
Equilibrage en volume de calcul pour une méthode parallèle à fronts multiples

Jean-Paul Boufflet * — **Piotr Breitkopf****
Christophe Denis *** — **Michel Vayssade****

* *Laboratoire Heudiasyc*

** *Laboratoire de Recherche en Mécanique Roberval*

Université de Technologie de Compiègne

F-60200 Compiègne

*** *Laboratoire d'Informatique LIP6*

Université Pierre et Marie Curie

8 rue du Capitaine Scott

F-75015 Paris

{Jean-Paul.Boufflet,Piotr.Breitkopf,Michel.Vayssade}@utc.fr

Christophe.Denis@lip6.fr

RÉSUMÉ. Nous utilisons une méthode parallèle à fronts multiples pour traiter de grands systèmes linéaires issus de la modélisation par éléments finis de problèmes de mécanique. Ce solveur direct est basé sur la méthode des compléments de Schur et utilise une approche par décomposition de domaine. Nous observons expérimentalement que le temps de calcul des sous-domaines équilibrés en volume de données peut varier du simple au double pour des sous-domaines de même taille. Nous explorons une stratégie d'équilibrage de charges qui utilise un modèle du comportement algorithmique de notre solveur pour corriger une partition initiale et aboutir à un équilibrage en volume de calculs.

ABSTRACT. We use a parallel multiple front method for solving large sparse linear systems issued from finite element modeling. This direct solver is based on the Schur complement method and uses a domain decomposition approach. We experimentally observe that the computing time over the subdomains may vary from simple to double for equal size subdomains. We investigate a load balancing strategy that uses a model of the computational behavior of our solver to improve an initial partition. The proposed heuristics balances estimated number of operations rather than the amount of data.

MOTS-CLÉS : décomposition de domaine, équilibrage en volume de calcul, solveur multifrontal.

KEYWORDS: domain decomposition, load balancing on amount of operations, multifrontal solver.

1. Introduction

Le traitement de grands systèmes linéaires creux est consommateur de temps de calcul et de nombreuses techniques parallèles existent pour traiter ce problème. L'approche frontale a été initialement proposée dans le contexte de la modélisation par éléments finis (Irons, 1970). Elle entrelace des opérations d'assemblage et d'élimination au sein d'une matrice frontale qui occupe un espace mémoire réduit par rapport à la matrice globale du système. Deux types d'assemblages partiels de la matrice frontale peuvent être implémentés. Dans le premier, l'opération d'assemblage est réalisée sur des matrices élémentaires en format compact, à chacune est associée une liste de variables. Dans le second, la matrice globale est stockée au format assemblé, des lignes et colonnes entières sont incorporées successivement dans la matrice frontale. Dans les deux cas, une variable est éliminée quand les équations où elle apparaît sont complètement sommées. À ce niveau, un complément de Schur est formé et une nouvelle opération d'assemblage peut être réalisée pour construire la prochaine matrice frontale. De plus amples détails sont donnés dans (Duff *et al.*, 1986) et (Duff *et al.*, 1993).

Un solveur frontal peut être parallélisé :

- en concevant un code spécifique qui traite directement le partitionnement, la renumérotation, l'équilibrage statique et dynamique de charge, l'affectation de ces tâches aux processeurs et la résolution ;
- en effectuant les tâches de partitionnement, de renumérotation, d'équilibrage statique, et de distribution des données dans des modules indépendants et en réutilisant un solveur séquentiel existant.

Le code présenté dans (Amestoy *et al.*, 2001) relève de la première approche. Jennifer Scott expose dans (Scott, 1999) la conception du solveur *MP42*. Ce code est basé sur les modifications et adaptations du code frontal développé par I. Duff et J. Reid (Duff *et al.*, 1993) et relève de la seconde approche. Dans le même esprit, nous utilisons une implémentation d'une méthode parallèle à front multiple développée dans le cadre du logiciel académique *SIC* (Système Interactif de Conception) développé au sein de la communauté de mécanique numérique (Escaig *et al.*, 1994), (Escaig *et al.*, 1997), (Breitkopf *et al.*, 1998). Les matrices élémentaires sont assemblées au sein d'une matrice frontale. Dès que les équations associées à une variable sont complètement sommées, elle est éliminée de la matrice frontale. Un des critères de la décomposition consiste à obtenir des partitions qui permettent de traiter le problème local dans la mémoire vive de chaque processeur.

