Great principles of operation of *Code_Aster*

Summary:

One presents here way summary the principles of operation of *Code_Aster* and principal rules of use.

This document remains a descriptive general and the reader will refer usefully to the other documents, for all the details of use.

1 Principles generals

Code Aster allows to carry out structural analyses for the phenomena thermics, mechanics, thermomechanical, or thermo-hydro-mechanics coupled, with a or not linear linear behavior, and calculations of internal acoustics.

Nonthe linearities relate to the behaviors of the materials (metallurgical plasticity, viscoplasticity, damage, effects, hydration and drying of the concrete,…), great deformations or great rotations and the contact with friction. One will refer to the plate of presentation of *Code_Aster* for the presentation of the various features.

The current industrial studies require the placement of tools of grid and graphic visualization, which are not part of the code. However, several tools are usable for these operations via procedures of interface integrated into the code.

To carry out a study, the user must, in general, prepare two data files:

- the file of **grid:**

  This file defines geometrical and topological description grid without choosing, at this stage, the type of formulation of the finite elements used or the physical phenomenon to model. Certain studies can result in using several files of grid.

  This file of grid, in general, is produced by orders integrated into *Code_Aster* starting from a file coming from a software of grid used out of preprocessor (SALOMÉ, GIBI, GMSH, IDEAS…). Information which this file must contain is specific to *Code_Aster*. They define classical entities of the finite element method:

  - **nodes** : points defined by one **name** and by theirs **Cartesian coordinates** in space **2D** or **3D**,
  - **meshs** : topological figures **named** plane or voluminal (not, segment, triangle, quadrangle, tetrahedron,…) to which will be able to apply various types of finite elements, boundary conditions or loadings.

  To improve safety of use and comfort of the operations of modeling and examination of the results, one can define, in the file of grid, of the higher levels of entities, having an unspecified property jointly and which could be used directly by their name:

  - **groups of nodes** : named lists of names of nodes,
  - **groups of meshs** : named lists of names of meshs.

  It will be noted, as of now, that all the handled geometrical entities (nodes, meshs, groups of nodes, groups of meshs) are **named by the user** and usable constantly by their name (**8 characters to the maximum**). The user will be able to use this possibility to identify explicitly certain parts of the studied structure and to thus facilitate the examination of the results. The classification of the entities is never clarified: it is used for only in-house to point on the values of the various associated variables.

- the file of **orders** : to define the text of order which allows:

  - of reading and if required enriching the data of the file by grid (or other sources of external results),
  - to affect the data of modeling on the entities of the grid,
  - to connect various operations of treatment: specific calculations, postprocessings,
  - to publish the results on various files.

  The text of order refers to the names of geometrical entities defined in the file of grid. It also makes it possible to define new groups constantly.

From the data-processing point of view, these two files are files **ASCII** in free format. One gives the principal characteristics here of them:

Syntax of the file of grid :

- length of line limited to 80 characters,
- the allowed characters are:
  - 26 capital letters **A-Z** and the 26 tiny ones **a-z** converted automatically in capital letters, except in the texts (provided between quotes),
  - ten figures **0-9** and signs of representation of the numbers (**+ -.**),
  - character **_white underlined** usable in keywords or names,
- a word must always start with a letter,
- character **white** is always a separator,
- character **%** indicate the beginning, until the end of the line, of one **comment**.
- The other rules of reading are specified in the booklet [U3.01.00]

Syntax of the command file :

- syntax related to the language Python, allowing to include instructions of this language
- character **#** indicate the beginning, until the end of the line, of one **comment**.
- The orders must start in column 1, unless they belong to a indenté block (buckles, test)

The other rules of reading are specified in the booklet [U1.03.01].

2 Grid

2.1 General information

The file of grid *Aster* can be written (for really elementary grids) or modified manually with any text editor. It is a file read in free format, structured in subfile or records by imposed keywords.

Various utilities were developed to facilitate the importation of grid in *Code_Aster*. One distinguishes:

- the utilities of conversion which allow the conversion of a file of grid produced by another software package (IDEAS, GIBI, GMSH…) in a file of grid to the format *Aster,*
- the reading command of a file of grid to format MED, produced by Salomé.

2.2 The file of grid *Aster*

The structure and the syntax of the file of **grid** *Aster* are detailed in **Booklet [U3.01.00]**.

The file of grid *Aster* first line until the first occurrence of a line is read begin with the word **END**. **This keyword is obligatory.** The file of grid is structured under - independent files starting with one **keyword** and finished by the imposed keyword **FINSF**.

