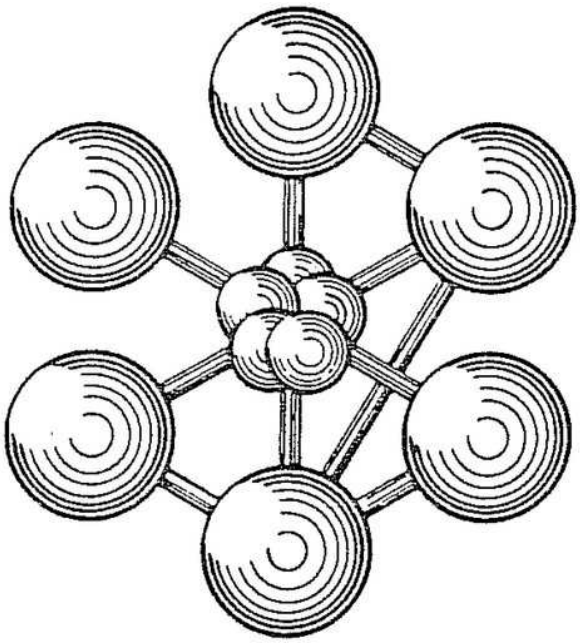
DÉVELOPPER CAST3M

T. CHARRAS, J. KICHENIN

DÉVELOPPER DANS CAST3M



ÉDITION 2011

Cast3M est un logiciel de calcul par la méthode des éléments finis pour la mécanique des structures et des fluides. Cast3M est développé au Département de Modélisation des Systèmes et Structures (DM2S) de la Direction de l'Énergie Nucléaire du Commissariat à l'Énergie Atomique et aux Énergies Alternatives (CEA).

Le développement de Cast3M entre dans le cadre d'une activité de recherche dans le domaine de la mécanique dont le but est de définir un instrument de haut niveau, pouvant servir de support pour la conception, le dimensionnement et l'analyse de structures et de composants.

Dans cette optique, Cast3M intègre non seulement les processus de résolution (solveur) mais également les fonctions de construction du modèle (pré-processeur) et d'exploitation des résultats (post-traitement). Cast3M est un logiciel « boîte à outils » qui permet à l'utilisateur de développer des fonctions répondant à ses propres besoins.

Cast3M est notamment utilisé dans le secteur de l'énergie nucléaire, comme outil de simulation ou comme plateforme de développement d'applications spécialisées. En particulier, Cast3M est utilisé par l'Institut de Radioprotection et de Sûreté Nucléaire (IRSN) dans le cadre des analyses de sûreté des installations nucléaires françaises.

Table des matières

Chapitre 1

Recommandations

1.1 Introduction

Nous allons préciser dans cette note quelques informations utiles aux développeurs du logiciel Cast3M. La lecture préalable des sections portant sur le langage ESOPE, sur les objets de Cast3M et sur les opérateurs de Cast3M est vivement recommandée. Certaines des recommandations auront en fait un caractère obligatoire, d'autres sont juste des règles de bonnes conduites qui malheureusement n'ont pas toujours été suivies. Ces recommandations sont là pour assurer la qualité et le portage de Cast3M sur tous types de machine.

Le langage utilisé pour développer Cast3M est ESOPE qui est lui-même une extension de fortran. La dernière étape du développement est la fourniture d'un cas test. Nous nous excusons pour le coté rébarbatif de cette liste et nous vous souhaitons bonne lecture.

1.2 Liste des recommandations

Les opérateurs : La pertinence d'un opérateur et sa syntaxe sont des éléments primordiaux du logiciel Cast3M. Il est très vivement recommandé d'en discuter avec l'équipe de développement du logiciel avant de se lancer dans la programmation.

Les variables : Elles ont six lettres au maximum et évite les mots clés du fortran (do, if, ...). Elles respectent la carte : IMPLICIT REAL*8(A-H,O-Z) qui doit être systématiquement placée en tête de chaque subroutine.

Les fonctions génériques : Il faut utiliser, quand cela est possible, les fonctions génériques du fortran 77, c'est-à-dire les fonctions qui ne présupposent pas le type de la variable passée en argument. Par exemple, pour calculer le cosinus de xva (double précision) utiliser COS(XVA) et non pas DCOS(XVA). Ceci facilite le portage sur différentes machines. Les principales sont :
max, min, mod
acos, asin, atan, atan2
cos, sin, tan
log, exp, erf
int, real, nint, sign
sqrt

Les structures de données : Les COMMON fortran et les déclarations de SEGMENT apparaissant dans plusieurs opérateurs ont des déclarations identiques. Ces déclarations doivent être isolées dans des fichiers séparés et incluses à l'aide d'une instruction –INC ...

L'usage du COMMON interne à un opérateur devient interdit pour des raisons de parallélisation du logiciel.

Les structures de données de type SEGMENT doivent être systématiquement supprimées en fin de travail ou au moins elles doivent être désactivées. L'opérateur « TEMPS » suivi de l'option « place » renseigne sur l'état des segments, nombre de segments actifs, nombre total de segments. . ..

Les sous-programmes : Leur nom ne doit pas dépasser six caractères. Juste après la carte subroutine . . . ; il faut placer l'instruction implicite de type suivante : IMPLICIT REAL*8(A-H,O-Z)

Un des critères de qualité formelle de Cast3M est de vérifier l'analyse d'intercompilation. Il faut veiller à respecter le nombre et les types des arguments lors des appels de sous-programmes.

Un sous programme ne peut recevoir en argument une variable ou un tableau appartenant à une structure SEGMENT s'il désire lui-même utiliser des instructions ESOPE

Commentaires, indentation : La programmation doit être lisible et les algorithmes utilisés doivent pouvoir s'en déduire facilement. Des commentaires aident si nécessaire la compréhension.

Les structures logiques utilisées (IF,DO,..) doivent être soulignées par l'indentation.

Chaque sous programme devra posséder un commentaire précisant sa fonction et au besoin une liste des entrées/sorties devra être commentée.

Entrées Sorties : En standard, un sous-programme ne doit effectuer ni entrée ni sortie. Il doit appeler les fonctions correspondantes de GIBIANE (LIROBJ, ECROBJ, . . .) pour communiquer avec l'analyseur de syntaxe. Normalement, un opérateur travaille silencieusement, c'est à dire qu'il n'émet aucun message. Malgré tout, des impressions supplémentaires peuvent être prévues. Elles devront se faire ou non suivant la valeur de la variable IIMPI du common COPTIO, qui en standard se trouve être égale à zéro et qui peut être modifiée par l'utilisateur avec la directive : « OPTION IMPI NN ; ». Elles se font alors sur l'unité logique IOIMP. IOIMP se trouve aussi dans le common COPTIO et peut être modifié par : « OPTION IMPR 'mon.fic' ; ».

Les erreurs : La validité des structures de données utilisées sera systématiquement contrôlée. La justesse des données également. En cas d'erreur, un message d'erreur sera émis par appel au sous programme ERREUR et le contrôle est rendu au sous programme PILOT. La variable IERR située dans le common COPTIO sera systématiquement testée après chaque appel de fonction pour contrôler la validité de l'exécution.

Pour imprimer un message d'erreur, il faut appeler le sous programmes ERREUR avec en argument un numéro d'erreur. Les données supplémentaires sont transmises par les variables INTERR, REAERR et MOTERR du common COPTIO. Si le message n'existe pas il faut l'ajouter dans le fichier 'GIBI.ERREUR' en respectant les règles suivantes :

- Un message contient deux ou quatre lignes. La première (et troisième) contient le numéro de l'erreur (sur 4 caractères), un blanc et le niveau de l'erreur sur un caractère. Ce niveau est égal à 2 pour arrêter l'exécution de l'opérateur, de la boucle répéter et des procédures actives.
- La deuxième ligne (et la quatrième) contient le texte du message en format A80.
- Des informations passées par les variables INTERR, REAERR, MOTERR peuvent être écrites dans le corps du message

%i1,%i2	représentent INTERR(1), INTERR(2) . . .
%r1,%r2	représentent REAERR(1), REAERR(2) . . .
%m3 :8	représente MOTERR(3 :8)

- Pour ajouter un message d'erreur, il faut prendre un numéro positif consécutif à celui du dernier message et écrire le message dans sa version française et dans sa version anglaise.
- Il est aussi possible d'introduire des messages d'information. Pour les différencier des messages d'erreur, leur numéro est négatif et le niveau d'erreur vaut 0.

Les cas tests : Nous rappelons que la fonction d'un cas test est de vérifier la non-régression du logiciel, c'est à dire vérifier que les résultats fournis demain seront les mêmes que ceux fournis hier. Il ne s'agit en aucun cas de faire une démonstration de la validité des résultats obtenus.

Les cas tests de validation des résultats, s'ils sont très rapides d'exécution, peuvent être utilisés. Sinon, il faut en inventer d'autres ou tronquer leur exécution. De nombreux cas tests utilisent une variable COMPLET positionnée en gibiane à VRAI ou FAUX pour faire la différence entre une exécution complète ou tronquée.

Ne pas faire de dessin dans les cas test ou en contrôler l'exécution par une variable gibiane GRAPH qui est positionnée à FAUX par l'instruction gibiane GRAPH=FAUX ;

Chapitre 2

Esope

2.1 Introduction

Le présent manuel n'a pas la prétention de tout dire sur ESOPE, si le lecteur veut en savoir plus il pourra consulter avec profit les rapports :

- Manuel d'utilisation ESOPE V.10.0 par P.VERPEAUX (DMT/93.617)
- Esope 10.1 dit Esope optimisé par P.VERPEAUX (DMT/94.710)

Notre objectif est juste de décrire les fonctions d'ESOPE utilisées dans le code Cast3M. En cela ce manuel est presque complet.

2.2 Présentation générale

ESOPE est une extension du langage fortran 77, il est donc nécessaire de posséder correctement le langage fortran avant de vouloir faire de l'ESOPE. L'objectif est de faciliter la gestion des données et de permettre la notion d'objet par la structuration des données.

Le code source, écrit en ESOPE, est dans un premier temps traduit en fortran 77 avant d'être envoyé en compilation. La traduction en fortran se fait en exécutant un petit programme appelé lui aussi ESOPE.

La partie d'ESOPE qui gère le transfert des données mémoire vive <-> mémoire de stockage ne sera pas exposé ici. C'est la bibliothèque GEMAT, incluse dans ESOPE, qui s'en charge et rares sont les applications où le programmeur doit faire un travail spécial.

Pour disposer d'un langage orienté OBJET il faut associer un ensemble de données à une variable. Il manque au fortran la notion d'ensemble de données et l'entité appelée SEGMENT a été ajoutée. Celle-ci répond aux deux exigences précitées :

- c'est un regroupement de variables fortran défini par le programmeur.
- elle est référencée par une seule variable appelée POINTEUR. La connaissance du pointeur suffit pour accéder à toutes les variables contenues dans la structure de données .

2.3 Manipulation des structures de données

Peu d'instructions ont été ajoutées à celles du fortran77. Elles servent à manipuler et utiliser les SEGMENTS.

Comme en fortran, on est conduit à avoir une instruction déclarative du SEGMENT puis des instructions qui agissent sur le SEGMENT. Il s'agit avant tout de :

- créer un segment (ou INItialiser) SEGINI
- SUPprimer un segment SEGSUP
- DESactiver un segment SEGDES
- ACTiver un segment SEGACT

– ajuster la taille d'un segment SEGADJ

2.3.1 Déclaration d'une SEGMENT

Avant la première instruction exécutable d'un sous-programme fortran, il faut déclarer les SEGMENTS qui seront utilisés.

Un SEGMENT peut contenir autant de variables fortran que nécessaire et de tous les types fortran admis. La déclaration se fait comme suit :

```
SEGMENT MONSEG
  INTEGER IJK,NOMB(LL,LC),JLC
  REAL X,XX(N)
  REAL*8 Y,YY(3,MM)
  CHARACTER*NBCA ICHA(8,IK),CHAI
ENDSEGMENT
```

La déclaration est comprise entre les mots SEGMENT et ENDSEGMENT. Rappelons que nous sommes en fortran et que ces instructions doivent être frappées au delà de la septième colonne.

MONSEG est le nom de la classe de la structure matérialisée par ce segment. Une structure appartenant à la classe MONSEG contiendra un entier IJK, un tableau d'entier NOMB, un entier JLC .... et enfin une chaine de NBCA caractères appelée CHAI.

Dans cet exemple IJK,NOMB et JLC sont des noms de variables de type fortran INTEGER, X ET XX sont de type RELLES, etc... Pour des raisons d'architecture de machines il est préférable de ne pas employer DOUBLE PRECISION mais REAL*8. ICHA et CHAI sont des variables contenant des chaines de NBCA caractères.

Remarquons que certaines dimensions de tableaux sont fixées tandis que d'autres font référence à des noms de variables entières (il est d'usage dans Cast3M de respecter les types implicites du fortran). C'est les valeurs de ces variables fortran au moment de l'initialisation du segment qui définiront les dimensions prises par les tableaux. Les variables dimensionnantes sont dans notre exemple LL,LC,N,MM,IK, elles ne sont pas incluses dans le SEGMENT, ne servant qu'au moment de sa création. Une instruction particulière permettra, si besoin est, de connaitre la longueur d'un tableau dans un SEGMENT déjà crée.

2.3.2 Création et initialisation d'un SEGMENT

Plaçons nous encore à l'intérieur d'un sous-programme fortran. Après y avoir déclaré le segment par l'instruction SEGMENT...ENDSEGMENT il faut définir une variable qui référencera la structure instanciée : la connaissance de cette variable donnant accès à toute la structure. Pour cela un nouveau type de variable fortran est inventée : c'est la variable pointeur dont la déclaration est :

```
POINTEUR nompointeur.nomsegment
```

par exemple, pour permettre l'instanciation de la classe MONSEG, on déclare :

```
POINTEUR MONS1.MONSEG,MONS2.MONSEG,...
```

Dans cet exemple MONS1 et/ou MONS2 permettent de référencer une structure de la classe MONSEG. MONS1 et MONS2 deviennent des variables fortran de type Integer une fois la traduction Esope->Fortran faite.

La création d'une structure de classe MONSEG est faite par l'appel à la fonction SEGINI en précisant la variable pointeur utilisée.

```
SEGINI MONS1
```

Dans notre exemple cela donne :

```
subroutine sp1 (......)
segment monseg
  integer ijk,nomb(ll,lc),jlc
  real x,xx(n)
  real*8 y,yy(3,mm)
  character*nbca icha(8,ik),chai
endsegment
POINTEUR MONS1.MONSEG,MONS2.MONSEG
  .
  .
SEGINI MONS1
  .
  .
return
end
```

Il faut, bien évidemment, que les variables dimensionnantes du segment soient définies préalablement à l'instruction SEGINI (sinon le résultat dépend du compilateur !). Nous aurions pu faire le jeu d'instructions :

```
subroutine sp1 (LL,LC,...)
segment monseg
  integer ijk,nomb(ll,lc),jlc
  real x,xx(n)
  real*8 y,yy(3,mm)
  character*nbca icha(8,ik),chai
endsegment
pointeur mons1.monseg
  .
  .
N=5
MM=3
IK=12
NBCA=4
SEGINI MONS1
MONS1.XX(3)=...
MONS1.ICHA(5,2)='BIEN'
  .
  .
return \\
end
```

Dans lequel le tableau YY est dimensionné à 3 lignes et 3 colonnes, tandis que ICHA est un tableau de chaines de 4 caractères qui à 8 lignes et 12 colonnes.

Une fois l'instruction SEGINI exécutée et tant que l'on reste dans le sous-programme, tous les éléments du SEGMENT peuvent être considérés par le programmeur comme des variables fortran normales. Pour les atteindre il faut rappeler qu'elles appartiennent à la structure référencée par la variable pointeur MONS1.
Remarques :

– à la création, toutes les variables contenues dans le segment sont mises à zéro.

– il est possible de créer un segment en en recopiant un autre (confère chapitre 4).

ESOPE offre la possibilité de libérer la place mémoire occupée par le segment. on peut soit supprimer le segment, soit le désactiver.

2.3.3 Suppression d'un SEGMENT

L'ordre de déclaration du segment étant en tête du sous-programme on peut à tout moment exécuter l'instruction SEGSUP dont la syntaxe est :

```
SEGSUP nom-variable-pointeur
```

Dans notre exemple, supposons que la structure de données MONSEG instanciée par MONS1 ne serve qu'à l'intérieur du sous-programme SP1, alors le segment doit être supprimé avant de quitter le sous-programme.

```
subroutine sp1 (ll,lc......)
segment monseg
  integer ijk,nomb(ll,lc),jlc
  real x,xx(n)
  real*8 y,yy(3,mm)
  character*nbca icha(8,ik),chai
endsegment
pointeur mons1.monseg
  .
  .
n=5
mm=3
ik=12
nbca=4
segini mons1
mons1.xx(3)=...
mons1.icha(5,2)='BIEN'
  .
  .
SEGSUP MONS1
  .
return
end
```

2.3.4 Désactivation d'un SEGMENT

En règle générale, au sortir d'un sous-programme fortran ou dés que possible, on désactive les segments pour rendre utilisable la mémoire vive qu'ils occupent. Cela se fait par l'instruction :

```
SEGDES nom-variable-pointeur
```

La mise en attente sur disque de travail ne se fera que si les demandes en mémoire le nécessite et si la place sur disque le permet. De toutes les façons une fois un segment désactivé, il aura besoin d'être réactivé avant de s'en servir à nouveau.

2.3.5 Activation d'un SEGMENT

Supposons que le segment MONS1 ait été défini puis désactivé dans le sous-programme fortran sp1 qui appelle sp2 et que l'on désire se servir de MONS1 dans ce dernier sous-programme. Sp2 devra contenir l'ordre déclaratif de la classe du segment et de plus il devra connaitre la valeur de la variable pointeur instanciant MONSEG. On peut la lui fournir soit par COMMON soit par argument (soit par un autre segment) puisqu'elle est avant tout de type Integer. Pour activer le segment if faut utiliser l'instruction :

```
SEGACT nom-variable-pointeur
```

Cette activation se fait en lecture seule, pour pouvoir modifier les valeurs contenues dans le segment, c'est à dire avoir accès en lecture/écriture, il faut terminer l'instruction par *MOD. L'instruction devient :

```
SEGACT nom-variable-pointeur*MOD
```

Par exemple on peut envisager la séquence suivante :

```
subroutine sp1 (ll,lc,...)
segment monseg
  integer ijk,nomb(ll,lc),jlc
  real x,xx(n)
  real*8 y,yy(3,mm)
  character*nbca icha(8,ik),chai
endsegment
pointeur mons1.monseg
  .
  .
segini mons1
INTER=MONS1
SEGDES MONS1
call sp2(INTER,... )
return
end

subroutine sp2(INTER,....)
segment monseg
  integer ijk,nomb(ll,lc),jlc
  real x,xx(n)
  real*8 y,yy(3,mm)
  character*nbca icha(8,ik),chai
endsegment
pointeur mons1.monseg
  .
  .
MONS1=INTER
SEGACT MONS1
AZ=MONS1.XX(2)
SEGDES MONS1
  .
  .
SEGACT MONS1*MOD
MONS1.XX(1)=MONS1.XX(1)*2
SEGDES MONS1
return
end
```

Une fois un segment activé, toutes les variables qu'il contient sont accessibles en lecture au même titre que les autres variables fortran, par contre il n'est pas possible de les modifier. En effet ,il n'est pas normal de vouloir modifier un objet qui existe déjà et les structures de mémoires servant transitoirement dans un opérateur doivent être crées en début d'opérateur et détruites en fin d'opérateur. Il est malgré tout possible de modifier un segment désactivé puis activé en faisant suivre le nom du pointeur à activer de *MOD (SEGACT MONS1*MOD).

Rappelons que LL,LC... ne sont pas des variables contenues dans le segment, l'instruction particulière suivante sert à connaitre les dimensions des tableaux

```
LHJ=NOMS1.NOMB(/i)
```

dans LHJ se trouve la valeur de la ième dimension du tableau NOMB.Les tableaux de chaines de caractères sont traités différemment. NOMS1.ICHA(/1) contient la longueur des chaines du tableau ICHA soit 4 dans notre exemple. NOMS1.ICHA(/i) contient la i-1 ème dimension du tableau ICHA.

Dans un sous-programme, il est possible de connaitre la dimension d'un tableau et Esope permet de changer sa dimension.

2.3.6 Ajustement d'un SEGMENT

L'instruction :

```
SEGADJ nom-variable-pointeur
```

s'emploie comme SEGINI. c'est à dire que **toutes** les variables dimensionnantes des tableaux du segment doivent être définies préalablement à l'appel.

ESOPE, suivant les demandes tronquera ou agrandira les tableaux en conservant les valeurs des variables qui existent avant et après l'instruction.

Exemple :

```
subroutine sp1 (ll,lc,n,mm,inter,...)
segment monseg
  integer ijk,nomb(ll,lc),jlc
  real x,xx(n)
  real*8 y,yy(3,mm)
  character*nbca icha(8,ik),chai
endsegment
pointeur mons1.monseg
  .
  .
MONS1=INTER
N=36
IK=18
SEGADJ MONS1
```

2.4 Les structures de la même classe

Pour pouvoir travailler simultanément sur plusieurs structures de même type, il faut avoir plusieurs variables-pointeurs : MONS1 et des autres. L'appartenance des variables à une structure ou à l'autre se faisant par le préfixe. Dans l'instruction POINTEUR, plusieurs nom-de-variables-pointeur peuvent être définis.

Pour travailler séquentiellement sur des structures de la même classe on peut conserver la valeur de la variable-pointeur dans une variable fortran de type INTEGER. Par exemple :

```
subroutine sp1 (ll,lc,...)
segment monseg
  integer ijk,nomb(ll,lc),jlc
  real x,xx(n)
  real*8 y,yy(3,mm)
  character*nbca icha(8,ik),chai
```

```
endsegment
pointeur mons1.monseg
INTEGER IUYT,IUYR
  .
  .
MM=4
SEGINI MONS1
  .
MONS1.XX(3)= 36
  .
  .
IUYT = MONS1
MM=8
SEGINI MONS1
IUYR = MONS1
  .
MONS1.XX(3)= 10
  .
  .
MONS1 = IUYT
YTR = MONS1.XX(3)
** ytr vaut dans cet exemple 36
SEGDES MONS1
  .
  .
MONS1 = IUYR
TEN = MONS1.XX(3)
** ten vaut dans cet exemple 10
SEGDES MONS1
etc...
```

2.5 Remarques

Nous allons donner une liste de remarques et de conseils.

