Une version parallèle des MAN par décomposition de domaine

Isabelle Galliet, Bruno Cochelin

Une version parallèle des MAN par décomposition de domaine

Isabelle Galliet — Bruno Cochelin

Laboratoire de Mécanique et d'Acoustique CNRS UPR 7051
Ecole Généraliste d'Ingénieurs de Marseille
Technopôle de Château Gombert
F-13383 Marseille Cedex 13
bruno.cochelin@egim-mrs.fr

RÉSUMÉ. *Ce papier est consacré à la mise en œuvre des Méthodes Asymptotiques Numériques sur une machine parallèle, par décomposition de domaine. Après avoir rappelé les principes généraux des MAN et les particularités des systèmes linéaires qui en résultent, on justifie l'utilisation d'une méthode de décomposition de domaine de type FETI dans sa version dite « pour seconds membres répétés ». Les performances de ce couple MAN et FETI sont discutées et analysées sur des exemples de problèmes d'élasticité en non linéaire géométrique.*

ABSTRACT. *This paper is concerned with the implementation of the asymptotic-numerical method on a parallel computer, with a domain decomposition technique. After some recalls on the ANM principles and on the particularities of the linear systems that have to be solved, it is shown that the so-called 'for repeted right-hand-side' FETI domain decomposition method is well adapted. The performance of this combination MAN+FETI is dicussed and analysed on various examples taken from geometrically non linear elasticity.*

MOTS-CLÉS : *Méthode asymptotique numérique, solveurs parallèles, FETI, solveurs itératifs, décomposition de domaine.*

KEYWORDS: *Asymptotic numerical method, parallel solver, FETI, iteratives solvers, domain decomposition.*

1. Introduction

La Méthode Asymptotique Numérique est une technique de continuation basée sur des développements en séries des branches de solution. A chaque pas de continuation, elle conduit à résoudre une succession de 20 à 30 systèmes linéaires à matrice invariante. En mécanique des structures, ces matrices sont typiquement les matrices de raideur tangente de la structure évaluées au point de départ du pas de continuation.

Le choix d'un bon solveur pour ces systèmes à matrices invariantes dépend bien entendu du nombre de degrés de liberté du problème et du type de machine utilisée. Pour les problèmes de petites et moyennes tailles, résolus sur une machine séquentielle, c'est assurément les solveurs directs à base de factorisation qui sont les plus performants. En revanche, la réponse n'est pas aussi évidente pour les problèmes de grandes et très grandes tailles traités sur une machine parallèle. On sait en effet que les solveurs directs, et notamment l'étape de descente-remonté, ont du mal à être efficaces en parallèle, et on peut se demander si les algorithmes itératifs ne présentent pas une alternative intéressante. Ce papier apporte des éléments de réponse à ces questions en suivant l'organisation suivante : à la section 2, on rappelle le principe des MAN en insistant sur les propriétés des systèmes linéaires à résoudre à chaque pas de continuation. On envisage ensuite plusieurs stratégies de parallélisme à la section 3, et on retient la méthode de décompostion de domaine FETI dans sa version pour seconds membres répétés. Les performances du couple MAN et FETI sont étudiées à la section 4, et les améliorations liées au stockage des directions de descente sont présentées à la section 5. Enfin, on termine par une analyse des conséquences de l'utilisation d'un solveur itératif dans la MAN à la section 6.

2. Rappel sur la MAN et la particularité des systèmes linéaires à résoudre

Dans ce paragraphe, on rappelle très brièvement le principe d'une MAN en ne développant que les points utiles pour le parallélisme. Le lecteur non initié pourra se rapporter à [AZR 93, COC 94, ZAH 99] et à la bibliographie des divers articles qui composent ce numéro spécial consacré à la MAN. Soit un problème mécanique non linéaire :

$$R(U,\beta) = L(U) + Q(U,U) - \beta F = 0 \qquad [1]$$

où U réprésente les inconnues (déplacements, contraintes...), β le paramètre de chargement, et R les équations (équilibre, comportement...). On suppose que ce problème mécanique est régulier et que, moyennant l'introduction de variables additionnelles adéquates, les équations sont quadratiques par rapport à U. Par exemple, en élasticité non linéaire géométrique, on retient le déplacement u et le deuxième tenseur de Piola-

Kirchhoff S, comme inconnues, $U = (u, S)$. Le vecteur R contient alors les équations d'équilibre et la relation de comportement.

$$R(u,S,\beta) = \left\{ \begin{array}{l} \int_{\Omega_0} S : \delta E(u) d\Omega_0 - \beta(\int_{\partial\Omega_{0_f}} F.\delta u dS + \int_{\Omega_0} f.\delta u dv) = 0 \quad \forall \delta u \\ S = D : (E^l(u) + E^{nl}(u,u)) \end{array} \right. \qquad [2]$$

Les MAN reposent sur le développement des branches de solution sous la forme de séries entières [3], en fonction du paramètre de pseudo-longueur d'arc [4]. Ces séries sont tronquées en pratique à un ordre N qui varie entre 20 et 30.

$$\begin{array}{l} U(a) = U_0 + aU_1 + a^2U_2 + \cdots + a^pU_p + \cdots + a^NU_N \\ \beta(a) = \beta_0 + a\beta_1 + a^2\beta_2 + \cdots + a^p\beta_p + \cdots + a^N\beta_N \end{array} \qquad [3]$$

$$a = < U - U_0, U_1 > + (\beta - \beta_0)\beta_1 \qquad [4]$$

En injectant les séries dans [1] et [4], et en identifiant selon les puissances de a, on est conduit à résoudre le problème suivant à l'ordre 1,

$$\begin{array}{l} R_1 = L(U_1) + Q(U_0, U_1) + Q(U_1, U_0) - \beta_1 F = 0 \\ < U_1, U_1 > + \beta_1^2 = 1 \end{array} \qquad [5]$$

et puis pour les ordres p suivants,

$$\begin{array}{l} R_p = L(U_p) + Q(U_0, U_p) + Q(U_p, U_0) + \sum_{r=1}^{p-r} Q(U_r, U_{p-r}) - \beta_p F = 0 \\ < U_1, U_p > + \beta_1\beta_p = 0 \end{array} \qquad [6]$$

Il s'agit ici de problèmes d'élasticité linéaire bien posés, où $\sum_{r=1}^{p-r} Q(U_r, U_{p-r})$ joue le rôle d'un chargement et d'une précontrainte connus. Comme ce terme dépend de tous les U_k précédents ($k < p$), on resout ces problèmes de façon successive. En pratique, on privilégie une formulation en déplacement et on discrétise ces problèmes à l'aide d'éléments finis tout à fait standard. En définitive, la détermination des séries [3] nécessite la résolution successive de :

$$\left\{ \begin{array}{lcll} K_T\ u_1 & = & f_{ext} & \\ K_T\ u_p^{nl} & = & -f_p^{nl} & p = 2, \cdots, N \end{array} \right. \qquad [7]$$

où K_T est la matrice de rigidité tangente au point de départ (U_0, β_0), f_{ext} le vecteur des charges extérieures appliquées, et f^{nl} est un chargement qui dépend des U_k déjà calculés. En complément de ces résolutions, il y a des opérations au niveau élémentaire telles que le calcul des contraintes S_p en chaque point de Gauss.