This file must comprise at least two subfiles:

- coordinates of all the nodes of the grid in a Cartesian reference mark 2D (COOR_2D) or 3D (COOR_3D).
- the description of all meshs (TRIA3, HEXA20, etc…), on which one will affect then physical properties, finite elements, boundary conditions or loadings.

It can possibly contain groups of nodes (GROUP_NO) or of meshs (GROUP_MA) to facilitate the operations of assignment, but also the examination of the results.

It is essential to create explicitly *at this stage meshs located on the borders of application loadings and boundary conditions. One will find then, in the file of grid:*
- *meshs of edge of the elements 2D necessary,*
- *meshs of face of the elements 3D solid masses necessary;*
- *groups of meshs of associated edge and/or face.*

This constraint becomes bearable when one uses an interface, which does the work starting from the indications provided at the time it grid (see the documents PRE_IDEAS [U7.01.01] or PRE_GIBI [U7.01.11]).

2.3 Utilities of conversion

These interfaces make it possible to convert the files, with or without format, used by various software packages or codes computer, with the conventional format of the file of grid *Aster*.

The currently available interfaces are those which make it possible to use maillor IDEAS, maillor GIBI of CASTEM 2000 and maillor GMSH.

2.3.1 Universal file IDEAS

The convertible file is the universal file defined by documentation I-DEAS (see Booklet [U3.03.01]). The recognition of version IDEAS used is automatic.

A universal file IDEAS consists of several independent blocks called "**dated sets".** Each "**set dated"** is framed by the character string **-1** and numbered. "Dated sets" recognized by the interface are described in the booklet [U3.03.01].

2.3.2 The file of grid GIBI

The interface is made using the order PRE_GIBI [U7.01.11]).
The convertible file is the ASCII file restored by the order TO SAVE FORMAT CASTEM 2000. The precise description of the interface is given in [U3.04.01].

2.3.3 The file of grid GMSH

The interface is made using the order PRE_GMSH [U7.01.31]).
The convertible file is the ASCII file restored by the order SAVE of GMSH.

2.4 The file of grid to format MED

The interface is made using the order LIRE_MAILLAGE (FORMAT=' MED') [U4.21.01].

MED (Modeling and Data exchanges) is a neutral format of data developed by EDF R & D and ECA for the data exchanges between computer codes. Files MED are binary and portable files. Reading of a file MED by LIRE_MAILLAGE, allows to recover a grid produced by any other code able to create a file MED on any other machine. This format of data is in particular used for the file-swappings of grids and of results enters *Code_Aster* and Salomé or the tool of refinement of grid LOBSTER.

2.5 The use of incompatible grids

Although the finite element method recommends the use of regular grids, without discontinuity, to obtain a correct convergence towards the solution of the continuous problem, it can be necessary for certain modelings to use incompatible grids: on both sides of a border, the grids do not correspond. The connection of these two grids is then managed on the level of the command file by the keyword LIAISON_MAIL order AFFE_CHAR_MECA. This makes it possible in particular to finely connect a zone with a grid with another zone where one can be satisfied with a coarse grid.

2.6 Adaptive grid

Starting from an initial grid, it is possible to adapt the grid, to minimize the made mistake, using the macro order MACR_ADAP_MAIL, which calls on the software LOBSTER. The software of adaptive grid LOBSTER functions on grids formed by segments, triangles, quadrangles, tetrahedrons, hexahedrons and pentahedrons.

This adaptation of grid is placed after the first calculation with *Code_Aster*. An indicator of the error will have been calculated. According to its value nets by mesh, the software LOBSTER will modify the grid. It is also possible to interpolate fields of temperature or displacement to the nodes of the old grid towards the new one [U7.03.01].

3 Orders

3.1 The command file

The file of **orders** contains a set of orders, expressed in a language specific to *Code_Aster* (which must respect syntax Python). These orders are analyzed and carried out by a software layer of *Code_Aster* called “supervisor”.

3.2 The role of the supervisor

The supervisor carries out various tasks, in particular:

- one **phase of checking** and of interpretation of the command file,
- one **production run** interpreted orders.

These tasks are detailed in the document [U1.03.01].

The command file is treated starting from the line where the first call to the procedure is **BEGINNING ()** or with the procedure **CONTINUATION ()**, and until the first occurrence of the order **END ()**. Orders located front **BEGINNING ()** or **CONTINUATION ()** and afterwards **END ()** are not carried out, but must be syntactically correct).

Syntactic phase of checking:

reading and syntactic checking of each order; any error of detected syntax is the object of a message, but the analysis continues,
checking that all the concepts used as arguments were declared in a preceding order like produced concept of an operator; it is also checked that the type of this concept corresponds to the type required for this argument.

