

To introduce a new elementary calculation

Summary:

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

In a few words, it is necessary:

- to introduce a small block of text into the catalogue of one `type_element`
- to write a new routine FORTRAN of name `te00ij.f` where `00ij` is a number with 4 digits

Contents

1 Introduction.....	3
2 Modification of the catalogue Elements/ther_hexa20.py.....	4
2.1 To find the name of the file to be modified.....	4
2.2 To modify the file ther_hexa20.py.....	5
2.2.1 Field of exit.....	5
2.2.2 Inlet limits.....	6
2.2.2.1 DDL_THER.....	7
2.2.2.2 NGEOMER.....	8
2.2.2.3 CMATERC.....	8
2.2.2.4 CCAMASS.....	8
2.2.2.5 CTEMPSR.....	8
3 To write (or modify) routine FORTRAN te0062. F90.....	9
3.1 Arguments of the routine.....	9
3.2 Presentation of some utilities used in the te0062. F90.....	9
3.2.1 Routine JEVECH.....	9
3.2.2 Routine ELREFE_INFO.....	10
3.2.3 Routine DFDM3D.....	10
3.2.4 Routine RCVALA.....	10
3.3 Routine TE0062.....	11
4 Details not used in the selected example.....	12
4.1 Description of the heading of one type_element.....	12
4.2 Local modes ELNO/ DIFF.....	12
4.3 Conventions of names for the local modes.....	12
4.4 To supplement...Names reserved for certain local modes (ddl_meca, ddl_ther, ddl_acou).....	12
4.5 Local fields of standard elementary vector or elementary matrix.....	13
4.6 Optional fields, routine tecach. F90.....	13
4.7 Family of points of Gauss "MATER".....	13

1 Introduction

For Code_Aster, an elementary calculation corresponds to a couple (standard of finite element, option of calculation). Examples of the types of finite element (`type_element`) :

- `MEDKTR3` : element `DKT` triangular with 3 nodes
- `THER_PENTA15` : element of thermics pentahedron to 15 nodes

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

In the rest of this document, the example which will be used to us as main idea will be that of the calculation of the heat flux at the points of integration (`FLUX_ELGA`) elements `PENTA15` modeling '3D' phenomenon 'THERMICS' (`type_element = THER_PENTA15`).

One will suppose that this elementary calculation still does not exist but only the option `FLUX_ELGA` already exist (for other finite elements) and that it `type_element THER_PENTA15` also exist (it can already calculate other options). In this document, we will try to answer the questions:

- What is necessary to make to carry out this new elementary calculation?
- Which source files is it necessary to modify or add?

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

- [D5.02.03] How to introduce a new option of elementary calculation? (for example a new postprocessing)
- [D5.02.04] How to introduce a new family of finite elements (`modeling`) ?
- [D5.02.01] How to introduce a new size or new components into an existing size?
- [D5.02.02] How to introduce a new type of mesh (`type_maille`) or a new element of reference (`ELREFE`) ?

We already said in the summary that the introduction of a new elementary calculation into Code_Aster required 2 actions:

- the addition of a block of text in the catalogue of `type_element` (here `THER_PENTA15`)
- the addition (or modification) of a routine FORTRAN of name `te00ij. F90`

We will detail these two actions successively.

2 Modification of the catalogue Elements/ther_hexa20.py

2.1 To find the name of the file to be modified

The first difficulty of solving is to find the name of the file catalogues to modify. Let us notice already that the name of `type_element` who concerns us (`THER_PENTA15`) is not inevitably familiar for us. We can discover it by using the order `AFFE_MODELE` on a grid containing of `PENTA15` :

```
MOTH = AFFE_MODELE ( GRID = E-MAIL,  
                     AFFE=_F (ALL = 'YES', MODELING = '3D',  
                               PHENOMENON = 'THERMAL'))
```

In the file `.mess` we can then see:

```
ON          132 MESHES OF THE GRID E-MAIL  
ONE WITH REQUEST THE ASSIGNMENT OF          132  
ONE WITH PU TO AFFECT SOME                  124  
  
MODELING    FINITE ELEMENT    TYPE NETS    NUMBER  
3D          THER_PENTA15      PENTA15    40  
3D          THER_FACE6        TRIA6      4  
3D          THER_FACE8        QUAD8      80
```

We see that modeling called '3D' applied to `PENTA15` led to the assignment of finite elements of type `THER_PENTA15`. It is the name which we seek.

