L'interface `zmat` aux lois de comportements mécaniques de `mfront`

T. Helfer

2014

RÉSUMÉ

Le Centre des Matériaux et les Mines de Paris sont associés à de nombreuses thèses avec le CEA et EDF.

Il est apparu d'un grand intérêt de permettre l'échange des lois de comportement mécanique entre le CEA, EDF et le Centre des Matériaux, chacun de ses organismes ayant développé son code aux éléments finis : `Cast3M` pour le CEA, `Aster` pour EDF et `ZeBuLoN` pour le Centre des Matériaux.

Une interface `mfront`, nommée `zmat`, pour le code `ZeBuLoN`, a donc été développée pour les lois de comportement en petites déformations et en grandes transformations. Cette interface s'appuie sur les classes du code `ZeBuLoN` pour une intégration aussi naturelle que possible : une loi générée par `mfront` ne doit pas se distinguer d'une loi native.

Nous présentons les détails de cet interface et comment l'utiliser dans le code `ZeBuLoN`, ainsi que des tests d'intégration.

Une même loi de comportement `mfront` peut aujourd'hui être partagée entre les utilisateurs des codes `Cast3M` (CEA), `Aster` (EDF) et `ZeBuLoN` (Centre des Matériaux).

SOMMAIRE

1 INTRODUCTION

Nous présentons dans cette note l'interface `zmat` qui permet d'utiliser les lois mécaniques `mfront` dans le code aux éléments finis `ZeBuLoN`.

La première partie décrit les spécificités du code aux éléments finis `ZeBuLoN` par rapport :

- aux conventions utilisées par `mfront` (rangement des tenseurs) ;
- aux fonctionnalités disponibles dans `mfront` et utilisées dans d'autres codes aux éléments finis (gestion de la contrainte plane, orthotropie, opérateur tangent) ;

La seconde partie décrit la mise en donnée d'une loi `mfront`. Nous y décrivons :

- l'utilisation des noms de glossaire ;
- la gestion des propriétés matériaux, des variables internes et des variables externes ;
- la déclaration des paramètres ;

La troisième partie décrit deux tests unitaires. La dernière partie décrit les évolutions possibles de l'interface.

2 SPÉCIFICITÉS DU CODE AUX ÉLÉMENTS FINIS ZEBULON

Nous décrivons ici quelques spécificités du codes aux éléments finis `ZeBuLoN`.

2.1 CONVENTIONS DE RANGEMENT DES TENSEURS

Les conventions de rangement des tenseurs adoptés dans `mfront` et `zmat` diffèrent. Les variables internes sont converties en entrée vers la convention utilisée dans `tfel` et en sortie vers la convention utilisée par `ZeBuLoN`.

2.1.1 Tenseurs d'ordre 2 symétriques

Dans `zmat`, un tenseur symétrique $\underline{\mathbf{a}}$ est stocké ainsi :

$$\left(a_{xx} \quad a_{yy} \quad a_{zz} \quad \sqrt{2}\,a_{xy} \quad \sqrt{2}\,a_{yz} \quad \sqrt{2}\,a_{xz}\right)$$

Dans `tfel`, ce même tenseur est représenté ainsi :

$$\left(a_{xx} \quad a_{yy} \quad a_{zz} \quad \sqrt{2}\,a_{xy} \quad \sqrt{2}\,a_{xz} \quad \sqrt{2}\,a_{yz}\right)$$

On passe de l'une à l'autre des conventions en intervertissant les deux dernières composantes.

2.1.2 Tenseurs d'ordre 2 (non symétriques)

Dans `zmat`, un tenseur $\underset{\sim}{\mathbf{a}}$ est stocké ainsi :

$$\left(a_{xx} \quad a_{yy} \quad a_{zz} \quad a_{xy} \quad a_{yz} \quad a_{zx} \quad a_{yx} \quad a_{zy} \quad a_{xz}\right)$$

Dans `tfel`, ce même tenseur est représenté ainsi :

$$\left(a_{xx} \quad a_{yy} \quad a_{zz} \quad a_{xy} \quad a_{yx} \quad a_{xz} \quad a_{zx} \quad a_{yz} \quad a_{zy}\right)$$

2.2 ORTHOTROPIE

L'orthotropie éventuelle du matériau est géré par `ZeBuLoN` en amont et en aval de l'appel à la loi de comportement.

