# Prise en compte des hypothèses de contraintes planes dans les lois générées par le générateur de code `mfront` : application à `Cyrano3`

**T. Helfer**

**Septembre 2014**

## RÉSUMÉ

La prise en compte des hypothèses de contraintes planes (généralisées ou non) a pour but de permettre une utilisation efficace des lois générées par `mfront` dans l'application combustible industrielle `Cyrano3`, développée par ÉDF au sein du projet `pleiades`.

En effet, pour réduire la description d'une tranche de crayon combustible à un problème $1D$, `Cyrano3` fait deux hypothèses de modélisation. La première est l'hypothèse d'axisymétrie.

La seconde concerne le traitement de la quasi-invariance du problème le long de l'axe du crayon. Deux hypothèses de modélisation sont disponibles dans `Cyrano3` :

- la contrainte axiale est homogène dans le combustible d'une part et dans la gaine d'autre part. Cette hypothèse de « contraintes planes généralisées » est spécifique à `Cyrano3`. Il s'agit de l'hypothèse utilisée par défaut.
- la déformation est homogène dans le combustible d'une part et dans la gaine d'autre part. C'est l'hypothèse classique de déformations planes généralisées utilisée dans `alcyone` ($1D$, $2D\,(r,\theta)$ et `germinal`.

De multiples solutions sont possibles en `mfront` pour implanter une loi de comportement utilisable en contraintes planes. La plupart de ces solutions conduisent à une implantation spécifique : les implantations qui en découlent ne peuvent être réutilisées pour les hypothèses standard (axisymétrie en déformations planes généralisées, déformations planes, déformations planes généralisées, axisymétrie, $3D$) utilisées par les autres applications combustibles de la plate-forme.

Cette note décrit une méthode, applicable aux implantations basées sur une intégration implicite et pour des lois écrites en petites déformations, qui consiste à :

- écrire les équations constitutives dans un bloc, indépendamment de l'hypothèse de modélisation ;
- prendre en compte l'hypothèse de contraintes planes par une équation additionnelle, décrite dans un bloc spécifique. Cette équation ne dépend que de la forme de la loi d'élasticité. Cette équation permet de calculer l'évolution d'un degré de liberté supplémentaire, la déformation axiale.

Cette méthode présente de nombreux avantages :

- l'implantation obtenue reste utilisable avec les hypothèses de modélisation classique.
- l'implantation obtenue est compatible avec les stratégies « grandes transformations » disponibles dans `mfront` (déformations logarithmiques notamment) qui nécessitent de connaître la déformation axiale ;
- le calcul automatique de la matrice tangente cohérente à partir de la jacobienne du système implicite reste valide sans aucune modification.

Cette méthode permet donc d'écrire une implantation d'une loi utilisable à la fois dans le code `Cyrano3` et dans les applications combustible CEA.

Son principal désavantage est d'introduire une variable interne supplémentaire (la déformation axiale) ce qui induit un coût mémoire et un coût CPU lié d'une part à la gestion par le code appelant de cette variable supplémentaire et d'autre part au fait que la résolution implicite doit prendre en compte une équation supplémentaire. Nous montrerons cependant que les implantations obtenues ont des performances numériques meilleures en moyenne que celles obtenues avec les lois internes de `Cyrano3` (optimisées pour l'hypothèse de contraintes planes généralisées).

# SOMMAIRE

# 1 INTRODUCTION

La prise en compte des hypothèses de contraintes planes (généralisées ou non) a pour but de permettre une utilisation efficace des lois générées par `mfront` dans l'application combustible industrielle `Cyrano3`, développée par ÉDF au sein du projet `pleiades`.

L'idée qui sous-tend ce travail est de permettre l'échange de fichiers `mfront` entre les applications CEA et `Cyrano3`.

## 1.1 RÔLE DE LA LOI DE COMPORTEMENT

Connaissant l'état du matériau à un instant $t$, caractérisé par les valeurs d'un certain nombre de variables internes $Z_i|_t$, et l'incrément de déformation totale $\Delta\underline{\epsilon}^{to}$ entre cet instant et l'instant $t+\Delta t$, le principal rôle des lois de comportement mécanique est de permettre le calcul de la contrainte $\underline{\sigma}|_{t+\Delta t}$ et des variables internes $Z_i|_{t+\Delta t}$ :

$$\left(Z_i|_t\,,\,\underline{\epsilon}^{to}\big|_t\,,\,\sigma|_t\right) \rightarrow \left(Z_i|_{t+\Delta t}\,,\,\sigma|_{t+\Delta t}\right)$$

## 1.2 HYPOTHÈSES DE CONTRAINTES PLANES

L'hypothèse de contraintes planes, utilisée classiquement pour décrire des structures minces, se traduit par la condition :

$$\sigma_z|_{t+\Delta t} = 0$$

où $\sigma_z$ est la contrainte axiale.