We could also have found this name by consulting the catalogue:

```
... /catalo/cataelem/Commons/phenomenons_modelisations.py  
...  
phen.add ('3D', Modeling (dim= (3.3), code=' 3D_ ',  
                          elements= (  
          (MT.HEXA8      , EL.THER_HEX8),  
          (MT.PENTA6     , EL.THER_PENTA6),  
          (MT.TETRA4     , EL.THER_TETRA4),  
          (MT.PYRAM5     , EL.THER_PYRAM5),  
          (MT.QUAD4      , EL.THER_FACE4),  
          (MT.TRIA3      , EL.THER_FACE3),  
          (MT.HEXA27     , EL.THER_HEX27),  
          (MT.HEXA20     , EL.THER_HEX20),  
          (MT.PENTA15    , EL.THER_PENTA15),
```

This catalogue gives the name of all them `type_element` associated with the various finite elements of modeling '3D'. For the type of mesh `PENTA15`, the name of `type_element` is well `THER_PENTA15`.

Once known the name `THER_PENTA15`, to find the name of the catalogue concerning this finite element, a `grep` should be made in the files of `catalo/cataelem/Elements/*` :

```
grep -wl THER_PENTA15 Elements/*.py
```

What should give: `Elements/ther_hexa20.py`

It is the name of the file which we must modify.

If we publish this file, we see that it contains the description of several elements: `THER_HEX20`, `THER_PENTA15`, ...

The elementary calculations carried out by the finite elements of this file are in general described during the description of the first element (here `THER_HEXA20`). Other elements being obtained by heritage of the first. The first element is, to some extent, a “model” element (for the others).

So that the element `THER_PENTA15` can calculate the option `FLUX_ELGA`, simplest is to add this elementary calculation in the description of the element “models”

What wants to say that our new elementary calculation will be (by default) accessible to all the finite elements described in the file catalogues.

In general, it is what one wishes because if one can do a calculation on one `PENTA15` , one can also do it for `TETRA4` , it `PENTA6` ,...

In the continuation of the document, one will continue to refer to the element `THER_PENTA15` , but actually, our development will relate to all the voluminal elements of modeling ‘3D’ (`THER_TETRA4` , `THER_TETRA10` ,..., `THER_HEXA27`).

If one did not wish to make this calculation elementary available for `PYRAM13` for example, description would be modified `PYRAM13` :

```
class THER_PYRAM13 (THER_HEXA20):
    meshType = MT.PYRAM13
    elrefe = (...)

    calculations = (
        OP.FLUX_ELGA (te=-1),
```

the number “- 1” compared to the option `FLUX_ELGA` indicate that this elementary calculation is not available for `PYRAM13` . A fatal error will be emitted if the user tries to use it.

2.2 To modify the file `ther_hexa20.py`

The block of text to be added in the argument “ `calculations` ” is:

```
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.ZVARCPG),
             ),
    para_out= ((OP.FLUX_ELGA.PFLUXPG, EFLUXPG),),
),
```

In this block of text, one can recognize different items:

- the name of the option which one wants “to carry out” (`FLUX_ELGA`)
- The number “62” who is the number of the routine `te00ij. F90` associated with elementary calculation (here: `te0062. F90`)
- the block of the inlet limits of elementary calculation (argument `para_in`)
- the block of the fields of exit of elementary calculation (argument `para_out`)

Note:

a field is “ `IN` ” is “ `OUT` ”, it cannot be “ `INOUT` ”. Blocks of the fields “ `IN` ” and “ `OUT` ” are described by lists of couples (`parameter, mode_local`).
Here, there are 6 fields “ `IN` ” and only one field “ `OUT` ”.

