Introduire une nouvelle modélisation dans AFFE_MODELE

Résumé :

Ce document décrit ce qu'il faut faire pour introduire une nouvelle modélisation dans l'opérateur AFFE_MODELE de *Code_Aster* .

En quelques mots, il faut :

- Ajouter la modélisation au catalogue de la commande AFFE_MODELE,
- Introduire un petit bloc de texte dans le catalogue des phénomènes et modélisations,
- Écrire un ou plusieurs catalogues d'éléments.
- Écrire les routines de calculs élémentaires spécifiques aux éléments de cette nouvelle modélisation.

L'objet de ce document est principalement de présenter le catalogue Commons/phenomenons_modelisations.py et la structure générale d'un catalogue de Elements/xxx.py.

Le reste des actions à faire est décrit dans le document [D5.02.05] « Introduire un nouveau calcul élémentaire ».

Table des Matières

1 Introduction....3
2 Ajout de la modélisation au catalogue de commande....3
3 Modification du catalogue des phénomènes et modélisations....3
3.1 Présentation du catalogue phenomenons_modelisations.py....3
3.1.1 Arguments : modelisation, dim et code :....4
3.1.2 Argument attrs....4
3.1.3 Argument elements....5
3.2 Introduction d'une nouvelle modélisation dans le catalogue phenomenons_modelisations.py....5
4 Création des catalogues d'éléments....7
4.1 Présentation sommaire d'un catalogue d'éléments....7

1 Introduction

Le choix de la modélisation s'effectue à travers l'opérateur AFFE_MODELE de *Code_Aster* . Par exemple, l'utilisateur écrira dans son fichier de commande :

```
MOME = AFFE_MODELE(MAILLAGE = MAIL,
                   AFFE=_F(TOUT = 'OUI', MODELISATION = 'AXIS_JOINT_HMS',
                           PHENOMENE = 'MECANIQUE'))
```

Dans l'objectif de pouvoir proposer à l'utilisateur d'autres modélisations, on va décrire dans ce document une méthodologie pour introduire une nouvelle modélisation dans *Code_Aster* .

Introduire une modélisation dans *Code_Aster* nécessite de se poser les questions suivantes:

- Dans quel phénomène vais-je ajouter ma modélisation ?
- Quelles sont les dimensions géométriques et topologiques des éléments finis ?
- Quelles sont les mailles concernées par cette modélisation ?
- Quels sont les éléments « principaux » et les éléments de bord ?
- Quels sont les attributs que l'on peut définir ?
- Quels sont les calculs réalisables avec cette modélisation ?

Nous répondrons à ces questions dans ce document.

D'autres questions relatives aux éléments finis sont traitées dans d'autres documents :

- [D5.02.01] Comment introduire une nouvelle grandeur ou de nouvelles composantes (CMP) dans une grandeur existante ?
- [D5.02.02] Comment introduire un nouveau type de maille (type_maille) ou un nouvel élément de référence (ELREFE) ?
- [D5.02.05] Comment introduire un nouveau calcul élémentaire ?

2 Ajout de la modélisation au catalogue de commande

- Catalogue à modifier : code_aster/Cata/Commands/affe_modele.py

La nouvelle modélisation est à ajouter à l'ensemble des modélisations utilisables par la commande AFFE_MODELE.
Il suffit de l'ajouter à la liste des possibles (champ into).

3 Modification du catalogue des phénomènes et modélisations

- Catalogue à modifier : Commons/phenomenons_modelisations.py

3.1 Présentation du catalogue phenomenons_modelisations.py

Ce catalogue se décompose en plusieurs parties : une partie dédiée à chaque phénomène : mécanique, thermique, acoustique, ...

