

Manuel d'Utilisation
Fascicule U1.0- : Synoptique
Document : U1.03.00

Les grands principes de fonctionnement du **Code_Aster**

Résumé :

On présente ici de façon sommaire les principes de fonctionnement du *Code_Aster* et les principales règles d'utilisation.

Ce document reste un descriptif général et le lecteur se reportera utilement aux autres documents, pour tous les détails d'utilisation.

1 Principes généraux

La version 6 du *Code Aster* permet de réaliser des calculs de structures pour les phénomènes thermique, mécanique, thermo-mécanique, ou thermo-hydro-mécanique couplé, avec un comportement linéaire ou non linéaire, et des calculs d'acoustique interne.

Les non linéarités portent sur les comportements des matériaux (plasticité, viscoplasticité, endommagement, effets métallurgiques, hydratation et séchage du béton, ...), les grandes déformations ou grandes rotations et le contact avec frottement. On se reportera à la plaquette de présentation de la version 6 pour la présentation des différentes fonctionnalités.

Les études industrielles courantes nécessitent la mise en œuvre d'outils de maillage et de visualisation graphique, qui ne font pas partie du Code. Cependant, plusieurs outils sont utilisables pour ces opérations par l'intermédiaire de procédures d'interface intégrées au Code.

Pour réaliser une étude, l'utilisateur doit, en général, préparer deux fichiers de données :

- le fichier de **maillage** :

pour définir la description géométrique et topologique du maillage sans choisir, à ce stade le type de formulation des éléments finis utilisés ou le phénomène physique à modéliser. Certaines études peuvent conduire à utiliser plusieurs fichiers de maillage.

Ce fichier de maillage est, en général, produit par une interface intégrée au *Code Aster* à partir d'un fichier provenant d'un logiciel de maillage utilisé en pré-processeur (GIBI, GMSH, IDEAS...). Les informations que doit contenir ce fichier sont spécifiques au *Code_Aster*. Elles définissent des entités classiques de la méthode des éléments finis :

- **nœuds** : points définis par un **nom** et par leurs **coordonnées cartésiennes** dans l'espace **2D** ou **3D**,
- **mailles** : figures topologiques **nommées** planes ou volumiques (point, segment, triangle, quadrangle, tétraèdre, ...) sur lesquelles pourront s'appliquer différents types d'éléments finis, de conditions aux limites ou de chargements.

Pour améliorer la sûreté d'utilisation et le confort des opérations de modélisation et de dépouillement des résultats, on peut définir, dans le fichier de maillage, des niveaux d'entités supérieurs, possédant en commun une propriété quelconque et qui pourront être utilisés directement par leur nom :

- **groupes de nœuds** : listes nommées de noms de nœuds,
- **groupes de mailles** : listes nommées de noms de mailles.

On notera, dès maintenant, que toutes les entités géométriques manipulées (nœuds, mailles, groupes de nœuds, groupes de mailles) sont **nommées par l'utilisateur** et utilisables à tout moment par leur nom (**8 caractères au maximum**). L'utilisateur pourra utiliser cette possibilité pour identifier explicitement certaines parties de la structure étudiée et faciliter ainsi le dépouillement des résultats. La numérotation des entités n'est jamais explicitée : elle sert uniquement en interne pour pointer sur les valeurs des différentes variables associées.

- le fichier de **commandes** : pour définir le texte de commande qui permet :
 - de lire et éventuellement enrichir les données du fichier de maillage (ou d'autres sources de résultats externes),
 - d'affecter les données de modélisation sur les entités du maillage,
 - d'enchaîner différentes opérations de traitement : calculs, post-traitements spécifiques,
 - d'éditer les résultats sur différents fichiers.

Le texte de commande fait référence aux noms d'entités géométriques définis dans le fichier de maillage. Il permet aussi de définir de nouveaux groupes à tout moment.

Du point de vue informatique, ces deux fichiers sont des fichiers **ASCII** en format libre. On en donne ici les caractéristiques principales :

Syntaxe du fichier de maillage :

- longueur de ligne limitée à 80 caractères,
- les caractères admis sont :
 - les 26 majuscules **A-Z** et les 26 minuscules **a-z** converties automatiquement en majuscules, sauf dans les textes (fournis entre quotes),
 - les dix chiffres **0-9** et les signes de représentation des nombres (**+ - .**),
 - le caractère **_ blanc souligné** utilisable dans des mots-clés ou des noms,
- un mot doit toujours commencer par une lettre,
- le caractère **blanc** est toujours un séparateur,
- le caractère **%** indique le début, jusqu'à la fin de la ligne, d'un **commentaire**.
- Les autres règles de lecture sont précisées dans le fascicule [U3.01.00]

Syntaxe du fichier de commandes :

- syntaxe liée au langage Python, permettant d'inclure des instructions de ce langage
- le caractère **#** indique le début, jusqu'à la fin de la ligne, d'un **commentaire**.
- Les commandes doivent commencer en colonne 1, à moins qu'elles ne fassent partie d'un bloc indenté (boucle, test)

Les autres règles de lecture sont précisées dans le fascicule [U1.03.01].

2 Maillage

2.1 Généralités

La structure et la syntaxe du fichier de **maillage** sont détaillées dans le **Fascicule [U3.01.00]**.

Ce fichier peut être rédigé (pour des maillages élémentaires) ou modifié manuellement avec n'importe quel éditeur de texte. C'est un fichier lu en format libre, structuré en blocs d'informations ou sous-fichier par des mots-clés imposés.

Plusieurs utilitaires de conversion sont disponibles pour permettre la conversion de fichiers de maillage produits par d'autres progiciels (IDEAS, GIBI, GMSH...) ou des fichiers de maillage au format MED.

