antipolis, Septembre 1982.

[Simo 98] SIMO JUAN C et HUGHES THOMAS J. R. *Computational inelasticity*. Springer, New York, 1998.

# LISTE DES TABLEAUX

# LISTE DES FIGURES

FIGURE 1 Appel d'une loi `mfront` dans la version `pleiades` de `Cast3M`. ..... 4
FIGURE 2 Appel à la loi de comportement générée depuis `Cast3M`. ..... 8
FIGURE 3 Appel direct à la loi de comportement générée depuis la version `pleiades` de `Cast3M`. ..... 9

# ANNEXE A DESCRIPTION DE L'INTERFACE

## ANNEXE A.1 CHOIX DU TYPE DE CALCUL

Le premier élément du tableau DDSOE permet de préciser le type de calcul à effectuer et le type de matrice de raideur attendue.

Une valeur strictement négative correspond à un calcul d'une matrice de prédiction. Dans ce cas, il n'y a pas d'intégration de la loi de comportement. Les valeurs supportées sont :

- -1, qui correspond au calcul de la matrice d'élasticité initiale (non endommagée) ;
- -2, qui correspond au calcul de la matrice sécante (matrice d'élasticité endommagée) ;
- -3, qui correspond au calcul de la matrice tangente.

Une valeur positive ou nulle du premier élément du tableau DDSOE conduit à l'intégration de la loi de comportement sur le pas de temps. Si cette intégration s'effectue avec succès, une matrice de raideur peut être calculée. Le type de matrice calculée dépend de la valeur du premier élément du tableau DDSOE :

- 1, correspond au calcul de la matrice d'élasticité initiale (non endommagée) ;
- 2, correspond au calcul de la matrice sécante (matrice d'élasticité endommagée) ;
- 3, correspond au calcul de la matrice tangente ;
- 4, correspond au calcul de la matrice tangente cohérente ;

## ANNEXE A.2 MÉTA-DONNÉES

# ANNEXE B RAPPEL SUR LES FORMULATIONS HYPOÉLASTIQUES DE CAST3M

Quand l'option GRANDS_DEPLACEMENTS est mise à VRAI, Cast3M propose différentes formulations hypoélastiques pour pouvoir utiliser des lois de comportement écrites dans l'hypothèse des petites déformations alors que les déformations ou les rotations peuvent devenir importantes.

Une formulation hypoélastique est tout d'abord une loi écrite « en vitesses » et se construit grossièrement ainsi :

- on calcule un incrément de déformation $\Delta \underline{\epsilon}^{to}$ sur la configuration de début de pas (ou éventuellement sur une configuration intermédaire, la configuration à mi-pas étant particulièrement intéressante d'un point de vue numérique [Simo 98]). Le calcul de l'incrément dépend de la formulation choisie.
- connaissant la contrainte en début de pas, on appelle la loi de comportement pour en déduire la contrainte en fin de pas $\underline{\sigma}|_{t+\Delta t}$ [2] sur la configuration utilisée pour calculer $\Delta \underline{\epsilon}^{to}$.
- $\underline{\sigma}|_{t+\Delta t}$ est porté sur la configuration de fin de pas suivant une méthode qui dépend de la formulation choisie. C'est le rôle de l'opérateur 'PICA'.

## ANNEXE B.1 LIMITES INTRINSÈQUES DES FORMULATIONS HYPOÉLASTIQUES

À partir du schéma général décrit au paragraphe précédent, nous pouvons faire quelques remarques importantes :

- *les formulations hypoélastiques, travaillant sur des incréments conduisent à un travail non nul lors d'un cycle fermé dans l'espace des déformations, même en l'absence de phénomènes dissipatifs* : elles sont donc particulièrement insatisfaisantes d'un point de vue théorique, même si en pratique ce travail est très faible [Forest 13].

2. On peut noter que calculer l'incrément de la contrainte serait plus cohérent avec l'hypothèse de formulation en vitesses.

— la déformation totale n'a pas de sens : seul son incrément $\Delta \underline{\epsilon}^{to}$ est défini. Sommer ces incréments est incohérent puisqu'ils ont tous été calculés sur des configurations différentes.
— de la même manière que la déformation totale n'a pas de sens, il est impossible que la loi utilise des variables internes tensorielles. En effet, il ne serait possible que de calculer leurs incréments, mais impossible de sommer ces incréments aux valeurs de début de pas.
— la contrainte apparaît comme une variable interne. Ce point est problématique, car les formulations hypoélastiques ne peuvent décrire des matériaux dont les propriétés élastiques dépendent du temps. Plus précisément, si l'on considère la loi élastique anisotherme, sa formulation en vitesses serait :

$$\dot{\underline{\sigma}} = \underline{\underline{\mathbf{D}}} : \dot{\underline{\epsilon}}^{el} + \dot{\underline{\underline{\mathbf{D}}}} : \underline{\epsilon}^{el}$$

Pour les raisons évoquées plus haut, $\underline{\epsilon}^{el}$ ne peut être défini. Le choix fait dans `Cast3M` et dans la plupart des lois de comportement natives semble être de négliger le second terme de cet équation, ce qui est une approximation forte.

Une autre limite importante, apportée par des considérations théoriques sur l'objectivité des lois, est que *les formulations hypoélastiques ne peuvent décrire que des matériaux isotropes* [Sidoroff 81, Sidoroff 82, Simo 98].

`Cast3M` n'impose malheureusement pas le respect de ses limites, laissant l'utilisateur responsable de la bonne utilisation de sa loi. Pire, pour les lois `umat`, les entrées suivantes sont improprement renseignées :
— la déformation totale (qui n'a aucun sens physique) ;
— les axes d'orthotropie (seules les lois isotropes devraient pouvoir être utilisées). Cette entrée est d'autant plus fausse qu'elle reste relative à la configuration initiale (ce qu'il est d'ailleurs la bonne chose à faire si la loi est réellement écrite en grandes transformations).

À cela s'ajoute le fait que l'utilisateur peut définir des variables internes sans préciser leurs natures scalaires ou tensorielles : du point de vue de `Cast3M`, toutes les variables internes de la loi sont scalaires. L'utilisateur peut donc facilement « oublier » la restriction citée plus haut.

## ANNEXE B.2 CHOIX DE LA FORMULATION HYPOÉLASTIQUE

Le choix de la formulation hypoélastique à utiliser se fait via le mot clé `EPSILON` lors de la définition du modèle mécanique. Pour des raisons historiques, *ce mot clé est aujourd'hui optionnel* : si il n'est pas précisé, `Cast3M` utilisera une formulation par défaut que l'utilisateur peut modifier par un appel à la commande `'OPTION' 'EPSILON'`.

Le fait que ce mot clé soit optionnel est potentiellement une source d'erreur importante, en particulier si la loi est effectivement écrite en grandes transformations. Il semble qu'une proposition visant à le rendre obligatoire ait été rejetée en réunion de développement `Cast3M`. Nous pensons que cela est regrettable.

Les formulations actuellement disponibles sont : `LINEAIRE`, `QUADRATIQUE`, `JAUMAN`, `TRUESDELL`[3].

3. L'option 'UTILISATEUR' sert à préciser que les lois de comportement sont écrites en grandes transformations et n'est pas traitée ici. Elle le sera au paragraphe 2.2.