Pour chaque phénomène, il existe un bloc correspondant à chaque modélisation. Par exemple, pour la modélisation 'AXIS_SI' du phénomène 'MECANIQUE', nous avons :

```
phen.add('AXIS_SI', Modelisation(dim=(2,2), code='AXS',
    attrs=(
        (AT.AXIS,'OUI'),
        (AT.NBSIGM,'4'),
        (AT.TYPMOD,'AXIS'),
    ),
    elements=(
        (MT.QUAD8     , EL.MEAXQS8),
        (MT.SEG3      , EL.MEAXSE3),
    )))
```

Nous allons présenter les arguments de ce bloc :

- Modelisation
- dim
- code
- attrs
- elements

3.1.1 Arguments : modelisation, dim et code :

- L'argument modelisation fournit le nom de la modélisation. Dans l'exemple ci-dessus, il s'agit de la modélisation 'AXIS_SI'.

- L'argument dim (2 entiers) fournit respectivement la dimension topologique et la dimension géométrique de la modélisation :
  - La dimension géométrique correspond à dimension de l'espace ambiant (3 ou 2),
  - la dimension topologique correspond à la dimension des éléments « principaux » de la modélisation.

  Dans cet exemple, la dimension topologique est identique à la dimension géométrique, mais ce n'est pas toujours le cas.

  Par exemple, une modélisation 'DKT' (coques minces) recueille des éléments dont les mailles support sont de dimension 2 (triangles, quadrangles), pourtant les nœuds du maillage sont exprimés dans le repère 3D (suivant X, Y, Z). On a donc pour cette modélisation: dim=(2, 3)

- L'argument code fournit comme son nom l'indique, un code. Il s'agit d'une chaîne de 3 caractères permettant d'identifier la modélisation. Pour la modélisation 'AXIS_SI', le code choisi est 'AXS'. Ce « code » est forcément différent pour toutes les modélisations. C'est un alias (sur 3 caractères exactement) du nom de la modélisation.

3.1.2 Argument attrs

L'argument attrs permet de définir les attributs portés par tous les éléments de la modélisation.

Exemple : attrs=(
(AT.AXIS,'OUI'), …

La liste attrs est formée de couples (attribut, valeur)

A quoi servent-ils?

Ils permettent de fournir des informations dans le source fortran et d'envisager des traitements en fonction de ces informations. Dans l'exemple ci-dessus, l'attribut AT.AXIS=OUI peut être interrogé dans une routine de calcul élémentaire afin de modifier le poids d'intégration des points de Gauss.

Les attributs servent également à regrouper des éléments qui ont des caractères en commun : par exemple, les éléments de « poutre », de « coque », …
Ces regroupements sont utilisés dans les blocs « CondCalcul » des catalogues d'options.

Les attributs sont tous définis dans le catalogue Commons/attributes.py
Il est très important de bien documenter ces attributs pour que leur sens soit le moins ambigu possible.

Remarques :

Les différents attributs définis dans ce catalogue sont affectés à l'ensemble des type_element de la modélisation. Si l'on veut qu'un attribut ne soit associé qu'à un seul type_element, il faut alors définir cet attribut dans le catalogue du type_element.

Attention : Lorsqu'un élément fini est utilisé par plusieurs modélisations (c'est souvent le cas des éléments de bord), il peut y avoir ambiguïté sur la valeur des attributs de cet élément fini. L'élément héritera de TOUS les attributs définis dans l'ensemble des modélisations qui l'utilisent. Si un même attribut est défini plusieurs fois avec des valeurs différentes, on lui affecte la valeur « ### ». C'est toujours le cas de figure de l'attribut MODELI.

Comment récupérer la valeur d'un attribut dans le source fortran ?

Les routines lteatt.F90 et teattr.F90 permettent d'accéder aux attributs d'un type_element.

3.1.3 Argument elements

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

Cet argument sert à définir tous les éléments de la modélisation (et de son bord). C'est une liste de couples (type_maille, type_element).

Dans notre exemple, le couple : (MT.QUAD8 , EL.MEAXQS8),

signifie que l'on attribue pour cette modélisation, l'élément de type MEAXQS8 aux mailles quadrangulaires à 8 nœuds de type QUAD8.

