Introduire de nouvelles fonctionnalités à CALC_EUROPLEXUS

Résumé :

Ce document décrit les étapes à suivre pour ajouter de nouvelles fonctionnalités à la macro-commande CALC_EUROPLEXUS. Il peut s'agir :

- d'éléments,
- de lois de comportement,
- de chargements.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Table des Matières

1 Introduction 3
2 Structure 4
2.1 Étapes 4
2.1.1 Lecture des données et traduction 4
2.1.2 Récupération et mise en forme des résultats 4
2.2 Catalogue de traduction 4
3 Ajout d'une nouvelle loi de comportement 5
3.1 Catalogue de commande 5
3.2 Dictionnaire des comportements 5
3.2.1 Étapes 5
3.2.2 Détails sur les transformations de variables internes. 5
3.2.2.1 Passage EPX vers Code_Aster 5
3.2.2.2 Passage Code_Aster vers EPX 6
3.3 Dictionnaire des lois ou matériaux 6
3.3.1 Étapes 6
3.4 Synthèse 7
4 Ajout d'un nouveau type d'élément 8
4.1 Dictionnaire des modélisations 8
4.2 Ajout d'un nouveau CARA_ELEM 9
4.2.1 Dictionnaire des caractéristique élémentaires 9
4.2.2 Étapes 9
4.3 Passage de contrainte à effort 10
5 Ajout d'un nouveau chargement 11
5.1 Ajout d'une liaison 11
5.1.1 Dictionnaire des liaisons 11
5.1.2 Étapes 11
5.2 Ajout d'un chargement 12
5.2.1 Dictionnaire des chargements 12
5.2.2 Étapes 12
5.3 Traitement des choix multiples 12

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1 Introduction

La macro-commande CALC_EUROPLEXUS a pour but de lancer un calcul avec le logiciel EUROPLEXUS (EPX dans la suite du document) à partir d'une mise en donnée classique de *Code_Aster*.

Parmi les fonctionnalités de *Code_Aster*, peu sont disponibles dans CALC_EUROPLEXUS. Cela s'explique d'une part par le fait qu'il doit y avoir une correspondance très proche entre la fonctionnalité en question dans *Code_Aster* et la fonctionnalité équivalente dans EPX. D'autre part cette fonctionnalité doit être intégrée à la macro-commande de façon à faire une traduction correcte entre *Code_Aster* et EPX.

Le présent document décrit de la manière la plus complète possible la marche à suivre pour ajouter une fonctionnalité souhaitée. Dans une première partie, on décrit de manière succincte la structure de la macro-commande en pointant sur les informations essentielles pour le développeur. Cela sera suivi par trois autres parties décrivant :

- l'ajout d'une loi de comportement ;
- l'ajout d'un type d'éléments (et d'un mot-clé d'AFFE_CARA_ELEM) ;
- l'ajout d'un chargement.

Dans chacun de ces cas, on part du postulat que le développeur connaît dans EPX la fonctionnalité équivalente à celle qu'il souhaite ajouter dans *Code_Aster*.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2 Structure

2.1 Étapes

La macro-commande CALC_EUROPLEXUS se décompose en trois principales étapes :
1) lecture des données d'entrée et traduction ;
2) lancement du calcul EPX ;
3) récupération et mise en forme des résultats.

Lors de l'ajout d'une fonctionnalité les deux étapes concernées sont 1) et 3).

2.1.1 Lecture des données et traduction

Cette étape correspond exactement à l'utilisation de CALC_EUROPLEXUS avec LANCEMENT='NON'. Elle produit le fichier de commande EPX et le maillage MED (comportant éventuellement un état initial).