2.2 Le fichier de maillage Aster

Le fichier de **maillage Aster** est lu de la première ligne jusqu'à la première occurrence d'une ligne débutant par le mot **FIN**. **Ce mot-clé est obligatoire**. Le fichier de maillage est structuré en sous-fichiers indépendants commençant par un **mot-clé** et terminé par le mot-clé imposé **FINSF**.

Ce fichier doit comporter au moins deux sous-fichiers :

- les coordonnées de tous les nœuds du maillage dans un repère cartésien 2D (COOR_2D) ou 3D (COOR_3D).
- la description de toutes les mailles (TRIA3, HEXA20, etc ...), sur lesquelles on affectera ensuite des propriétés physiques, des éléments finis, des conditions aux limites ou des chargements.

Il peut contenir éventuellement des groupes de nœuds (GROUP_NO) ou de mailles (GROUP_MA) pour faciliter les opérations d'affectation, mais aussi le dépouillement des résultats.

Il est indispensable de créer explicitement à ce stade les mailles situées sur les frontières d'application des chargements et des conditions aux limites. On trouvera alors, dans le fichier de maillage :

- les mailles de bord des éléments 2D nécessaires,
- les mailles de face des éléments 3D massifs nécessaires;
- les groupes de mailles d'arête et/ou de face associés.

Cette contrainte devient supportable lorsque l'on utilise une interface, qui fait le travail à partir des indications fournies lors du maillage (voir les documents PRE_IDEAS [U7.01.01] ou PRE_GIBI [U7.01.11]).

2.3 La description des mailles

Les conventions de description de la topologie des mailles et les conditions d'utilisation des différents types de mailles sont décrites dans le fascicule [U3.01.00].

Les principaux types de mailles reconnus sont identifiés par les mots clés réservés suivants [U3.01] :

/	POI1					maille ponctuelle
/	SEG2	/	SEG3	/	SEG4	segments à 2, 3, ou 4 nœuds
/	TRIA3	/	TRIA6	/	TRIA7	triangles à 3, 6 ou 7 nœuds
/	QUAD4	/	QUAD8	/	QUAD9	quadrangles à 4, 8 ou 9 nœuds
/	HEXA8	/	HEXA20	/	HEXA27	hexaèdres à 8, 20 ou 27 nœuds
/	PENTA6	/	PENTA15			pentaèdres à 6 ou 15 nœuds
/	TETRA4	/	TETRA10			tétraèdres à 4 ou 10 nœuds
/	PYRAM5	/	PYRAM13			pyramides à 5 ou 13 nœuds

2.4 Les interfaces

Ces interfaces permettent de convertir les fichiers, avec ou sans format, utilisés par différents progiciels ou codes de calcul, au format conventionnel du fichier de maillage Aster.

Les interfaces disponibles actuellement sont celles qui permettent d'utiliser le mailleur IDEAS, le mailleur GIBI de CASTEM 2000, le mailleur GMSH, et de traiter les fichiers de maillage au format d'échange MED.

2.4.1 Le fichier universel IDEAS

L'interface est réalisée à l'aide de la commande PRE_IDEAS [U7.01.01]

Le fichier convertible est le fichier universel défini par la documentation I-DEAS (voir Fascicule [U3.03.01]). La reconnaissance de la version IDEAS utilisée est automatique.

Un fichier universel IDEAS est constitué de plusieurs blocs indépendants dénommés "**data sets**". Chaque "**data set**" est encadré par la chaîne de caractères **-1** et numéroté. Les "**data sets**" reconnus par l'interface sont décrits dans le fascicule [U3.03.01].

2.4.2 Le fichier de maillage GIBI

L'interface est réalisée à l'aide de la commande PRE_GIBI [U7.01.11]).

Le fichier convertible est le fichier ASCII restitué par la commande SAUVER FORMAT de CASTEM 2000. La description précise de l'interface est donnée en [U3.04.01].

2.4.3 Le fichier de maillage GMSH

L'interface est réalisée à l'aide de la commande PRE_GMSH [U7.01.31]).

Le fichier convertible est le fichier ASCII restitué par la commande SAVE de GMSH.

2.4.4 Le fichier de maillage au format MED

L'interface est réalisée à l'aide de la commande `LIRE_MALLAGE` (`FORMAT : 'MED'`) [U4.21.01].

MED (Modélisation et Echanges de Données) est un format de données neutre développé par EDF R&D pour les échanges de données entre codes de calcul. Les fichiers MED sont des fichiers binaires et portables. La lecture d'un fichier MED par `LIRE_MALLAGE`, permet de récupérer un maillage produit par tout autre code capable de créer un fichier MED sur toute autre machine. Ce format de données est notamment utilisé pour les échanges de fichiers de maillages et de résultats entre ASTER et l'outil de raffinement de maillage HOMARD. La description précise de l'interface est donnée en [U7.01.21].

2.5 L'utilisation de maillages incompatibles

Bien que la méthode des éléments finis préconise l'utilisation de maillages réguliers, sans discontinuité, pour obtenir une convergence correcte vers la solution du problème continu, il peut être nécessaire pour certaines modélisations d'utiliser des maillages incompatibles : de part et d'autre d'une frontière, les maillages ne se correspondent pas. Le raccordement de ces deux maillages est alors géré au niveau du fichier de commandes par le mot-clé `LIAISON_MAIL` de la commande `AFFE_CHAR_MECA`. Ceci permet en particulier de raccorder une zone maillée finement avec une autre zone où on peut se contenter d'un maillage grossier.

2.6 Le maillage adaptatif

A partir d'un maillage initial, il est possible d'adapter le maillage, pour minimiser l'erreur commise, à l'aide de la macro commande `MACR_ADAP_MAIL`, qui fait appel au logiciel HOMARD. Le logiciel de maillage adaptatif HOMARD fonctionne sur des maillages formés de segments, triangles, tétraèdres.

Cette adaptation de maillage se place après un premier calcul avec le *Code_Aster*. Un indicateur de l'erreur aura été calculé. En fonction de sa valeur maille par maille, le logiciel HOMARD modifiera le maillage. Il est également possible d'interpoler des champs de température ou de déplacement aux nœuds de l'ancien maillage vers le nouveau [U7.03.01].

