To introduce a new quantity (or component)

Summarized:

This document describes what it is necessary to do to introduce a new quantity into *Code_Aster* or a new component in an existing quantity.

In a few words, to add a quantity, it is necessary:

- to modify the catalog describing the quantities,
- to add the name of the quantity in a catalog of commands.

To introduce a component into an existing quantity, it is enough:

- to enrich the catalog of quantities.

Contents

1 For

Code_Aster, a quantity is made up of an identifier (its name), of a type and a list of components. Example

of quantity: DEPL_R

- : name of the quantity of real displacements to the nodes. One associates with this quantity the real type R and the component DX , DY , DZ , DRX , DRY , DRZ , …. In

Code_Aster , the elementary simple quantities and the are distinguished quantities. The elementary quantities are attached to the vectors or matrixes elementary. They are built starting from the simple quantities. All these quantities (simple and elementary) are described in the file (badly named!) quantity _simple__.catastrophes. The description of the elementary quantities is made with [§4]8

will try to answer the questions: What

- is necessary to make to add a new quantity? What
- is necessary to make to add a new component in an existing quantity? Which
- catalogs is necessary it to modify?

The introduction of a new quantity requires two actions:

- modification of the catalog of quantities: quantity _simple.cata, modification
- of the catalog of the commands: c_nom _grandeur.capy We

will detail these two actions successively. Modification

2 of the catalog of quantities “quantity _simple__.catastrophes” One has

in this paragraph structure of the catalog quantity _simple__.catastrophes, and one describes how one proceeds to add a quantity or a component. Description

2.1 of the catalog the catalog

quantity _simple.cata is present in the directory compelem directory catalo. We present below an extract of this catalog: GRANDEUR_SIMPLE

```
__<< Standard
ABSC_R  : R Curvilinear abscisse   along a telegraphic mesh ABSC:
curvilinear abscisse
        ABSC1: curvilinear abscisse
       of the 1st node of a SEG2 ABSC2: curvilinear abscisse
       of the 2nd node of SEG2 >> ABSC_R
= R
       ABSC   ABSC1    ABSC     2 << Standard
STAOUDYN  : R Parameters of Newmark if dynamic computation STAOUDYN =
       0: static = 1: dynamics
                ALFNMK: parameter
       of Newmark ALPHA DELNMK: parameter
       of Newmark DELTA >> STAOUDYN
=
    R STAOUDYN ALFNMK    DELNMK the block   defining
```

a quantity arises by: A zone delimited

- by “<<” and “> >” intended for the comments. One described there in some lines the quantity which one defines, its type, and its components. For example, for the first block delimited by “<<” and “> >”, the quantity of name ABSC_R of the type R represents the curvilinear abscisse along a telegraphic mesh; it is made up of three components ABSC, ABSC1 , ABSC2 respectively characterizing the curvilinear abscisse, the curvilinear abscisse of the first node of the SEG2 and the curvilinear abscisse of the second node of the SEG2. One line (

- or several so the number of components is consequent) defining the quantity, its components, and type FORTRAN of its components (R, C, F, I, K8, …). This line presents itself in the form: nom_de_grandor

= standard composante_1 composante_2… composante_n Note::

For each

quantity, there is only one possible zone of comments (enters << and >>). This zone must be placed before the description of the quantity. It is logically associated and will be printed for him in certain error messages to help the user. When these comments are written, it is initially to the user that it is necessary to think. Addition of

2.2 a simple quantity the eager

developer to add a simple quantity will have to define a name of quantity, to allot a type, and to associate of the components with its quantity. Choice of the type

2.2.1 the type

to be associated with the quantity is type FORTRAN of the values of its components. The developer will have to choose a type among the choices below: R: reality,

- I : integer
- , C : complex
- , K8: chain
- of 8 characters, K16: chain
- of 16 characters, K24: chain
- of 24 characters,… Name of
- the quantity

2.2.2 the first

thing to be made (after having to determine a type) is to define a name of quantity which is sufficiently explicit and nondefined in this catalog. How to define

a name of quantity? The names

are represented by a character string of with more the 8 characters. In the preceding example, the two definite quantities have as a name: ABSC_R and STAOUDYN . Note:

