Manuel d'Utilisation
Fascicule U4.5- : Méthodes de résolution
Document : U4.55.03

Opérateur FACT_GRAD

1 But

Construire une matrice de préconditionnement pour une résolution par gradient conjugué. Cette matrice permet d'accélérer la convergence de l'algorithme du gradient conjugué (opérateur RESO_GRAD [U4.55.04]). Cet opérateur ne s'applique qu'à des matrices réelles et symétriques.

Produit une structure de données de type matr_asse_*.

2 Syntaxe

```
matfac [matr_asse_*] = FACT_GRAD

            (  ♦  MATR_ASSE =  mat,              /  [matr_asse_DEPL_R]
                                                 /  [matr_asse_TEMP_R]
                                                 /  [matr_asse_PRES_R]

               ◊  PRE_COND =   'LDLT_INC' ,          [DEFAUT]

               ◊  NIVE_REMPLISSAGE =  /  0,          [DEFAUT]
                                      /  n,          [I]

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

            )  ;
```

```
si MATR_ASSE :    [matr_asse_DEPL_R]     alors   [*] -> DEPL_R
                  [matr_asse_TEMP_R]             [*] -> TEMP_R
                  [matr_asse_PRES_R]             [*] -> PRES_R
```

3 Opérandes

3.1 Opérande MATR_ASSE

♦ MATR_ASSE =

Nom de la matrice réelle et symétrique que l'on veut préconditionner.

3.2 Opérande PRE_COND

◊ PRE_COND =

Méthode de préconditionnement :

'LDLT_INC' : la matrice de préconditionnement est obtenue par une décomposition LDL^T incomplète de la matrice assemblée.
Cette décomposition est plus ou moins complète, suivant le niveau de remplissage. La matrice résultat matfac est de type matr_asse.

3.3 Opérande NIVE_REMPLISSAGE

◊ NIVE_REMPLISSAGE = / 0
/ n

La matrice de préconditionnement (P) utilisée pour accélérer la convergence du gradient conjugué est obtenue en factorisant de façon plus ou moins complète la matrice initiale (A).

Si niv = 0 (défaut)

P a le même stockage que A. La factorisation est incomplète car on n'utilise pour les calculs que les termes que l'on peut stocker dans P.

P représente donc une approximation (médiocre) de A^{-1} ; son stockage est faible.

Si niv = 1

On stocke dans P en plus des termes qui avaient leur place dans le stockage initial, les "descendants" de première génération des termes initiaux. En effet lors de la factorisation, un terme nul dans A peut devenir non nul dans P. On obtient ainsi le remplissage de niveau 1.

Si niv = 2, ...

Le même procédé est repris : la matrice P remplie au niveau niv-1 crée les termes de la matrice P au niveau niv.

Plus niv est grand, plus la matrice P est proche de A^{-1} et donc plus le gradient conjugué converge vite (en nombre d'itérations).

En revanche, plus niv est grand plus le stockage de P devient volumineux (en mémoire et sur disque) et plus les itérations sont coûteuses en CPU.

Les premiers essais ont montré (approximativement) que la taille de P valait :

- 3,5* taille (A) pour niv = 1
- 7,5* taille (A) pour niv = 2

Notre expérience de ce mot clé est encore limitée et nous conseillons d'utiliser la valeur par défaut (niv = 0).

Si niv = 0 ne permet pas au gradient conjugué de converger, on essaiera successivement les valeurs niv = 1, 2, 3.

3.4 Opérande INFO

◊ INFO =

1 : pas d'impression,
2 : cette option est réservée aux développeurs. Impressions intermédiaires sur le fichier message.

4 Exemple d'utilisation

```
nu     =  NUME_DDL( MATR_RIGI= mel, METHODE= 'GCPC'  ) ;
matas  =  ASSE_MATRICE ( MATR_ELEM= mel, NUME_DDL= nu   ) ;
kmatas =  FACT_GRAD    ( MATR_ASSE= matas ) ;
```