3 Commandes

3.1 Le fichier de commande

Le fichier de **commande** contient un ensemble de commandes, exprimées dans un langage spécifique au *Code_Aster*. En complément des caractéristiques de fichier décrites au paragraphe 1, on trouvera la syntaxe détaillée du langage dans le fascicule [U6.02.00]. Ces commandes sont analysées et exécutées par une couche logicielle du *Code_Aster* appelée « superviseur ».

3.2 Le rôle du superviseur

Le superviseur réalise différentes tâches, notamment :

- une **phase de vérification** et d'interprétation du fichier de commande,
- une **phase d'exécution** des commandes interprétées.

Ces tâches sont détaillées dans le fascicule [U1.03.01].

Le fichier de commandes est traité à partir de la ligne où se trouve le premier appel à la procédure `DEBUT()` ou à la procédure `POURSUITE()`, et jusqu'à la première occurrence de la commande `FIN()`. Les commandes situées avant `DEBUT()` ou `POURSUITE()` et après `FIN()` ne sont pas exécutées, mais doivent être syntaxiquement correctes).

- **Phase de vérification syntaxique :**

- lecture et vérification syntaxique de chaque commande ; toute erreur de syntaxe détectée fait l'objet d'un message, mais l'analyse se poursuit,
- vérification que tous les concepts utilisés comme arguments ont été déclarés dans une commande précédente comme concept produit d'un opérateur ; on vérifie aussi que le type de ce concept correspond au type demandé pour cet argument.

- **Phase d'exécution :**

- le superviseur active successivement les différents opérateurs et procédures, qui réalisent les tâches prévues.

3.3 Les principes et la syntaxe du langage de commande

La conception modulaire d'Aster permet de présenter le Code comme une suite de commandes indépendantes :

- les **procédures**, qui ne produisent pas directement de résultats, mais assurent, entre autre, la gestion des échanges avec les fichiers externes,
- les **opérateurs**, qui réalisent une opération de gestion de données ou de calcul et produisent un **concept** résultat auquel l'utilisateur donne un nom.

Ces concepts représentent des structures de données, que l'utilisateur peut manipuler. Ces **concepts** sont **typés** au moment de leur création et ne pourront être utilisés que comme argument d'entrée du type correspondant.

Les procédures et les opérateurs échangent donc les informations nécessaires et des valeurs par l'intermédiaire des **concepts nommés**

La syntaxe complète des commandes et ses implications sur la rédaction du fichier de commandes sont détaillés dans le fascicule [U1.03.01]. On donne ici un exemple de quelques commandes (extraites de l'exemple commenté en [U1.05.00]) :

```
mail = LIRE_MALLAGE ( )

mod1 = AFFE_MODELE(MALLAGE = mail,
                  AFFE=_F(TOUT='OUI',
                          PHENOMENE='MECANIQUE',
                          MODELISATION='AXIS'))

f_y = DEFI_FONCTION ( NOM_PARA = 'Y'
                    VALE =_F ( 0., 20000.,
                               4., 0. )
                    )

charg = AFFE_CHAR_MECA_F ( MODELE = mod1
                        PRES_REP =_F ( GROUP_MA = ('lfa', 'ldf'),
                                      PRES = f_y ) )
.....
res1 = MECA_STATIQUE( MODELE=mod1,
                    .....
                    EXCIT=_F(CHARGE = charg),
                    ....)

res1 = CALC_ELEM ( reuse=res1, RESULTAT=res1,
.....
                  MODELE=mod1,
                  OPTION=('SIGM_ELNO_DEPL', 'EPSI_ELNO_DEPL'))
```

On notera quelques points généraux, que l'on peut observer sur l'exemple précédent :

- toute commande commence en première colonne,
- la liste des opérandes d'une commande est obligatoirement entre parenthèses, ainsi que les listes d'éléments,
- un `nom_de_concept` ne peut apparaître qu'une seule fois dans le texte de commande comme concept produit, à gauche du signe `=`,
- la **réutilisation d'un concept existant comme concept produit**, n'est possible que pour les opérateurs spécifiés à cet effet. Lorsque l'on utilise cette possibilité (concept réentrant), la commande utilise alors le mot-clé réservé « reuse ».

Cette opération se fait :

- soit avec écrasement des valeurs initiales. A titre d'exemple signalons la factorisation en place d'une matrice de rigidité :
`matass = FACT_LDLT (reuse=matass, MATR_ASSE= matass)`
- soit avec enrichissement du concept.

3.4 Règle de surcharge

Une **règle de surcharge** utilisable, notamment pour toutes les opérations d'affectation, a été ajoutée aux règles d'utilisation d'un `mot_clé_facteur` avec plusieurs listes d'opérandes :

- les affectations se font en superposant les effets des différents `mot_clé`,
- en cas de conflit, le dernier `mot_clé` l'emporte sur les précédents.

Exemple : on souhaite affecter des différents matériaux MAT1, MAT2 et MAT3 à certaines mailles :

```
mater = AFFE_MATERIAU ( MAILLAGE= mon_mail
    AFFE = _F( TOUT = 'OUI', MATER = MAT1 ),
            _F( GROUP_MA = 'mail2', MATER = MAT2 ),
            _F( GROUP_MA = 'mail1', MATER = MAT3 ),
            _F( MAILLE = ( 'm7', 'm8' ), MATER = MAT3 ) )
```

- On commence par affecter le matériau MAT1 à toutes les mailles.
- On affecte ensuite le matériau MAT2 au groupe de mailles `mail2` qui contient, les mailles `m8`, `m9` et `m10`.
- On affecte enfin le matériau MAT3 au groupe de mailles `mail1` (`m5`, `m6` et `m7`) et aux mailles `m7` et `m8`, ce qui est source de conflit puisque la maille `m7` fait déjà partie du groupe `mail1`. La règle de surcharge sera alors appliquée et on obtiendra finalement le champ de matériau suivant :

```
MAT1 : mailles m1 m2 m3 m4
MAT2 : mailles m9 m10
MAT3 : mailles m5 m6 m7 m8
```