Modules dédiés

Le corps de la macro (calc_europlexus_ops.py) traite successivement les différentes entrées (modèle, caractéristiques élémentaires, comportement et matériaux, chargements, …). Dans la plupart des cas, les traitements effectifs sont faits dans des modules python dédiés. Ces modules se trouvent dans le répertoire Calc_epx/.

calc_epx_geom.py	Traitement du modèle
calc_epx_mate.py	Traitement des matériaux et comportements
calc_epx_cara.py	Traitement des caractéristiques élémentaires
calc_epx_char.py	Traitement des chargements
calc_epx_poutre.py	Traitement spécial des orientations des poutres et discrets

Écriture du fichier de commande

Le fichier de commande EPX est un fichier texte fonctionnant par mot-clé de quatre caractères. Il possède une structure assez régulière (bien que de nombreuses exceptions existent). Le module calc_epx_struc.py a été créé pour uniformiser au maximum la mise en donnée EPX en cherchant les plus petites unités régulières possibles pour remonter jusqu'aux directives (mots-clés principaux). Différents objets ont été créés pour faciliter le développement de la macro et produire un fichier de commande lisible et structuré.

2.1.2 Récupération et mise en forme des résultats

Les résultats issus d'EPX sont contenus dans un fichier MED. La lecture de ce fichier et la mise en forme des résultats dans un objet evol_noli est faite par une seconde macro-commande : LIRE_EUROPLEXUS appelée par CALC_EUROPLEXUS. On précise toutefois que cette macro-commande peut être appelée de manière indépendante de CALC_EUROPLEXUS.

2.2 Catalogue de traduction

Le catalogue de traduction (calc_epx_cata.py) est le module central de CALC_EUROPLEXUS et LIRE_EUROPLEXUS du point de vue d'un développeur souhaitant ajouter une fonctionnalité. C'est dans ce fichier que sont détaillées toutes les fonctionnalités autorisées ainsi que leur traduction EPX (dans la mesure du possible).

L'ajout de toute fonctionnalité commence donc par l'édition de ce fichier.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3 Ajout d'une nouvelle loi de comportement

On liste les différentes actions à opérer sur le code pour ajouter une nouvelle loi de comportement. Certaines actions sont détaillées dans des paragraphes dédiés quand cela est nécessaire.

3.1 Catalogue de commande

Ajouter le nom du nouveau comportement dans le fichier `c_comportement.capy` dans la partie `COMMAND == 'CALC_EUROPLEXUS'`.

3.2 Dictionnaire des comportements

Dans le catalogue de traduction (`calc_epx_cata.py`), dupliquer un couple clé/valeur du dictionnaire `cata_compor`.
Par exemple pour le comportement `VMIS_ISOT_TRAC` ce couple est le suivant :

```
'VMIS_ISOT_TRAC' : {
        'LOI'    : ['ELAS', 'TRACTION'],
        'BESOIN' : ['o', 'o',],
        'REPEAT' : ['n', 'n'],
        'NOM_EPX' : 'VMIS ISOT',
        'NOM_EPX_CH_MED' : 'ISOT',
        'NB_VAR_ASTER' : 2,
        'NB_VAR_EPX'   : 5,
        'TRANSFO'      : True,
            },
```

3.2.1 Étapes

1) Remplacer la clé existante par le nouveau comportement.
2) Renseigner les mots-clé matériaux nécessaires au fonctionnement de la loi dans la clé `'LOI'` .
3) Il suffit de recopier la liste `mc_mater` du module python du comportement (`./bibpyt/Comportement/` *mon_comportement* `.py`).
4) Dire pour chacun d'eux dans `'BESOIN'` si leur présence est obligatoire ou facultative (`'o'/'f'`) et dans `'REPEAT'` s'ils peuvent être répétés ou non (`'y','n'`). Dans la grande majorité des cas la présence de ces mots-clés dans le matériau est obligatoire et ils ne peuvent pas être répétés (aménagement pour les spécificités de `GLRC_DAMAGE`).
5) Dans `'NOM_EPX'` donner le nom du matériau correspondant dans EPX (mot-clé de la directive `MATE`). Dans certains cas, la chaîne de quatre caractères permettant de connaître le matériau EPX associé à un champ donné dans le fichier MED n'est pas celle formée par les quatre premiers caractères de `'NOM_EPX'`. Dans ce cas il faut ajouter la clé `'NOM_EPX_CH_MED'` avec la bonne valeur.
6) Indiquer respectivement dans `'NB_VAR_ASTER'` et `'NB_VAR_EPX'` le nombre de variables internes de la loi dans *Code_Aster* et EPX (champ `ECRO` dans EPX).
7) Indiquer dans le mot-clé `'TRANSFO'` si une routine de transformation des variables internes d' EPX vers *Code_Aster* est nécessaire et donc à développer (`True/False`).
8) Si besoin, développer la transformation des variables internes de *Code_Aster* vers EPX pour envoyer un champ de variables internes (non nulles) en état initial.

