Opérateur CALC_IFS_DNL

1 But

L'objet de cette commande est de permettre les calculs fluides-structures couplés en régime transitoire non-linéaire. Pour cela, on vient coupler *Code_Aster*, pour la partie structure, à *Code_Saturne*, pour le domaine fluide, *via* le superviseur YACS de Salomé.

La méthode de couplage est de type partitionné Neuman-Dirichlet. Pour résoudre le problème structure, on se base sur l'opérateur DYNA_NON_LINE, dont on reprend très largement la syntaxe.

Table des matières

1 But....1
2 Syntaxe....3
3 Principe de fonctionnement....6
4 Définition de la discrétisation temporelle....6
4.1 Mot-clé PAS_INIT....6
5 Définition de l'interface fluide-structure....7
5.1.1 Opérande GROUP_MA_IFS....7
5.1.2 Opérande NOM_CMP_IFS....7
5.1.3 Opérandes UNITE_NOEUD et UNITE_ELEM....7

2 Syntaxe

```
resu  [evol_noli] = CALC_IFS_DNL  (
   ◊  reuse           =  dynanl,                        [evol_noli]
   ♦  MODELE          =  mo,                            [modele]
   ♦  CHAM_MATER      =  chmat,                         [cham_mater]
   ◊  MODE_STAT       =  modestat,                      [mode_meca]
   ◊  CARA_ELEM       =  carac,                         [cara_elem]
   ◊  MASS_DIAG       =  /'OUI',
                         /'NON',

   ◊  EXCIT           =_F (
                  ◊  TYPE_CHARGE  =  /'FIXE_CSTE',     [DEFAUT]
                                     /'SUIV',
                                     /'DIDI',
                  ◊  CHARGE       =  chi,              [char_meca]
                                                       [char_cine_meca]
                  ◊  /  FONC_MULT =  fi ,              [fonction]
                     /  DEPL      =  depl,             [fonction]
                        VITE      =  vite,             [fonction]
                        ACCE      =  acce,             [fonction]
                  ◊  MULT_APPUI   =  /'NON',           [DEFAUT]
                                     /'OUI',
                  ◊  DIRECTION    =  (d1, d2, d3),     [l_R]
                  ◊  NOEUD        =  lno,              [l_noeud]
                  ◊  GROUP_NO     =  lgrno,            [l_gr_noeud]
               ),

   ◊  SOUS_STRUC      =_F ( voir le document [U4.51.03] ),

   ◊  AMOR_RAYL_RIGI  =  /'TANGENTE',                  [DEFAUT]
                         /'ELASTIQUE',

   ◊  AMOR_MODAL      =_F (
                  ♦  MODE_MECA    =  mode,             [mode_meca]
                  ♦  /AMOR_REDUIT =  l_amor,           [l_R]
                     /LIST_AMOR   =  lisamor           [listr8]
                  ◊  NB_MODE      =  /nbmode,          [I]
                                     /9999,            [DEFAUT]
                  ◊  REAC_VITE    =  /'OUI',           [DEFAUT]
                                     /'NON',
               ),

   ♦    |  COMP_INCR   =_F ( voir le document [U4.51.11] ),
        |  COMP_ELAS   =_F ( voir le document [U4.51.11] ),

   ◊  ETAT_INIT       =_F ( voir le document [U4.51.03]
                     |  VITE         =  vite,          [cham_no_DEPL_R]
                     |  ACCE         =  acce,          [cham_no_DEPL_R]
               ),

   ◊  EXCIT_GENE      =_F (
                  ◊  FONC_MULT    =  fomult,      [fonction_sdaster]
                  ♦  VECT_GENE    =  vecgen,         [vect_asse_gene]
               ),

   ◊  NEWTON          =_F ( voir le document [U4.51.03] ),

   ◊  RECH_LINEAIRE   =_F ( voir le document [U4.51.03] ),
```

```
        ◊  SOLVEUR      =  _F ( voir le document [U4.50.01] ),

        ◊  CONVERGENCE  =  _F ( voir le document [U4.51.03] ),
        ◊  MODE_VIBR =_F(
           ◊  NB_FREQ          =  / 3,                          [DEFAUT]
                                  / nbfreq,                     [I]
           ◊  MATR_RIGI        =  / 'ELASTIQUE',                [DEFAUT]
                                  / 'TANGENTE',
                                  / 'SECANTE',
           ◊  BANDE            =     intba  ,                   [listr8]
           ◊  INST_CALCUL      =  / 'LISTE_ARCHIVAGE',          [DEFAUT]
                                  / 'TOUT_PAS',
           ),

        ◊  CRIT_FLAMB   = _F ( voir le document [U4.51.03] ),
        ◊  SENSIBILITE  = _F ( voir le document [U4.50.02] ),

        ◊  ARCHIVAGE    = _F ( voir le document [U4.51.03]),

        ♦  SCHEMA_TEMPS =_F (
                         ♦ SCHEMA =  /  'NEWMARK',
                                     /  'HHT',
                                     /  'THETA_METHODE',
                                     /  'DIFF_CENT',
                                     /  'TCHAMWA',
                                     /  'KRENK',                        ◊
   COEF_MASS_SHIFT  =  /  0.,       [DEFAUT]
                                                 /  coef,     [R]

{ Si SCHEMA      =  'NEWMARK'
                        ◊     BETA   =  /  0.25,              [DEFAUT]
                                        /  beta,              [R]
                        ◊     GAMMA  =  /  0.5,               [DEFAUT]
                                        /  gamm,              [R]
}
{ Si SCHEMA      =  'HHT'
                     ◊     ALPHA  =  /  -0.3,                [DEFAUT]
                                     /  alph,                [R]
                     ◊     MODI_EQUI =    /  'OUI',
                                          /  'NON',          [DEFAUT]
}
{ Si SCHEMA      =  'THETA_METHODE'
                        ◊     THETA  =  /  1.,                [DEFAUT]
                                        /  theta,             [R]
}
{ Si SCHEMA      =  'TCHAMWA'
                     ◊     PHI    =  /  1.05                 [DEFAUT]
                                     /  phi,                 [R]
}
{ Si SCHEMA      =  'KRENK'
                     ◊     KAPPA  =  /  1.                   [DEFAUT]
                                     /  kappa,               [R]
}
{ Si SCHEMA      =  /  'NEWMARK',
                    /  'HHT',
                    /  'THETA_METHODE',
                    ♦  FORMULATION  =  /  'DEPLACEMENT',
                                       /  'VITESSE',
```

```
                                          /   'ACCELERATION',
      }
      { Si SCHEMA      =   /   'DIFF_CENT',
                           /   'TCHAMWA',
                               ♦   FORMULATION  =      'ACCELERATION',
                               ◊    STOP_CFL    =   /   'OUI',        [DEFAUT]
                                                    /   'NON',
      }, ),
```

```
◊  OBSERVATION  = _F   ( voir le document [U4.51.03]),
◊  AFFICHAGE    = _F   ( voir le document [U4.51.03]),

◊  PROJ_MODAL =_F   (
           ♦  MODE_MECA =  mode,                [mode_meca]
           ◊  NB_MODE      =  /  nbmode,        [I]
                              /  9999,          [DEFAUT]
           ◊  /MASS_GENE   =  massgen           [matr_asse_gene_R]
              RIGI_GENE    =  rigigen           [matr_asse_gene_R]
              /  AMOR_GENE =  amorgen           [matr_asse_gene_R]
              ◊     DEPL_INIT_GENE = deplgen,   [ vect_asse_gene ]
              ◊     VITE_INIT_GENE = vitegen,   [ vect_asse_gene ]
              ◊     ACCE_INIT_GENE = accegen,   [ vect_asse_gene ]
                ) ,

◊  INFO      =      / 1,                        [DEFAUT]
                    / 2 ,

◊  TITRE = tx    ,                              [Kn]
```