Production run:

the supervisor activates successively the various operators and procedures, which carry out the tasks envisaged.

3.3 Principles and the syntax of the process control language

Modular concept of *Code_Aster* allows to present the code like a succession of independent orders:

procedures, which does not produce results directly, but ensure, amongst other things, the management of the exchanges with the external files,
operators, which carries out an operation of calculation or data management and produces one **concept** result to which the user gives a name.

These concepts represent structures of data, that the user can handle. These **concepts** are **typified** at the time of their creation and could be used only like argument of entry of the corresponding type.

The procedures and the operators thus exchange the necessary information and of the values via **named concepts.**

The complete syntax of the orders and its implications on the drafting of the command file are detailed in the booklet [U1.03.01]. Here an example of some orders is given (extracted the example with accompanying notes in [U1.05.00]):

```
e-mail = LIRE_MAILLAGE  ()

mod1 = AFFE_MODELE (GRID = e-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 (MODEL = mod1
        PRES_REP =_F (GROUP_MA = ('lfa', 'ldf'),
                    CLOSE = f_y))
.....
res1 = MECA_STATIQUE (MODELE=mod1,
                      ......
                      EXCIT=_F (LOAD = charg),
                      ….)

res1 = CALC_CHAMP (reuse=res1, RESULTAT=res1,
                   MODELE=mod1,
                   CONTRAINTE= ('SIGM_ELNO'),
                   DEFORMATION= ('EPSI_ELNO),);
```

Some points generals will be noted, which one can observe on the preceding example:

- any order starts in first column,
- the list of the operands of an order is obligatorily between brackets, as well as the lists of elements,
- one `nom_de_concept` can only appear**only once** in the text of order like produced concept, on the left of the sign `=`,
- **re-use of an existing concept like produced concept**, is not possible that for the operators specified for this purpose. When one uses this possibility (réentrant concept), the order then uses the reserved keyword " `reuse` ".
  - an order is made up of one or more `mot_clé` or `mot_cle_factor`, the latter themselves being composed of a list of `mot_clé` between brackets and preceded by the prefix `_F`. In the example suggested, the order `AFFE_CHAR_MECA_F` use it `mot_clé MODEL` and it `mot_cle_factor PRES_REP`, who is composed of both `mot_clé GROUP_MA` and `NEAR`.

This operation is done:

- maybe with crushing of the initial values. As example let us announce factorization in place of a matrix of rigidity:
  `masted = TO FACTORIZE (reuse=matass, MATR_ASSE= masted)`
- maybe with enrichment of the concept.

3.4 Rule of overload

One **rule of overload** usable, in particular for all the operations of assignment, was added to the rules of use of one mot_cle_factor with several lists of operands:

- the assignments are done by superimposing the effects of different mot_clé,
- in the event of conflict, the last mot_clé overrides the precedents.

Example: one wishes to affect various materials MAT1, MAT2 and MAT3 with certain meshs:

```
to subdue = AFFE_MATERIAU (MAILLAGE= mon_mail
        AFFE = _F (ALL = 'YES', MATER = MAT1),
                  _F (GROUP_MA = 'mail2', MATER = MAT2),
                  _F (GROUP_MA = 'mail1', MATER = MAT3),
                  _F (MESH = ('m7', 'm8'), MATER = MAT3))
```

- One starts by affecting material MAT1 with all the meshs.
- One affects then material MAT2 with the group of meshs mail2 who contains, the meshs m8, m9 and m10.
- One affects finally material MAT3 with the group of meshs mail1 (m5, m6 and m7) and with the meshs m7 and m8 , which is source of conflict since the mesh m7 fact already part of the group mail1. The rule of overload will then be observed and one will obtain finally the field of following material:

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

The progressive effect of the various material assignments is illustrated in the table below.

Name of the mesh	Material field after the 1st assignment	Material field after the 2nd assignment	Material field after the 3rd assignment	final material field
m1	MAT1	MAT1	MAT1	MAT1
m2	MAT1	MAT1	MAT1	MAT1
m3	MAT1	MAT1	MAT1	MAT1
m4	MAT1	MAT1	MAT1	MAT1
m5	MAT1	MAT1	MAT3	MAT3
m6	MAT1	MAT1	MAT3	MAT3
m7	MAT1	MAT1	MAT3	MAT3
m8	MAT1	MAT2	MAT2	MAT3
m9	MAT1	MAT2	MAT2	MAT2
m10	MAT1	MAT2	MAT2	MAT2

3.5 Rule of remanence

The rule of preceding overload must be supplemented by another rule to specify what occurs when one can affect several quantities for each occurrence of a keyword factor.