3.2.2 Détails sur les transformations de variables internes.

3.2.2.1 Passage EPX vers *Code_Aster*

Si `'TRANSFO'` est `False` et qu'il y a plus de composantes côté *Code_Aster* que côté EPX, la lecture des variables internes échouera. Il faut a minima activer `'TRANSFO'` et programmer l'ajout du nombre de composantes manquantes en leur affectant la valeur 0.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Si 'TRANSFO' est False et qu'il y a au moins autant de composantes côté EPX que côté *Code_Aster*, La composante i d'EPX est copiée sur la composante i de *Code_Aster* si $i \leq NbVarAster$. Les autres composantes sont oubliées.

Si 'TRANSFO' est True , il faut alors programmer la transformation. Pour cela il faut ajouter une nouvelle routine (en dupliquant une déjà présente) dans le module /bibpyt/Calc_epx/trans_var_int.py. Le nom de cette routine doit être tr_e2a_*mon_comportement*. Cette routine doit ensuite être branchée au reste du code dans la méthode transfo_var_int de la classe LireEPX() de lire_europlexus_ops.py.

3.2.2.2 Passage *Code_Aster* vers EPX

Quand le mot-clé VARI_INT de ETAT_INIT est mis à 'OUI', alors un champ de variables internes est ajouté à l'état initial envoyé à EPX. Lors de l'ajout d'une nouvelle loi de comportement, si rien n'est fait, le champ initial de variables internes sera nul sur les mailles sur lesquelles le nouveau comportement est affecté.
Pour que les valeurs des variables internes soient transférées il faut développer la transformation de *Code_Aster* vers EPX de la loi. Pour cela il faut :

- Créer une nouvelle routine de transformation sur le modèle de tr_a2e_vmis_john_cook dans le module ./bibpyt/Calc_epx/trans_var_int.py. Elle se nommera tr_e2a_*mon_comportement*.
- Ajouter l'appel à la nouvelle routine dans var_int_a2e du même module.

3.3 Dictionnaire des lois ou matériaux

Dans le catalogue de traduction (calc_epx_cata.py), dupliquer un couple clé/valeur du dictionnaire cata_lois . Les clés de ce dictionnaire sont formées par le nom du comportement et le mot-clé matériau (de DEFI_MATERIAU) que l'on souhaite définir séparés par un '/' .

Voici les trois premiers couples de ce dictionnaire :

```
'ELAS/ELAS' : {
        'PARA'     : ['E', 'NU', 'RHO', 'AMOR_ALPHA', 'AMOR_BETA'],
        'PARA_EPX' : ['YOUNG', 'NU', 'RO', 'KRAY', 'MRAY'],
        'BESOIN'   : ['o', 'o', 'o', 'f', 'f'],
        'TYPE'     : ['reel', 'reel', 'reel', 'reel', 'reel'],
                   },
'VMIS_ISOT_TRAC/ELAS' : {
        'PARA'     : ['E', 'NU', 'RHO',],
        'PARA_EPX' : ['YOUNG', 'NU', 'RO',],
        'BESOIN'   : ['o', 'o', 'o',],
        'TYPE'     : ['reel', 'reel', 'reel',],
                   },
'VMIS_ISOT_TRAC/TRACTION' : {
        'PARA'     : ['SIGM',],
        'PARA_EPX' : [['ELAS', 'TRAC'],],
        'BESOIN'   : ['o',],
        'TYPE'     : ['fonc',],
                   },
```