Syntaxe spécifique à l'IFS :

```
♦  PAS_INIT         =     pdtinit               [R]
♦  GROUP_MA_IFS     =    lgrmaifs,              [l_gr_maille]
♦  NOM_CMP_IFS      =    lcompifs,              [l_Kn]
◊  UNITE_NOEUD      =     /  ulnoeud,           [I]
                          /  81,                [DEFAUT]
◊  UNITE_ELEM       =     /  ulelem,            [I]
                          /  82,                [DEFAUT]
```

)

3 Principe de fonctionnement

La méthode de couplage est de type partitionné Neuman-Dirichlet. Pour résoudre le problème structure, on se base sur l'opérateur DYNA_NON_LINE de *Code_Aster.* Le domaine fluide sera résolu avec *Code_Saturne*. Les maillages à l'interface n'étant pas obligatoirement conformes, il faut utiliser un opérateur de projection de champs : on a choisit d'utiliser PROJ_CHAMP. Qui va gérer toutes les étapes de projection (*Code_Saturne* n'aura donc aucune projection à faire en interne).

On peut résumer l'algorithme de couplage ainsi :
à chaque pas de temps, *Code_Aster* envoie les déplacement et vitesse calculés à *Code_Saturne* , qui en déduit une déformation de la grille fluide et résout le problème fluide dessus (en description ALE). Les efforts fluide à la parois sont ensuite envoyé vers *Code_Aster* qui peut alors résoudre le nouveau problème structure sur un pas.

Sous cette forme simple l'algorithme est explicite et cela impose un pas de temps assez petit pour des raisons de stabilité conditionnelle [R5.05.05]. En pratique ce n'est pas forcément très pénalisant car la résolution du problème fluide réclame souvent un pas de temps assez petit.
Il est possible de définir une version implicitée de la méthode de couplage. Il suffit, à chaque pas de temps d'introduire une processus itératif de type point fixe. Cela permet d'utiliser un pas de temps plus grand, mais avec un surcoût de calcul lié aux itérations de point fixe.