1. Pour avoir dans un sous-programme des variables incluses dans un segment, il est souvent préférable de passer en argument le pointeur de tout le segment. Ceci devient obligatoire, si on veut de nouveau utiliser des instructions SEG... dans le sous-programme.
2. Il ne faut pas utiliser comme variable dimensionnante d'un segment une variable appartenant à ce même segment.
3. On peut activer ou désactiver ou supprimer plusieurs segment par la même instruction en séparant le nom des variables-pointeurs par une virgule.

   ```
   SEGDES MONS1,MONS2,...
   ```

4. Si on veut utiliser fréquemment la notion de segment ajustable il est bon de ne mettre qu'un seul tableau dans le segment.
5. Il est d'usage de ne pas polluer la mémoire en supprimant les segments de travail temporaires et en désactivant les segments qui ne servent plus localement dans ce sous-programme ou dans l'opérateur.
6. L'existence des segments évite l'emploi de COMMON de travail.

7. On peut dupliquer un segment existant. Pour dupliquer le segment MONS1 en le recopiant dans MONS2 on fera :

   ```
   SEGINI,MONS2=MONS1
   ```

8. En lisant le source Esope de Cast3M on remarque des instructions, commençant en colonne 1, du type :

   ```
   -INC S*****.INC
   ```

   Cette instruction demande de remplacer la ligne par le contenu du fichier S*****.INC. Tous les systèmes d'exploitations ne permettent pas à ESOPE de le faire lui-même, et dans le cas contraire, il faut faire le travail soi même avant de soumettre le source à la traduction fortran 77. Nous avons procédé ainsi pour deux raisons principales :
   - La qualité est garantie en ne réécrivant pas les instructions contenues dans le fichier.
   - Un fichier correspond souvent à une structure d'OBJET et sa déclaration sera strictement identique dans tous les sous-programmes l'utilisant.

9. Si vous avez des remarques supplémentaires, prenez la peine de nous les signaler, nous vous en serions reconnaissant.

2.6 Simplification

2.6.1 Nom-de-variable-pointeur

La notion de classe d'objet et de pointeur permettant l'instanciation d'un objet particulier de la classe n'avait pas été complètement dégagée au moment de la spécification du langage Esope. D'un autre coté, pour identifier une variable appartenant à un objet instancié il faut rappeler le nom-de-variable-pointeur comme préfixe de la variable. Une simplification possible est d'attribuer un nom par défaut à ce nom-de-variable-pointeur. La solution la plus simple est de prendre le nom de la classe de la structure. Ainsi, dans les exemples précédents, MONSEG est la variable qui identifie la classe de la structure et qui peut servir de nom par défaut. On peut construire l'exemple suivant :

```
subroutine sp1 (ll,lc,...)
segment monseg
  integer ijk,nomb(ll,lc),jlc
  real x,xx(n)
  real*8 y,yy(3,mm)
  character*nbca icha(8,ik),chai
endsegment
  .
  .
mm=4
SEGINI MONSEG
  .
MONSEG.XX(3)= 36
****  ou encore directement
XX(3)= 36
  .
  .
SEGDES MONSEG
  .
SEGACT MONSEG
etc...
```

2.6.2 Exemple d'utilisation complet

Dans ce dernier exemple nous cherchons à donner un aperçu de l'utilisation des segments dans Cast3M. Il s'agit de construire le triangle de Pascal (le pivot initial n'est pas forcement égal à 1). Chaque ligne est représentée par un segment. On veut pouvoir avoir accès directement à la n-iéme ligne. La structure de notre objet triangle de Pascal peut être :

```
SEGMENT MPASC
  INTEGER ILIGN(L)
ENDSEGMENT

SEGMENT LIGN
  REAL*8 XLIGN(M)
ENDSEGMENT
POINTEUR LIG1.LIGN
```

Le tableau ILIGN contient la valeur des pointeurs sur le segment LIGN et les valeurs sont dans le tableau XLIGN

Le premier sous-programme crée un triangle de pascal avec XUN comme pivot et avec LI lignes, et le second sous programme écrira un objet triangle de Pascal reçu en argument.

```
  SUBROUTINE SP11 (XUN,LI,IRET)
  IMPLICIT REAL*8(A-H,O-Z)
  SEGMENT MPASC
    INTEGER ILIGN(L)
  ENDSEGMENT
  SEGMENT LIGN
    REAL*8 XLIGN(M)
  ENDSEGMENT
  POINTEUR LIG1.LIGN
  L = LI
  SEGINI MPASC
  IRET = MPASC
  IF ( LI.EQ.0) THEN
  SEGDES MPASC
  RETURN
  ENDIF
  M=1
  SEGINI LIGN
  XLIGN(1)= XUN
  ILIGN(1)=LIGN
  DO 1 I=2,LI
  M= I
  SEGINI LIG1
  ILIGN(I)= LIG1
  LIG1.XLIGN(1) = XUN
  LIG1.XLIGN(I) = XUN
  DO 2 KO = 1,I-2
  LIG1.XLIGN(KO+1) = XLIGN(KO) + XLIGN(KO+1)
2 CONTINUE
  SEGDES LIGN
  LIGN=LIG1
```

```
1 CONTINUE
  SEGDES LIGN
  SEGDES MPASC
  RETURN
  END


  SUBROUTINE SP2(IPASC)
  IMPLICIT REAL*8(A-H,O-Z)
  SEGMENT MPASC
    INTEGER ILIGN(L)
  ENDSEGMENT
  SEGMENT LIGN
    REAL*8 XLIGN(M)
  ENDSEGMENT
  POINTEUR LIG1.LIGN
  MPASC=IPASC
  SEGACT MPASC
  DO 1 I=1,ILIGN(/1)
  LIGN= ILIGN(I)
  SEGACT LIGN
  WRITE(6,*) (XLIGN(K),K=1,XLIGN(/1))
  SEGDES LIGN
1 CONTINUE
  SEGDES MPASC
  RETURN
  END
```

Chapitre 3

Les objets

3.1 Introduction

Les TYPES abstraits sont des regroupements d'informations, certains ne représentent que des données mathématiques ou informatiques (entier, réel, liste de mots...), d'autres ont un caractère plus physique et s'adaptent à une modélisation par éléments finis (champs par élément, champs par point, matrices de conductivité...). En addition de la description formelle des informations contenues dans le type, il est fourni la description informatique, ceci représente en fait les méthodes d'accès, de création et de suppression de ces types et une fois instancié nous sommes en présence d'un objet qui a un type et des méthodes.

Le support informatique des types abstraits est le segment Esope (sauf pour entier, réel,logique et mot). Nous donnerons en annexe la liste informatique de tous les objets de Cast3M, et nous détaillerons seulement des types qui ont une réalité physique, c'est à dire les types MAILLAGE, MODELE, CHPOINT, MCHAML et un autre qui appartient à l'analyseur de syntaxe, c'est à dire la TABLE.

L'idée qui sous-tend la définition d'un type abstrait est de répondre à la question : « qu'est-il nécessaire et suffisant de garder comme informations sur un objet, vu l'utilisation qui est envisageable ? ». Par exemple si on voulait définir un objet « couteau », il ne faudrait pas garder l'information qu'il passe ou non en machine à laver, si on est sur et certain de ne jamais en posséder une. La définition d'un type abstrait est stratégique, en modifier un conduit toujours à un travail lourd et fastidieux. L'introduction d'un nouveau type abstrait est aussi un travail important, il faut programmer toute l'intendance autour du type : l'écriture, la sauvegarde-restitution, les opérations élémentaires ... La pratique montre qu'il est très rare que les types existants ne soient pas suffisants pour satisfaire les besoins. Dans la suite nous ne parlerons plus de TYPE abstrait, nous utiliserons directement le mot objet car nous préciserons directement la structure informatique.

3.2 L'objet MAILLAGE

cet objet est supporté par le segment MELEME dont voici la liste :

3.2.1 Segment associé au maillage

```
*       -INC SMELEME
*
*     L'OBJET  MAILLAGE  REPRESENTE UNE TOPOLOGIE
*
      SEGMENT      MELEME
      INTEGER      ITYPEL
      INTEGER      NUM(NBNN,NBELEM)
      INTEGER      LISOUS(NBSOUS),LISREF(NBREF)
      INTEGER      ICOLOR(NBELEM)
```

```
      ENDSEGMENT
      POINTEUR     IPT1.MELEME,IPT2.MELEME,IPT3.MELEME,IPT4.MELEME
      POINTEUR     IPT5.MELEME,IPT6.MELEME,IPT7.MELEME,IPT8.MELEME
      POINTEUR     IPT9.MELEME
*
*     OBJET GEOMETRIQUE SIMPLE (UN SEUL TYPE D'ELEMENT)
*
*     ITYPEL   : NUMERO DU TYPE D'ELEMENT
*     NBSOUS   : 0 PAS DE SOUS-OBJETS
*     NBREF    : NOMBRE DE REFERENCES ( COTES FACES ....
*     LISREF   : LISTE DES REFERENCES (POINTEURS SUR MELEME
*     NBNN     : NOMBRE DE POINTS DANS L'ELEMENT
*     NBELEM   : NOMBRE D ELEMENTS DANS LE MELEME
*     NUM(I,J) : NUMERO DU IEME NOEUD DU JIEME ELEMENT
*     ICOLOR   : COULEUR DE CHAQUE ELEMENT
*
*     OBJET GEOMETRIQUE COMPLEXE (PLUSIEURS TYPES D'ELEMENTS)
*
*     ITYPEL   : PAS UTILISE
*     NBSOUS :NOMBRE DE SOUS-OBJETS(OBJETS GEOMETRIQUES SIMPLES)
*     LISOUS   : LISTE DES SOUS-OBJETS (POINTEURS SUR MELEME)
*     NBREF    : NOMBRE DE REFERENCES ( COTES FACES ....
*     LISREF   : LISTE DES REFERENCES (POINTEURS SUR MELEME
*     NBNN     : 0
*     NBELEM   : 0
*
```

Pour un objet maillage composé d'un seul type d'élément, les informations conservées sont :

ITYPEL : numéro du type de l'élément (un triangle à 3 nœuds porte le numéro 4) d'après NOMS du common CGEOMC initialisé par block data.

NUM : tableau donnant les nbnn nœuds des nbelem éléments.

ICOLOR : tableau d'entiers précisant la couleur de chaque élément. L'ordre des couleurs est le suivant : 'BLEU','ROUGE','ROSE','VERT','TURQUOISE','JAUNE','BLANC'

LISREF : tableau pouvant référencer d'autres objets maillages se rapportant au premier, par exemple le contour ou les cotés...

Un objet maillage complexe, c'est-à-dire contenant plusieurs types d'éléments, sera différent. Il contient surtout la liste des objets maillages élémentaires qui le composent.

LISOUS : liste des pointeurs référençant les maillages élémentaires.

3.2.2 Exemple d'utilisation

Illustrons ceci par un exemple de programmation montrant comment dérouler un objet MAILLAGE. Le but, sans aucun intérêt, du sous-programme est de faire la somme de tous les numéros de nœuds qui apparaissent dans la description des éléments de l'objet maillage référencé par ipoin.

```
      subroutine sp1 (ipoin)
C     insertion des lignes declarations du segment relatif a l'objet
C     maillage
```

```
  SEGMENT      MELEME
  INTEGER      ITYPEL
  INTEGER      NUM(NBNN,NBELEM)
  INTEGER      LISOUS(NBSOUS),LISREF(NBREF)
  INTEGER      ICOLOR(NBELEM)
  ENDSEGMENT
  POINTEUR    IPT1.MELEME,IPT2.MELEME,IPT3.MELEME,IPT4.MELEME
  POINTEUR    IPT5.MELEME,IPT6.MELEME,IPT7.MELEME,IPT8.MELEME
  POINTEUR    IPT9.MELEME
C  supposons que la variable ipoin, passee en argument soit la
C  valeur du pointeur de l'objet maillage. on utilise l'alias IPT1
  ipt1=ipoin
  segact ipt1
  nsom=0
C recuperons par la dimension de lisous, si l'objet est
C elementaire ou complexe. nbso sera le nombre d'objets
C elementaires composant l'objet maillage ipoin.
  nbso =max(1, ipt1.lisous(/1))
C  faisons une boucle sur les sous objets
  do 1 i = 1,nbso
  if(nbso.eq.1) then
    meleme=ipt1
  else
    meleme = ipt1.lisous(i)
    segact meleme
  endif

C boucle sur les elements
  do 2 k=1,num(/2)
C boucle sur les noeuds de l'element
  do 3 m = 1, num(/1)
  ia = num(m,k)
  nsom=nsom+ia
3 continue
2 continue
  if(nbso.ne.1) segdes meleme
1 continue
  segdes ipt1
  return
  end
```

3.2.3 Remarques

- Beaucoup d'objets contiennent une référence à un objet maillage.
- Par construction, deux sous-objets ne peuvent pas être composés d'éléments de même type. Une partition du maillage par type d'éléments est une propriété qu'il faut respecter, beaucoup d'opérateurs l'utilisent.
- L'objet maillage support de champ par point est composé d'éléments de type POI1, c'est à dire que ITYPEL=1 et qu'il n'y a qu'un point par élément.

3.3 L'objet CHPOINT

Un champ défini aux nœuds du maillage (les points) s'appelle un CHPOINT (prononcez champoint) et se trouve dans le segment MCHPOI. Un champ par point existe à priori sur tous les nœuds mais il est inutile de le définir partout où il vaut zéro. Ainsi le choix informatique fait consiste à ne spécifier dans un champoint que les nœuds directement concernés, de plus on réalise une partition des nœuds par les inconnues qu'il supporte. Pour un type de nœuds les informations conservées sont le noms des composantes et leur numéro d'harmonique(pour un calcul 2D-axis). On regroupe donc dans un segment ces deux informations avec un pointeur sur l'objet maillage contenant la liste des nœuds concernés(éléments POI1 à 1 nœud par élément) ainsi qu'un pointeur sur un segment contenant le tableau des valeurs. Comme une partition est réalisée il faut, au dessus du segment, un chapeau contenant la liste des partitions et quelques informations générales : la nature du champ, le titre du champ, le mode de calcul dans lequel le champ a été crée, et un type (qui n'est pas vraiment utilisé).

Le champoint est un peu lourd à utiliser du fait de la partition réalisée, il peut être intéressant de le mettre sous la forme informatique contenue dans le segment MTRAV et ensuite d'appeler le sous-programme CRECHP pour réaliser la partition.

3.3.1 Liste des segments du CHPOINT

```
*             -INC SMCHPOI
*
*     OBJET CHPOINT  : REPRESENTE UN CHAMP DISCRETISE PAR POINT
*
      SEGMENT MCHPOI
       CHARACTER*8  MTYPOI
       CHARACTER*72 MOCHDE
       INTEGER      JATTRI(NAT)
       INTEGER      IPCHP(NSOUPO),IFOPOI
      ENDSEGMENT
      POINTEUR MCHPO1.MCHPOI,MCHPO2.MCHPOI,MCHPO3.MCHPOI,MCHPO4.MCHPOI
*
      SEGMENT MSOUPO
       CHARACTER*4 NOCOMP(NC)
       INTEGER     IGEOC,IPOVAL
       INTEGER     NOHARM(NC)
      ENDSEGMENT
      POINTEUR MSOUP1.MSOUPO,MSOUP2.MSOUPO,MSOUP3.MSOUPO,
     #         MSOUP4.MSOUPO,MSOUP5.MSOUPO
*
      SEGMENT MPOVAL
       REAL*8 VPOCHA(N,NC)
      ENDSEGMENT
      POINTEUR MPOVA1.MPOVAL,MPOVA2.MPOVAL,MPOVA3.MPOVAL,
     #         MPOVA4.MPOVAL,MPOVA5.MPOVAL,MPOVA6.MPOVAL
*
*     MSOUPO DECRIT UNE PARTITION DE LA GEOMETRIE
*
*     MTYPOI : TYPE DU CHPOINT
*     MOCHDE : TITRE
*     JATTRI : ATTRIBUT DE NATURE
*              JATTRI(1) NATURE DIFFUSE OU DISCRETE DU CHAMP
*                0: INDETERMINE  1: DIFFUS  2:DISCRET
```

```
*              JATTRI(2) LIBRE POUR L'INSTANT
*                0:              1:         2:
*     IPCHP  : POINTEURS SUR LES SEGMENTS MSOUPO
*     IFOPOI : CORRESPOND A L'OPTION IFOUR(VOIR CCOPTIO)
*     IGEOC  : POINTEUR SUR UN OBJET MELEME
*     NOCOMP : NOMS DES COMPOSANTES DU CHAMP
*     NOHARM : NUMERO DE L'HARMONIQUE CORRESPONDANT A LA COMPOSANTE
*              NOCOMP ,SI IFOPOI=1
*     IPOVAL : POINTEUR SUR LE SEGMENT MPOVAL
*     VPOCHA(I,J) : VALEUR DU CHAMP IEME POINT,JIEME COMPOSANTE
*
*   -END INCLUDE
```

3.3.2 Exemple d'utilisation du CHPOINT

L'exemple traité n'a pas d'intérêt si ce n'est de dérouler tous les segments de l'objet. Soit un opérateur qui calcule la somme des composantes de nom UX des nœuds dont le numéro est supérieur à 100.

```
      SUBROUTINE SOMEXE
      IMPLICIT REAL*8 (A-H,O-Z)
-INC SMCHPOI
-INC CCOPTIO
-INC SMELEME
C les trois lignes precedentes veulent dire qu'il faut les remplacer
C par les segments representant le CHPOINT  et le MAILLAGE(MELEME) et
C par le common COPTIC

C lecture d'un objet champoint
      CALL LIROBJ('CHPOINT ',MCHPOI,1,IRETOU)
C la lecture est obligatoire, IERR du common COPTIC nous renseigne
C pour savoir si elle s'est bien passee.
      IF(IERR.NE.0) RETURN
      SEGACT MCHPOI
      XX = 0.D0
C boucle sur les sous zones
      DO 1 I=1,IPCHP(/1)
      MSOUPO=IPCHP(I)

C dans une zone, boucle sur le nom des composants
      DO 2 J=1,NOCOMP(/2)
      IF(NOCOMP(J).NE.'UX  ') GO TO 2
      MELEME = IGEOC
      SEGACT MELEME
      MPOVAL=IPOVAL
      SEGACT MPOVAL

C boucle sur le numero des points
      DO 3 K=1,NUM(/2)
      IF(NUM(1,K).GT.100) XX = XX + VPOCHA(K,J)
  3   CONTINUE
```

```
C desactivation des segments
      SEGDES MELEME,MPOVAL
  2   CONTINUE
      SEGDES MSOUPO
  1   CONTINUE
      SEGDES MCHPOI

C ecriture du resultat dans la pile
      CALL ECRREE(XX)
      RETURN
      END
```

3.4 L'objet MMODEL ou modèle

Cet objet supporte les informations qui permettent d'associer une formulation à un maillage.

Par formulation, on sous-entend le type de calcul (thermique, mécanique,..), le type de matériau (élastique,plastique,..), le type d'élément fini (coq4,cub8,..). Le modèle est en quelque sorte la clef qui permet d'exploiter les champs par éléments, et il est donc, la plupart du temps, associé au(x) champ(s) par élément dans les données des opérateurs.

3.4.1 Liste de l'objet MMODEL

```
*    -INC SMMODEL
*
*     OBJET DE TYPE "MODELE"
*
      SEGMENT,MMODEL
         INTEGER KMODEL(N1)
      ENDSEGMENT
       POINTEUR MMODE1.MMODEL,MMODE2.MMODEL
*
      SEGMENT,IMODEL
         INTEGER IMAMOD,NEFMOD
         INTEGER INFMOD(MN3)
         CHARACTER*16 CONMOD
         CHARACTER*16 FORMOD(NFOR),MATMOD(NMAT)
      ENDSEGMENT
       POINTEUR IMODE1.IMODEL,IMODE2.IMODEL
*
*     KMODEL : POINTEUR SUR DES OBJETS MODELES ELEMENTAIRES
*     IMAMOD : POINTEUR SUR UN OBJET MAILLAGE ELEMENTAIRE
*     NEFMOD : NUMERO DE L'ELEMENT FINI DANS NOMTP
*     INFMOD : INFORMATIONS VARIEES RELATIVES AU MODELE
*              ( VIDE PAR DEFAUT )
*     INFMOD(1) : NOMBRE DE PTS D'INTEGRATION DANS L'EPAISSEUR
*
*     CONMOD : NOM DU CONSTITUANT (BLANC PAR DEFAUT )
*     FORMOD : SUITE DE MOTS DEFINISSANT LA FORMULATION
*              (EX: THERMIQUE , LIQUIDE + MECANIQUE, ...)
*     MATMOD : SUITE DE MOTS CARACTERISANT LE TYPE DE MATERIAU
```

```
*              (EX: ORTHOTROPE, PLASTIQUE PARFAIT, ...)
```

3.4.2 Remarques

Dans de nombreux opérateurs, la donnée de base est le modèle. Par exemple, dans la fabrication des matrices de rigidités élémentaires obtenue par l'instruction :

```
RIG1  = RIGI  MOD1  MAT1 ;
```

On cherche à fabriquer les matrices élémentaires associées à l'objet modèle MOD1. MAT1 est alors un champ par élément de propriétés, qui doit contenir les propriétés nécessaires pour l'opérateur RIGI, mais qui peut en contenir beaucoup d'autres, y compris relatives à des zones de maillage qui n'ont rien à voir avec celles sous-tendues par MOD1.

De manière plus précise, l'objet MMODEL peut être un ensemble de zones élémentaires, caractérisées par un type d'élément fini unique. Dans ce cas l'opérateur traite ces zones les unes après les autres et va chercher pour chacune, dans MAT1, les informations dont il a besoin.

Chaque modèle élémentaire pointe sur un maillage (variable INAMOD). Pour pouvoir utiliser simultanément des modèles élémentaires différents pointant sur le même maillage, comme par exemple dans le cas des coques multicouches, il est nécessaire de les différencier en leur attribuant des noms de constituants (variable CONMOD).

Le tableau INFMOD permet, en cas de besoin, de rajouter des informations nécessaires. Actuellement seul INFMOD(1) est utilisé et contient le nombre de points d'intégration dans l'épaisseur, pour les éléments de coques intégrés dans l'épaisseur.

3.5 L'objet MCHAML ou champ par élément

Cet objet supporte tous les champs qui sont définis sur les éléments. Pour pouvoir être correctement exploité, il faut souvent connaitre la formulation de l'élément. Celle-ci est dans l'objet MODELE qui est à fournir avec l'objet MCHAML dans la plupart des opérateurs.