3.3.1 Étapes

1) Remplacer la clé existante par la clé à ajouter.
2) Dans 'PARA', entrer les noms des paramètres du matériau dans *Code_Aster*.
3) Dans 'PARA_EPX', entrer les noms des paramètres correspondant dans EPX. Dans certains cas, il n'y a pas correspondance parfaite entre les paramètres et un traitement spécial doit être effectué. On doit alors faire en sorte que la liste 'PARA_EPX' contienne à l'indice en question une autre liste afin de pouvoir détecter qu'un traitement spécial est à faire. Ce traitement est à programmer dans une routine à ajouter dans le module calc_epx_mate.py et à brancher dans la routine get_para_all du même module après la ligne :
   ```
   if type(para_epx) is list:
   ```
4) Dire pour chacun d'eux dans 'BESOIN' si leur présence est obligatoire ou facultative ('o'/'f').

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

5) Donner le type de chacun d'eux dans 'TYPE' , réel ou fonction ('reel'/'fonc').

3.4 Synthèse

Dans le cas d'une loi de comportement parfaitement identique entre *Code_Aster* et EPX, l'ajout se résume à des modifications de catalogues/dictionnaires.

Pour les autres cas, des traitements spécifiques ciblés sont à programmer. On en compte pour l'instant deux. La transformation des composantes de variables internes et la correspondance des paramètres des lois. Pour ces deux traitements, il suffit de dupliquer l'existant et de l'adapter au cas en question.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4 Ajout d'un nouveau type d'élément

Contrairement à *Code_Aster*, une modélisation ou géométrie EPX (directive GEOM) est associée à une maille support unique. Il est important d'avoir cela à l'esprit avant d'entreprendre l'ajout d'un nouvel élément.

4.1 Dictionnaire des modélisations

La première étape consiste à ajouter au dictionnaire cata_modelisa du catalogue de traduction une clé du nom de la modélisation *Code_Aster*. Voici plusieurs clés de ce dictionnaire :

```
'Q4GG' : {
        'MODE_EPX': {
            'TRIA3' : ['T3GS'],
            'QUAD4' : ['Q4GS'],
                     },
        'ETAT_INIT' : True,
        'RESU_ELEM' : True,
        'CONT_ASTER' : ['NXX', 'NYY', 'NXY', 'MXX', 'MYY', 'MXY', 'QX', 'QY'],
        'MC_CARA' : 'COQUE',
        'MODI_REPERE' : 'COQUE',
              },
'POU_D_E' : {
        'MODE_EPX': {
            'SEG2' : ['POUT']
                    },
        'ETAT_INIT' : False,
        'RESU_ELEM' : False,
              },
'DIS_TR' : {
        'MODE_EPX': {
            'POI1' : ['APPU', 'PMAT']
                    },
        'ETAT_INIT' : False,
        'RESU_ELEM' : False,
              },
```

