Introduire une nouvelle grandeur (ou composante)

Résumé :

Ce document décrit ce qu'il faut faire pour introduire une nouvelle grandeur dans *Code_Aster* ou une nouvelle composante dans une grandeur existante.

En quelques mots, pour ajouter une grandeur, il faut :

- modifier le catalogue décrivant les grandeurs,
- ajouter le nom de la grandeur dans un catalogue de commandes.

Pour introduire une composante dans une grandeur existante, il suffit :

- d'enrichir le catalogue des grandeurs.

Table des Matières

1 Introduction

Pour Code_Aster, une grandeur est composée d'un identifiant (son nom), d'un type et d'une liste de composantes.

Exemple de grandeur :

- DEPL_R : nom de la grandeur des déplacements réels aux nœuds. On associe à cette grandeur le type réel R et les composantes DX, DY, DZ, DRX, DRY, DRZ, ...

Dans *Code_Aster*, on distingue les grandeurs simples et les grandeurs élémentaires. Les grandeurs élémentaires sont attachées aux vecteurs ou matrices élémentaire. Elles sont construites à partir des grandeurs simples. Toutes ces grandeurs (simples et élémentaires) sont décrites dans le fichier (mal nommé !) grandeur_simple__.cata. La description des grandeurs élémentaires est faite au [§8]

Nous essaierons de répondre aux questions :

- Que faut-il faire pour ajouter une nouvelle grandeur ?
- Que faut-il faire pour ajouter une nouvelle composante dans une grandeur existante ?
- Quels catalogues faut-il modifier ?

L'introduction d'une nouvelle grandeur nécessite deux actions:

- la modification du catalogue des grandeurs : grandeur_simple.cata,
- la modification du catalogue des commandes : c_nom_grandeur.capy

Nous allons détailler successivement ces deux actions.

2 Modification du catalogue des grandeurs «grandeur_simple__.cata»

On présente dans ce paragraphe la structure du catalogue grandeur_simple__.cata, et on décrit comment on procède pour ajouter une grandeur ou une composante.

2.1 Description du catalogue

Le catalogue grandeur_simple.cata est présent dans le répertoire compelem du répertoire catalo. Nous présentons ci-dessous un extrait de ce catalogue :

```
GRANDEUR_SIMPLE__
<<  ABSC_R  Type:R  Abscisse curviligne le long d'un maillage filaire
       ABSC  : abscisse curviligne
       ABSC1 : abscisse curviligne du 1er noeud d'un SEG2
       ABSC2 : abscisse curviligne du 2ème noeud d'un SEG2
>>
       ABSC_R   = R    ABSC     ABSC1    ABSC2
<<  STAOUDYN Type:R Paramètres de Newmark si calcul dynamique
       STAOUDYN = 0 : statique
                = 1 : dynamique
       ALFNMK : paramètre de Newmark ALPHA
       DELNMK : paramètre de Newmark DELTA
>>
    STAOUDYN = R    STAOUDYN ALFNMK   DELNMK
```

Le bloc définissant une grandeur se présente par :

- Une zone délimitée par '<<' et '>>' destinée aux commentaires. On y décrit en quelques lignes la grandeur que l'on définit, son type, et ses composantes. Par exemple, pour le premier bloc délimité par '<<' et '>>' , la grandeur de nom ABSC_R de type R représente l'abscisse curviligne le long d'un maillage filaire; elle est composée de trois composantes ABSC, ABSC1, ABSC2 caractérisant respectivement l'abscisse curviligne, l'abscisse curviligne du premier nœud du SEG2 et l'abscisse curviligne du second nœud du SEG2.

- Une ligne (ou plusieurs si le nombre de composantes est conséquent) définissant la grandeur, ses composantes, et le type fortran de ses composantes (R,C,F,I,K8,...). Cette ligne se présente sous la forme :

```
nom_de_grandeur = type composante_1 composante_2 ... composante_n
```

Remarque :

Pour chaque grandeur, il n'y a qu'une seule zone de commentaires possible (entre << et >>). Cette zone doit être placée **avant** la description de la grandeur. Elle lui est logiquement associée et sera imprimée dans certains messages d'erreur pour aider l'utilisateur. Quand on écrit ces commentaires, c'est d'abord à l'utilisateur qu'il faut penser.

2.2 Ajout d'une grandeur simple

Le développeur désireux ajouter une grandeur simple va devoir définir un nom de grandeur, attribuer un type, et associer des composantes à sa grandeur.

2.2.1 Choix du type

Le type à associer à la grandeur est le type fortran des valeurs de ses composantes. Le développeur devra choisir un type parmi les choix ci-dessous:

- R : réel,
- I : entier,
- C : complexe,
- K8 : chaine de 8 caractères,
- K16 : chaine de 16 caractères,
- K24 : chaine de 24 caractères,
- ...

2.2.2 Nom de la grandeur

La première chose à faire (après avoir déterminer un type) est de définir un nom de grandeur qui soit suffisamment explicite et non défini dans ce catalogue.

Comment définir un nom de grandeur?

