**Manuel d'Utilisation**
**Fascicule U4.5- : Méthodes de résolution**
**Document : U4.50.01**

# Mot clé SOLVEUR

## 1 But

Choisir le mode de stockage des matrices et l'algorithme de résolution. Ce mot clé facteur se retrouve dans un certain nombre de commandes conduisant à la résolution de systèmes linéaires. Pour les algorithmes de résolution il permet de choisir entre factorisation "classique" de type "GAUSS" (`'LDLT'`), factorisation multi-frontale (`'MULT_FRONT'`), gradient conjugué préconditionné $\mathrm{ILU}(k)$ (`'GCPC'`) ou solveur FETI par décomposition de domaines (`'FETI'`).

Pour chaque type de solveur, certains paramètres numériques facultatifs sont accessibles et sont décrits ici. Par défaut, c'est le solveur `'MULT_FRONT'` qui est utilisé. Le solveur 'FETI', quant à lui, est encore en phase de développement, il n'est donc pas conseillé de l'utiliser sans conseils préalables de l'équipe de développement.

# 2 Syntaxe

```
◊  SOLVEUR = _F (

   # Factorisation de type "multi-frontale" :
   /  METHODE = 'MULT_FRONT' ,                                  [DEFAUT]

      # Paramètre numérique
            ◊  RENUM            =  /  'METIS' ,                 [DEFAUT]
                                   /  'MD'    ,
                                   /  'MDA'   ,
      # Paramètres fonctionnels
            ◊  STOP_SINGULIER   =  /  'OUI'  ,                  [DEFAUT]
                                   /  'NON'  ,
            ◊  NPREC            =  /  8      ,                 [DEFAUT]
                                   /  nprec  ,                 [I]


   # Factorisation "classique" de type Gauss :
   /  METHODE = 'LDLT' ,

      # Paramètre numérique
            ◊  RENUM            =  /  'RCMK' ,                 [DEFAUT]
                                   /  'SANS' ,
      # Paramètres fonctionnels
            ◊  STOP_SINGULIER   =  /  'OUI'  ,                  [DEFAUT]
                                   /  'NON'  ,
            ◊  NPREC            =  /  8      ,                 [DEFAUT]
                                   /  nprec  ,                 [I]


   # Méthode itérative du gradient conjugué :
   /  METHODE = 'GCPC' ,

      # Paramètres numériques
            ◊  PRE_COND      =     'LDLT_INC' ,                [DEFAUT]
            ◊  NIVE_REMPLISSAGE =  / 0    ,                   [DEFAUT]
                                   / niv  ,
      # Paramètres fonctionnels
            ◊  NMAX_ITER     =     / 0,                        [DEFAUT]
                                   / niter,                    [I]
            ◊  RESI_RELA     =     / 10-6  ,                   [DEFAUT]
                                   / resi ,                    [R]
```

```
# Méthode de décomposition de domaines FETI :
/  METHODE = 'FETI' ,

   # Paramètres fonctionnels
         ♦  PARTITION     =      sdfeti
         ◊  NMAX_ITER     =      / 0 ,                [DEFAUT]
                                 / niter,             [I]
         ◊  RESI_RELA     =      / 10-6  ,            [DEFAUT]
                                 / resi ,             [R]
   # Paramètres du problème d'interface
         ◊  PRE_COND      =      / 'LUMPE' ,          [DEFAUT]
                                 / 'SANS'  ,
         ◊  SCALING       =      / 'MULT'  ,          [DEFAUT]
                                 / 'SANS'  ,
         ◊  TYPE_REORTHO_DD=     / 'GSM'   ,          [DEFAUT]
                                 / 'GS'    ,
                                 / 'IGSM'  ,
                                 / 'SANS'  ,
         ◊  NB_REORTHO_DD =      / 0 ,                [DEFAUT]
                                 / nb_reortho,        [I]
   # Paramètres des problèmes locaux
         ◊  RENUM            =   /  'METIS' ,         [DEFAUT]
                                 /  'MD'    ,
                                 /  'MDA'   ,
         ◊  STOP_SINGULIER   =   /  'OUI'  ,          [DEFAUT]
                                 /  'NON'  ,
         ◊  NPREC            =   /  8      ,          [DEFAUT]
                                 /  nprec  ,          [I]
   # Paramètres de tests, divers
         ◊  VERIF_SDFETI  =      /'OUI' ,             [DEFAUT]
                                 /'NON' ,
         ◊  TEST_CONTINU     =   /10-6   ,            [DEFAUT]
                                 /test_continu,       [R]
         ◊  STOCKAGE_GI      =   /'CAL' ,             [DEFAUT]
                                 /'OUI',
                                 /'NON',
         ◊  INFO_FETI        =   /'FFFFFFFFF',        [DEFAUT]
                                 /info_feti,          [K9]


# Paramètre commun à tous les solveurs
◊  SYME      =  /  'NON'  ,                          [DEFAUT]
                /  'OUI'  ,

                ),
```

