To introduce a new option of elementary calculation

Summary:

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

Contents

1 Introduction

For Code_Aster, an elementary option of calculation corresponds to a calculation resulting in producing one or more fields “by elements”

Examples of options of calculation (option) :

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

The 3 preceding examples show that elementary calculations can intervene a little everywhere in Code_Aster: in the orders of calculation (matrices and second member), as in the orders of postprocessing (calculation of flows in thermics, the mechanical energies,…)

One option of elementary calculation a name (K16) has and is described in a catalogue arranged in the repertoire catalo/cataelem/Options

To do an elementary calculation corresponds to ask for the calculation of one option on a list of finite elements (ligrel). This calculation is carried out by the call to the routine “hat” of all elementary calculations: calculation. F90

Calculation is known as “elementary” because they are the finite elements of ligrel who calculate. So that a calculation 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 a “local” result logically (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 calculation of an elementary option of calculation can produce several fields. It is for example the case of the option RAPH_MECA used in the operator STAT_NON_LINE : one calculates at the same time a stress field, a field of internal variables and a field of “residues”.

The total field produced by an elementary calculation perhaps one cham_elem or one resuelem. The difference between the two types in structure of data is small: resuelem is an elementary field containing of the matrices or elementary vectors intended to be assembled to form a total matrix or a second member.

When an elementary calculation produces one 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 catalogue of an option must have.
To write the catalogue of a new option is not an end in itself. It is only one preliminary stage so that finite elements can calculate this option. This is explained in the document [D5.02.05] “To introduce a new elementary calculation”.

2 To find a name for this catalogue

The whole of the catalogues of option of *Code_Aster* is in the repertoire `Options` repertoire `catalo/cataelem`. The objective is to design a catalogue relative to this new option and to store it in the repertoire `Options`.

By convention, the name of the catalogue of option is that of the option. The name of an option has with more the 16 characters. Thus, the catalogue of the option 'FLUX_ELGA' will be `flux_elga.py`.

3 Contained catalogue

The catalogue chosen as example is the following (`flux_elga.py`):

```
from cataelem.Tools.base_objects importation InputParameter,
OutputParameter, Option, CondCalcul
importation cataelem.Commons.physical_quantities ace PHY
importation cataelem.Commons.parameters ace SP
importation cataelem.Commons.attributes ace AT

PNBSP_I  = InputParameter (phys=PHY.NBSP_I, container=' CARA! .CANBSP',
comment= """  PNBSP_I:  NUMBER OF """ LAYER)

PVARCPR  = InputParameter (phys=PHY.VARI_R, container=' FLEW!
&&CCPARA.VARI_INT_N',
comment= """  PVARCPR: VARIABLES OF ORDER  """)

PFLUXPG  = OutputParameter (phys=PHY.FLUX_R, type=' ELGA',
comment= """  PFLUXPG: FLOW AT THE POINTS OF GAUSS """)


FLUX_ELGA = Option (
    para_in= (
        SP.PCACOQU,
        SP.PCAMASS,
        SP.PGEOMER,
        SP.PHARMON,
        SP.PMATERC,
           PNBSP_I,
        SP.PTEMPER,
        SP.PTEMPSR,
           PVARCPR,
    ),
    para_out= (
           PFLUXPG,
    ),
    condition= (
      CondCalcul ('+', ((AT.PHENO, 'HT'), (AT.BORD, '0'),)),
    ),
    comment= """  FLUX_ELGA: CALCULATION OF FLOW AT THE POINTS OF GAUSS
""",
)
```

Let us comment on this file:

3.1 Parameters

One starts by describing the parameters of the option which are specific to the option (i.e., those which are not “shared” parameters found in `cataelem.Commons.parameters`).

There are parameters of entry (`InputParameter`) and of the parameters of exit (`OutputParameter`)

For each parameter, it is necessary to indicate:
- its name (what is sign on the left ‘=’)
- size associated with this parameter
- a comment

If it is a parameter of entry, one can also indicate a “localization” of the field which is associated with this parameter. This localization is a character string having a particular format: several “fields” separated by the sign “!”. This localization makes it possible to indicate to the program where to find (by default) the field parameter: in which structure of data, with which sequence number if this structure of data is one `sd_resultat` .

Example of localization: `container=' CARA! .CANBSP'`
What wants to say: the name of the field is obtained by concaténant the chain `'.CANBSP'` in the name of `sd_cara_elem`.