3.5 Les bases de données associées à une étude

Le *Code_Aster* s'appuie, pour la gestion de toutes les structures de données associées aux différents concepts manipulés, sur le progiciel JEVEUX. Celui-ci prend en charge la gestion de l'**espace mémoire** demandé par l'utilisateur lors de la demande d'exécution (paramètre **Mémoire** exprimé en Mégaoctets). Cet espace est fréquemment insuffisant pour conserver en mémoire centrale toutes les structures de données. Le progiciel prend alors en charge, la gestion des échanges entre la mémoire centrale et des mémoires auxiliaires sur fichiers.

Chaque entité est affectée, lors de sa création par le code, à un **fichier d'accès direct**. Celui-ci peut être considéré comme une **base de données**, puisqu'il contient, en fin d'exécution le **répertoire** (noms et attributs) qui permet d'exploiter tous les segments de valeurs qu'il contient.

Le Code_Aster utilise plusieurs bases de données :

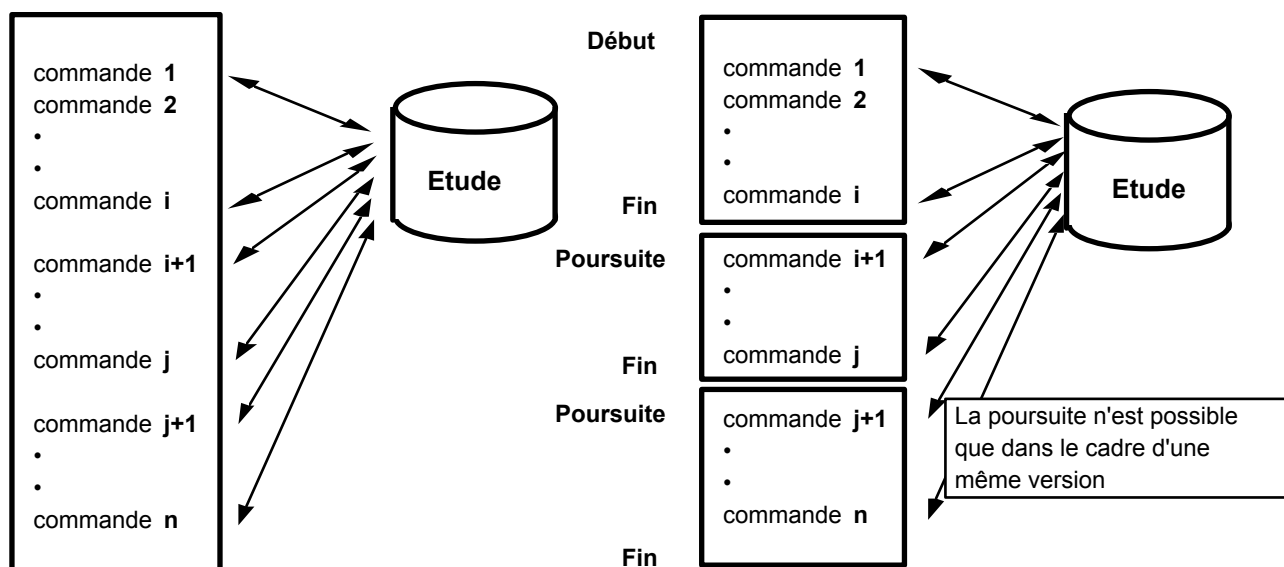
- la base de données **GLOBALE**, qui contient tous les concepts produits par les opérateurs, ainsi que le contenu de certains catalogues sur lesquels s'appuient les concepts ; le fichier associé à cette permet la poursuite ultérieure d'une étude. Elle doit donc être gérée par l'utilisateur.
- les autres bases de données, utilisées uniquement par le Superviseur et les opérateurs, au cours d'une exécution, ne nécessitent pas d'intervention particulière de l'utilisateur.

Réaliser une étude, c'est demander l'enchaînement de plusieurs commandes :

- des procédures pour échanger des fichiers avec le monde extérieur,
- des opérateurs pour créer des concepts produit au fur et à mesure du déroulement des opérations de modélisation et de calcul.

Les commandes qui correspondent à cet enchaînement d'opérations peuvent être réalisées de différentes façons, à partir du module exécutable unique du Code_Aster :

- en une seule exécution séquentielle, sans intervention de l'utilisateur,
- en fractionnant l'étude en plusieurs exécutions successives, avec réutilisation des résultats antérieurs; à partir de la seconde exécution, l'accès à la base de données se fait en **poursuite**; à l'occasion d'une poursuite, on peut redemander la dernière commande, si elle s'est interrompue prématurément (manque de temps, données incomplètes ou incorrectes détectées en phase d'exécution, ...).



Pour gérer ces possibilités, on notera que trois commandes jouent un rôle primordial. Ce sont celles qui correspondent aux **procédures** qui activent le superviseur :

- DEBUT ()** **obligatoire** pour la **première** exécution d'une étude,
- POURSUITE ()** **obligatoire** à partir de la **deuxième** exécution d'une étude,
- FIN ()** **obligatoire** pour toutes les exécutions.

Pour une étude donnée, on peut soumettre des fichiers de commande ayant la structure suivante :

Remarques :

- La commande **INCLUDE** permet d'inclure dans un flot de commandes le contenu d'un autre fichier de commande. Ceci permet notamment, de conserver un fichier des commandes principales lisible et de placer dans des fichiers annexés des données numériques volumineuses (ex : définition de fonctions).
- Les fichiers de commandes peuvent être découpés en plusieurs fichiers qui seront exécutés l'un après l'autre, avec sauvegarde intermédiaire de la base de données. Pour cela, il faut définir les fichiers de commande successifs, dont le suffixe sera : **.com1**, **.com2**, ..., **.com9**. Les exécutions de ces fichiers sont enchaînées. La base de données de la dernière exécution qui s'est bien terminée est conservée.