La partition initiale du domaine en sous-domaines peut être obtenue en utilisant des outils standard *PARTY* (Preis, 1998), *JOSTLE* (Walshaw, 2000), *METIS* (Karypis *et al.*, 1998) *SCOTCH* (Pellegrini, 1997) et *CHACO* (Hendrickson, 1995), basés sur des techniques de partitionnement de graphes. La donnée d'entrée est un graphe qui représente le maillage par éléments finis associé au domaine. La partition est composée d'un ensemble de sous-graphes partiels, chacun d'eux correspondant à un sous-domaine. À cette étape le but est de minimiser les communications et d'équilibrer les

volumes de données. Cette approche est induite par les deux hypothèses suivantes :

- le coût de calcul est proportionnel au nombre de sommets du sous-graphe partiel (nombre d'éléments finis du sous-domaine) ;
- l'ordre de traitement de ces sommets n'influence pas le calcul.

Nous avons expérimentalement observé que les temps d'assemblage/élimination peuvent significativement varier entre les sous-domaines. Pour remédier à ce problème nous avons dans un premier temps exploré des techniques de renumérotation (Boufflet *et al.*, 2000), (Boufflet *et al.*, 2001). Le temps de calcul sur chaque sous-domaine a été réduit mais le déséquilibre était toujours observé. Cette expérience pratique semble confirmer l'analyse présentée par Bruce Hendrickson dans (Hendrickson, 1998) et (Hendrickson, 2000) : répartir équitablement les volumes de données n'induit pas systématiquement des temps de calcul équilibrés.

Dans cet article, nous développons une heuristique qui corrige une partition initiale de façon à équilibrer les efforts de calcul. Nous utilisons *METIS* pour construire cette partition initiale. Ensuite, un estimateur du nombre d'opérations d'une tâche de factorisation de notre solveur est utilisé pour évaluer le volume de calcul de chaque sous-domaine. Dans la phase d'équilibrage nous utilisons ces estimations pour décider du transfert d'éléments finis entre les sous-domaines. Nous obtenons ainsi des partitions équilibrées en volume de calculs estimé. Enfin, nous validons la démarche en mesurant les temps de calcul réels.

Nous utilisons comme jeu test un ensemble de maillages issus de modélisations par éléments finis (Rassineux, 2004) et des données du projet (*PARASOL*, 2003). Compte tenu des résultats obtenus sur ce jeu test avec l'implémentation du solveur parallèle à front multiple de *SIC*, il semble intéressant d'équilibrer une partition en tenant compte des nombres d'opérations sur les sous-domaines plutôt que des volumes de données.

Les motivations de l'approche proposée sont décrites dans la section 2. Dans la section 3, nous présentons une synthèse de l'approche à front multiple et un modèle du solveur, ce qui nous permet d'introduire l'estimation du coût algorithmique des tâches de calcul. Dans la section 4, nous exposons le principe de l'heuristique qui utilise les estimations des volumes d'opérations des tâches pour corriger une partition initialement équilibrée en volume de données. Nous détaillons dans la section 5 le protocole expérimental et les critères retenus pour mesurer l'intérêt de ce type d'approche. Les résultats expérimentaux sont présentés dans la section 6.

2. Motivations

Ces travaux s'inscrivent dans la suite des travaux présentés dans (Escalaig, 1992), (Negre, 1997) et (Lassignardie, 2001). L'expérience pratique avec des méthodes à front multiple montre qu'un comportement général peut être observé indépendamment du type de maillage utilisé.

Les figures 1 et 2 ont pour objet de présenter une synthèse des phénomènes observés. Il ne s'agit pas d'un exemple pour un maillage particulier. Des exemples réels sont présentés dans (Denis, 2003).