L'hypothèse de contraintes planes généralisées, utilisée par le code combustible `Cyrano3`, s'écrit :

$$\sigma_z|_{t+\Delta t} = \sigma_z^{\text{équilibre}}\left(t+\Delta t\right)$$

où $\sigma_z^{\text{équilibre}}\left(t+\Delta t\right)$ est une valeur imposée qui résulte de l'équilibre axial du crayon.

Dans les deux cas, la résolution de l'équilibre mécanique ne permet plus de déterminer la déformation axiale totale $\epsilon_z^{to}$ qui devient un degré de liberté supplémentaire en chaque point de GAUSS.

Pour simplifier la présentation, nous ne traiterons généralement que l'hypothèse de contraintes planes, le traitement des contraintes planes généralisées ne présentant pas de difficulté supplémentaire.

## 1.3 QUELQUES TRAITEMENTS DE LA CONDITION DE CONTRAINTES PLANES

Nous présentons ici quelques traitements classiques de l'hypothèse de contraintes planes.

### 1.3.1 Prise en compte explicite

Une façon classique de traiter l'hypothèse de contraintes planes est de la traiter explicitement dans l'implantation de la loi de comportement. Par exemple, pour une modélisation plane $x, y$, un comportement élastique peut être représenté par une loi effective :

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sqrt{2}\sigma_{xy} \end{pmatrix} = \frac{E}{\left(1-\nu^2\right)} \begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & 1-\nu \end{pmatrix} \begin{pmatrix} \epsilon_{xx}^{to} \\ \epsilon_{yy}^{to} \\ \sqrt{2}\epsilon_{xy}^{to} \end{pmatrix}$$

Nous pouvons noter que :

- la composante axiale des déformations $\epsilon_z^{to}$ n'apparaît pas : celle-ci peut être calculée `a posteriori` en utilisant la loi d'élasticité standard ;
- le tenseur d'élasticité effectif sera désigné par le qualificatif altéré dans cette note (et dans `mfront`).

La prise en compte explicite de l'hypothèse de contraintes planes est sans doute la solution la plus efficace numériquement. Elle se fait en généralisant le traitement de l'élasticité traitée précédemment.

Un des désavantages de cette méthode est qu'elle utilise un formalisme tensoriel particulier dans lequel la composante axiale est omise. Cette représentation tensorielle réduite n'est aujourd'hui pas supportée dans `tfel` (la librairie mathématique sur laquelle s'appuie `mfront`).

L'autre désavantage de cette méthode est qu'elle conduit à proposer plusieurs implantations d'une même loi de comportement : l'une gérant l'hypothèse de contraintes planes, l'autre gérant les autres hypothèses de modélisation (axisymétrie en déformations planes généralisées, déformations planes, déformations planes généralisées, axisymétrie, $3D$). Le temps de développement est ainsi considérablement accru. Du point de vue de l'assurance qualité, la duplication d'une même information physique dans différentes implantations nous semble peu souhaitable.

**Cas des lois de comportement isotropes** Les lois de comportement isotropes, pour lesquelles `mfront` propose des analyseurs spécifiques [Helfer 13b], forment une exception notable pour deux raisons :

— des algorithmes efficaces pour traiter ce type de lois ont été proposés dans la littérature [Simo 98, Doghri 00]. Ces algorithmes n'utilisent pas de formalisme tensoriel particulier car l'intégration de la loi de comportement peut être réduite à une équation scalaire [Simo 98] ou un système de deux équations [Doghri 00] (le formalisme tensoriel réduit est un intermédiaire de calcul permettant d'obtenir cette équation scalaire) ;

— ces analyseurs réduisent la loi de comportement aux écoulements plastiques et viscoplastiques qui les définissent. `mfront` masque les détails liés à l'algorithme d'intégration et il est donc possible de prendre en charge l'hypothèse de contraintes planes sans que l'utilisateur n'ait à dupliquer l'information physique.

Ces lois ne seront pas traitées dans cette note et l'implantation de ces algorithmes reste pour l'instant une perspective intéressante.

### 1.3.2 Traitement à l'appel de la loi de comportement par une boucle externe

Dans le cas de l'interface `umat` et lorsque la loi de comportement ne gère pas l'hypothèse de contraintes planes mais est utilisable en déformations planes généralisées, `mfront` propose un algorithme générique de traitement de l'hypothèse de contraintes planes [Helfer 13a].