3.6 Aide à la définition des valeurs

3.6.1 Substitution de valeurs

Plusieurs commandes sont disponibles pour aider l'utilisateur à définir les valeurs utilisées comme arguments, donc paramétrer son fichier de commandes :

- donner un nom à une ou plusieurs valeurs :
 nom = DEFI_VALEUR (TYPE = [valeur]) ;
 ou tout simplement :
 nom = valeur

- Evaluer certaines expressions mathématiques :
 EVAL (expression)

Par exemple :

```
Eptub = 26.187E-3
```

```
Rmoy = 203.2E-3
```

```
Rext = DEFI_VALEUR ( R8 = EVAL ( "" Rmoy+(Eptub/2)"" ) )
```

```
cara = AFFE_CARA_ELEM ( MODELE = modele  
    POUTRE =_F ( GROUP_MA = tout , SECTION : 'CERCLE',  
        CARA = ( 'R','EP' ) , VALE = ( Rext , Eptub ) ) )
```

Ces possibilités se traduisent par une simple substitution des valeurs chaque fois que le Superviseur rencontre le nom choisi par l'utilisateur.

3.6.2 Fonctions de un ou plusieurs paramètres

Il est également souvent nécessaire d'utiliser des grandeurs fonctions d'autres paramètres.

Celles-ci peuvent être :

- soit définies sur un fichier externe lu par la commande, LIRE_FONCTION.
- soit définies dans le fichier de commande par :
 - DEFI_CONSTANTE produit un concept fonction avec une seule valeur constante,
 - DEFI_FONCTION produit un concept fonction pour une grandeur fonction d'un paramètre réel,
 - DEFI_NAPPE produit un concept fonction pour une liste de fonctions d'une même grandeur, chaque élément de la liste correspondant à une valeur d'un autre paramètre réel.

Le concept produit par ces opérateurs est de type fonction et ne peut être utilisé que comme argument d'opérandes qui acceptent ce type. Les opérateurs qui acceptent un argument de type fonction ont pour suffixe F (ex : AFFE_CHAR_MECA_F). Les fonctions sont dans ce cas définies point par point, avec un interpolation linéaire par défaut, donc affines par morceaux.

Les fonctions créées sont des tables discrètes des grandeurs spécifiées à la création. Lors d'une recherche de valeur, on procède suivant les caractéristiques spécifiées, par recherche directe ou par interpolation dans la table (linéaire ou logarithme). On peut spécifier, lors de la création de la fonction, le prolongement hors du domaine de définition de la table, avec différentes règles, ou l'interdire.

- soit définies à l'aide de leur expression analytique par l'opérateur FORMULE : par exemple :

```
omega = 3.566 ;
```

```
linst = ( 0., 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10,  
0.20, 0.40 )
```

```
F = FORMULE(REEL = '(REEL:INST) = COS(OMEGA*INST)')  
F1=CALC_FONC_INTERP( FONCTION=F, VALE_R= linst,  
    NOM_RESU='ACCE', )
```

La fonction analytique $F(t)=\cos(\omega t)$ est ensuite calculée par `CALC_FONC_INTERP` pour les instants de la liste `l_inst` liste d'instant `t`.

3.7 Comment rédiger son fichier de commande avec EFICAS ?

Pour rédiger un fichier de commandes du *Code_Aster*, le plus immédiat consiste à partir d'un exemple déjà rédigé par d'autres. En particulier, l'ensemble des tests du *Code_Aster* constitue souvent une bonne base de départ pour une nouvelle modélisation.

Mais il y a mieux : l'outil **EFICAS** permet de rédiger de façon interactive et conviviale son fichier de commandes, en proposant pour chaque commande la liste des mots clés possibles en vérifiant automatiquement la syntaxe, et en donnant accès à la documentation du Manuel d'utilisation (fascicules [U4] et [U7]).

4 Les grandes étapes d'une étude

Les grandes étapes d'une étude sont dans le cas général :

- la préparation du travail, qui se termine après la lecture du maillage,
- la modélisation au cours de laquelle sont définies et affectées toutes les propriétés des éléments finis et des matériaux, les conditions aux limites et les chargements,
- le calcul peut alors être réalisé par l'exécution de méthodes de résolution globales [U4.5-], qui s'appuient éventuellement sur des commandes de calcul et d'assemblage de matrice et vecteurs [U4.6-]
- les opérations de post-traitements complémentaires au calcul [U4.8-],
- les opérations d'impression des résultats [U4.9-]
- les opérations d'échange de résultats avec d'autres logiciels (visualisation graphique par exemple) [U7.05-]

Une autre façon d'utiliser le *Code_Aster* consiste à exploiter des outils métiers, disponibles dans le Code sous forme de `MACRO_COMMANDES` : citons par exemple les outils métiers :

- ASCOUF (modélisation de coudes fissurés ou de coudes avec sous-épaisseurs),
- ASPIC (modélisation de piquages fissurés ou non fissurés),
- GOUJ2ECH (modélisation du comportement des assemblages filetés).

4.1 Démarrer l'étude et acquérir le maillage

On ne reviendra pas ici sur la fragmentation possible du fichier de commande, qui a été présentée dans un paragraphe précédent.

La première commande exécutable est :

```
DEBUT ( )
```

Les argument de cette commande ne sont utiles que pour les opérations de maintenance ou dans le cas de très grosses études.

