How to read the documentation of the orders

Summary:

This note is a guide of reading of the U4 booklets and U7 of Manuel d'Utilisation.

She explains in particular the significance of the méta-characters and the typographical conventions used for the description of the syntax of the orders.

All examples set here are given as illustration and do not replace the complete description of the orders appearing in the booklets U4 and U7.

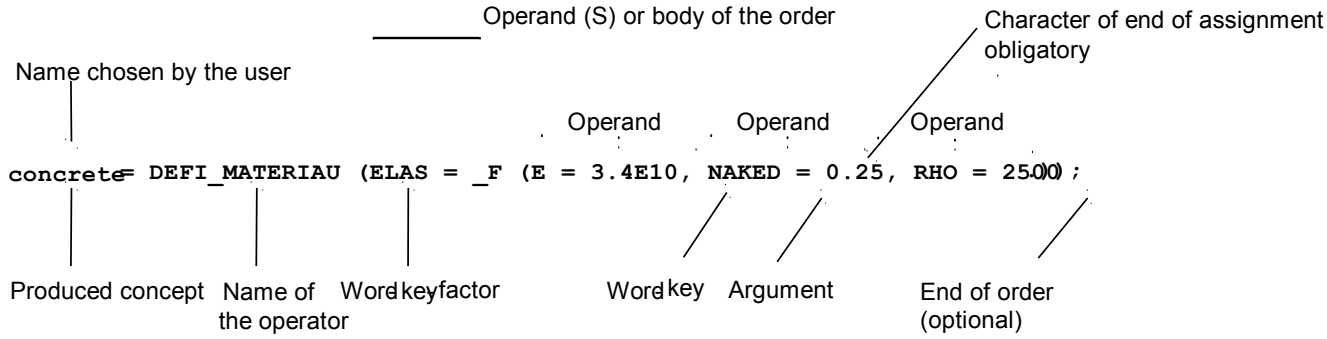
Contents

1 Recalls on the syntax of the orders of *Code_Aster*

The process control language and its supervisor are completely described in the document [U1.03.01]. One recalls here some notions on syntax of the orders of *Code_Aster*.

In *Code_Aster*, one understands by the generic term of orders at the same time them **operators**, them **procedures** and macro-orders of the process control language. An operator provides one **produced concept** typified (by the operator) and named by the user. A procedure does not generate a produced concept, it achieves **actions** such as impressions or resource allocations.

In the example below, one recalls the vocabulary which is used in the description of the orders.



Terminology *Aster*

An operand is thus the unit consisted a keyword and its argument. However, in the documentation of the orders, one often indicates the operands of an operator or a procedure by the name of their keyword. For example: RHO, simple keyword, or ELAS, keyword factor.

The term of **produced concept** is generic for all the operators, it is the result of the work of the operator.

Here in the example DEFI_MATERIAU, there was creation of the structure of data of the type to subdue (material), named concrete by the user. It gathers the denominations (keyword E, NAKED, RHO) and the values (arguments 3.4E10, 0.25, 2500.) mechanical elastic characteristics (keyword factor ELAS) material.

The term of concept **of standard result** applies to the exits of the operators of calculation, i.e. physical fields of sizes (displacements, temperatures, constraints, efforts, modes, etc…) on the nodes or the meshs at various moments or for various frequencies.

The concept result comprises in general **under types.**

2 Standard plan of the documents of use of the orders

Each document of presentation of an order comprises the following chapters:

- Goal,
- Syntax,
- Operands,
- Examples (possibly).

This presentation makes it possible to the user to find in only one document all knowledge necessary to the implementation of an order.

3 Paragraph Drank

One states the functionality filled by the order (actions carried out). One also specifies the types of the expected concepts as starter and the produced concept, as well as characteristics of the order.

This paragraph is also displayed by the search engines; it thus contains only text without equations or formula.

Example: Operator STAT_NON_LINE [U4.51.03]

Goal :

To calculate the quasi-static mechanical evolution of a structure into nonlinear.