Pour cela, une variable, la déformation axiale $\epsilon_z^{to}$ est déclarée et l'algorithme cherche à déterminer sa valeur en fin de pas de temps.

Cet algorithme se décompose ainsi :

— 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 corriger le fait que la condition de contraintes planes n'est pas vérifiée de manière exacte. La loi de comportement en déformations planes généralisées est alors appelée.

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

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

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

— si la contrainte axiale obtenue n'est toujours pas nulle (à une précision donnée près), la méthode itérative de la sécante est utilisée pour estimer un incrément convenable. À chaque itération, la loi de comportement en déformations planes généralisées est appelée.

Cette méthode a un coût numérique important : pour les lois les plus simples, il faut en général $4$ à $5$ itérations pour que la méthode de la sécante converge, ce qui multiplie d'autant le coût d'intégration de la loi (par rapport à l'intégration de la loi en déformations planes). Ce surcoût est rédhibitoire pour un code industriel tel que `Cyrano3`.

### 1.3.3 Traitement par le code éléments finis appelant

Il est possible de traiter la condition de contraintes planes au niveau du code appelant, ce qui évite de développer une implantation spécifique. Deux approches sont particulièrement intéressantes :

— dans le code `ZeBuLoN`, chaque élément fini $\Omega^e$ possède des degrés de liberté supplémentaire correspondant aux déformations axiales en chacun des $N_g$ points de GAUSS [Besson 97]. Les forces internes nodales s'écrivent alors :

$$\vec{R} = \int_{\Omega^e} \begin{bmatrix} {}^tB\,\sigma \\ {}^tI\,\sigma \end{bmatrix} \mathrm{d}\,V$$

où $I$ est une matrice possédant $4$ lignes et $N_g$ colonnes. Le terme ${}^tI\,\sigma$ est, en chaque point de GAUSS, proportionnel à $\sigma_z$, de sorte qu'en l'absence de force imposée associée aux déformations axiales, l'équilibre de la structure satisfait naturellement l'hypothèse de contraintes planes. Cette méthode augmente la largeur de front de la matrice de raideur par le nombre de points de GAUSS de l'élément. Il n'est pas mentionné si le nombre d'itérations nécessaires pour atteindre l'équilibre mécanique est augmenté par rapport au cas où la loi de comportement gère l'hypothèse de contraintes planes.

— `Aster` propose un algorithme dû à DE BORST [Proix 12] qui utilise la convergence de l'équilibre global pour assurer la condition de contraintes planes. `Cyrano3` semble utiliser un algorithme équivalent quand la loi de comportement ne renvoie pas une solution vérifiant l'hypothèse de contrainte planes généralisées. Dans les deux cas, le nombre d'itérations nécessaire pour atteindre l'équilibre mécanique nous a semblé significativement plus important que quand l'hypothèse de contraintes planes est gérée par la loi de comportement.

Le principal défaut de ces méthodes est donc d'induire un coût supplémentaire.

Si nous ne pouvons juger de l'efficacité de la méthode proposée dans `ZeBuLoN`, nous avons l'expérience directe des algorithmes utilisés dans `Aster` et dans `Cyrano3`.

Dans le cas de ce dernier, le surcoût induit est incompatible avec les exigences de l'application en termes de performances : lors des premiers tests de lois `mfront` dans `Cyrano3`, qui ne portait que sur la loi de comportement de la gaine, le nombre d'itérations mécaniques supplémentaires conduisait à un temps de calcul plus de deux fois plus important qu'avec les lois natives de l'application (qui gèrent l'hypothèse de contraintes planes généralisées en interne) alors même que le temps d'intégration de la loi de comportement semblait équivalent. C'est ce constat qui a motivé le travail présenté ici.

## 1.4 PLAN DE LA NOTE

La brève bibliographie faite au paragraphe précédent permet de conclure que seules les implantations traitant explicitement l'hypothèse de contraintes planes généralisées sont compatibles avec les besoins de perfor-

mances numériques de l'application `Cyrano3`. Cela semble obliger à réaliser deux implantations différentes d'une même loi de comportement. Nous proposons dans cette note un compromis acceptable.

Nous proposons en section 2 une modification de l'algorithme implicite qui permet de prendre en compte l'hypothèse de contraintes planes généralisées. Cette modification s'inspire des techniques présentées au paragraphe précédent et peut être utilisée pour la plupart des lois de comportement intégrées en implicite.