2.3 HYPOTHÈSES DE MODÉLISATION

L'idée que la loi de comportement doit pouvoir être écrite de manière indépendante de l'hypothèse de modélisation est très fortement ancrée dans le code aux éléments finis `ZeBuLoN`.

En particulier l'hypothèse de contraintes planes est gérée aux niveaux des éléments finis et non par la loi de comportement [Besson 97].

En pratique, la loi de comportement n'a accès qu'à la dimension d'espace. Nous avons du faire le choix d'associer à chaque dimension d'espace une hypothèse de modélisation, qui est potentiellement différente de l'hypothèse utilisée pour le calcul de structure :

- en $1D$, l'hypothèse d'axisymétrie et de déformations planes généralisées est utilisée ;
- en $2D$, l'hypothèse des déformations planes généralisées est utilisée.

Ces choix peuvent parfois être problématiques. En particulier, dans les lois de comportement orthotropes, nous retrouvons une difficulté que nous avons déjà soulevées (voir [Helfer 13]) : la définition de la matrice de HILL peut être incohérente avec la définition des axes d'orthtropie et/ou la définition du tenseur d'élasticité.

2.4 TRANSFORMATIONS FINIES

Le code aux éléments finis `ZeBuLoN` propose trois formulations en grandes transformations :

- `Updated_Lagrangian`, qui associe la contrainte de CAUCHY $\underline{\sigma}$ au taux de déformation $\underline{\mathbf{D}}$.
- `Lagrangian_PK1` qui associe au premier tenseur de PIOLA-KIRCHHOFF le gradient de la transformation ;
- `Total_Lagrangian` qui associe au second tenseur de PIOLA-KIRCHHOFF le tenseur de GREEN-LAGRANGE.

Les lois grandes transformations générées par `mfront` ne sont aujourd'hui compatibles qu'avec la formulation `Updated_Lagrangian`.

2.5 PRÉDICTION ET MATRICE TANGENTE COHÉRENTE

2.5.1 Matrice de prédiction

La notion de matrice de prédiction n'est pas utilisée dans le code aux éléments finis `ZeBuLoN` actuellement.

2.5.2 Matrice tangente cohérente

Le code aux éléments finis `ZeBuLoN` utilise, pour la convergence globale de l'équilibre, tout opérateur fourni par la loi de comportement. À chaque intégration, le code précise si cette matrice doit être calculée. Par rapport au code aux éléments finis `Aster`, `ZeBuLoN` ne précise pas aujourd'hui si l'opérateur attendu est la matrice d'élasticité initiale, la matrice d'élasticité endommagée (opérateur sécant), l'opérateur tangent ou l'opérateur tangent cohérent.

Nous avons fait le choix d'utiliser par défaut la matrice tangente cohérente : ainsi, le paramètre `smt` de la directive `mfront @TangentOperator` est toujours égal à `CONSISTENTTANGENTOPÉRATEUR`.

À terme, il serait judicieux de pouvoir modifier ce choix du type d'opérateur depuis le jeu de données.

2.5.3 Cas des transformations finies

Pour les lois écrites en transformations finies, nous avons désigné par `DSIG_DD` la matrice tangente attendue par `ZeBuLoN`.

Si l'utilisateur fournit une autre matrice tangente, `mfront` essaiera de la convertir.

Par exemple, si l'utilisateur fournit pour matrice tangente la dérivée de la contrainte de KIRCHHOFF $\underline{\tau}$ par rapport à l'incrément spatial du gradient de la transformation $\Delta \underset{\sim}{\mathbf{F}} = \underset{\sim}{\mathbf{F}}_{t+\Delta t} \underset{\sim}{\mathbf{F}}_{t}^{-1}$ [1], les transformations suivantes seront réalisées :

$$\frac{\partial \underline{\tau}}{\partial \Delta \underset{\sim}{\mathbf{F}}} \quad \Rightarrow \quad \frac{\partial \underline{\tau}}{\partial \underset{\sim}{\mathbf{F}}_{t+\Delta t}} \quad \Rightarrow \quad \texttt{DSIG\_DD}$$

