Définition d'un champ de contraintes et d'un champ de variables internes initiaux

Résumé :

On explique comment fabriquer deux des champs constituant l'état initial d'un calcul non-linéaire (STAT_NON_LINE) : le champ de contraintes et le champ de variables internes.

- les composantes du champ de contraintes ont une forme "analytique" (par exemple : état d'un sol soumis au "poids des terres"),
- les composantes du champ de variables internes sont des constantes non nulles.

Dans les deux cas, la solution consiste à enchaîner un certain nombre de commandes CREA_CHAMP.

Pour le champ de contraintes, la difficulté consiste à évaluer les "formules analytiques" (OPERATION='EVAL'). Pour le champ de variables internes, la difficulté provient du fait que la grandeur associée aux variables internes (VARI_R) a un nombre a priori indéterminé de composantes : 'V1', 'V2', ...

Les solutions proposées sont mises en œuvre dans le test ZZZZ130A.

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

1 Définition du champ de contraintes analytique

On suppose que le modèle contient des éléments finis de milieu continu (MODELISATION='3D').
On veut qu'en chaque point de Gauss, les composantes des contraintes aient les expressions suivantes :

```
SIZZ = RHO*G*Z
SIXX = SIYY = KP*SIZZ
```

où :
RHO : masse volumique
G : accélération de la pesanteur
Z : 3éme coordonnée d'espace
KP : coefficient de "poussée" des sols

La solution proposée consiste à :

1) définir trois fonctions "formules" correspondant à SIXX, SIYY et SIZZ,
2) constituer un champ dont les composantes sont les fonctions précédentes,
3) évaluer les formules du champ en lui fournissant le champ de géométrie nécessaire à leur évaluation.

1.1 Étape 1 : définir les formules

```
RHO=1000.
G=10.
KP=3.

SIZZ = FORMULE(REEL="""(REEL:Z) = RHO*G*Z   """)
SIXX = FORMULE(REEL="""(REEL:Z) = KP*SIZZ(Z)   """)
SIYY = FORMULE(REEL="""(REEL:Z) = KP*SIZZ(Z)   """)
```

1.2 Étape 2 : créer le champ de formules SIG1

```
SIG1=CREA_CHAMP(OPERATION='AFFE', TYPE_CHAM='ELGA_NEUT_F',
                MODELE=MO,  PROL_ZERO='OUI',
     AFFE=_F( TOUT = 'OUI', NOM_CMP = ('X1','X2','X3',),
                            VALE_F  = (SIXX,SIYY,SIZZ,)))
```

Remarques

- *le champ SIG1 que l'on crée est un cham_elem aux points de Gauss (ELGA),*
- *les seuls champs pouvant avoir des composantes de type "fonctions" sont les champs de la grandeur NEUT_F . Il faudra donc se rappeler que la composante 'X1' de SIG1 est en réalité 'SIXX' , etc. ...,*
- *le mot clé PROL_ZERO='OUI' est obligatoire car pour tous les types d'élément, les cham_elem_NEUT_R ont actuellement 6 composantes : 'X1' , 'X2' , ..., 'X6' . Il faut donc accepter de "prolonger" par zéro le champ sur les 3 composantes non affectées. Le prolongement par "zéro" pour un champ dont les composantes sont des textes (noms des fonctions) consiste à affecter la chaîne " " à chaque composante absente du champ. Attention donc, il ne s'agit pas d'une fonction nulle. On peut le constater en utilisant INFO=2 pour imprimer le champ SIG1 .*

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

1.3 Étape 3 : évaluer les formules du champ SIG1

Le champ SIG1 est un champ connu aux points de Gauss des éléments du modèle. En chaque point, on va vouloir évaluer les fonctions SIXX, SIYY et SIZZ. Pour cela, il faut disposer des valeurs de toutes les variables apparaissant dans les fonctions (ici Z). Ces variables doivent être connues sur les mêmes points que le champ de fonctions. Il faut donc disposer d'un champ contenant la géométrie des points de Gauss (cham_elem_GEOM_R / ELGA).

Ce champ de géométrie des points de Gauss (CHXG) est obtenu à partir du maillage (MA) par les 2 commandes suivantes :

```
CHXN=CREA_CHAMP(OPERATION='EXTR', TYPE_CHAM='NOEU_GEOM_R',
      NOM_CHAM='GEOMETRIE', MAILLAGE=MA )

CHXG=CREA_CHAMP(OPERATION='DISC', TYPE_CHAM='ELGA_GEOM_R',
      MODELE=MO,   CHAM_GD=CHXN
```

La première commande extrait le champ de géométrie (aux nœuds) du maillage. La seconde transforme le champ de géométrie aux nœuds en un champ de géométrie aux points de Gauss en utilisant les fonctions de forme des éléments finis du modèle.