Nous montrerons alors que l'implantation de la loi de comportement peut se scinder en deux blocs de code distincts :

- l'un décrit les équations constitutives de la loi et est utilisé quelle que soit l'hypothèse de modélisation ;
- l'autre, de quelques dizaines de lignes et qui ne dépend que de la loi d'élasticité, permet la prise en compte de l'hypothèse de contraintes planes et n'est utilisé que si nécessaire.

Nous arrivons ainsi à une implantation qui peut être utilisée dans toutes les hypothèses de modélisation.

Nous démontrerons en section 3 que cette implantation est numériquement efficace.

# 2 TRAITEMENT GÉNÉRIQUE DE L'HYPOTHÈSE DE CONTRAINTES PLANES DANS UN ALGORITHME IMPLICITE

La résolution implicite est aujourd'hui considéré comme la méthode d'intégration la plus efficace. Son traitement dans `mfront` est décrit dans la documentation de référence [Helfer 13b].

## 2.1 DESCRIPTION DE LA SOLUTION PROPOSÉE

L'hypothèse de contraintes planes est traitée en introduisant la déformation axiale comme une variable interne supplémentaire. La méthode présentée n'est valide que si les hypothèses suivantes sont vérifiées :

1. la loi de comportement utilise la déformation élastique $\underline{\epsilon}^{el}$ comme variable interne ;
2. la déformation totale n'intervient que dans l'équation implicite $f_{\underline{\epsilon}^{el}}$ associée à la déformation élastique ;

### 2.1.1 Modification des équations du système implicite

Nous supposons que l'équation associée à la déformation élastique est la partition des déformations :

$$f_{\underline{\epsilon}^{el}} = \Delta\,\underline{\epsilon}^{el} - \Delta\,\underline{\epsilon}^{to} - \underbrace{\Delta\,\epsilon_z^{to}\,\vec{e}_z\otimes\vec{e}_z}_{\text{Incrément de déformation axiale}} + \dots \tag{1}$$

La dérivée de cette équation par rapport à la déformation axiale est donnée par :

$$\frac{\partial f_{\underline{\epsilon}^{el}}}{\partial\,\Delta\,\epsilon_z^{to}} = \vec{e}_z\otimes\vec{e}_z = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}$$

Le point important de la méthode est *qu'aucune des autres équations du système implicite n'est modifiée.*

## 2.2 ÉQUATION ASSOCIÉE À LA DÉFORMATION AXIALE

L'équation associée à la déformation axiale $\epsilon_z$ est la nullité de la contrainte axiale $\sigma_z$ :

$$f_{\epsilon_z^{to}} = \frac{1}{E'}\sigma_z \tag{2}$$

où $E'$ est un facteur de normalisation tel que $f_{\epsilon_z^{to}}$ soit de l'ordre de grandeur des déformations.

Cette équation est écrite de manière *purement implicite* afin d'assurer la vérification de la condition de contraintes planes en fin de pas de temps.

Dans le cas où l'élasticité est linéaire et isotrope (loi de HOOKE), nous écrirons :

$$f_{\epsilon_z^{to}} = (1-\nu)\,\left(\epsilon_{zz}^{el}+\Delta\,\epsilon_{zz}^{el}\right)+\nu\,\left[\epsilon_{xx}^{el}+\Delta\,\epsilon_{xx}^{el}+\epsilon_{yy}^{el}+\Delta\,\epsilon_{yy}^{el}\right]$$

La seule dérivée non nulle de cette équation est :

$$\frac{\partial f_{\epsilon_z^{to}}}{\partial\,\Delta\,\underline{\epsilon}^{el}} = \begin{pmatrix}\nu\\ \nu\\ 1-\nu\\ 0\end{pmatrix}$$

**Contraintes planes généralisées** Pour les contraintes planes généralisées, l'équation (2) devient :

$$f_{\epsilon_z^{to}} = \frac{1}{E'}\left(\sigma_z - \sigma_z^{\text{équilibre}}\Big|_{t+\Delta\, t}\right)$$

## 2.3 PRINCIPE DE LA MÉTHODE

Les équations (1) et (2) sont les seules modifications à apporter à la loi de comportement pour que celle-ci soit valide en contraintes planes (généralisées ou non).

Deux caractéristiques de cette méthode sont particulièrement intéressantes :

1. la modification portée par l'équation (1) est linéaire. On peut donc calculer $f_{\underline{\underline{\epsilon}}^{el}}$ sans tenir compte du terme supplémentaire $\Delta\,\epsilon_z^{to}\,\vec{e}_z \otimes \vec{e}_z$. Celui-ci peut être ajouté si nécessaire ;
2. l'équation $f_{\epsilon_z^{to}}$ qui assure la condition de contraintes planes ne fait intervenir que les déformations élastiques et ne dépend que de la loi d'élasticité utilisée. Deux cas se rencontrent en pratique : la loi de HOOKE isotrope et la loi de HOOKE orthotrope. D'autres lois d'élasticité sont plus rares.

