To introduce a new elementary computation option

Abstract:

This document describes what it is necessary to do to introduce a new elementary computation option into *Code_Aster* .

Contents

1 Introduction

For Code_Aster, an elementary computation option correspond to a computation resulting in producing one or more fields “by elements”

Examples of computation options (option):

- RIGI_MECA : computation of the stiffness (elastic behavior)
- FLUX_ELGA : computation of heat flux knowing the temperature with nodes
- CHAR_MECA_PESA_R : computation of the second member related to a loading of gravity.

The 3 preceding examples show that elementary computations can intervene a little everywhere in Code_Aster: in the commands of computation (matrixes and second member), as in the commands of postprocessing (computation of flux in thermal, the mechanical energies,…)

An elementary computation option has a name (K16) and is described in a catalog arranged in the directory catalo/options

To do an elementary computation corresponds to ask for the computation of an option on a list of finite elements (ligrel). This computation is carried out by the call to the routine “hat” of all elementary computations: calcul.f. The computation

is known as “elementary” because in fact the finite elements of the ligrel calculate. So that a computation is “elementary”, it is necessary that it is “local”, i.e. that a finite element can calculate its participation by being aware only of local data (local fields on itself).
Knowing only local quantities, it calculates one logically result “local” (the participation of the finite element).
The collection of the results calculated by the various elements constitutes a field (total) “by elements”. It is a discontinuous field by nature between the elements. Note:

the computation of an elementary computation option can produce several fields. It is for example the case of option RAPH _MECA used in operator STAT_NON_LINE : one calculates at the same time a stress field and a field of local variables.

The global field produced by an elementary computation perhaps a cham _elem or a resuelem . The difference between the two types in data structure is small: the resuelem is an elementary field containing of the matrixes or elementary vectors intended to be assembled to form a total matrix and a second member. When

an elementary computation produces a cham _elem, this one perhaps ELNO , ELGA or ELEM . A field ELNO contains values on the nodes of the elements. A field ELEM is a constant field by mesh. A field ELGA is a field containing of the values on the points of integration (or Gauss) of the elements. In

the following paragraph, we will detail the syntax which the catalog of an option must have. To write the catalog of a new option is not an end in itself. It is only one preliminary stage so that of the finite elements can calculate this option. This is explained in the document [D5.02.05] “To introduce a new elementary computation”. Creation

2 of the catalog option To find

2.1 a name for this catalog

all the catalogs of option of Code_Aster is in the directory options `of` the directory catalo . The purpose is to design a catalog relative to this new option and to store it in the directory options . `By`

convention, the name of the catalog of option is that of the option. The name of an option has with more the 16 characters. Thus, the catalog of option "FLUX_ELGA" `will be` flux `_elga.cata. Structure`

2.2 of the catalog the structure

of the catalog is the following one: %& MODIFICATION

```
OPTIONS….  Nom_de

_l_option_en_majuscule << line
 (S) of comment describing option>> OPTION
__IN
   parameter
_in_1 quantity    _du_champ_associé_au_paramètre_in_1 << line
 (S) of comment describing the parameter 1>> parameter
_in_2 quantity    _du_champ_associé_au_paramètre_in_2 << line
 (S) of comment describing the parameter 2>>…
parameter
_in_n quantity    _du_champ_associé_au_paramètre_in_n << line
 (S) of comment describing the parameter n>> OUT
   parameter
_out_1 grandeur_du_champ_associé_au_paramètre_out_1 localization  << line
 (S) of comment describing parameter 1 of sortie>> parameter
_out_2 grandeur_du_champ_associé_au_paramètre_out_2 localization  << line
 (S) of comment describing parameter 2 of sortie>>…
the first
```

line of a catalog of option is traditional "card MODIFICATION " `of` the sources of Aster. Syntax is "%& MODIFICATION `OPTIONS…". When`
a new option is introduced, it is necessary to write: "%
& AJOUT `OPTIONS" One` line

writes with the second the name of the option in capital letter (FLUX_ELGA`), followed` at line following of a comment between "<<" `and "` >>" to describe the functionality of the computation option explicitly. For example

, for the option FLUX_ELGA , `one can` write: << FLUX_ELGA

```
    : CALCUL HEAT FLUX WITH Gauss points >> Ensuite
```

, one lists all the couples (parameter of entry, quantity) useful for elementary computation. There still, one adds a comment to describe each parameter. For example

, one of the parameters useful for computation, is parameter PGEOMER `associated` inter alia with the coordinates with the nodes. One can write for example: PGEOMER

```
   GEOM_  R << PGEOMER
```

```
        : COORDINATED NODES >> Once
```

all the defined parameters of entry, one describes the parameters of output in the same way, with the difference, that a localization should be added: ELEM_

- _: for a constant field by element, ELGA_
- _: for a field with Gauss points of the element, ELNO_
- _: for a field at nodes by element. RESL_
- _: for a field of the resuelem type (matrix or vector). Small

shortened example: %& MODIFICATION

```
OPTIONS… RIGI_MECA
OPTION
__IN
   PGEOMER
        GEOM_  R << Geometry    of nodes >> PMATERC
        ADRSJEVE  << Material  coded >> PCACOQU
        CACOQU  << Characteristic    of the shell elements >>… OUT
        __
   PMATUUR
        MDEP_  R RESL_  _<< elastic     stiffness matrix >>…
        Some
```

note: The fields

parameters of the option (IN or OUT) all are defined by a couple (parameter, quantity). The quantity is often sufficient to characterize the field. For example, when one speaks about the field of GEOM_ R (geometry of the nodes), everyone understands what it acts. But it happens that an option requires several fields of the same quantity. To think for example of an option which would have like inputted fields 2 fields of displacements (quantity DEPL_R): one corresponding to displacement at the beginning of time step and the other corresponding to a displacement increment. To distinguish these various fields, one associates a “small name” with each field: it is the parameter. For our example, one would write: PDEPLM

```
   DEPL_R << Displacement “-” (beginning of time step) >> PDEPLD
   DEPL_R << Displacement increment >> When
```

there is no ambiguity (only one field for a given quantity), it is of use to name the parameter associated with this quantity by adding one “P” in front of the name of the quantity. For example: PCACOQU

```
CACOQU  <<… >> the catalog
```

of an option is used by all the finite elements which calculate this option. The list of the fields “in” must thus be a list “wraps” fields required by the elements. For example, the shell elements in general need geometrical information (thickness for example) which is not in the field of geometry (coordinated nodes of the mesh). These missing geometrical data are in a field of data structure CARA_ELEM (of the quantity cacoqu). One will thus find often in the catalog of an option the block: PCACOQU

```
CACOQU << Caractéristiques of the shell elements. Comes
                  from the sd cara_elem. >> As
```

the catalog of an option is a “envelope” of the needs for the elements, the list of the fields parameters of an option can be long. This is why it is very important to comment on each parameter field between <<… >>. One imposes

that the person in charge of an option chooses for each output field a “localization” (ELGA_ _, ELNO _ _,…). The goal of this stress is to force a certain homogeneity for the various finite elements which calculate this option. That also makes it possible to be able “to typify ” the total `cham_ elem` produced. A cham_ `elem will be` thus always either ELGA , `or` ELNO `or` ELEM .