Par exemple, si l'utilisateur fournit pour matrice tangente la dérivée de la second contrainte de PIOLA-KIRCHHOFF $\underline{\mathbf{S}}$ par rapport au tenseur de GREEN-LAGRANGE les transformations suivantes seront réalisées :

$$\frac{\partial \underline{\mathbf{S}}}{\partial \underline{\varepsilon}^{GL}} \quad \Rightarrow \quad \frac{\partial \underline{\mathbf{S}}}{\partial \underline{\mathbf{C}}} \quad \Rightarrow \quad \frac{\partial \underline{\tau}}{\partial \underset{\sim}{\mathbf{F}}_{t+\Delta t}} \quad \Rightarrow \quad \texttt{DSIG\_DD}$$

3 DÉFINITION D'UNE LOI DE COMPORTEMENT DANS LE JEU DE DONNÉES ZEBULON

```
1  ***behavior Norton
2   **material_properties
3     YoungModulus 150e9
4     PoissonRatio   0.33
5     A[0] 8.e-67
6     A[1] 1.
7     E 8.2
8   **out_of_bounds_policy Strict
9   **parameters
10    epsilon 1.e-12
```

FIGURE 1 : Mise en données d'une loi de NORTON.

La figure 1 illustre la plupart des points évoqués dans cette section.

3.1 NOMS DE GLOSSAIRE

Les noms de glossaire sont utilisés pour la définition des différents éléments de la loi dans le jeu de données `ZeBuLoN` (fichier d'extension `inp`). La température (dont le nom de glossaire est `Temperature`) constitue une exception : celle-ci doit être désignée par `temperature` par compatibilité avec les conventions `ZeBuLoN`.

3.2 PROPRIÉTÉS MATÉRIAU

Les propriétés matériau sont définies dans les fichiers `mfront` par le mot clé `@MaterialProperty`).

1. Il s'agit de la matrice tangente attendue par le `Code-Aster`.

Elles sont renseignées dans la définition du matériau sous l'une des entrées `**model_coef` (syntaxe standard `ZeBuLoN`) ou `**material_properties` (compatibilité avec `mfront`).

Les propriétés matériaux sont alors définies comme pour les lois natives de `ZeBuLoN` : nous nous sommes basées pour cela sur la classe `ZSET::COEFF`.

Les propriétés matériau sont évaluées en fin de pas de temps par le code aux éléments finis `ZeBuLoN` avant l'intégration puis transmises à `mfront`.

3.2.1 Cas des tableaux de propriétés matériau

Nous n'avons pas trouver de solution native pour gérer les tableaux de propriétés matériau. Nous avons donc adopter la convention suivante : si l'utilisateur défini dans le fichier `mfront` un tableau de 10 propriétés matériau de nom de glossaire `MP`, il devra définir dans le jeu de données `ZeBuLoN` les 10 propriétés :

$$\texttt{MP[0]}, \texttt{MP[1]}, \ldots, \texttt{MP[9]}$$

Cette convention est incohérente avec celle utilisée pour les tableaux de variable internes.

3.2.2 Limitations

L'utilisation des propriétés matériau souffre dans la version actuelle des limitations suivantes :

- il ne faut pas définir de propriétés dépendant des variables internes ;
- il n'y a pas d'évolution au cours du pas de temps des propriétés.

3.3 MATRICE D'ÉLASTICITÉ

Sans le cas où l'utilisateur précise, via le mot clé `@RequireStiffnessTensor`, que le code `ZeBuLoN` doit fournir la matrice d'élasticité à la loi de comportement, celle-ci doit être définie dans le jeu de données sous le mot clé `**elasticity`, comme dans le cas des lois natives.

3.4 VARIABLES INTERNES

Les variables internes sont gérées de la même manière que dans le cas d'une loi native.

3.5 VARIABLES EXTERNES

Chacune des variables externes définies par le mot clé `@ExternalStateVariable` dans le fichier `mfront` doit être définie dans le jeu de données `ZeBuLoN` sous la section `**paramater`.

Cas de la température La température est désignée dans le jeu de données `ZeBuLoN` sous le nom `temperature` par compatibilité avec les conventions `ZeBuLoN`. Sa définition est aujourd'hui obligatoire.

3.6 PARAMÈTRES DE LA LOI

Il est possible de modifier les paramètres d'une loi de comportement sous le mot clé `**parameters`. Les paramètres sont définies par des nombres réels.

Note La modification d'un paramètre affecte toutes les instances d'une même loi.