Nonthe linearity is related either to the behavior of material (for example plastic), or with the geometry (for example in great displacements). To have details on the method of resolution employed, one will refer to the reference material [R5.03.01].

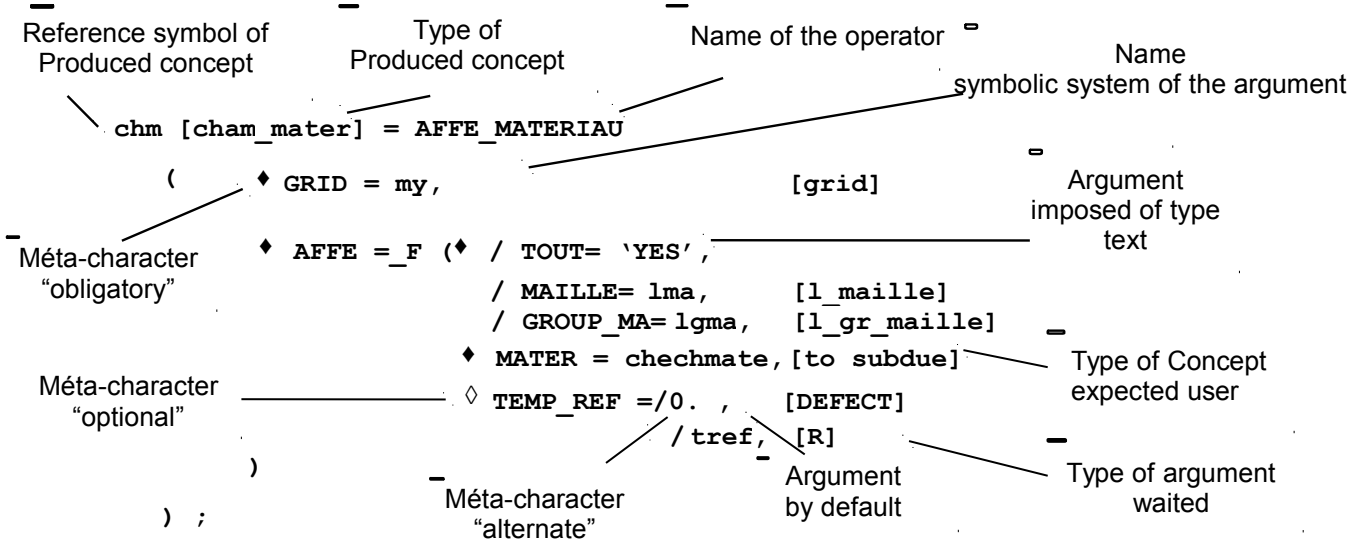
The evolution can be studied in several successive work (réentrant concept), either in continuation (the calculated last moment is the initial moment of following calculation), or recovery some on the basis of one former moment.

If the time necessary to carry out calculation is not sufficient, the program stops, but the already calculated results are saved if a database were defined in the profile of study of the user. Product a structure of data of the type evol_noli.

4 Paragraph Syntax

One gives, in this paragraph, the whole of the operands of the order. One specifies, for each operand, using méta-characters and of indentations suitable for the typographical presentation of the orders (cf example of the operator AFFE_MATERIAU) :

- the name of the operator,
- the name of the keywords,
- the reference symbols user of the concept produces and the arguments of the keywords,
- obligatory or optional character of the operands (statute),
- alternatives in the choices of the operands,
- types of the arguments expected by the keywords,
- the values by default taken by the arguments in the case of optional operands,
- the type of the produced concept, when it is about an operator.



Presentation of the syntax (partial) of the operator AFFE_MATERIAU

4.1 Méta-characters of statute of operands (♦ ◊ / |)

Four méta-characters are used to indicate the statute of the operands. It is necessary to understand here by statute of the operands their obligatory or optional declaration and the nature of the alternatives in the choices of the operands.

