Notice d'utilisation du couplage entre *Code_Aster* et les modules de lois de comportement Zmat et UMAT

Résumé :

On décrit ici l'utilisation du couplage entre *Code_Aster* et les modules d'intégration de lois de comportements :

- ZMAT, du code Zebulon, de l'ENSMP.
- UMAT, routine « utilisateur » dont les arguments sont spécifiés par le code Abaqus.

Attention : l'utilisation de ces lois de comportement « à façon » implique une validation spécifique pour l'étude envisagée, car on se place hors du domaine qualifié de *Code_Aster*..

Il est également possible d'utiliser l'interface avec UMAT pour des comportements générés par Mfront.

Table des Matières

1 Modalités d'utilisation de Zmat

1.1 Description de la bibliothèque Zmat

La bibliothèque Zmat peut être appelée pour l'intégration des relations de comportement depuis *Code_Aster* de manière transparente pour l'utilisateur.

Zmat contient un module d'intégration de lois de comportement et une bibliothèque de comportements comprenant l'élasto-visco-plasticité, l'endommagement, les modèles multi-échelle …

De plus, l'utilisateur peut construire son propre comportement, en décrivant les équations régissant ce comportement à l'aide d'un langage de haut niveau (Zebfront), l'intégration étant ensuite réalisée par Zmat.

1.2 Licence et droits d'accès

Les droits d'utilisation de Zmat ont été acquis pour l'utilisation interne EDF. Hors de ce cadre, la licence de Zmat peut être acquise auprès du Centre des Matériaux de l'ENSMP.

Pour autant Zmat n'est pas un pré-requis de *Code_Aster* , on peut construire et utiliser *Code_Aster* avec ou sans la bibliothèque Zmat.

La maintenance corrective et évolutive (en particulier par rapport à l'évolution des systèmes d'exploitation, et des compilateurs) de la bibliothèque Zmat relève de l'ENSMP. Les routines d'interface sont maintenues par l'équipe de développement de *Code_Aster* .

1.3 Machines utilisables à EDF R&D

Zmat est physiquement installé, donc utilisable, sur le serveur de développement centralisé de *Code_Aster.* Si une installation spécifique sur une machine locale est réalisée (ce qui n'est pas fait par défaut sur les machines locales EDF R&D), l'utilisation de Zmat sur cette machine *nécessite* l'accès à la licence d'utilisation sur la machine centralisée.

1.4 Documentation

La documentation de Zmat est disponible sous forme de fichiers au format pdf, dans le répertoire d'installation de Zmat, en général `[ASTER_ROOT]/public/Z8.*/HANDBOOK` où `[ASTER_ROOT]` est le répertoire principal de l'installation de *Code_Aster.* Sur le serveur Aster, la documentation de Zmat est dans le répertoire `/aster/local/Z8.*/HANDBOOK` .

Les documents relatifs à Zmat sont `z_mat_manual.pdf` et `devel_manual.pdf` ainsi que `examples_manual.pdf`.

1.5 Précautions d'emploi

L'interface (et le module Zmat) prévoit les variables de commande : température, fluence, corrosion, séchage, hydratation.

Les comportements sont accessibles en petites déformations (mot clé `DEFORMATION='PETIT'`) ou en grandes déformations (`DEFORMATION='GDEF_HYPO_ELAS'`). Il faut dans ce dernier cas préciser dans le fichier de données Zmat, `***behavior gen_evp lagrange_rotate` (ou `lagrange_polar`).

Remarque sur les performances

L'intégration du comportement en utilisant la bibliothèque Zmat peut prendre un peu plus de temps que l'intégration du modèle équivalent inclus à *Code_Aster* (dans une proportion variant en général entre 1 et 2).

Précaution d'emploi

L'utilisateur définit certes son comportement et les valeurs des coefficients matériau dans le fichier Zmat, mais il doit aussi définir un matériau élastique (E et ν) s'il veut utiliser la matrice élastique ou la prédiction élastique. Il faut alors prendre garde à fournir des valeurs cohérentes avec celles du fichiers Zmat. De même, pour les commandes de post-traitement (CALC_CHAMP , POST_ELEM), certaines options utilisent les coefficients élastiques : dans ce cas, même remarque que précédemment.

Tel que réalisée actuellement, l'interface Code_Aster-Zmat ne permet pas d'établir un lien entre les matériaux gérés par Zmat (et définis dans le ficher de données Zmat) et ceux gérés par Code_Aster (en particulier le catalogue matériau).