3.7 Gestion du dépassement d'une borne

La manière de gérer le dépassement d'une borne de validité de la loi peut-être paramétrée via l'entrée `**out_of_bounds_policy`. Trois valeurs sont admises :

- `Strict` qui conduit à une erreur dans l'intégration de la loi ;
- `Warning` qui affiche une erreur sur la sortie standard ;
- `None` qui ne fait rien (option par défaut).

4 Tests unitaires

Nous avons mené deux tests d'intégration :

— l'une en petites déformations en utilisant une loi de Norton ;
— une en grandes transformations en utilisant une loi de Saint-Venant-Kirchhoff orthotrope ;

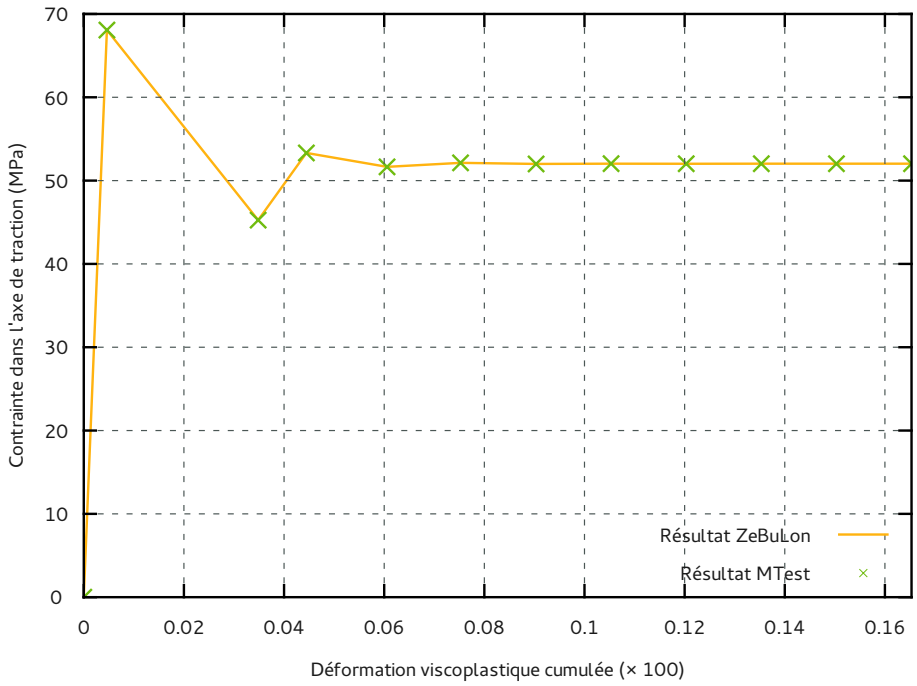

Figure 2 : Comparaison `ZeBuLoN/mtest` sur un essai de traction uniaxiale d'un matériau obéissant à la loi de Norton.

4.1 La loi de Norton

Le premier test consiste en une traction uniaxiale d'un matériau obéissant à la loi de Norton. Ce test a été mené en contraintes planes.

Ce test a permis de vérifier que pratiquement l'ensemble des points évoqués plus haut, et notamment la convergence quadratique de l'algorithme d'équilibre global.

Nous avons utilisé une intégration semi-implicite pour faire apparaître des oscillations dans la réponse globale. Le résultat obtenu avec `ZeBuLoN` est comparé au résultat obtenu avec `mtest` en figure 2.

4.2 La loi de Saint-Venant-Kirchhoff orthotrope

```
1  @Parser DefaultFiniteStrainParser;
2  @Behaviour OrthotropicSaintVenantKirchhoffElasticity;
3
4  @OrthotropicBehaviour;
5  @RequireStiffnessTensor;
6
7  @Integrator{
8    const StrainStensor e = computeGreenLagrangeTensor(F1);
9    // second Piola-Kirchhoff stress
10   const StressStensor s = D*e;
11   // conversion to Cauchy stress tensor
12   sig = convertSecondPiolaKirchhoffStressToCauchyStress(s,F1);
13 }
14
15 @TangentOperator<DS_DEGL>{
16   Dt = D;
17 }
```

Figure 3 : Implantation d'une loi de Saint-Venant-Kirchhoff orthotrope.