Les champs supportés par le MCHAML (prononcez chamélem) sont de types réels ou sont des pointeurs vers d'autres objets.

3.5.1 Liste de l'objet MCHAML

```
*      -INC SMCHAML
*
*      OBJET DE TYPE "MCHAML" : CHAMP PAR ELEMENT
*
       SEGMENT,MCHELM
          CHARACTER*(L1) TITCHE
          CHARACTER*16   CONCHE(N1)
          INTEGER IMACHE(N1),ICHAML(N1)
          INTEGER INFCHE(N1,N3),IFOCHE
       ENDSEGMENT
       POINTEUR MCHEL1.MCHELM,MCHEL2.MCHELM,MCHEL3.MCHELM
       POINTEUR MCHEL4.MCHELM,MCHEL5.MCHELM,MCHEL6.MCHELM
*
       SEGMENT,MCHAML
          CHARACTER*8 NOMCHE(N2)
          CHARACTER*16 TYPCHE(N2)
          INTEGER IELVAL(N2)
```

```
      ENDSEGMENT
      POINTEUR MCHAM1.MCHAML,MCHAM2.MCHAML,MCHAM3.MCHAML
      POINTEUR MCHAM4.MCHAML,MCHAM5.MCHAML,MCHAM6.MCHAML
*
      SEGMENT,MELVAL
         REAL*8 VELCHE(N1PTEL,N1EL)
         INTEGER IELCHE(N2PTEL,N2EL)
      ENDSEGMENT
      POINTEUR MELVA1.MELVAL,MELVA2.MELVAL,MELVA3.MELVAL
      POINTEUR MELVA4.MELVAL,MELVA5.MELVAL,MELVA6.MELVAL
*
*     TITCHE   : TITRE DU CHAMELEM
*     CONCHE   : NOM DU CONSTITUANT ( BLANC PAR DEFAUT )
*     IFOCHE   : CORRESPOND A L'OPTION IFOUR ( CF CCOPTIO )
*     IMACHE   : POINTEURS SUR DES MAILLAGES ELEMENTAIRES
*     ICHAML   : POINTEURS SUR DES SEGMENTS "MCHAML"
*     INFCHE   : ENSEMBLE D'INFORMATIONS OU LES VALEURS PAR DEFAUT SONT 0
*     INFCHE(.,1)=0 POUR DES VALEURS INDEPENDANTES DU REPERE
*                 =1 POUR DES VALEURS DEFINIES DANS LES REPERES LOCAUX
*                 =2 POUR DES VALEURS DEFINIES DANS LE REPERE GLOBAL
*     INFCHE(.,2)= LIBRE
*                   (  ANCIENNEMENT LE NUMERO DE LA COUCHE )
*     INFCHE(.,3)= NUMERO DE L'HARMONIQUE DE FOURIER
*     INFCHE(.,4)=0 POUR DES VALEURS DEFINIES AUX NOEUDS
*                 =POINTEUR SUR UN SMINTE POUR DES VALEURS DEFINIES AILLEU
*     INFCHE(.,5)=0 POUR DES CHAMPS DE DEFORMATIONS ET CONTRAINTES USUELS
*                 =1 POUR DES CHAMPS DE DEFORMATIONS ET CONTRAINTES
*                    EN FORMULATION "MASSIF" POUR DES ELEMENTS DE COQUE OU
*                    DE POUTRE OU DE TUYAU
*     INFCHE(.,6):  LE SMINTE POINTE SUR UN SEGMENT D'INTEGRATION :
*                 =1 AUX NOEUDS
*                 =2 AU CENTRE DE GRAVITE
*                 =3 AUX POINTS DE GAUSS POUR LA RIGIDITE
*                 =4 AUX POINTS DE GAUSS POUR LA MASSE
*                 =5 AUX POINTS DE GAUSS POUR LES CONTRAINTES
*
*     NOMCHE   : NOMS DES COMPOSANTES DU CHAMELEM
*     TYPCHE   : TYPES DES COMPOSANTES DU CHAMELEM:
*                 - TYPCHE(..)='REAL*8' POUR DES COMPOSANTES REELLES,
*                 - TYPCHE(..)='POINTEUR M...' POUR DES COMPOSANTES
*                   NON SCALAIRES, DECRITES PAR UN SEGMENT DE NOM "M.."
*     IELVAL   : POINTEURS SUR DES SEGMENTS "MELVAL"
*
*     VELCHE(N1PTEL,N1EL) : VALEUR D'UNE COMPOSANTE REELLE POUR LE
*                           "N1PTEL"-EME NOEUD DU "N1EL"-EME ELEMENT.
*     IELCHE(N2PTEL,N2EL) : POINTEUR SUR UN SEGMENT REPRESENTANT UNE
*                           COMPOSANTE NON SCALAIRE POUR LE
*                           "N2PTEL"-EME NOEUD DU "N2EL"-EME ELEMENT.
*
*     REMARQUES:
*
```

```
*     - SEUL L'UN DES TABLEAUX "VELCHE" ET "IELCHE" EST UN TABLEAU NON
*       VIDE.
*
*     - N.PTEL=N.EL=1  ("."= 1 OU 2)  IMPLIQUE CHAMP UNIFORME DANS LE
*       MAILLAGE.
*
*     - N.PTEL=1  ("."= 1 OU 2)  IMPLIQUE CHAMP CONSTANT PAR ELEMENT.
*
*   ATTENTION :  PAR ZONE ELEMENTAIRE , ON ENTEND UNE ZONE POUR
*                LAQUELLE ON A LES MEMES IMACHE, CONCHE, ET CERTAINS
*                INFCHE
```

3.5.2 Remarques

Pour mettre en relation une zone d'un objet MODELE avec une zone d'un objet MCHAML on s'appuie sur la valeur des pointeurs du maillage (IMACHE). Celui-ci est forcément un objet MAILLAGE élémentaire, c'est à dire n'ayant qu'un type d'élément géométrique. Ce besoin de concordance au niveau des pointeurs du maillage nous crée régulièrement des soucis mais ...

Il existe des sous-programmes dont le but est de « mettre en face » les zones élémentaires d'un objet MODELE et les zones élémentaires des MCHAML. Le mieux est de s'inspirer d'un opérateur existant. Par exemple le sous programme VMISES qui appelle VMISPO calcule un chamelem de Von Mises à partir d'un chamelem de contraintes, d'un MODELE et d'un chamelem de caractéristiques géométriques si nécessaire.

3.6 L'objet TABLE

Cet objet appartient au langage dans la mesure où l'analyseur de syntaxe le traite de façon particulière. D'un autre coté, il peut être utilisé par le programmeur pour regrouper des informations et joue ainsi le rôle de supplétif par rapport aux objets.

3.6.1 Segment associé à la table

```
*        -INC SMTABLE
*
*     OBJET DE TYPE TABLE
*
      SEGMENT MTABLE
            INTEGER       MLOTAB
            CHARACTER*(8) MTABTI(M),MTABTV(M)
            REAL*8        RMTABI(M)
            INTEGER       MTABII(M),MTABIV(M)
            REAL*8        RMTABV(M)
      ENDSEGMENT
      POINTEUR MTAB1.MTABLE,MTAB2.MTABLE,MTAB3.MTABLE
*
*     MLOTAB     : LONGUEUR DE LA TABLE ( <> ALLOCATION )
*     MTABTI(I)  : TYPE DE L INDICE POINTANT SUR LE IEME ELEMENT
*                :  DE LA TABLE
*     MTABTV(I)  : TYPE DE L'OBJET POINTE PAR LE IEME INDICE
*     MTABII(I)  : VALEUR DE L'INDICE  (SI PAS REEL)
*     RMTABI(I)  : VALEUR DE L'INDICE  (SI REEL)
```

```
*     MTABIV(I)  : VALEUR DE L'OBJET   (SI PAS REEL)
*     RMTABV(I)  : VALEUR DE L'OBJET   (SI REEL)
*
```

Créer un segment Esope qui contient plusieurs type de variables prend du temps, on dimensionne donc le segment table par M qui peut être supérieur au nombre réel d'objets contenus, pour l'instant, dans la table. Bien que tous les types d'objets de Cast3M aient une valeur associée de type informatique « integer », l'efficacité du traitement nous a conduit à mettre directement dans le segment la valeur des objets « flottant ». L'integer associé aux objets ayant un segment pour support est la valeur du pointeur, pour les entiers c'est directement leurs valeurs et pour les mots c'est la position de la chaine dans un tableau de chaines géré par l'analyseur de syntaxe, etc...

La mise d'un objet dans la table ou l'acquisition d'un objet d'une table pose de légers problèmes qui résultent de la traduction en valeur entière de certains objets.

Deux sous-programmes permettent le dialogue, le premier pour accéder à un objet dans une table connaissant la valeur de l'indice et le second pour positionner dans une table un objet et son indice.

```
      SUBROUTINE ACCTAB(MTABLE,TAPIND,IVALIN,XVALIN,CHARIN,LOGIN,IOBIN,
     $                        TYPOBJ,IVALRE,XVALRE,CHARRE,LOGRE,IOBRE)
C
C  ****  DONNE ACCES A UN OBJET DANS UNE TABLE de POINTEUR MTABLE
C  ****  ET CONNAISSANT LE TYPE DE L'INDICE ( TAPIND )
C  ****  ET LA VALEUR DE L'INDICE  QUI SERA DANS DIFFERENTS
C  ****  VARIABLES SUIVANT SON TYPE .
C  ****  ENTIER dans IVALIN;FLOTTANT dans XVALIN;
C  ****  MOT dans CHARIN;LOGIQUE dans LOGIN;
C  ****  AUTRE dans IOBIN
C  ****  ON PEUT PRECISER LE TYPE D'OBJET ATTENDU DANS TYPOBJ CE
C  ****  QUI PROVOQUE UN MESSAGE D'ERREUR S'IL N'EXISTE PAS
C  **** EN SORTIE : TYPOBJ TYPE DE L'OBJET AU CAS OU TYPOBJ ETAIT = ' '
C  ****             VALEUR DE L'OBJET DANS IVALRE SI ENTIER; XVALRE SI
C  ****             FLOTTANT; CHARRE SI MOT ( DE LA LONGUEUR DE LA
C  ****             CHAINE ENVOYEE EN ARGUMENT);LOGRE SI LOGIQUE;
C  ****             IOBRE POUR TOUT AUTRE TYPE
C
```

pour réaliser l'opération de mise dans la table il faut appeler le sous programme ECCTAB.

```
      SUBROUTINE ECCTAB(MTABLE,TAPIND,IVALIN,XVALIN,CHARIN,LOGIN,IOBIN,
     $                        TAPOBJ,IVALRE,XVALRE,CHARRE,LOGRE,IOBRE)
C
C  **** MET UN OBJET DANS UNE TABLE
C  **** TAPIND TYPE DE L'INDICE  CHARACTER*(*)
C  ****        PUIS LA VALEUR DE L'INDICE  IVALIN SI ENTIER
C  ****                                    XVALIN SI FLOTTANT
C  ****                                    CHARIN SI MOT
C  ****                                    LOGIN SI LOGIQUE
C  ****                                    IOBIN POUR TOUT AUTRE TYPE
C  **** TAPOBJ TYPE DE L'OBJET CHARACTER*(*)
C  ****        PUIS LA VALEUR DE L'INDICE  IVALRE SI ENTIER
C  ****                                    XVALRE SI FLOTTANT
C  ****                                    CHARRE SI MOT
C  ****                                    LOGRE SI LOGIQUE
```

```
C  ****                                          IOBRE POUR TOUT AUTRE TYPE
C  ****
```

3.6.2 Remarques

Si un message d'erreur est émis, la variable IERR du COMMON COPTIC a une valeur différente de zéro. L'objet TABLE est le seul objet qu'il est autorisé de modifier sans que l'utilisateur ne l'ait demandé.