Ces caractéristiques permettent de scinder l'implantation de la loi en deux parties :

— le premier décrit la loi de comportement sans prendre en compte l'hypothèse de contraintes planes. Ce bloc définit de fait une implantation qui est valide pour toutes les hypothèses de modélisation classiques (axisymétrie en déformations planes généralisées, déformations planes, déformations planes généralisées, axisymétrie, $3D$).

— le second est spécifique aux contraintes planes.

Voyons maintenant comment cela se traduit en pratique.

# 3 APPLICATION À LA LOI DE NORTON

Nous traitons dans cette section la loi non-linéaire la plus simple, la loi de NORTON.

## 3.1 IMPLANTATION À L'AIDE DE MFRONT

La première partie de l'implantation déclare que nous allons intégrer la loi en implicite et nous donnons son nom :

```
1 @Parser Implicit;
2 @Behaviour ImplicitNorton;
```

Par défaut, la loi ne sera pas disponible en contraintes planes ou en contraintes planes généralisées. La ligne suivante précise que la loi supporte toutes les hypothèses de modélisation :

```
4 @ModellingHypotheses {".+"};
```

L'hypothèse de contraintes planes étant vérifiée à la précision de l'algorithme implicite près, il est conseillé d'utiliser une précision relativement basse. Dans nos tests, une valeur du critère d'arrêt de $10^{-11}$ nous a semblé satisfaisante (par défaut, cette valeur est de $10^{-8}$)[1]. Nous le précisons ainsi :

```
5 @Epsilon 1.e-11;
```

Les lignes suivantes déclarent les propriétés matériau, les variables locales et la variable interne de la loi :

```
11 @MaterialProperty stress young; /* mandatory for castem */
12 young.setGlossaryName("YoungModulus");
13 @MaterialProperty real nu;     /* mandatory for castem */
14 nu.setGlossaryName("PoissonRatio");
15
16 @LocalVariable real      lambda;
17 @LocalVariable real      mu;
18
19 // store the Von Mises stress
20 // for the tangent operator
21 @LocalVariable real seq;
22 // store the derivative of the creep function
23 // for the tangent operator
24 @LocalVariable real df_dseq;
25 // store the normal tensor
26 // for the tangent operator
27 @LocalVariable Stensor n;
28
29 @StateVariable real     p;
30 @PhysicalBounds p in [0:*[;
```

Nous déclarons ensuite la déformation axiale $\epsilon_z^{to}$, qui est spécifique aux hypothèses de contraintes planes. Pour l'hypothèse de contraintes planes généralisées, nous déclarons également la contrainte imposée comme une variable externe dont l'évolution est fournie par le code appelant :

1. Une valeur basse est de toute manière requise pour avoir une matrice tangente cohérente de bonne qualité quand celle-ci est déduite de la partie supérieure de l'inverse de la jacobienne, ce qui est une technique très utilisée.

```
@StateVariable<PlaneStress> strain etozz;
PlaneStress::etozz.setGlossaryName("AxialStrain");

@StateVariable<AxisymmetricalGeneralisedPlaneStress> strain etozz;
AxisymmetricalGeneralisedPlaneStress::etozz.setGlossaryName("AxialStrain");
@ExternalStateVariable<AxisymmetricalGeneralisedPlaneStress> stress sigzz;
AxisymmetricalGeneralisedPlaneStress::sigzz.setGlossaryName("AxialStress");
```

Deux syntaxes particulières sont utilisées ici :

- après un mot clé, les chevrons `'<'` et fermant `'>'` permettent de restreindre l'action de ce mot clé à certaines hypothèses de modélisation. Ici la déformation axiale ne sera définie que pour les hypothèses de contraintes planes ;
- pour appeler une méthode (ici `setGlossaryName`) pour une variable qui n'est définie que pour une hypothèse de modélisation particulière, le nom de la variable est préfixée par le nom de l'hypothèse. Ainsi, `PlaneStress::etozz` désigne la déformation axiale associée à l'hypothèse de contraintes planes.

Le fait d'associer à la déformation axiale le nom de glossaire `AxialStrain` est particulièrement important. Ceci permet à `mfront` d'identifier quelle variable interne porte la déformation axiale et `mfront` pourra l'utiliser dans les stratégies « grandes transformations » pour un calcul correct de la contrainte de CAUCHY [Helfer e].