4.2.1 Implantation en `mfront`

Nous considérons ici une loi d'élasticité simple : la loi de Saint-Venant-Kirchhoff étendue au cas orthotrope. L'implantation de cette loi est donnée en figure 3. Nous pouvons faire plusieurs remarques :

- le mot-clé `@OrthotropicBehaviour` est inutile pour une adhérence au code `ZeBuLoN`. Il a été retenu par compatibilité avec les codes `Cast3M` et `Aster` ;
- nous déléguons au code `ZeBuLoN` l'évaluation de la matrice d'élasticité ;
- la contrainte de sortie est la contrainte de Cauchy ;
- nous donnons l'opérateur tangent comme la dérivée du second tenseur de Piola-Kirchhoff (en fin de pas) par rapport à la déformation de Green-Lagrange (en fin de pas).

4.2.2 Mise en données

Nous considérons un essai de traction à 70% de déformation sur un cube de longueur unité. La mise en données du problème est reportée en figure 4. Nous pouvons noter l'utilisation du mot clé `***elasticity` pour définir la matrice d'élasticité : nous nous appuyons sur les classes standard de `ZeBuLoN` pour l'évaluation de celle-ci.

4.2.3 Résultats

La figure 5 compare les résultats obtenus à la solution analytique. Nous pouvons noter une bonne convergence de la résolution globale en une itération par pas de temps, ce qui est à nuancer par le grand nombre de pas de temps.

```
1  ****calcul
2  ***mesh  updated_lagrangian
3   **file testcube1.geof
4  ***resolution
5    **sequence 1
6      *time  120.
7      *increment 100
8      *iteration 10
9      *ratio 0.01
10     *algorithm p1p2p3
11   **init_d_dof
12   **automatic_time global 3 epcum 0.01
13     *security 1.2
14     *divergence 2. 8
15     *first_dtime 0.0001
16 ***bc
17   **impose_nodal_dof
18       haut U3 1. tab
19       base U3 0.
20       gauche U1 0.
21       avant U2 0.
22 ***equation
23       **mpc1  droit U1
24       **mpc1 arriere U2
25 ***table
26   **name tab
27     *time  0.0  120.0
28     *value 0.0  0.7
29   **name temp_tab
30     *time  0.0  120.0
31     *value 340. 340.
32 ***output
33  **verbose
34  **save_parameter
35  **curve
36   *gauss_var 1 1 F13 F11 F22 F33 F12 F21 F23 F32 F31 sig31 sig11 sig22 sig33
      sig12 sig23
37 ***parameter
38  **table temperature
39       1.0 temp_tab
40 ***material
41     *file test1.inp
42     *rotation
43       x3 0. 0. 1.
44       x1 1. 0. 0.
45 ****return
46
47 ***behavior OrthotropicSaintVenantKirchhoffElasticity
48  **elasticity cubic
49        y1111 198600.0
50        y1122 136200.0
51        y1212 104700.0
52 ***return
```

FIGURE 4 : Mise en données d'un essai de traction avec la loi de SAINT-VENANT-KIRCHHOFF.

