

Introduire une nouvelle option de calcul élémentaire

Résumé :

Ce document décrit ce qu'il faut faire pour introduire une nouvelle option de calcul élémentaire dans *Code_Aster*.

Table des Matières

1 Introduction.....	3
2 Création du catalogue d'option	4
2.1 Trouver un nom pour ce catalogue.....	4
2.2 Structure du catalogue.....	4
2.3 Bloc COND_CALCUL_.....	6

1 Introduction

Pour Code_Aster, une option de calcul élémentaire correspond à un calcul conduisant à produire un ou plusieurs champs « par éléments »

Exemples d'options de calcul (`option`) :

- `RIGI_MECA` : calcul de la rigidité (comportement élastique)
- `FLUX_ELGA` : calcul du flux thermique connaissant la température aux nœuds
- `CHAR_MECA_PESA_R` : calcul du second membre lié à un chargement de pesanteur.

Les 3 exemples précédents montrent que les calculs élémentaires peuvent intervenir un peu partout dans Code_Aster : dans les commandes de calcul (matrices et seconds membre), comme dans les commandes de post-traitement (calcul des flux en thermique, des énergies mécaniques, ...)

Une `option` de calcul élémentaire a un nom (K16) et est décrite dans un catalogue rangé dans le répertoire `catalo/options`

Faire un calcul élémentaire correspond à demander le calcul d'une `option` sur une liste d'éléments finis (`ligrel`). Ce calcul est réalisé par l'appel à la routine « chapeau » de tous les calculs élémentaires : `calcul.f`.

Le calcul est dit « élémentaire » car ce sont les éléments finis du `ligrel` qui font le calcul. Pour qu'un calcul soit « élémentaire », il faut qu'il soit « local », c'est à dire qu'un élément fini puisse calculer sa participation en ayant uniquement connaissance de données locales (champs locaux sur lui-même). Ne connaissant que des quantités locales, il calcule logiquement un résultat « local » (la participation de l'élément fini).

La collection des résultats calculés par les différents éléments constitue un champ (global) « par éléments ». C'est un champ par nature discontinu entre les éléments.

Remarque : le calcul d'une option de calcul élémentaire peut produire plusieurs champs. C'est par exemple le cas de l'option `RAPH_MECA` utilisée dans l'opérateur `STAT_NON_LINE` : on calcule en même temps un champ de contrainte et un champ de variables internes.

Le champ global produit par un calcul élémentaire peut-être un `cham_elem` ou un `resuelem`. La différence entre les deux types de structure de données est petite : le `resuelem` est un champ élémentaire contenant des matrices ou vecteurs élémentaires destinés à être assemblés pour former une matrice globale et un second membre.

Lorsqu'un calcul élémentaire produit un `cham_elem`, celui-ci peut-être `ELNO`, `ELGA` ou `ELEM`. Un champ `ELNO` contient des valeurs sur les nœuds des éléments. Un champ `ELEM` est un champ constant par maille. Un champ `ELGA` est un champ contenant des valeurs sur les points d'intégration (ou de Gauss) des éléments.

Dans le paragraphe suivant, nous allons détailler la syntaxe que doit avoir le catalogue d'une option. Écrire le catalogue d'une nouvelle option n'est pas une fin en soi. Ce n'est qu'une étape préliminaire pour que des éléments finis puissent calculer cette option. Ceci est expliqué dans le document [D5.02.05] « Introduire un nouveau calcul élémentaire ».

2 Création du catalogue d'option

2.1 Trouver un nom pour ce catalogue

L'ensemble des catalogues d'option de *Code_Aster* se trouve dans le répertoire `options` du répertoire `catalo`. L'objectif est de concevoir un catalogue relatif à cette nouvelle option et de le stocker dans le répertoire `options`.

Par convention, le nom du catalogue d'option est celui de l'option. Le nom d'une option a au plus 16 caractères. Ainsi, le catalogue de l'option 'FLUX_ELGA' sera `flux_elga.cata`.

2.2 Structure du catalogue

La structure du catalogue est la suivante:

```
%& MODIF OPTIONS ...

Nom_de_l_option_en_majuscule
<< ligne(s) de commentaire décrivant l'option>>
OPTION __
  IN __
paramètre_in_1    grandeur_du_champ_associé_au_paramètre_in_1
<< ligne(s) de commentaire décrivant le paramètre 1>>
paramètre_in_2    grandeur_du_champ_associé_au_paramètre_in_2
<< ligne(s) de commentaire décrivant le paramètre 2>>
...
paramètre_in_n    grandeur_du_champ_associé_au_paramètre_in_n
<< ligne(s) de commentaire décrivant le paramètre n>>
  OUT __
paramètre_out_1 grandeur_du_champ_associé_au_paramètre_out_1 localisation
<< ligne(s) de commentaire décrivant le paramètre 1 de sortie>>
paramètre_out_2 grandeur_du_champ_associé_au_paramètre_out_2 localisation
<< ligne(s) de commentaire décrivant le paramètre 2 de sortie>>
...
```

La première ligne d'un catalogue d'option est la traditionnelle « carte MODIF » des sources d'Aster. La syntaxe est « %& MODIF OPTIONS ... ».

Quand on introduit une nouvelle option, il faut écrire :

« %& AJOUT OPTIONS »

On écrit à la deuxième ligne le nom de l'option en majuscule (FLUX_ELGA), suivi à la ligne suivante d'un commentaire entre '<<' et '>>' pour décrire explicitement la fonctionnalité de l'option de calcul.