If it is a parameter of exit, it is necessary to add information on the “type” of the field associated with this parameter (`type=' xxx'`). The possible types are:
- `'ELEM'`: for a constant field by element,
- `'ELGA'`: for a field at the points of Gauss of the element,
- `'ELNO'`: for a field with the nodes by element.
- `'RESL'`: for a field of the type `resuelem` (matrix or vector).

3.2 Option

One describes then the option itself (`FLUX_ELGA = Option (...)`).
On the left sign ‘=’, one finds the name of the option (in capital letters).
The description of the option is very simple: one indicates the list of the parameters of entry (`para_in= (...)`) and lists it parameters of exit (`para_out= (...)`).

The parameters appearing in these two lists are is shared parameters (for example: `SP.PMATERC`) maybe of the parameters preset higher in the catalogue of the option (for example: `PFLUXPG`)

3.3 Block `condition` / `CondCalcul`

This block is very important. It makes it possible to define the whole of the elements which are concerned with the option (those which should calculate it).

Let us take the example of the block corresponding to the option `RIGI_MECA_GE` :

```
  condition= (
      CondCalcul ('+', ((AT.PHENO, 'ME'), (AT.BORD, '0'),)),
      CondCalcul ('-', ((AT.PHENO, 'ME'), (AT.DISCRET, 'YES'),)),
    ),
```

It is a question of indicating the whole of **all** the finite elements for which this option has direction (and which thus should know to calculate it). By indicating them all, one expresses, has crontrario, that all the other elements are not concerned with this option (and thus which they should not calculate this option).

One indicates “packages” of finite elements using the attributes defined in the catalogues of elements or the catalogue `phenomenons_modelisations.py`.

In the preceding example, one defines two “packages” of elements:
1) those which have the attribute `PHENO=' ME'` **and** the attribute `BORD=' 0 '`
2) those which have the attribute `PHENO=' ME'` **and** the attribute `DISCRET=' OUI'`

The whole of all the elements having to calculate the option is obtained in “adding” or “withdrawing” the packages defined according to the “sign” of `CondCalcul` (`'+'` or `'-'`).

In the preceding example, the whole of the elements having to calculate the option `RIGI_MECA_GE` is obtained by taking all the elements of the phenomenon `MECHANICS` , which is “principal” (`BORD=' 0 '`) and by withdrawing the discrete elements.

The number of “lines” `CondCalcul` is not limited. The order of these lines important: one draws up the list of the elements concerned with the option by additions and removals of packages in the order of the list.

3.4 Comment

One finishes the description of an option by a field comment (argument `comment= '…'`).
This comment can be used, for example, to clarify an error message.
It must explicitly describe the goal of the option of calculation.

For example, for the option `FLUX_ELGA`, one can write:

```
Comment=' FLUX_ELGA: CALCULATION OF THE HEAT FLUX AT THE POINTS OF
GAUSS'
```

3.5 Remarks

With each field parameter of the option corresponds a couple (name of the parameter, size). The size is often sufficient to characterize the field. For example, when one speaks about the field about `GEOM_R` (geometry of the nodes), everyone understands what it acts.

But it happens that an option requires several fields of the same size. To think for example of an option which would have like inlet limits 2 fields of displacements (size `DEPL_R`) : one corresponding to displacement at the beginning of the step of time and the other corresponding to an increment of displacement.
The parameters are used to distinguish these various fields. One associates a “small name” with each field: it is the name of the parameter.

For our example, one would write:

```
PDEPLMR = InputParameter (phys=PHY.DEPL_R,
                          comment= """ PREVIOUS URGENT DISPLACEMENT """)
PDEPLPR = InputParameter (phys=PHY.DEPL_R,
                          comment= """ INCREMENT OF DISPLACEMENT. """)
```

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

```
PCACOQU = InputParameter (phys=PHY.CACOQU,…)
```

The catalogue of an option is used by all the finite elements which calculate this option. The list of the fields parameters must thus be a list "wraps" fields used by the elements.

For example, the elements of hull in general need geometrical information (thickness,…) who are not in the field of geometry (coordinated nodes of the grid). These missing geometrical data are in a field of the structure of data CARA_ELEM (of the size cacoqu). One will thus find often in the catalogue of an option the block:

```
PCACOQU = InputParameter (phys=PHY.CACOQU,…)
```

As the catalogue 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 important to comment on all the fields parameters.

One imposes that the person in charge of an option chooses for each field of exit a "type" (ELGA , ELNO ,…). The goal of this constraint 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" it cham_elem total produced. One cham_elem will be thus always is ELGA , that is to say ELNO that is to say ELEM .