2.2.1 Field of exit

Let us start with the field “OUT” (drank elementary calculation):

PFLUXPG (the parameter) is the name given to the field of flow result for the option FLUX_ELGA .

This name was selected when the catalogue of the option (Options/FLUX_ELGA.py) was introduced into the code. This name is used in all the catalogues of the finite elements which calculate the option FLUX_ELGA .

EFLUXPG is the local mode associated with the parameter. It makes it possible to describe the structure of the local field of flow for the elements THER_PENTA15 . We will return there later.

Let us look at what the catalogue of the option contains FLUX_ELGA concerning its field of exit:

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

FLUX_ELGA = Option (
    para_in= (
        SP.PCACOQU,
        ...
        PVARCPR,
    ),
    para_out= (
        PFLUXPG,
    ),
)
```

It is seen that the field parameter PFLUXPG is a field of the size FLUX_R and that it is a field by elements at the points of Gauss (type=' ELGA')

It is the “specifications” of the development that we must realize: we must calculate a field of FLUX_R on the points of Gauss of THER_PENTA15 . The choice of the type of field (here ELGA) is imposed by the option. On the other hand components of the size FLUX_R that one will use are with the choice of the element THER_PENTA15 . It is the object of mode_local EFLUXPG .

This mode_local is described in the catalogue of the element:

```
EFLUXPG = LocatedComponents (phys=PHY.FLUX_R, type=' ELGA', location='
RIGI',
                             components= ('FLOW', 'FLUY', 'FLUZ',))
```

It is seen there that this mode_local relates to the size well FLUX_R and that it describes well the structure of a field “ELGA” (it is in the specifications imposed by the option). The choice of the components: FLOW , FLUY and FLUZ nothing astonishing has since the element is 3D .

The name of the components which one can use in the description of a mode_local is given in the catalogue of the sizes (Commons/physical_quantities.py) . One can read there concerning the size FLUX_R :

```
FLUX_R = PhysicalQuantity (type=' R',
                           components= (
                               'FLOW',
                               'FLUY',
                               'FLUZ',
                               'FLUX_INF',
                               ...
                           ))
```

2.2.2 Inlet limits

The list of the inlet limits has the same structure as that the fields of exit: it is a list of couples (parameter, mode_local). The parameters are imposed by the catalogue of the option.

For FLUX_ELGA one finds in the catalogue of the option 9 fields parameters:

```
FLUX_ELGA = Option (  
    para_in= (  
        SP.PCACOQU,  
        SP.PCAMASS,  
        SP.PGEOMER,  
        SP.PHARMON,  
        SP.PMATERC,  
        PNBSP_I,  
        SP.PTEMPER,  
        SP.PTEMPSR,  
        PVARCPR,  
    ),  
,
```

It is necessary to choose in this list the parameters which will be useful for type_element THER_PENTA15.

After reflection, the 5 following parameters are retained:

- PTEMPER : it is the field of temperature of which it the gradient should be calculated
- PGEOMER : the field of geometry of the element (its coordinates) is necessary to calculate the gradient.
- PMATERC : the material field is necessary to be able to know thermal conductivity (average)
- PCAMASS : it is the field containing the possible local reference mark of the elements 3D . It is necessary if the material is not isotropic: the anisotropic characteristics are given in a local reference mark.
- PTEMPSR : this field is used to transmit the moment of calculation. It is necessary here because, in thermics, when the coefficients material are functions (DEFI_MATERIAU / THER_FO for example), these functions can depend on time.

Once one retained the parameters useful for type_element , it is necessary to allot to them to each one one mode_local . One will choose:

```
PTEMPER → DDL_THER  
PGEOMER → NGEOMER  
PCAMASS → CCAMASS  
PMATERC → CMATERC  
PTEMPSR → CTEMPSR
```

These local modes must be described in the catalogue of the shared local modes (Commons/located_components.py) or in the catalogue of the element (ther_hexa20.py).

Let us comment on these local modes:

2.2.2.1 DDL_THER