1) Dans le dictionnaire 'MODE_EPX', ajouter une clé du nom de la maille support de la modélisation EPX à ajouter.
2) La valeur associée à cette clé doit être une liste, dans la plupart des cas de longueur 1, contenant la ou les noms des modélisations EPX correspondantes. Si la liste possède plusieurs éléments cela signifie qu'il existe plusieurs choix possibles. Dans les cas existants pour le moment on tranche entre les multiples modélisations à partir d'informations contenues dans la structure de données cara_elem . D'une manière générale, dans ces cas particuliers, il faut ajouter des traitements spécifiques. Ce point est détaillé dans la partie 4.2.2 à l'item 7 .
3) Mettre ETAT_INIT à True si un champ de contrainte initial sur cet élément peut être envoyé à EPX. **Cela nécessite des développements dans EPX.** Sinon False.
4) Mettre RESU_ELEM à True si les champs de contraintes (et de variables internes) issus d'EPX peuvent être récupérés et mis en forme pour être intégrés à un résultat de *Code_Aster*. Sinon False.
5) Si 'RESU_ELEM' est True , indiquer dans le liste 'CONT_ASTER' les composantes de contraintes *Code_Aster* de cet élément. Attention elles doivent être données dans l'ordre des composantes correspondantes dans EPX. On précise qu'il faut obligatoirement que les contraintes dans *Code_Aster* et EPX correspondent parfaitement à l'exception de l'ordre et d'une transformation contrainte/effort.
6) Si 'RESU_ELEM' et qu'il est nécessaire d'opérer une transformation contrainte/effort sur le champ de contraintes pour cet élément, indiquer dans 'MC_CARA' le mot-clé de AFFE_CARA_ELEM auquel cet élément se réfère.
7) S'il est nécessaire d'appliquer aux contraintes un changement de repère, il faut l'indiquer dans 'MODI_REPERE'. Dans ces rares cas, des développements seront à faire pour traiter ce besoin.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4.2 Ajout d'un nouveau CARA_ELEM

L'élément à ajouter peut avoir besoin d'informations contenues dans un mot-clé de AFFE_CARA_ELEM. Si ce mot-clé n'est pas encore programmé dans CALC_EUROPLEXUS (voir dictionnaire cata_cara_elem du catalogue de traduction), il est nécessaire de l'ajouter.

4.2.1 Dictionnaire des caractéristique élémentaires

Pour cela il faut dupliquer un couple clé/valeur du dictionnaire cata_cara_elem et remplacer la clé par le mot-clé en question. On donne ici les premières valeurs de ce dictionnaire :

```
'COQUE': [
        {
            'SELECT' : {'EPAIS' : None,
                        },
            'TITRE': 'COQUES',
            'DIRECTIVE': 'COMPLEMENT',
            'MOT_CLE_ASTER': 'EPAIS',
            'MOT_CLE_EPX': 'EPAIS',
            'VERIF': {
                      'VECTEUR': None,  # pris en compte ailleurs
                      'ANGL_REP': None,  # pris en compte ailleurs
                      'COQUE_NCOU': [1],
                      'A_CIS': None,            # Pas de prise en compte par Exp
                      'COEF_RIGI_DRZ': None ,   # Pas de prise en compte par Exp
                      'MODI_METRIQUE':['NON',],
                      },
        }
    ],
    'POUTRE': [
        {
            'SELECT' : {'SECTION': 'RECTANGLE',
                        },
            'TITRE': 'ELEMENTS POUTRES',
            'DIRECTIVE': 'COMPLEMENT',
            'MOT_CLE_EPX': 'GEOP RECT',
            'CARA_ASTER': [' ', ' ', ' ', 'HY', 'HZ'],
            'CARA_EPX': ['VX', 'VY', 'VZ', 'AY', 'AZ'],
            'VERIF': {
                      'VARI_SECT': ['CONSTANT',],
                      },
        },

        {
            'SELECT' : {'SECTION': 'CERCLE',
                        },
            'TITRE': 'ELEMENTS POUTRES',
            'DIRECTIVE': 'COMPLEMENT',
            'MOT_CLE_EPX': 'GEOP CIRC',
            'CARA_ASTER': [' ', ' ', ' ', 'R'],
            'CARA_EPX': ['VX', 'VY', 'VZ', 'DEXT'],
            'COEF_MULT' : [1., 1., 1., 2.],
            'VERIF': {
                      'VARI_SECT': ['CONSTANT',],
                      },
        }
```

Les valeurs des clés sont des listes de dictionnaires. Cela semble inutile dans les exemples ci-dessus mais cela permet de traiter le cas complexe de la clé 'DISCRETS'. On donne ensuite les significations des différentes clés permises dans ces dictionnaires. Cela est dû à la difficulté d'établir des correspondances simples entre les informations de *Code_Aster* (pas organisées de la même manière selon les mots-clés de AFFE_CARA_ELEM) et celles d'EPX (rarement uniformes).