De même, le fait d'associer à la contrainte axiale imposée `sigzz` le nom de glossaire `AxialStress` permettra à `Cyrano3` de savoir comment remplir les champs d'entrée de la loi de comportement.

Les variables locales sont ensuite initialisées :

```
/* Initialize Lame coefficients */
@InitLocalVariables{
  using namespace tfel::material::lame;
  lambda = computeLambda(young,nu);
  mu = computeMu(young,nu);
} // end of @InitLocalVariables
```

La loi d'élasticité est ensuite précisée :

```
@ComputeStress{
  sig = lambda*trace(eel)*Stensor::Id()+2*mu*eel;
} // end of @ComputeStresss
```

Notons bien que la loi d'élasticité est écrite sans référence à une hypothèse de modélisation : pour les hypothèses de contraintes planes, la condition de contraintes planes est imposée par la résolution et non dans la loi d'élasticité.

La partie commune à toutes les hypothèses de modélisation, l'intégration de la loi de comportement, détaillée par ailleurs [Helfer 13b], est ensuite donnée :

```
@Integrator{
  const real A = 8.e-67;
  const real E = 8.2;
  seq = sigmaeq(sig);
  const real tmp = A*pow(seq,E-1.);
  df_dseq = E*tmp;
  real inv_seq(0);
  n = Stensor(0.);
  if(seq > 1.e-8*young){
    inv_seq = 1/seq;
    n       = 1.5*deviator(sig)*inv_seq;
  }
```

```
63    feel += dp*n-deto;
64    fp   -= tmp*seq*dt;
65    // jacobian
66    dfeel_ddeel += 2.*mu*theta*dp*inv_seq*(Stensor4::M()-(n^n));
67    dfeel_ddp    = n;
68    dfp_ddeel    = -2*mu*theta*df_dseq*dt*n;
69  } // end of @Integrator
```

La partie spécifique à l'hypothèse de contraintes planes est alors donnée par le code suivant :

```
71  @Integrator<PlaneStress,Append,AtEnd>{
72    // the plane stress equation is satisfied at the end of the time
73    // step
74    const stress szz = (lambda+2*mu)*(eel(2)+deel(2))+lambda*(eel(0)+deel(0)+eel(1)+deel(1));
75    fetozz   = szz/young;
76    // modification of the partition of strain
77    feel(2) -= detozz;
78    // jacobian
79    dfeel_ddetozz(2)=-1;
80    dfetozz_ddetozz  = real(0);
81    dfetozz_ddeel(2) = (lambda+2*mu)/young;
82    dfetozz_ddeel(0) = lambda/young;
83    dfetozz_ddeel(1) = lambda/young;
84  }
```

Nous précisons que :

- ce bloc n'est valide qu'en contraintes planes (option `PlaneStress`) ;
- ce bloc doit être ajouté à la suite de la partie commune à toutes les modélisations (options `Append` et `AtEnd`) ;

De même, la partie spécifique à l'hypothèse de contraintes planes généralisées est :

```
86  @Integrator<AxisymmetricalGeneralisedPlaneStress,Append,AtEnd>{
87    // the plane stress equation is satisfied at the end of the time
88    // step
89    const stress szz = (lambda+2*mu)*(eel(1)+deel(1))+lambda*(eel(0)+deel(0)+eel(2)+deel(2));
90    fetozz   = (szz-sigzz-dsigzz)/young;
91    // modification of the partition of strain
92    feel(1) -= detozz;
93    // jacobian
94    dfeel_ddetozz(1)=-1;
95    dfetozz_ddetozz  = real(0);
96    dfetozz_ddeel(1) = (lambda+2*mu)/young;
97    dfetozz_ddeel(0) = lambda/young;
98    dfetozz_ddeel(2) = lambda/young;
99  }
```

Enfin, nous effectuons le calcul de matrice tangente cohérente :

```
101 @IsTangentOperatorSymmetric true;
102 @TangentOperator{
103   using namespace tfel::material::lame;
104   if((smt==ELASTIC)||(smt==SECANTOPERATOR)||
105      (smt==TANGENTOPERATOR)){
106     computeAlteredElasticStiffness<hypothesis,Type>::exe(Dt,lambda,mu);
107   } else if (smt==CONSISTENTTANGENTOPERATOR){
108     StiffnessTensor Hooke;
109     Stensor4 Je;
110     computeElasticStiffness<N,Type>::exe(Hooke,lambda,mu);
111     getPartialJacobianInvert(Je);
112     Dt = Hooke*Je;
```

```
113     } else {
114       return false;
115     }
116   }
```

