To introduce a new modeling into AFFE_MODELE

Summary:

This document describes what it is necessary to do to introduce a new modeling into the operator AFFE_MODELE of *Code_Aster* .

In a few words, it is necessary:

- To add modeling to the catalogue of the order AFFE_MODELE,
- To introduce a small block of text into the catalogue of the phenomena and modelings,
- To write one or more catalogues of elements.
- To write the routines of elementary calculations specific to the elements of this new modeling.

The object of this document is mainly to present the catalogue Commons/phenomenons_modelisations.py and the general structure of a catalogue of Elements/xxx.py.

The rest of the actions to be made is described in the document [D5.02.05] "To introduce a new elementary calculation".

Contents

1 Introduction

The choice of modeling is carried out through the operator AFFE_MODELE of *Code_Aster* . For example, the user will write in his command file:

```
KID = AFFE_MODELE (GRID = E-MAIL,
                       AFFE=_F (ALL = 'YES', MODELING = 'AXIS_JOINT_HMS',
                                PHENOMENON = 'MECHANICAL'))
```

In the objective to be able to propose to the user of other modelings, one will describe in this document a methodology to introduce a new modeling into *Code_Aster* .

To introduce a modeling into *Code_Aster* require to put the following questions:

- In which phenomenon I will add my modeling?
- Which are geometrical and topological dimensions finite elements?
- Which are the meshs concerned with this modeling?
- Which are the "principal" elements and the elements of edge?
- Which are the attributes which one can define?
- Which are realizable calculations with this modeling?

We will answer these questions in this document.

Other relative questions with the finite elements are treated in other documents:

- [D5.02.01] How to introduce a new size or new components (CMP) in an existing size?
- [D5.02.02] How to introduce a new type of mesh (type_maille) or a new element of reference (ELREFE) ?
- [D5.02.05] How to introduce a new elementary calculation?

2 Addition of modeling to the catalogue of order

- Catalogue to modify: code_aster/Catastrophes/Commands/affe_modele.py

New modeling is to be added to the whole of modelings usable by the order AFFE_MODELE.
It is enough to add it to the list of possible (field into).

3 Modification of the catalogue of the phenomena and modelings

- Catalogue to modify: Commons/phenomenons_modelisations.py

3.1 Presentation of the catalogue phenomenons_modelisations.py

This catalogue breaks up into several parts: a part dedicated to each phenomenon: mechanics, thermics, acoustics,…

For each phenomenon, there exists a block corresponding to each modeling. For example, for modeling 'AXIS_SI'phenomenon'MECHANICS', we have:

```
 phen.add ('AXIS_SI', Modeling (dim= (2.2), code=' AXS',
```

```
attrs= (
     (AT.AXIS, 'YES'),
     (AT.NBSIGM, '4'),
     (AT.TYPMOD, 'AXIS'),
),
elements= (
     (MT.QUAD8      , EL.MEAXQS8),
     (MT.SEG3       , EL.MEAXSE3),
)))
```

We will present the arguments of this block:

- Modeling
- dim
- code
- attrs
- elements

3.1.1 Arguments: modeling, dim and code :

- The argument modeling provides the name of modeling. In the example above, it is modeling 'AXIS_SI'.

- The argument dim (2 entireties) respectively provides topological dimension and the geometrical dimension of modeling:
  - Geometrical dimension corresponds to dimension of ambient space (3 or 2),
  - topological dimension corresponds corresponding to "principal" elements of modeling.

  In this example, topological dimension is identical to geometrical dimension, but it is not always the case.

  For example, a modeling 'DKT' (thin hulls) collects elements whose meshs support are of dimension 2 (triangles, quadrangles), however the nodes of the grid are expressed in the reference mark 3D (according to X , Y , Z). One thus has for this modeling: dim= (2.3)

- The argument codes provides as its name indicates it, a code. It is about a chain of 3 characters making it possible to identify modeling. For modeling 'AXIS_SI' , the selected code is 'AXS'. This "code" is inevitably different for all modelings. It is one alias (on 3 characters exactly) of the name of modeling.

3.1.2 Argument attrs

The argument attrs allows to define the attributes carried by all the elements of modeling.

Example: attrs= (
(AT.AXIS, 'YES'),…

The list attrs is made of couples (attribute, value)

For what are used they?

They make it possible to provide information in source FORTRAN and to consider treatments according to this information. In the example above, the attribute AT.AXIS=OUI can be questioned in an elementary routine of calculation in order to modify the weight of integration of the points of Gauss.

The attributes are also used to gather elements which are joint: for example, elements of “beam”, “hull”,…
These regroupings are used in the blocks “ CondCalcul ” catalogues of options.

The attributes all are defined in the catalogue Commons/attributes.py
It is very important to document these attributes well so that their direction is the least ambiguous possible.

Note:

The various attributes defined in this catalogue are assigned to the whole of type_element modeling. If it is wanted that an attribute is associated only with one type_element, it is then necessary to define this attribute in the catalogue of Type_element.

Caution: When a finite element is used by several modelings (it is often the case of the elements of edge), there can be ambiguity on the value of the attributes of this finite element. The element will inherit ALL the attributes defined in the whole of modelings which use it. If the same attribute is defined several times with different values, one affects the value to him “ ### ”. It is always the case of the attribute MODELI.