Les systèmes linéaires [7] ont deux particularités : la première, qui est une conséquence directe des dévelopements asymptotiques, est qu'ils ont tous la même matrice de rigidité. La deuxième propriété est que les vecteurs f_i^{nl} sont en général plongés dans un espace de dimension bien plus petite que N. On illustre ici sur deux exemples

concrets en visualisant les coefficients $\alpha(i,j)$ de la décomposition de Gram-schmidt de ces vecteurs f_i^{nl}.

$$\begin{cases} f_2^{nl} = \alpha(2,2) f_2^{nl^*} \\ \cdots \\ f_p^{nl} = \alpha(p,2) f_2^{nl^*} + \cdots + \alpha(p,p) f_p^{nl^*} \\ \cdots \\ f_n^{nl} = \alpha(N,2) f_2^{nl^*} + \cdots + \alpha(N,p) f_p^{nl^*} + \cdots + \alpha(N,N) f_n^{nl^*} \end{cases} \quad [8]$$

Le premier exemple est une coque circulaire courbe pincée [NAJ 98]. Les coefficients montrés à la figure 1 ont été normalisés par $\alpha(i,3)$ pour les i impairs et par $\alpha(i,2)$ pour les i pairs. Cette distinction entre les termes pairs et impairs provient d'une alternance entre les problèmes de membrane et de flexion sur ces modèles de coques. On peut voir sur la courbe que les coefficients $\alpha(i,k)$ sont petits dès que k est supérieur à 10. De plus, la superposition des $\alpha(i,k)$ pour les i pairs et les i impairs montre la relative colinéarité entre les f_p^{nl} pairs et celle entre ceux impairs. Finalement, cette figure illustre le fait que l'espace auquel appartiennent les seconds membres est de faible dimension.

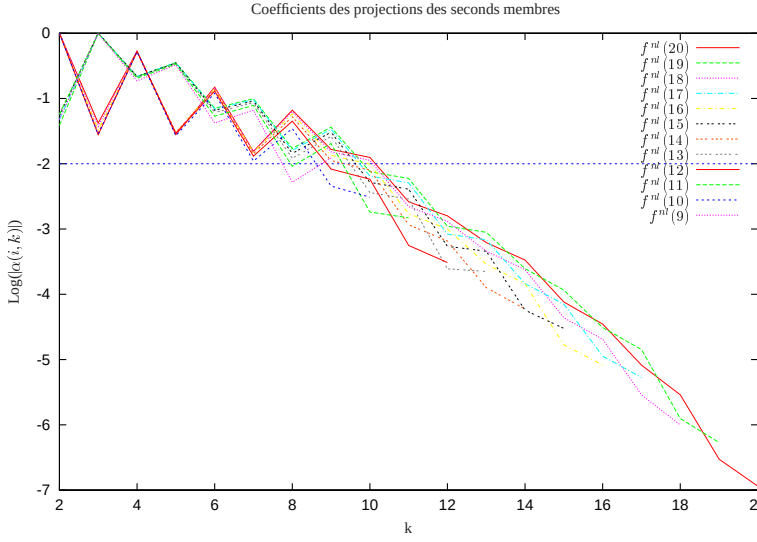

Figure 1. *Représentation des seconds membres dans une base orthonormale pour l'exemple d'une coque circulaire pincée*

Pour confirmer ce résultat,on donne également les $\alpha(i,k)$ pour l'exemple très différent de la figure 2. Il s'agit cette fois d'un carré inhomogène (2D en contraintes planes) maillé régulièrement par 6400 éléments quadrilatères quadratiques (Q8, 8 nœuds et 2 ddls par nœud), sur lequel on applique des forces en diagonale. Dans cet exemple, les coefficients de Poisson des 2 matériaux sont égaux à 0.3 et les modules d'Young sont de 200000 MPa pour le matériau 1 et de 10 Mpa pour le matériau 2,

ce qui représente une forte hétérogénéité. Sur la figure 3, on donne les coefficients $\alpha(i,k)$ qui cette fois sont tous normalisés par $\alpha(i,3)$, car il n'y a pas de parité membrane/flexion dans cet exemple. Cette fois encore, on peut remarquer que les $\alpha(i,k)$ sont très petits dès que k est supérieur à 7. En effet à partir de k=8, les $\alpha(i,k)$ sont tous 10^4 fois plus petits que le plus grand $\alpha(i,k)$. Finalement, les f_i^{nl}, dans cet exemple, peuvent être assez justement représentés par leur projection sur les 6 premiers vecteurs de la base orthonormée $f_k^{nl^*}$ $k = 2, \cdots, 7$.

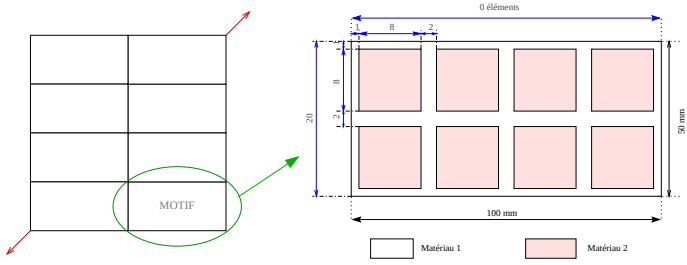

Figure 2. *Description géométrique du carré inhomogène*

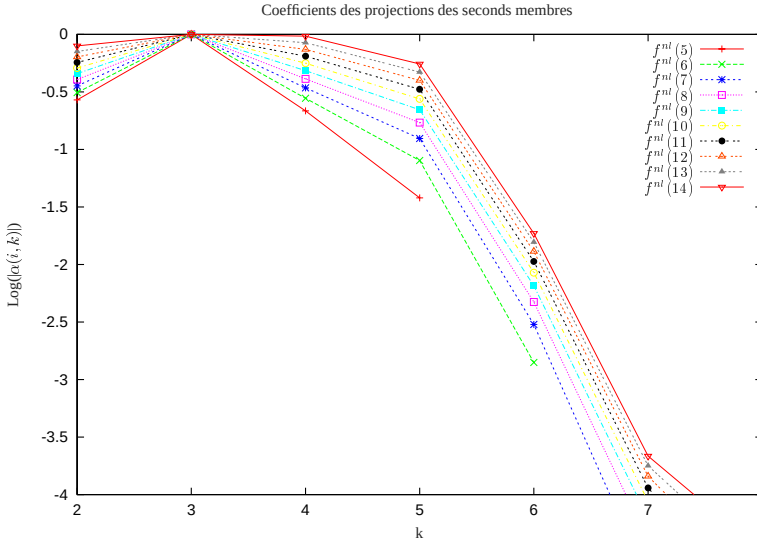

Figure 3. *Représentation des seconds membres du carré dans une base orthonormale*

