
Notice d'utilisation de MFront avec code_aster

Résumé :

On décrit ici l'utilisation du couplage entre **code_aster** et le module d'intégration de lois de comportement MFront, qui est diffusé en Open Source sous licence GPL et CECILL-A (<http://tfel.sourceforge.net>).

MFront permet de définir une loi de comportement de façon simple, proche des équations physiques, sans avoir à se soucier des méthodes de résolution. Il offre des classes de manipulation de données (tenseurs, par exemple) de façon conviviale et met à disposition des algorithmes de résolution des équations non linéaires associées efficaces en termes de performances. MFront est adapté aussi bien aux schémas temporels implicites qu'explicites.

Ces comportements sont utilisables dans **code_aster** pour les modélisations 3D et 2D, coques, THM et pour les éléments de joints. Les grandes déformations peuvent être utilisées avec `GDEF_LOG`. De plus, MFront produit un code efficace, car les opérations tensorielles sont optimisées comme le montrent les *benchmarks* réalisés avec des comportements équivalents de **code_aster**.

Il faut bien noter que l'utilisation de ces lois de comportement « à façon » (mot-clé `RELATION='MFRONT'` sous `COMPORTEMENT`) implique une validation spécifique pour l'étude envisagée, car on se place hors du domaine qualifié de **code_aster**.

Ce document décrit la façon d'utiliser un comportement défini par l'utilisateur à l'aide de MFront dans un fichier de commandes de **code_aster**.

1 Définition d'une loi de comportement avec MFront

1.1 Écriture d'un comportement MFront

On écrit la loi de comportement dans le fichier MFront, suivant les conseils fournis dans le tutoriel[1], ou en s'inspirant d'un comportement déjà disponible dans les tests de `code_aster`.

Nous prendrons ici pour exemple le comportement visco-plastique de Lemaître écrit sous forme implicite. Le fichier « Lemaitre.mfront » s'écrit :

```
@Parser Implicit;
@Behaviour Lemaitre;
@Algorithm NewtonRaphson_NumericalJacobian;
@Theta 0.5 ;

@RequireStiffnessTensor;
@MaterialProperty real m;
@MaterialProperty real UNSurK;
@MaterialProperty real UNSurM;

@StateVariable real p;

@ComputeStress{ sig = D*eel;}

@Integrator{
  const real seq = sigmaeq(sig);
  const real p_ = max((p+theta*dp),1.e-8) ;
  if(seq > 0.){
    real slem = seq*UNsurK/pow(p_,UNsurM);
    real vp = pow(slem,m) ;
    Stensor n = 1.5*deviator(sig)/seq;
    feel += vp*dt*n;
    fp -= vp*dt;
  }
  feel -= deto;
}

@TangentOperator{
  Stensor4 Je;
  getPartialJacobianInvert(Je);
  Dt = D*Je;
}
```

Cette loi est équivalente à la loi LEMAITRE, disponible dans `code_aster` [R5.03.08].

1.2 Création de la bibliothèque dynamique à partir d'un comportement MFront

Pour être utilisable dans `code_aster`, le comportement MFront ainsi programmé doit être transformé en bibliothèque dynamique. Pour cela, il faut le « compiler ». Cela peut être fait de deux façons :

1) **Solution recommandée : utilisation de `CREA_LIB_MFRONT`**

La macro-commande `CREA_LIB_MFRONT` permet de construire la bibliothèque automatiquement. Il suffit de fournir le fichier MFront en entrée sur une unité logique.

```
CREA_LIB_MFRONT( UNITE_MFRONT=10,  
                 UNITE_LIBRAIRIE=11, )
```

Ce mode de fonctionnement est à privilégier, notamment car il facilite l'utilisation dans AsterStudy. Pour plus d'informations, voir [U7.03.04].

2) Pour les experts : construction de la bibliothèque à la main

Dans un terminal, « compiler » le fichier MFront :

```
mfront --obuild -interface=aster Lemaitre.mfront
```

Cela crée deux répertoires, `include` et `src`. Ce dernier contient en particulier la bibliothèque dynamique `libAsterBehaviour.so` qui sera fournie dans les données de l'étude.

Remarque : compilation en mode DEBUG

Si l'utilisateur souhaite déboguer sa loi, il doit la construire de cette façon :

```
CXXFLAGS="-g `tfel-config --oflags --compiler-flags`" mfront --obuild  
--interface=aster Lemaitre.mfront
```

1.3 Utilisation du comportement dans le fichier de commandes

L'utilisation du comportement MFront dans le fichier de commandes s'effectue de la façon suivante :

1. Les données matériaux sont fournies dans `DEFI_MATERIAU` [U4.43.01], sous `MFRONT` / `MFRONT_FO`.

Par exemple pour la loi de Lemaitre :

```
young=210000.  
poisson=0.3  
m=11.  
un_sur_k = 0.00032840723  
un_sur_m = 0.178571429
```

```
MATFRONT=DEFI_MATERIAU(  
    ELAS=_F(E=young, NU=poisson),  
    MFRONT=_F(LISTE_COEF = (young , poisson ,  
                          m, un_sur_k, un_sur_m)),  
    ),  
    )
```

Les valeurs de `LISTE_COEF` (`young`, `poisson`, ...) sont à donner dans l'ordre des `@MaterialProperties` du fichier MFront.