That is to say for example:

```
CHMEC1=AFFE_CHAR_MECA (  MODELE=MO,
            FORCE_INTERNE= (
```

```
        _F (ALL      = 'YES', FX = 1.          ),
        _F (GROUP_MA = 'GM1',           FY = 2.),
    ))
```

The rule of overload tells us that the second occurrence of FORCE_INTERNE overload the first.

But what is worth FX on a mesh belonging to GM1 ? was it erased by the second occurrence?

If the only rule of overload is observed, FX is not defined on GM1.

The rule of remanence makes it possible to preserve the value of FX.
If the rule of remanence is observed, FX preserve the affected value as a preliminary. All the elements of the model have a value for FX and elements of GM1 for a value for FX and FY.

3.6 Bases memory associated with a study

Code_Aster rest, for the management of all the structures of data associated with the various concepts handled, on the library JEVEUX. This one deals with the management of**memory capacity** asked by the user at the time of the request for execution (parameter **Memory** expressed in Megabytes). This space is frequently insufficient to store central all the structures of data. The library takes then charges some, the management of the exchanges between the main memory and the auxiliary storages on files.

Each entity is affected, during its creation by the code, with one **file of direct access**. This one can be regarded as one **database**, since it contains, at the end of the execution it **repertoire** (names and attributes) which makes it possible to exploit all the segments of values that it contains.

Code_Aster use several databases:

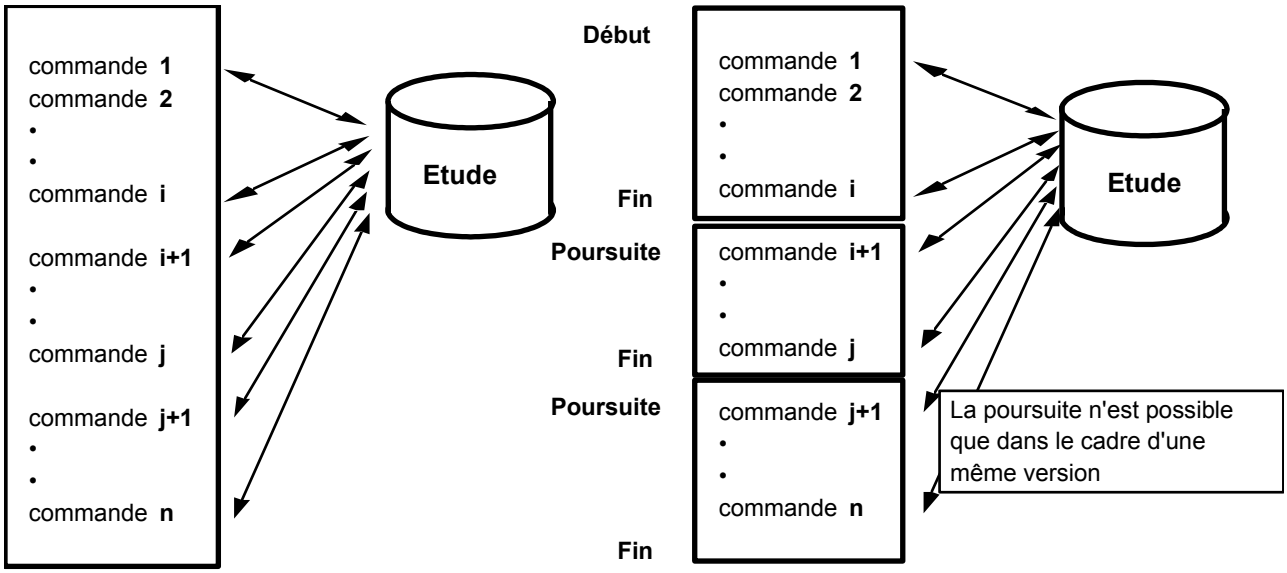
- the database TOTAL, who contains all the concepts produced by the operators, as well as the contents of certain catalogues on which the concepts are pressed; the file associated with this base allows the later continuation of a study. It must thus be managed by the user.
- the other databases, used only by the Supervisor and the operators, during an execution, do not require a particular intervention of the user.

To carry out a study, it is to ask for the sequence of several orders:

- procedures to exchange files with the external world,
- operators to create concepts produced as course of operation of modeling and calculation.

The orders which correspond to this sequence of operations can be realized various ways, starting from the single achievable module of *Code_Aster* :

- in only one sequential execution, without intervention of the user,
- by splitting the study in several successive executions, with re-use of the former results; starting from the second execution, the access to the database is done in **continuation**; at the time of a continuation, one can ask again the last order, if it stopped prematurely (lack of time, incomplete or incorrect data detected in production run,…).