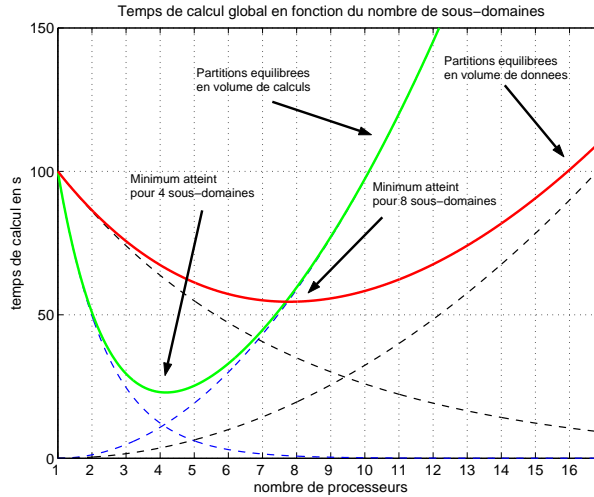


Figure 1. Temps de calcul global, représenté par les courbes pleines, en fonction du nombre de sous-domaines. Les courbes pointillées montrent l'évolution des deux fonctions qui composent le temps de calcul global. Il s'agit de l'approche mono-niveau, chaque processeur traite un seul sous-domaine et un unique problème interface est construit puis résolu

La figure 1 montre l'évolution des temps de calculs globaux en fonction du nombre de sous-domaines N_s pour un maillage. À l'issue des factorisations partielles de chaque sous-domaine nous obtenons des condensées locales qui sont vues comme les matrices élémentaires de super-éléments. On considère qu'un unique problème interface est construit puis résolu en utilisant le solveur frontal. Il s'agit d'une approche mono-niveau. Les super-éléments sont assemblés, puis le système obtenu est factorisé, enfin la restitution des inconnues internes est effectuée en parallèle. Jennifer Scott (Scott, 1999) signale l'augmentation rapide de la place mémoire et du temps de calcul pour traiter le problème interface quand le nombre de sous-domaines augmente. Nous observons ces mêmes phénomènes en utilisant cette approche mono-niveau. La figure 1 permet de visualiser les phénomènes et montre les avantages escomptés d'un équilibre en volume de calcul. Nous affectons un seul sous-domaine par processeur. Nous avons reporté la tendance de l'évolution des temps de calculs en fonction du nombre de sous-domaines/processeurs pour des partitions équilibrées en volume de données et pour ces mêmes partitions corrigées en volume de calculs.

Le temps de calcul global peut se décomposer comme la somme de deux fonctions. La première représente le temps de calcul du sous-domaine ayant le plus gros volume

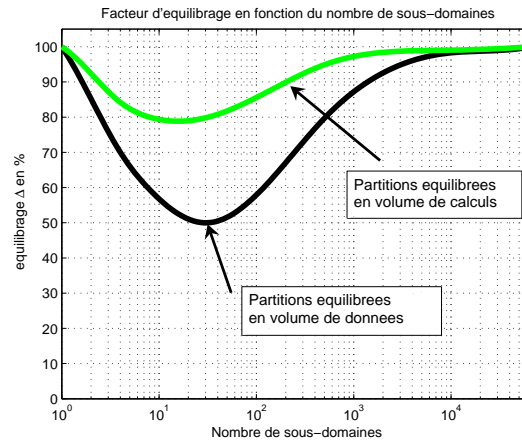


Figure 2. Facteur d'équilibrage Δ^{mesu} en fonction du nombre de sous-domaines

de calcul. Le volume de calcul correspond à la factorisation partielle du sous-domaine et à la restitution des degrés de libertés internes qui fait suite à la résolution du problème interface. Elle est strictement décroissante (courbes pointillées décroissantes sur la figure 1). En effet, quand le nombre de sous-domaines augmente le nombre d'opérations pour factoriser partiellement un sous-domaine diminue. Il en est de même pour la restitution. Le temps de factorisation partielle du sous-domaine ayant le plus gros volume de calcul induit *de facto* la barrière de synchronisation pour le traitement du problème interface. À l'extrême limite nous avons un seul élément fini par sous-domaine et par processeur. Dans ce cas, la partition est parfaitement équilibrée mais aucun calcul ne peut être effectué sur le processeur : toute la quantité de calcul est en fait reportée sur l'unique problème interface.

La seconde fonction, représentée par des courbes pointillées strictement croissantes sur la figure 1, correspond à la somme des temps de :

- communications ;
- construction du problème interface ;
- résolution du problème interface.

Pour un même maillage, le temps de calcul global est la somme de ces deux fonctions. Il existe donc un temps de calcul global minimum.