4.2.2 Étapes

1) Renseigner dans dans 'SELECT', les mots-clés permettant de faire le choix entre les différents cas proposés. Dans le cas des poutres, cela permet de distinguer les poutres rectangulaires des poutres circulaires. Dans le cas des coques, il n'y a qu'une possibilité, cependant il faut quand

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

même renseigner un mot-clé pour que cela fonctionne. Si la valeur de la clé n'a pas d'importance, il faut donner « None ». La seule présence du mot-clé permet de faire le choix.

2) Donner un titre, qui sera en commentaire dans le fichier de commande EPX pour cette caractéristique élémentaire.
3) Indiquer dans quelles directives EPX doivent être intégrées les informations relatives à ce mot-clé. Dans la plupart des cas il s'agit de la directive COMP ou COMPLEMENT mais les informations sur les éléments discrets doivent se trouver dans la directive MATE.
4) Indiquer dans 'MOT_CLE_EPX', le mot-clé EPX correspondant.
5) Si la ou les valeurs d'un seul mot-clé *Code_Aster* sont à récupérer, il faut alors renseigner 'MOT_CLE_ASTER' avec le mot-clé *Code_Aster* en question. Si la valeur d'une seule caractéristique du mot-clé 'CARA' est à récupérer et qu'EPX n'utilise pas un autre mot-clé que la valeur de 'MOT_CLE_EPX' pour déclarer cette valeur, il faut également renseigner 'MOT_CLE_ASTER' avec le nom de la caractéristique.
6) Sinon, renseigner dans 'CARA_ASTER' la liste des noms des caractéristiques *Code_Aster* du mot-clé CARA à récupérer.
7) Si le fait d'affecter ces caractéristiques élémentaires à un groupe de mailles donné permet de trancher parmi plusieurs modélisations EPX possibles, indiquer dans 'MODE_EPX' la modélisation en question.
8) Généralement, quand plusieurs valeurs sont à récupérer, EPX associe un mot-clé à chacune d'elles. Il faut alors indiquer dans 'CARA_EPX' les noms de ces mot-clés (caractéristiques). Dans le cas idéal, la caractéristique d'indice i dans la liste doit correspondre à la valeur d'indice i dans la liste des valeurs récupérées. Il peut arriver que certaines caractéristiques dont a besoin EPX ne soient pas transmises par *Code_Aster via* ce chemin. Il faut cependant les indiquer quand même, on verra dans les points suivants comment faire les bons branchements.
9) Si des caractéristiques d'EPX ne sont pas fournies par cette voie et que 'CARA_ASTER' est présent, les listes CARA_EPX et CARA_ASTER doivent tout de même être de même longueur. Ainsi si la caractéristique d'indice i dans la liste CARA_EPX n'a pas de correspondance dans le mot-clé CARA, il faut mettre ' ' en indice i de CARA_ASTER.
10) Si des caractéristiques d'EPX ne sont pas fournies par cette voie et que 'MOT_CLE_ASTER' est présent, il faut renseigner le mot-clé 'IS_VALE_ASTER' dont la valeur sera une liste de même longueur que la liste CARA_EPX, comportant True en indice i si la valeur associée à la caractéristique de même indice de la liste CARA_EPX est présente dans la liste de valeur récupérées et False sinon.

Remarque :

> *Il est indispensable que la valeur d'indice i de la liste des valeurs récupérées corresponde à la caractéristique de même indice dans la liste CARA_EPX à laquelle on aurait retiré des caractéristiques non fournies.*

11) Pour brancher les caractéristiques fournies par une autre voie, il faut enrichir le dictionnaire dic_gr_cara_supp. Ces données proviennent actuellement du traitement du modèle et du mot-clé ORIENTATION de AFFE_CARA_ELEM.
12) Enfin, si besoin, ajouter la clé 'VERIF', en suivant les exemples existants, pour faire la vérification que certains mot-clés *Code_Aster* prennent bien une des valeurs attendues.