It is advised

> *to introduce the type of the quantity into its name (for example ABSC_R), because it is simpler for the developer to handle a quantity whose type is implicitly known for him. Usually, one makes it precede by the character "_" (underscore) in order to dissociate it from the symbolic name of the quantity. Names of the components*

2.2.3 the second

thing to be made is to reflect on the components. One must put the following questions: Do I need

- to name each component explicitly? Does one know
- by advance the number of components? First case

(simplest): one knows precisely the number of components and one wishes to name each one of them. For that, one defines for each component a name of in more the 8 characters. It is the case of the following example: << Standard

```
ABSC_R   :  R  Curvilinear  abscisse     along  a  telegraphic  mesh  ABSC:
curvilinear abscisse
        ABSC1: curvilinear abscisse
       of the 1st node of a SEG2 ABSC2: curvilinear abscisse
       of the 2nd node of SEG2 >> ABSC_R
= R
       ABSC   ABSC1    ABSC     2 Second    cases
```

: one wishes to define a quantity of which the number of components is consequent and the name of each one of them imports little. With this intention, there exists in Code_Aster of the quantities for this kind of situation. The quantities NEUT_X (where X the type of the quantity represents). Example: <<

NEUT_R

```
Standard  : R "neutral" Quantity of real type X (1): component
       1 X (2): component
       2 X (3): component
       3 X (4): component
```

```
        4 X (5): component
        5… >> NEUT_R

= R
    X [90]   This example
```

describes a "neutral" quantity of type R which can collect with more the 90 components. Note:

One informs

in grandeur_simple __.catastrophes the maximum number of components which a quantity can lay out. The number of components of a quantity necessary to elementary computation is defined in each catalog of element. The use of quantities "NEUT_X" makes it possible to avoid the multiplication of the quantities. A third

case can arise : There exists in Code_Aster a "special" quantity which can have an undetermined number of components. It is quantity VARI_R. One **does not distinguish** the components from this quantity. The component count is given apart from the catalogs. One associates with the quantity a conventional name of component, name VARI. Example: VARI_R

= R

```
    VARI Note:
```

Component

VARI is not exploitable . The components of a field of variables VARI_R are named V1, V2,... quantity VARI_R is often used for the constitutive laws nonlinear. There exists also quantity VAR2_R similar to VARI_R. The developer is invited to consult the catalog grandeur_simple __.catastrophes for more information on this quantity. Addition of

2.3 a component in an existing quantity. It is enough

of the components to enrich the list by the quantity by a new name. Obviously, you will have to comment on the component. Example: Let us suppose

that
one wishes to enrich quantity TEMP_F by a component describing the parameter of Lagrange due to the dualisation by the limiting conditions. Description

of current quantity TEMP_F : << TEMP_ F

```
Standard  : K8 unknown Temperature of thermal phenomenon TEMP: temperature
           TEMP_MIL:
       temperature on Average average (shells) TEMP_INF  :
       temperature on lower face (shells) TEMP_SUP:
       temperature on the upper face (shells) >> TEMP_F
= K8
    TEMP   TEMP_MIL   TEMP_INF TEMP_SUP the various
```

stages are: to establish

- a name for this new component: choice LAGR appears sufficiently significant. to introduce
- a comment describing the new component. After the realization

of these two stages, one leads to: << Standard

```
TEMP_F  : K8 unknown Temperature of thermal phenomenon TEMP: temperature
       TEMP_MIL:
       temperature on Average average (shells) TEMP_INF  :
       temperature on lower face (shells) TEMP_SUP:
       temperature on the upper face (shells) LAGR: parameter
       of Lagrange of has the dualisation of the boundary conditions
       >> TEMP_F
= K8
    TEMP   TEMP_MIL   TEMP_INF TEMP_SUP LAGR From now on,
```

component LAGR of the K8 type describing the parameter of Lagrange is associated with quantity TEMP_F. Modification

