Structure de Données sd_matr_asse

Résumé :

Ce document décrit les structures de données sd_matr_asse : les matrices creuses.

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

Table des matières

1 Généralités....3
2 Arborescences....3
3 Contenu des OJB....3
3.1 Objet .REFA....3
3.2 Objet .VALM....5
3.3 Objet .CONL....6
3.4 Objet .DIGS....6
3.5 Objet .PERM....6
3.6 Objet .LIME....6
3.7 Objets .UALF, .VALF, .WALF....6
4 Objets liés à la présence de charges cinématiques éliminées....7
4.1.1 Objet .CCID....7
4.1.2 Objet .CCLL....7
4.1.3 Objet .CCVA....7
4.1.4 Objet .CCII....7
4.1.5 Remarques concernant les matrices « distribuées »....7

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

1 Généralités

Les objets de type sd_matr_asse représentent les matrices assemblées carrées (au sens des éléments finis). Ce sont en général de gros objets. Ces matrices sont creuses, ce qui explique que leur structure ne soit pas simplement un tableau carré.

Une sd_matr_asse peut provenir d'un assemblage de sd_matr_elem ou d'une combinaison linéaire d'autres sd_matr_asse.

Les tableaux décrivant le stockage de la sd_matr_asse sont dans la structure sd_stockage d'un sd_nume_ddl [D4.06.07].

Il existe des sd_matr_asse symétriques et des sd_matr_asse non-symétriques. Mais on suppose que la topologie de la matrice est toujours symétrique. C'est à dire que les termes non nuls sont disposés symétriquement par rapport à la diagonale.

2 Arborescences

sd_matr_asse (K19) ::=record

♦ '.REFA' : OJB S V K24

♦ '.VALM' : OJB XD V R/C NUM() nbobj=1/2

◊ # si il existe des ddls de Lagrange :
♦ '.CONL' : OJB S V R

◊ # si la sd_matr_asse provient d'une manière ou d'une autre de matrices élémentaires :
♦ '.LIME' : OJB S V K24

◊ # si la sd_matr_asse a été "factorisée" (routine preres.F90) :
/ # si factorisée avec MULT_FRONT :
♦ '.VALF : OJB XD V R/C NUM()
◊ # si la matrice est non symétrique :
♦ '.WALF : OJB XD V R/C NUM()
/ # si factorisée avec LDLT :
♦ '.UALF : OJB XD V R/C NUM()
/ # si factorisée avec GCPC + LDLT_INC :
♦ '.PERM : OJB S V I
/ # si factorisée avec LDLT ou MULT_FRONT :
♦ '.DIGS : OJB S V R/C

◊ # si il existe des charges cinématiques :
♦ '.CCID' : OJB S V I
♦ '.CCLL' : OJB S V I
♦ '.CCVA' : OJB S V R/C
♦ '.CCII' : OJB S V I

3 Contenu des OJB

3.1 Objet .REFA

.REFA S V K24 dim=20

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

.REFA(1) = nom du sd_maillage sous-jacent.

.REFA(2) = nom du sd_nume_ddl.

.REFA(3) = ' ' / 'ELIMF' / 'ELIML'
' ' : Il n'existe pas de charges cinématiques.
'ELIMF' : Il existe des charges cinématiques.
Une partie du .VALM a été copiée dans .CCVA
.VALM a été partiellement remplacé par une matrice unité (sur les ddls éliminés)
'ELIML' : Il existe des charges cinématiques.
.CCVA n'a pas encore été créé
.VALM n'a pas encore été recopié dans .CCVA
voir mtmchc.f pour les détails

.REFA(4) = nom de l'OPTION de calcul (ou bien la chaine: '&&MELANGE').

.REFA(5) = ' '

.REFA(6) = ' '

.REFA(7) = ' ' / nomsolv
nomsolv : nom du solveur à utiliser (par défaut) lors des résolutions.

.REFA(8) = ' ' / 'ASSE' / 'DECT' / 'DECP'
' ' ou 'ASSE' : matrice dans son état initial (non factorisée)
'DECT' : matrice entièrement factorisée
'DEPT' : matrice partiellement factorisée (possible uniquement si LDLT)

.REFA(9) = 'MS' / 'MR'
'MS' : matrice symétrique
'MR' : matrice non-symétrique

.REFA(10) = / 'NOEU': les ddls de la matrice sont portés par des nœuds
/ 'GENE': les ddls de la matrice sont des ddls généralisés.