These méta-characters are not part of the process control language. They have only one function of documentary presentation and do not have to thus be used for the drafting of the command file.

4.1.1 Obligatory or optional operands

They are located by the presence at the top of a black or white rhombus.

- ♦ black rhombus: it is obligatory to declare in the order the operands which follow this sign.
- ◊ white rhombus: the declaration of the operands which follow this sign is optional. In the event of absence of the operand, the order will affect possibly one or of the values by default.

Example : operator DEFI_LIST_ENTI (definition of a list of strictly increasing entireties whose values are regularly spaced)

```
Li =  DEFI_LIST_ENTI
                (    ♦ BEGINNING = deb. ,
                   ◊  INTERVAL = _F    (  ♦  JUSQU_A   =  yew,
                                          ♦  NOT       =  ipas,
                                       )
                 )
```

- It is obligatory to declare the operand identified by the keyword BEGINNING and to provide deb. who is the first entirety of the list to be built.
- It is not obligatory to declare the operand identified by the keyword factor INTERVAL. In this case the list of entireties will be summarized with only one entirety of value deb. (this is specified in the description of the operands).
- If the operand INTERVAL is declared, then it is obligatory to declare the operand JUSQU_A who specifies the whole end yew interval to be cut out with a constant step and the operand NOT who indicates the step ipas of interval division.

4.1.2 Alternatives in the choice of the operands

They are located by the presence at the top of each choice of the alternative:

- of one / (slash): exclusive alternative, only one choice among those proposed,
- of one | (pipe, semi colonist): nonexclusive alternative, one or more choice among those proposed.

Example of exclusive alternative: operator AFFE_MODELE (assignment of the type of finite elements on whole or part of a grid).

```
Mo   = AFFE_MODELE (
       ♦      GRID = my
       ♦      AFFE = _F (   ♦   /   ALL    =   'YES',
                                /   MESH     =   e-mail ,          [l_maille]
                                /    NODE    =   noeu   ,          [l_noeud]
                                /   GROUP_MA  =   g_mail ,         [l_gr_maille]
                                /   GROUP_NO  =   g_noeu ,         [l_gr_noeud]
                  ..............
                            )
                               )   ;
```

In the operand AFFE (obligatory) it should be indicated where will be affected, on the grid, the type of finite element specified in the operands PHENOMENON and MODELING same order:

- maybe on all the grid (ALL),
- maybe on certain meshs (MESH),
- maybe on certain nodes (NODE),
- maybe on certain groups of meshs (GROUP_MA),
- maybe on certain groups of nodes (GROUP_NO).

Example of nonexclusive alternative:

operator AFFE_CHAR_MECA operand DDL_IMPO (assignment of displacements imposed on degrees of freedom).

```
DDL_IMPO = _F (   ♦   /   ALL    =      'YES',
                       / NODE    =  lno,      [l_noeud]
                       / GROUP_NO=     lgno,     [l_gr_noeud]
                      /   MESH  =      lma ,     [l_maille]
                      /   GROUP_MA=    lgma,     [l_gr_maille]
                  ♦   |   DX   =       ux  ,     [R]
                      |   DY   =       uy  ,     [R]
                      |   DZ   =       uz  ,     [R]
                      |   DRX  =        □ X   ,  [R]
                      |   DRY MARTINI =      □ there  ,  [R]
                      |   DRZ  =        □ Z   ,  [R]
                      |   GRX  =       G   ,     [R]
                      |   PRES=        p   ,     [R]
                      |   PHI  =       □   ,     [R]
                      |   TEMP=        T   ,     [R]
                      |   PRE1=        pr1 ,     [R]
                      |   PRE2=        pr2 ,     [R]
                  )
```

In this operator, it is necessary to specify obligatorily:

- the scope of application on the grid: everywhere (ALL), on certain nodes (NODE) or on certain groups of nodes (GROUP_NO),
- on which degrees of freedom with which specified values by the user.

