Nouveautés de la version 3.0 de TFEL

T. Helfer

2013

RÉSUMÉ

Ce document décrit les principales nouveautés de la version 3.0 de TFEL, et plus particulièrement sur mfront.

SOMMAIRE

1 NOUVEAUTÉS MFRONT

1.1 SUPPORT DES DÉFORMATIONS LOGARITHMIQUES POUR LES CODES COMBUSTIBLES $1D$ ÉCRITS EN PETITES DÉFORMATIONS

1.2 POSSIBILITÉ DE GÉRER UN PAS DE TEMPS ADAPTATIF DEPUIS LA LOI DE COMPORTEMENT

1.3 CALCUL DES DILATATIONS LIBRES, DU TENSEUR D'ÉLASTICITÉ, DU TENSEUR DE HILL ET DE PROPRIÉTÉS MATÉRIAUX PAR LES LOIS DE COMPORTEMENT

Nous traitons dans ce paragraphe les fonctionnalités relatives au calcul des dilatations libres, du tenseur d'élasticité, du tenseur de Hill et, de manière générale, de propriétés matériau, par la loi de comportement.

Il s'agit d'un sujet d'apparence anodine, mais qui s'est avéré étonnamment complexe pour trois raisons :
- la prise en compte des contraintes planes pour le calcul du tenseur d'élasticité. Ce point est discuté au paragraphe **??** ;
- la prise en compte des évolutions des propriétés au cours du calcul, notamment avec la température dont l'évolution sur le pas de temps est connu par la loi de comportement. Nous reportons la discussion de ce point au paragraphe 1.3.5.
- le cas des matériaux orthotropes.

Pour les matériaux orthotropes, la difficulté provient essentiellement du problème de la cohérence de la définition des axes matériau entre les différentes hypothèses de modélisation.

Pour illustrer cette difficulté, rappelons que tous les codes imposent que :
- la direction axiale soit la troisième direction matériau en $2D$ déformation plane ou en $2D$ contrainte plane.
- la direction orthoradiale soit la la troisième direction matériau en axisymétrie.

Dans les versions précédentes de `mfront`, ses difficultés étaient gérées ainsi :
- pour le calcul des dilatations libres, on laissait le code appelant faire leur évaluation : le code fournissait alors à la loi un incrément de déformation dit « mécanique »[1].
- pour le calcul du tenseur d'élasticité, l'utilisateur pouvait utiliser le mot clé `@RequireStiffnessTensor` pour que `mfront` rajoute à la liste des propriétés matériau les propriétés élastiques. Celles-ci devaient alors être fournies par le code dans « le bon ordre ». Charge à l'utilisateur de ne pas se tromper : il fallait en particulier être prudent si l'on changeait d'hypothèse de modélisation et penser à inverser l'ordre des axes.
- pour le calcul du tenseur de HILL, on testait explicitement l'hypothèse de modélisation et l'on devait intervertir l'ordre des coefficients de HILL.

Ces trois solutions sont insatisfaisantes pour les raisons suivantes :
- pour les deux premiers points (dilatations libres et tenseur d'élasticité), une partie de l'information physique se trouve en dehors de la loi.
- pour gérer correctement les grandes transformations, le calcul des dilatations libres, ou du moins leur traitement, doit être réalisé par la loi de comportement.
- dans tous les cas, on multiplie les causes d'erreurs potentielles.

La meilleure solution serait de permettre l'évaluation dans la loi de comportement des dilatations libres, du tenseur d'élasticité et du tenseur de Hill.

Une première étape pour résoudre ces difficultés est de permettre à l'utilisateur de spécifier une convention de rangement des axes d'orthotropie.

1. Il s'agit de l'incrément de déformation totale auquel on a soustrait l'incrément des dilatations libres.

1.3.1 Convention de rangement des axes en orthotropie

1.3.2 Calcul des déformations libres

Le mot clé `@ComputeThermalExpansion`

Le mot clé `@Swelling`

Le mot clé `@AxialGrowth`

Le mot clé `@ComputeStressFreeExpansion` Le mot clé `@ComputeStressFreeExpansion` permet à l'utilisateur d'implanter le calcul de dilatations libres quelconques. L'utilisateur doit mettre à jour des tenseurs d'ordre deux symétriques nommés respectivement `dl0_l0` et `dl1_l0`. `dl0_l0` représente la variation de longueur du corps par rapport à la longueur initial en début de pas de temps, et `dl1_l0` cette variation en fin de pas. En pratique, ces deux tenseurs sont généralement diagonaux.