Les deux exemples que l'on vient de donner montrent que l'espace auquel appartiennent les seconds membres de la MAN est de faible dimension, comme cela est évoqué dans [NOO 80]. On cherchera, dans la parallélisation des MAN, à exploiter cette caractéristique en même temps que l'invariance de la matrice.

3. Diverses stratégies de parallélisation d'une MAN

Outre quelques opérations au niveau des éléments, dont la parallélisation est naturelle, le calcul des séries [3] repose sur la résolution d'une suite de systèmes linéaires à matrice invariante. Pour la parallélisation de ces systèmes, on distingue :

– **Les solveurs directs parallèles.** Ils sont robustes et très peu sensibles au mauvais conditionnement des matrices, mais ils réclament une place mémoire importante pour la décomposition. Ils sont *a priori* bien adaptés pour les multi-résolutions car l'opération lourde qu'est la décomposition n'est faite qu'une seule fois. Cependant l'étape de descente-remontée qui doit être effectuée pour chaque nouveau membre est difficile à paralléliser efficacement.

– **Les solveurs itératifs parallèles.** La parallélisation est plus simple et plus efficace que pour les solveurs directs et les besoins en ressources informatiques sont moindres. En revanche, il faut impérativement un bon préconditionneur et ces algorithmes sont en principe mal adaptés pour une suite de systèmes à matrice invariante.

– **Les solveurs par décomposition de domaine.** En utilisant des solveurs directs séquentiels au niveau des sous-domaines et un solveur itératif pour le problème d'interface global, on peut tirer parti des avantages respectifs des méthodes précédentes sans trop souffrir de leurs inconvénients. Notamment, le problème d'interface étant de taille très inférieure à celle de la structure globale, on peut envisager des méthodes de type Krylov-Augmenté [REY 96] (stockage et réutilisation des directions de descente) pour adapter l'algorithme itératif à la résolution de système avec plusieurs seconds membres.

On choisit ici cette dernière solution et on retient la méthode FETI (formulation duale du problème d'interface) qui, grâce à l'adjonction de problème grossier à chaque itération, est extensible et présente une convergence rapide [FAR 91, FAR 94b]. On adaptera la MAN à la version de FETI dite « pour seconds membres répétés »[FAR 94a].

3.1. *Quelques rappels sur la méthode FETI*

Après avoir découpé le domaine Ω, en N_s sous-domaines disjoints, on réécrit le système d'équation sous la forme :

$$\left\{ \begin{array}{lcll} K^{(s)} u^{(s)} & = & f^{(s)} - {B^{(s)}}^T \lambda & s = 1 \cdots N_s \\ \displaystyle\sum_{s=1}^{N_s} B^{(s)} u^{(s)} & = & 0 & \end{array} \right. \qquad [9]$$

où $K^{(s)}$, $u^{(s)}$ et $f^{(s)}$ représentent respectivement la matrice de raideur, le déplacement et la force associés au sous-domaine $\Omega^{(s)}$, $B^{(s)}$ une matrice booléenne signée représentant la trace de l'interface sur le sous-domaine $\Omega^{(s)}$ et λ un vecteur de multiplicateurs de Lagrange contenant les forces d'interaction entre les sous-domaines. La première équation de [9] donne l'équilibre local sur chaque sous-domaine et la

deuxième la continuité du déplacement à l'interface. En tirant $u^{(s)}$ de la première équation et en reportant dans la seconde on forme le problème d'interface dual, c'est-à-dire posé en termes des forces d'interface.

Le découpage en sous-domaines peut entraîner l'apparition de sous-domaines dits flottants, sur lesquels le nombre de conditions limites en déplacement est insuffisant pour que le problème local soit bien posé. Dans ce cas, la valeur de $u^{(s)}$ est donnée par :

$$u^{(s)} = {K^{(s)}}^{+} \left(f^{(s)} - {B^{(s)}}^{T} \lambda \right) + R^{(s)} \alpha^{(s)} \qquad [10]$$

où ${K^{(s)}}^{+}$ est une inverse généralisée, $R^{(s)}$ le noyau de $K^{(s)}$ et $\alpha^{(s)}$ un vecteur de la taille du noyau. Le produit $R^{(s)}\alpha^{(s)}$ représente une combinaison linéaire arbitraire des modes rigides pour le sous-domaine flottant. La condition de solvabilité de l'équation [9] entraine des conditions sur les forces d'interfaces (auto-équilibre du vecteur chargement) qui s'écrivent :

$${R^{(s)}}^{T} K^{(s)} u^{(s)} = {R^{(s)}}^{T} \left(f^{(s)} - {B^{(s)}}^{T} \lambda \right) = 0 \qquad [11]$$

En substituant l'équation [10] dans [9] et en utilisant [11], on obtient après quelques manipulations le problème d'interface dual :

$$\begin{bmatrix} F_I & -G_I \\ -G_I^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \alpha \end{bmatrix} = \begin{bmatrix} d \\ -e \end{bmatrix} \qquad [12]$$

avec

$$\begin{cases} F_I &= \sum_{s=1}^{N_s} B^{(s)} {K^{(s)}}^{+} {B^{(s)}}^{T} = \sum_{s=1}^{N_s} F_I^{(s)} \\ G_I &= \left[B^{(1)} R^{(1)} \dots B^{(N_f)} R^{(N_f)} \right] \\ \alpha &= \left[{\alpha^{(1)}}^{T} \dots {\alpha^{(N_f)}}^{T} \right]^{T} \\ d &= \sum_{i=1}^{N_s} B^{(s)} {K^{(s)}}^{+} f^{(s)} \\ e &= \left[{f^{(s)}}^{T} R^{(s)} \right]^{T} \; s = 1 \dots N_f \end{cases}$$