Par exemple, pour l'option FLUX_ELGA, on peut écrire:

```
<< FLUX_ELGA : CALCUL DU FLUX THERMIQUE AUX POINTS DE GAUSS >>
```

Ensuite, on liste tous les couples (paramètre d'entrée, grandeur) utiles au calcul élémentaire. Là encore, on ajoute un commentaire pour décrire chaque paramètre.

Par exemple, l'un des paramètres utile au calcul, est le paramètre PGEOMER associé entre autres aux coordonnées des nœuds. On peut écrire par exemple :

```
PGEOMER GEOM_R
<< PGEOMER : COORDONNEES DES NOEUDS >>
```

Une fois tous les paramètres d'entrée définis, on décrit le ou les paramètres de sortie de la même manière, à la différence, qu'il faut ajouter une localisation :

- ELEM__ : pour un champ constant par élément,
- ELGA__ : pour un champ aux points de Gauss de l'élément,
- ELNO__ : pour un champ aux nœuds par élément.
- RESL__ : pour un champ de type `resuelem` (matrice ou vecteur).

Petit exemple abrégé :

```
%& MODIF OPTIONS ...
RIGI_MECA
OPTION__
  IN__
    PGEOMER  GEOM_R    << Géométrie des noeuds >>
    PMATERC  ADRSJEVE  << Matériau codé >>
    PCACOQU  CACOQU    << Caractéristiques des éléments de coque >>
    ...
  OUT__
    PMATUUR  MDEP_R    RESL__    << matrice de rigidité élastique >>
    ...
```

Quelques remarques :

Les champs paramètres de l'option (IN ou OUT) sont tous définis par un couple (paramètre, grandeur). La grandeur est souvent suffisante pour caractériser le champ. Par exemple, quand on parle du champ de `GEOM_R` (géométrie des nœuds), tout le monde comprend de quoi il s'agit. Mais il arrive qu'une option nécessite plusieurs champs d'une même grandeur. Penser par exemple à une option qui aurait comme champs d'entrée 2 champs de déplacements (grandeur `DEPL_R`) : l'un correspondant au déplacement au début du pas de temps et l'autre correspondant à un incrément de déplacement. Pour distinguer ces différents champs, on associe un « petit nom » à chaque champ : c'est le paramètre.

Pour notre exemple, on écrirait :

```
PDEPLM DEPL_R << Déplacement '- ' (début du pas de temps) >>
PDEPLD DEPL_R << Incrément de déplacement >>
```

Quand il n'y a pas d'ambiguïté (un seul champ pour une grandeur donnée), il est d'usage de nommer le paramètre associé à cette grandeur en ajoutant un 'P' devant le nom de la grandeur. Par exemple :

```
PCACOQU CACOQU << ... >>
```

Le catalogue d'une option est utilisé par tous les éléments finis qui calculent cette option. La liste des champs « in » doit donc être une liste « enveloppe » des champs nécessités par les éléments. Par exemple, les éléments de coque ont en général besoin d'informations géométriques (épaisseur par exemple) qui ne se trouvent pas dans le champ de géométrie (coordonnées des nœuds du maillage). Ces données géométriques manquantes se trouvent dans un champ de la structure de données `CARA_ELEM` (de la grandeur `cacoqu`). On trouvera donc souvent dans le catalogue d'une option le bloc :

```
PCACOQU CACOQU << Caractéristiques des éléments de coque.
Provient de la sd cara_elem. >>
```

Comme le catalogue d'une option est une « enveloppe » des besoins des éléments, la liste des champs paramètres d'une option peut être longue. C'est pourquoi il est très important de commenter chaque champ paramètre entre << ... >>.

On impose que le responsable d'une option choisisse pour chaque champ de sortie une « localisation » (`ELGA__` , `ELNO__` , ...). Le but de cette contrainte est de forcer une certaine homogénéité pour les différents éléments finis qui calculent cette option. Cela permet aussi de pouvoir

« typer » le `cham_elem` global produit. Un `cham_elem` sera ainsi toujours soit `ELGA` , soit `ELNO` soit `ELEM` .

2.3 Bloc `COND_CALCUL__`

A la fin du catalogue d'une option, il est très important d'ajouter un petit bloc indiquant les éléments finis concernés par cette option (ceux qui devraient la calculer).

Prenons l'exemple du bloc correspondant à l'option `RIGI_MECA_GE` :

```
COND_CALCUL__    PLUS__    PHENO='ME' BORD='0'  
                  MOINS__   PHENO='ME' DISCRET='OUI'
```

Il s'agit de désigner l'ensemble de TOUS les éléments finis pour lesquels cette option a du sens (et qui donc devraient savoir la calculer). En les désignant tous, on exprime, a contrario, que tous les autres éléments ne sont pas concernés par cette option (et donc qu'ils ne doivent pas calculer cette option).

On désigne des « paquets » d'éléments finis utilisant les attributs définis dans les catalogues d'éléments ou dans le catalogue `phenomene_modelisation__`.

Dans l'exemple précédent, on définit deux « paquets » d'éléments :

- 1) ceux qui ont l'attribut `PHENO='ME'` **et** l'attribut `BORD='0'`
- 2) ceux qui ont l'attribut `PHENO='ME'` **et** l'attribut `DISCRET='OUI'`

L'ensemble de tous les éléments devant calculer l'option est obtenue en « ajoutant » ou en « retirant » les paquets définis selon le mot clé `PLUS__` ou `MOINS__` .

Dans l'exemple précédent, l'ensemble des éléments devant calculer l'option `RIGI_MECA_GE` est obtenu en prenant tous les éléments du phénomène `MECANIQUE` , qui sont « principaux » (`BORD='0'`) et en retranchant les éléments discrets.