# 3 Opérandes

## 3.1 Opérande METHODE

◊ METHODE =

Ce mot clé permet de choisir la méthode de résolution des systèmes linéaires :

/ 'MULT_FRONT' (**défaut**) Solveur direct de type 'multi-frontale'. Le stockage matriciel est 'MORSE' et donc proscrit tout pivotage. Cette méthode est parallélisée en mémoire partagée (Open MP) et peut être exécutée sur plusieurs processeurs (*via* l'interface Astk menu Options – Options de lancement). La matrice initiale est stockée dans un seul objet JEVEUX et sa factorisée est répartie sur plusieurs, donc peut être déchargée partiellement et automatiquement sur disque.

/ 'LDLT' Solveur direct avec factorisation de Crout par blocs (sans pivotage). Le stockage matriciel est 'ligne de ciel' ou 'SKYLINE'. On a une pagination de la mémoire complètement paramétrable (la matrice est décomposée en blocs gérés en mémoire de façon indépendante et déchargés sur disque au fur et à mesure) qui permet de passer de gros cas mais qui se paye par des accès disques coûteux.

/ 'GCPC' Solveur itératif de type gradient conjugué avec préconditionnement ILU($k$). Le stockage de la matrice est alors 'MORSE'. La matrice initiale et sa factorisée incomplète sont stockées, chacune, dans un seul objet JEVEUX.

/ 'FETI' Solveur par décomposition de domaines de type FETI : gradient conjugué préconditionné projeté (GCPPC) pour le problème d'interface et solveur direct multi-frontale pour les inversions des matrices de rigidité locales.
Les problèmes locaux étant inversés par la multi-frontale, leurs matrices associées, matrices de rigidité locales et factorisées, sont traitées comme tel (cf. ci-dessus).

Les valeurs par défaut des autres mots clés sont alors prises automatiquement en fonction de la méthode choisie.

| | Robustesse | Mémoire (RAM/Disque) | CPU | Paramétrage | Petit cas (<$10^3$ DDL) | Cas standards (<$10^6$ DDL) | (Très) gros cas (>$10^6$ DDL) |
|---|---|---|---|---|---|---|---|
| ***MULT_FRONT*** DEFAUT | Bonne | RAM : faible Disque : importante | Bon | Rien à faire | non | oui | Plutôt non ou avec la version parallèle |
| LDLT | Bonne | Du fait de la pagination on peut moduler l'espace disque et la RAM | Coûteux | Rien à faire | oui | non | non |
| GCPC | Très variable | Très variable suivant le niveau de précondi tionnement | Très variable suivant le niveau de précondi tionnement | A adapter au cas par cas | oui | Oui si thermique ou pb mécanique bien conditionné | Plutôt non |
| FETI | Plutôt bonne | RAM : faible Disque : importante | Bon | A adapter au cas par cas | non | Oui avec au moins **$10^4$** DDL par sous-domaine | Oui avec au moins **$10^4$** DDL par sous-domaine |

## 3.2 METHODE : 'MULT_FRONT'

◊ RENUM =

Cet argument permet de renuméroter les nœuds du modèle :

/ 'MD' ("Minimum Degré") cette numérotation des nœuds minimise le remplissage de la matrice lors de sa factorisation.

/ 'MDA' ("Minimum Degré Approché") cette numérotation est en principe moins optimale que 'MD' en ce qui concerne le remplissage mais elle est plus économique à calculer. Elle est toutefois préférable à 'MD' pour les gros modèles (≥ 50 000 ddls).

/ 'METIS' (**défaut**) Autre méthode de numérotation basée sur une dissection emboîtée. Cette méthode n'est possible que sur le serveur de calcul dédié au projet Aster (Alpha Serveur TRU64) sauf à installer soi-même l'exécutable METIS. Sur cette machine, c'est la méthode la plus efficace (en temps CPU et en mémoire).