Les noms sont représentés par un chaine d'au plus 8 caractères. Dans l'exemple précédent, les deux grandeurs définies ont pour nom : ABSC_R et STAOUDYN.

Remarque :

Il est conseillé d'introduire le type de la grandeur dans son nom (par exemple ABSC_R), car il est plus simple pour le développeur de manipuler une grandeur dont le type lui est implicitement connu. Usuellement, on le fait précéder du caractère «_» (underscore) afin de le dissocier du nom symbolique de la grandeur.

2.2.3 Noms des composantes

La seconde chose à faire est de réfléchir sur les composantes. On doit se poser les questions suivantes :

- Ai-je besoin de nommer explicitement chaque composante ?
- Connait-on par avance le nombre de composantes ?

Premier cas (le plus simple) : on connaît précisément le nombre de composantes et on souhaite nommer chacune d'elles. Pour cela, on définit pour chaque composante un nom d'au plus 8 caractères. C'est le cas de l'exemple suivant :

```
<<  ABSC_R  Type:R  Abscisse curviligne le long d'un maillage filaire
       ABSC  : abscisse curviligne
       ABSC1 : abscisse curviligne du 1er noeud d'un SEG2
       ABSC2 : abscisse curviligne du 2ème noeud d'un SEG2
>>
       ABSC_R   = R    ABSC     ABSC1    ABSC2
```

Second cas : on souhaite définir une grandeur dont le nombre de composantes est conséquent et dont le nom de chacune d'elles importe peu. Pour ce faire, il existe dans Code_Aster des grandeurs pour ce genre de situation. Les grandeurs NEUT_X (où X représente le type de la grandeur).

Exemple:

```
<<  NEUT_R Type:R Grandeur ''neutre'' de type réel
      X(1) : composante 1
      X(2) : composante 2
      X(3) : composante 3
      X(4) : composante 4
      X(5) : composante 5
      ...
```

```
>>
    NEUT_R    = R     X[30]
```

Cet exemple décrit une grandeur «neutre» de type R pouvant recueillir au plus 30 composantes.

Quand le nombre de composantes dépasse 30, il est également possible d'utiliser les grandeurs « neutres » N120_R , N120_I et N480_I. Celles-ci peuvent contenir respectivement 120 composantes de type R , 120 composantes de type I et 480 composantes de type I .

Remarques :

On renseigne dans grandeur_simple__.cata le nombre maximal de composantes qu'une grandeur puisse disposer. Le nombre de composantes d'une grandeur nécessaire au calcul élémentaire est défini dans chaque catalogue d'élément.
L'utilisation des grandeurs «NEUT_X» permet d'éviter la multiplication des grandeurs.

Un **troisième cas** peut se présenter : Il existe dans Code_Aster une grandeur «spéciale» pouvant avoir un nombre indéterminé de composantes. Il s'agit de la grandeur **VARI_R** . On ne distingue pas les composantes de cette grandeur. Le nombre de composante est donné en dehors des catalogues. On associe à la grandeur un nom conventionnel de composante, le nom VARI .

Exemple:

```
    VARI_R    = R     VARI
```

Remarques :

La composante VARI n'est pas exploitable.
Les composantes d'un champ de grandeur VARI_R sont nommées V1, V2, …
La grandeur VARI_R est souvent utilisée pour les lois de comportement non linéaire.
Il existe aussi la grandeur VAR2_R similaire à VARI_R. Le développeur est invité à consulter le catalogue grandeur_simple__.cata pour plus d'informations sur cette grandeur .

2.3 Ajout d'une composante dans une grandeur existante.

Il suffit d'enrichir la liste des composantes de la grandeur par un nouveau nom. Bien évidemment, vous devrez commenter la composante.

Exemple:
Supposons que l'on souhaite enrichir la grandeur TEMP_F par une composante décrivant le paramètre de Lagrange dû à la dualisation des conditions limites.

Description de la grandeur TEMP_F actuel :

```
<<  TEMP_F Type:K8 Température inconnue du phénomène thermique
       TEMP     : température
       TEMP_MIL : température sur le feuillet Moyen  (coques)
       TEMP_INF : température sur la face inférieure (coques)
       TEMP_SUP : température sur la face supérieure (coques)
>>
    TEMP_F    = K8    TEMP TEMP_MIL TEMP_INF TEMP_SUP
```

Les différentes étapes sont:

- établir un nom pour cette nouvelle composante: le choix LAGR paraît suffisamment significatif.
- introduire un commentaire décrivant la nouvelle composante.

Après la réalisation de ces deux étapes, on aboutit à :

```
<<  TEMP_F Type:K8 Température inconnue du phénomène thermique
       TEMP : température
       TEMP_MIL : température sur le feuillet Moyen  (coques)
       TEMP_INF : température sur la face inférieure (coques)
       TEMP_SUP : température sur la face supérieure (coques)
       LAGR : paramètre de Lagrange du a la dualisation des
       conditions aux limites
>>
    TEMP_F   = K8   TEMP TEMP_MIL TEMP_INF TEMP_SUP LAGR
```

Désormais, la composante `LAGR de type K8` décrivant le paramètre de Lagrange est associée à la grandeur `TEMP_F`.