Enfin, on réécrit [12] sous la forme d'un problème de minimisation sous contrainte :

$$\begin{array}{ll} \min_\lambda \Phi(\lambda) & = \frac{1}{2} \lambda^T F_I \lambda - \lambda^T d \\ G_I^T \lambda & = e \end{array} \qquad [13]$$

Losque la matrice F_I est symétrique définie positive, on peut résoudre avec une méthode de gradient conjugé avec une projection à chaque itération pour satisfaire la contrainte $G_I^t \lambda = e$. L'étape de projection conduit à la résolution d'un système de petite taille, global sur tous les sous-domaines, permettant ainsi de bien propager l'erreur globale, et d'accélérer la convergence [FAR 00]. Deux préconditionneurs sont classiquement utilisés dans la méthode FETI. Le premier, appelé préconditionneur

optimal de Dirichlet, est basé sur l'interprétation mécanique du problème inverse à l'interface. Ce préconditionneur est donné par l'équation [14], dans laquelle $S^{(s)^+}$ est une inverse généralisée de la matrice du complément de Schur du sous-domaine $\Omega^{(s)}$, et W une matrice diagonale de pondération. Cependant, ce préconditionneur coûte cher en temps de calcul et en place mémoire. En effet, il nécessite de calculer les $K_{ii}^{(s)^{-1}}$, donc de stocker et de factoriser les matrices $K_{ii}^{(s)}$, et de rajouter à chaque itération une étape de descente/remontée.

$$\widetilde{F}_I^{D^{-1}} = \sum_{s=1}^{N_s} W B^{(s)} \begin{bmatrix} 0 & 0 \\ O & S^{(s)} \end{bmatrix} B^{(s)^T} W \qquad [14]$$

$$S^{(s)} = K_{bb}^{(s)} - K_{ib}^{(s)^T} K_{ii}^{(s)^{-1}} K_{ib}^{(s)} \qquad [15]$$

$$W(i,i) = \frac{1}{\text{Nombre de sous-domaines auxquels appartient le nœud } i} \qquad [16]$$

Pour ces raisons, un autre préconditionneur, appelé économique LUMPED, dont la définition se trouve dans l'équation [17], est aussi utilisé par FETI. Ce préconditionneur est dit économique car il ne nécessite aucun stockage supplémentaire et ne demande que des produits matrice/vecteur de la taille du problème d'interface.

$$\widetilde{F}_I^{L^{-1}} = \sum_{s=1}^{N_s} W B^{(s)} \begin{bmatrix} 0 & 0 \\ O & K_{bb}^{(s)} \end{bmatrix} B^{(s)^T} W \qquad [17]$$

4. Rendement et extensibilité du couplage MAN et FETI

Dans ce paragraphe, on s'intéresse à l'efficacité parallèle du couplage MAN et d'une version de FETI sans stockage des directions de descente entre les systèmes linéaires. Dans un premier temps, on regarde les rendements obtenus avec cette association. La deuxième partie de ce paragraphe est, quant à elle, consacrée à la vérification de l'extensibilité parallèle du couple. Les résultats que l'on présente ici sont obtenus sur le Cray T3E de l'IDRIS, où l'on travaille en mode dédié ce qui assure d'avoir un processeur par tâche ou processus et donc par sous-domaine. Tous les tests effectués dans cette partie portent sur une poutre homogène encastrée en flexion. Ce problème est résolu en 2 dimensions (2D) sous l'hypothèse des contraintes planes. Les poutres sont maillées régulièrement par des éléments quadratiques à 8 nœuds et 2 ddls par nœud, Q8. Le coefficient de Poisson du matériau constituant les poutres est de 0.3 et le module d'Young de 200 000 MPa.

4.1. *Rendement du couplage*

Pour calculer le rendement, on travaille à taille de problème constante (55 322 ddls) et on augmente le nombre de sous-domaines de 4 à 22, figure 4. Lors des découpages, on essaie de garder le plus possible des sous-domaines de même taille pour favoriser l'équilibrage du travail à effectuer par chaque processeur.

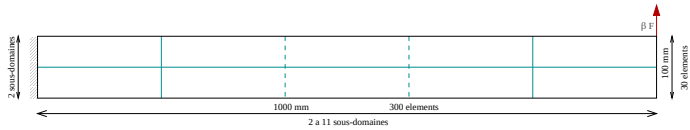

Figure 4. *Poutre encastrée avec un nombre constant de ddls*

Le problème est résolu avec un pas de MAN à l'ordre 20. On utilise dans FETI le préconditionneur optimal de Dirichlet et la précision demandée dans FETI est de 10^{-7}.

On donne dans le tableau 1, le nombre de ddls de chaque sous-structure, le nombre de ddls de l'interface, ainsi que le pourcentage d'interface par rapport à la structure initiale, la somme des temps CPU sur l'ensemble des processeurs, le temps dit « de restitution »qui représente la durée du calcul pour l'utilisateur, et enfin la taille mémoire maximale utilisée en million de mots de 64 bits(Mw).

Ns	Taille des des sous-domaines	Taille de l'inter-face (%)	Temps CPU total (s)	Temps de restitution (s)	Taille mémoire max. (Mw)
4	14 162	1 322 (2.4)	1 293	**339**	56.9
6	9 462	1 382 (2.5)	1 298	**229**	60.2
8	7 112	1 442 (2.6)	1 307	**175**	63.5
10	5 702	1 502 (2.7)	1 289	**140**	66.7
12	4 762	1 562 (2.8)	1 298	**120**	70.0
14	4 104	1 622 (2.9)	1 311	**106**	73.0
16	3 634	1 682 (3.0)	1 318	**95**	76.5
18	3 258	1 742 (3.1)	1 311	**85**	79.7
20	2 882	1 802 (3.2)	1 294	**77**	83.1
22	2 734	1 862 (3.4)	1 307	**74**	85.9

Tableau 1. *Résultats sur la poutre par rapport au nombre de sous-domaines*

On peut voir dans ce tableau que le temps CPU total est pratiquement constant, quel que soit le nombre de processeurs. Ceci montre que l'on garde un bon rendement pour le couplage MAN et FETI. Pour le vérifier, on montre les variations du rendement en fonction du nombre de processeurs sur la courbe de la figure 5, avec la définition suivante de rendement :

$$\text{Rendement} = \frac{\text{Temps de restitution pour 4 sous-domaines}}{\text{Temps de restitution pour N sous-domaines}} * \frac{4}{N} \qquad [18]$$