Méta-character | indicate that the user can impose a value of displacement on **one** (the symbol ♦ indicate that one needs at least one of them) or **several** degrees of freedom (DX, DY, DZ, DRX, DRY MARTINI, DRZ, GRX, NEAR, PHI, TEMP, PRE1, PRE2) beforehand indicated nodes.

4.1.3 Combinations of the méta-characters of choice of the operands

These méta-characters can be combined to illustrate the multiplicity of the choices in certain orders.

Example : order DEFI_MATERIAU (definition of a material by its properties of behavior)

For a study of thermomechanics, one needs to define a material having **at the same time** mechanical characteristics (ELAS) and thermics (THER) from where use of the pipe: |

But in each choice, one is obliged to choose if the properties of material are dependent (_FO) or not of the temperature from where use of the slash: / ; cf below:

```
my =   DEFI_MATERIAU     (      |  / ELAS = _F   (  ♦   E =       yg,
                                                    ♦ NAKED =      naked,
                                                    ◊ RHO =     rho,
                                                    ◊ ALPHA =   dil,
                                                 )
       ......
                                  / ELAS_FO =_F (  ♦   E =      f1,
                                                    ♦ NAKED =      f2,
                                                    ◊ RHO =     f3,
                                                    ◊ ALPHA =   f4,
                                                 )

                                  |  / THER = _F   (  ♦    RHO_CP =     CP,
                                                     ♦    LAMBDA = ,
                                                  )
       ......
                                  / THER_FO =_F(  ♦    RHO_CP = g1,
                                                     ♦   LAMBDA =  g2,
                                                  )
       ......
                            );
```

4.2 Méta-characters of the type of concept or argument

Like the méta-characters of statutes of operands, the hooks [] and the star * are not part of the process control language. They have only one function of documentary presentation.

4.2.1 Types of concepts or arguments []

They frame the type of the concepts produced as well as the type of the arguments.

Example : order AFFE_MODELE (Assignment of the finite elements on the meshs of a grid)

```
Mo [model] = AFFE_MODELE
                    (  ♦  GRID = my,                    [grid]
                       ♦  AFFE =_F  (  ♦  /  ALL= 'YES',
                                          /  MESH  = e-mail, [l_maille]
.....................
        )
```

In the example above, one thus specifies that the concept produced by AFFE_MODELE is of type model and that the concept expected like argument of the keyword MESH must be of type l_maille (i.e list of mesh).

4.2.2 Type of the produced concept [*]

This méta-character indicates that the type of the produced concept, or under type of the produced concept of standard result, depends on the types of the arguments of certain operands. In this case the various possibilities are registered after the syntax of the order.

Example: order CREA_CHAMP

In this example, ch2 will be a field with the nodes, a map or a field by element according to the value of TYPE_CHAM.

```
ch2 [*] = CREA_CHAMP
     (  ♦ TYPE_CHAM =  / 'NOEU_xxxx',                  [KN]
                       / 'CART_xxxx',
                       / 'ELGA_xxxx',
                       / 'ELNO_xxxx',
                       / 'ELEM_xxxx',
     )
```

If	TYPE_CHAM =	'NOEU_TEMP_R'	then [*] =	CHAM_NO_TEMP_R
		'NOEU_DEPL_R'		CHAM_NO_DEPL_R
		…		
		'CART_TEMP_R'		CARTE_TEMP_R
		'CART_DEPL_R'		CARTE_DEPL_R
		…		

4.3 Comments

For certain complex orders such as AFFE_CARA_ELEM or DEFI_MATERIAU for example, the character of comment is employed to comment on the alternatives of the operands. It has the same direction that in the process control language and is interpreted like such by the supervisor.