.REFA(11) = 'MPI_COMPLET' / 'MPI_INCOMPLET' / 'MATR_DISTR'
'MPI_COMPLET' : Les objets .VALM (et .CCVA) sont "complets"
'MPI_INCOMPLET' : Les objets .VALM (et .CCVA) sont "incomplets" du fait d'un calcul MPI distribué. Chaque processeur n'assemble que les éléments finis qui lui sont affectés.
'MATR_DISTR' : Si MATR_DISTRIBUEE='OUI'
Cette option dimensionne au plus juste la matrice assemblée en fonction du nombre d'inconnues présentes sur le processeur courant.

Remarque :
si 'MPI_INCOMPLET', les objets .VALM et .CCVA sont alloués à leur taille normale, mais ils sont partiellement remplis de "0". si 'MATR_DISTR', l'objet .VALM est cette fois d'une taille plus petite que dans le cas 'MPI_INCOMPLET'.

.REFA(17) = ' ' / 'XFEM_PRECOND'
' ' : Aucun pré-conditionneur XFEM n'a été appliqué à la matrice. La résolution se poursuit sans traitement particulier, relatif au pré-conditionnement XFEM.
'XFEM_PRECOND' : La matrice a été modifiée à l'aide du pré-conditionneur XFEM. Il faut prendre en compte cette modification dans la suite du calcul.
Remarque :

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

Si la matrice est modifiée par le pré-conditionneur XFEM, cela implique, qu'il y a au moins un objet pré-conditionneur stocké en mémoire. Cet objet est utile pour la suite du calcul (pour la modification du second membre et de la solution).

.REFA(18) = ' ' / '&&XFEM_PC_1'

' ' : Aucun pré-conditionneur XFEM n'est stocké.

'&&XFEM_PC_1' : Le pré-conditionneur XFEM est stocké dans un emplacement « JEVEU » **fixe**.

Concernant le format de stockage de l'objet « JEVEU », la matrice de pré-conditionnement est stockée dans une nouvelle sd_matr_asse. On génère un nouvel *.VALM* (non-symétrique) et un nouveau *NUME_DDL* respectant la numérotation des équations de la matrice initiale. La gestion de l'objet pré-conditionneur XFEM s'effectue comme une « sd_matr_asse » classique, décrite dans ce document.

Remarque :

Même si l'adresse du pré-conditionneur est stockée dans le .REFA de la matrice originale, il n'existe pas de chaînage des adresses « sd_matr_asse ». En effet, l'adresse du pré-conditionneur est fixe et on n'a pas besoin de connaître l'adresse de la matr_asse pour tester l'existence d'un pré-conditionneur, ce qui simplifie la gestion en mémoire de ces deux objets interdépendants.
Par ailleurs, comme le pré-conditionneur XFEM n'est effectif qu'en calcul séquentiel, l'application réciproque de l'adresse d'un pré-conditionneur vers la matr_asse originale n'est pas construite pour le moment.

.REFA(19) = matred / ' '

Nom de la matrice « réduite » obtenue en supprimant les équations de Lagrange (fonctionnalité SOLVEUR / ELIM_LAGR='OUI').On parle aussi de matrice « fille »

.REFA(20) = matmere / ' '

Nom de la matrice « mère » de la matrice si celle-ci a été obtenue en supprimant les équations de Lagrange de matmere (fonctionnalité SOLVEUR / ELIM_LAGR='OUI').

DOCU('.REFA') = / 'ASSE' matrice initiale,
/ 'DECT' matrice complètement factorisée,
/ 'DECP' matrice partiellement factorisée.

3.2 Objet .VALM

.VALM XD V R/C NUM()

Cet objet contient les valeurs des termes non nuls de la matrice (au format Morse). Il y a 1 (ou 2) élément(s) dans cette collection (2 si la matrice est non symétrique).

Le 1er élément de la collection contient la partie supérieure de la matrice,
Le 2ème contient la partie inférieure.

Remarques :

** Pour les matrices non-symétriques, la diagonale est donc stockée 2 fois ;*
On convient de n'utiliser que la diagonale stockée dans la partie supérieure (VALM(1)) ;
** L'arrangement des termes de la matrice dans les blocs est expliqué dans la documentation de la sd_stockage [D4.06.07] ;*
** Même si la matrice n'est pas symétrique, son profil (mapping des termes non nuls) reste symétrique. Un seul profil s'applique donc sur ses deux parties supérieure et inférieure.*

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

3.3 Objet .CONL

.CONL S V R dim=neq

neq est le nombre d'équations du système

Cet objet facultatif n'est présent que si il existe au moins un ddl de type 'LAGR' :
V(ieq) = c si ieq correspond à un DDL nommé 'LAGR',
1. sinon.
c est le coefficient de conditionnement des ddls de Lagrange.