To manage these possibilities, it will be noted that three orders play a central role. It is those which correspond to **procedures** who activate the supervisor:

- **BEGINNING ()** **obligatory** for **first** execution of a study,
- **CONTINUATION ()** **obligatory** from **second** execution of a study,
- **END ()** **obligatory** for all the executions.

For a given study, one can subject command files having the following structure:

Remarks :

- *The order INCLUDE allows to include in a flood of orders the contents of another command file. This in particular makes it possible to preserve a file of the principal orders readable and to place in annexed files of the bulky numerical data (ex: definition of functions).*
- *The command files can be cut out in several files which will be carried out one after the other, with intermediate safeguard of the database. For that, it is necessary to define the successive command files, whose suffix will be: .com1 , .com2 ,..., .com9. The executions of these files are connected. The database of the last execution which finished well is preserved.*

3.7 Assistance with the definition of the values

3.7.1 Substitution of values

It is possible to parameterize a command file.

For example:

```
Eptub = 26.187E-3

Rmoy  = 203.2E-3

Rext  = Rmoy+ (EPtub/2)

will cara  = AFFE_CARA_ELEM (MODEL =  model
         BEAM =_F (  GROUP_MA = all,  SECTION: 'CIRCLE',
                     CARA    = ('R', 'EP'),   VALE  = (Rext, EPtub)))
```

3.7.2 Functions of one or more parameters

It is also often necessary to use sizes functions of other parameters.

Those can be:

- that is to say definite on an external file read by the order LIRE_FONCTION.
- that is to say defined in the command file by:
  - DEFI_CONSTANTE who produces a concept function with only one constant value,
  - DEFI_FONCTION who produces a concept function for a size function of a real parameter,
  - DEFI_NAPPE who produces a concept function for a list of functions of the same size, each element of the list corresponding to a value of another real parameter.
    The concept produced by these operators is of type function and can be used only as argument of operands which accept this type. The operators who accept an argument of the type function have as a suffix F (ex: AFFE_CHAR_MECA_F). The functions in this case are defined point by point, with a linear interpolation by default, therefore closely connected by pieces.

    The functions created are discrete tables of the sizes specified with creation. At the time of a search for value, one proceeds according to the specified characteristics, by direct research or interpolation in the table (linear or logarithm). One can specify, during the creation of the function, the prolongation out of field of definition of the table, with various rules, or prohibit it.
- that is to say definite using their analytical expression by the operator FORMULA. For example:

```
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 = FORMULA (VALE = ''' COS (OMEGA*INST) ''')
F1=CALC_FONC_INTERP (FONCTION=F, VALE_PARA= linst,
                     NOM_RESU=' ACCE',)
```

The analytical function $F(t)=\cos(\Omega t)$ is then calculated by CALC_FONC_INTERP for the moments of the list linst list of moments T.

3.8 How to write its command file with EFICAS?

To write a command file of *Code_Aster*, most immediate consists starting from an example already written by others. In particular, the whole of the tests of *Code_Aster* often constitute a starting good base for a new modeling. They are documented as a documentation of validation.

But there is better: the tool **EFICAS** allows to write in an interactive way and convivial its command file, by proposing for each order the list of the possible keywords by checking syntax automatically, and by giving access to the documentation of the Instruction manual (booklets [U4] and [U7]).

4 Great stages of a study

The great stages of a study are in the case general:

- preparation of the work, which finishes after the reading of the grid,
- modeling during which are definite and affected all the properties of the finite elements and materials, the boundary conditions and the loadings,
- calculation can then be carried out by the execution of total methods of resolution [U4.5-], which are possibly based on orders of calculation and assembly of matrix and vectors [U4.6-]
- operations of postprocessings complementary to calculation [U4.8-],
- operations of impression of the results [U4.9-]
- operations of exchange of results with other software (graphic visualization for example) [U7.05-]

4.1 To start the study and to acquire the grid

One will not reconsider here the possible fragmentation of the command file, which was presented in a preceding paragraph.

The first achievable order is:

```
BEGINNING ( )
```

The argument of this order are useful only for the maintenance actions or in the case of very large studies.