Ce calcul fait intervenir une petite subtilité :

- la matrice élastique est la matrice « altérée », c'est à dire qu'elle prend l'hypothèse de modélisation en compte. Pour cela, `mfront` met à disposition la fonction `computeAlteredElasticStiffness` ;
- pour un calcul de la matrice tangente cohérente par une méthode quasi-automatique [Helfer 13b], il est nécessaire d'utiliser la matrice d'élasticité non altérée. Celle-ci est calculée par la fonction `computeElasticStiffness` qui prend en argument la dimension de l'espace mais pas l'hypothèse de modelisation.

## 3.2 Test à l'aide de `mtest`

Nous pouvons tester notre implantation à l'aide de `mtest` [Helfer 14]. Pour cela, nous commençons par compiler la loi avec l'interface `Cyrano3` :

```
$ mfront --obuild -interface=cyrano norton.mfront
```

Nous choisissons de modéliser un essai de fluage dans la direction radiale. Les contraintes orthoradiales et axiales sont donc nulles.

Le fichier d'entrée du test est le suivant :

```
1  @ModellingHypothesis 'AxisymmetricalGeneralisedPlaneStress';
2  @Behaviour<cyrano> 'libCyranoBehaviour.so' 'cyranoimplicitnorton';
3
4  @MaterialProperty<constant> 'YoungModulus'      150.e9;
5  @MaterialProperty<constant> 'PoissonRatio'        0.3;
6
7  @Real 'srr' 20.e6;
8  @ImposedStress 'SRR' 'srr';
9  // Initial value of the elastic strain
10 @Real 'EELRR0'  0.00013333333333333333;
11 @Real 'EELZZ0' -0.00004;
12 @InternalStateVariable 'ElasticStrain' {'EELRR0','EELZZ0','EELZZ0'};
13 @InternalStateVariable 'AxialStrain'   'EELZZ0';
14 // Initial value of the total strain
15 @Strain {'EELRR0',0.,'EELZZ0'};
16 // Initial value of the stresses
17 @Stress {'srr',0.,0.};
18
19 @ExternalStateVariable 'Temperature' 293.15;
20 @ExternalStateVariable 'AxialStress' 0.;
21
22 @Times {0.,3600 in 20};
```

En ligne 15, la déformation totale axiale à l'instant initial est imposée nulle, puisque cette valeur est ici portée par la variable interne `AxialStrain`.

La contrainte axiale d'équilibre est imposée via la variable externe `AxialStress`.

L'exécution du cas test permet de vérifier la bonne convergence de l'équilibre. Ainsi, la convergence au dernier pas de temps est la suivante :

```
iteration 1 : 0.000111943 5.74196e+06 (0.00222946 0 -0.00108729)
```

```
iteration 2 : 4.37109e-05 1.35551e+06 (0.00225815 0 -0.0011023)
iteration 3 : 3.87204e-06 104347 (0.00226066 0 -0.00110366)
iteration 4 : 2.50326e-08 673.864 (0.00226068 0 -0.00110367)
iteration 5 : 1.05845e-12 0.0299959 (0.00226068 0 -0.00110367)
iteration 6 : 3.34299e-14 0.00587752 (0.00226068 0 -0.00110367)
```

Nous voyons que chaque itération permet de gagner deux ordres de grandeurs, ce qui est caractéristique d'une convergence quadratique : la matrice tangente cohérente est de bonne qualité.

Une exécution de `mtest` dans un mode plus verbeux affiche la valeur de la matrice tangente cohérente ressortie par la loi (par simplicité, nous avons omis les termes liés aux multiplicateurs de LAGRANGE :

```
5.99688e+10 -2.28882e-05 5.48282e+10
      0            0           0
5.48282e+10  7.62939e-06 1.20988e+11
```

Rappelons que la composante axiale est la seconde composante des tenseurs en contraintes planes généralisées. Nous voyons que la seconde ligne est nulle. La seconde colonne est numériquement nulle.

Nous pouvons comparer l'opérateur tangent cohérent à une approximation numérique. Pour cela, ajoutons les lignes suivantes au fichier mtest :

```
1  @CompareToNumericalTangentOperator true;
2  @TangentOperatorComparisonCriterium 3e6;
3  @NumericalTangentOperatorPerturbationValue 1.e-8;
```

Nous vérifions ainsi numériquement la qualité de la matrice tangente cohérente.

