Opérateur DEFI_BASE_REDUITE

Le but de l'opérateur est de construire la base réduite à partir d'un calcul non-linéaire (thermique ou mécanique) ou d'un calcul linéaire paramétrique.
Il existe deux méthodes :

- Pour les problèmes non-linéaires, l'opérateur s'appuie sur une sd résultat de type `evol_ther` ou `evol_noli` et réalise une décomposition aux valeurs singulière (SVD) sur le transitoire ou une stratégie de type POD incrémentale ;
- Pour les problèmes linéaires paramétriques, on utilise un algorithme de type glouton. Dans ce cas o n définit le système linéaire à résoudre.

Deux types de bases peuvent être produites :

- les bases dites « primales » : elles s'appuient sur les champs de déplacements pour la mécanique et sur les champs de températures pour la thermique ;
- les bases dites « duales » : elles s'appuient sur les champs de contraintes pour la mécanique et sur les champs de flux pour la thermique.

L'opérateur produit un concept de type `mode_empi` .

Table des Matières

1 Syntaxe....3
2 Opérandes....5
2.1 Opérande BASE....5
2.2 Opérande OPERATION....5
2.3 Opérandes pour les stratégies POD....5
2.3.1 Opérande RESULTAT....5
2.3.2 Opérande NOM_CHAM....5
2.3.3 Opérande TYPE_BASE....5
2.3.4 Opérandes AXE et SECTION....6
2.3.5 Opérandes NB_MODE / TOLE_SVD....6
2.3.6 Opérande TOLE....6
2.4 Opérandes pour les stratégies GLOUTON....6
2.4.1 Opérandes MATR_ASSE et VECTEUR....7
2.4.2 Opérandes VARI_PARA , NB_VARI_COEF et TYPE_VARI_COEF....7
2.4.3 Opérande SOLVEUR....8
3 Exemples d'utilisation....9

1 Syntaxe

```
base = DEFI_BASE_REDUITE (

             ◊  BASE      =   base ,                          [base_empi]

 # Type d'opération

             ◊  OPERATION =   |' POD ' ,                       [ DEFAUT ]
                              |'POD_INCR',
                              |'GLOUTON',

# Si OPERATION='POD_INCR'
             ◊  TOLE      =  /tole_incr,                      [R]
                             /1.0E-10,                        [DEFAUT]

# Si OPERATION='POD_INCR' ou 'POD'

# options pour les résultats produits par STAT_NON_LINE

             ♦  RESULTAT  =  resu,                            [evol_noli]

             ♦  NOM_CHAM  =  |'DEPL'
                             |'SIEF_NOEU'

# options pour les résultats produits par THER_NON_LINE

             ♦  RESULTAT  =  resu,                            [evol_ther]

             ♦  NOM_CHAM  =  |'TEMP'
                             |'FLUX_NOEU'

             ◊  TYPE_BASE =  |'3D',                           [DEFAUT]
                             |'LINEIQUE',

# options pour TYPE_BASE = 'LINEIQUE'

             ♦  AXE       =  |'OX',
                             |'OY',
                             |'OZ'

             ♦  SECTION   =  l_grno,                          [l_gr_noeud]

# options de sélection du nombre de modes

             ◊  TOLE_SVD  =  /tole,                           [R]
                             /1.0E-6,                         [DEFAUT]
             ◊  NB_MODE   =  nbmode,                          [I]
```

```
      # Si OPERATION='GLOUTON'

                    ♦  NB_VARI_COEF    =   nbvaricoef,          [ I ]
                    ◊   TYPE_VARI_COEF  =   |' DIRECT ',         [DEFAUT]

                    ◊   VARI_PARA          =   _F(
                       ♦   NOM_PARA     =   nompara,            [ para_fonc ]
                       ♦   VALE_PARA       =   valepara,              [ R ]
                       ♦    VALE_INIT      =    valeinit              [ R ]
                                           ),

                    ◊   MATR_ASSE          =   _F(
                       ♦   MATRICE      =   matrice ,            / [ matr_ asse
      _DEPL_R]
                                                                 /[matr_asse_DEPL_C]
                        ◊   COEF_R         =   coefr ,              [ R ]
                        ◊   COEF_C         =    coefc ,             [ C ]
                       ◊    FONC_R         =   foncr ,                [fonc  tion
/formule]
                        ◊   FONC_C         =    foncc ,              [fonc tion
      /formule]
                                           ),

                   ◊   VECTEUR         =   vecteur              [cham_no]
                    ◊   COEF_R             =   coefr ,              [ R ]
                    ◊   COEF_C             =   coefc ,              [ C ]
                   ◊   SOLVEUR         =   _F(voir le document [U4.50.01]),

      # Pour toutes les opérations
                   ◊   TITRE     =   titre,                  [l_Kn]

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

2 Opérandes

L'opérateur sort une structure de données de type mode_empi (modes empiriques).

2.1 Opérande BASE