3.2 Introduction d'une nouvelle modélisation dans le catalogue phenomenons_modelisations.py

Tout d'abord, il faut se placer dans la partie correspondant au phénomène de votre modélisation (MECANIQUE , THERMIQUE ou ACOUSTIQUE). Ensuite place à l'écriture du bloc correspondant à votre modélisation.

Vous devez commencer par :

- choisir un nom pour votre modélisation (au plus 16 caractères),
- attribuer un code à votre modélisation (3 caractères exactement),

- déterminer les « dimensions » de votre modélisation.

Vous devez alors vous poser la question des attributs portés par les éléments de la nouvelle modélisation. Il est important de parcourir la liste de tous les attributs existants pour savoir si vos éléments sont concernés par ces attributs. Par exemple, ce serait une erreur de ne pas déclarer `AT.AXIS='OUI'` pour un élément axisymétrique.

L'étape suivante consiste à choisir :

- les types de mailles que vous souhaitez associer à votre modélisation. Vous pouvez consulter le catalogue `Commons/mesh_types.py` pour prendre connaissance des types de mailles présents dans *Code_Aster* ainsi que de leurs éléments de référence ,
- le nom du type de l'élément fini que vous souhaitez associer à chaque type de maille. Vous devez déterminer un nom d'au plus 16 caractères qui soit suffisamment explicite pour connaître le type de sa maille à la lecture de son nom.

Nous venons de répondre à quelques questions concernant la nouvelle modélisation.

Pour aller plus loin, il faut écrire le (ou les) catalogue(s) décrivant les nouveaux éléments finis de la modélisation, ainsi que les routines fortran `te00ij.f` réalisant leurs calculs élémentaires.

Dans le paragraphe suivant, nous décrivons (assez succinctement) le catalogue des type_element.

Pour de plus amples détails sur ce catalogue et sur l'écriture des routines fortran associées, on se reportera au document [D5.02.05] « Introduire un nouveau calcul élémentaire »

4 Création des catalogues d'éléments

- Les catalogues d'éléments sont localisés dans `.../catalo/cataelem/Elements`
- Un catalogue permet de décrire une famille d'éléments « voisins » (par exemple les différents éléments d'une même modélisation) . Le catalogue porte le nom du premier élément qui y figure.

4.1 Présentation sommaire d'un catalogue d'éléments

Nous allons présenter les grandes lignes. Pour plus d'informations, le lecteur est invité à consulter la documentation `D5.02.05` (« *introduire un calcul élémentaire* »).

Nous allons présenter un extrait du catalogue `ther_hexa20.py` . Nous allons mettre en avant l'option de calcul `FLUX_ELGA` qui permet de calculer le flux thermique aux points de Gauss de l'élément à partir du champ de température.

```
...

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',...,),...),
        )
    calculs = (
        ...

        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), ),
        ),
        ...
```

Texte 1: Extrait du catalogue `ther_hexa20.py`

- On commence par décrire les modes locaux spécifiques aux éléments décrits dans ce fichier (`MVECTTR, MMATTR, …`)

- Tous les éléments décrits dans ce fichier, sont des classes python qui héritent de la classe `Element` . `THER_HEXA20` étant le premier élément du fichier, c'est lui qui donne le nom du fichier.
  Les autres éléments du fichier (`THER_PENTA15, THER_TETRA10` , …) sont obtenus par héritage de `THER_HEXA20` .

- Les premières lignes :

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

Permettent de déclarer le type de maille associé à l'élément (`MT.HEXA20`) et son (ou ses) éléments de référence (`elrefe = (…)`)

- Vient ensuite la liste de tous les calculs élémentaires réalisés par l'élément (argument `calculs`).
  Les éléments du tuple `calculs` sont les calculs élémentaires.
  Par exemple pour `FLUX_ELGA` :

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

  On indique le numéro de la routine `te00ij` (ici `te0062.F90`) associé au calcul élémentaire, ainsi que la description des champs d'entrée et de sortie du calcul élémentaire. La description d'un champ est faite en associant (dans un couple) le paramètre de l'option et le mode local qui lui est associé.