1.3.3 Calcul du tenseur d'élasticité

Cas des contraintes planes

Le mot clé `@ComputeStiffnessTensor`

1.3.4 Propriétés matériaux externes

1.3.5 Évaluation des propriétés matériaux externes

1.4 Briques élémentaires de lois de comportement

1.5 Nouvelles interfaces

1.5.1 L'interface `Europlexus`

1.5.2 L'interface `Abaqus`

1.5.3 L'interface `AbaqusExplicit`

1.6 Utilitaires

1.6.1 L'utilitaire `mfront-doc`

L'utilitaire `mfront-doc` permet de générer un fichier au format `Markdown` à partir des commentaires écrits dans un fichier `mfront`. Ce fichier peut ensuite être converti en différents formats (pdf, docx, tex, html) par l'utilitaire `pandoc`.

1.6.2 L'utilitaire `mfront-query`

L'utilitaire `mfront-query` permet d'obtenir en ligne commande des informations sur un fichier `mfront`.

1.7 Création du module `mfront`

Les fichiers de `mfront` peuvent désormais être analysés en `python` depuis le module éponyme.

`http://tfel.sourceforge.net/mfront-python.html`

1.8 Développements divers

1.8.1 Refactorisation du code

Les classes `TargetsDescription` et `LibraryDescription`

1.8.2 Options en ligne de commandes

L'option `-D` L'option `-D` permet de définir des macros pour le préprocesseur `C/C++` en ligne de commande.

Substitutions `mfront` permet désormais de substituer certaines parties des fichiers d'entrée par d'autres depuis la ligne de commande. La syntaxe retenue est relativement classique et s'inspire du logiciel `autoconf`.

Plus précisément, les options passées à `mfront` en ligne de commande de la forme `-@XXX@=YYY`, où `XXX` est une chaine de caractères quelconque de ne contenant pas le caractère `@`, substituent chaque occurrence de `-@XXX@` par `YYY` dans les fichiers traités.

Par exemple, considérons l'appel suivant :

```
$mfront -@p1_value@=12.9 test.mfront
```

Toutes les occurrences de `@p1_value@` seront remplacées par `12.9`.

2 Nouveautés `mtest`

2.1 Modélisation des tubes sous pression

2.2 Lecture des données dans un fichier

2.3 Création du module `mtest`

Les fonctionnalités de `mtest` sont désormais accessibles en `python` depuis le module éponyme.

Ce module permet à la fois de traiter le point matériel et les tubes sous pressions.

Une description de ce module est donnée à la page suivante :

`http://tfel.sourceforge.net/mtest-python.html`

Note Dans les versions précédentes, un module `mfront.mtest` avait ce rôle. Celui existe toujours pour assurer la compatibilité avec les versions précédentes. Son utilisation conduit cependant à un message d'avertissement conseillant d'utiliser le nouveau module.

2.4 NOUVELLES MÉTHODES D'ACCÉLÉRATION DE CONVERGENCE

2.5 PAS DE TEMPS ADAPTATIF

3 NOUVEAUTÉS `TFEL`

3.1 PASSAGE AU STANDARD C++-11

3.2 NOUVEAUTÉS DE LA LIBRAIRIE MATHÉMATIQUE

3.2.1 Nouvelles méthodes dans la classes `t2tot2`

La classe `t2tot2` représente une transformation linéaire des l'espace des tenseurs non symétriques vers l'espace des tenseurs non symmétriques.

De nouvelles méthodes retournant des valeurs remarquables de ces transformations ont été ajoutées :
- `Id` renvoie l'identité ;
- `IxI` renvoie le produit tensoriel du tenseur identité non symétrique par lui-même ;
- `K` renvoie le dérivée du déviateur d'un tenseur non symétrique par rapport à ce tenseur :

$$\underline{\underline{\mathbf{K}}} = \underline{\underline{\mathbf{I}}} - \frac{1}{3}\,\underset{\sim}{\mathbf{I}} \otimes \underset{\sim}{\mathbf{I}}$$

3.2.2 Fonctions isotropes de tenseurs et dérivées

Un tenseur symétrique $\underline{\mathbf{s}}$ peut toujours être diagonalisé. Soit $\lambda_1, \lambda_2, \lambda_3$ ses trois valeurs propres.

Le tenseur $\underline{\mathbf{s}}$ se décompose ainsi :

$$\underline{\mathbf{s}} = \sum_{i=1}^{3} \lambda_i \, \underline{\mathbf{n}}_i \tag{1}$$

où les tenseurs $\underline{\mathbf{n}}_i$ sont appelés tenseurs propres de $\underline{\mathbf{s}}$.