On peut alors évaluer les fonctions grâce à l'opérateur CREA_CHAMP / OPERATION='EVAL' :

```
SIG2=CREA_CHAMP(OPERATION='EVAL', TYPE_CHAM='ELGA_NEUT_R',
      MODELE=MO, CHAM_F=SIG1,  CHAM_PARA=( CHXG,)   )
```

Le champ (SIG2) obtenu par évaluation d'un champ de la grandeur NEUT_F est un champ de la grandeur NEUT_R dont les composantes ont les mêmes noms que les composantes de NEUT_F : 'X1', 'X2', ..., 'X6'.

Attention :

Les composantes 'X4', 'X5', 'X6' sont indéfinies (en réalité elles contiennent le réel le plus grand possible), car elles correspondent à une fonction inexistante.

Il reste encore à changer la grandeur du champ SIG2 (NEUT_R -> SIEF_R) pour terminer la fabrication de notre champ de contraintes analytique :

```
SIGINI=CREA_CHAMP(OPERATION='ASSE', TYPE_CHAM='ELGA_SIEF_R',
      MODELE=MO,  PROL_ZERO='OUI',
      ASSE=_F( TOUT = 'OUI',   CHAM_GD = SIG2,
            NOM_CMP = ('X1',  'X2',  'X3',),
            NOM_CMP_RESU = ('SIXX','SIYY','SIZZ',)))
```

Remarques :

- *seules les composantes 'X1', 'X2' et 'X3' du champ SIG2 sont recopiées dans cette opération pour donner les composantes 'SIXX', 'SIYY', 'SIZZ' du champ SIGINI . Ce champ de contraintes doit contenir également les composantes liées aux cisaillements ('SIXY', 'SIYZ', 'SIXZ'). Pour les obtenir (avec une valeur nulle), il faut utiliser le prolongement par zéro (PROL_ZERO='OUI'),*
- *les manipulations faites pour obtenir les composantes de cisaillement nulles, auraient été plus simples si on avait explicitement affecté sur ces 3 composantes une fonction nulle. On n'aurait pas eu à "jouer" avec les prolongements. Mais on aurait bénéficié de la coïncidence que les grandeurs SIEF_R et NEUT_R ont toutes les deux 6 composantes pour les cham_elem (ELGA) sur les éléments du modèle.*

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

2 Définition du champ de variables internes non nul

2.1 Problème

On veut créer un champ de variables internes initiales pour la commande STAT_NON_LINE. Ce champ ne doit pas être nul partout. Plus précisément, on veut :

```
STAT_NON_LINE :
    COMPORTEMENT=(_F( GROUP_MA='MASSIF',RELATION = 'CJS'),
                  _F( GROUP_MA='BETON', RELATION = 'ENDO_LOCAL'),),
```

pour la relation de comportement 'CJS' (16 variables internes), on veut affecter :
V1 = 1.0 et V9 = 9.0

pour la relation de comportement 'ENDO_LOCAL' (2 variables internes), on veut affecter :
V2 = 2.0

2.2 1ère méthode

L'opérateur à utiliser est CREA_CHAMP / OPERATION='AFFE'. Il permet d'affecter (par maille ou GROUP_MA) les valeurs que l'on souhaite. La difficulté provient du fait que la grandeur associée aux variables internes (VARI_R) est différente des autres : on ne sait pas a priori quelles sont ses composantes. D'ailleurs le nom de ses composantes traduit cette ignorance : 'V1', 'V2', ...
Selon le comportement que choisira l'utilisateur dans STAT_NON_LINE, le nombre de variables internes change. Dans notre exemple, le comportement 'CJS' nécessite 16 variables alors que 'ENDO_LOCAL' n'en utilise que 2.

L'opération d'affectation se fait de la façon suivante :

```
VAIN1=CREA_CHAMP(OPERATION='AFFE', TYPE_CHAM='ELGA_VARI_R',
      MODELE=MO, PROL_ZERO='OUI',
      AFFE=(
        _F( GROUP_MA= 'BETON', NOM_CMP= 'V2', VALE = 2.),
        _F( GROUP_MA= 'MASSIF',
            NOM_CMP= ('V1','V9','V16',),
            VALE =     (1.,  9.,   0.,)),
   )
)
```

Remarques importantes :

- *Le mot clé PROL_ZERO='OUI' permet de n'affecter que les composantes non nulles. Mais comme la commande n'a pas connaissance du nombre de variables internes portées par les mailles, elle se base sur le numéro affecté le plus élevé.*
  Dans l'exemple ci-dessus, sur le groupe 'MASSIF', il est important d'affecter 'V16' (ici à 0.) pour que le champ possède 16 composantes.
- *Il est mportant pour le calcul non-linéaire qui va suivre que le champ de variables internes soit cohérent avec les comportements que l'on choisira. I*
  Ici, il faut que les mailles du groupe 'BETON' aient 2 variables internes (et seulement 2) et celles du groupe 'MASSIF' en aient 16.