3.7 Annexe : liste des objets

```
*  -INC SMATTAC
*
*     OBJET DE TYPE 'ATTACHE '
      SEGMENT MATTAC
       INTEGER LISATT(N)
      ENDSEGMENT
       POINTEUR MATTA1.MATTAC,MATTA2.MATTAC
*
      SEGMENT MSOUMA
       CHARACTER*4 ITYATT
       INTEGER IPMATK(M),IATREL(N),IGEOCH,IPHYCH
      ENDSEGMENT
       POINTEUR MSOUM1.MSOUMA,MSOUM2.MSOUMA
*
      SEGMENT MJONCT
       CHARACTER*4 MJODDL,MJOTYP
       INTEGER MJOPOI,ISTRJO(N),IPCHJO(N),IPOSJO(N)
      ENDSEGMENT
       POINTEUR MJONC1.MJONCT,MJONC2.MJONCT
*
      SEGMENT MGEOCH
       CHARACTER*4 MPOPRO(NP)
       INTEGER     INORCH(NI),IMAPRO(N1)
       REAL*8      RJEUCH(NJ),TAIPRO(NT)
      ENDSEGMENT
       POINTEUR MGEOC1.MGEOCH,MGEOC2.MGEOCH
*
      SEGMENT MPHYCH
       REAL*8 RAIPRO,AMOPRO(NA),FROPRO(NF)
      ENDSEGMENT
       POINTEUR MPHYC1.MPHYCH,MPHYC2.MPHYCH
*
*     LISATT : POINTEURS SUR LES SEGMENTS MSOUMA
*     ITYATT : TYPE DE LA LIAISON:MECA,FLUI,CHOC
*     IGEOCH : POINTEUR SUR LE SEGMENT MGEOCH:CARACTERISTIQUES GEOMETRI
*              QUES DU CHOC
*     IPHYCH : POINTEUR SUR LE SEGMENT MPHYCH:CARACTERISTIQUES PHYSIQUES
*              DU CHOC
*     IPMATK : POINTEURS SUR DES OBJETS DE TYPE RIGIDITE
*     IATREL : POINTEURS SUR LES SEGMENTS MJONCT
*     MJOPOI : POINT ASSOCIE AU MJONCT
*     MJODDL : =LX (RESP.FLX) SI MJONCT LIBRE (RESP.BLOQUE)
*     MJOTYP : RAPPEL DE ITYATT=TYPE DE LA LIAISON.
*     ISTRJO : POINTEURS SUR LES MSOSTU D UN OBJET STRUCTURE
*     IPCHJO : POINTEURS SUR OBJETS DE TYPE CHPOINT
*     IPOSJO : POINTEURS SUR OBJETS DE TYPE MELEME (CENTRE DU PROFIL)
*     INORCH : POINTEURS SUR OBJETS DE TYPE POINT DEFINISSANT LE REPERE
*              DU PLAN DE CHOC
*     RJEUCH : FLOTTANT
```

```
*     IMAPRO : POINTEURS SUR OBJETS DE TYPE MELEME : PROFILS DE CHOC
*     MPOPRO : MOTS:POSITION RELATIVE DES DEUX PROFILS
*     TAIPRO : FLOTTANTS:TAILLES DES PROFILS
*     RAIPRO : FLOTTANT:RAIDEUR
*     AMOPRO : FLOTTANT:AMORTISSEMENT
*     FROPRO : FLOTTANT:COEF DE FROTTEMENT
*
*  -ENDINCLUDE
```

```
C=====================================================================
C     OBJET DE TYPE 'BASEMODA'
      SEGMENT MBASEM
        INTEGER LISBAS(N)
      ENDSEGMENT
      POINTEUR MBASE1.MBASEM,MBASE2.MBASEM
C
      SEGMENT MSOBAS
        INTEGER IBSTRM(NIBST)
      ENDSEGMENT
      POINTEUR MSOBA1.MSOBAS,MSOBA2.MSOBAS
C
C     LISBAS(I) : POINTEUR SUR MSOBAS
C     IBSTRM(1) : POINTEUR SUR OBJET STRUCTURE (SEGMENT MSOSTU)
C     IBSTRM(2) : POINTEUR SUR OBJET MSOLUT (LES MODES)
C     IBSTRM(3) : POINTEUR SUR OBJET MSOLUT (LES SOLUTIONS STATIQUES)
C     IBSTRM(4) : POINTEUR SUR OBJET ATTACHE (LES LIAISONS)
C     IBSTRM(5) : POINTEUR SUR OBJET MSOLUT (LES PSEUDO MODES)
C
C=====================================================================
```

```
*
*     -INC SMCHAML
*
*     OBJET DE TYPE "CHAMELEM" : CHAMP PAR ELEMENT
*
      SEGMENT,MCHELM
         CHARACTER*(L1) TITCHE
         CHARACTER*16   CONCHE(N1)
         INTEGER IMACHE(N1),ICHAML(N1)
         INTEGER INFCHE(N1,N3),IFOCHE
      ENDSEGMENT
      POINTEUR MCHEL1.MCHELM,MCHEL2.MCHELM,MCHEL3.MCHELM
      POINTEUR MCHEL4.MCHELM,MCHEL5.MCHELM,MCHEL6.MCHELM
*
      SEGMENT,MCHAML
         CHARACTER*8 NOMCHE(N2)
         CHARACTER*16 TYPCHE(N2)
         INTEGER IELVAL(N2)
      ENDSEGMENT
      POINTEUR MCHAM1.MCHAML,MCHAM2.MCHAML,MCHAM3.MCHAML
      POINTEUR MCHAM4.MCHAML,MCHAM5.MCHAML,MCHAM6.MCHAML
*
      SEGMENT,MELVAL
         REAL*8 VELCHE(N1PTEL,N1EL)
         INTEGER IELCHE(N2PTEL,N2EL)
      ENDSEGMENT
      POINTEUR MELVA1.MELVAL,MELVA2.MELVAL,MELVA3.MELVAL
      POINTEUR MELVA4.MELVAL,MELVA5.MELVAL,MELVA6.MELVAL
*
*     TITCHE   : TITRE DU CHAMELEM
*     CONCHE   : NOM DU CONSTITUANT ( BLANC PAR DEFAUT )
*     IFOCHE   : CORRESPOND A L'OPTION IFOUR ( CF CCOPTIO )
*     IMACHE   : POINTEURS SUR DES MAILLAGES ELEMENTAIRES
*     ICHAML   : POINTEURS SUR DES SEGMENTS "MCHAML"
*     INFCHE   : ENSEMBLE D'INFORMATIONS OU LES VALEURS PAR DEFAUT SONT 0
*     INFCHE(.,1)=0 POUR DES VALEURS INDEPENDANTES DU REPERE
*                 =1 POUR DES VALEURS DEFINIES DANS LES REPERES LOCAUX
*                 =2 POUR DES VALEURS DEFINIES DANS LE REPERE GLOBAL
*     INFCHE(.,2)= LIBRE
*                  (  ANCIENNEMENT LE NUMERO DE LA COUCHE )
*     INFCHE(.,3)= NUMERO DE L'HARMONIQUE DE FOURIER
*     INFCHE(.,4)=0 POUR DES VALEURS DEFINIES AUX NOEUDS
*                 =POINTEUR SUR UN SMINTE POUR DES VALEURS DEFINIES AILLEU
*     INFCHE(.,5)=0 POUR DES CHAMPS DE DEFORMATIONS ET CONTRAINTES USUELS
*                 =1 POUR DES CHAMPS DE DEFORMATIONS ET CONTRAINTES
*                    EN FORMULATION "MASSIF" POUR DES ELEMENTS DE COQUE OU
*                    DE POUTRE OU DE TUYAU
*     INFCHE(.,6):  LE SMINTE POINTE SUR UN SEGMENT D'INTEGRATION :
*                 =1 AUX NOEUDS
*                 =2 AU CENTRE DE GRAVITE
*                 =3 AUX POINTS DE GAUSS POUR LA RIGIDITE
```

```
*               =4 AUX POINTS DE GAUSS POUR LA MASSE
*               =5 AUX POINTS DE GAUSS POUR LES CONTRAINTES
*
*     NOMCHE   : NOMS DES COMPOSANTES DU CHAMELEM
*     TYPCHE   : TYPES DES COMPOSANTES DU CHAMELEM:
*                - TYPCHE(..)='REAL*8' POUR DES COMPOSANTES REELLES,
*                - TYPCHE(..)='POINTEUR M...' POUR DES COMPOSANTES
*                  NON SCALAIRES, DECRITES PAR UN SEGMENT DE NOM "M..."
*     IELVAL   : POINTEURS SUR DES SEGMENTS "MELVAL"
*
*     VELCHE(N1PTEL,N1EL) : VALEUR D'UNE COMPOSANTE REELLE POUR LE
*                           "N1PTEL"-EME NOEUD DU "N1EL"-EME ELEMENT.
*     IELCHE(N2PTEL,N2EL) : POINTEUR SUR UN SEGMENT REPRESENTANT UNE
*                           COMPOSANTE NON SCALAIRE POUR LE
*                           "N2PTEL"-EME NOEUD DU "N2EL"-EME ELEMENT.
*
*     REMARQUES:
*
*     - SEUL L'UN DES TABLEAUX "VELCHE" ET "IELCHE" EST UN TABLEAU NON
*       VIDE.
*
*     - N.PTEL=N.EL=1  ("."= 1 OU 2)  IMPLIQUE CHAMP UNIFORME DANS LE
*       MAILLAGE.
*
*     - N.PTEL=1  ("."= 1 OU 2)  IMPLIQUE CHAMP CONSTANT PAR ELEMENT.
*
*   ATTENTION :  PAR ZONE ELEMENTAIRE , ON ENTEND UNE ZONE POUR
*                LAQUELLE ON A LES MEMES IMACHE, CONCHE, ET CERTAINS
*                INFCHE
```

```
* -INC SMCHARG
*
*
*     OBJET DE TYPE 'CHARGEME'
      SEGMENT MCHARG
       INTEGER KCHARG(N)
       CHARACTER*8 CHANAT(N)
       CHARACTER*4 CHANOM(N)
      ENDSEGMENT
       POINTEUR MCHAR1.MCHARG,MCHAR2.MCHARG
*
      SEGMENT ICHARG
       CHARACTER*8 CHATYP
       INTEGER ICHPO1,ICHPO2,ICHPO3
      ENDSEGMENT
       POINTEUR ICHAR1.ICHARG,ICHAR2.ICHARG
*
*     KCHARG(I) : POINTEUR SUR LE SEGMENT ICHARG
*     CHANAT    : MOT INDIQUANT LA NATURE DU CHARGEMENT: FORCE OU DEPLA
*     CHANOM    : MOT INDIQUANT LE NOM DU CHARGEMENT
*     CHATYP    : TYPE DE L OBJET : CHPOINT MCHAML ou TABLE
*     ICHPO1    : POINTEUR SUR L OBJET PRECEDEMMENT DEFINI
*     ICHPO2    : POINTEUR SUR UN OBJET LISTREEL: INSTANTS T ou TABLE
*     ICHPO3    : POINTEUR SUR UN OBJET LISTREEL: AMPLITUDES AU TEMPS T
*
```

```
*            -INC SMCHPOI
*
*     OBJET CHPOINT  : REPRESENTE UN CHAMP DISCRETISE PAR POINT
*
      SEGMENT MCHPOI
       CHARACTER*8  MTYPOI
       CHARACTER*72 MOCHDE
       INTEGER      JATTRI(NAT)
       INTEGER      IPCHP(NSOUPO),IFOPOI
      ENDSEGMENT
      POINTEUR MCHPO1.MCHPOI,MCHPO2.MCHPOI,MCHPO3.MCHPOI,MCHPO4.MCHPOI
*
      SEGMENT MSOUPO
       CHARACTER*4 NOCOMP(NC)
       INTEGER     IGEOC,IPOVAL
       INTEGER     NOHARM(NC)
      ENDSEGMENT
      POINTEUR MSOUP1.MSOUPO,MSOUP2.MSOUPO,MSOUP3.MSOUPO,
     #         MSOUP4.MSOUPO,MSOUP5.MSOUPO
      SEGMENT MPOVAL
       REAL*8 VPOCHA(N,NC)
      ENDSEGMENT
      POINTEUR MPOVA1.MPOVAL,MPOVA2.MPOVAL,MPOVA3.MPOVAL,
     #         MPOVA4.MPOVAL,MPOVA5.MPOVAL,MPOVA6.MPOVAL
*     MSOUPO DECRIT UNE PARTITION DE LA GEOMETRIE
*     MTYPOI : TYPE DU CHPOINT
*     MOCHDE : TITRE
*     JATTRI : ATTRIBUT DE NATURE
*              JATTRI(1) NATURE DIFFUSE OU DISCRETE DU CHAMP
*                0: INDETERMINE  1: DIFFUS  2:DISCRET
*              JATTRI(2) LIBRE POUR L'INSTANT
*                0:              1:         2:
*     IPCHP  : POINTEURS SUR LES SEGMENTS MSOUPO
*     IFOPOI : CORRESPOND A L'OPTION IFOUR(VOIR CCOPTIO)
*     IGEOC  : POINTEUR SUR UN OBJET MELEME
*     NOCOMP : NOMS DES COMPOSANTES DU CHAMP
*     NOHARM : NUMERO DE L'HARMONIQUE CORRESPONDANT A LA COMPOSANTE
*              NOCOMP ,SI IFOPOI=1
*     IPOVAL : POINTEUR SUR LE SEGMENT MPOVAL
*     VPOCHA(I,J) : VALEUR DU CHAMP IEME POINT,JIEME COMPOSANTE
*    -END INCLUDE
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     OBJET DE TYPE 'BLOQ''STRU'
C
      SEGMENT/MCLSTR/(ISOSTR(N),IRIGCL(N)),MCLST1.MCLSTR,MCLST2.MCLSTR
C
C     ISOSTR(N)   : POINTEUR SUR UN SEGMENT MSOSTU (SOUS-STRUCTURE)
C     IRIGCL(N)   : POINTEUR SUR UN OBJET RIGIDITE (BLOCAGE)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
*        -INC SMCOORD
*
*      COORDONNEES DES POINTS ( POINTEUR DANS CCOPTIO )
*
       SEGMENT              MCOORD
       REAL*8               XCOOR((IDIM+1)*NBPTS)
       ENDSEGMENT
*
*    LES COORDONNEES SE SUIVENT ET NE SE RESSEMBLENT PAS
*    LA DENSITE EST LA (IDIM+1) COORDONNEE
*    D'OU
*    X =XCOOR((IP-1)*(IDIM+1)    +1)
*    Y =XCOOR((IP-1)*(IDIM+1)    +2)
*    Z =XCOOR((IP-1)*(IDIM+1)    +3)     SI IL Y A LIEU
*
*    D =XCOOR(IP*(IDIM+1))                IDIM EST DANS CCOPTIO
*
*    POUR DES RAISONS D'USAGE IL EST DEMANDE A NOS UTILISATEURS DE
*    NE JAMAIS DESACTIVE CE SEGMENT
*
```

```
*     -INC SMDEFOR
*
*      OBJET DEFORME  : DEFORMEE D'UNE STRUCTURE
*
       SEGMENT/MDEFOR/(AMPL(NDEF)*D,IELDEF(NDEF),ICHDEF(NDEF)
      # ,JCOUL(NDEF),MTVECT(NDEF)),MDEFO1.MDEFOR,MDEFO2.MDEFOR
*
*      AMPL   : TABLEAU DES COEFF DES DEFORMES
*      IELDEF : TABLEAU DES ELEMENTS DEFORMES
*      ICHDEF : TABLEAU DES CHAMPS DE DEFORMATION
*      JCOUL  : TABLEAU DES COULEURS DES DEFORMES
*      MTVECT : POINTEUR SUR UN OBJET DE TYPE VECTEUR
*
```

```
*       -INC SMELEME
*
*     L'OBJET  MAILLAGE  REPRESENTE UNE TOPOLOGIE
*
      SEGMENT      MELEME
      INTEGER      ITYPEL
      INTEGER      NUM(NBNN,NBELEM)
      INTEGER      LISOUS(NBSOUS),LISREF(NBREF)
      INTEGER      ICOLOR(NBELEM)
*     INTEGER      IGEOME(NBELEM)
      ENDSEGMENT
      POINTEUR     IPT1.MELEME,IPT2.MELEME,IPT3.MELEME,IPT4.MELEME
      POINTEUR     IPT5.MELEME,IPT6.MELEME,IPT7.MELEME,IPT8.MELEME
      POINTEUR     IPT9.MELEME
*
*     OBJET GEOMETRIQUE SIMPLE (UN SEUL TYPE D'ELEMENT)
*
*     ITYPEL   : NUMERO DU TYPE D'ELEMENT (D'APRES ILCOUR DANS CCGEOME)
*     NBSOUS   : 0 PAS DE SOUS-OBJETS
*     NBREF    : NOMBRE DE REFERENCES ( COTES FACES ....
*     LISREF   : LISTE DES REFERENCES (POINTEURS SUR MELEME
*     NBNN     : NOMBRE DE POINTS DANS L'ELEMENT
*     NBELEM   : NOMBRE D ELEMENTS DANS LE MELEME
*     NUM(I,J) : NUMERO DU IEME NOEUD DU JIEME ELEMENT
*     ICOLOR   : COULEUR DE CHAQUE ELEMENT
**    IGEOME   : POINTEUR SUR UNE GEOMETRIE SOUS-JACENTE A L'ELEMENT
*
*     OBJET GEOMETRIQUE COMPLEXE (PLUSIEURS TYPES D'ELEMENTS)
*
*     ITYPEL   : PAS UTILISE
*     NBSOUS   : NOMBRE DE SOUS-OBJETS (OBJETS GEOMETRIQUES SIMPLES)
*     LISOUS   : LISTE DES SOUS-OBJETS (POINTEURS SUR MELEME)
*     NBREF    : NOMBRE DE REFERENCES ( COTES FACES ....
*     LISREF   : LISTE DES REFERENCES (POINTEURS SUR MELEME
*     NBNN     : 0
*     NBELEM   : 0
*
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     OBJET DE TYPE 'ELEM''STRU'
C
      SEGMENT/MELSTR/(ISOSTU(N),IMELEM(N)),MELST1.MELSTR,MELST2.MELSTR
C
C     IMELEM(N)   : POINTEUR SUR UN OBJET MELEME (GEOMETRIE)
C     ISOSTU(N)   : POINTEUR SUR UN SEGMENT MSOSTU (SOUS-STRUCTURE)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
*
```

```
*  -INC SMEVOLL
*
      SEGMENT MEVOLL
       CHARACTER*8 ITYEVO
       CHARACTER*72 IEVTEX
       INTEGER IEVOLL(N)
      ENDSEGMENT
       POINTEUR MEVOL1.MEVOLL,MEVOL2.MEVOLL
*
      SEGMENT KEVOLL
       INTEGER IPROGX,IPROGY,NUMEVX
       CHARACTER*4  NUMEVY
       CHARACTER*8  TYPX,TYPY
       CHARACTER*12 NOMEVX,NOMEVY
       CHARACTER*72 KEVTEX
      ENDSEGMENT
       POINTEUR KEVOL1.KEVOLL,KEVOL2.KEVOLL
*
*  OBJET DE TYPE 'EVOLUTIO'
*  IEVOLL : POINTEUR SUR LE SEGMENT KEVOLL
*  ITYEVO : SOUS-TYPE  REEL     , COMPLEXE  ( 2 COURBES ) ...
*  IEVTEX : TITRE DE L'ENSEMBLE DES COURBES
*  IPROGX : POINTEUR SUR UN SEGMENT DE TYPE TIPX (ABSCISSES)
*
*  IPROGY : POINTEUR SUR UN SEGMENT DE TYPE TYPY (ORDONNEES)
*
*  NOMEVX : NOM AFFECTE AUX ABSCISSES
*  NOMEVY : NOM AFFECTE AUX ORDONNEES
*  NUMEVX : COULEUR DE LA COURBE
*  NUMEVY : TYPE DE LA COURBE REEL MODU PHAS PREE PIMA ...
*  KEVTEX : TITRE DE LA COURBE
*
```

```
*
*     DEFINITION DU TYPE 'LISTCHPO': SUITE D'OBJETS 'CHPOINT'.
*
      SEGMENT,MLCHPO
         INTEGER ICHPOI(N1)
      ENDSEGMENT
      POINTEUR MLCHP1.MLCHPO, MLCHP2.MLCHPO, MLCHP3.MLCHPO
*
*     "ICHPOI" CONTIENT DES POINTEURS SUR DES OBJETS 'CHPOINT'.
*
*     ATTENTION:  CE TYPE NE DEFINIT PAS UN TYPE D'OBJET MAIS UN TYPE
*                 DE "REGROUPEMENT HOMOGENE D'OBJETS".
*     CE QUI IMPLIQUE, EN PARTICULIER, QU'UNE INSTRUCTION DE LA FORME:
*        AA*LISTCHPO = OPERATEUR BB*CHPOINT ... ;
*     DANS LAQUELLE "AA" VA ETRE COMPLETE AVEC "BB" N'AUTORISE PAS LA
*     DESTRUCTION ULTERIEURE DE "BB", MEME SI "AA" EST SEUL MENTIONNE
*     DANS LES INSTRUCTIONS SUIVANTES.
*
*=====================================================================
```

```
C=====================================================================
C     OBJET DE TYPE 'LIST''ENTI'. DEFINIT UNE LISTE D'ENTIERS.
      SEGMENT/MLENTI/(LECT(JG)),MLENT1.MLENTI,MLENT2.MLENTI,
     1 MLENT3.MLENTI
C=====================================================================
```

```
*=====================================================================
*
*     OBJET DE TYPE 'LISTMOTS'. DEFINIT UNE LISTE DE MOTS.
*
      SEGMENT MLMOTS
       CHARACTER*(JGN) MOTS(JGM)
      ENDSEGMENT
      POINTEUR MLMOT1.MLMOTS,MLMOT2.MLMOTS,MLMOT3.MLMOTS
      POINTEUR MLMOT4.MLMOTS,MLMOT5.MLMOTS,MLMOT6.MLMOTS
*
*=====================================================================
```

```
C=====================================================================
C     OBJET DE TYPE 'LIST''REEL'. DEFINIT UNE LISTE DE REELS.
      SEGMENT/MLREEL/(PROG(JG)*D),MLREE1.MLREEL,MLREE2.MLREEL,
     1 MLREE3.MLREEL
C=====================================================================
```

```
C***********************************************************************
*     -INC SMMODEL
*
*      OBJET DE TYPE "MODELE"
*
       SEGMENT,MMODEL
          INTEGER KMODEL(N1)
       ENDSEGMENT
        POINTEUR MMODE1.MMODEL,MMODE2.MMODEL
*
       SEGMENT,IMODEL
          INTEGER IMAMOD,NEFMOD
          INTEGER INFMOD(MN3)
          CHARACTER*16 CONMOD
          CHARACTER*16 FORMOD(NFOR),MATMOD(NMAT)
       ENDSEGMENT
        POINTEUR IMODE1.IMODEL,IMODE2.IMODEL
*
*      KMODEL : POINTEUR SUR DES OBJETS MODELES ELEMENTAIRES
*      IMAMOD : POINTEUR SUR UN OBJET MAILLAGE ELEMENTAIRE
*      NEFMOD : NUMERO DE L'ELEMENT FINI DANS NOMTP
*      INFMOD : INFORMATIONS VARIEES RELATIVES AU MODELE
*               ( VIDE PAR DEFAUT )
*      INFMOD(1) : NOMBRE DE PTS D'INTEGRATION DANS L'EPAISSEUR
*
*      CONMOD : NOM DU CONSTITUANT (BLANC PAR DEFAUT )
*      FORMOD : SUITE DE MOTS DEFINISSANT LA FORMULATION
*               (EX: THERMIQUE , LIQUIDE + MECANIQUE, ...)
*      MATMOD : SUITE DE MOTS CARACTERISANT LE TYPE DE MATERIAU
*               (EX: ORTHOTROPE, PLASTIQUE PARFAIT, ...)
```

```
* -INC SMNUAGE
*
*     OBJET DE TYPE 'NUAGE'
*
      SEGMENT MNUAGE
       CHARACTER*8 NUANOM(NVAR)
       CHARACTER*8 NUATYP(NVAR)
       INTEGER  NUAPOI(NVAR)
      ENDSEGMENT
      SEGMENT NUAVIN
       INTEGER NUAINT(NBCOUP)
      ENDSEGMENT
      SEGMENT NUAVFL
       REAL*8 NUAFLO(NBCOUP)
      ENDSEGMENT
      SEGMENT NUAVMO
       CHARACTER*8 NUAMOT(NBCOUP)
      ENDSEGMENT
      SEGMENT NUAVLO
        LOGICAL NUALOG(NBCOUP)
      ENDSEGMENT
       POINTEUR MNUAG1.MNUAGE,MNUAG2.MNUAGE,MNUAG3.MNUAGE
       POINTEUR NUAVI1.NUAVIN,NUAVI2.NUAVIN,NUAVI3.NUAVIN
       POINTEUR NUAVF1.NUAVFL,NUAVF2.NUAVFL,NUAVF3.NUAVFL
       POINTEUR NUAVL1.NUAVLO,NUAVL2.NUAVLO,NUAVL3.NUAVLO
       POINTEUR NUAVM1.NUAVMO,NUAVM2.NUAVMO,NUAVM3.NUAVMO
*
*     NVAR      : NOMBRE DE VARIABLES
*     NBCOUP    : NOMBRE DE POINTS
*     NUANOM    : TABLEAU DES NOMS DES VARIABLES
*     NUATYP    : TABLEAU DES TYPES DES VARIABLES
*     NUAPOI    : TABLEAU DE POINTEUR VERS NUAVIN OU NUAVFL OU NUAVMO
*     NUAVAR    : TABLEAU DE POINTEURS SUR LES VARIABLES
*     NUAFLO    : TABLEAU DES VALEURS SI FLOTTANT
*     NUAMOT    : TABLEAU DES VALEURS SI MOT
```

```
*
*         -INC SMRIGID
*
*      OBJET RIGIDITE
*
      SEGMENT   MRIGID
        CHARACTER*8       MTYMAT
        REAL*8            COERIG(NRIGEL)
        INTEGER           IRIGEL(NRIGE,NRIGEL)
        INTEGER           ICHOLE,IMGEO1,IMGEO2,IFORIG
        INTEGER           ISUPEQ
      ENDSEGMENT
      POINTEUR  RI1.MRIGID,RI2.MRIGID,RI3.MRIGID
      POINTEUR  RI4.MRIGID,RI5.MRIGID,RI6.MRIGID
*
      SEGMENT   XMATRI
        REAL*8            RE(NLIGRD,NLIGRP)
      ENDSEGMENT
      POINTEUR  XMATR1.XMATRI,XMATR2.XMATRI,XMATR3.XMATRI
      POINTEUR  XMATR4.XMATRI,XMATR5.XMATRI,XMATR6.XMATRI
*
      SEGMENT   IMATRI
        INTEGER             IMATTT(NELRIG)
      ENDSEGMENT
      POINTEUR  IMATR1.IMATRI,IMATR2.IMATRI,IMATR3.IMATRI
      POINTEUR  IMATR4.IMATRI,IMATR5.IMATRI,IMATR6.IMATRI
*
      SEGMENT   DESCR
        CHARACTER*4         LISINC(NLIGRP),LISDUA(NLIGRD)
        INTEGER             NOELEP(NLIGRP),NOELED(NLIGRD)
      ENDSEGMENT
      POINTEUR  DES1.DESCR
*
      SEGMENT   IMGEOD
        INTEGER             IMGEOR(NBGEOR)
      ENDSEGMENT
*   NRIGEL        : NOMBRE D'OBJET ELEMENTAIRE DE MRIGIDITE
*   NLIGRP        : NOMBRE D'INCONNUES PRIMALES D'UNE MATRICE
*                   ELEMENTAIRE.
*   NLIGRD        : NOMBRE D'INCONNUES DUALES D'UNE MATRICE
*                   ELEMENTAIRE.
*   COERIG(I)     : COEFFICIENT MULTIPLICATEUR
*   IRIGEL(1,I)   : POINTEUR SUR L'OBJET GEOMETRIE
*   IRIGEL(2,I)   : POINTEUR SUR UN OBJET GEOMETRIQUE (CAS FROTTEMENT)
*   IRIGEL(3,I)   : POINTEUR SUR LE SEGMENT DESCRIPTIF D'UNE
*                        MATRICE ELEMENTAIRE.(SEGMENT DESCR)
*   IRIGEL(4,I)   : POINTEUR SUR LE SEGMENT CONTENANT LES POINTEURS
*                        DES MATRICES DE MRIGIDITE DE CHAQUE ELEMENTS.
*                        (SEGMENT IMATRI)
*   IRIGEL(5,I)   : NUMERO DE L'HARMONIQUE DE FOURIER
*   IRIGEL(6,I)   : NATURE DE LA RELATION DEFINISSANT LA RIGIDITE
```

```
*                  0  EGALITE
*                 -1  INEGALITE INFERIEURE
*                 +1  INEGALITE SUPERIEURE
*   IRIGEL(7,1)   : 0   LA MATRICE EST SYMETRIQUE
*                 : 1   LA MATRICE EST ANTISYMETRIQUE
*                 : 2   LA MATRICE NE POSSEDE PAS DE SYMETRIES
*   RE(I,J )      : LISTE DE VALEURS DE LA MATRICE ELEMNTAIRE
*   IMATTT(I)=IPO : IPO EST LE POINTEUR SUR LE SEGMENT DE TYPE
*                        XMATRI CONTENANT LA MATRICE DE MRIGIDITE DU
*                        I EME ELEMENT DE L'OBJET MRIGIDITE CONSIDERE.
*   NOELEP(I)=J   : LA I EME INCONNUE PRIMAL DE LA MATRICE PORTE SUR
*                   LE J EME NOEUD DE L'ELEMENT.
*   LISINC(I)=INO : LA I EME INCONNUE PRIMAL DE LA MATRICE EST DE
*                   TYPE INO
*   NOELED(I)=J   : LA I EME INCONNUE DUALE  DE LA MATRICE PORTE SUR
*                   LE J EME NOEUD DE L'ELEMENT .
*   LISDUA(I)=INO : LA I EME INCONNUE DUALE DE LA MATRICE EST DE
*                   TYPE INO
*
*   MTYMAT         : TYPE DE LA MATRICE RIGIDITE OU MASSE
*
*   ICHOLE        : SI DIFFERENT DE ZERO EST EGAL AU POINTEUR
*                        SUR SEGMENT DE TYPE MMATRI(VOIR SMMATRI)
*                        CONTENANT LA MATRICE DEJA TRIANGULARISEE.
*   ISUPEQ        : POINTEUR EVENTUEL SUR UNE TABLE (UNILATERAL)
*   IMGEO1        : UNE RESOLUTION A DEJA EU LIEU.POINTEUR SUR
*                        SUR UN TABLEAU CONTENANT LES POINTEURS DES
*                        OBJETS GEOMETRIQUES CREES.(TABLEAU DANS IMGEOD)
*   IMGEO2        : POINTEUR SUR UN PROTOTYPE DU CHAMPOINT DUAL
*   IFORIG        : CORRESPOND A L'OPTION IFOUR (VOIR CCOPTIO)
*     LA MATRICE ELEMENTAIRE  EST ORGANISEE COMME BON VOUS SEMBLE ELLE
*     EST SEULEMENT TRIANGULAIRE INFERIEURE STOCKEE LIGNE PARLIGNE
*
```

```
*  -INC SMSOLUT
*
*     OBJET DE TYPE'SOLUTION'
*
      SEGMENT MSOLUT
       CHARACTER*8 ITYSOL
       INTEGER     MSOLIS(NIPO),MSOLIT(NIPO)
      ENDSEGMENT
       POINTEUR MSO1.MSOLUT,MSO2.MSOLUT
*
      SEGMENT MMODE
       REAL*8      FMMODD(LVALM)
       INTEGER     IMMODD(NIMOD)
      ENDSEGMENT
       POINTEUR MMOD1.MMODE
*
      SEGMENT MSOLRE
       REAL*8      SOLRE(N)
      ENDSEGMENT
       POINTEUR MSOLR1.MSOLRE,MSOLR2.MSOLRE
*
      SEGMENT MSOLEN
       INTEGER     ISOLEN(N)
      ENDSEGMENT
       POINTEUR MSOLE1.MSOLEN,MSOLE2.MSOLEN
*
*
*   ITYSOL  :  SOUS TYPE DE L'OBJET SOLUTION
*   MSOLIS(I) : POINTEUR SUR DIFFERENTS SEGMENTS (SUIVANT LA VALEUR
*               DE ITYSOL ET CELLE DE I)
*   MSOLIT(I) : DESIGNE LE TYPE DES OBJETS CONTENUS DANS LE MSOLEN
*               DE POINTEUR MSOLIS(I).
*               2 POUR LES CHPOINT , 5 POUR LES CHAMELEM ,
*              14 POUR LES MJONCT...
*               LA CORRESPONDANCE TYPE DES OBJETS--NUMERO EST CONTENUE
*               DANS LE SOUS PROGRAMME TYPFIL
*
*
*   * I=1,4 :  ZONE COMMUNE A TOUS LES ITYSOL :
*     ------------------------
*     3.  POINTEUR SUR UN MELEME                                  0
*     4.  POINTEUR SUR MSOLEN (LISTE DE MMODE )                   0
*
*   * ITYSOL = MODE
*     -------------
*     5.  POINTEUR SUR MSOLEN (DEPLACEMENTS   )         MSOLIT = 2
*     6.          ID.         (CONTRAINTES    )                  5
*     7.          ID.         (VON MISES      )                  5
*     8.          ID.         (VITESSES       )                  2
*     9.          ID          (ACCELERATIONS  )                  2
*
```

```
*   * ITYSOL = SOLUSTAT OU PSEUMODE
*     ----------------------------
*     5.  POINTEUR SUR MSOLEN (DEPLACEMENTS   )          MSOLIT = 2
*     6.          ID.         (CONTRAINTES    )                   5
*     7.          ID.         (VON MISES      )                   5
*     8.          ID.         (VITESSES       )                   2
*     9.          ID.         (ACCELERATIONS  )                   2
*     10.         ID.         (MJONCT         )                   14
*
*   * ITYSOL = DYNAMIQU
*     -----------------
*     1.  POINTEUR SUR MSOLRE (LISTE DES TEMPS)          MSOLIT = 0
*     2.  POINTEUR SUR MSOLEN (LISTE DES PAS  )                   0
*     5.          ID.         (DEPLACEMENT    )                   2
*     6.          ID.         (CONTRAINTES    )                   5
*     7.          ID.         (VONMISES       )                   5
*     8.          ID.         (VITESSES       )                   2
*     9.          ID.         (ACCELERATIONS  )                   2
*     10.         ID.         (VARIABLES DE LIAISON)              2
*     13.         ID.         (VARIABLES D'USURE)                 2
*-------------------------------------------------------------------------
```

```
C=====================================================================
C     OBJET DE TYPE 'STRU''CTUR'
C
      SEGMENT/MSTRUC/(LISTRU(N)),MSTRU1.MSTRUC,MSTRU2.MSTRUC
      SEGMENT/MSOSTU/(ITYSOU,ISRAID,ISMASS,ISCHAM(NS)),MSOST1.MSOSTU,MSOS
     &T2.MSOSTU
C
C     LISTRU : POINTEURS SUR SEGMENT MSOSTU
C     ITYSOU : TYPE DE LA STRUCTURE
C     ISRAID : POINTEUR SUR UN OBJET DE TYPE RIGIDITE
C     ISMASS : POINTEUR SUR UN OBJET DE TYPE RIGIDITE ( MASSE )
C     ISCHAM(NS) : POINTEURS SUR LES NS CHAMELEM ASSOCIES A LA STRUCTURE
C=====================================================================
```

```
C=====================================================================
C  OBJET DE TYPE      'SUPERELE'
      SEGMENT ,MSUPER
           INTEGER MRIGTO,MSUPEL,MSURAI,MBLOQU,MSUMAS,MCROUT
      ENDSEGMENT
      POINTEUR MSUPE1.MSUPER
C
C    MRIGTO : POINTEUR SUR UN OBJET RIGIDITE CONTENANT L'OBJET RIGIDITE
C             LU ET LES BLOQUAGES SUPPLEMENTAIRES SUR LES NOEUDS MAITRES.
C
C    MSUPEL : POINTEUR D'UN OBJET GEOM DONT LES ELEMENTS, DE TYPE POINT,
C             REPRESENTENT LES NOEUDS MAITRES.
C
C    MSURAI : POINTEUR SUR L'OBJET RIGIDITE EQUIVALLENTE.
C
C    MBLOQU : PRECISE QUE LES N PREMIERS RIGIDITE DE MRIGTO SONT LES
C             BLOQUAGES CREES POUR LE SUPER ELEMENT
C
C    MSUMAS : POINTEUR SUR LA MASSE EQUIVALLENTE
C
C    MCROUT : POINTEUR SUR UN OBJET MATRICE CONTENANT LA DECOMPOSITION
C             DE CROUT MODIFIEE
```

```
*        -INC SMTABLE
*
*     OBJET DE TYPE TABLE
*
      SEGMENT MTABLE
            INTEGER       MLOTAB
            CHARACTER*(8) MTABTI(M),MTABTV(M)
            REAL*8        RMTABI(M)
            INTEGER       MTABII(M),MTABIV(M)
            REAL*8        RMTABV(M)
      ENDSEGMENT
      POINTEUR MTAB1.MTABLE,MTAB2.MTABLE,MTAB3.MTABLE
*
*     MLOTAB     : LONGUEUR DE LA TABLE ( <> ALLOCATION )
*     MTABTI(I)  : TYPE DE L INDICE POINTANT SUR LE IEME ELEMENT DE LA
*                :    TABLE
*     MTABTV(I)  : TYPE DE L'OBJET POINTE PAR LE IEME INDICE
*     MTABII(I)  : VALEUR DE L'INDICE  (SI PAS REEL)
*     RMTABI(I)  : VALEUR DE L'INDICE  (SI REEL)
*     MTABIV(I)  : VALEUR DE L'OBJET   (SI PAS REEL)
*     RMTABV(I)  : VALEUR DE L'OBJET   (SI REEL)
*
```

```
C=====================================================================
C
C    OBJET DE TYPE 'TEXT','E   '
      SEGMENT , MTEXTE
        CHARACTER MTEXT*72
        INTEGER   NCART
        INTEGER   MTRADC
      ENDSEGMENT
        POINTEUR MTEXT1.MTEXTE , MTEXT2.MTEXTE
      SEGMENT , MTRADU
        INTEGER  MTRAD(0)
      ENDSEGMENT
C
C=====================================================================
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       SEGMENT/MVECTD/(VECTBB(INC)*D),MVECT1.MVECTD,MVECT2.MVECTD,
     1 MVECT3.MVECTD
C
C      OBJET DE TYPE 'VECT''DOUB'
C      LISTE DE REELS DOUBLE PRECISION
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
*          -INC SMVECTE
*
*     OBJET DE TYPE VECTEUR
*
      SEGMENT MVECTE
       CHARACTER*4 NOCOVE(NVEC,ID)
       INTEGER IGEOV(NVEC),NOCOUL(NVEC),ICHPO(NVEC)
       REAL*8  AMPF(NVEC)
      ENDSEGMENT
       POINTEUR MVECT1.MVECTE,MVECT2.MVECTE,MVECT3.MVECT
*
*     IGEOV :  POINTEUR SUR OBJET MELEME ( NON UTILISE)
*     AMPF  :  AMPLIFICATION
*     NOCOUL:  COULEUR
*     ICHPO :  TABLEAU DES CHAMPOINTS
*     NOCOVE:  NOM DES COMPOSANTES
*
```