2 Utiliser un comportement existant de la bibliothèque Zmat

2.1 Contenu du ficher Zmat

Le ficher Zmat permet à la fois de définir le comportement de fournir les valeurs des paramètres (constants ou variant avec la température) pour ce comportement. La syntaxe de ce fichier est décrite dans le document z_mat_manual.pdf.

Le fichier suivant est celui utilisé dans le cas-test zmat001a :

```
% fichier zmat001a.33
***material
*integration theta_method_a 1. 1.e-9 50
***behavior gen_evp
**elasticity
young 145200.
poisson 0.3
**potential gen_evp
*flow plasticity
*criterion mises
*kinematic nonlinear
C  63767.
D 341.
*isotropic nonlinear
R0  87.
b 2.3
Q 64.
***return
```

Ceci correspond exactement au modèle VISC_CIN1_CHAB de *Code_Aster* (extrait du cas-test ssnv101d) :

```
ACIER1=DEFI_MATERIAU(ELAS=_F(E     = 145200.,
                             NU    = 0.3,),
                     CIN1_CHAB=_F(R_I = 151.,
                                  R_0 = 87.,
                                  B   = 2.3,
                                  C_I = 63767.,
                                  G_0 = 341.,))
```

2.2 Particularités du ficher de commandes et du profil d'étude de *Code_Aster*

L'utilisation de Zmat se traduit pour l'utilisateur de *Code_Aster* de la façon suivante :

1. Sous le mot-clé COMP_INCR de STAT_NON_LINE , il faut préciser RELATION='ZMAT' , pour aller lire le fichier contenant les données Zmat. Ce fichier permet de décrire les paramètres matériau.
2. Toujours sous COMP_INCR , un mot-clé UNITE permet de définir l'unité logique sur laquelle ont vient lire le ficher Zmat et bien sûr les mots clés habituels : GROUP_MA , DEFORMATION (PETIT, ou GDEF_HYPO_ELAS),
3. Le mot-clé NB_VARI (sous COMP_INCR) permet de préciser le nombre de variables internes du comportement. Le nombre de variables internes peut être plus grand que nécessaire (le stockage des variables internes occupera plus de place que nécessaire en mémoire), mais bien sûr pas inférieur au nombre réellement utilisé (Zmat fourni alors un message d'erreur). Dans le ficher de message, au début de l'exécution de la commande STAT_NON_LINE ou DYNA_NON_LINE, Zmat précise le nombre de variables internes nécessaires, et leurs significations. Il est utile de prendre en compte ces informations, en particulier pour le dépouillement des résultats. (Le nombre de variables internes peut aussi être calculé grâce à l'utilitaire Zpreload, dont un exemple d'appel via python est donné dans le test zmat001a).
4. Dans astk, par rapport à une étude classique, il suffit d'ajouter le fichier Zmat et y associer le numéro d'unité logique défini au mot-clé UNITE .
5. Dans le cas d'un calcul STAT/DYNA_NON_LINE en poursuite, il faut penser à fournir à nouveau le ficher Zmat dans le profil d'étude. Le test zmat003a permet d'illustrer ce point.

2.3 Exemples

Les cas-tests zmat001 à zmat010 valident le couplage entre *Code_Aster* et Zmat, mais constituent également un base d'exemples de mise en œuvre.

1. zmat001 : test de traction-cisaillement avec une loi de Chaboche (analogue au test ssnv101 avec VISC_CIN1_CHAB) en déformations planes (modélisation A) et 3D (modélisations B et C). On y trouve également une possibilité d'utiliser Zpreload pour récupérer le nombre de variables internes du comportement Zmat.
2. zmat002 : thermo-plasticité en traction simple (VMIS_ISOT_TRAC), modélisation A en axisymétrique, modélisation B en contraintes planes. L'intérêt de ce test est de décrire la variation des coefficients avec la température.
3. zmat003 : test de traction-cisaillement utilisant une loi définie dans le fichier zmat003a.32 (voir paragraphe suivant). Ce test valide également l'utilisation en POURSUITE .
4. zmat004 : test de comparaison entre le modèle MONOCRISTAL et son analogue dans Zmat.
5. zmat005 : test de comparaison entre les grandes déformations GDEF_HYPO_ELAS et les grandes déformations SIMO_MIEHE (hsnv121) en thermo-élastoplasticité de Von Mises à écrouissage isotrope (ou cinématique linéaire).
6. zmat006 : test de validation de Zmat en contraintes planes (méthode De Borst).
7. zmat007 : test de comparaison entre le modèle MONOCRISTAL et son analogue dans Zmat, pour un agrégat comportant 10 grains, d'orientations diverses.
8. zmat008 : test de tractions-rotations multiples, pour valider la prise en compte des rotations finies pour les grandes déformations GDEF_HYPO_ELAS.
9. zmat009 : test de cisaillement simple pour valider sur une solution analytique les grandes déformations GDEF_HYPO_ELAS.
10. zmat010 : test de comparaison entre l'élasticité cubique de Zmat et ELAS_ORTH de Code_Aster.