Attention :

Si le modèle comporte d'autres types de comportement (pour lesquels on ne souhaite pas initialiser le champ avec des valeurs non nulles), il faut également leur affecter explicitement des valeurs nulles. Cet inconvénient (avoir à connaître TOUS les comportements utilisés et leur nombre de variables internes) peut être levé avec la 2ème méthode ci-dessous (mais c'est plus compliqué).

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

2.3 2ème méthode

Cette méthode (plus compliquée) permet de n'affecter explicitement que les mailles qui possèdent des composantes non nulles.

Le problème est d'obtenir un champ contenant le bon nombre de variables internes pour chaque maille en fonction du comportement qui lui sera affecté dans STAT_NON_LINE. Pour résoudre ce problème, on va réaliser un calcul non-linéaire fictif (avec les comportements réels). Le champ de variables internes produit sera alors un bon "modèle" de champ.

On fera donc :

1) calcul non-linéaire fictif => UBID
2) extraction du champ de variables internes (VBID) du résultat UBID
3) affectation des valeurs non nulles dans le champ VAIN2
4) mise à zéro de VBID + surcharge des valeurs de VAIN2 pour produire le résultat VAIN22

2.3.1 Calcul non-linéaire fictif

```
BETON=DEFI_MATERIAU( ELAS=_F( E = 20000., NU = 0.),
                ECRO_LINE=_F(  SY = 6., D_SIGM_EPSI = -10000.)  )

 MASSIF=DEFI_MATERIAU(    ELAS=_F(  E = 35.E3, NU = 0.15),
     CJS=_F(  BETA_CJS = -0.55, GAMMA_CJS = 0.82, PA = -100.0,
                 RM = 0.289, N_CJS = 0.6, KP = 25.5E3, RC = 0.265, A_CJS =
0.25, ) )

CHMAT=AFFE_MATERIAU(  MAILLAGE=MA,AFFE=(
       _F( GROUP_MA = 'MASSIF', MATER = MASSIF),
       _F( GROUP_MA = 'BETON', MATER = BETON), ))

TEMPS1=DEFI_LIST_REEL(  VALE=(0.,1.) )
CHAR_U1=AFFE_CHAR_MECA( MODELE=MO,
         DDL_IMPO=_F( NOEUD = ( 'N1','N2','N3', ), DX=0.,DY=0.,DZ=0.) )

UBID=STAT_NON_LINE( MODELE=MO, CHAM_MATER=CHMAT,
       EXCIT= _F( CHARGE = CHAR_U1,),
       COMPORTEMENT=(_F( GROUP_MA='MASSIF',RELATION = 'CJS'),
                     _F( GROUP_MA='BETON', RELATION = 'ENDO_LOCAL'),),
       NEWTON=_F(  MATRICE = 'ELASTIQUE'),
       CONVERGENCE=_F( ARRET = 'NON',  # pour continuer sans convergence
                ITER_GLOB_MAXI = 1, ITER_INTE_MAXI = 1),
      INCREMENT=_F( LIST_INST = TEMPS1),
              )
```


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

2.3.2 Récupération du champ de variables internes "modèle"

```
VABID=CREA_CHAMP(OPERATION='EXTR', TYPE_CHAM='ELGA_VARI_R', INFO=1,
      NOM_CHAM='VARI_ELGA', RESULTAT=UBID, NUME_ORDRE=1,)
```

Remarque :

VABID n'est pas nul.

2.3.3 Affectation des valeurs non nulles dans une carte de NEUT_R

```
VAIN2=CREA_CHAMP(OPERATION='AFFE', TYPE_CHAM='CART_NEUT_R', MODELE=MO,
     AFFE=(
           _F( GROUP_MA= 'BETON', NOM_CMP= ('X2',), VALE = (2.,)),
           _F( GROUP_MA= 'MASSIF', NOM_CMP= ('X1','X9',), VALE = (1.,9.,)),
         )
     )
```

2.3.4 Mise à zéro du champ de variables internes "modèle" et surcharge des valeurs non nulles

```
VAIN22=CREA_CHAMP(OPERATION='ASSE', TYPE_CHAM='ELGA_VARI_R', MODELE=MO,
      # mise à zéro :
      ASSE=(_F(TOUT= = 'OUI', CHAM_GD = VABID, CUMUL='OUI', COEF_R=0.),
      # surcharge des valeurs non nulles :
            _F(GROUP_MA= 'BETON', CHAM_GD = VAIN2, CUMUL='OUI', COEF_R=1.,
               NOM_CMP=('X2',),NOM_CMP_RESU=('V2',),),
            _F(GROUP_MA= 'MASSIF', CHAM_GD = VAIN2, CUMUL='OUI', COEF_R=1.,
               NOM_CMP=('X1','X3'),NOM_CMP_RESU=('V1', 'V9',),),
          )
      )
```

Remarque ;

Pour la mise à zéro et la surcharge des valeurs non nulles, on utilise les mots clés CUMUL='OUI' et COEF_R=0 .

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