Chapitre 4

Les opérateurs

4.1 Introduction

La structure informatique de Cast3M entraine qu'il n'est pas nécessaire de connaitre les autres opérateurs pour pouvoir en introduire un nouveau, ceux-ci sont en effet orthogonaux les uns aux autres. Par contre la communication avec le monde externe, via l'analyseur de syntaxe est absolument primordiale de même que la connaissance des structures d'informations (les objets) concernées par l'opérateur.

Rappelons que la syntaxe de la commande élémentaire dans Cast3M est :

```
RESULTAT = OPERATEUR OPERANDES ;
```

Dans ce manuel nous regarderons le branchement de l'opérateur dans le gestionnaire de commandes puis nous préciserons quelques interfaces avec le monde externe ainsi que certaines règles de fonctionnement de l'analyseur de syntaxe.

4.2 Branchement d'un opérateur

Le sous-programme PILOT est le seul qui doit être modifié. C'est lui qui connait la liste des opérateurs et qui leur donne le contrôle si l'analyseur de syntaxe a détecté le nom de l'opérateur dans la commande de l'utilisateur. Schématiquement un tableau de chaines de caractères de 4 lettres, initialisé par data, contient la liste des opérateurs. L'appel au sous-programme LIRMO2 retourne la position, dans le tableau de chaines de l'opérateur invoqué par la commande de l'utilisateur.Il suffit alors d'utiliser un GOTO indicé pour se brancher sur le sous-programme point d'entrée de l'opérateur.

4.2.1 Réalisation pratique

La liste du sous-programme PILOT est longue mais on peut en extraire ces quelques lignes.

```
  SUBROUTINE PILOT

  PARAMETER(NDIR3=49)
  PARAMETER(NDIR2=178)
  PARAMETER(NDIR1=179,NDIR1P=....,NDIR=NDIR1+NDIR2+NDIR3)
  CHARACTER*4 MDIR(NDIR),MDIR1(NDIR1),...,MDIR3(NDIR3)
  EQUIVALLENCE (MDIR(1),MDIR1(1)),(MDIR(NDIR1P)=MDIR2(1)),...
  DATA MDIR1/'OPTI','FIN ','DUMP',...
    .
    .
  DATA MDIR2/'COPI','DIMN','SAUV',...
```

```
      .
     DATA MDIR3/'NUAG','WEIP','KHIS','...
     *'KPRO','FFOR','RAYE','RAYN','VSUR','TRAJ'/
C l'ecriture compliquee avec plusieurs tableaux en equivalence
C n'est justifiee que par des raisons strictement informatiques
       .
       .
     CALL LIRMO2(MDIR,NDIR,0,I,ICOHCO)
     IF (I.NE.0) GO TO 100
       .
       .
100  LOCERR=MDIR(I)
       .
     IF(I.LE.100)
       .
     IF(I.LE.200)
       .
     IF(I.LE.300)
       .
     IF(I.LE.400)
       .
     IF (I.LE.500)
     *GOTO (501,502,503,504,505,506),I-400
       .
       .
506  CALL TRAJEC
GO TO 1
END
```

Les seules lignes à modifier pour introduire l'opérateur de nom TITI et de sous-programme principal TITISP sont :

```
C incrementation de 1 de NDIR3
     PARAMETER(NDIR3=50)
       .
C  ajout du nom TITI dans la liste des operateurs
     *'KPRO','FFOR','RAYE','RAYN','VSUR','TRAJ','TITI'/
       .
C ajout d'un indice dans la liste du goto indice
     *GOTO (501,502,503,504,505,506{\bf ,507}),I-400
       .
```

Enfin il faut ajouter juste avant l'instruction END les deux lignes :

```
507  CALL TITISP
     GO TO 1
     END
```

La seule précaution à prendre est de vérifier que le nom TITI n'est pas déjà le nom d'un opérateur.

Le gestionnaire de commandes fera en sorte que le sous-programme TITISP soit appelé si la commande invoque l'opérateur TITI. Il faut maintenant écrire la chaine de sous-programmes appelée par TITISP.

4.3 Interface opérateur-monde externe

L'opérateur a besoin de savoir sur quel objet il doit travailler et si la commande a des options particulières. Il lui faut donc connaitre les opérandes fournis par l'utilisateur dans sa commande. Ensuite, une fois le travail effectué l'opérateur devra rendre son ou ses résultats au monde externe.

4.3.1 Acquisition d'un objet

La première action d'un opérateur est souvent d'acquérir les structures de données (OBJET) à partir desquelles il travaillera. Pour cela les sous-programmes LIR*** ont été créés. En étant exhaustif, on peut citer :

1. SUBROUTINE LIRENT (IVAL,ICOND,IRETOU) Lecture d'un entier
2. SUBROUTINE LIRREE (XVAL,ICOND,IRETOU) Lecture d'un flottant
3. SUBROUTINE LIRCHA (CHAR,ICOND,IRETOU) Lecture d'une chaine
4. SUBROUTINE LIRLOG (LOGI,ICOND,IRETOU) Lecture d'un logique
5. SUBROUTINE LIROBJ (MTYP,IRET,ICOND,IRETOU) Lecture d'un objet
6. SUBROUTINE LIRTAB (MTYP,IRET,ICOND,IRETOU) Lecture d'une table sous-typée

***** ICOND est un entier de valeur 0 ou 1 précisant si la présence de l'objet est obligatoire ou non. Un message d'erreur est émis par le lecteur en cas de besoin, de plus la variable IERR du COMMON COPTIO prend une valeur différente de zéro.

***** IRETOU est un entier de valeur 1 ou 0 précisant en retour de la lecture si la demande a pu être réalisée avec succès ou non. Dans le cas de la lecture d'une chaine IRETOU contient la longueur de la chaine lue.

***** IVAL est un entier qui contient en retour la valeur lue.

***** XVAL est un REAL*8 qui contient en retour la valeur lue.

***** CHAR est une chaine de caractères, elle contient en retour, cadrée à gauche et tronquée ou complétée par des blancs, la chaine lue.

***** LOGI est un logique (variable fortran LOGICAL)

***** MTYP est une chaine de 8 caractères précisant le type de l'objet attendu (ou le sous-type de l'objet TABLE si appel à LIRTAB).

***** IRET est la valeur associée à l'objet lu. Dans le cas d'objet crée par les opérateurs de Cast3M, c'est la valeur transmise lors de l'enregistrement de l'objet par les sous-programmes ECR*** qui suivent.

4.3.2 Écriture d'un objet

Le nom des sous-programmes évoque l'écriture parce qu'il s'agit en fait d'écrire dans la pile des objets qui sont en attente de lecture. Les sous-programmes ECR***, qui servent aussi à rendre un résultat, sont :

1. SUBROUTINE ECRENT(IVAL) Pour rendre un entier
2. SUBROUTINE ECRREE(XVAL) Pour rendre un flottant
3. SUBROUTINE ECRCHA(CHAR) Pour rendre une chaine de caractères
4. SUBROUTINE ECRLOG(LOGI) Pour rendre un logique
5. SUBROUTINE ECROBJ(MTYP,IRET) Pour rendre un objet de type MTYP et de valeur associée IRET.

Les variables ont la même signification que précédemment.

Pour lire ou écrire dans une table il faut utiliser les sous-programme ACCTAB ou ECCTAB qui ne font pas, à proprement parler, partie de l'analyseur de syntaxe. Ces deux sous-programmes sont détaillés dans la partie concernant les OBJETS.

4.3.3 Reconnaissance du type de la donnée

Certains opérateurs généraux peuvent traiter des objets de types variés. Un sous-programme permet de connaitre à l'avance le type de la prochaine donnée à lire.

– SUBROUTINE QUETYP (MTYP,ICOND,IRETOU)

En retour MTYP contient sur 8 caractères le type du premier objet en attente de lecture. Icond et IRETOU ont la même signification que précédemment.

4.3.4 Lecture d'une chaine parmi une liste

Dans un opérateur, la lecture de mots-clés optionnels peut facilement se faire à l'aide de LIRMOT.

– SUBROUTINE LIRMOT (LISMO,NBMOT,IRAN,ICOND)

***** LISMO est un tableau de NBMOT chaines de caractères.

***** IRAN est le rang de la chaine lue dans le tableau LISMO.

***** ICOND demande ou non un message d'erreur si la demande n'a pas pu être satisfaite.

4.4 Remarques

– Pour en savoir plus nous recommandons le rapport : « Langage GIBIANE : définition » DEMT/89-108 (il est repris dans le guide du développement)
– Si un opérateur rend plusieurs résultats, il faut les écrire en commençant par le dernier (pile LIFO).
– Le lecteur explore les données tant qu'il n'a pas rencontré un objet du type désiré ou tant qu'il ne rencontre pas une chaine de caractère. Ceci permet la lecture non positionnelle des objets de type différents.
– L'accès aux informations contenues dans les objets de type TABLE est faite par le sous-programme ACCTAB et la mise d'informations dans un objet TABLE se fait à l'aide de ECCTAB.

4.5 Exemple complet et message d'erreur

Le common « COPTIO » que l'on peut mettre dans un sous-programme par l'instruction :
-INC CCOPTIO
contient certaines informations générales. En particulier la dimension de l'espace, les unités d'entrées-sorties et la communication avec le gestionnaire d'erreurs.
Ainsi la variable IERR, contenue dans le common, représente le niveau d'erreur atteint pendant l'exécution de la commande élémentaire. Les tableaux INTERR, REAERR et la chaine MOTERR permettent de communiquer avec les messages d'erreurs contenus dans le fichier gibi.erreur.
Nous nous proposons de faire un opérateur qui lit une syntaxe :
FLOT1 MOT1 FLOT2
Dans cet exemple MOT1 sera soit le mot SUPERIEUR soit le mot INFERIEUR. L'opérateur rendra un logique vrai si la phrase est vrai.

```
  SUBROUTINE EXEMPL
  IMPLICIT REAL*8(A-H,O-Z)
-INC CCOPTIO
  CHARACTER*9 MOT(2)
  DATA MOT/'SUPERIEUR','INFERIEUR'/
  CALL LIRMOT(MOT,2,0,ICAS)
  IF(ICAS.EQ.0)  THEN
    MOTERR(1:9) = MOT(1)
    MOTERR(10:18) = MOT(2)
    CALL ERREUR ( 777)
```

```
    RETURN
  ENDIF
  CALL LIRREE(XV1,1,IRETOU)
  IF(IERR.NE.0) RETURN
  CALL LIRREE(XV2,1,IRETOU)
  IF(IERR.NE.0) RETURN
  IF( ICAS.EQ.1) THEN
    IF ( XV1.GT.XV2) THEN
      CALL ECRLOG(.TRUE.)
    ELSE
      CALL ECRLOG(.FALSE.)
    ENDIF
  ELSE
    IF ( XV1.GT.XV2) THEN
      CALL ECRLOG(.FALSE.)
    ELSE
      CALL ECRLOG(.TRUE.)
    ENDIF
  ENDIF
  RETURN
  END
```

Le message d'erreur 777 est supposé faire une impression qui ressemble à :
On attend un des mots : SUPERIEUR ou INFERIEUR
Le tableau INTERR permet de passer des valeurs entières et REAERR des valeurs réelles. Dans le fichier gibi.erreur, un message d'erreur est composé de deux ou quatre lignes. La première et la troisième contiennent le numéro de l'erreur et son niveau (le niveau sert à initialiser la variable IERR et à partir de 2 le lecteur interne passe à l'instruction suivante en sortant du bloc REPETER ou de la procédure). La deuxième et la quatrième ligne servent à définir le message qui va être imprimé. Un numéro d'erreur négatif est utilisé pour les messages normaux, par exemple pour les textes des listes d'objets.Ci-joint l'en-tête du fichier et un ou deux extraits.

```
9999 0 C GIBI      ERREUR    CHAT      96/03/18    21:15:02     2086
Liste des messages a traduire pour castem
9999 0
une ligne introduite par 9999 est commentaire: par 9998 est la
9999 0
liste des langues: par 9997 est la def de la langue  : par
9999 0
 9996 la fin du fichier 9995 introduit la langue utilisee par defaut
9999 0
Format du fichier : lrecl=80
9999 0
 premiere ligne numero message I4,1x,niveau erreur I1
9999 0
deuxieme ligne texte A80
9999 0
prendre un numero inferieur ou superieur ( consecutif)
9999 0
%i1 represente interr(1)  common coptio
9999 0
```

```
%r3 represente reaerr(3)  common coptio
9999 0
%m3:8 represente moterr(3:8)  common coptio
9998 0 ceci est la liste des langues utilisees(FRAN en premier)
FRAN ANGL
9995 0  langue par defaut a changer si besoin est
FRAN
9997 0 ceci est la langue qui va etre utilisee
FRAN
-307 0
Matrice elementaire non-symetrique
-306 0
*****     FIN DE LA PROCEDURE INITIALE    ******
-305 0
*****  EXECUTION D'UNE PROCEDURE INITIALE ******
-304 0
ATTENTION : On ne prend pas en compte la moyenne pour un bruit blanc
-304 0
a distribution exponentielle (valeur lue : %r1 )
-303 0
Impossible de regenerer l'element %i5 ayant deux fois le meme noeud.
-302 0
Liste des procedures surchargees par l'utilisateur :
-302 0
**************************************************


 188 2
On cherche un chpoint qui contient des contributions modales
 189 2
Pas de frequence propre dans l'intervalle fourni
 190 2
On veux lire un entier superieur ou egal a %i1 au lieu de %i2
 191 2
On veux lire un flottant superieur ou egal a %r1 au lieu de %r2
 192 2
Impossible d'orienter les forces de pression: direction donnee
 192 2
perpendiculaire a la force de pression au point indique
 193 2
Impossible d'utiliser cet operateur pour la formulation %m1:8
 194 2
Impossible de calculer les contraintes pour la formulation %m1:8
 195 2
Changement de signe du jacobien dans l'element %i1. Maillage incorrect
 196 2
Le type d'un des operandes n'est pas compatible avec l'operateur %m1:8
 197 2
Le mot %m1:4 n'est pas un nom de composante reconnu
```

Chapitre 5

Implantation d'une nouvelle loi d'évolution / loi de comportement mécanique

5.1 Préambule

Le logiciel Cast3M est écrit en langage Fortran77 auquel ont été ajoutées quelques instructions. Ce nouveau langage est baptisé ESOPE et sa structure est identique à celle du Fortran. Il existe donc des sous-programmes qui s'appellent les uns les autres. En pratique la connaissance de ESOPE n'est pas demandée pour implanter une nouvelle loi simple, mais peut s'avérer utile pour des lois très compliquées. De plus Cast3M propose à l'utilisateur un langage dénommé GIBIANE qui permet de créer et manipuler des OBJETS, avec lequel on peut créer des procédures encapsulant des séquences de calcul. L'utilisation de ces procédures conserve aux jeux données proprement dits, exprimés en GIBIANE, un caractère paramétrique et synthétique.
A l'installation du logiciel, un répertoire est créé qui contient l'ensemble des fichiers source. Ainsi, le sous-programme fortran(esope), de nom ABCDEF, se trouve dans le fichier **abcdef.eso**. C'est pourquoi nous parlerons par la suite de sous-programme **abcdef.eso**. Le suffixe **.eso** est donné pour préciser que le langage est ESOPE et qu'il faudra lui appliquer la moulinette de traduction en pur Fortran77.
A priori pour compiler un sous-programmes il faut passer la commande :

```
compil abcdef
```

Ceci compile le contenu du fichier **abcdef.eso** après l'avoir traduit en pur Fortran77.
Pour faire un load module exécutable de Cast3M il suffit ensuite de passer la commande :

```
essai
```

On est conduit à travailler dans un répertoire contenant des sous-programmes modifiés et/ou créés pour le développement. L'exécution de Cast3M par la commande :

```
castem aa.dgibi
```

dans le répertoire contenant ce nouvel exécutable entrainera l'utilisation du load module. L'opérateur `UTIL` permet d'utiliser des procédures crées ou modifiées rassemblées dans un fichier **mesprocedures**, au cours de la prochaine exécution de Cast3M :

```
util proc 'mesprocedures' ;
fin ;
```

5.2 Introduction

Pour des raisons de régularité et de généralité, un nouvel opérateur COMP — comportement — remplace l'opérateur ECOU — écoulement. Ce document annule et remplace le rapport [1].
L'implantation d'une nouvelle loi d'évolution et en particulier d'une loi de comportement mécanique nécessite la création d'un nouvel exécutable cast obtenu à partir de la modification et/ou l'ajout d'un certain nombre de sous-programmes, mais également la mise à jour de procédures de calculs telles PASAPAS, et bien entendu la définition de jeux de données moncas.dgibi appropriés. Un tel travail s'articule autour des points suivants, repris dans la construction de ce document :

- 1^{eme} *phase* : définition d'un nouveau modèle de matériau — type de formulation ;
- 2^{eme} *phase* : description des champs physiques participants à la loi — en mécanique, déformations, contraintes, caractéristiques du matériau, variables internes, ... ;
- 3^{eme} *phase* : calcul incrémental.