◊ STOP_SINGULIER = 'OUI' (**défaut**) / 'NON'

Lorsqu'au terme de la factorisation, on constate qu'un terme diagonal $d'$ est devenu très petit (par rapport à ce qu'il était avant la factorisation $d$ ), c'est que la matrice est (probablement) presque singulière.

Soit $n = \log\left|\frac{d}{d'}\right|$, ce rapport de magnitude indique que sur une équation (au moins) on a perdu $n$ chiffres significatifs.

Si $n$ > nprec (mot clé NPREC ci-dessous), on considère que la matrice est singulière. Si l'utilisateur a indiqué : STOP_SINGULIER = 'OUI', le code s'arrête alors en ERREUR FATALE, sinon l'exécution se poursuit avec émission d'une alarme.

**Remarque :**

*Toute perte importante de chiffres significatifs lors d'une factorisation est un indicateur d'un problème mal posé. Plusieurs causes sont possibles (liste non exhaustive) :*

- *des conditions aux limites de blocage de la structure insuffisantes,*
- *des relations linéaires redondantes,*
- *des données numériques très hétérogènes (termes de pénalisation trop grands), ...*

◊ NPREC = nprec (**défaut**=8)

C'est le nombre qui sert à déterminer si la matrice est singulière (ou non) (cf. mot clé STOP_SINGULIER ci-dessus).

## 3.3 METHODE : 'LDLT'

◊ RENUM =

Cet argument permet de renuméroter si on le désire les nœuds du modèle :

'SANS' On garde l'ordre initial donné dans le fichier de maillage,

'RCMK' (**défaut**) "Reverse Cuthill-MacKee", cet algorithme de renumérotation est souvent efficace pour réduire la place nécessaire au stockage "ligne de ciel" de la matrice assemblée et pour réduire le temps nécessaire à la factorisation de la matrice.

◊ STOP_SINGULIER

Voir [§3.2].

◊ NPREC

Voir [§3.2].

## 3.4 METHODE : 'GCPC'

◊ PRE_COND = 'LDLT_INC' (**défaut**)

Méthode de préconditionnement : la matrice de préconditionnement est obtenue par une décomposition $\mathbf{LDLT}$ incomplète de la matrice assemblée.

◊ NIVE_REMPLISSAGE = / 0 (**défaut**)
/ niv

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

Si niv = 0

$\mathbf{P}$ a le même stockage que $\mathbf{A}$. La factorisation est incomplète car on n'utilise pour les calculs que les termes que l'on peut stocker dans $\mathbf{A}$. $\mathbf{P}$ représente donc une approximation (médiocre) de $\mathbf{A}^{-1}$ ; son stockage est donc plus raisonnable.

Si niv = 1

On stocke dans $\mathbf{P}$ 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 $\mathbf{A}$ peut devenir non nul dans $\mathbf{P}$. On obtient ainsi le remplissage de niveau 1.

Si niv = 2, ...

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

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

- 1*taille($\mathbf{A}$) pour niv = 0
- 3,5*taille($\mathbf{A}$) pour niv = 1
- 7,5*taille($\mathbf{A}$) 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…

◊ NMAX_ITER = niter (**défaut**=0)

Nombre d'itérations maximum de l'algorithme de résolution itératif. Si niter = 0 alors le nombre maximum d'itérations est calculé comme suit :
niter = nequ/2 où nequ est le nombre d'équations du système.

◊ RESI_RELA = resi (**défaut**=$10^{-6}$)

Critère de convergence de l'algorithme : c'est un critère relatif sur le résidu :

$$\frac{\|\mathbf{r}_m\|}{\|\mathbf{b}\|} \leq resi$$

$\mathbf{r}_m$ est le résidu à l'itération $m$

$\mathbf{b}$ est le second membre et $\|\ \|$ la norme euclidienne

## 3.5 METHODE : 'FETI'

♦ PARTITION = sdfeti

Nom utilisateur de l'objet SD_FETI décrivant le partitionnement en sous-domaines. Il est généré par un appel préalable à l'opérateur DEFI_PART_FETI [U4.23.05].