Une fonction $\underline{\mathbf{f}}$ de tenseur d'ordre 2 symétrique est dite *isotrope* si elle vérifie :

$$\underline{\mathbf{f}}\,(\tilde{\underline{\mathbf{s}}}) = \widetilde{\underline{\mathbf{f}}\,(\underline{\mathbf{s}})}$$

pour changement de base représenté par $\tilde{}$.

Toute fonction définie par une série entière est isotrope. Ainsi les fonctions logarithme $\log$ et exponentielle $\exp$ sont des fonctions isotropes. Dans ce cas, f désigne à la fois une fonction réelle et une fonction de tenseur d'ordre 2 symétrique.

Une telle fonction peut être évaluée à l'aide des vecteurs propres et des tenseurs propres :

$$f\,(\underline{\mathbf{s}}) = \sum_{i=1}^{3} f\,(\lambda_i)\,\underline{\mathbf{n}}_i \tag{2}$$

L'équation (2) peut être dérivée si f est dérivable.

La classe `stensor` comporte désormais les méthodes `computeIsotropicFunction`, `computeIsotropicFunctionDerivative` et `computeIsotropicFunctionAndDerivative` qui permettent respectivement de calculer la valeur d'une fonction isotrope, la valeur de sa dérivée ou les deux en même temps.

Trois fonctions du même nom sont également disponibles.

3.2.3 Implémentation de l'opération de `push_forward` et de `pull_back` pour les tenseurs d'ordre 4

3.2.4 Convertion du second tenseur de Piola-Kirchhoff en contraintes de Cauchy exprimées dans le repère corotationnel

Le code `Abaqus-Explicit` fournit le tenseur de dilatation $\underline{\mathbf{U}}$ et demande à ce que soit renvoyé le tenseur des contraintes de CAUCHY $\underline{\tilde{\sigma}}$ dans le repère corotationnel. Ce dernier est relié au second tenseur de Piola-Kirchhoff $\underline{\mathbf{S}}$ par la formule :

$$\underline{\tilde{\sigma}} = \frac{1}{\det\left(U\right)}\underline{\mathbf{U}}.\underline{\mathbf{S}}.\underline{\mathbf{U}}$$

Cette opération est effectuée de manière optimisée par la fonction :

```
convertSecondPiolaKirchhoffStressToCorotationnalCauchyStress
```

L'opération inverse est réalisée par la fonction :

```
convertCorotationnalCauchyStressToSecondPiolaKirchhoffStress
```

3.3 CHANGEMENT ET NOUVEAUTÉS DE TFEL/UTITILTIES

3.3.1 La classe `CxxTokenizer`

La classe `CxxTokenizer`, à la base de la lecture des fichiers `mfront` et `mtest`, a été réécrite pour de meilleures performances et une plus grande conformité au standard C++.

3.4 NOUVELLES OPTIONS DE COMPILATION

3.4.1 Options `cmake`

L'option `enable-fast-math` L'option `enable-fast-math` permet de sélectionner des options de compilations permettant une optimisation plus agressive du code en autorisant le compilateur à ne pas respecter la norme `IEEE 754`.

Pour les compilateurs `clang` et `gcc`, cette option ajoute le drapeau `-ffast-math`.

Pour le compilateur `Visual Studio`, cette option supprime le drapeau `/fp:strict`.

Pour le compilateur `intel`, cette option n'a pas d'effet.

4 PUBLICATION

5 ÉLÉMENTS D'ASSURANCE QUALITÉ

5.1 INDICATEURS DE LA QUALITÉ DU CODE

5.1.1 Analyse statique du code

L'outil `scan-build`

L'outil `Coverity`

5.1.2 Sanitizers

5.1.3 Cas tests de vérification et taux de couverture

`gcov`

5.2 DOCUMENTATION

5.2.1 Formation

5.2.2 Documentation en ligne de commande

Documentation des mots-clés `mfront`

Documentation des mots-clés `mtest`

5.2.3 Documentation des modules python de `mtest`

5.2.4 Page wiki pleiades

5.3 PORTABILITÉ

5.3.1 Portage sur Microsoft Visual Studio 2015

5.3.2 Liste des compilateurs supportés

5.4 LISTES DES CHANGEMENTS CONDUISANT À DES INCOMPATIBILITÉS DE VERSION

Différents codes se basent sur la librairie `TFEL` : l'architecture `pleiades`, les applications basées sur cette architecture, et en particulier `licos`.