For the reading of the grid, coming from a software of external grid, one can operate in two ways:

- to convert the file of a software package into a file with the format *Code_Aster* by an execution separated, which allows, possibly, to modify it by word processing and to preserve it:

```
BEGINNING ( )
PRE_IDEAS ( )
END ( )
```

the normal study will be able to then begin for example by:

```
BEGINNING     ( )
my =   LIRE_MAILLAGE    ( )
```

- to convert the file right before the lira:

```
BEGINNING     ( )
PRE_IDEAS     ( )
my =   LIRE_MAILLAGE    ( )
```

4.2 To assign data of modeling to the grid

To build the modeling of a mechanical problem, thermal or acoustic, it is essential to assign to the topological entities grid:

- a model of finite element,
- properties of the materials (law of behavior and parameters of the law),
- geometrical or mechanical characteristics complementary,
- boundary conditions or loadings.

These assignments are obtained by various operators whose name is prefixed by AFFE_. The syntax and the operation of these operators use the facilities brought by the rules already mentioned previously on the use of the keywords factor.

4.2.1 Definition of a field of assignment

To carry out an assignment, it is essential to define a field of assignment per reference to the names of the topological entities defined in the file grid. Five keywords are usable for that, according to the specification of the operator:

- to refer to all the grid by `TOUT= 'YES'`
- to assign to meshs by `MAILLE=` (list of names of meshs)
- to assign to groups of meshs by `GROUP_MA=` (list of names of groups of meshs)
- to assign to nodes by `NOEUD=` (list of names of nodes)
- to assign to groups of nodes by `GROUP_NO=` (list of names of groups of nodes)

4.2.2 To affect the type of finite element

On the meshs of the studied structure, which are at this stage only topological entities, it is essential to affect:

- one or more phenomena studied: `'MECHANICAL', 'THERMAL', 'ACOUSTIC' ;`
- a model of finite element compatible with the topological description of the mesh. This assignment induces an explicit list of degrees of freedom in each node and a law of interpolation in the element.

One uses for that the operator `AFFE_MODELE` [U4.41.01], which can be called several times on the same grid. It uses the rule of overload.

Foot-note :

For a study with several treated phenomena (`'MECHANICAL' , 'THERMAL'`), it is essential to build a model for each phenomenon, by as many calls to `AFFE_MODELE` . On the other hand, for a given calculation (mechanical, thermal,…) one needs one and only one model.

To know the characteristics of the various finite elements available, one will refer to the booklets [U2-], and [U3-].

4.2.3 To affect material characteristics

It is necessary to assign to this stage of the material characteristics, and the associated parameters, with each finite element of the model (except for the directly definite discrete elements by a matrix of rigidity, mass and/or damping). In other words, `DEFI_MATERIAU` is used to define a material and `AFFE_MATERIAU` is used to define a material field by association of the grid. For a given calculation, one needs one and only one field of materials.

One can also use the validated characteristics of the catalogue material using the procedure `INCLUDE_MATERIAU` [U4.43.02].

A certain number of models of behavior are usable: rubber band, rubber band orthotropic, thermal, acoustic, elastoplastic, elastoviscoplastic, endommagment. Let us note that it is possible to define several material characteristics for the same material: rubber band and thermics, elastoplastic, thermoplastic,…

4.2.4 To assign characteristics to the elements

During the use of certain types of elements, for the phenomenon `'MECANIQUE'`, the geometrical definition deduced from the grid does not make it possible to describe them completely.

One must assign to the meshs the missing characteristics:

- for **hulls** : the constant thickness on each mesh and a reference mark of reference for the representation of the state of stress,
- for **beams, bars and pipes** : characteristics of the cross section, and possibly orientation of this section around neutral fibre.

These operations are accessible by the operator AFFE_CARA_ELEM [U4.42.01]), which uses, to simplify the drafting of the order, the rule of overload.

Another opportunity is given by this operator: that to introduce, directly in the model, of **matrices** of rigidity, of mass or damping on meshs POI1 (or of the nodes) or of the meshs SEG2. These matrices correspond to the types of discrete finite elements with 3 or 6 degrees of freedom per node DIS_T or DIS_TR who must be affected at the time of the call to the operator AFFE_MODELE.

4.2.5 To affect the boundary conditions and the loadings

These operations are, in general, essential. They are carried out by several operators whose name is prefixed by AFFE_CHAR or CALC_CHAR. On the same model, one will be able to carry out several calls to these operators to define, as the study, of the boundary conditions and/or the loadings.

The operators used differ with the phenomenon:

'MECHANICAL'	AFFE_CHAR_CINE	
	AFFE_CHAR_MECA	data of the type reality only
	AFFE_CHAR_MECA_F	data of the type function
'THERMAL'	AFFE_CHAR_THER	data of the type reality only
	AFFE_CHAR_THER_F	data of the type function
'ACOUSTIC'	AFFE_CHAR_ACOU	data of the type reality only

Moreover, one can establish the seismic loading to carry out a calculation of answer moving relative compared to the supports, using the order CALC_CHAR_SEISME.