Les deux premières phases servent à initialiser le logiciel pour qu'il soit capable automatiquement de récupérer les données fournis par l'utilisateur : le développement porte sur les opérateurs MODE, MATE, ... La troisième phase correspond à la programmation de la loi proprement dite dans l'opérateur COMP — qui remplace l'opérateur ECOU —, et à la mise au point des procédures qui facilitent l'utilisation de la loi — par exemple PASAPAS.

Trois possibilités sont proposées pour l'implantation :

1. description des champs dans des sous-programmes — listes des noms des composantes — : pour les phases 1 et 2 rien n'est changé vis-à-vis de l'opérateur ECOU [1]. Puis modification des sous-programmes de COMP avec appel éventuel à de nouveaux sous-programmes dans **coml7.eso** ou **coml8.eso**. il peut être nécessaire de modifier PASAPAS pour adresser les données convenables à COMP.
2. modèles VISCO_EXTERNE : en mécanique, un intégrateur dédié au fluage polynomial permet de traiter des expressions où interviennent des champs paramètres et/ou des variables internes non-standards [5, 6]. L'intégrateur est contenu dans le sous-programme **ccreep.eso**. Les listes des noms de composantes de ces paramètres "externes" et de ces variables internes sont associées à l'objet MMODEL dans le jeu de données (voir exemple creep04_cisXY.dgibi). Les champs de paramètres externes servent à composer le chargement figurant dans la table argument de la procédure PASAPAS ; celle-ci sait gérer le calcul a priori sans modification.
3. modèle NON_LINEAIRE UTILISATEUR : en mécanique, Cast3M offre la possibilité d'exploiter des sous-programmes fortran77 au standard UMAT d'ABAQUS, permettant de traiter des comportements très variés [5, 6]. La description des composantes matériaux, des noms de variables internes et des noms de champs de paramétrages est possible dans le fichier utilisateur (voir exemple umat05.dgibi). Il faut modifier le contenu du sous-programme **umat.eso** qui existe dans Cast3M. La procédure PASAPAS sait gérer le calcul a priori sans modification .

Les variantes 2 et 3 — réservées à la mécanique — dispensent de réaliser les phases 1 et 2 du développement et conduisent à intervenir en phase 3 sur des sous-programmes spécifiques : les lois de comportement mécanique complexes seront difficiles à réaliser dans ce cadre. La variante 1 offre beaucoup plus de possibilités.
Ce document essaye de donner un aperçu sur l'organisation du logiciel, il donne donc parfois des informations qui ne sont pas strictement nécessaires à l'implantation d'une nouvelle loi de comportement.

5.3 Phase 1 - Définition d'un nouveau modèle de matériau

Il est nécessaire de mettre à jour l'opérateur MODE, qui crée un objet MMODEL, association d'une géométrie (objet MAILLAGE), d'une formulation physique (ex : MECANIQUE), d'un comportement linéaire (ex :

ELASTIQUE ANISOTROPE), d'un comportement non-linéaire (ex : VISCOPLASTIQUE CHABOCHE) et d'une formulation élément fini (ex : CUB8,DKT). C'est le sous-programme `modeli.eso` qui effectue le travail en appelant d'autres sous-programmes. Aucun calcul n'a lieu pendant cette étape, seule la description d'un nouveau type de données est à faire. Dans de nombreux opérateurs où un objet `MMODEL` figure en argument, la première action consiste à analyser les informations de celui-ci, ce qui oriente la suite des calculs. En mécanique, l'utilitaire contenu dans le sous-programme **nomate.eso**, qui attribue un numéro à la loi de comportement, est systématiquement utilisé ; il convient donc de le mettre à jour, si besoin est, en même temps que `MODE`.

5.3.1 Variantes 2 et 3

Il n'est pas nécessaire d'intervenir dans l'opérateur `MODE` ni l'utilitaire **nomate.eso** — l'utilisateur attribue un numéro.

– exemple syntaxe modèle VISCO_EXTERNE dans le fichier de données :

```
LCPAR24 = MOTS 'TFIS' 'TUO2' 'FACF' 'DSIU' 'DGRA' ;
LCVAR24 = MOTS 'PSUP' 'QSUP' ;

mo_uti2 = MODE mail1 'MECANIQUE' 'ELASTIQUE' 'ISOTROPE'
                  'VISCO_EXTERNE' 'GENERAL' 'NUME_LOI' 24
                  'PARA_LOI' LCPAR24 'C_VARINTER' LCVAR24 ;
```

– exemple syntaxe modèle NON_LINEAIRE UTILISATEUR dans le fichier de données :

```
LCM05_0 = MOTS 'YOUN' 'NU  ' ;

LCM05_1 = MOTS 'E001' 'S001' 'E002' 'S002' 'E003' 'S003'
            'E004' 'S004' 'E005' 'S005' 'E006' 'S006'
            'E007' 'S007' 'E008' 'S008' 'E009' 'S009' ;

LCMAT05 = LCM05_0 et LCM05_1 ;
LCVAR05 = MOTS 'EPSE' 'ENXX' 'ENYY' 'ENZZ' 'GNXY' 'GNXZ' 'GNYZ' ;

mo_util = MODE vol1 'MECANIQUE' 'ELASTIQUE' 'ISOTROPE'
                 'NON_LINEAIRE' 'UTILISATEUR' 'NUME_LOI' 5
                 'C_MATERIAU' LCMAT05 'C_VARINTER' LCVAR05 ;
```

5.3.2 Variante 1

Liste des formulations disponibles (définies dans **modeli.eso** dans le tableau `MOFORM`). Au besoin il faudra en ajouter une nouvelle.

```
'THERMIQUE'   'MECANIQUE'   'LIQUIDE'        'CONVECTION'
'POREUX'      'DARCY'       'FROTTEMENT'     'RAYONNEMENT'
'MAGNETODYNAMIQUE'          'NAVIER_STOKES'  'MELANGE'
```

modeli.eso appele un sous-programme **modelj.eso** (j=1,n) suivant la formulation lue (ex : "model2.eso" est appelé dans le cas de la formulation MECANIQUE). Ces sous-programmes servent à dresser la liste de tous les mots acceptés et ceci en fonction de la formulation.

Dans le cas MECANIQUE, (géré par **model2.eso**) le seul comportement linéaire est : ELASTIQUE. La liste des options disponibles aujourd'hui (définies dans **modela.eso**) est :

```
'ISOTROPE'     'ORTHOTROPE'        'ANISOTROPE'   'PORPOR'
'HOMOGENEISE'  'UNIDIRECTIONNEL'   'SECTION'      'ARMATURE'
```

Les comportements non-linéaires disponibles aujourd'hui sont définis dans **modnli.eso**, la liste est :

```
'PLASTIQUE'        'FLUAGE'           'VISCOPLASTIQUE'   'VISCO_EXTERNE'
'ENDOMMAGEMENT'    'PLASTIQUE_ENDOM'   'NON_LINEAIRE'
```

Remarque : La classification en catégorie PLASTIQUE FLUAGE VISCO... est très formelle. Elle permet de donner un semblant de structuration au logiciel car elle oriente certaines actions de type déclaratives (et non calculatoires). Par exemple les noms des composantes d'un modèle de comportement PLASTIQUE seront définies dans **idplas.eso** *alors que celles pour un modèle de type fluage seront définies dans* **idflua.eso**. *Cependant, vis-à-vis du traitement dans* COMP *ou* PASAPAS *ces catégories sont indifférentes.*

Suivant le comportement non-linéaire **model2.eso** se branche sur un des sous-programmes :

```
PLASTIQUE         ----->  modpla.eso
FLUAGE            ----->  modflu.eso
VISCOPLASTIQUE    ----->  modvis.eso
ENDOMMAGEMENT     ----->  modend.eso
PLASTIQUE_ENDOM   ----->  modple.eso
NON_LINEAIRE      ----->  modenl.eso
VISCO_EXTERNE     ----->  modvix.eso
```

Dans chacun de ces sous-programmes on initialise la liste des modèles disponibles. Par exemple dans **modflu.eso** on trouve :

```
'NORTON'    'BLACKBURN'   'RCCMR_316'     'RCCMR_304'
'LEMAITRE'  'POLYNOMIAL'  'BLACKBURN_2'   'CERAMIQUE'
'MAXWELL'   'COMETE'      'CCPL'          'X11'
'BPEL_RELAX''BETON_URGC'  'SODERBERG'     'MAXOTT'
```

Enfin **nomate.eso** affecte un numéro au modèle de matériau et ensuite quelques vérifications sont faites dans **modeli.eso**.

Exemple : J=2 $\Rightarrow$ MODEL2 $\Leftrightarrow$ "Formulation mécanique"

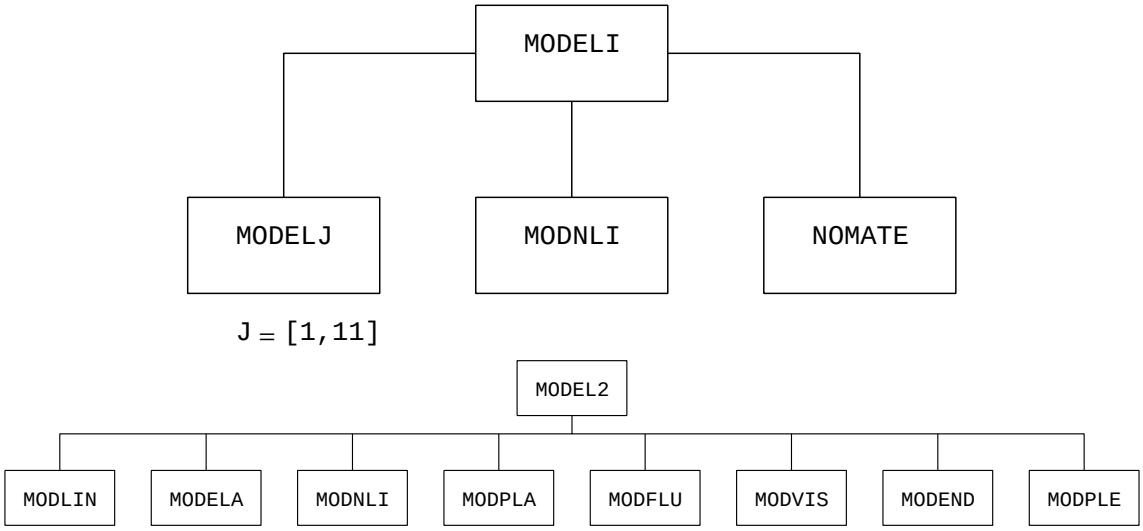


FIG. 5.1 – Organigramme des procédures et sous programmes utilisés pour la description d'un modèle.

5.3.3 Sous-programmes à modifier

En résumé :

MODELI.ESO

Création de l'objet MMODEL. En général rien à faire ici sauf dans le cas d'une nouvelle formulation simple ou couplée.

1. En cas d'une nouvelle formulation, différente des formulations simples ou couplées déjà existantes, aiguiller le programme sur un nouveau sous programme MODELJ (J>13) pour définir les noms des modèles de matériau et les noms de type d'éléments finis utilisables par une telle formulation. Exemple en mécanique des fluides :

```
ELSE IF(LESFOR(1).EQ.'NAVIER_STOKES') THEN
   CALL MODE11(MOPROP,NPROP,MOTEF,NBTEF,N1MAX,N2MAX)
```

2. En cas d'une nouvelle formulation mécanique, différente de celles déjà existantes dans MODNLI.ESO, on doit charger la valeur par défaut dans LESPRO(NMAT). Dans ce cas les lignes à ajouter sont de la forme :

```
(    ....       ELSE IF (IPLAC.EQ.7) THEN
                 --
                 --                    )
           ELSE IF (IPLAC.EQ.8) THEN
*
*      Par defaut : NOUVFORMULATION ----> NOUVCOMPORTEMENT
*
               NMAT=NMAT+1
               LESPRO(NMAT)='NOUVCOMPORTEMENT'
```

MODELJ.ESO

Création d'un objet MMODEL spécifique à une formulation nouvelle.

– Dans le cas d'une formulation déjà existante, i.e. thermique, mécanique, liquide, convection, poreux etc.., modifier un des sous programmes appelés par MODELJ($1 \leq J \leq 11$), pour la définition des noms des modèles de matériaux et des noms des types d'éléments finis utilisables par une telle formulation.

Remarque : Pour introduire un nouveau modèle de comportement mécanique il faudrait modifier MODEL2 mais la programmation de MODEL2 entraine que les modifications sont à faire dans les sous-programmes appelés (modnli, modpla, mod...)

MODNLI.ESO

Définition des noms de comportement non-linéaires dans le cas d'une formulation mécanique. Ce sous-programme n'est à modifier que pour ajouter un nouveau type de comportement non-linéaire (plastique, fluage, viscoplastique,..). **En général rien à faire ici non plus ! Par contre il faut modifier un des sous-programmes modpla, modflu, modvis, modend, modple ou modenl.**

MODPLA.ESO

Définition des comportements de type PLASTIQUE, ajouter le nouveau. Par exemple, pour introduire un modèle de comportement plastique s'appelant "TRESMOU" on aura besoin d'insérer les lignes :

```
NMOD=62
 .
 .
MOMODL(62)='TRESMOU'
```

MODFLU.ESO

Définition des comportements de type FLUAGE, ajouter le nouveau.

MODVIS.ESO

Définition des comportements de type VISCOPLASTIQUE, ajouter le nouveau.

MODEND.ESO

Définition des comportements de type ENDOMMAGEMENT, ajouter le nouveau.

MODPLE.ESO

Définition des comportements de type ENDOMMAGEMENT, ajouter le nouveau.

MODENL.ESO

Définition des comportements de type NON_LINEAIRE, ajouter le nouveau.

MODVIX.ESO

Définition des comportements de type VISCO_EXTERNE, ajouter le nouveau.

NOMATE.ESO

Attribue un numéro à un modèle de matériau, ce numéro sera celui qui gouvernera l'exécution de l'écoulement non-linéaire faite dans le sous-programme approprié de **coml7eso**. Pour cela il faut initialiser la variable INATU (qui s'appelera INPLAS dans **coml6.eso**) par une séquence d'instructions comme suit :

```
                        ELSE IF (IMOD.EQ.??) THEN
*     NOUVCOMPORTEMENT
                           INATU=120
```

Cette séquence d'instructions est à placer en fin de la séquence de IF ... ELSEIF... qui est elle-même située après l'appel au sous-programme MODXXX (XXX = ELA,PLA,FLU,VIS,END,PLE) concerné. La valeur qu'il faut confronter à IMOD est à trouver dans le sous-programme MODXXX concerné en face de NOUVCOMPORTEMENT. Par exemple dans le cas déjà cité de l'incorporation d'un modèle plastique "TRESMOU", il faudra insérer derrière :

```
                        ELSE IF(IMOD.EQ.61) THEN
*    JOINT_COAT
                           INATU=119
```

les lignes suivantes : (la valeur 120 représente le fait que c'est le 120ème modèle de comportement toutes catégories confondues, information tirée des commentaires du sous-programme **nomate.eso**)

```
                        ELSE IF (IMOD.EQ.62) THEN
*     TRES_MOU
                           INATU=120
```

5.4 Phase 2 - Champs relatifs — caractéristiques du matériau

La loi physique associée à l'objet MMODEL est une relation entre un ensemble de champs physiques décrits généralement dans Cast3M à l'aide de champs par éléments MCHAML. On notera que certains champs sont étroitement liés au MMODEL — les variables internes pour un modèle de mécanique non-linéaire —, alors que d'autres ont une portée plus générale — température, temps, . . . L'utilisateur et le logiciel identifient les variables à l'aide des noms de composantes de ces MCHAML. Jusqu'à présent les noms de composantes devaient être écrits dans les sous-programmes. Les modèles VISCO_EXTERNE et UTILISATEUR permettent désormais de renseigner certains noms dans le jeu de données. Nous citons dans ce chapitre des utilitaires qui servent à décrire le noms des composantes des champs par élément associés principalement à la formulation mécanique. Ces utilitaires sont appelés par des opérateurs tels MODE, MATE, EXTR, EXIS, COMP, BSIGM, RIGI, ...

5.4.1 Opérateur VARI

Cet opérateur ne crée pas de nouveaux champs physiques, mais calcule la valeur effective des champs selon celle des paramètres, en utilisant des objets MMODEL et MCHAML en argument. Une application fréquente est la variation de caractéristiques matériaux en fonction de la température : l'utilisateur fait figurer la courbe des caractéristiques selon la température dans un MCHAML à partir duquel VARI établit le MCHAML des valeurs pour une température donnée. Il faut envisager de modifier VARI si l'un des champs de paramètres est définie par

une fonction qui ne peut se traduire ni par un objet EVOLUTION, ni par un objet NUAGE à deux colonnes : plus clairement, VARI sait réaliser une interpolation linéaire dans une courbe, ou entre une famille de courbes.
Cast3M dispose désormais de l'utilitaire **comput.eso**, appelé par VARI, qui facilite les calculs d'instanciation plus complexes. L'utilisateur substitue un développement approprié au contenu du sous-programme standard et crée un nouvel exécutable. Dans le jeu de données, des instructions telles que

```
mo_uti2 = MODE mail1 'MECANIQUE' 'ELASTIQUE' 'ISOTROPE' ;
lpyoun = MOTS 'T   ' 'PORO' 'YOGC' ;
lpnu   = MOTS 'T   ' 'PORO' 'YOGC' ;
ma_uti2 = MATE mo_uti2 'YOUN' lpyoun 'NU  ' lpnu   'RHO ' xrho ;
```

permettent de construire le MCHAML de caractéristiques donné en argument de VARI, qui fait alors appel à comput.eso pour instancier YOUN et NU. Si la température T apparaît, il est nécessaire qu'elle soit placée en première position dans les objets LISTMOTS tels lpyoun.

5.4.2 Sous-programmes utilitaires

Ceux-ci doivent notamment être mis à jour si l'on formule un nouvel élément fini :

IDCONT.ESO

Ce sous-programme renvoie le nombre et le nom des composantes du tableau des contraintes.

IDDEFO.ESO

Ce sous-programme renvoie le nombre et le nom des composantes du tableau des déformations.

IDDEIN.ESO

Ce sous-programme renvoie le nombre et le nom des composantes du tableau des déformations inélastiques.

IDGRAD.ESO

Ce sous-programme renvoie le nombre et le nom des composantes du tableau des gradients de déplacements.

5.4.3 Variables internes

Le sous-programme **idvari.eso** renvoie le nombre et le nom des composantes des variables internes. Cependant la variante VISCO_EXTERNE dispose de 4 variables internes prédéfinies EC0, ESW0, P et QTLD, auxquelles l'utilisateur peut adjoindre d'autres composantes dans le jeu de données moncas.dgibi. Dans cette variante, l'utilisateur ne dispose que des composantes matériau élastique YOUN, NU, ALPH et RHO. Les autres seront calculées dans **creep.eso**.
La variante NON_LINEAIRE UTILISATEUR conduit à spécifier les composantes des champs de variables internes et de caractéristiques dans le jeu de données. Aucune composante par défaut.

– syntaxe pour les caractéristiques du modèle UTILISATEUR, suite du Chapitre 3 :

```
ma_u0 = MATE mo_util 'YOUN' youn0 'NU  ' 0.33 ;

epsp2 = (prog 0. 1. 2. 3. 4. 5.)*1.e-2 ;
Y1 = (prog 0. 200. 250. 280. 300. 310.)*1.e6 ;
ma_u11 = MATE mo_util 'E001' (EXTR epsp2  1) 'S001' (EXTR Y1  1)
                      'E002' (EXTR epsp2  2) 'S002' (EXTR Y1  2)
                      'E003' (EXTR epsp2  3) 'S003' (EXTR Y1  3)
```

```
                    'E004' (EXTR epsp2  4) 'S004' (EXTR Y1  4)
                    'E005' (EXTR epsp2  5) 'S005' (EXTR Y1  5) ;
```

```
ma_util = ma_u0  et ma_u11
```

Pour ce modèle, seul le type REAL*8 est admis pour le champ de caractéristiques utilisé par COMP ; ainsi une courve est traduite par une double liste étiquetée et non par un objet EVOLUTION.

Dans le cas le plus général, on est conduit à modifier **idvari.eso** quand on veut introduire un nouveau modèle de comportement.

IDVARI.ESO

Déclarations des noms de composantes des variables internes :
- pour un matériau linéaire : le paramètre MATEPL est mis à 0
- pour un matériau non linéaire :

1. Matériau plastique : le paramètre IPLAC correspond au type de plasticité défini dans **modpla.eso**. Le paramètre MATEPL correspond au numéro du modèle de matériau défini dans nomate.eso. En cas de rajout d'un nouveau type de plasticité, charger les noms des variables internes dans le vecteur LESOBL et les variables facultatives dans LESFAC. Charger la variable NBROBL au nombre total de variables internes obligatoires et NBRFAC le nombre de variables internes facultatives. Les variables internes sont chargées dans l'un des sous-programmes **idvar6.eso et idvar7.eso**.
2. Matériau fluage : IPLAC et MATEPL sont définis respectivement dans **modflu.eso et nomate.eso**. En cas de rajout d'un nouveau type de fluage, charger les variables internes dans l'un des sous-programmes **idvar1.eso, idvar2.eso ou idvar3.eso**.
3. Matériau viscoplastique : IPLAC et MATEPL sont définis respectivement dans **modvis.eso et nomate.eso**. En cas de rajout d'un nouveau type de viscoplasticité, charger les variables internes dans le sous programme **idvar4.eso**.
4. Matériau endommageable : IPLAC et MATEPL sont définis respectivement dans **modend.eso et nomate**. En cas de rajout d'un nouveau type d'endommagement, charger les variables internes dans **idvar5.eso**.
5. Matériau plastique endommageable : IPLAC et MATEPL sont définis respectivement dans **modple.eso et idplen.eso**. En cas de rajout d'un nouveau type d'endommagement, charger les variables internes dans **idvar6.eso**.
6. Matériaux VISCO_EXTERNE et NON_LINEAIRE UTILISATEUR : non concernés

En résumé : Il faut modifier idvari.eso pour définir MATEPL **et pour diriger l'appel vers l'un des IDVARX,** — par défaut c'est **idvar6.eso** qui sera appelé ; modifier bien entendu **idvarx.eso**. (Exemple en Annexe).