Dans la mesure du possible, les changements pouvant conduire à des incompatibilités sont évités. Elles ne sont introduites que si leurs conséquences sont minimales. Voici les principaux exemples connus à ce jour et une évaluation de leurs conséquences :

— modification majeur des classes d'interface de `mfront`. Les interfaces `pleiades`, `germinal` et `licos` pour les propriétés matériaux et les modèles sont impactées. Les conséquences de ces changements sont faibles car des copies locales de ces interfaces est présente dans les tests de `mfront`. Ces copies ayant été mise à jour, le travail de portage est direct (recopie de ces nouvelles versions dans les applications cible, recompilation et test).

— changement du nom d'une données membre dans la classe `CxxTokenizer` : `fileTokens` devient `tokens`. Cette donnée membre est utilisée par les classes de lecture des jeux de données de l'architecture `pleiades` et de `licos`. Seules une dizaine de lignes de codes sont concernées. Cette donnée membre y est utilisée pour avoir l'adresse de la fin de fichier : il est conseillé d'utiliser la méthode `end` prévue à cet effet plutôt que d'accéder directement à cette donnée membre.
— retrait de la classe `tfel::utilities::Name` : cette classe est utilisée pour la sauvegarde/reprise dans `licos`. Cette classe reposait sur la définition d'une méthode statique `getName` dans la plupart des classes de `TFEL` ce qui posait des problèmes de maintenance. Nous conseillons de reprendre cette classe dans `licos` et d'en adapter le code quand cela est nécessaire. Il s'agit d'une action ciblée d'une ou deux journées qu'il est possible de déléguer à la tierce maintenance applicative du projet `pleiades`.

5.5 CORRECTIONS D'ANOMALIES

5.5.1 Ticket # 25

Les valeurs initiales de la déformation ou du gradient de la transformation n'étaient pas renseignées dans les fichiers `mtest` généré en cas de non-convergence d'une loi.

Plus de détails peuvent être trouvés à la page :

`https://sourceforge.net/p/tfel/tickets/25/`

5.5.2 Ticket # 26

La méthode `setMaximumIncrementValuePerIteration` n'était pas disponible pour les variables d'integration (déclarées par `@IntegrationVariable`), mais uniquement pour les variables d'états (déclarées par `@StateveVariable`).

Plus de détails peuvent être trouvés à la page :

`https://sourceforge.net/p/tfel/tickets/26/`

5.5.3 Ticket # 27

La dépendance de la librairie `TFELMaterial` envers la librairie `TFELUtilities` n'était pas prise en compte par l'utilitaire `tfel-config`.

Plus de détails peuvent être trouvés à la page :

`https://sourceforge.net/p/tfel/tickets/27/`

5.5.4 Ticket # 28

La documentation de reference sur les lois de comportement contenait une erreur dans l'expression de la matrice jacobienne de l'exemple décrivant l'implantation implicite d'une loi de comportement viscoplastique orthotrope.

Plus de détails peuvent être trouvés à la page :

`https://sourceforge.net/p/tfel/tickets/28/`

5.5.5 Ticket # 29

L'exemple d'utilisation générée par l'interface `Cast3M` pour des lois de comportement orthotrope était invalide : les coefficients décrivant le repère d'orthotropie était initialisé explicitement.

Plus de détails peuvent être trouvés à la page :

`https://sourceforge.net/p/tfel/tickets/29/`

5.5.6 Ticket # 31

Par cohérence avec le jeu de données `mtest`, les méthodes suivantes ont été ajoutées à la classe `python` éponyme :

- `setStrainEpsilon`
- `setDeformationGradientEpsilon`
- `setOpeningDisplacementEpsilon`
- `setStressEpsilon`
- `setCohesiveForceEpsilon`

Plus de détails peuvent être trouvés à la page :

`https://sourceforge.net/p/tfel/tickets/31/`

5.5.7 Ticket # 32

L'interface `python` de `mtest` ne permettait pas de fixer les valeurs initiales des inconnues (déformation, gradient de la déformation ou saut de déplacement suivant les cas) et des des forces (contrainte de Cauchy, force cohésive). Les méthodes sont `setDrivingVariablesInitialValues` et `setThermodynamicForcesInitialValue` ont été ajoutées pour corriger ce point.

Par cohérence avec le jeu de données `mtest`, les méthodes suivantes ont également été ajoutées à la classe `python` éponyme :

- `setStrain`
- `setDeformationGradient`
- `setOpeningDisplacement`
- `setStress`
- `setCohesiveForce`

Plus de détails peuvent être trouvés à la page :

`https://sourceforge.net/p/tfel/tickets/32/`

6 PERSPECTIVES

6.0.1 Lois de comportement à gradient de variables internes