DE LA RECHERCHE À L'INDUSTRIE

Présentation de `MTest`

— T. Helfer

www.cea.fr

Sommaire

Présentation

- test unitaire de lois de comportement ;
- un utilitaire nommé `mtest` ;
- une librairie C++ `libTFELMtest.so` ;
- un module `python` nommé `mtest` (dans le package mfront) ;

Exemple

```
/*!
 * \file   tvergaard.mtest
 * \brief
 * \author Helfer Thomas
 * \brief  09 avril 2013
 */

@Behaviour<umat> 'src/libUmatBehaviour.so' 'umattvergaard';
@MaterialProperty<constant> 'NormalStiffness'              2.e16;
@MaterialProperty<constant> 'TangentialStiffness'          2.e16;
@MaterialProperty<constant> 'MassDensity'                  0.;
@MaterialProperty<constant> 'NormalThermalExpansion'       0.;
@MaterialProperty<constant> 'MaximumStress'              200.e6;
@MaterialProperty<constant> 'MaximumOpeningDisplacement'   3.e-6;

@ExternalStateVariable 'Temperature' 293.15;

@ImposedOpeningDisplacement 'Un' {0.:0.,1800.:1.5e-6,
    2400.:0.,2600:-1.e-8,3000:0.,3600.:3.e-6};

@Times {0.,1800 in 10, 2400,
    2600 in 5, 3000 in 1,3600 in 20};
```

- un fichier simple (!)
- `mtest tvergaard.mtest`

Fonctionnalités

- possibilité de piloter en contraintes ou/en déformations ou de manière mixte ;
- l'algorithme de résolution peut largement être paramétré :
  - algorithme d'accélération de `Cast3M` (indispensable ? pour l'interface `umat`) ;
  - matrice de prédiction, matrice tangente cohérente (interface `Aster`) ;
  - sous-découpage du pas de temps ;
  - etc...
- possibilité de comparer les résultats à une solution analytique ou des fichiers de références (non régression) ;
- les lois `mfront` peuvent générer des fichiers `mtest` en cas de non convergence (mais ça a un coût !)

Python

Exemple de code python

```python
import std
import tfel.tests
import tfel.math
from mfront.mtest import *

m = MTest()

umax = 3.e-6
m.setBehaviour("umat","src/libUmatBehaviour.so","umattvergaard")
m.setMaterialProperty('NormalStiffness',2.e16)
m.setMaterialProperty('TangentialStiffness',2.e16)
m.setMaterialProperty('MassDensity',0.)
m.setMaterialProperty('NormalThermalExpansion',0.)
m.setMaterialProperty('MaximumStress',200.e6)
m.setMaterialProperty('MaximumOpeningDisplacement',3.e-6)

m.setExternalStateVariable("Temperature",293.15)

# Attention, pour pouvoir modifier l'evolution du chargement, il faut
# le declarer avec un dictionnaire !
m.setImposedOpeningDisplacement('Un',{0.:0.})

s  = MTestCurrentState()
wk = MTestWorkSpace()

m.setOutputFileName("castemtvergaard3.res")
m.completeInitialisation()
m.initializeCurrentState(s)
m.initializeWorkSpace(wk)

t = [3.6*i for i in range(0,1001)]
m.printOutput(t[0],s)

# do the job
for i in range(0,len(t)-1):
    m.setEvolutionValue('Un',t[i+1],umax*t[i+1]/t[-1])
    m.execute(s,wk,t[i],t[i+1])
    print s.s0
    m.printOutput(t[i+1],s)
```

- Possibilité de piloter « finement » le chargement :
  - tube en pression interne asservie pour avoir une vitesse de déformation diamétrale constante.
- Peut servir à intégrer `mtest` dans un code de recalage des paramètres d'une loi :
  - voir les perspectives ;

Perspectives

Conclusions

- `mtest` a été initialement développé pour introduire des tests de non régression des lois de comportement mécanique générées par `mfront` :
  - plus de 80 tests en gestion de configuration
- un outil bien utile ;

Perspectives

- recalage de paramètres :