Remarque : La variable interne représentant la déformation plastique équivalente doit de préférence s'appeler EPSE, car c'est ce nom qui est utilisé, par défaut, par la procédure PASAPAS .

5.4.4 Caractéristiques

La lecture des caractéristiques des matériaux se fait par l'opérateur MATE qui appele le sous-programme fortran **matcar.eso** (via **mater.eso**). Celui-ci commence par appeler le sous-programme **idmatr.eso** qui fabrique une liste de données obligatoires et une liste de données facultatives qui sont fonction de la formulation, du comportement linéaire et du comportement non-linéaire.

Après avoir donné l'organigramme des séquences d'appels des différents sous-programmes (peu sont à modifier), nous dresserons la liste des actions à mener.

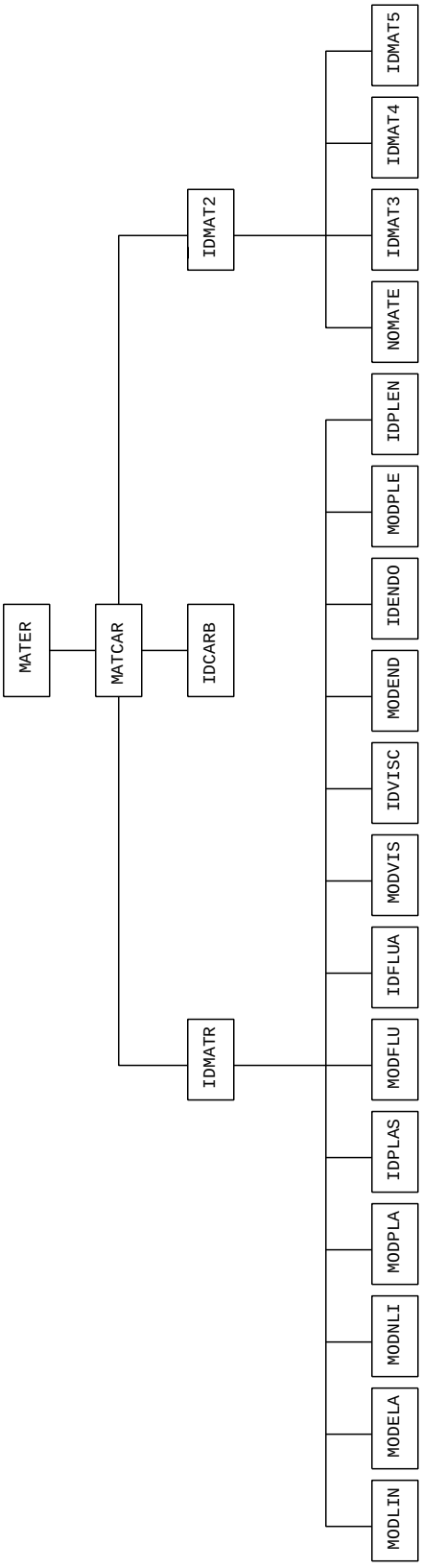


FIG. 5.2 – Organigramme des procédures et sous-programmes utilisés pour la gestion des caractéristiques du matériau.

MATER.ESO et MATCAR.ESO

Création d'un champ de caractéristiques matérielles et/ou géométriques. Rien à faire ici.

IDMATR.ESO

Chargement des noms de composantes de matériaux selon le type de formulation. Cette étape est aussi purement descriptive, il s'agit de définir la liste des composantes obligatoires et facultatives à lire puis à la rigueur de faire quelques vérifications ou mises en forme. Certaines données peuvent être fournies par l'utilisateur sous une forme facile pour lui puis transcrite sous une autre forme pour le logiciel (par exemple la lecture des directions d'orthotropie dans un repère cylindrique permet de fabriquer pour chaque élément les composantes de la loi de Hooke dans un repère cartésien), dans ce cas il faudra faire la mise en forme dans un nouveau sous-programme qui sera appelé par IDMAT2.
Ainsi :

1. Si l'on a ajouté un nouveau comportement linéaire dans **modela.eso**, alors il faut déclarer le nom des composantes des caractéristiques du matériau dans **idmatr.eso** (exemple en Annexe). *Cette situation est très rare quand on introduit un modèle de comportement sans introduire simultanément de nouveaux éléments*.
2. Pour la partie non-linéaire du comportement, la déclaration des noms de caractéristiques du comportement se fait dans le sous-programme appelé par **idmatr.eso** qui suivant la formulation est (voir annexe) :

```
PLASTIQUE          ----->  idplas.eso
FLUAGE             ----->  idflua.eso
VISCOPLASTIQUE     ----->  idvisc.eso
ENDOMMAGEMENT      ----->  idendo.eso
PLASTIQUE_ENDOM    ----->  idplen.eso
NON_LINEAIRE       ----->  idelnl.eso
```

IDPLAS.ESO

Chargement des noms de composantes de matériaux plastiques pour chaque type de plasticité définie dans MODPLA. Ajouter les noms de composantes pour le NOUVCOMPORTEMENT plastique.

IDFLUA.ESO,IDVISC.ESO,IDENDO.ESO,IDPLEN.ESO,IDELNL.ESO

Similaires à **idplas.eso** en cas de FLUAGE,VISCOPLASTICITE, ENDOMMAGEMENT et comportement PLASTIQUE ENDOMMAGEABLE, ou NON_LINEAIRE — autre que UTILISATEUR.

IDMAT2.ESO

Création du `MCHAML` correspondant à un matériau orthotrope.

Remarque : s'il est inutile de prétraiter l'information donnée par l'utilisateur ne rien faire ici ni dans les sous-programmes suivants

IDMAT3.ESO

Définition des directions d'orthotropie d'une coque en 3D, et d'un élément joint 3D.

IDMAT4.ESO et IDMAT5.ESO

Calcul des cosinus directeurs des axes d'orthotropie par rapport au repère local de l'élément.

5.5 Phase 3 - Calcul incrémental, opérateur COMP

5.5.1 Présentation

Le recours à des procédures encapsulant l'appel à l'opérateur COMP, telles que PASAPAS, est recommandé dès que l'on étudie des phénomènes évolutifs. Il peut être nécessaire de modifier ces procédures pour utiliser de nouvelles lois en ajoutant les instructions traitant les nouveaux champs nécessaires. Le détail de l'algorithme utilisé dans **pasapas.procedur et increme.procedur** est décrit dans [2].
Nous décrivons dans ce chapitre la programmation de l'opérateur COMP dont la notice est :

```
Operateur COMP                          Voir aussi : MODE MATE
--------------                                       CARA

CHE1 = COMP  MOD1 CHE2 CHE3 ;

Objet :
_______

L'operateur COMP etablit l'evolution des champs relatifs a un
modele physique, lois de comportement, d'etat ou bien cinetique,
entre un instant initiale et un instant final.
Il est necessaire de preciser l'objet MMODEL qui induit les lois,
l'etat initial des champs necessaires a la formulation contenue
dans le modele et l'etat final des variables de controle.

Applications possibles : lois de comportement en mecanique,
transitions de phase en metallurgie ...

Commentaire :
_____________

MOD1  : type MMODEL

CHE2  : type MCHAML, ensemble des grandeurs decrivant l'etat
        initial pour chaque modele elementaire, les champs etant
        identifies par des noms de composantes (en 4 lettres)
        et de constituants. Sont inclus notamment la date 'TEMP',
        la temperature 'T   '.

CHE3  : type MCHAML, ensemble des grandeurs decrivant l'etat
        final pour chaque modele elementaire, les champs etant
        identifies de la meme maniere que ci-dessus.
        Sont inclus notamment la date 'TEMP', la temperature 'T   '.

CHE1  : type MCHAML, ensemble des grandeurs decrivant l'etat final.


Remarques :
```

Mecanique : Les contraintes, variables internes, deformations inelastiques, deformations totales, ainsi eventuellement que les caracteristiques materiau et geometrique, les parametres externes du modele (s'il en existe) et autres grandeurs relatives a l'etat initial sont rangees dans CHE2. La deformation totale, les caracteristiques de materiau et geometriques, les parametres externes du modele (s'il en existe) relatifs a l'etat final sont ranges dans CHE3. CHE1 contient alors entre autre les nouvelles contraintes, variables internes et deformations inelastiques.

Metallurgie : La temperature, le temps et les caracteristiques materiau initiales, telles les proportions de phases ou les tailles de grain, sont rangees dans CHE2. La temperature finale est rangee dans CHE3. CHE1 contient les caracteristiques finales, notamment les proportions de phases.

5.5.2 Organigramme

Le propos de COMP est de traiter au niveau de chaque volume élémentaire représentatif du milieu des lois et relations liant des grandeurs physiques a priori quelconques.

L'objet MMODEL induit le calcul réalisé. Les objets MCHAML représentent précisément le milieu puisqu'on peut décrire une grandeur en tout point à partir des valeurs aux points de Gauss et des fonctions de forme. On distingue l'état initial — CHE2 — et l'état final — CHE1 reprend les valeurs de CHE3 complétées de celles calculées.

L'opérateur COMP est associé à un certain nombre d'objets temporaires définis dans DECHE.INC. Les variables et tableaux nommés dans ces objets mettent l'accent sur les calculs thermo-poro-mécaniques. **Cependant toute grandeur figurant dans** CHE2 **ou** CHE3 **est collectée**. Le segment DECHE permet de recenser toutes les données fournies.

L'opérateur se compose de 4 niveaux de boucles imbriquées dont la fonction est de trier les données, renseigner les objets temporaires wrk52 , wrk522, wrk53 et wrk54, réaliser les calculs avec ceux-ci puis ranger les résultats.

STRUCTURE INFORMATIQUE DE COMP

```
- appel a "coml.eso"
|  - lecture des donnees
|  - decompose les MCHAML de donnees en objets temporaires DECHE
|  - appel a "coml2.eso"
|    |  - boucle sur les formulations envisageables (voir notice MODE)
|    |  - boucle sur les zones de modeles elementaires
|    |    *  - reconstruit les champs de caracteristiques pour les etats 1
|    |    *    et 2 afin de realiser si besoin des projections
|    |    *  - trie les DECHE associes a la sous-zone, change au besoin les
|    |    *    points supports et projette eventuellement les grandeurs dont
|    |    *    le support n'est pas approprie
|    |    *  - appel a "coml6.eso"
|    |    *    |   - initialise et renseigne wrk53
|    |    *    |   - appel a "comouw.eso" : initialise wrk52 et wrk522
|    |    *    |   - appel a "comcri.eso" : cree les DECHE associees aux
|    |    *    |     grandeurs a calculer
|    |    *    |   - appel a "comtri.eso" : verifie le type des variables
|    |    *    |     des MELVAL
|    |    *    |   - appel a "comdef.eso" : renseigne wrk53 et les
|    |    *    |     segments IECOU et NECOU
|    |    *    |   - boucle sur les elements
|    |    *    |     *   - appel a "comrot.eso" si materiau non isotrope
|    |    *    |     *   - boucle sur les points de calcul
|    |    *    |     *     *  - appel a  "comval.eso" : renseigne wrk52,
|    |    *    |     *     *     wrk53, wrk522 donnees aux points de GAUSS
|    |    *    |     *     *  - appel a "comara.eso" : reorganisation
|    |    *    |     *     *  - appel a "coml7.eso"
|    |    *    |     *     *    |   - appel a un sous-programme associe a la
C    C    *    C     *     *    C     loi ou la relation physique induite par
O    O    *    O     *     *    O     le modele selon la formulation puis la
M    M    *    M     *     *    M      valeur de INPLAS pour la mecanique
L    L    *    L     *     *    L     (dans ce cas, le numero du comportement
|    2    *    6     *     *    7       est attribue dans "nomate.eso")
```

```
|    |    *    |     *     *    |    - appel a "coml8.eso" qui relaie
|    |    *    |     *     *    |      "coml7.eso" pour la mecanique
|    |    *    |     *     *  - fin de "coml7.eso"
|    |    *    |     *     *  - appel a "defer1.eso" puis "defer2.eso" pour
|    |    *    |     *     *    messages d'erreur
|    |    *    |     *     *  - appel a "comsor.eso" qui range les resultats
|    |    *    |     *     *    dans les MELVAL
|    |    *    |     *   - fin boucle sur points de calcul
|    |    *    |     *   - appel si besoin a "compou.eso" (poutres et tuyaux)
|    |    *    |   - fin de boucle sur les elements
|    |    *    |   - appel a "comfin.eso" : fermeture et destruction des
|    |    *    |     objets temporaires
|    |    *  - fin de "coml6.eso
|    |    *  - met a jour l'inventaire des DECHE
|    |    *  - desactivation et suppression de segments
|    |  - fin de boucle sur les zones de modeles elementaires
|    |  - fin de boucle sur les formulations
|  - fin de "coml2.eso"
|  - construction du MCHAML des resultats
|  - compression des MELVAL constants
|  - destruction des DECHE
|  - ecriture des resultats pour gibiane (ecrobj.eso)
-fin de "coml.eso"
```

Le contenu de DECHE.INC :

```
* segment deroulant le mcheml
      SEGMENT,DECHE
        INTEGER INDEC
        CHARACTER*16 CONDEC
        CHARACTER*8 PHADEC
        INTEGER  IMADEC, IELDEC, IFODEC
        INTEGER INFDEC(N3)
        CHARACTER*8 NOMDEC
        CHARACTER*16 TYPDEC
      ENDSEGMENT
      POINTEUR DEC1.DECHE,DEC2.DECHE
* INDEC : indice du MCHELM
* CONDEC : constituant
* PHADEC : phase
* IMADEC : pointeur maillage
* IELDEC : pointeur melval
* IFODEC : CORRESPOND A IFOCHE (voir MCHAML)
* INFDEC : CORRESPOND A INFCHE (voir MCHAML)
* NOMDEC : nom de composante
* TYPDEC : type du champ

c segment des noms de composantes relatives a un champ physique
      SEGMENT NOMID
        CHARACTER*8 LESOBL(NBROBL),LESFAC(NBRFAC)
      ENDSEGMENT
* LESOBL : compoantes obligatoires
```

```
* LESFAC : composantes facultatives

* segment des types de composantes relatives a un champ physique
      SEGMENT NOTYPE
        CHARACTER*16 TYPE(NBTYPE)
      ENDSEGMENT

* segment des adresses des deche associes a un NOMID
      segment pilnec
       integer pilobl(mobl,mran),pilfac(mfac,mran)
      endsegment

c segment des pointeurs sur nomid et pilnec
      segment domdec
        integer jnomid, jilnec
      endsegment

* pile des domdec (les indices se referent au DATA LISMOT ci-dessous)
      segment piluc(iiluc)
* pile des deche concernes par un modele elementaire
      segment pilcon(iilcon)
* pile de tous les deche correspondant aux melvals vus par l utilisateur
       segment pilmel(iimel)

*
c information sur l element fini
      SEGMENT INFO
       INTEGER INFELE(JG)
      ENDSEGMENT

C  liste de reference identifiant les champs
      PARAMETER (NMOT=25)
      CHARACTER*8 LISMOT(NMOT),MOT
      DATA LISMOT / 'NOEUD   ', 'GRAVITE ', 'RIGIDITE', 'MASSE   ',
     1              'STRESSES', 'DEPLACEM', 'FORCES  ', 'REACTUAL',
     1              'FVOLUMIQ', 'GRADIENT', 'CONTRAIN', 'DEFORMAT',
     1              'MATERIAU', 'CARACTER', 'TEMPERAT', 'PRINCIPA',
     1              'MAHOOKE ', 'HOTANGEN', 'DILATATI', 'VARINTER',
     1              'GRAFLEXI', 'VONMISES', 'VINMETAL', 'DEFINELA',
     1              'PARAMEXT'/

* segments pour passer les informations au point d integration

* donnees brutes en entree de COMP, completees par les valeurs calculees
      SEGMENT WRK52
* temps initial, final
       REAL*8 temp0, tempf
*
       REAL*8 scal0(nsca), scalf(nsca)
* deplacements
       REAL*8 depl0(ndep), deplf(ndep)
```

```
* forces
       REAL*8 forc0(nfor), forcf(nfor)
* gradients
       REAL*8 grad0(ngra), gradf(ngra)
* contraintes
       REAL*8 SIG0(NSTRS), SIGF(NSTRS),DSIGT(NSTRS)
* deformations totales
       REAL*8 epst0(NDEFO),DEPST(NDEFO),epstf(NDEFO)
* caracteristiques materiau
       REAL*8 xmat0(ncara),XMAT(NCARA),xmatf(NCARA),
     &   VALMAT(NCARA),valma0(ncara)
c type et nom de composantes relatifs a VALMAT et valma0
       CHARACTER tyval(ncara)*16, commat(ncara)*8
c pointeur sur les melval de caracteristiques materiau a l etat final
       INTEGER ivalma(ncara)
* caracteristiques geometriques
       REAL*8 xcar0(ncarb),XCARB(NCARB),xcarbf(NCARB),VALCAR(NCARB)
c type et nom de composantes pour les caracteristiques
       CHARACTER tycar(ncarb)*16,comcar(ncarb)*8
* temperatures
       REAL*8 ture0(ntur), turef(ntur)
* contraintes principales
       REAL*8 prin0(npri), prinf(npri)
* matrices de hook
       REAL*8 maho0(nmah), mahof(nmah)
* matrices hook tangent
       REAL*8 hota0(nhot), hotaf(nhot)
* variables internes
       REAL*8 VAR0(NVARI),VARF(NVARI)
* gradients flexion
       REAL*8 graf0(ngrf), graff(ngrf)
*
       REAL*8 rhas0(nrhi), rhasf(nrhi)
* deformations inelastiques
       REAL*8 DEFP(NDEIN),EPIN0(NDEIN),EPINF(NDEIN)
* parametres externes
       REAL*8 PAREX0(NPAREX),PAREXF(NPAREX)
* fourre tout pour les comp non prevues et/ou munies d un autre constituant
       CHARACTER*16 typexo(nexo)
       CHARACTER*8 nomexo(nexo)
       CHARACTER*16 conexo(nexo)
       REAL*8 exova0(nexo),exova1(nexo)
      ENDSEGMENT
      segment wrk522
* temps initial, final
       integer  mkktp0, mkktpf
*
        integer  mkkal0(nsca), mkklaf(nsca),mkkalz
* deplacements
       integer  mkkpl0(ndep), mkkplf(ndep),mkkplz
* forces
```

```
         integer  mkkrc0(nfor), mkkrcf(nfor),mkkrcz
* gradients
         integer  mkkad0(ngra), mkkadf(ngra),mkkadz
* contraintes
          integer  mkkIG0(NSTRS), mkkIGF(NSTRS),mkkIGT(NSTRS),mkkigz
* deformations totales
         integer  mkkst0(NDEFO),mkkPST(NDEFO),mkkstf(NDEFO),mkkstz
* caracteristiques materiau
         integer  mkkat0(ncara),mkkMAT(NCARA),mkkatf(NCARA),
     &   mkkVAT(NCARA),mkkva0(ncara),mkkatz
c pointeur sur les melval de caracteristiques materiau a l etat final
        INTEGER mkklma(ncara)
* caracteristiques geometriques
        integer  mkkar0(ncarb),mkkARB(NCARB),mkkrbf(NCARB),mkkVCR(NCARB)
        integer mblcar,mkkarz
* temperatures
        integer  mkkre0(ntur), mkkref(ntur),mkkrez
* contraintes principales
        integer mkkin0(npri), mkkinf(npri),mkkinz
* matrices de hook
        integer  mkkho0(nmah), mkkhof(nmah),mkkhoz
* matrices hook tangent
        integer mkkta0(nhot), mkktaf(nhot),mkktaz
* variables internes
        integer  mkkVR0(NVARI),mkkVRF(NVARI),mkkvrz
* gradients flexion
        integer  mkkaf0(ngrf), mkkaff(ngrf),mkkafz
*
        integer  mkkas0(nrhi), mkkasf(nrhi),mkkasz
* deformations inelastiques
        integer mkkEFP(NDEIN),mkkPN0(NDEIN),mkkPNF(NDEIN),mkkpnz
* parametres externes
        integer  mkkEX0(NPAREX),mkkEXF(NPAREX),mkkexz
* fourre tout pour les comp non prevues et/ou munies d un autre constituant
        integer mkkvx0(nexo),mkkvx1(nexo),mkkvxz
      ENDSEGMENT
*
* quelques donnees pratiques
      SEGMENT WRK53
        INTEGER MATE,INPLAS,MELE,IPMAIL,NPINT,NBNN,NBELEM
        INTEGER NFOR,NMAT,NUMAT,NUCAR,NMATR,NCARF
        INTEGER MFR,NBG,NBGS,NSTRS,LRE,LW,LW2,LHOOK,LHOO2,IPORE,NBNO
        INTEGER NVART,NMATT,NCARR,JECHER,ISTEP,NBINT,JNOID
        INTEGER NBPTEL,NEL,N2PTEL,N2EL,KERRE
        INTEGER ITHHER
        CHARACTER*8  CMATE
        CHARACTER*16 CONM
        LOGICAL LOGSUC,LOGVIS,LUNI1,LUNI2,LOGRE,LOGIN
        REAL*8 PRECIS,SECT,COB,XMOB,EPAIST
        REAL*8 BID(6),BID2(6)
        REAL*8 COORGA(3),LCARAC
```

```
      ENDSEGMENT
* MATE : numero de materiau
* INPLAS : numero loi de comportement
* MELE : numero element fini
* IPMAIL : pointeur maillage
* NPINT : nombre points d integration coque
* NBNN : nombre de noeuds
* NBELEM : nombre d elements
* MFR : numero formulation
* NBPTEL : nombre de points d integrations
* NEL : nombre d elements
* KERRE : indicateur d erreur
* CMATE : description comportement
* CONM : constituant du modele
* PRECIS : precision
* SECT : section
* EPAIST : epaisseur
* COORGA(3) : coordonnees cartesiennes du point d'integration courant
* LCARAC : longueur caracteristique de l'element courant

* tableaux intermediaires
       SEGMENT WRK54
        REAL*8 DDHOOK(LHOOK,LHOOK)
        REAL*8 DDAUX(LHOOK,LHOOK)
        REAL*8 TXR(IDIM,IDIM),DDHOMU(LHOOK,LHOOK)
        REAL*8 XLOC(IDIM,IDIM),XGLOB(IDIM,IDIM)
        REAL*8 D1HOOK(LHOOK,LHOOK),ROTHOO(LHOOK,LHOOK)
        CHARACTER*16 TYMAT(NCXMAT)
        REAL*8 XMAT1(NCXMAT),XMAT2(NCXMAT)
      ENDSEGMENT
*
```

5.5.3 Développement

Il s'agit avec COMP d'être à même d'utiliser toutes les grandeurs figurant parmi les données transmises par l'opérateur, et de retrouver celles-ci dans des tableaux aux appellations conventionnelles au niveau des sous-programmes appelés par **coml7.eso et coml8.eso**. A ce titre la liste de mots LISMOT contenue dans DECHE.INC décrit les champs a priori recherchés et est exploitée systématiquement dans les sous-programmes

- **comou2.eso** , appelé par **comouw.eso**
- **cotype.eso**, appelé par **comcre.eso**, appelé par **comcri.eso**
  appelé par **comtri.eso**
- **comval.eso**
- **comsor.eso**

En particulier, les noms de champs spécifiés, pour les modèles VISCO_EXTERNE ou NON_LINEAIRE UTILISATEUR, sous l'étiquette PARA_LOI par l'utilisateur lors de la déclaration du modèle, correspondent à l'étiquette 'PARAMEXT' de LISMOT et aux tableaux PAREX0 et PAREXF de wrk52.