3.4 Objet .DIGS

.DIGS S V R/C dim=2*neq

neq est le nombre d'équations du système

Cet objet n'est présent que si la matrice a été factorisée par 'LDLT' ou 'MULT_FRONT'

DIGS(1:neq) : valeurs de la diagonale de la matrice initiale
DIGS(neq+1:2*neq) : valeurs de la diagonale de la matrice factorisée

3.5 Objet .PERM

.PERM S V I dim=neq

neq est le nombre d'équations du système

Cet objet n'est présent que si la matrice est le résultat du pré-conditionnement par 'GCPC' + 'LDLT_INC' d'une autre matrice. Il sert à établir la correspondance entre la numérotation des équations dans la matrice initiale et sa matrice de pré-conditionnement. En effet, la matrice de pré-conditionnement a été renumérotée avec l'algorithme 'RCMK' (Reverse CUTHILL - Mc KEE)

Soit : B=FACTORISER(MATR_ASSE=A, METHODE='GCPC', PRE_COND='LDLT_INC')

PERM(ieq_A) : ieq_B
où ieq_A et ieq_B sont les numéros d'équations dans les matrices A et B.

Cet objet a le même contenu que l'objet .M2LC de la sd_stockage.

3.6 Objet .LIME

Liste des noms des sd_matr_elem à l'origine de la sd_matr_asse.
Cet objet existe si la sd_matr_asse provient de l'assemblage de sd_matr_elem ou de la combinaison de sd_matr_asse possédant eux-même un objet .LIME.
Cet objet n'existe pas toujours et il vaut mieux éviter de s'en servir.

3.7 Objets .UALF, .VALF, .WALF

Ces collections contiennent les termes de la matrice factorisée.

.UALF contient la matrice factorisée avec LDLT :
symétrique : collection de taille *nblocs*
non-symétrique : collection de taille *2*nblocs*

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

.VALF contient la matrice factorisée avec MULT_FRONT de la partie supérieure
.WALF contient la matrice factorisée avec MULT_FRONT de la partie inférieure

Remarques :

** L'arrangement des termes de la matrice dans les blocs est expliqué dans la documentation de la sd_stockage [D4.06.07] ;*
** Pour le stockage LDLT (.UALF), le nombre de blocs est doublé en cas d'une matrice non symétrique. Les nblocs premiers correspondent aux valeurs associées à la partie supérieure de la matrice tandis que les derniers nblocs correspondent à la partie inférieure de la matrice.*
** Même si la matrice n'est pas symétrique, son profil (mapping des termes non nuls) reste symétrique. Un seul profil s'applique donc sur ses deux parties supérieure et inférieure.*

4 Objets liés à la présence de charges cinématiques éliminées

4.1.1 Objet .CCID

.CCID S V I dim=neq+1

.CCID(ieq)= 1 si le ddl ieq éliminé,
0 sinon.
.CCID(neq)= nelim : nombre de ddls éliminés

4.1.2 Objet .CCLL

.CCLL S V I dim=3*nelim

nelim est le nombre de ddls "éliminés".

```
.CCLL((i-1)*3+1)    : ieq
.CCLL((i-1)*3+2)    : i1
.CCLL((i-1)*3+3)    : i2
```

ieq est le numéro de l'équation correspondante au i^{ème} ddl éliminé,
i1 est le nombre de termes stockés pour l'équation ieq
i2 est le nombre cumulé de termes stockés pour les équations j < i

4.1.3 Objet .CCVA

.CCVA S V R/C

L'objet .CCVA contient les colonnes de la matrice initiale correspondantes aux ddls à éliminer (ceux qui sont imposés par une charge "cinématique"). Plus précisément, on stocke la sous-matrice des termes sur des lignes correspondantes à des DDLs libres et des colonnes à des DDLs imposés.

4.1.4 Objet .CCII

.CCII S V I

L'objet .CCII a la même structure que l'objet .CCVA. Il contient les indices de lignes des termes stockés dans le .CCVA

4.1.5 Remarques concernant les matrices « distribuées »

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

Lorsque la matrice est distribuée,

- L'objet .CCID est identique sur tous les processeurs et sa dernière valeur (nelim) est le nombre total (global) de ddls éliminés.
- L'objet .CCLL est de même longueur sur tous les processeurs mais son contenu est différent car il contient des numéros d'équations locaux.
- Les objets .CCII et .CCVA sont de longueurs différentes sur les différents processeurs et leurs contenus sont liés au matrices locales.

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