4.3 Passage de contrainte à effort

Les champs de contraintes dans EPX sont toujours exprimés en contraintes et jamais en efforts même pour les éléments de structure. Il est parfois nécessaire d'opérer une transformation des contraintes en efforts lors de l'ajout d'un nouvel élément. Il a été précisé en 4.1 à l'item 6, d'ajouter la clé 'MC_CARA' aux informations sur l'élément. Si la transformation n'est pas encore programmée pour ce mot-clé de AFFE_CARA_ELEM , il faut ajouter ce cas supplémentaire dans la méthode prep_cont_2_eff de la classe Lire_EPX() du module lire_europlexus_ops.py .

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

5 Ajout d'un nouveau chargement

Les chargements au sens de *Code_Aster* se scindent en deux directives distinctes dans EPX : LINK et CHAR. Le catalogue de traduction suit cette règle en séparant les chargements et les liaisons dans deux dictionnaires différents : cata_liais et cata_char.

5.1 Ajout d'une liaison

5.1.1 Dictionnaire des liaisons

Pour ajouter une nouvelle liaison il faut dupliquer un couple clé/valeur du dictionnaire cata_liais.

```
cata_liais = {
    'DDL_IMPO' : {
        'MOT_CLE_EPX': ['BLOQ','DEPL'],
        # le choix entre BLOQ et DEPL est fait en dur dans calc_epx_char
        # il est fait selon la présence d'une fonction ou non
        # si fonction => DEPL , sinon BLOQ
        'ASTER'     : ['DX', 'DY', 'DZ', 'DRX', 'DRY', 'DRZ'],
        'EPX'       : ['1', '2', '3', '4', '5', '6'],
        'ENTITE'    : ['GROUP_MA', 'GROUP_NO'],
        'NB_CLE_MAX' : {'BLOQ' : 6,
                        'DEPL' : 1,
                       },
        'VALE_IMPO' : {'BLOQ' : 0.,
                       'DEPL' : None,
                       },
        'FONC_MULT' : {'BLOQ' : False,
                       'DEPL' : True,
                       },
                   },
    'RELA_CINE_BP': {
        'MOT_CLE_EPX': ['LCAB'],
        'ASTER': ['CABLE_BP'],
        'EPX': None,
        'MOT_CLE_VERIF': ['SIGM_BPEL', 'RELA_CINE', 'TYPE_EPX'],
        'VALE_VERIF': ['NON', 'OUI', ['ADHE', 'GLIS', 'FROT']],
        'FONC_MULT': False,
    },
    'LIAISON_MAIL' : {
        'MOT_CLE_EPX': ['RELA'],
        'FONC_MULT': False,
    },
}
```

5.1.2 Étapes

1) Remplacer le nom de la clé par le nom du mot-clé de AFFE_CHAR_MECA à ajouter.
2) Indiquer dans 'MOT_CLE_EPX' le nom de la ou les liaisons correspondantes dans EPX. La valeur doit être une liste, le plus souvent constituée d'un seul élément. Plusieurs éléments présents dans la liste indiquent que le mot-clé peut correspondre à plusieurs liaisons dans EPX. La marche à suivre dans ce cas est décrite en 5.3 .
   Quand 'MOT_CLE_EPX' prend la valeur RELA, comme pour LIAISON_MAIL, CALC_EUROPLEXUS va traduire la liste de relations contenue dans une table attachée au concept char_meca. On ne suit pas ensuite les autres points (sauf le point 8). Un branchement sera par contre nécessaire dans AFFE_CHAR_MECA pour que la table contenant les relations soit créée (voir la routine bibfor/load/calirc.F90).
3) Indiquer dans 'ASTER' la liste des mots-clés à traduire.
4) Indiquer dans 'EPX' la liste des traductions de ces mots-clés. Indiquer None si la syntaxe EPX de la liaison ne permet pas de rentrer dans ce cadre (attention ce n'est possible que si la liste 'ASTER' n'a qu'un élément).
5) Indiquer dans 'ENTITE' les mots-clés (parmi GROUP_MA et GROUP_NO) pouvant décrire l'ensemble sur lequel s'applique la liaison. Ne pas indiquer ce mot-clé si les syntaxes *Code_Aster* et/ou EPX ne rentrent pas dans ce cadre.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