```
DDL_THER = LocatedComponents (phys=PHY.TEMP_R, type=' ELNO',  
    components= ('TEMP',))
```

This mode_local indicate that the local field expected by the finite element is a field on the nodes of the element (ELNO).

All nodes of the element (15 for one PENTA15) the same components carry.

The list of the components carried by these nodes is reduced to only one component of the size TEMP_R : TEMP .

2.2.2.2 NGEOMER

```
NGEOMER = LocatedComponents (phys=PHY.GEOM_R, type=' ELNO',  
                             components= ('X', 'Y', 'Z',))
```

This `mode_local` indicate that the local field expected by the finite element is a field on the nodes of the element (`ELNO`).

The list of the components carried by these nodes is `X`, `Y`, `Z` because the element is 3D .

2.2.2.3 CMATERC

```
CMATERC = LocatedComponents (phys=PHY.ADRSJEVE, type=' ELEM',  
                             components= ('I1',))
```

This `mode_local` indicate that the local field expected by the finite element is a constant field on the element (`ELEM`).

The list of the components carried by this element is `I1` .

For reasons of performance, the data-processing structure representing material on the level of an element known as "is coded" (one speaks then about coded material), it is represented by an integer (component `I1` size `ADRSJEVE`) who is an address memory. One will see later how this material is used in the utility routines.

2.2.2.4 CCAMASS

```
CCAMASS = LocatedComponents (phys=PHY.CAMASS, type=' ELEM',  
                             components= ('It, 'ALPHA', 'BETA', 'KAPPA', 'X', 'Y', 'Z',))
```

This `mode_local` corresponds to a constant field on the element (`ELEM`).

7 components (`C`, `ALPHA`, ...) are real numbers which make it possible to represent the change of Total reference mark → Local

We will not say any here.

2.2.2.5 CTEMPSR

```
CTEMPSR = LocatedComponents (phys=PHY.INST_R, type=' ELEM',  
                             components= ('INST', 'DELTAT', 'THETA', 'KHI', 'R', 'RHO',))
```

It is noticed that 6 components are indicated in `mode_local` : `INST`, `DELTAT`, ... However only the value of the moment `INST` we is useful to be able to evaluate possible functions of time.

The other components are probably useless and they could be withdrawn from `mode_local` .

However, it should not be done too brutally. Indeed, it `mode_local` `CTEMPSR` is used by other options which they need more than information (`DELTAT` : step value of time,...)

If one wanted to improve the legibility of the catalogue of our element, one could duplicate it `mode_local` :

```
CTEMPSR = LocatedComponents (phys=PHY.INST_R, type=' ELEM',  
                             components= ('INST', 'DELTAT', 'THETA', 'KHI', 'R', 'RHO',))  
CTEMPS1 = LocatedComponents (phys=PHY.INST_R, type=' ELEM',  
                             components= ('INST',))
```

For our option `FLOW_ELGA` , one could then describe the local field of moment by the couple (`PTEMPSR`, `CTEMPS1`)

3 To write (or modify) routine FORTRAN te0062. F90

We already saw that the catalogue of the element THER_PENTA15 indicated that the calculation of the option FLUX_ELGA will happen in routine FORTRAN te0062. F90. The object of this routine is to calculate (for a finite element) the local field of flow on the points of Gauss of the element. In a general way, a routine te00ij. F90 always has like drank to calculate fields "OUT" starting from its fields "IN".

3.1 Arguments of the routine

All routines te00ij. F90 the same arguments have. The reason is that the routine which calls them (te0000. F90) is written once and for all and that one does not want to change it to each time one adds an elementary calculation.

Two only arguments of the routines te00ij. F90 are arguments of entry:

- OPTION is a character string containing the name of the option (for us: FLUX_ELGA)
- NOMTE is a character string containing the name of type_element (for us: THER_PENTA15)

These 2 arguments can be used (or not).

The argument is used OPTION when one treats in the same routine te00ij. F90 2 options (or more) which resemble each other.

One can then write tests like:

```
...  
IF (OPTION.EQ. 'FLUX_ELGA') THEN  
...  
ELSE IF (OPTION.EQ. 'FLUX_ELNO') THEN  
...  
ENDIF
```

In the same way, one in general treats all the elements of the same modeling (TETRA4 , TETRA10 ,...) in the same routine te00ij. F90. The argument NOMTE allows to distinguish certain treatments. One could for example imagine a small block of code specific to the pyramidal elements.

But true arguments of the routines te00ij. F90 are actually their fields "IN" and their fields "OUT". These arguments are "underground" and one reaches it via the 2 utility routines JEVECH and TECACH that one will present in the following paragraph.

3.2 Presentation of some utilities used in the te0062. F90

3.2.1 Routine JEVECH

This routine makes it possible to recover the address memory of a local field. For example, in the routine te0062. F90, one finds:

```
CAL JEVECH ('PGEOMER', 'IT, IGEOM)
```

The 1st argument of the routine JEVECH is the name of parameter who interests us.

The 2nd argument is "documentary" (it is not used in the code). It indicates if the parameter is a field "IN" (access in reading 'It) or an "OUT" field (access in writing 'E').

The 3rd argument (of exit) is the address of the zone memory containing the local field.

It is to the programmer of the routine te0062. F90 to ensure coherence between the use that it makes of this address and what is written in the catalogue of the element concerning this parameter:

- Since it parameter `PGEOMER` is associated with the size `GEOM_R` and that this size is of real type, the address `IGEOM` is an address in the common `JEVEUX ZR`
- Since it `mode_local` `NGEOMER` (associate with `PGEOMER`) is described in the catalogue like a field with the nodes having 4 components (X THERE Z) by `Nœud`, it is to the programmer of knowing that the field room represented in memory with the address `ZR (IGEOM)` is length $3*15$ (3 components X, Y, Z for each of the 15 nodes).

More precisely, the programmer must know that it will find in the memory `JEVEUX` :

```
ZR (IGEOM-1+1) - > 'X' of node 1  
ZR (IGEOM-1+2) - > 'Y' of node 1  
ZR (IGEOM-1+3) - > 'Z' of node 1  
ZR (IGEOM-1+4) - > 'X' of node 2  
ZR (IGEOM-1+5) - > 'X' of node 2  
...  
ZR (IGEOM-1+45) - > 'Z' of node 15
```

3.2.2 Routine `ELREFE_INFO`

To supplement...

3.2.3 Routine `DFDM3D`

To supplement...

3.2.4 Routine `RCVALA`

To supplement...

3.3 Routine TE0062

```
SUBROUTINE TE0062 (OPTION, NOMTE)
IMPLICIT NUN
CHARACTER*16 OPTION, NOMTE

C --- BEGINNING DECLARATIONS NORMALISEES JEVEUX -----
INTEGER ZI
COMMON /IVARJE/ZI (1)
REAL*8 ZR
COMMON /RVARJE/ZR (1)
COMPLEX*16 ZC
COMMON /CVARJE/ZC (1)
LOGICAL ZL
COMMON /LVARJE/ZL (1)
CHARACTER*8 ZK8
CHARACTER*16 ZK16
CHARACTER*24 ZK24
CHARACTER*32 ZK32
CHARACTER*80 ZK80
COMMON /KVARJE/ZK8 (1), ZK 16(1), ZK 24(1), ZK 32(1), ZK 80(1)
C --- FINE DECLARATIONS NORMALISEES JEVEUX -----

INTEGER ICODRE
CHARACTER*8 NOMRES (1)
AVERAGE REAL*8, FLUXX, FLUXY, FLUXZ, DFDX (27), DFDY (27), DFDZ (27),
WEIGHT
INTEGER JGANO, IPOIDS, IVF, IDFDE, IGEOM, IMATE, NO, KP, NPG1, I,
IFLUX,
&          ITEMPS, ITEMPE, NDIM, NOS
C-----