Les tableaux typexo, nomexo, conexo, exava0 et exova1 sont renseignés avec les champs qui n'ont pas été identifiés au travers de la liste LISMOT.

L'objet temporaire wrk52 est construit en distinguant état initial — terminaison 0 — et état final — terminaison f. En mécanique, les contraintes initiales sont dans sig0, l'incrément de déformation depst est la différence des déformations finales et initiales, les caractéristiques xmat et xmatf correspondent à l'état final. L'opérateur calcule les contraintes et les variables internes finales dans sigf et varf.

Quelques tableaux utilitaires sont inclus — par exemple dsigt pour les contraintes. Les tableaux peuvent paraître redondants : valmat et xmat pour les caractéristiques matériau. On notera que pour les caractéristiques on dispose des noms des composantes et de leurs types. Par ailleurs, des tableaux permettent d'exploiter les champs non-désignés dans la liste LISMOT.

L'objet temporaire wrk522 est un préconditionnement pour renseigner wrk52 dans **comval.eso**.

L'objet temporaire wrk54 est un utilitaire permettant de calculer les opérateurs d'élasticité avec le sous-programme **calsig.eso**.

L'objet temporaire wrk53 consigne des données communément employées.

Pour prendre en compte une nouvelle loi de comportement en mécanique il suffit généralement de modifier :

1. **coml8.eso** : il faut insérer le sous-programme propre à la loi, par exemple sous la forme

```
      IF (INPLAS.EQ.???)   THEN
        CALL CMALOI(wrk52,wrk53,wrk54,wrk27,IB,IGAU,
&                NBPGAU,ecou,necou,iecou)
      ENDIF
```

   wrk27 représentant éventuellement un segment utilitaire propre à cette loi.

2. **cotype.eso** : quand ce n'est pas une loi VISCO_EXTERNE ou UTILISATEUR et quand la loi utilise des champs qui ne sont pas décrits localement par des réels, il est nécessaire de rajouter autant de séquences que nécessaire, en suivant les adresses induites par la liste LISMOT, suivant le modèle

```
IF (INPLAS.EQ.???)   THEN
   NBTYPE=YYYY
   SEGINI NOTYPE
   MOTYPE=NOTYPE
   .
   .
   .
   TYPE(K)=TYPEVAR (K=1,...,YYYY)
   .
   .
   .
 ENDIF
```

   Signification des variables

   INPLAS

   Le paramètre INATU de **NOMATE.ESO** (numéro de nature du matériau). est à confronter au numéro affecté au nouveau modèle dans ce sous-programme.

   YYYY

   Nombre de composantes

TYPEVAR

Chaine de caractères valant, soit REAL*8 si la K^{eme} composante est un réel, soit par exemple POINTEUREVOLUTIO s'il s'agit d'une courbe. On n'écrit la séquence que si l'une des composantes est associée à un POINTEURZZZZZZZZ, pour la grandeur physique — caractéristiques, variables internes, ...— envisagée.
Remarque : Dans les instructions TYPE(K)=TYPEVAR concernant les caractéristiques on mettra d'abord les caractéristiques plastiques obligatoires, puis les caractéristiques facultatives.

Pour introduire des ensembles de lois faisant intervenir des grandeurs physiques non recensées dans la liste LISMOT — champs électromagnétiques, vecteurs flux, ...—, on peut envisager d'introduire les labels adéquats dans la liste LISMOT, en allongeant celle-ci ou encore en renommant les étiquettes 'RIGIDITE', 'MASSE ' et 'STRESSES' de la liste qui sont inemployées. Par la suite on sera conduit à ajouter les tableaux nécessaires dans wrk52 et wrk522, puis à modifier principalement

- **comou2.eso**
- **cotype.eso**
- **comcri.eso**
- **comval.eso**
- **comsor.eso**
- **coml7.eso**

On aura soin de recompiler tous les sous-programmes utilisant DECHE.INC.

5.6 Sous-programmes principaux de COML6.ESO

COMOUW.ESO

Dimensionne les objets temporaires wrk52 et wrk522 et détermine l'association entre les segments DECHE et les composantes des différents tableaux.

COMCRI.ESO

Crée les segments DECHE destinés à recueillir les résultats.

COMTRI.ESO

Vérifie le type des données.

COMDEF.ESO

Renseigne les segments wrk53, necou, iecou et xecou.

COMROT.ESO

Matériaux inélastiques orthotropes, anisotropes et unidirectionnels :

- en formulation massive : on cherche les coordonnées des noeuds de l'élément courant pour le calcul des axes locaux.
- cas particulier de l'ACIER_UNI

COMVAL.ESO

Renseigne l'objet intermédiaire `wrk52` à partir des segments `MELVAL`. Ce programme se réfère à la liste `LISMOT`. Si besoin est, on cherche la section de l'élément courant, on prend en compte l'épaisseur et l'excentrement dans le cas des coques minces avec ou sans cisaillement transverse.

COMARA.ESO

Quelques calculs intermédiaires et réarrangement éventuel des données des tableaux `xmat0, xmat et tymat`, de sorte que les caractéristiques du matériau sont souvent placées dans l'ordre suivant :

– caractéristiques élastiques obligatoires :

$$E, \nu$$

– caractéristiques "élastiques" facultatives :

$$\alpha, \rho$$

– caractéristiques non linéaires obligatoires (dans l'ordre des déclarations dans les sous-programmes IDPLAS,IDFLUA,IDVISC,IDENDO,IDPLEN)
– caractéristiques non linéaires facultatives

COML7.ESO

Gère l'appel des sous-programmes traitant les lois et relations ; est relayé par **coml8.eso** pour la mécanique.

DEFER1.ESO

Traite les messages d'erreur en cas de problème de convergence

DEFER2.ESO

Traite les autres messages d'erreur en général à partir de la variable `KERRE` de l'objet temporaire `wrk53` (0 si pas d'erreur). rappelons qu'il faut également modifier **gibi.erreur** lorsqu'un nouveau message est ajouté .

COMSOR.ESO

On remplit les `MELVAL` concernés à partir de `wrk52`. Ce programme se réfère à la liste `LISMOT`.

COMPOU.ESO

Réalise des moyennes pour les cas particuliers des poutres et tuyaux (sauf Timoshenko) et renseigne les `MELVAL`.

COMFIN.ESO

Supprime les objets temporaires `wrk52 et wrk522` et quelques autres segments.

Signalons l'existence de quelques utilitaires pour la mécanique :

– **calsig.eso** permet de calculer un incrément de contraintes en supposant un comportement élastique, connaissant un incrément de déformation. Cet utilitaire, appelé directement dans **coml7.eso** ou bien par les sous-programmes propres aux diverses lois, a besoin d'un certain nombre d'informations qui se trouvent en partie dans les variables suivantes :

IMAT	entier	
ICAR	entier	
VALMAT	real*8	
VALCAR	real*8	
LHOOK	entier	
N2EL	entier	
N2PTEL	entier	
NBNO	entier	Ces quantités sont là pour être passées à "calsig.eso"
NBPGAU	entier	qui calcule l'incrément de contraintes en comportement
LW	entier	élastique .Pour la dimension des tableaux prendre exemple
NBGMAT	entier	dans le sous-programme "prpvar.eso"
NELMAT	entier	
TXR	real*8	
XLOC	real*8	
D1HOOK	real*8	
ROTHOO	real*8	
DDHOMU	real*8	
CRIGI	real*8	
NUMAT	entier	
NUCAR	entier	
DSIGT	real*8	sortie de calsig, incrément des contraintes en élastique

– **ccoinc.eso**, appelé dans **coml7.eso** traite l'écoulement plastique par une méthode inspirée du "radial-return".
– **cconst.eso**, appelé dans **coml7.eso** traite les lois de viscosité avec une méthode de Runge-Kutta d'ordre 4.

5.7 Conclusion

Nous avons évoqué les 3 phases d'implantation d'un nouveau modèle dans Cast3M : i) déclaration, ii) reconnaissance des champs impliqués, iii) calcul incrémental.

Pour introduire une loi d'évolution physique ou un modèle de comportement mécanique non-linéaire dans Cast3M, la connaissance approfondie d'ESOPE n'est pas utile. Si le développeur doit intervenir dans une petite dizaine de sous-programmes, il peut constater que dans la plupart des cas très peu de lignes fortran sont à ajouter.

Nous espérons que le lecteur aura bien pris conscience que s'il possède déjà une boite fortran réalisant l'écoulement de son modèle, l'incorporation dans Cast3M est normalement l'affaire de un ou deux jours.

Dans les cas de modèles très compliqués (par exemple le modèle de MAXWELL dont le nombre de variables internes n'est pas connu par avance), une maîtrise plus grande du logiciel est souhaitable (contacter l'équipe de développement).

Nous éspérons que ce document s'enrichira et nous sommes prêts à le rendre plus explicite, en fonction de vos suggestions.

5.8 Annexe

Nous considérons par exemple l'implantation du modèle viscoplastique de KOCKS.

5.8.1 Phase 1 : Déclaration d'un modèle mécanique non-linéaire

1. Le premier travail est à faire dans **modvis.eso** où il faut déclarer le modèle en introduisant la ligne :

```
MOMODL(9)='KOCKS'
```

2. Il faut ensuite déclarer un numéro global de comportement non-linéaire dans **nomate.eso**. Cela se traduit par les lignes :

```
   ....
   ....
*     'KOCKS'                                        INATU = 70
   ....
   ....
   ....
                        ELSE IF (IMOD.EQ.9 ) THEN
*     KOCK
                           INATU = 70
```

5.8.2 Phase 2 : Déclaration des paramètres d'un modèle mécanique non-linéaire

1. Ensuite dans **idvari.eso** il faut remplir la variable `MATEPL` par la séquence suivant :

```
                ELSE IF (IPLAC.EQ.9 ) THEN
*    KOCK
                   MATEPL=70
                   INMAT=INMAT+1
```

Puis prévoir l'appel à un des **idvarX.eso** (ici **idvar4.eso**) par la séquence suivante ou seule la partie `.OR.MATEPL.EQ.70` (en majuscule) a été ajoutée :

```
*
      else if (matepl.eq.25.or.matepl.eq.63.OR.MATEPL.EQ.70) then
         call idvar4(matepl,mfr,ifour,npint,ipcomp,nbrobl,nbrfac)
```

2. Il faut enfin déclarer le noms des variables internes dans **idvar4.eso**. Le retour des **idvarX.eso** est le pointeur IPCOMP qui pointe sur un segment de type NOMID. Dans notre exemple il fallait introduire dans **idvar4.eso** la séquence :

```
*----------------------------------------------------------------------
*      MODELE  ELASTIQUE VISCOPLASTIQUE  DE KOCKS
*
      ELSE IF ( MATEPL .EQ. 70 ) THEN
         IF ( MFR.EQ.1.AND.(IFOUR.NE.-2.AND.(IFOUR.NE.1))) THEN
           NBROBL = 2
           SEGINI NOMID
           LESOBL(1) = 'EPSE'
           LESOBL(2) = 'S   '
           IPCOMP=NOMID
           SEGDES NOMID
           RETURN
         ENDIF
*   OPTION INDISPONIBLE
      CALL ERREUR(19)
      RETURN
```

Remarque : IFOUR *traduit le MODE de l'opérateur OPTIO :*
- *-3 : DEFORMATION PLANE GENERALISEE*
- *-2 : CONTRAINTES PLANES*
- *-1 : DEFORMATION PLANE*
- *0 : AXISYMETRIQUE*
- *1 : SERIE DE FOURIER*
- *2 : TRIDIMENSIONNEL*

et MFR *est le numéro de formulation élément fini voir* **nummfr.eso**. MFR *est égal à 1 pour les éléments massifs.*

3. Puis on doit déclarer dans **idvisc.eso** le nom des paramètres matériau de la loi de comportement de KOCKS. Ce qui est réalisé en introduisant la séquence suivante dans le sous-programme :

```
      ELSE IF (IPLAC.EQ.9) THEN

*    VISCOPLASTIQUE MODELE DE KOCK

         MLMOTS=MOOBL
         JGA=MOTS(/2)
         JGM=JGA+10
         SEGADJ MLMOTS
         MOOBL=MLMOTS
         MOTS(JGA+1)='A   '
         MOTS(JGA+2)='B   '
         MOTS(JGA+3)='M   '
         MOTS(JGA+4)='Q   '
         MOTS(JGA+5)='R   '
         MOTS(JGA+6)='H0  '
         MOTS(JGA+7)='AP  '
```

```
      MOTS(JGA+8)='SB  '
      MOTS(JGA+9)='N   '
      MOTS(JGA+10)='S0  '
      GOTO 9999
```

Dans cet exemple, il n'y a pas de paramètre facultatif, sinon on aurait procédé de la même façon à partir de `MLMOTS = MOFAC`.

L'instruction `MOTS(/2)` *demande la 2ème dimension du tableau* `MOTS` *(étant un tableau de chaînes de caractères, on considère le nombre de caractères par chaîne comme la première dimension et le nombre de chaînes comme la deuxième dimension.* `SEGADJ MLMOTS` *demande de redimensionner le segment avec les nouvelles dimensions courantes (* `JGN et JGM`*).*

5.8.3 Phase 3 : Préparation de l'appel à l'écoulement non-linéaire

1. Pas d'intervention dans **cotype.eso** : en effet les paramètres matériaux et les variables internes sont de type `REAL*8`, qui est le type par défaut.
2. La variable `LOGVIS` est initialisée à `.true.` dans **comdef.eso**

```
       if (  inplas .eq. 17 .or.
     2   (  inplas .ge. 19 .and. inplas .le. 25) .or.
     4      inplas .eq. 61 .or.
     4      inplas .eq. 63 .or.
     4      inplas .eq. 65 .or.
     1      inplas .eq. 53 .or.
     1      inplas .eq. 29 .or.
     2      inplas .eq. 44 .or.  inplas .eq. 45  .or.
-->  2      INPLAS .EQ. 70 .OR.  inplas .eq. 74  .or.
     9      inplas .eq. 76 .or.
     9      inplas .eq. 77 .or.  inplas .eq.  82     ) then
          logvis=.true.
       endif
```

La variable `IND` prend la valeur 1, le nombre de sous-pas et la précision des itérations interne sont initialisées

```
       IF ( INPLAS .EQ. 17 .OR. INPLAS .EQ. 19 .OR.
     2     INPLAS .EQ. 20 .OR. INPLAS .EQ. 61 .OR.
     3     INPLAS .EQ. 21 .OR. INPLAS .EQ. 65 .OR.
     4     INPLAS .EQ. 22 .OR. INPLAS .EQ. 23 .OR.
     5     INPLAS .EQ. 24 .OR. INPLAS .EQ. 70 .OR.
     6     INPLAS .EQ. 25 .OR. INPLAS .EQ. 29 .OR.
     7     INPLAS .EQ. 43 .OR. INPLAS .EQ. 44 .OR.
     8     INPLAS .EQ. 45 .OR. INPLAS .EQ. 53 .OR.
     9     INPLAS .EQ. 63 .OR. INPLAS .EQ. 66 .OR.
     a     INPLAS .EQ. 74 .OR. INPLAS .EQ. 76 .OR.
     b     INPLAS .EQ. 84 .OR. INPLAS .EQ. 85 .OR.
     c     INPLAS .EQ. 86 .OR.
     d     INPLAS .EQ. 77 .OR. INPLAS .EQ. 82 .OR.
     e     INPLAS .EQ. 106 .OR. INPLAS .EQ. 102 .OR.
```

```
     f     INPLAS .EQ. 107 .OR. INPLAS .EQ. 108) THEN
*

         IND = 1
*
       if (inplas.eq.44.or.inplas.eq.45) then
         MSOUPA = 2000
       else
         MSOUPA = 200
       endif
         PRECIS = 5.d-7
      ENDIF
```

3. Il est nécessaire d'intervenir dans **coml7.eso**. En effet, le modèle de KOCKS est intégré par une méthode classique de Runge-Kutta pour lequel un utilitaire est disponible : le sous-programme **cconst.eso**. Il faut donc appeler ce sous-programme dans la séquence suivante :

```
*-----------------------------------------------------------------------
*     modeles de viscoplasticite integres par consti
*-----------------------------------------------------------------------
      IF ( INPLAS .EQ. 17 .OR.
     2     (INPLAS .GE. 19 .AND. INPLAS .LE. 25) .OR.
     3      INPLAS .EQ. 61 .OR. INPLAS .EQ. 63  .OR.
     4      INPLAS .EQ. 53 .OR. INPLAS .EQ. 102 .OR.
     5      INPLAS .EQ. 44 .OR. INPLAS .EQ. 76  .OR.
     6      INPLAS .EQ. 45 .OR. INPLAS .EQ. 77  .OR.
     7      INPLAS .EQ. 84 .OR. INPLAS .EQ. 85  .OR.
     8      INPLAS .EQ. 86 .OR. INPLAS .EQ. 70  .OR.
     9      INPLAS .EQ. 107) THEN
*
 ...
*
       CALL CCONST(wrk52,wrk53,wrk54,WRK7,WRK8,WRK9,
     1        NVARI,NSSINC,INV,IFORB,TETA1,TETA2,FI1,FI2,
     4     TLIFE,NCcor,IB,IGAU,NBPGAU,KERREU1,iecou,xecou)

...
```

4. Dans cet exemple, l'écoulement profite de la structure de **cconst.eso** ; cependant pour le modèle de KOCKS il faut appeler dans **cconst.eso** un nouveau sous-programme **incre8.eso** qui réalise le calcul d'un incrément de déformation et de variables internes. le sous-programme **advan1.eso** a lui aussi été légèrement modifié.

5.8.4 Introduction d'un nouveau comportement élastique

Nous prendrons l'exemple du modèle de la SECTION de la poutre à fibre.

1. On introduit dans **modela.eso** la ligne :

```
          MOMODL(7)='SECTION'
```

2. Puis il est nécessaire de définir dans **idmatr.eso** le nom des caractéristiques obligatoires et facultatives du comportement linéaire. Ceci se fait par :

```
           ELSE IF (IPLAC.EQ.7) THEN
C
C      MODELE ET MATERIAU DE LA SECTION DU MODELE A FIBRE
C
              JGM=2
              SEGINI MLMOTS
              MOOBL=MLMOTS
              MOTS(1)='MODS'
              MOTS(2)='MATS'
              JGM=1
              SEGINI MLMOTS
              MOFAC=MLMOTS
              MOTS(1)='MAHO'
```

SEGINI MLMOTS *est un ordre ESOPE demandant de créer un segment de type* MLMOTS*, qui d'après sa définition dans le* -INC SMLMOTS *que voici :*

```
*=============================================================
*
*     OBJET DE TYPE 'LISTMOTS'. DEFINIT UNE LISTE DE MOTS.
*
      SEGMENT MLMOTS
       CHARACTER*(JGN) MOTS(JGM)
      ENDSEGMENT
      POINTEUR MLMOT1.MLMOTS,MLMOT2.MLMOTS,MLMOT3.MLMOTS
      POINTEUR MLMOT4.MLMOTS,MLMOT5.MLMOTS,MLMOT6.MLMOTS
*
*=====================================================
```

contient un tableau de JGM *chaînes de* JGN *caractères chacune. Les valeurs des pointeurs sont sauvées dans* MOOBL *pour le pointeur des noms de caractéristiques obligatoires et dans* MOFAC *pour celui des noms de caractéristiques facultatives. Ce sont ces tableaux qui permettront de remplir le segment* NOMID *qui contient :*

```
      SEGMENT NOMID
        CHARACTER*8 LESOBL(NBROBL),LESFAC(NBRFAC)
      ENDSEGMENT
```

Bibliographie

[1] T. Charras. Implantation d'une nouvelle loi de comportement. Rapport DMT/98-029. Technical report, CEA, 1998.

[2] A. De Gayffier. Algorithme non linéaire de CASTEM 2000 (NONLIN et INCREME). Rapport DMT/94-188. Technical report, CEA, 1994.

[3] S. Felix. Implantation d'un nouvel élément fini dans le code CASTEM 2000. Rapport DMT/93-654. Technical report, CEA, 1993.

[4] S. Felix. Implantation d'un nouveau modèle de comportement dans le code de calcul CASTEM 2000. Rapport DMT/94-530. Technical report, CEA, 1994.

[5] M. Lainet. PROJET PLEÏADES. Spécifications / Conception d'une évolution du code CAST3M pour la prise en compte d'un modèle de comportement mécanique non-linéaire défini à l'externe par l'utilisateur. Révisions pour compatibilité avec ABAQUS. Guide d'utilisation. Rapport CS/311-1/AJ02A009/RAP/02/07/15 Version 2.2. Technical report, CS, 2002.

[6] M. Lainet. PROJET PLEÏADES. évolution du code CAST3M pour la prise en compte d'un modèle de comportement mécanique non-linéaire défini à l'externe par l'utilisateur. Rapport de validation. Rapport CS/311-1/AJ02A009/RAP/03/01/09 Version 1.0. Technical report, CS, 2003.

[7] D. Le Boulch. Implantation d'un nouveau modèle de comportement dans le code de calcul CASTEM 2000. Rapport DMT/97-215. Technical report, CEA, 1997.