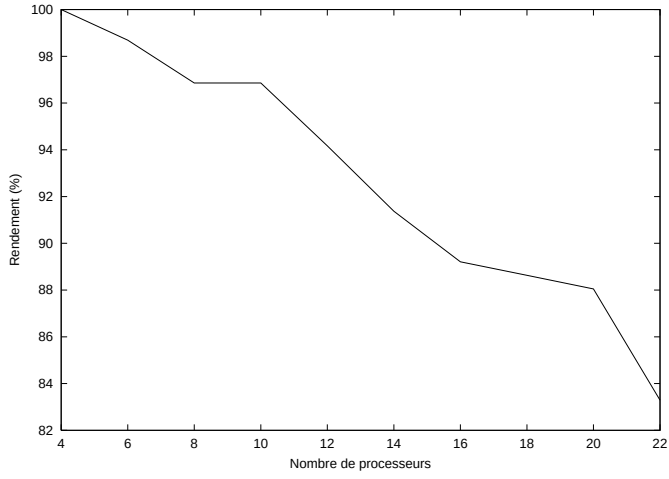

Figure 5. *Rendement du couple MAN et FETI*

On peut tout d'abord noter que le rendement reste très bon, supérieur à 83 %, même pour 22 processeurs. Deux raisons expliquent ces bons résultats. La première tient au bon rendement que l'on peut obtenir avec FETI, et la deuxième est que dans une MAN, seuls le calcul de quelques produits scalaires et la résolution des systèmes linéaires doivent faire l'objet d'un traitement parallèle particulier. En effet, le reste des calculs se faisant élément par élément, la parallélisation par sous-domaine donne sur ces parties des rendements de 100 %. Il faut cependant noter une augmentation de 51 % de la mémoire nécessaire pour résoudre le problème entre 4 et 22 processeurs, ce qui peut s'expliquer par l'augmentation de la taille de la frontière par rapport à la taille de chaque sous-domaine.

On vient de voir que l'association de la MAN et de FETI procurait un bon rendement parallèle, on va maintenant regarder comment se comporte cette association vis-à-vis de l'extensibilité parallèle.

4.2. *Extensibilité du couple MAN et FETI*

Pour tester l'extensibilité pratique du couple MAN+FETI, on garde une taille de sous-domaine constante (14 162 ddls), et on augmente le nombre de sous-domaines de 4 à 28, comme le montre la figure 6. Pour ces tests, la proportion de frontière globale par rapport au problème global est constante (2.9 %), ainsi que celle de frontière de chaque sous-domaine par rapport à la taille du sous-domaine. Pour résoudre ce problème, on effectue un pas de MAN à l'ordre 20. On utilise pour FETI le préconditionneur de DIRICHLET, et on demande une précision de 10^{-4}. On donne dans le tableau 2, le nombre de ddls du problème global ainsi que celui du problème d'interface. On précise aussi les temps de restitution, et le temps CPU passé à résoudre les

systèmes ainsi que le nombre total d'itérations nécessaires pour résoudre l'ensemble des 20 systèmes linéaires. On précise enfin la taille mémoire maximale utilisée pour résoudre ce problème.

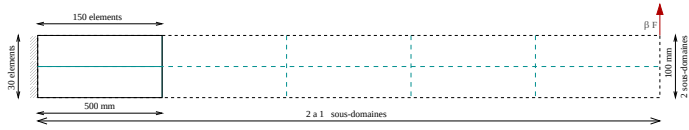

Figure 6. *Poutre encastrée avec taille des sous-domaines constante*

Ns	Taille problème	Taille interface	Temps de restitution	Temps FETI	Nombre itérations	Taille (Mw) mémoire
6	82 922	1 982	328	69.8	237	14.25
8	110 522	2 642	334	70.0	236	14.26
10	138 122	3 302	332	69.5	220	14.27
12	165 722	3 962	341	68.9	229	14.27
14	193 322	4 622	352	67.9	224	14.28
16	220 922	5 282	357	68.0	221	14.28
18	248 522	5 942	363	68.0	221	14.28
20	276 122	6 602	361	67.4	218	14.28
22	303 722	7 262	367	66.8	215	14.29
24	331 322	7 922	375	65.9	210	14.29
26	358 922	8 582	379	65.8	208	14.29
28	386 522	9 242	381	65.6	208	14.29

Tableau 2. *Résultats pour la poutre encastrée à taille des sous-domaines constante*

On peut observer dans ce tableau que le nombre d'itérations pour résoudre l'ensemble des 20 systèmes linéaires diminue de 12 % entre 6 et 28 sous-domaines. On retrouve cette même diminution sur le temps passé dans le solveur FETI. Ces résultats s'expliquent par l'extensibilité numérique de FETI. En revanche, on constate une augmentation de 20 % du temps de restitution des résultats, qui provient en fait d'une implémentation imparfaite du calcul des produits scalaires globaux dans la MAN. Enfin, on peut noter que la taille mémoire n'augmente pas, ce qui signifie que cette taille n'est pas influencée par la taille du problème d'interface global, mais par la taille de la frontière par rapport au nombre de ddls de chaque sous-domaine. Il faut noter qu'ici, ce rapport est constant.

5. Efficacité de la réutilisation des directions de descente

Dans cette partie, on va montrer les apports de la version de FETI dite « pour seconds membres répétés ». L'idée directrice est de conserver les directions de descente

obtenues lors des permières résolutions pour résoudre, plus rapidement, les systèmes suivants. Compte tenu des propriétés de colinéarité des second membres f_p^{nl} qui interviennent dans la MAN, on peut s'attendre à une certaine efficacité de ces techniques.