3 Définir un nouveau comportement dans Zmat

Le nouveau comportement est défini dans un fichier qui sera compilé, indépendamment de *Code_Aster*, de façon à produire une bibliothèque dynamique, utilisée par la bibliothèque Zmat principale au moment de l'exécution.

Ni le fichier « source », ni la bibliothèque construite n'ont besoin de figurer dans le profil astk.

3.1 Ecrire un nouveau comportement dans Zmat

L'écriture d'un nouveau comportement se fait à l'aide d'un langage de haut niveau (Zebfront, permettant des opérations tensorielles …) sur base de C++. On se reportera à la documentation de Zmat pour la description de la syntaxe : `devel_manual.pdf`.

Avertissement

> *Zmat impose que le nom de ce fichier commence par une majuscule et ait l'extension ' `.z` ' pendant la phase de compilation.*

Voici le fichier (nommé `Chab1.z` pour les exemples suivants) définissant le comportement `chab1` utilisé dans le test `zmat003` (fichier `zmat003a.32`), équivalent du comportement `VISC_CIN1_CHAB` de *Code_Aster* :

```
#include <Elasticity.h>
#include <External_parameter.h>
#include <Basic_nl_behavior.h>
#include <Basic_nl_simulation.h>
#include <Flow.h>
#include <Criterion.h>
#include <Isotropic.h>
#include <Print.h>

@Class CHAB1 : BASIC_NL_BEHAVIOR {
  @Name chab1;
  @SubClass ELASTICITY E;
  @Coefs R0, Q, b;
  @Coefs K, n, C, D;
  @tVarInt eel, alpha;
  @sVarInt evcum;
  @tVarAux evi, X;
  @sVarAux R;
};

@StrainPart {
 evi = eto - eel;
 sig = *E*eel;
 if (m_flags&CALC_TG_MATRIX) m_tg_matrix=*E;
}

@Derivative {
 sig = *E*eel;
 X = (2.0*C/3.0)*alpha;
  R = R0 + Q*(1.-exp(-b*evcum));

 TENSOR2 sigeff = deviator(sig - X);
 double J = sqrt(1.5*(sigeff|sigeff));
 double f = J - R  ;

 deel = deto;
```

```
 if (f>0.0) {
           devcum = pow(f/K,n);
   TENSOR2 norm = sigeff*(1.5/J);
           deel -= devcum*norm;
   if (C>0.0) dalpha = devcum*(norm - D*alpha);
 }
}
```

3.2 Compilation du nouveau comportement

On décrit ci-après les différentes étapes à suivre pour construire la librairie dynamique qui sera utilisée par Zmat pour intégrer le comportement utilisateur.

Remarque préliminaire

La librairie dynamique sera utilisée par Zmat sur la machine d'exécution du calcul *Code_Aster*, elle doit donc être compilée sur cette même machine pour éviter les problèmes d'hétérogénéité des bibliothèques.

•Organisation des fichiers (en prenant comme base `/home/user/repertoire`) :

```
/home/user/repertoire    /   library_files
                         /   material      /
                          /  material      /  Chab1.z
```

1) Créer un fichier `library_files` contenant :

```
!MESSAGE User Z7 project
!TOP Makefile.Motif.c++

!DYNAMIC
#!INSTALL_LIBS
!CFLAGS -I${Z7PATH}/include
!BFLAGS -L${Z7PATH}/PUBLIC/lib-${Z7MACHINE}

!MAKE target: lib

#
#  The main place for user f iles.
#
!INC   material
!SRC   material material

!DEBUG material

!LIBLIB -ZL Zmat_base
!LIB Zmat_ut_TyPeMaCHiNe material

!!RETURN
```

Le fichier `.z` sera pris dans le sous-répertoire `material`

La librairie dynamique sera nommée `libZmat_ut_Linux4.so` sous Linux, `libZmat_ut_OSF1.so sous TRU64` , etc.