3 Modification du catalogue «c_nom_grandeur.capy»

Une deuxième étape à ne pas oublier lors de l'introduction d'une nouvelle grandeur simple dans Code_Aster, est la modification du catalogue `c_nom_grandeur.capy`.

Ce catalogue est présent dans le répertoire `commun` du répertoire `catapy`.

Il recense toutes les grandeurs simples présentes dans Code_Aster.

Extrait:

```
def C_NOM_GRANDEUR() : return  (
"ABSC_R",
"ADRSJEVE",
"ADRSJEVN",
"CAARPO",
"CACABL",
"CACOQU",
"CADISA",
"CADISK",
…
)
```

A quoi sert-il ?

Il est utilisé par le superviseur python pour vérifier la saisie des utilisateurs dans les fichiers de commandes. Par exemple, lors de la création d'un champ (opérateur `CREA_CHAMP`), il faut fournir au mot-clé `TYPE_CHAM` une chaine de caractères décrivant le type de champ à construire (localisation et grandeur). Si `TYPE_CHAM = 'ELNO_SIEF_R'` (champ de contraintes réel aux nœuds par élément), Code_Aster exploite le catalogue `c_nom_grandeur.capy` afin de vérifier si la chaine de caractères `'SIEF_R'` saisie par l'utilisateur correspond à une grandeur.

4 Grandeurs « élémentaires »

Jusqu'à présent, nous n'avons parlé que des grandeurs « simples ». Une grandeur « simple » est une liste de composantes nommées.

Les champs aux nœuds (cham_no_xxx), les cartes (carte_xxx) et les champs par éléments (cham_elem_xxx) sont tous associés à une grandeur « simple ». Par exemple, un champ aux nœuds de déplacement est associé à la grandeur DEPL_R. Sur chacun des nœuds du maillage, ce champ peut « porter » une (ou plusieurs) composantes de la grandeur DEPL_R.

Les grandeurs « élémentaires » sont celles qui sont associées aux structures de données resuelem (vecteurs ou matrices élémentaires).

Un resuelem est un « champ » contenant 1 vecteur élémentaire (ou 1 matrice élémentaire) par maille. La grandeur associée à ce champ est une grandeur élémentaire.

4.1 Syntaxe

La description des grandeurs élémentaires dans le fichier grandeur_simple__.cata est faite derrière le mot clé GRANDEUR_ELEMENTAIRE__.

Exemples :

```
GRANDEUR_ELEMENTAIRE__
   ...
   VDEP_R 1 DEPL_R
   ...
   MDEP_R 2 DEPL_R   DEPL_R   MS
   MDNS_R 2 DEPL_R   DEPL_R   MR

   ...
```

4.1.1 Grandeur « vecteur élémentaire »

Une grandeur de type « vecteur élémentaire » (par exemple VDEP_R ci-dessus) est simplement décrite par le nombre « 1 » (vecteur -> dimension = 1) et la grandeur simple associée à la grandeur élémentaire (ici DEPL_R).

4.1.2 Grandeur «matrice élémentaire »

Une grandeur de type «matrice élémentaire » (par exemple MDEP_R ci-dessus) est décrite par le nombre « 2 » (matrice -> dimension = 2) et les 2 grandeurs simples (ligne et colonne) associées à la grandeur élémentaire (ici DEPL_R et DEPL_R).

Remarque : la syntaxe permet de définir une matrice élémentaire avec 2 grandeurs simples différentes (par exemple : DEPL_R, TEMP_R) mais cette possibilité n'est jamais utilisée dans le code.

Pour les grandeurs « matrice », on complète cette description par l'un des 2 mots réservés : MS (matrice symétrique) ou MR (matrice non-symétrique).

4.2 Conventions de stockage

Les scalaires (en général des réels) qui permettent de représenter une grandeur élémentaire sont nombreux : par exemple, la matrice de rigidité (symétrique) d'un élément mécanique MECA_HEXA20 contient (60*61/2) réels.

Il n'est pas question de nommer tous ces réels (comme les composantes d'une grandeur simple).

Des conventions sont nécessaires concernant le stockage de ces scalaires.

Pour un vecteur élémentaire (destiné à être assemblé pour donner un second membre), on cherche à stocker quelque chose qui ressemble à un champ « aux nœuds » pour chaque élément. La convention de stockage est naturelle. Par exemple :

N1			N2			N3			...
DX	DY	DZ	DX	DY	DZ	DX	DY	DZ	...
1	2	3	4	5	6	7	8	9	...

Pour une matrice élémentaire, deux conventions sont nécessaires :

Pour une matrice symétrique, on stocke dans l'ordre suivant :

1				
2	3			
4	5	6		
7	8	9	10	
11	12	13	14	15

Pour une matrice non-symétrique, on stocke dans l'ordre suivant :

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Sachant que l'ordre des lignes et des colonnes est le même que celui d'un vecteur élémentaire.