How to recover the value of an attribute in source FORTRAN?

Routines lteatt. F90 and teattr. F90 give access the attributes of one type_element.

3.1.3 Argument elements

Example:
```
elements= (
         (MT.QUAD8      , EL.MEAXQS8),
         (MT.SEG3       , EL.MEAXSE3),
    )
```

This argument is used to define all the elements of modeling (and of its edge). It is a list of couples (type_maille, type_element).

In our example, the couple: (MT.QUAD8 , EL.MEAXQS8),

mean that one allots for this modeling, the element of the type MEAXQS8 with the quadrangular meshs with 8 nodes of the type QUAD8.

3.2 Introduction of a new modeling into the catalogue phenomenons_modelisations.py

First of all, it is necessary to be placed in the part corresponding to the phenomenon of your modeling (MECHANICS , THERMIQEU or ACOUSTIQUE). Then place with the writing of the block corresponding to your modeling.

You must start with:

- to choose a name for your modeling (with more the 16 characters),
- to allot a code to your modeling (3 characters exactly),
- to determine “dimensions” of your modeling.

You must then ask you the question of the attributes carried by the elements of the new modeling. It is important to traverse the list of all the existing attributes to know if your elements are concerned with these attributes. For example, it would be an error not to declare `AT.AXIS=' OUI'` for an axisymmetric element.

The following stage consists in choosing:

- types of meshs which you wish to associate with your modeling. You can consult the catalogue `Commons/mesh_types.py` to become acquainted with the types of meshs present in *Code_Aster* as well as their elements of reference ,
- the name of the type of the finite element which you wish to associate with each type of mesh. You must determine a name of with more the 16 characters which is sufficiently explicit to know the type of its mesh to the reading of its name.

We have just answered some questions concerning new modeling.

To go further, it is necessary to write it (or them) catalogues (S) describing the new finite elements of modeling, as well as routines FORTRAN `te00ij.f` carrying out their elementary calculations.

In the following paragraph, we describe (rather briefly) the catalogue of type_element.

For further details on this catalogue and the writing of associated routines FORTRAN, one will refer to the document [D5.02.05] “To introduce a new elementary calculation”

4 Creation of the catalogues of elements

- The catalogues of elements are localised in … /catalo/cataelem/Elements
- A catalogue makes it possible to describe a family of “close” elements (for example various elements of the same modeling). The catalogue is called after the first element which there figure.

4.1 Summary presentation of a catalogue of elements

We will present the broad outlines. For more information, the reader is invited to consult documentation D5.02.05 (“ *to introduce an elementary calculation* ”).

We will present an extract of the catalogue ther_hexa20.py . We will propose the option of calculation FLUX_ELGA who allows to calculate the heat flux at the points of Gauss of the element starting from the field of temperature.

```
…

MVECTTR  = ArrayOfComponents (phys=PHY.VTEM_R, locatedComponents= (DDL_THER

MMATTTR  = ArrayOfComponents (phys=PHY.MTEM_R, locatedComponents= (DDL_THER
DDL_THER))


#------------------------------------------------------------
class THER_HEXA20 (Element):
    meshType = MT.HEXA20
    elrefe = (
             ElrefeLoc (MT.H20, gauss = ('RIGI=FPG27',…,),…),
         )
    calculations = (
        …

        OP.FLUX_ELGA (te=62,
            para_in= ((SP.PCAMASS, CCAMASS), (SP.PGEOMER, NGEOMER),
                      (SP.PMATERC, LC.CMATERC), (SP.PTEMPER, DDL_THER),
                      (SP.PTEMPSR, CTEMPSR), (OP.FLUX_ELGA.PVARCPR, LC.ZVARC
                     ),
            para_out= ((OP.FLUX_ELGA.PFLUXPG, EFLUXPG),),
        ),
        …
```

Text 1: Extracted the catalogue ther_hexa20.py

- One starts by describing the local modes specific to the elements described in this file (MVECTTR, MMATTR,…)

- All the elements described in this file, are classes python which inherit the class Element . THER_HEXA20 being the first element of the file, it is him who gives the name of the file.
  Other elements of the file (THER_PENTA15, THER_TETRA10 ,…) are obtained by heritage of THER_HEXA20 .

- First lines:

```
class THER_HEXA20 (Element):
       meshType = MT.HEXA20
       elrefe = (
            ElrefeLoc (MT.H20, gauss = ('RIGI=FPG27',…,),…),
       )
```

Allow to declare the type of mesh associated with the element (`MT.HEXA20`) and its (or its) elements of reference (`elrefe = (…)`)

- Then the list comes from all the elementary calculations carried out by the element (argument `calculations)`. Elements of the tuple `calculations` are elementary calculations.
  For example for `FLUX_ELGA` :

```
 OP.FLUX_ELGA (te=62,
     para_in= ((SP.PCAMASS, CCAMASS), (SP.PGEOMER, NGEOMER),
               …
               ),
     para_out= ((OP.FLUX_ELGA.PFLUXPG, EFLUXPG),),
),
```

  The number of the routine is indicated `te00ij` (here `te0062. F90`) associated with elementary calculation, as well as the description of and the exit inlet limits of elementary calculation. The description of a field is made by associating (in a couple) the parameter of the option and the local mode which is associated for him.