C      -- recovery of information concerning the element of reference:
CAL ELREFE_INFO (FAMI=' RIGI', NPG=NPG1,
&                JPOIDS=IPOIDS, JVF=IVF, JDFDE=IDFDE)

C      -- recovery of the addresses of the local fields:
CAL JEVECH ('PGEOMER', 'IT, IGEOM)
CAL JEVECH ('PMATERC', 'IT, IMATE)
CAL JEVECH ('PTEMPSR', 'IT, ITEMPS)
CAL JEVECH ('PTEMPER', 'IT, ITEMPE)
CAL JEVECH ('PFLUX_R', 'E', IFLUX)

C      -- recovery of AVERAGE conductivity:
CAL RCVALA (ZI (IMATE), '', 'THER', 1, 'INST', ZR (ITEMPS), 1,
'LAMBDA',
&          LAMBDA, ICODRE, 1)

C      -- buckle on the points of Gauss:
C 20 KP=1, NPG1
C      -- recovery of the derivative of the functions of form
C      (on the real element):
CAL DFDM3D (NO, KP, IPOIDS, IDFDE, ZR (IGEOM), DFDX, DFDY, DFDZ,
WEIGHT)

C      -- calculation of the gradient of the field of temperature:
FLUXX=0.0D0
```

```
FLUXY=0.0D0
FLUXZ=0.0D0

C 10 I=1, NO
  FLUXX=FLUXX+ZR (ITEMPE-1+I) *DFDX (I)
  FLUXY=FLUXY+ZR (ITEMPE-1+I) *DFDY (I)
  FLUXZ=FLUXZ+ZR (ITEMPE-1+I) *DFDZ (I)
10 CONTINUOUS

C  -- calculation of flow and storage in the field result:
ZR (IFLUX+ (KP-1) *3)  =-LAMBDA*FLUXX
ZR (IFLUX+ (KP-1) *3+1) =-LAMBDA*FLUXY
ZR (IFLUX+ (KP-1) *3+2) =-LAMBDA*FLUXZ
20 CONTINUOUS

END
```

Comments on this routine:

It should not be forgotten that this routine must be able to treat all the elements 3D : TETRA4 ,..., HEXA27 . This is why, tables DFDX , DFDY and DFDZ are dimensioned to 27 which is the "max" amongst nodes of a voluminal element.

The family of points of Gauss which is used here is 'RIGI'. 'C' is it which is given in argument of the routine ELREFE_INFO . This family is coherent with the choice made in the catalogue:

```
EFLUXPG = LocatedComponents (phys=PHY.FLUX_R, type=' ELGA', location='
RIGI',
```

```
components= ('FLOW', 'FLUY', 'FLUZ',))
```

The routine RCVALA is used to recover the value of thermal conductivity (LAMBDA). As this parameter can be a function of time, one provides the value of the moment in argument of entry of the routine RCVALA .

gradient of the temperature is calculated in two times. The routine DFDM3D the gradient of the functions of form on the element into 1 point of Gauss given makes it possible to calculate. One can then use the gradient of the functions of form and the value of the temperature on the nodes to calculate the gradient of the temperature.

The storage of the flows calculated with the address IFLUX respect strictly the convention of storage of the local fields: 3 components FLOW , FLUY and FLUZ for each point of Gauss.

4 Details not used in the selected example

4.1 Description of the heading of one type_element

To supplement...

4.2 Local modes ELNO/ DIFF

To supplement...

4.3 Conventions of names for the local modes

To supplement...

4.4 To supplement...Names reserved for certain local modes (ddl_meca, ddl_ther, ddl_acou)

To supplement...

Warning : The translation process used on this website is a "Machine Translation". It may be imprecise and inaccurate in whole or in part and is provided as a convenience.

Copyright 2021 EDF R&D - Licensed under the terms of the GNU FDL (<http://www.gnu.org/copyleft/fdl.html>)

4.5 Local fields of standard elementary vector or elementary matrix

To supplement...

4.6 Optional fields, routine tecach. F90

To supplement...

4.7 Family of points of Gauss "MATER"

To supplement...