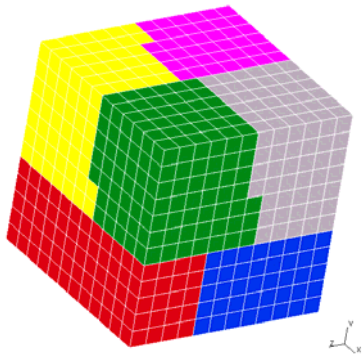

**Figure 3.5-a : Exemple de structure partitionnée en sous-domaines**

◊ NMAX_ITER = niter (**défaut**=0)

Nombre d'itérations maximum du GCPPC résolvant le problème d'interface. Si niter = 0 alors le nombre maximum d'itérations est calculé comme suit :
niter = max(nbi/100,10) où nbi le nombre d'inconnues du problème d'interface.

◊ RESI_RELA = resi (**défaut**=$10^{-6}$)

Critère de convergence de l'algorithme : c'est un critère relatif sur le résidu projeté du problème d'interface

$$\frac{\|\mathbf{P}\mathbf{r}_m\|}{\|\mathbf{b}\|} \leq resi$$

$\mathbf{r}_m$ est le résidu à l'itération $m$

$\mathbf{P}$ l'opérateur de projection

$\mathbf{b}$ est le second membre et $\|\ \|$ la norme euclidienne

◊ PRE_COND =

Cet argument permet de choisir le type de préconditionneur pour le GCPPC :

| | |
|---|---|
| 'SANS' | Pas de préconditionnement. |
| 'LUMPE' (**défaut**) | Préconditionnement lumpé. |

Normalement le préconditionneur lumpé conduit à un gain en itérations et en CPU, sans surcoût mémoire.

◊ SCALING =

Cet argument permet de choisir le type de scaling (mise à l'échelle) adopté pour le préconditionneur. Il n'est donc pris en compte que si PRE_COND est différent de 'SANS'.

'SANS' Pas de phase de scaling.
'MULT' (**défaut)** Mise à l'échelle par la multiplicité des nœuds d'interface.

Normalement la phase de scaling conduit à un gain en itérations et en CPU, sans surcoût mémoire. Surtout lorsque le partitionnement produit beaucoup de points de jonction (points appartenant à plus de deux sous-domaines).

◊ TYPE_REORTHO_DD =

Cet argument permet de choisir le type de réorthogonalisation des directions de descente. Il est lié au paramètre NB_REORTHO_DD.

'SANS' Pas de réorthogonalisation des méthodes de descente.
'GS' Réorthogonalisation de Gram-Schmidt.
'GSM' (**défaut)** Réorthogonalisation de Gram-Schmidt Modifié.
'IGSM' Réorthogonalisation de Gram-Schmidt Modifié Itératif.

Cette phase permet de lutter contre la propension des directions de descente du GCPPC à perdre leur orthogonalité. En théorie, IGSM est meilleure que GSM qui est lui même supérieur à GS. En pratique, le meilleur compromis « surcoût calcul/qualité d'orthogonalité » est souvent réalisé par GSM.

◊ NB_REORTHO_DD = nb_reortho (**défaut**=0)

Nombre de directions de descente initiales utilisées dans la phase de réorthogonalisation. En principe, plus il est grand, meilleure est la convergence, mais plus grand est aussi le surcoût calcul et mémoire. Il faut donc trouver un compromis entre ces éléments.
Si nb_reortho = 0 alors ce nombre est calculé comme suit :
nb_reortho = max(niter/10,5) où niter le nombre maximal d'itérations définies ci-dessus.

◊ RENUM

Voir [§3.2].

◊ STOP_SINGULIER

Voir [§3.2].

◊ NPREC

Voir [§3.2].

◊ VERIF_SDFETI = 'OUI' (**défaut**) / 'NON'

On comptabilise les incohérences en terme de nom de modèle et de noms de chargement, entre le paramétrage de l'opérateur appelant le mot-clé SOLVEUR et celui fournit à l'opérateur de partitionnement qui reste stocké dans la SD_FETI. Il faut que les noms de modèles soient identiques et que la liste des chargements de l'opérateur appelant soit incluse dans celle de DEFI_PART_OPS. Si ce n'est pas le cas et si VERIF_SDFETI = 'OUI', on s'arrête en ERREUR_FATALE, sinon on émet une ALARME.

◊ TEST_CONTINU = test_continu (**défaut**=$10^{-6}$)

Critère du test de continuité à l'interface : c'est un critère relatif sur les valeurs (non nulles) des inconnues aux interface. Si on est en dessus du critère, il y'a émission d'une ALARME.