2) Charger les fichiers d'environnement nécessaires pour Zmat. Le plus simple est de charger tous les fichiers nécessaires à la construction de la version de *Code_Aster* incluant Zmat.
Il y a donc le fichier principal /aster/etc/codeaster/profile.sh (remplacer `/aster` par `/opt/aster` ou autre selon l'installation) et les fichiers d'environnement supplémentaires déclarés dans le fichier de configuration de la version (par exemple `/aster/STAxx/config.txt`, lignes `ENV_SH`).

Ce qui donnera sur le serveur de référence :

```
prompt> . /aster/etc/codeaster/profile.sh
prompt> . /aster/agla/tool/init_zmat.sh
```

3) Création du `Makefile` Zmat :

```
prompt> Zsetup
```

4) Compilation du comportement déposé dans le répertoire `material`

```
prompt> Zmake
```

3.3 Utilisation du comportement développé par l'utilisateur

Dans le profil d'étude astk, on fournit comme précédemment un fichier contenant la valeur des paramètres pour le comportement utilisateur. Le nom `chab1` est important et doit correspondre au nom fourni à l'attribut `@Name` du fichier `.z` (voici le fichier `zmat003a.33`) :

```
***behavior chab1
**YE isotropic
young 195000.
poisson 0.3
**model_coef
K 600.0
n 3.5
C 0.0
D 0.0
R0 30.0
Q 270.
b 100.
***return
```

On fournit le fichier de commandes dans lequel il faut définir la variable d'environnement `ZEBU_PATH` pour indiquer à Zmat où se trouve la bibliothèque dynamique que l'on vient de compiler. On retrouve alors dans le fichier (extraits de `zmat003a.comm`) :

```
# définition de la variable ZEBU_PATH
import os
os.environ['ZEBU_PATH'] = '/home/user/repertoire'

[...]

# définition des propriétés élastiques du matériau
# qui doivent être cohérentes avec celle du fichier Zmat
ACIER=DEFI_MATERIAU(ELAS=_F(E = 195000.,
                            NU = 0.3),)

[...]

# utilisation du comportement dans le calcul non linéaire
# UNITE doit correspondre à la valeur définie dans astk
CALCNL=STAT_NON_LINE(...,
                     COMP_INCR=_F(RELATION = 'ZMAT',
                                  UNITE = 33,
                                  NB_VARI = 26,
                                  DEFORMATION = 'PETIT'),
                     ..., )
```

4 Modalités d'utilisation des routines de type UMAT

4.1 Description des routines UMAT

UMAT est un format de routine *fortran* familier aux utilisateurs du code Abaqus, servant à intégrer leurs propres lois de comportement. Le contenu de cette routine est entièrement libre et à la charge de l'utilisateur, et doit permettre d'intégrer la loi de comportement, c'est à dire en un point d'intégration, de calculer le tenseur des contraintes, les variables internes, et l'opérateur tangent (cf. [D5.04.01] - Introduire un nouveau comportement).

Pour plus de précisions sur les routines UMAT, consulter la documentation du Code Abaqus.

L'entête d'une routine UMAT se présente comme ceci :

```
SUBROUTINE UMAT( STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
                 RPL,DDSDDT,DRPLDE,DRPLDT,
                 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
                 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
                 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC)
```

Brièvement, les arguments en entrée et en sortie d'un routine UMAT sont les suivants :

4.1.1 Variables en entrée

Argument	Signification (spécification Abaqus)	Valeurs transmise par Code_Aster
NDI	Nombre de composantes de contraintes (hors cisaillement) au point d'intégration courant	3
NSHR	Nombre de composantes de contraintes de cisaillement au point d'intégration courant	1 (D_PLAN, AXIS, C_PLAN) ou 3 (en 3D)
NTENS	Nombre total de composantes de contraintes et de cisaillement	NTENS = NDI+NSHR
NPROPS	Nombre de paramètres matériaux	50 (valeur fixe)
NSTATV	Nombre de variables internes associées au comportement.	NB_VARI sous COMP_INCR
CMNAME	Nom du comportement	UMAT
COORDS	Tableau contenant les coordonnées du point d'intégration courant	*Non défini*
PROPS(NPROPS)	Tableau des paramètres matériaux	C1 à Cxx dans DEFI_MATERIAU
TIME(1)	Temps du pas au début de l'incrément	$\Delta t = t_i - t_{i-1}$
TIME(2)	Temps total au début de l'incrément	t_{i-1}
DTIME	Incrément de temps	$\Delta t = t_i - t_{i-1}$