3 of the catalog “c_nom_grandeur.capy” a second phase

not to be forgotten during the introduction of a new simple quantity into Code_Aster, is the modification of the catalog c_nom_grandeur.capy . This catalog is

present in the common directory of the directory catapy . It counts all

the simple quantities present in Code_Aster. Extract: def C_NOM

_GRANDEUR

```
    (): is return ("ABSC_R", "ADRSJEVE
    ", "ADRSJEVN
    ", "CAARPO
    ", "CACABL
    ", "CACOQU
    ", "CADISA
    ", "CADISK
    ",…) A
    what useful
    it
    ?
```

It is used by

the supervisory python to check the seizure of the users in the command files. For example, during the creation of a field (operator CREA_CHAMP), it is necessary to provide TYPE_CHAM to the key word a character string describing the type of field to be built (localization and quantity). If TYPE_CHAM = “ELNO_SIEF_R ” (real stress field with the nodes by element), Code_Aster exploits the catalog c_nom_grandeur.capy in order to check if the character string “SIEF_R” seized by the user corresponds to a quantity. “Elementary”

4 quantities Until now, we

spoke only about the “simple” quantities. A “simple” quantity is a list of named components. The fields at nodes

(cham_no_xxx), the cards (carte_xxx) and the fields by elements (cham_elem_xxx) all are associated with a “simple” quantity. For example, a field at nodes of displacement is associated with quantity DEPL_R. On each node of the mesh, this field can “carry” one (or several) component of quantity DEPL_R. The “ elementary”

quantities are those which are associated with the data structures resuelem (elementary vectors or matrixes). A resuelem is

a “ field” containing 1 elementary vector (or 1 elementary matrix) per mesh. The quantity associated with this field is an elementary quantity. Syntax the description

4.1 of the elementary

quantities in the file grandeur_simple__.catastrophes is made behind key word GRANDEUR_ELEMENTAIRE . Examples: ELEMENTARY

GRANDEUR_

```
__… VDEP_R 1 DEPL_R
   …
   MDEP_R 2 DEPL_R
   DEPL_R
   ms MDNS_R 2   DEPL_R   DEPL_R
   MR.… Quantity   "   vector
```

4.1.1 elementary” a quantity of type

“elementary vector” (for example VDEP_R above) is simply described by the number “1” (vector - > dimension = 1) and the simple quantity associated with the elementary quantity (here DEPL_R). Quantity “ stamps elementary

4.1.2 ” a quantity of the type

“stamps elementary” (for example MDEP_R above) is described by the number “2” (stamps - > dimension = 2) and the 2 simple quantities (line and column) associated with the elementary quantity (here DEPL_R and DEPL_R). Note: syntax

makes it possible to define an elementary matrix with 2 different simple quantities (for example: DEPL_R, TEMP_R) but this possibility is never used in the code. For the quantities “

stamps”, one supplements this description by one of the 2 reserved words: Ms (symmetric matrix) or MR. (nonsymmetrical matrix). Conventions of storage

4.2 the scalars (in general

of realities) which make it possible to represent an elementary quantity are numerous: for example, the stiffness matrix (symmetric) of a machine element MECA_HEXA20 contains (60*61/2) real . It is not question

to name all these realities (like the components of a simple quantity). Conventions are

necessary concerning the storage of these scalars. For an elementary vector

(intended to be assembled to give a second member), one seeks to store something which resembles a field “with the nodes” for each element. The convention of storage is natural. For example: N1 N2 N3… DX DY DZ

DX			DY			DZ			DX
DY	DZ	…	1	2	3 4	5	6 7	8	9…
	For				an elementary				matrix

, two conventions are necessary: For a symmetric matrix

, one stores in the following order: 1 2 3 4 5 6 7 8 9 10

11				
	12			
13		14		
15		For		
	a non symmetrical matrix			

, one stores in the following order: 1 2 3 4 5 6 7 8 9 10

11		12	13
	14	15	
16	Knowing		
that		the order	

of the lines and the columns is the same one as that of an elementary vector.