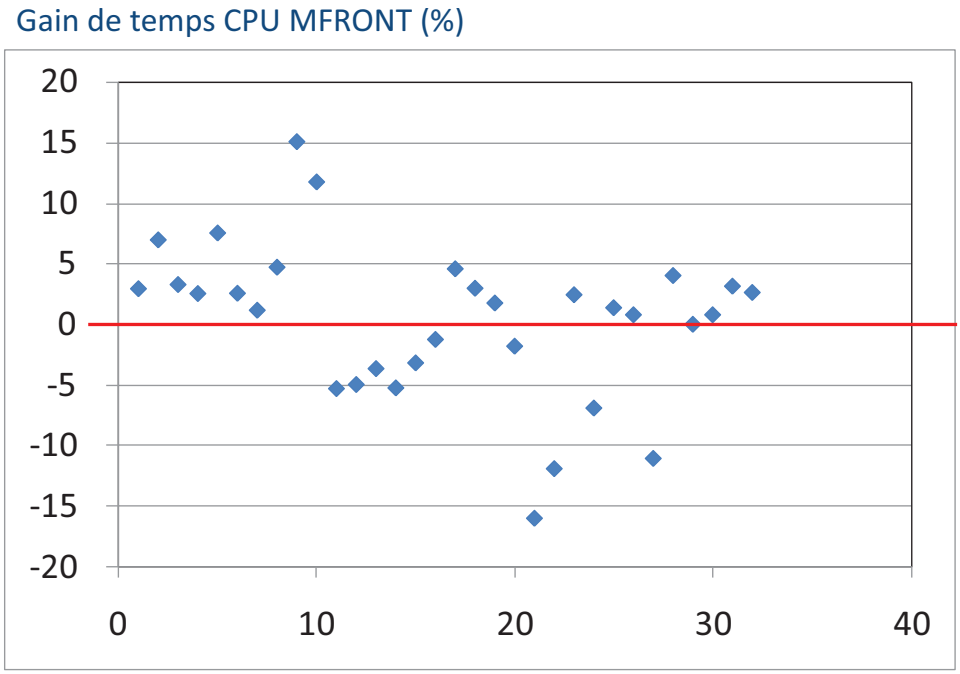


**FIGURE 1 :** Comparaison des performances obtenues avec l'implantation `mfront` et l'implantation native d'une même loi de comportement viscoplastique orthotrope de gaine. Cette loi est testée sur l'ensemble de la base de validation usuelle de `Cyrano3` (le numéro du cas test est reporté en abscisse).

# 4 CONCLUSIONS

Cette note a présenté comment il est possible de traiter les hypothèses de contraintes planes tout en conservant une implantation unique quelle que soit l'hypothèse de modélisation considérée : une même implantation peut désormais être partagée entre les applications `alcyone` et `Cyrano3`.

Pour mesurer les performances de la solution proposée, l'équipe `Cyrano3` a comparé les temps de calcul obtenue avec l'implantation `mfront` et l'implantation native d'une même loi de comportement viscoplastique orthotrope de gaine sur l'ensemble de la base de validation usuelle de plus de $30$ cas tests [2]

Les performances obtenues, illustrées en figure 1 sont légèrement supérieures à celles obtenues avec les lois natives de `Cyrano3`, avec des différences pouvant aller jusqu'à $20\%$ dans les deux sens.

Sur la base de ce cas test, les développeurs de `Cyrano3` considère que la solution proposée est compatible avec les exigences industrielles de cette application.

## 4.1 PERSPECTIVES

Bien que seules les implantations implicites des lois de comportement aient été considérées dans cette note, les développements proposés ici peuvent être appliqués aux implantations basées sur les méthodes de RUNGE-KUTTA.

Pour être complet, c'est à dire pour que les développeurs de `Cyrano3` aient accès à toutes les potentialités de `mfront`, il serait nécessaire d'enrichir les analyseurs spécifiques qui décrivent les lois de comportement

2. La loi utilisée étant généralement hors de son domaine d'usage, les résultats n'ont aucune valeur physique. Il a cependant été vérifié que les deux implantations conduisaient au même résultat.

isotropes (plasticité et viscoplasticité).

# 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.

[Doghri 00] DOGHRI ISSAM. *Mechanics of deformable solids : linear, nonlinear, analytical, and computational aspects*. Springer, Berlin ; New York, 2000.

[Helfer 13a] 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 13b] 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 et PROIX JEAN-MICHEL. *MTest : un outil de test unitaire de lois comportement mécanique*. Note technique 14-016, CEA DEN/DEC/SESC/LSC, 2014.

[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).

[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.

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

# LISTE DES FIGURES