Toutes les données échangées (elles sont scalaires ou vectorielles) entre les deux codes de calcul passent par des appels YACS. L'utilisation du couplage IFS passe donc obligatoirement par Salomé qui va piloter les deux codes : *Code_Aster* et *Code_Saturne*. On ne peut donc mener ce type de calcul en utilisant classiquement les interfaces de lancement de *Code_Aster* : astk ou as_run.
Cette documentation se borne à décrire l'utilisation au sens *Code_Aster* uniquement.

La résolution de la partie structure se fait grâce à l'opérateur DYNA_NON_LINE, ce qui explique que la syntaxe de CALC_IFS_DNL soit en très grande partie identique. On revoit donc à la documentation U4.53.01 pour tous les mot-clés communs à DYNA_NON_LINE.

Les seules différences de syntaxe, qui sont détaillées dans cette documentation, sont liées à :
- la gestion du temps : on n'utilise pas le mot-clé facteur INCREMENT, car le pilotage en temps est géré par le coupleur lui-même
- la définition des caractéristiques de l'interface fluide-structure.

CALC_IFS_DNL produit un concept de type evol_noli habituel.

4 Définition de la discrétisation temporelle

Le pilotage en temps est en fait déporté hors de *Code_Aster*. Plus précisément, le coupleur va évaluer à chaque instant de calcul le pas courant et le fournir aux deux codes que sont *Code_Aster* et *Code_Saturne*. En pratique, chacun de ces codes ne fournit qu'un pas de temps initial qui permet au coupleur d'évaluer le premier pas de temps.

De même, les informations de temps initial et temps final d'étude sont défini au niveau du coupleur lui-même et pas dans le fichier de commande.

Code_Aster récupérera toutes ces informations (instant initial, instant final, pas courant) *via* YACS.

4.1 Mot-clé PAS_INIT

Définit le pas de temps initial pour le couplage IFS, au sens du pas de temps pertinent pour le calcul structure seul. *Code_Saturne* définit de même son propre pas de temps initial et le coupleur va alors renvoyer aux deux codes le pas de temps initial qui sera réellement utilisé pour la résolution couplée. En pratique, ce pas de temps couplé sera le minimum des deux pas de temps venant des codes, afin de respecter les conditions de stabilité et qualité de la solution sur chacun des deux domaines.

Si l'on utilise un schéma en temps explicite dans *Code_Aster*, alors, bien évidemment, le pas de temps initial devra respecter la condition de Courant (ou CFL, [U4.53.01] et [R5.05.05]).

5 Définition de l'interface fluide-structure

L'utilisateur doit spécifier l'interface fluide-structure. Il convient de rappeler qu'à cette interface, les maillages fluides et solides ne sont pas forcément conformes. De plus, *Code_Aster* gérant toutes les étapes de projection entre le fluide et le solide, il faut lui donner les informations du maillage fluide.

5.1.1 Opérande GROUP_MA_IFS

Cette opérande permet de définir le groupe de mailles du maillage solide qui est à l'interface fluide-structure.

5.1.2 Opérande NOM_CMP_IFS

On spécifie quelles composantes de l'effort seront transmises à l'interface fluide-structure, dans le repère absolu.

Par exemple :
```
NOM_CMP_IFS = ('FX', 'FY' , 'FZ' ),
```
Pour avoir transmission complète des efforts en 3D.
Si on souhaite ne transmettre que certaines composantes, il suffit d'exclure les composantes inutilisées. On peut ainsi réaliser des conditions de glissement à la paroi.

5.1.3 Opérandes UNITE_NOEUD et UNITE_ELEM

Ces opérandes permettent de définir les unité logiques des fichiers contenant les maillages correspondants à l'interface fluide-structure issue du maillage fluide. La résolution par Code_Saturne se faisant en volumes finis, il est nécessaire de définir deux maillage distincts pour les projections de champs à échanger.

Dans le couplage Neuman-Dirichlet, le code structure fournit au code fluide les déplacements et vitesse à l'interface. Ce sont donc des données aux nœuds du maillage solide que l'on projette sur les nœuds du maillage fluide. Le maillage fluide de l'interface est récupéré de *Code_Saturne*, *via* YACS et sera écrit, au format de maillage Aster dans le fichier ayant l'unité logique UNITE_NOEUD (qui vaut 81 par défaut). On peut ainsi aussi récupérer ce maillage en post-traitement si besoin est.

La deuxième étape du couplage se fait dans l'autre sens : le code fluide fournit au code structure les efforts à l'interface (suivant les composantes données avec NOM_CMP_IFS). Plus précisément, *Code_Saturne* étant un code en volumes finis, les efforts surfaciques calculés sont constants par face et ce que *Code_Aster* récupère est en fait les résultantes par face des éléments, exprimées aux nœuds milieux. Pour que *Code_Aster* puisse projeter sur le maillage structure, il faut donc disposer du maillage des nœuds milieux du maillage fluide à l'interface. Ce maillage est aussi récupéré via YACS et on l'écrit au format Aster dans l'unité logique UNITE_ELEM (qui vaut 82 par défaut).