Example for AFFE_CARA_ELEM :

```
POUTRE=_F (♦ / MESH =      lma,                          [l_maille]
             / GROUP_MA =     lgma,                       [l_gr_maille]
          ♦  /  SECTION = 'GENERAL',
                /  # constant section
                   ◊  CARA= | 'With'
                            | 'IY' | 'IZ'
list of the choices            | 'AY' | 'AZ'
possible for one         | 'EY' | 'EZ'
constant section         | 'JX'
                            | 'RY' | 'RZ' | 'RT',
                /  # variable section
                   ◊  CARA= | 'A1'  | 'A2'
                            | 'IY1' | 'IY2' | 'IZ1' | 'IZ2'
list of the choices            | 'AY1' | 'AY2' | 'AZ1' | 'AZ2'
possible for one         | 'EY1' | 'EY2' | 'EZ1' | 'EZ2'
variable section            | 'JX1' | 'JX2'
                            | 'RY1' | 'RY2' | 'RZ1' | 'RZ2' | 'RT1' | 'RT2',
.....
      )
```

4.4 Types of the arguments expected by the keywords

The keywords of the operands expect arguments which correspond, in general, with four classes:

- values, one then specifies by a reference symbol the accepted data-processing type (real, whole, character string, etc…),
- imposed texts, then texts ('YES', 'HY1') are indicated between quotes,
- names of topological entities simple (name of node, meshs, or lists of names), declared in the file of grid, or the names of groups of nodes or meshs, or lists of names of groups of nodes or meshs,
- names and lists of names of concepts produced by the operators.

The table below gathers all the principal types of the arguments expected by the keywords:

1) [R]	reality	3.
[l_R]	list of realities	(1. , 3. , 7.)
[I]	entirety	7
[l_I]	list of entireties	(9, 6,1,9)
[C]	complex	IH 1.1,7.8 or MP 10. , 1.57
[l_C]	list of complexes	(IH 1.1,7.), (IH 4.7,9.)
[TXM]	unconstrained text (name of TITLE…)	**'my title'**
[KN]	text lower or equal to N characters	'INST'
[l_Kn]	list of texts lower or equal to N characters	('SIXX', 'SIYY', 'SIXY')
[node]	name of node	N23
[l_noeud]	list of names of nodes	(N23, N24, N25)
[gr_noeud]	name of group of nodes	NBORD6

[l_gr_noeud]	list of names of groups of nodes	(NBORD, NBASE, NBORD)
[mesh]	name of mesh	M34
[l_maille]	list of name of mesh	(M34, M35)
[gr_maille]	name of group of meshs	MPIQUAGE
[l_gr_maille]	list of names of groups of meshs	(MSOM, MDROI, MGA)
[type_concept]	type of concept (or field) produced beforehand with generally automatic checking of the type	monresu
[l_type_concept]	list of the type of concept user	(resu1, resu2)

4.5 Types of the concepts produced in *Aster*

One uses the méta-character of choice of exclusive alternative/to mean the plurality of concept expected behind a keyword.

Example: operator ASSE_MATRICE (assembly of the elementary matrices contained in a list of concepts of the type matr_elem_*.)

```
my [matr_asse_*] = ASSE_MATRICE
               (  ♦  MATR_ELEM =     lmel,  /          [l_matr_elem_DEPL_R]
                                            /          [l_matr_elem_DEPL_C]
                                            /          [l_matr_elem_TEMP_R]
                                            /          [l_matr_elem_TEMP_C]
                                            /          [l_matr_elem_PRES_R]
                                            /          [l_matr_elem_PRES_C]
…
               );
```

```
if MATR_ELEM       [matr_elem_DEPL_R]      then      [*]      □     DEPL_R
                  [matr_elem_DEPL_C]                          □     DEPL_C
                  [matr_elem_TEMP_R]                          □     TEMP_R
                  [matr_elem_TEMP_C]                          □     TEMP_C
                  [matr_elem_PRES_R]                          □     PRES_R
                  [matr_elem_PRES_C]                          □     PRES_C
```

In the example above the concept expected in argument of MATR_ELEM can be various types and type of the last concept in argument by the user will depend (according to stated rules Ci - above) typing on the concept produced by the operator ASSE_MATRICE.