The boundary conditions and loadings can be defined according to their nature:

- with the nodes,
- on meshs of edge (edge or face) or meshs support of finite elements, created in the file grid. On these meshs the operator AFFE_MODELE has affected the types of finite elements necessary.

For the detailed description of the operands of these operators and the rules of orientation of the meshs support (total reference mark, local reference mark or unspecified reference mark) one will refer to the documents [U4.44.01], [U4.44.02], and [U4.44.04].

The boundary conditions can be dealt with two ways:

- by "elimination" of the degrees of freedom imposed (for linear mechanical models implementing only boundary conditions kinematics (degrees of freedom blocked) **without linear relation.** One will define in this case the boundary conditions by the order AFFE_CHAR_CINE.
- by dualisation [R3.03.01]. This method, thanks to its greater general information, makes it possible to treat all the types of boundary conditions (degree of freedom imposed, linear relations between degrees of freedom,…) ; the method used results in adding 2 multipliers of LAGRANGE for each ddl imposed or each linear relation.

Each concept produced by the call to these operators, of type AFFE_CHAR, corresponds to a system of boundary conditions and loadings indissociable. In the orders of calculation, one can incorporate these concepts while providing for the operands LOAD a list of concepts of this type.

4.3 To carry out calculations by total orders

4.3.1 THERMAL analysis

To calculate to it (S) field (S) of temperature corresponding to a linear thermal analysis or not linear:

- **stationary** (moment 0),
- **evolutionary** whose moments of calculation are specified by an as a preliminary definite list of realities

The orders to be used are:

- THER_LINEAIRE for a linear analysis [U4.54.01],
- THER_NON_LINE for a nonlinear analysis [U4.54.02],
- THER_NON_LINE_MO for a problem of live loads in permanent mode [U4.54.03].

Calculations of the matrices and elementary and assembled vectors necessary to the implementation of the methods of resolution are dealt with by these operators.

4.3.2 STATIC analysis

To calculate the mechanical evolution of a structure subjected to a list of loadings:

- MECA_STATIQUE [U4.51.01]: linear behavior, with superposition of the effects of each loading,
- MACRO_ELAS_MULT [U4.51.02]: linear behavior, by distinguishing the effects of each loading,
- STAT_NON_LINE [U4.51.03]: quasi-static evolution of a structure subjected to a history of loading in small or great transformations, made of a material whose behavior is linear or not linear, with taking into possible account of the contact and friction.

If this mechanical calculation corresponds to a study of **thermoelasticity**, one will refer to one **moment** thermal calculation already carried out. If the material were defined with **characteristics depending on the temperature**, those are interpolated for the temperature corresponding to the moment of required calculation.
For the problems of thermohydromecanic coupled, it is the operator STAT_NON_LINE who is used to solve simultaneously the 3 problems thermics, hydraulics and mechanics.
Calculations of the matrices and elementary and assembled vectors necessary to the implementation of the methods of resolution are dealt with by these operators.

4.3.3 MODAL analysis

To calculate the clean modes and eigenvalues of the structure (correspondent to a vibratory problem or a problem of buckling).

- CALC_MODES with OPTION among ['BAND', 'CENTER', 'PLUS_*', 'ALL'] [U4.52.02]: calculation of the clean modes by simultaneous iterations; the eigenvalues and vector clean are real or complex;
- CALC_MODES with OPTION among ['NEAR', 'ADJUSTS', 'SEPARATE'] [U4.52.02]: calculation of the clean modes by iterations opposite; the eigenvalues and vector clean are real or complex;
- CALC_MODES with OPTION=' BANDE' and cutting of the band in several sub-bands [U4.52.02]: barge modal analysis by cutting out the interval of frequency under intervals;
- MODE_ITER_CYCL [U4.52.05]: calculation of the clean modes of a structure with cyclic repetitivity starting from a base of real clean modes.

These operators require as a preliminary the calculation of the matrices assembled with the order ASSEMBLY [U4.61.21] or ASSE_MATRICE [U4.61.22].

4.3.4 DYNAMIC analysis

To calculate the dynamic response, linear or not linear, of the structure. Several operators are available. One can quote for example:

DYNA_LINE_TRAN [U4.53.02]: temporal dynamic response of a linear structure subjected to a transitory excitation,
DYNA_LINE_HARM [U4.53.02]: dynamic response complexes of a linear structure subjected to a harmonic excitation,
DYNA_TRAN_MODAL [U4.53.21]: transitory dynamic response in coordinated generalized by modal recombination.

These three operators require as a preliminary the calculation of the assembled matrices [U4.61-].