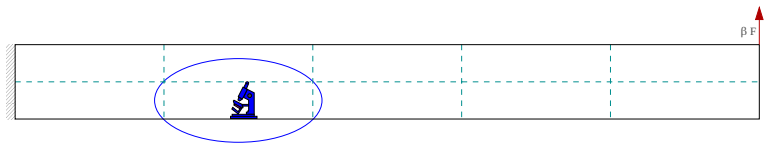

Figure 7. *Poutre hétérogène en flexion*

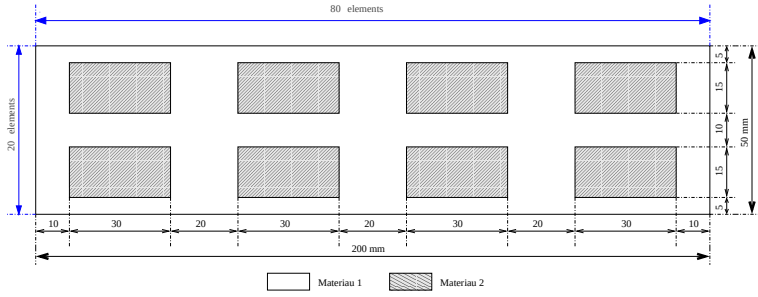

Figure 8. *Motif de la poutre hétérogène*

L'exemple étudié dans cette partie est toujours celui d'une poutre encastrée en flexion, mais cette fois elle est hétérogène, (figures 7 et 8). Cette poutre est maillée régulièrement avec 16 000 éléments de type Q8. Elle a été découpée en 10 sous-domaines. En ce qui concerne les caractéristiques mécaniques de la poutre, le coefficient de Poisson des deux matériaux est de 0.3, le module d'Young du matériau 1 est de 200 000 MPa (acier), et celui du matériau 2 prend les valeurs suivantes pour faire varier le niveau d'hétérogénéité :

– 200 000 MPa (poutre homogène),

– 2 000 MPa (poutre moyennement hétérogène),

– 10 MPa (poutre extrêmement hétérogène).

On résout ce problème avec 1 pas de MAN à l'ordre 20. Ainsi, on doit résoudre avec FETI une série de 20 systèmes linéaires à matrice invariante. On résout ces systèmes avec un préconditionneur de Dirichlet, sans ajout de problèmes grossiers supplémentaires. La précision demandée pour le gradient conjugué est de 10^{-7}.

Sur cet exemple, on compare le nombre d'itérations, la place mémoire et le temps CPU pour différentes valeurs du nombre maximum de vecteurs de direction de descente stockés entre les systèmes. Lorsqu'on écrit 0 direction stockée cela signifie que l'on stocke les directions de descente au cours de la résolution d'un système linéaire

pour faire de la réorthogonalisation interne complète, mais qu'on ne les garde pas entre les différents systèmes.

Remarque : Il faut noter que dans tous les exemples traités, on stocke indifféremment toutes les directions de descente. Pour optimiser le nombre de directions à stocker, il faudrait connaître l'influence du stockage d'une partie seulement de ces directions pour chaque système. Par exemple, il pourrait être décidé de ne stocker que les premières ou les dernières directions.

5.1. *Influence sur le nombre d'itérations*

On donne dans le tableau 3, le nombre total d'itérations nécessaires pour résoudre les 20 systèmes linéaires, pour les 3 cas d'hétérogénéité, selon le nombre de directions de descente stockées. On remarque que pour le problème homogène, on a une forte diminution du nombre d'itérations (66.6 %) dès que l'on stocke 100 directions de descente. Il n'est donc pas intéressant de stocker plus de directions de descente. **Il faut noter que la résolution des 5 derniers systèmes ne demande au maximum qu'une itération pour chaque système**. Pour les deux autres cas, pour lesquels les matrices sont moins bien conditionnées, on arrive à une diminution de plus de 55 %. Notons que dans le dernier cas, on pourrait diminuer le nombre d'itérations car celui-ci est encore supérieur au nombre de directions de descente stockées.

Young 2 (MPa)	200 000	2 000	10
0 direction	368	461	468
50 directions	182	232	259
Gain(%)	50.5	49.7	44.7
100 directions	123	217	236
Gain(%)	66.6	52.9	49.6
150 directions	123	199	213
Gain(%)	66.6	56.8	54.5
200 directions	123	198	210
Gain(%)	66.6	57.0	55.1

Tableau 3. *Nombre total d'itérations en fonction du nombre de directions de descente stockées*

Pour la suite, on voudrait pouvoir automatiser le choix du nombre de directions de descente à stocker. Mais ce nombre, comme on vient de le voir dépend fortement du problème à résoudre. Dans la littérature, on trouve souvent que le nombre de directions de descente à stocker dépend de la place mémoire dont on dispose. C'est pour cette raison que l'on va maintenant regarder la place mémoire maximale utilisée pour résoudre nos problèmes.

5.2. *Influence sur la taille mémoire*

On reporte dans le tableau 4 la place mémoire, en million de mots de 64 bits (Mwords), que l'on utilise au maximum pour effectuer les calculs. On peut voir qu'il y a une augmentation de 2,4 % entre la version sans réorthogonalisation complète et avec, ce qui reste faible. Par contre, il n'y a plus qu'une augmentation de 0,8 % entre une version avec réorthogonalisation complète et une version avec 200 directions de descente stockées entre les systèmes linéaires, ce qui est presque négligeable. Cette faible augmentation de la place mémoire provient de la petite dimension du problème d'interface (2 000 ddls).

	Mwords	Augmentation(%)
Sans réorthogonalisation	106.2	
0 direction	108.7	2.4
50 directions	108.9	2.5
100 directions	109.1	2.7
150 directions	109.3	2.9
200 directions	109.5	3.1

Tableau 4. *Mémoire maximum utilisée (Mwords) avec ou sans stockage des directions de descente*

Finalement, pour notre problème, la place mémoire supplémentaire due à l'utilisation de la version de FETI dite « pour seconds membres répétés »ne représente pas une restriction. Ainsi, on ne s'appuie pas sur le critère de la place mémoire disponible pour déterminer automatiquement le nombre de directions de descente à stocker, mais plutôt l'expérience tirée des exemples que l'on a traités et sur la remarque suivante : les seconds membres f_p^{nl}, p=2 à N se situent dans un espace de faible dimension, mais le premier second membre F_{ext} n'est en général pas dans cet espace.

Ainsi en pratique, **on décide de ne pas conserver les vecteurs de direction de descente liés à la résolution du premier système**. Cependant, on se sert du nombre d'itérations du premier système pour déterminer le nombre de directions de descente que l'on va stocker. Ainsi, on a choisi, à la vue de quelques exemples, de conserver un nombre de directions de descente égal à **6 fois le nombre d'itérations nécessaires pour atteindre la convergence dans le premier système**.

5.3. *Influence sur le temps de calcul*

Avant de donner les gains de temps obtenus avec l'utilisation de la version de FETI dite « pour seconds membres répétés », on montre sur le tableau 5 la répartition du temps CPU sur un des processeurs entre les trois grandes parties du code. Ces trois parties correspondant au temps nécessaire pour le calcul et l'assemblage des vecteurs seconds membres f_p^{nl} , le temps dit de factorisation, *i.e.* le temps de calcul de $K^{(s)^+}$

et de $R^{(s)}$, et enfin le temps passé dans toutes les itérations du gradient conjugué de FETI. On rappelle ici que l'étape de factorisation n'est réalisée qu'une seule fois, puisque la matrice est invariante pour les 20 systèmes. Les méthodes d'accélération de convergence du gradient conjugué de type Bloc-Krylov ne peuvent évidemment diminuer que le temps de calcul nécessaire pour réaliser les itérations. On peut voir sur le tableau que sur cet exemple à 30 000 ddls ces itérations n'occupent que la moitié du temps de calcul.