Pour la lecture du maillage, provenant d'un logiciel de maillage extérieur, on peut opérer de deux façons :

- convertir le fichier d'un progiciel par une exécution séparée, ce qui permet, éventuellement, de le modifier par traitement de texte et de le conserver :

```
DEBUT ( )  
PRE_IDEAS ( )  
FIN ( )
```

l'étude normale pourra alors débiter par exemple par :

```
DEBUT ( )  
ma = LIRE_MALLAGE ( )
```

- convertir le fichier juste avant de le lire :

```
DEBUT ( )  
PRE_IDEAS ( )  
ma = LIRE_MALLAGE ( )
```

4.2 Affecter des données de modélisation au maillage

Pour construire la modélisation d'un problème mécanique, thermique ou acoustique, il est indispensable d'affecter aux entités topologiques du maillage :

- un modèle d'élément fini,
- les propriétés des matériaux (loi de comportement et paramètres de la loi),
- des caractéristiques géométriques ou mécaniques complémentaires,
- des conditions aux limites ou des chargements.

Ces affectations sont obtenues par différents opérateurs dont le nom est préfixé par **AFFE_**. La syntaxe et le fonctionnement de ces opérateurs utilisent les facilités apportées par les règles déjà mentionnées précédemment sur l'utilisation des mots clés facteur.

4.2.1 Définition d'un domaine d'affectation

Pour réaliser une affectation, il est indispensable de définir un domaine d'affectation par référence aux noms des entités topologiques définies dans le fichier maillage. Cinq mots-clés sont utilisables pour cela, suivant la spécification de l'opérateur :

- faire référence à tout le maillage par **TOUT= 'OUI'**
- affecter à des mailles par **MAILLE= (liste de noms de mailles)**
- affecter à des groupes de mailles par **GROUP_MA= (liste de noms de groupes de mailles)**
- affecter à des nœuds par **NŒUD= (liste de noms de nœuds)**
- affecter à des groupes de nœuds par **GROUP_NO= (liste de noms de groupes de nœuds)**

4.2.2 Affecter le type d'élément fini

Sur les mailles de la structure étudiée, qui ne sont à ce stade que des entités topologiques, il est indispensable d'affecter :

- un ou plusieurs phénomènes à étudier : **'MECANIQUE'**, **'THERMIQUE'**, **'ACOUSTIQUE'** ;
- un modèle d'élément fini compatible avec la description topologique de la maille. Cette affectation induit une liste explicite de degrés de liberté en chaque nœud et une loi d'interpolation dans l'élément.

On utilise pour cela l'opérateur **AFFE_MODELE** [U4.41.01], qui peut être appelé plusieurs fois sur le même maillage. Il utilise les règles de surcharge et de rémanence.

Nota :

*Pour une étude avec plusieurs phénomènes traités (**'MECANIQUE'**, **'THERMIQUE'**), il est indispensable de construire un modèle pour chaque phénomène, par autant d'appels à **AFFE_MODELE**. Par contre, pour un calcul donné (mécanique, thermique, ...) il faut un et un seul modèle.*

Pour connaître les caractéristiques des différents éléments finis disponibles on se reportera aux fascicules [U2-], et [U3-].

4.2.3 Affecter des caractéristiques de matériaux

Il est nécessaire d'affecter à ce stade des caractéristiques de matériau, et les paramètres associés, à chaque élément fini du modèle (sauf pour les éléments discrets définis directement par une matrice de rigidité, de masse et/ou d'amortissement). En d'autres termes, `DEFI_MATERIAU` sert à définir un matériau et `AFFE_MATERIAU` sert à définir un champ de matériau par association du maillage. Pour un calcul donné, il faut un et un seul champ de matériau.

On peut également utiliser les caractéristiques validées du catalogue matériau à l'aide de la procédure `INCLUDE_MATERIAU` [U4.43.02].

Un certain nombre de modèles de comportement sont utilisables : élastique, élastique orthotrope, thermique, acoustique, élastoplastique, élastoviscoplastique, endommagement. Notons qu'il est possible de définir plusieurs caractéristiques de matériaux pour un même matériau : élastique et thermique, élasto-plastique, thermo-plastique, ...

4.2.4 Affecter des caractéristiques aux éléments

Lors de l'utilisation de certains types d'éléments, pour le phénomène 'MECANIQUE', la définition géométrique déduite du maillage ne permet pas de les décrire complètement.

On doit affecter aux mailles les caractéristiques manquantes :

- pour les **coques** : l'épaisseur constante sur chaque maille et un repère de référence pour la représentation de l'état de contrainte,
- pour les **poutres, barres et tuyaux** : les caractéristiques de la section transversale, et éventuellement l'orientation de cette section autour de la fibre neutre.

Ces opérations sont accessibles par l'opérateur `AFFE_CARA_ELEM` [U4.42.01]), qui utilise, pour simplifier la rédaction de la commande, les règles de surcharge et de rémanence.

Une autre possibilité est offerte par cet opérateur : celle d'introduire, directement dans le modèle, des **matrices** de rigidité, de masse ou d'amortissement sur des mailles `POI1` (ou des nœuds) ou des mailles `SEG2`. Ces matrices correspondent aux types d'éléments finis discrets à 3 ou 6 degrés de liberté par nœud `DIS_T` ou `DIS_TR` qui doivent être affectés lors de l'appel à l'opérateur `AFFE_MODELE`.

4.2.5 Affecter les conditions aux limites et les chargements

Ces opérations sont, en général, indispensables. Elles sont réalisées par plusieurs opérateurs dont le nom est préfixé par `AFFE_CHAR` ou `CALC_CHAR`. Sur un même modèle, on pourra réaliser plusieurs appels à ces opérateurs pour définir, au fur et à mesure de l'étude des conditions aux limites et/ou des chargements.