TEMP	Température au début de l'incrément	$T(t_{i-1})$
DTEMP	Incrément de température	$T(t_i)-T(t_{i-1})$
CELENT	Longueur caractéristique de l'élément	*Non défini*
NOEL	Numéro de l'élément	*Non défini*
NPT	Numéro du point d'intégration	numéro du point de Gauss dans l'élément courant
LAYER	Numéro de la couche pour les coques (sous-point d'intégration)	*Non défini*
KSPT	Numéro du point d'intégration dans la sous-couche courante	Numéro de sous-point d'intégration
KSTEP	Numéro du pas	numéro du sous-pas en cas de re-découpage local
KINC	Numéro de l'incrément	*Non défini*
DROT(3,3)	Matrice d'incrément de rotation, présente si la base pour le matériau tourne avec l'élément.	Calculée à partir des 3 angles nautiques
DFGRD0(3,3)	Tableau contenant le gradient de transformation au début de l'incrément	*Non défini*
DFGRD1(3,3)	Tableau contenant le gradient de transformation à la fin de l'incrément	*Non défini*
STRAN(NTENS)	déformations mécaniques totales au début de l'incrément. (les déformations thermiques sont retirées).	$\varepsilon(t_{i-1})-\varepsilon^{th}(t_{i-1})$
DSTRAN(NTENS)	incréments de déformations mécaniques. Les déformations thermiques sont retirées.	$\Delta\varepsilon(t_{i-1})-\Delta\varepsilon^{th}(t_{i-1})$
PREDEF	Tableau des valeurs interpolées des champs externes imposés au début de l'incrément	Les valeurs fournies sont : 'IRRA','SECH','HYDR','CORR', 'NEUT1','NEUT2', dans cet ordre. (la valeur est NaN si la variable de commande n'est pas définie).
DPRED	Tableau des incréments des champs externes imposes	Incréments de variable de commande dans le même ordre que PREDEF.

4.1.2 Variables produites ou modifiées

Argument	Signification (spécification Abaqus)	Valeurs transmise à Code_Aster
STRESS(NTENS)	Tenseur des contraintes de Cauchy au début du pas de temps. Doit être modifié pour donner les contraintes à la fin du pas de temps.	Stockées dans le champ SIEF_ELGA, utilisées pour le calcul des forces internes (résidu)

STATEV(NSTATV)	Tableau contenant les variables internes de la loi de comportement.	Stockées dans le champ VARI_ELGA
DDSDDE(NTENS,NTENS)	Opérateur tangent (symétrique) du modèle de comportement	Utilisé pour le calcul de la matrice tangente.
PNEWDT	Rapport du nouveau pas de temps suggéré sur le pas de temps initial	Si PNEWDT < 1, on tente un re-découpage du pas de temps
SSE, SPD, SCD	Énergie élastique, de dissipation plastique et énergie de fluage.	Non utilisé

Pour les analyses thermo-mécaniques couplées uniquement :

RPL	Production de chaleur volumétrique par unité de temps à la fin de l'incrément provoquée par la mécanique.	Non utilisé
DDSDDT(NTENS)	Variation de l'incrément des contraintes par rapport à la température	Non utilisé
DRPLDE(NTENS)	Variation de la production de chaleur par rapport aux incréments de déformation	Non utilisé
DRPLDT	Variation de la production de chaleur par rapport à la température	Non utilisé

Les limitations actuelles de l'interface Aster-Umat sont :
- Petites déformations, provisoirement (on ne peut prendre en compte actuellement les grandes déformations que de façon approchée, avec toutes les limitations évoquées dans [R5.03.21] et [R5.03.24], via DEFORMATION='PETIT_REAC')
- Les modélisations supportées sont : 3D, AXIS, D_PLAN (et C_PLAN via DEBORST)
- Les énergies ne sont pas récupérées par *Code_Aster* actuellement ;
- Pas de couplage thermo-mécanique pour le moment.

4.2 Utilisation des routines UMAT dans une étude

4.2.1 Création de la bibliothèque dynamique à partir d'une routine fortran umat.f

La bibliothèque dynamique contenant la routine UMAT doit être préparée avant l'exécution du calcul. Pour cela, l'utilisateur dispose d'un moyen simple de compiler cette bibliothèque en utilisant l'utilitaire as_run [U1.04.00], (avec une version d'astk supérieure ou égale à 1.8.3).