2. Sous le mot-clé `COMPOTEMENT` des commandes de calcul non-linéaire `STAT_NON_LINE`, `DYNA_NON_LINE`, ou `SIMU_POINT_MAT` :

```
MFi=STAT_NON_LINE(MODELE=MO,  
                 CHAM_MATER=CMF,  
                 INCREMENT=_F(LIST_INST = L_INST, NUME_INST_FIN = 5),  
                 NEWTON=_F(REAC_ITER=1),  
                 EXCIT=( _F(CHARGE = CH1),  
                       _F(CHARGE = CH2)),  
                 COMPOTEMENT=_F(RELATION='MFRONT',  
                                UNITE_LIBRAIRIE=11,  
                                NOM_ROUTINE='asterlemaitre',),  
                 )
```

UNITE_LIBRAIRIE=11 doit être cohérent avec la valeur utilisée dans `CREA_LIB_MFRONT` pour construire la bibliothèque.

NOM_ROUTINE='asterlemaitre' permet de spécifier la loi de comportement choisie, ce qui est utile si plusieurs lois ont été compilées ensemble. Ce nom est construit par concaténation de « aster » et du nom défini par `@Behaviour` dans le fichier MFront.

Il est possible d'ajouter sous `COMPOTEMENT`, dans le cas où `RELATION='MFRONT'` les mots-clés suivants (cf. [U4.51.11]) :

- `ITER_INTE_MAXI` (100 par défaut)
- `RESI_INTE_MAXI` (1.E-8 par défaut)
- `ITER_INTE_PAS` (0 par défaut)
- `DEFORMATION` (`PETIT` par défaut) on peut utiliser `GDEF_LOG` pour tous les comportements écrits en petites déformations et `SIMO_MIEHE` pour certains comportements écrits spécifiquement avec cette formulation.

Le reste du fichier de commandes est inchangé. Par exemple, si on utilise la loi de Lemaître définie dans `Lemaitre.mfront` dans le test `ssna01a`, on peut comparer les résultats avec ceux obtenus par la loi de `LEMAITRE` de **code_aster**.

Remarque :

Si l'on a compilé la bibliothèque MFront à la main, il remplace le mot-clé `UNITE_LIBRAIRIE` par `LIBRAIRIE` et fournir le nom du fichier dans le répertoire d'exécution (cf. [U4.51.11]).

1.4 Modélisations disponibles

Les lois de comportement MFront sont utilisables nativement en 3D (et ses dérivés comme `3D_SI` et `3D_INCO_*`), en déformations planes (`D_PLAN_*`) et en axisymétrique (`AXIS_*`). Pour le cas des contraintes planes (`C_PLAN`), il y a deux situations :

- soit on utilise le mode contraintes planes natif de MFront en sélectionnant `ALGO_CPLAN='ANALYTIQUE'` dans le mot-clé facteur `COMPOTEMENT`. Dans ce cas, le fichier MFront devra utiliser, suivant la valeur de `@ModellingHypothesis`, l'intégrateur adéquat ;
- soit on utilise l'algorithme de De Borst (voir [R5.03.03]) avec `ALGO_CPLAN='DEBORST'` dans le mot-clé facteur `COMPOTEMENT`. Dans ce cas, le fichier MFront devra pouvoir utiliser une modélisation de type axisymétrique.

Pour les comportements unidimensionnels (`TUYAU_*`, `POU_D_EM` et `POU_D_TGM`), il faut nécessairement utiliser l'algorithme de De Borst car MFront ne supporte pas ces modélisations.

1.5 Modèles de grandes déformations

Tous les modèles de grandes déformations existants dans `code_aster` sont utilisables avec des comportements MFront, en particulier '`PETIT_REAC`', '`GROT_GDEP`', '`SIMO_MIEHE`' et '`GDEF_LOG`'.

Les modèles `PETIT_REAC` et `GDEF_LOG` de `code_aster` s'appuient nécessairement sur une loi écrite en petites déformations, il n'y a donc rien à faire de particulier dans le fichier MFront.

Pour `SIMO_MIEHE` et `GROT_GDEP`, il faudra choisir dans le fichier MFront une des instructions suivantes, selon le modèle utilisé dans la loi :

```
@AsterFiniteStrainFormulation[aster] SIMO_MIEHE
```

ou

```
@AsterFiniteStrainFormulation[aster] GROT_GDEP
```

Remarque s :

Il y a deux modes de fonctionnement du modèle `GROT_GDEP` dans `code_aster`.

Quand le modèle de comportement est écrit en **petites déformations** alors la structure peut subir **uniquement des grandes rotations et des grands déplacements** (c'est le cas pour la totalité des lois natives au format `code_aster` sauf `ELAS_HYPER` et des lois MFront qui ne font pas usage de `@AsterFiniteStrainFormulation`).

En revanche, quand on utilise la loi `ELAS_HYPER` ou une loi MFront qui fait usage de `@AsterFiniteStrainFormulation[aster]` `GROT_GDEP` , alors le comportement est résolu en **grandes déformations** .

2 Exemples d'utilisation

D'autres exemples sont fournis dans le tutoriel disponible sur le site internet de MFront (<http://tfel.sourceforge.net/documentations.html>).

Les cas-tests `mfron*` permettent de vérifier le mot clé `RELATION='MFRONT'` , avec un fichier MFront fourni en donnée du test.

3 Références

- [1] T. Helfer, J.-M. Proix. *Écriture de lois de comportement avec MFront : tutoriel*. Décembre 2014. <http://tfel.sourceforge.net/documents/tutoriel/tutoriel.pdf>