Les opérateurs utilisés diffèrent avec le phénomène :

'MECANIQUE'	<code>AFFE_CHAR_CINE</code>	données de type réel uniquement
	<code>AFFE_CHAR_MECA</code>	données de type fonction
	<code>AFFE_CHAR_MECA_F</code>	données de type réel uniquement
'THERMIQUE'	<code>AFFE_CHAR_THER</code>	données de type réel uniquement
	<code>AFFE_CHAR_THER_F</code>	données de type fonction
'ACOUSTIQUE'	<code>AFFE_CHAR_ACOU</code>	données de type réel uniquement

De plus, on peut établir le chargement sismique pour effectuer un calcul de réponse en mouvement relatif par rapport aux appuis, à l'aide de la commande `CALC_CHAR_SEISME`.

Les conditions aux limites et chargements peuvent être définis suivant leur nature :

- aux nœuds,
- sur des mailles de bord (arête ou face) ou des mailles support d'éléments finis, créées dans le fichier maillage. Sur ces mailles l'opérateur `AFFE_MODELE` a affecté les types d'éléments finis nécessaires.

Pour la description détaillée des opérandes de ces opérateurs et les règles d'orientation des mailles support (repère global, repère local ou repère quelconque) on se reportera aux documents [U4.44-01], [U4.44-02], et [U4.44-04].

Les conditions aux limites peuvent être traitées de deux façons :

- par « élimination » des degrés de liberté imposés (pour des modèles mécaniques linéaires ne mettant en œuvre que des conditions aux limites cinématiques (degrés de libertés bloqués) **sans relation linéaire**. On définira dans ce cas les conditions aux limites par la commande `AFFE_CHAR_CINE`.
- par dualisation [R3.03.01]. Cette méthode du fait de sa plus grande généralité permet de traiter tous les types de conditions aux limites (degré de liberté imposé, relations linéaires entre degrés de liberté, ...) ; la méthode utilisée conduit à ajouter 2 multiplicateurs de LAGRANGE pour chaque ddl imposé ou chaque relation linéaire.

Chaque concept produit par l'appel à ces opérateurs, de type `AFFE_CHAR`, correspond à un système de conditions aux limites et de chargements indissociable. Dans les commandes de calcul, on peut agréger ces concepts en fournissant pour les opérandes `CHARGE` une liste de concepts de ce type.

4.3 Réaliser les calculs par des commandes globales

4.3.1 Analyse THERMIQUE

Pour calculer le(s) champ(s) de température correspondant à une analyse thermique linéaire ou non linéaire :

- **stationnaire** (instant 0),
- **évolutive** dont les instants de calcul sont spécifiés par une liste de réels définie au préalable

Les commandes à utiliser sont :

- `THER_LINEAIRE` pour une analyse linéaire [U4.54.01],
- `THER_NON_LINE` pour une analyse non linéaire [U4.54.02],
- `THER_NON_LINE_MO` pour un problème de charges mobiles en régime permanent [U4.54.03].

Les calculs des matrices et vecteurs élémentaires et assemblés nécessaires à la mise en œuvre des méthodes de résolution sont pris en charge par ces opérateurs.

4.3.2 Analyse STATIQUE

Pour calculer l'évolution mécanique d'une structure soumise à une liste de chargements :

- `MECA_STATIQUE` [U4.51.01] : comportement linéaire, avec superposition des effets de chaque chargement,
- `MACRO_ELAS_MULT` [U4.51.02] : comportement linéaire, en distinguant les effets de chaque chargement,
- `STAT_NON_LINE` [U4.51.03] : évolution quasi-statique d'une structure soumise à une histoire de chargement en petites ou grandes transformations, faite d'un matériau dont le comportement est linéaire ou non linéaire, avec prise en compte possible du contact et frottement.

Si ce calcul mécanique correspond à une étude de **thermo-élasticité**, on fera référence à un **instant** du calcul thermique déjà réalisé. Si le matériau a été défini avec des **caractéristiques dépendant de la température**, celles-ci sont interpolées pour la température correspondant à l'instant de calcul demandé.

Pour les problèmes de thermohydraulique couplé, c'est l'opérateur `STAT_NON_LINE` qui est utilisé pour résoudre simultanément les 3 problèmes thermique, hydraulique et mécanique.

Les calculs des matrices et vecteurs élémentaires et assemblés nécessaires à la mise en œuvre des méthodes de résolution sont pris en charge par ces opérateurs.

4.3.3 Analyse MODALE

Pour calculer les modes propres et valeurs propres de la structure (correspondant à un problème vibratoire ou à un problème de flambement).

- `MODE_ITER_SIMULT` [U4.52.03] : calcul des modes propres par itérations simultanées ; les valeurs propres et vecteur propres sont réels ou complexes,
- `MODE_ITER_INV` [U4.52.04] : calcul des modes propres par itérations inverses ; les valeurs propres et vecteur propres sont réels ou complexes,
- `MACRO_MODE_MECA` [U4.52.02] : allège l'analyse modale en découpant automatiquement l'intervalle de fréquence en sous intervalles,
- `MODE_ITER_CYCL` [U4.52.05] : calcul des modes propres d'une structure à répétitivité cyclique à partir d'une base de modes propres réels.

Ces quatre opérateurs nécessitent au préalable le calcul des matrices assemblées [U4.61-].

4.3.4 Analyse DYNAMIQUE

Pour calculer la réponse dynamique, linéaire ou non linéaire, de la structure. Plusieurs opérateurs sont disponibles. On peut citer par exemple :

`DYNA_LINE_TRAN` [U4.53.02] : réponse dynamique temporelle d'une structure linéaire soumise à une excitation transitoire,

`DYNA_LINE_HARM` [U4.53.02] : réponse dynamique complexe d'une structure linéaire soumise à une excitation harmonique,

`DYNA_TRAN_MODAL` [U4.53.21] : réponse dynamique transitoire en coordonnées généralisées par recombinaison modale.

Ces trois opérateurs nécessitent au préalable le calcul des matrices assemblées [U4.61-].

`DYNA_NON_LINE` [U4.53.01] : réponse dynamique temporelle d'une structure non linéaire soumise à une excitation transitoire, qui calcule également les matrices assemblées.