DYNA_NON_LINE [U4.53.01]: temporal dynamic response of a nonlinear structure subjected to a transitory excitation, which also calculates the assembled matrices.

4.4 Results

Results produced by the operators realizing of calculations by finite elements [U4.3-], [U4.4-] and [U4.5-] are of two principal types:

- maybe of the type cham_no or cham_elem (by node or element) when it acts operators producing one field (for example TO SOLVE),
- maybe of the type RESULT strictly speaking which gathers whole of fields.

A field in a concept of the type RESULT is located:

- by a variable of access which can be:
  - a simple sequence number referring to the order in which the fields were arranged,
  - a parameter preset according to the type of the concept RESULT :
    - frequency or number of mode for one RESULT type mode_meca,
    - moment for one RESULT type evol_elas, temper, dyna_trans or evol_noli.
- by a reference symbol of field which refers to the type of the field: displacement, speed, state of stress, efforts generalized,…

Besides the variables of access, other parameters can be attached to a kind of concept RESULT.

The various fields are built-in a concept result:

- maybe by the operator who created the concept, a total order (MECA_STATIQUE, STAT_NON_LINE,…) or a simple order (CALC_MODES, DYNA_LINE_TRAN,…),
- maybe during the execution of an order which makes it possible to add an option of calculation in the form of a field by element or of a field to the nodes (CALC_CHAMP) ; it is then said explicitly that one enriches the concept:

```
resul = operator (reuse=resu1, RESULT = resul…)   ;
```

4.5 To exploit the results

The whole of the preceding orders has building permit various concepts which are exploitable by operators of postprocessing of calculations:

- operators generals of postprocessing (see booklet [U4.81]), for example CALC_CHAMP, POST_ELEM, POST_RELEVE_T,
- operators of breaking process (see booklet [U4.82]), for example CALC_G,
- metallurgy operator: CALC_META,
- mechanical postprocessing statics (see booklet [U4.83]), for example POST_FATIGUE, POST_RCCM,
- dynamic mechanical postprocessing (see booklet [U4.84]), for example POST_DYNA_ALEA, POST_DYNA_MODA_T.

- operators of extraction:
  - of a field in a concept result CREA_CHAMP [U4.72.04],
  - of a field in coordinates generalized for a dynamic calculation with modal base RECU_GENE [U4.71.03],
  - of a function of evolution of a component starting from a concept result RECU_FONCTION [U4.32.03],
  - and of restitution of a dynamic response in the physical base REST_GENE_PHYS [U4.63.31],
  - an operator of postprocessing of functions or tablecloths CALC_FONCTION who allows searchs for peaks, of extremums, linear combinations,… [U4.32.04].

Lastly, two procedures IMPR_RESU [U4.91.01] and IMPR_FONCTION [U4.33.01] the impression and possibly the creation of exploitable files by other software packages allow, in particular of graphic visualization. One will retain in particular graphic visualization by IDEAS, GMSH, or GIBI whatever the tool for grid used at the beginning.

5 Error message and print file

code_aster writing relative information with calculation in a file whose significance is the following one:

File	Contents
MESSAGE	Information on the course of calculation. Repetition of the command file, provided and its interpretation by **code_aster**. Execution time of each order. Messages emitted during the execution, *cf.* hereafter.

Other files are used for the interfaces with the programs of graphic examination.

One distinguishes various types of messages who are described below.

Code	Type of message
F	error message fatal, the execution stops after various impressions. The concept created by the current order is lost. Nevertheless, the concepts produced previously are validated and bases it TOTAL is saved. It is used within the framework of the serious detection of error which cannot allow the normal continuation of an order of *Code_Aster.*
E	error message, the execution continues a little bit: this kind of message makes it possible to analyze a series of errors before the program stop. *The emission of a message of the type* <E> *is always followed by the emission of a message of the type* <F>. The produced concepts are validated and bases it TOTAL is available for one CONTINUATION.
S	error message, the concepts created during the execution, including current order, are validated by the supervisor, the execution stops with “clean” closing of the base TOTAL. It is thus reusable in CONTINUATION. This message makes it possible in particular to be secured against a problem of convergence or time during an iterative process.
With	message of alarm. The number of messages of alarm is limited automatically to 5 messages **successive** identical. **It is recommended to the users who have messages of the type A “to repair” their command file to make them disappear.**
I	message of information bound for the users. It is recommended to become acquainted with the contents of these messages, without that requiring however a modification of the command file or a reinforced vigilance.

NB: Lbe exceptions behave exactly like errors <S>. It is by way of errors <S> adapted to a typical case.