```
◊  BASE            =  base,                          [base_empi]
```

Il est possible d'enrichir une base de modes empiriques par les opérations suivantes. Dans ce cas, on la fournit ici.

2.2 Opérande OPERATION

```
◊  OPERATION    =  |' POD ' ,                          [ DEFAUT ]
                   |'POD_INCR',
                   |'GLOUTON',
```

Ce mot-clef permet de choisir la manière de calculer les modes empiriques :

- Par une POD (OPERATION='POD') : on réalise une SVD sur la matrice des snapshots contenant les résultats sur un transitoire. Cette opération est exacte (au sens de l'extraction des modes empiriques) mais peut-être potentiellement coûteuse (en CPU et en mémoire) ;
- Par une POD incrémentale (OPERATION='POD_INCR') : on construit la base empirique de manière incrémentale (voir [R5.01.50]). Cette méthode est moins précise que la POD classique mais beaucoup moins coûteuse. De plus elle permet d'enrichir une base empirique existante avec de nouveaux résultats ;
- Par une méthode glouton (OPERATION='GLOUTON', voir [R5.01.50]) sur des problèmes paramétriques.

2.3 Opérandes pour les stratégies POD

2.3.1 Opérande RESULTAT

```
♦  RESULTAT  =  resu,
```

Nom de la structure de données résultat à analyser pour générer la base réduite. Deux types de résultat possibles : evol_noli ou evol_ther.

Limitations d'usage :

- Ne fonctionne qu'en 3D (thermique et mécanique) ;
- Ne peut utiliser de conditions limites de Dirichlet par dualisation (AFFE_CHAR_MECA ou AFFE_CHAR_THER). Il faut utiliser des conditions limites de Dirichlet par élimination (AFFE_CHAR_CINE).

2.3.2 Opérande NOM_CHAM

```
♦  NOM_CHAM  =|'DEPL'
              |'SIEF_NOEU'
              |'TEMP'
              |'FLUX_NOEU'
```

On précise le type de champ de base réduite :

- 'DEPL' ou 'SIEF_NOEU' si la structure de donnée est de type evol_noli ;
- 'TEMP' ou 'FLUX_NOEU' si la structure de donnée est de type evol_ther.

2.3.3 Opérande TYPE_BASE

```
◊  TYPE_BASE =|  '3D',
```

```
                  | 'LINEIQUE',
```

On précise le type de base réduite.

Le cas linéique est spécifique à la simulation numérique du soudage. Ce cas nécessite de préciser l'axe de soudage et des informations concernant la section de la zone soudée, perpendiculaire à l'axe de soudage.

2.3.4 Opérandes AXE et SECTION

Opérandes utilisées dans le cas linéique (spécifique à la simulation numérique du soudage).

♦ AXE =|'OX',
       |'OY',
       |'OZ',

Opérande permettant de préciser l'axe du soudage.

♦ SECTION = l_grno,

Opérande permettant de préciser le groupe de nœuds contenu dans la première section du maillage de la zone soudée.

Remarque :
Ces opérandes permettent de définir une numérotation locale utilisée pour le calcul des modes réduits.

2.3.5 Opérandes NB_MODE / TOLE_SVD

◊ TOLE_SVD = tole

Désigne la tolérance retenue pour la SVD. La valeur par défaut est de 1.0E-6.

Les modes sélectionnés sont ceux dont la valeur singulière est supérieure à tole x sing_maxi où sing_maxi est la valeur singulière maximale donnée par la SVD.

Remarque : la POD incrémentale (OPERATION='POD_INCR') fait également une SVD mais pas sur la matrice des clichés. Ce paramètre est donc également utile dans ce cas.

◊ NB_MODE = nbmode

Nombre de modes retenus pour la construction de la base réduite.

L'utilisateur doit renseigner uniquement l'une des deux opérandes pour fixer le nombre de modes retenus pour la construction de la base réduite.

2.3.6 Opérande TOLE

```
◊ TOLE      =  /tole_incr,                    [R]
               /1.0E-10,                      [DEFAUT]
```

Ce paramètre permet de régler la précision de la POD incrémentale.

2.4 Opérandes pour les stratégies GLOUTON

La méthode OPERATION='GLOUTON' permet de construire une base empirique sur un problème paramétré. Le principe de base est de construire le système linéaire qui résout le problème visé et de donner la liste des paramètres qui varient. La base empirique est alors construite par un algorithme de type Glouton (greedy algorithm) auquel il faut fournir la variation des paramètres.

Le système linéaire s'écrira ainsi :

$$\sum_{i=1}^{n_m} \alpha_i(\gamma_{j=1,n_p}) \boldsymbol{M}_i = \beta \boldsymbol{V} \quad (1)$$

Il contient au maximum $n_m = 8$ matrices assemblées $\boldsymbol{M}_i$ (réelles ou complexes). Ces matrices peuvent être construites de manière classique dans code_aster (CALC_MATR_ELEM et ASSE_MATRICE par exemple). Devant chaque matrice, il y a un coefficient $\alpha_{i=1,n_m}$, qui est soit constant (réel ou complexe), soit une fonction ou une formule (réelle ou complexe) qui dépend au maximum de $\gamma_{j=1,n_p=5}$ paramètres.
Enfin, le second membre $\boldsymbol{V}$ est un simple champ nodal avec un coefficient constant β réel ou imaginaire.
Pour construire la base empirique, il faut parcourir l'esapce des paramètres ce que l'utilisateur défini dans le §2.4.2.

2.4.1 Opérandes MATR_ASSE et VECTEUR

Ces opérandes vont construire le système linéaire à résoudre.