4.4 Les résultats

Les résultats produits par les opérateurs réalisant des calculs par éléments finis [U4.3-], [U4.4-] et [U4.5-] sont de deux types principaux :

- soit du type de champ (par éléments ou aux nœuds) lorsqu'il s'agit d'opérateurs ne produisant qu'un seul champ (par exemple `RESO_LDLT`),
- soit du type `RESULTAT` à proprement parler qui regroupe des ensembles de champs, accessibles par une variable permettant de les distinguer (instant pour un résultat issu d'un calcul évolutif, fréquence pour un résultat provenant d'un algorithme de recherche de modes propres ou de réponse harmonique, ...).

Un champ dans un concept de type `RESULTAT` est repéré :

- par une variable d'accès qui peut être :
 - un simple numéro d'ordre se référant à l'ordre dans lequel les champs ont été rangés,
 - un paramètre défini selon le type du concept `RESULTAT` :
 - fréquence ou numéro de mode pour un `RESULTAT` du type `mode_meca`,
 - instant pour un `RESULTAT` du type `evol_elas`, `temper`, `dyna_trans` ou `evol_noli`.

- par un nom symbolique de champ faisant référence au type du champ : déplacement, vitesse, état de contrainte, efforts généralisés, ...

En plus des variables d'accès d'autres paramètres peuvent être attachés à un type de concept RESULTAT. Les contenus de ces concepts sont complètement décrits dans le fascicule [U5-].

Les différents champs sont incorporés dans un concept résultat :

- soit par l'opérateur qui crée le concept, une commande globale (MECA_STATIQUE, STAT_NON_LINE, ...) ou une commande simple (MODE_ITER_SIMULT, DYNA_LINE_TRAN, ...),
- soit lors de l'exécution d'une commande qui permet d'ajouter une option de calcul sous forme d'un champ par élément (CALC_ELEM) ou d'un champ aux nœuds (CALC_NO) ; on dit alors explicitement que l'on enrichit le concept :

resul = opérateur (reuse=resul, RESULTAT = resul ...) ;

4.5 Exploiter les résultats

L'ensemble des commandes précédentes a permis de construire différents concepts qui sont exploitables, par des opérateurs de post-traitement des calculs :

- opérateurs généraux de post-traitement (voir fascicule [U4.81]), par exemple CALC_ELEM, CALC_NO, POST_ELEM, POST_RELEVE_T,
- opérateurs de mécanique de la rupture (voir fascicule [U4.82]), par exemple CALC_G_THETA,
- opérateur de métallurgie : CALC_META,
- post-traitement mécanique statique (voir fascicule [U4.83]), par exemple POST_FATIGUE, POST_RCCM,
- post-traitement mécanique dynamique (voir fascicule [U4.84]), par exemple POST_DYNA_ALEA, POST_DYNA_MODA_T.
- opérateurs d'extractions :
 - d'un champ dans un concept résultat RECU_CHAMP [U4.63.1],
 - d'un champ en coordonnées généralisées pour un calcul dynamique avec base modale RECU_GENE [U4.63.2],
 - d'une fonction d'évolution d'une composante à partir d'un concept résultat RECU_FONCTION [U4.63.3],
 - et de restitution d'une réponse dynamique dans la base physique REST_BASE_PHYS,
 - un opérateur de post-traitement de fonctions ou de nappes CALC_FONCTION qui permet des recherches de pics, d'extremums, des combinaisons linéaires, ... [U4.21.9].

Enfin deux procédures IMPR_RESU [U4.91.01] et IMPR_COURBE [U4.33.01] permettent l'impression et éventuellement la création de fichiers exploitables par d'autres progiciels notamment de visualisation graphique. On retiendra notamment la visualisation graphique par IDEAS, GMSH, ou GIBI quel que soit l'outil de maillage utilisé au départ.

5 Fichiers d'impression et messages d'erreur

Aster écrit les informations relatives au calcul dans trois fichiers dont la signification est la suivante.

Fichier	Contenu
ERREUR	Erreurs rencontrées lors de l'exécution
MESSAGE	Informations sur le déroulement du calcul. Répétition du fichier de commande, fourni et son interprétation par Aster. Temps d'exécution de chaque commande. Messages "système"
RESULTAT	Uniquement les résultats demandés écrits expressément à la demande de l'utilisateur et les messages d'erreur

D'autres fichiers sont utilisés pour les interfaces avec les programmes de dépouillement graphique.

On distingue différents types de messages d'ERREUR. Les messages émis seront dirigés uniquement en fonction de leur type :

Code	Type de message	Fichiers de sortie
F	message d'erreur fatale, l'exécution s'arrête après diverses impressions. Les concepts créés au cours de l'exécution sont perdus. Il est utilisé dans le cadre de la détection d'erreur grave ne pouvant permettre la poursuite normale d'une commande Aster	ERREUR MESSAGE RESULTAT
E	message d'erreur, l'exécution continue un peu : ce type de message permet d'analyser une série d'erreurs avant l'arrêt du programme. (par exemple, l'analyse syntaxique du fichier de commandes par le Superviseur). <i>L'émission d'un message de type <E> est toujours suivi par l'émission d'un message de type <F>.</i>	ERREUR MESSAGE RESULTAT
S	message d'erreur, les concepts créés au cours de l'exécution sont validés par le superviseur, l'exécution s'arrête avec fermeture "propre" de la base GLOBALE. Elle est donc réutilisable en POURSUITE. Ce message permet en particulier de se prémunir d'un arrêt système au cours d'un processus itératif.	ERREUR MESSAGE RESULTAT
A	message d'alarme. Le nombre de messages d'alarme est limité automatiquement à 5 messages successifs identiques. Il est recommandé aux utilisateurs qui ont des messages de type A de "réparer" leur fichier de commandes pour les faire disparaître	MESSAGE RESULTAT
I	message d'information du superviseur	MESSAGE