5 Paragraph Operands

One describes, for each operand the direction of the operand for this order, the nature and the type of the arguments expected by the keywords, and the restrictions and difficulties of employment.

For example, in the documentation of the operator AFFE_MATERIAU, for the operand AFFE, operand intended to specify on which (S) entity (S) topological (S) of the grid of name my the material of name will be affected chechmate product by the operator DEFI_MATERIAU, one will read:

♦ AFFE

Keyword factor which makes it possible to affect various materials on “pieces” of the grid.

/ ALL = 'YES',

This keyword makes it possible to affect on all the meshs of the grid.

/ GROUP_MA = lgma,

This keyword makes it possible to affect on a list of groups of meshs of the grid.

/ MESH = lma,

This keyword makes it possible to affect on a list of meshs of the grid.

With each group of meshs, (keyword GROUP_MA) or each list of meshs (keyword MESH), or with all the grid (keyword ALL) a material is affected chechmate, which is a concept produced by one of the operators DEFI_MATERIAU [U4.43.01] or DEFI_COMPOSITE [U4.42.03].

If a mesh appears explicitly (or implicitly) in several occurrences of the keyword factor AFFE, the rule of overload is observed: it is the last assignment which precedes [U2.01.08].

6 Phases of checking/execution

The paragraph Syntax of the documentation of use is the exact reflection of the catalogue of the order. This catalogue is a file which understands, written in the language of the supervisor, all the rules on the keywords: presence, exclusion, implication, contained…

Editor EFICAS exploits this catalogue of order and allows so the user, with final the made up file is valid, to obtain a correct command set.

With the execution of the study, the supervisor of *Code_Aster* reproduced the same task of syntactic checking: either overall for all the file, or while alternating with the execution, orders by order.

Moreover, during the execution itself of the orders (entered part FORTRAN of the source code), of the additional checks can be made. They are constraints impossible to manage on the level of the process control language (equality of cardinals of different lists…).

7 Typography and indentations

For the legibility of the documents concerning to the orders, all that refers to syntax is printed in `organizes Courier 10 points`. One differentiates the various types of functional elements (produced concept, keyword, keyword factor, argument) by the use of capital letters and tiny.

In capital letters:
- names of the operators, the procedures
- names of the keywords and the keywords factors,
- imposed arguments of standard text (those are between ‘quotes’ as in the syntax of the orders).

In small letters:
- names of the produced concepts,
- reference symbols of the arguments,
- types of the produced concepts and the arguments.

Into mixed tiny - capital letter when the produced concept admits under type. This one appears in capital letters as well as type FORTRAN of the size of under type.

One reinforces the legibility of syntax by the use of indentations. They are used with the location of the blocks of operands and the release of a group as operands under a keyword factor. Are also used they to lay out the brackets of the same block under the same balance.

Example:

```
my [matr_asse_*] = ASSE_MATRICE

     (  ♦  MATR_ELEM =     lmel,  /                  [l_matr_elem_DEPL_R]
                                  /                  [l_matr_elem_DEPL_C]
                                  /                  [l_matr_elem_TEMP_R]
                                  /                  [l_matr_elem_PRES_C]

        ♦  NUME_DDL  =     naked,                        [nume_ddl]

        ◊  CHAR_CINE =     lcha,  /                  [l_char_cine_meca]
                                   /                 [l_char_cine_ther]
                                  /                  [l_char_cine_acou]

        ◊  INFORMATION =                 /  1,              [DEFECT]
                                  /  2,

   );


if MATR_ELEM     [matr_elem_DEPL_R]         then      [*]          DEPL_R
                 [matr_elem_DEPL_C]                                DEPL_C
                 [matr_elem_TEMP_R]                                TEMP_R
                 [matr_elem_PRES_C]                                PRES_C
```