```
◊   MATR_ASSE          =  _F(
   ♦   MATRICE      =   matrice ,          / [ matr_ asse _DEPL_R]
                                          /[matr_asse_DEPL_C]
    ◊   COEF_R          =   coefr ,              [ R ]
    ◊   COEF_C          =    coefc ,             [ C ]
   ◊    FONC_R          =   foncr ,               [fonc tion /formule]
    ◊   FONC_C          =    foncc ,              [fonc tion /formule]
                        ),
```

O n donne la liste des matrices assemblées $\boldsymbol{M}_i$ par le mot-clef facteur MATR_ASSE . Le nom de la matrice (provenant par exemple de la commande ASSE_MATRICE) est donné dans MATRICE .
On choisit ensuite le coefficient devant chaque matrice. Ce coefficient est soit constant (réel par COEF_R ou complexe par COEF_C), soit une fonction (réelle par FONC_R ou complexe par FONC_C). Dans le cas d'une fonction, elle doit dépendre des paramètres dont la liste (la variation) est donnée par le mot-clef facteur VARI_PARA (voir § 2.4.2).

```
◊   VECTEUR         =   vecteur            [cham_no]
◊    COEF_R             =   coefr ,             [ R ]
◊    COEF_C             =   coefc ,             [ C ]
```

C es mots-clefs définissent le second membre $\boldsymbol{V}$ par le mot-clef VECTEUR et le coefficient β réel (COEF_R) ou complexe (COEF_C). Le second membre est forcément constant.

2.4.2 Opérandes VARI_PARA , NB_VARI_COEF et TYPE_VARI_COEF

Ces opérandes définissent l'espace paramétrique parcouru pour construire la base empirique.

```
♦   NB_VARI_COEF    =   nbvaricoef,        [ I ]
 ◊   TYPE_VARI_COEF  =  |' DIRECT ',        [DEFAUT]
```

NB_VARI_COEF donne le nombre de paramètres quand on parcoure l'espace paramétrique. Le mot-clef TYPE_VARI_COEF permet de dire que l'on va donner *explicitement* la liste des valeurs des paramètres (seul mode disponible pour l'instant).

```
 ◊  VARI_PARA       =  _F(
    ♦   NOM_PARA    =   nompara,           [ para_fonc ]
    ♦   VALE_PARA       =   valepara,           [ R ]
    ♦    VALE_INIT      =    valeinit           [ R ]
```

$)$,

L e mot-clef facteur VARI_PARA donne la liste des paramètres $\gamma_{j=1,n_p=5}$ et leurs valeurs. Pour chaque occurrence, on donne le nom du paramètre dans NOM_PARA . Ce paramètre est forcément utilisé dans les coefficients $\alpha_{i=1,n_m}$ devant les matrices. Pour chaque paramètre, on donne la liste de ses valeurs par VALE_PARA . La longueur du vecteur VALE_PARA est forcément égale à NB_VARI_COEF . Il faut également donner une valeur initiale VALE_INIT (qui initialise l'algorithme glouton).

2.4.3 Opérande SOLVEUR

Ce mot-clef donne les paramètres du solveur utilisé (voir [U4.50.01]). En effet l'algorithme glouton va résoudre un grand nombre de fois le système linéaire défini.

3 Exemples d'utilisation

Exemple d'utilisation du mode incrémental par enrichissement (OPERATION='POD_INCR') :

```
mat1  = AFFE_MATERIAU(AFFE=_F(TOUT='OUI',MATER=acier1))
resu1 = STAT_NON_LINE(CHAM_MATER = mat1,...)
mode1 = DEFI_BASE_REDUITE(RESULTAT=resu1,
                          OPERATION='POD',
                          NOM_CHAM='DEPL')
mat2  = AFFE_MATERIAU(AFFE=_F(TOUT='OUI',MATER=acier2))
resu2 = STAT_NON_LINE(CHAM_MATER = mat2,…)
mode1 = DEFI_BASE_REDUITE(reuse=mode1,
                          OPERATION='POD_INCR',
                          RESULTAT=resu2,
                          NOM_CHAM='DEPL')
```

La base empirique mode1 a été construite sur deux jeux de paramètres matériaux.

Remarques :

- Dans l'exemple ci-dessus, le premier DEFI_BASE_REDUITE aurait pu être utilisé en mode incrémental (OPERATION='POD_INCR') ;
- Avec une tolérance très faible (mot-clef TOLE), le mode incrémental tend vers une SVD classique.