Young 2 (MPA)	200 000	2 000	10
Scd membres	51%	47%	46%
Factorisation	5%	4%	4%
Itérations	44%	49%	50%

Tableau 5. *Répartition des temps CPU de calcul sur un processeur*

Une des différences essentielles entre les méthodes incrémentales classiques et la MAN réside dans le calcul des seconds membres. En effet, pour une méthode incrémentale classique on a beaucoup de matrices à inverser mais le calcul des seconds membres est immédiat, alors que pour une MAN, on a très peu de matrices à décomposer (une par pas) mais on a de nombreux seconds membres à calculer, ici 20 par pas. De plus, le calcul de ces derniers est plus coûteux que le calcul d'un vecteur résidu d'équilibre. Ainsi, on peut voir dans le tableau 5, que 50 % du temps est passé dans le calcul des seconds membres. Avec la version séquentielle du code, équipée d'un solveur direct de type CROUT standard, on montre que le temps CPU passé dans la formation des 20 seconds membres devient faible devant la décomposition de la matrice tangente dès que l'on dépasse les 10 000 ddls. Ici, les bonnes performances de FETI font que le temps CPU passé dans la formation des seconds membres est équivalent à celui des résolutions même pour 30 000 ddls. Avec cette version, il faut prendre un nombre de ddls beaucoup plus important pour que le temps CPU des seconds membres devienne petit devant celui des résolutions.
On notera cependant que, pour des questions de lisibilité et de généralité du code, on n'a pas cherché à pousser trop loin l'optimisation de la formation des seconds membres.

Finalement, on donne dans le tableau 6, le temps CPU, en seconde, nécessaire à un processeur pour réaliser toutes les itérations en fonction du nombre de directions de descente conservées. On peut voir sur ce tableau que l'on peut gagner au maximum entre 50,4 et 59,3 % sur le temps CPU, contre 55,1 à 66,6 % pour les itérations. Cette différence, due à l'augmentation des coûts d'initialisation et de réorthogonalisation, est faible. Ceci montre que le gain de temps obtenu est bon. Cette technique risque de devenir encore plus intéressante en dynamique non linéaire pour la MAN, pour laquelle le nombre de seconds membres atteint plusieurs centaines.

Young 2 (MPA)	200 000	2 000	10
0 direction	79.2	98.4	99.3
50 directions	44.1	53.4	59.5
Gain(%)	44.3	45.7	40.1
100 directions	32.3	50.6	54.4
Gain(%)	59.2	48.6	45.2
150 directions	32.2	47.1	50.4
Gain(%)	59.3	52.1	49.2
200 directions	32.4	46.9	49.3
Gain(%)	59.1	52.3	50.4

Tableau 6. *Temps CPU (s) pour les itérations en fonction du nombre de directions de descente stockées*

5.4. *Conclusion sur l'utilisation de la version de FETI pour seconds membres répétés*

Finalement, les résultats obtenus montrent que l'utilisation de la version de FETI dite pour seconds membres répétés, basée sur les méthodes de type Bloc-Krylov, sont efficaces pour les systèmes linéaires à matrice invariante issus des MAN. Certes, le coût de calcul des séries n'est plus de l'ordre de 1 à 2 fois celui d'un calcul élastique, comme c'est le cas en scalaire avec un solveur direct. Il faut cette fois plutôt de 5 à 10 fois le temps d'un calcul élastique pour aller à l'ordre 20. En revanche, on conserve la bonne propriété que pour aller de l'ordre 20 à l'ordre 30, les dix résolutions supplémentaires sont rapides car elles ne demandent de rajouter que quelques directions de descente seulement. L'impact sur le temps total dépend alors du ratio entre le temps de formation des seconds membres et le temps passé dans les itérations. **Il faut noter que sans la conservation des directions de descente, la dépendance du temps de calcul avec l'ordre N de troncature des séries serait beaucoup plus marquée** .

6. Influence de la précision du solveur itératif

Lorsque les systèmes linéaires issus d'une MAN sont résolus par un solveur direct de type CROUT, la précision obtenue est en rapport direct avec celle de la machine. Dans ce travail, où on utilise un solveur itératif pour le problème d'interface, la précision de résolution des systèmes linéaires est un paramètre ϵ fixé par l'utilisateur. On regarde ici l'influence de ce paramètre sur la qualité des séries[3]. Pour cela, on s'intéresse à l'évolution du vecteur résidu $R(U,\beta)$ avec le paramètre de chemin a. Pour les séries [3] tronquées à l'ordre N, on a :

$$R(\sum_{i=0}^{N} a^i U_i, \sum_{i=0}^{N} a^i \beta_i) = R_0 + aR_1 + \ldots + a^N R_N + a^{N+1} R_{N+1} + \ldots + a^{2N} R_{2N}. \quad [19]$$

où R_0 est le résidu au point de départ, et les vecteurs R_i pour $i = 1, .., N$ représentent le résidu des N systèmes linéaires. La norme de ces vecteurs depend directement du paramètre de précision ϵ du solveur itératif.

Pour illustrer, on reprend l'exemple de la poutre hétérogène en flexion découpée en 10 sous-domaines. Les caractéristiques du maillage de la poutre sont résumées dans le tableau 7. On choisit le cas fortement hétérogène avec $E_2 = 10MPa$. Pour résoudre ce problème, on applique un pas MAN à l'ordre 20. On reporte sur la figure 9 le logarithme de la norme du résidu obtenu en fonction du paramètre de chemin a. Pour ce calcul, on utilise dans FETI le préconditionneur optimal de Dirichlet, et on ne fait aucune réorthogonalisation des directions de descente lors du gradient conjugué. On donne à titre indicatif le nombre d'itérations, ainsi que le temps CPU(s) de FETI suivant la précision demandée dans le tableau 6.