◊ STOCKAGE_GI = 'CAL' (**défaut**) / 'OUI' / 'NON'

Lorsque le nombre de sous-domaines augmente, un objet devient proéminent, c'est $\mathbf{G}_I$ la matrice des traces des modes de corps rigides sur l'interface. Elle est utilisée dans la phase de projection, c'est-à-dire 10 + nombre_itérations_FETI * 4. Pour permettre à l'utilisateur d'adapter le compromis « taille mémoire/temps CPU », son stockage est paramétrable :

- Si STOCKAGE_GI = 'OUI', elle est calculée et stockée une fois pour toute. Cela nécessite plus de mémoire mais moins de temps calcul lorsqu'on s'en sert.
- Si STOCKAGE_GI = 'NON', c'est l'inverse, elle est recalculée à chaque fois que cela est nécessaire.
- Si STOCKAGE_GI = 'CAL', le choix 'OUI' ou 'NON' va être calculé automatiquement. Si la taille de la matrice est inférieure à la taille moyenne des matrices de rigidité locales, on stocke ('OUI'), sinon, on recalcule ('NON').

◊ INFO_FETI = info_feti (**défaut=**'FFFFFFFFF')
Chaîne de caractères permettant de paramétrer les affichages de l'algorithme FETI, de ses pré et post-traitements ainsi que de ses tests de cohérence. Ce monitoring est indépendant du mot-clé INFO. Etant souvent très verbeux et parfois coûteux en mémoire et en CPU, il doit être utilisé de préférence sur des petits cas et plutôt pour des activités de développements.

- Si INFO_FETI(1:1)='T' : déroulement général de l'algorithme.
- Si INFO_FETI(2:2)='T' : contenu des structures de données hors CHAM_NO et MATR_ASSE.
- Si INFO_FETI(3:3)='T' : contenu des structures de données CHAM_NO et MATR_ASSE.
- Si INFO_FETI(4:4)='T': affichage de variables intermédiaires.
- Si INFO_FETI(5:5)='T': détails des routines d'assemblages.
- Si INFO_FETI(6:6)= 'T': tests de validité des modes de corps rigides.
- Si INFO_FETI(7:7)= 'T': test de la définie-positivité de l'opérateur d'interface.
- Si INFO_FETI(8:8)= 'T': test des orthogonalités du GCPPC.
- Si INFO_FETI(9:9)= 'T': affichages généraux (taille de l'interface, des matrices de rigidité locales et de leurs factorisées, nombre total de modes rigides…) et profiling des différentes étapes de FETI, détaillées par sous-domaine.

## 3.6 Mot clé SYME

◊ SYME = / 'OUI'
/ 'NON'

Si la matrice du système linéaire $\mathbf{A}$ est non-symétrique, le mot clé SYME = 'OUI' permet de symétriser cette matrice avant la résolution du système. La matrice alors est remplacée par $\mathbf{A}' = \frac{1}{2}\left(\mathbf{A} + \mathbf{A}^T\right)$.

**Attention :**

*La symétrisation de la matrice $\mathbf{A}$ conduit donc à résoudre un autre problème que celui que l'on cherche à résoudre ! En réalité, cette possibilité (SYME = 'OUI') n'est utile que dans les commandes non-linéaires (comme STAT_NON_LINE par exemple), pour lesquelles la convergence vers la solution est obtenue par itérations successives. Chaque itéré est obtenu par "estimation" et l'on vérifie ensuite qu'il est "solution". Dans ce cas, une erreur légère sur les itérés n'empêche pas de converger vers la bonne solution. L'intérêt de ce mot clé est de gagner du temps lors de la résolution des systèmes linéaires. Le tout est de savoir si la symétrisation perturbe beaucoup (ou non) la solution du système linéaire ? On peut citer (par exemple) le cas des modèles 3D (ou coque) avec pression suiveuse pour lesquels la symétrisation fait gagner beaucoup de temps.*

# 4 Exemples

## 4.1 Solveur par défaut

Il n'y a rien à écrire ! Mais on peut aussi écrire : SOLVEUR=_F( )

## 4.2 Gradient conjugué

On veut utiliser le gradient conjugué. On pense que la convergence sera plus efficace si l'on autorise un pré-conditionnement plus poussé (NIVE_REPLISSAGE=1).

```
SOLVEUR=_F(METHODE ='GCPC', NIVE_REMPLISSAGE=1,)
```