6) Si la liste 'ASTER' est supérieure à 1, indiquer dans 'NB_CLE_MAX' le nombre de mots-clés de cette liste pouvant être présents dans une même occurrence de la liaison. Si ce nombre est 1, il n'est pas nécessaire d'indiquer cette clé, c'est la valeur par défaut.
7) Si les valeurs prises par les mots-clés de la liste 'ASTER' sont imposées, indiquer cette valeur dans 'VALE_IMPO'.
8) Indiquer dans 'FONC_MULT' si la liaison doit être accompagnée d'une fonction multiplicatrice (True/False).
9) Indiquer dans 'MOT_CLE_VERIF' les mots-clés dont on doit seulement vérifier la valeur, s'il en existe.
10) Indiquer dans 'VALE_VERIF' au même indice les valeurs que doivent prendre les mots-clés.
11) Chaque mot-clé présent dans l'occurrence de la liaison *Code_Aster* doit se trouver dans une les listes 'ASTER' , 'ENTITE' ou 'MOT_CLE_VERIF'. Cela empêche l'utilisateur d'utiliser des mots-clés non pris en compte sans qu'il en soit averti.
12) Si besoin, indiquer dans 'COEF_MULT' le coefficient par lequel doivent être multipliées les valeurs issues de *Code_Aster* lors de la traduction dans EPX.

5.2 Ajout d'un chargement

5.2.1 Dictionnaire des chargements

Pour ajouter un nouveau chargement, il faut dupliquer un couple clé/valeur du dictionnaire cata_char. On donne ici une clé de ce dictionnaire :

```
cata_charge = {
    'FORCE_COQUE' : {
        'TYPE_CHAR'  : 'FACTO',
        'MOT_CLE_EPX': ['PRES COQU'],
        'ASTER'      : ['PRES'],
        'EPX'        : None,
        'COEF_MULT'  : -1,
        'ENTITE'     : ['GROUP_MA'],
                  },
              }
```

5.2.2 Étapes

Les étapes sont exactement les mêmes que pour une liaison à deux exceptions près :

1) Indiquer dans 'TYPE_CHAR', le type EPX du chargement (FACTO ou CONST). Dans le cas FACTO, le chargement doit être accompagné d'une fonction multiplicatrice, mais pas dans le cas CONST.
2) Ne pas donner le mot-clé 'FONC_MULT' puisqu'il est sous-entendu par le type de chargement.

5.3 Traitement des choix multiples

Il peut arriver qu'un chargement ou une liaison de *Code_Aster* puisse correspondre à plusieurs chargements ou liaisons dans EPX.

Tout d'abord il faut identifier ce qui va déterminer quelle possibilité est retenue. L'analyse de ce critère et le choix qui en découle doivent être programmés en dur dans le module calc_epx_char.py dans la routine export_charge après la liste de code :

```
if len(mot_cle_epx) > 1:
```

Le dictionnaire doit ensuite être adapté à ces multiples possibilités avec la logique suivante. Chaque clé dont la valeur n'est pas identique pour chacun des cas doit être un dictionnaire dont les clés sont les différents noms des chargements ou liaisons possibles et dont les valeurs sont les valeurs souhaitées pour la clé en question. La liaison DDL_IMPO en est une illustration.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)