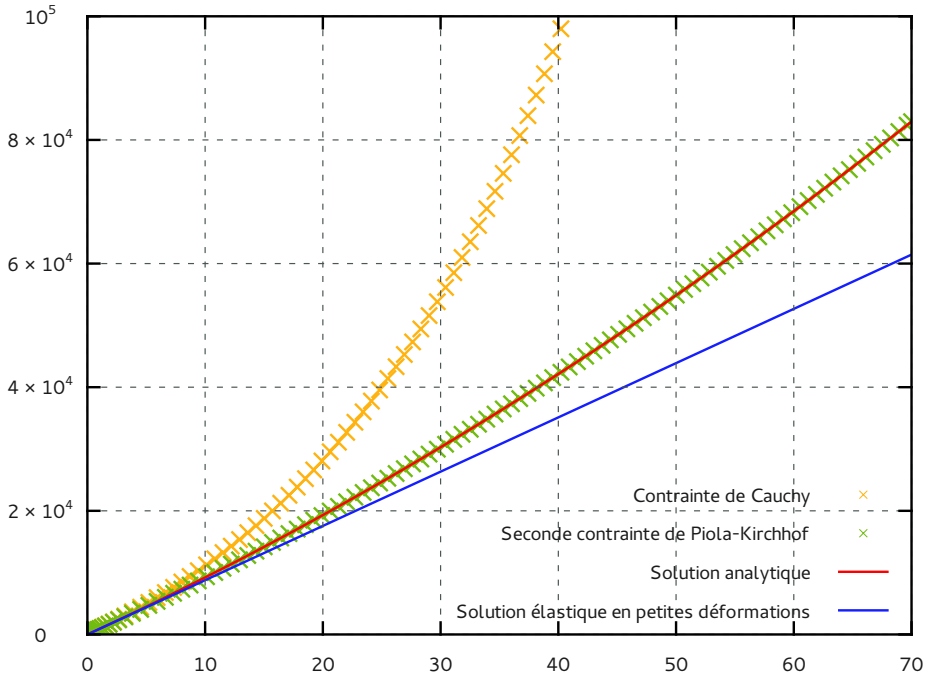

FIGURE 5 : Comparaison des résultats à la solution analytique.

5 AMÉLIORATIONS POSSIBLES

Différentes améliorations sont possibles :

- il serait intéressant d'avoir une interface `zmat` pour les propriétés matériaux ;
- la variation sur le pas de temps des propriétés matériau n'est pas prise en compte : celles-ci sont toujours évaluées en fin de pas ;
- à notre connaissance, il n'est pas possible dans `zmat` de définir des tableaux de propriétés matériau ou des tableaux de variables externes alors des tableaux de variables internes sont supportés. Nous avons contourné ceci en déclarant autant de propriétés matériau ou de variables externes que nécessaire. Par exemple, si la loi de comportement nécessite un tableau `A` de 10 propriétés matériau, l'utilisateur devra déclarer dans le jeu de données les propriétés matériau :

  `A[0]`, `A[1]`, etc...

  Cette convention n'est pas homogène avec celle utilisée par `ZeBuLoN` pour désigner un élément d'un tableau de variables internes dans le jeu de données.
- le code aux éléments finis `ZeBuLoN` propose aujourd'hui trois formulations en grandes transformations : `Updated_Lagrangian`, `Lagrangian_PK1`, `Total_Lagrangian`. Seule la formulation `Updated_Lagrangian` est aujourd'hui supportée dans `mfront`. Le support des deux autres formulations est triviale. Ainsi, en utilisant les convertions automatiques entre les différents opérateurs tangents possibles proposés par `mfront`, une même loi pourrait être utilisée quelque soit la formulation utilisée.
- Pour l'instant, l'opérateur tangente demandé à la loi est toujours la matrice tangente cohérente. À terme, deux évolutions apparaissent intéressantes :
  - il serait judicieux de pouvoir modifier ce choix du type d'opérateur depuis le jeu de données. ;
  - il serait intéressant d'introduire, dans `mfront`, un nouveau type d'opérateur tangent. En effet, dans les cas adoucissant, une technique possible consiste à modifier de manière intelligente la matrice tangente

cohérente (en « oubliant » certains termes par exemple). Cette technique est par exemple utilisé dans l'implantation `mfront` de la loi de MAZARS.

- pour l'instant, les affichages dans les lois de comportements utilisent les flux de sortie standard du C++ qui ne permettent qu'un affichage à l'écran. Pour une meilleure intégration dans `ZeBuLoN`, il serait intéressant que l'on puisse substituer à ces flux de sortie ceux prévus par `ZeBuLoN` qui permettent un plus de l'affichage écran une trace dans un fichier associé au calcul.

6 CONCLUSIONS

Une interface `zmat` pour le code aux éléments finis `ZeBuLoN` est aujourd'hui disponible : une même loi de comportement `mfront` peut aujourd'hui être partagée entre les utilisateurs des codes `Cast3M` (CEA), `Aster` (EDF) et `ZeBuLoN` (Centre des Matériaux).

Pour conclure, soulignons que le travail nécessaire à la réalisation de cette interface a été très réduit (l'interface en elle-même est implantée dans un fichier d'environ 1300 lignes), essentiellement grâce aux facilités offertes par le code `ZeBuLoN` : gestion de l'orthotropie en amont et en aval de la loi de comportement, gestion des contraintes planes, etc...

RÉFÉRENCES

[Besson 97] BESSON J. et FOERCH R. *Large scale object-oriented finite element code design*. Computer Methods in Applied Mechanics and Engineering, Mars 1997, vol 142, n° 1–2, p 165–187.

[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.

LISTE DES FIGURES