	Structure	Sous-structures	Interface
Eléments	16 000	1 600	
ddls	97 762	10 002	2 242 (2.3 %)

Tableau 7. *Caractéristiques de la discrétisation par éléments finis*

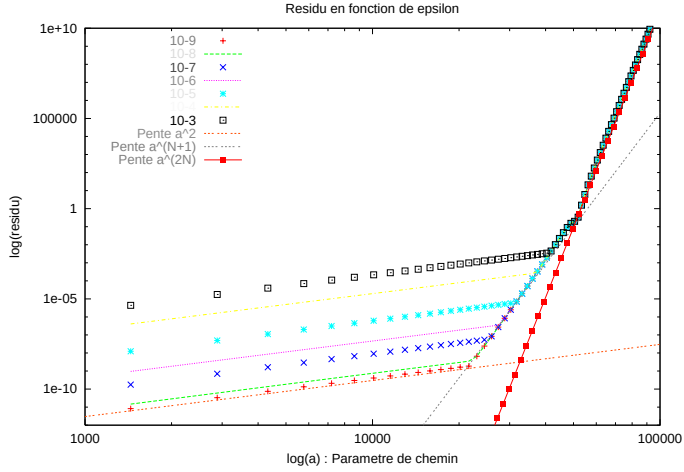

Figure 9. *Evolution du résidu de la MAN en fonction de la précision demandée pour les systèmes linéaires*

On remarque tout d'abord que chaque courbe de la figure 9 présente trois pentes différentes. Ceci s'explique par le fait que suivant la valeur du paramètre a ce sont les termes aR_1, $a^{N+1}R_{N+1}$ ou $a^{2N}R_{2N}$ qui sont prépondérants dans l'expression du résidu. On peut remarquer sur cet exemple que le paramètre de précision ϵ n'a pour effet que de reduire la qualité des séries à l'intérieur du rayon de convergence des

séries. On en tire donc la règle suivante [GAL 00] : si on souhaite obtenir une branche de solution [3] avec une tolérance donnée sur le vecteur résidu $R(U, \beta)$, disons par exemple $\|R\| < 10^{-4}$, alors, il est suffisant de résoudre les systèmes linéaires avec une précision ϵ équivalente à cette tolérance. En pratique, on prend une précision légèrement supérieure, mettons par exemple 10^{-5}, pour plus de sécurité. Autrement dit, il n'est donc pas nécessaire de forcer la précision lors de la résolution des systèmes linéaires, les erreurs commises n'étant pas amplifiées lors de la formation et de la résolution des systèmes linéaires suivants.

ϵ	Itérations	Temps CPU de FETI(s)
10^{-3}	18	85.7
10^{-4}	23	105.9
10^{-5}	27	119.3
10^{-6}	32	138.6
10^{-7}	36	153.1
10^{-8}	42	178.4
10^{-9}	50	209.0
10^{-10}	N'atteind pas la précision demandée	

Tableau 8. *Influence de la précision demandée pour le solveur itératif sur la convergence de FETI*

7. Conclusion

Dans ce papier, nous avons utilisé la méthode FETI dans sa version dite « pour seconds membres répétés »pour implémenter la MAN sur une machine parallèle. Les rendements et l'extensilité que nous avons obtenus avec ce couple FETI + MAN sont tout à fait satisfaisants. Nous avons également mis en évidence que les techniques de conservation des directions de descente introduites pour accélérer la convergence des dernières résolutions sont efficaces. Elles permettent d'aller à des ordres de troncature élevés sans trop aggraver les temps de calcul. On tend alors à retrouver avec un tel solveur itératif, la même propriété que l'on a avec un solveur direct, à savoir que le temps de calcul des séries ne dépend par trop fortement de l'ordre de troncature N. Enfin, nous avons analysé l'influence de la précision du solveur itératif, et montré que ce paramètre pouvait être facilement choisi en accord avec la tolérance demandée sur le résidu de la branche de solution.

Le code de calcul développé permet de faire de la continuation de branches de solutions sur des problèmes de structures minces en non linéaire géométrique. Quelques benchmarks standard du domaine sont présentés dans [GAL 00]. On notera toutefois que pour ces calculs, la matrice F_I du problème d'interface n'est plus en général définie positive. Nous renvoyons à [GAL 00] et [FAR 00] pour une discussion sur ces aspects.

8. Bibliographie

[AZR 93] AZRAR L., COCHELIN B., DAMIL N., POTIER-FERRY M., « An asymptotic numerical method to compute the post-buckling behavior of elastic plates and shells », *International Journal for Numerical Methods in Engineering*, vol. 36, 1993, p. 1251-1277.

[COC 94] COCHELIN B., DAMIL N., POTIER-FERRY M., « The Asymptotic-Numerical-Method : an efficient perturbation technique for nonlinear structural mechanics », *Revue Européenne des Éléments Finis*, vol. 3, 1994, p. 281-297.

[FAR 91] FARHAT C., ROUX F. X., « A method of finite element tearing and interconnecting and its parallel solution algorithm », *International Journal for Numerical Methods in Engineering*, vol. 32, 1991, p. 1205-1227.

[FAR 94a] FARHAT C., CRIVELLI L., ROUX F. X., « Extending substructure based iterative solvers to multiple loads and repeated analysis », *Computer Methods in Applied Mechanics and Engineering*, vol. 117, 1994, p. 195-209.

[FAR 94b] FARHAT C., ROUX F. X., « Implicit parallel processing in structural mechanics », *Computational Mechanics Advances*, vol. 2, 1994, p. 47-72, N° 1.

[FAR 00] FARHAT C., PIERSON K., LESOINNE M., « The second generation FETI methods and their application to parallel solution of large-scale linear and geometrically non-linear structural analysis problems », *Computer Methods in Applied Mechanics and Engineering*, vol. 184, 2000, p. 333-374.

[GAL 00] GALLIET I., « Une version parallèle des méthodes asymptotiques numériques. Application à des structures complexes à base d'élastomères », Mémoire de Thèse de Doctorat, École Supérieure de Mécanique de Marseille, Université d'Aix-Marseille II, 2000.

[NAJ 98] NAJAH A., COCHELIN B., DAMIL N., POTIER-FERRY M., « A critical review of Asymptotic Numerical Methods », *Archives of Computational Methods in Engineering*, vol. 5, 1998, p. 31-50.

[NOO 80] NOOR A., PETERS J., « Reduced basis technique for non linear analysis of structures », *AIAA*, vol. 18, 1980, p. 455-462.

[REY 96] REY C., LÉNÉ F., « Reuse of Krylov spaces in the solution of large scale nonlinear elasticity problems », *Ninth international Conference on Domain Decomosition Methods*, Norway, June 1996.

[ZAH 99] ZAHROUNI H., COCHELIN B., POTIER-FERRY M., « Asymptotic Numerical Method for shells with finite rotation », *Computer Methods in Applied Mechanics and Engineering*, vol. 175, 1999, p. 271-285.