Le fonctionnement est le suivant :

1) l'utilisateur écrit sa routine de comportement umat.f (les noms du fichier et de la routine sont quelconques puisqu'ils sont rappelés au moment du calcul) ;
2) il produit la bibliothèque dynamique associée à ce comportement de la façon suivante :
   - cd /chemin/fichiers/etude
   - as_run --make_shared -o libumat.so umat.f
1) pour lancer l'étude, dans astk, il indique :
   - /chemin/fichiers/etude/libumat.so **type="nom"** , UL=0, en Donnée (le fichier sera recopié dans le répertoire de travail sans changer de nom)
   - dans le .comm, il indique sous COMP_INCR :

- RELATION='UMAT',
- LIBRAIRIE='libumat.so',
  (ou le chemin complet : LIBRAIRIE='/chemin/fichiers/etude/libumat.so')
- NOM_ROUTINE='umat_' (on peut omettre le caractère « _ »)

Par ce mécanisme, il est tout à fait possible d'avoir plusieurs comportements UMAT différents (plusieurs routines dans une même bibliothèque ou des bibliothèques différentes, voir par exemple le test umat002a).

4.2.2 Utilisation dans le fichier de commandes

Le couplage UMAT / Code_Aster se traduit dans le fichier de commandes de la façon suivante :

- Les données nécessaires du champ matériau sont fournies dans l'opérateur DEFI_MATERIAU [U4.43.01], sous le mot clé UMAT / UMAT_FO .
- Sous le mot-clé COMP_INCR de STAT_NON_LINE, DYNA_NON_LINE, ou SIMU_POINT_MAT :
  - spécifier RELATION='UMAT' ;
  - sous le mot_clé NB_VARI, préciser le nombre de variables internes du comportement ;
  - pour prendre en compte l'hypothèse des contraintes planes, utiliser ALGO_C_PLAN="DEBORST" ;
  - indiquer le chemin vers la bibliothèque sous le mot-clé LIBRAIRIE et le nom de la routine contenue dans la bibliothèque sous le mot-clé NOM_ROUTINE, comme décrit ci-dessus.
- les mot-clés : RESI_INTE_RELA, ITER_INTE_MAXI, RESO_INTE, PARM_THETA, ne sont pas utilisés, car les arguments d'entrée de UMAT en permettent pas de les prendre en compte.

Exemples : voir les tests umat001 (test thermo-élastique avec STAT_NON_LINE), umat002 (test analytique multi-directionnel avec SIMU_POINT_MAT).

4.3 Utilisation des comportements MFront dans une étude

4.3.1 Licence et droits d'accès

MFRONT est un logiciel permettant d'intégrer des lois de comportement, développé par la CEA/Cadarache dans le cadre de PLEIADES (cf. CR-AMA-13.049.pdf). Son usage est donc actuellement restreint à EDF/R&D et il est accessible seulement en interne EDF pour des tests car l'utilisation d'une loi de comportement par ce mécanisme (pour une étude de R&D ou autre) est actuellement hors du périmètre assurance qualité.

MFront est physiquement installé, donc utilisable, sur le serveur de développement centralisé de *Code_Aster* et sur certaines machines locales Calibre7 de EDF/R&D.

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

La bibliothèque dynamique contenant le comportement MFront doit être préparée avant l'exécution du calcul. Pour cela, l'utilisateur dispose d'un moyen simple de compiler cette bibliothèque en utilisant mfront (cf. CR-AMA-13.049.pdf).

Le fonctionnement est le suivant :

- l'utilisateur écrit sa routine de comportement maloi.mfront ;
- il produit la bibliothèque dynamique associée à ce comportement de la façon suivante :
  - mfront --obuild maloi.mfront --interface=aster

4.3.3 Utilisation dans le fichier de commandes

L'utilisation du comportement Mfront s'effectue de la façon suivante :

- dans astk :/chemin/fichiers/etude/libmfront.so **type="nom"** ,UL=0, en Donnée ;
- Les données matériau sont fournies dans DEFI_MATERIAU [U4.43.01], sous UMAT / UMAT_FO .
- Sous le mot-clé COMP_INCR de STAT_NON_LINE, DYNA_NON_LINE, ou SIMU_POINT_MAT :
  - RELATION='MFRONT',
  - LIBRAIRIE='libxxx.so',

- NOM_ROUTINE='astermaloi'
- sous le mot_clé NB_VARI, préciser le nombre de variables internes du comportement ;

- pour prendre en compte l'hypothèse des contraintes planes, utiliser ALGO_C_PLAN="DEBORST" ;
- Exemples : voir les tests MFRON01, MFRON02.