Sur la figure 1, dans le cas de partitions équilibrées en volume de données, nous observons un temps de calcul minimum avec 8 processeurs. Les partitions corrigées en volume de calculs permettent d'obtenir le minimum pour 4 processeurs seulement. C'est ce que nous observons généralement : les partitions corrigées donnent un meilleur gain. Cependant, comme on peut l'observer sur la figure 1, la courbe

avec des partitions optimisées en volume de calculs croît beaucoup plus vite que la courbe avec des partitions équilibrées en volume de données. Pour un nombre fixé de sous-domaines, les outils standard de partitionnement minimisent leur objectif principal, la taille du problème interface, tout en équilibrant les volumes de données. Notre objectif principal est d'équilibrer les volumes de calculs à partir d'une décomposition équilibrée en volume de données tout en limitant l'augmentation du problème interface. Nous observons généralement une taille plus importante pour le problème interface d'une partition équilibrée en volume de calculs pour un même nombre de sous-domaines. La croissance de la courbe peut être atténuée en traitant le problème interface en multiniveau.

La figure 2 montre l'évolution du facteur d'équilibrage Δ^{mesu} en fonction de $\text{Log}(N_s)$. Δ^{mesu} fournit un indicateur de l'équilibrage en volume de calculs et vaut 100 % si les volumes de calculs effectués lors des factorisations partielles des sous-domaines sont égaux. Le premier et le dernier point de ce type de courbe sont connus. Le premier correspond au domaine initial traité sur un processeur. L'équilibrage est donc $\Delta^{mesu} = 100\%$. Le dernier correspond à un nombre de sous-domaines égal au nombre d'éléments finis du maillage donc $\Delta^{mesu} = 100\%$. Dans ce cas, le processeur n'effectue aucun travail de factorisation. Après avoir atteint un minimum pour un nombre relativement faible de processeurs, les deux courbes de la figure 2 atteignent rapidement des valeurs de facteurs d'équilibrage proche de 100 %. À partir de ces minimums, les écarts de volume de calculs s'atténuent très vite à mesure que le nombre de sous-domaines augmente.

Les deux courbes présentent l'objectif de l'approche : en corrigeant des partitions nous voulons nous rapprocher d'un équilibre parfait ce qui se traduira par l'applatissage de la courbe pour des partitions équilibrées en volume de données (figure 2) et par conséquent le temps de calcul global sera amélioré. Nous observons sur le début de la courbe pour des partitions équilibrées en volume de calculs que nous améliorons sensiblement l'équilibrage.

Le travail présenté est motivé par les conclusions suivantes :

- le minimum de la courbe de la figure 1 est obtenu avec un faible nombre de sous-domaines ;
- l'amélioration de l'équilibrage (figure 2) permet de diminuer la valeur du minimum de la courbe en trait plein pour des partitions équilibrée en volume de données de la figure 1 en le décalant vers la gauche, donc vers un nombre de sous-domaines réduit ;
- les gains potentiels sur la courbe d'équilibrage (figure 2) diminuent quand le nombre de sous-domaines augmente ;
- pour un faible nombre de sous-domaines les temps de communication deviennent insignifiants vis-à-vis des temps de factorisation.

Ces observations justifient le choix de l'architecture matérielle visée et montre qu'un nombre important de processeurs n'est pas nécessaire dans ce contexte. La dé-

marche présentée dans la suite de ce papier est adaptée à des clusters de PC de petite taille.

Dans nos expérimentations nous avons utilisé un cluster d'une dizaine de machines.

3. Modélisation et formulation

Nous considérons le système linéaire $K \cdot u = f$. Nous supposons que la matrice K n'est pas singulière. Il n'est pas dans notre intention de décrire en détail une méthode à front multiple. Une présentation exhaustive a été donnée par (Duff *et al.*, 1986) et (Duff *et al.*, 1993). Nous présentons une synthèse qui nous permet d'identifier les tâches de base d'une méthode parallèle à front multiple. Puis nous présentons une modélisation de l'organisation des calculs effectués au sein du solveur de *SIC*. Nous exposons ensuite l'estimateur du volume d'opérations d'une tâche algorithmique de factorisation partielle d'un sous-domaine.

Le maillage est modélisé à l'aide du graphe G_{elem} des éléments finis. Nous utilisons une décomposition de domaine sans recouvrement. Les principales étapes qu'effectue le solveur sont :

- 1) partitionnement du graphe G_{elem} en N_s sous-domaines $SD^{(j)}$;
- 2) factorisation partielle en parallèle de chaque sous-domaine $SD^{(j)}$;
- 3) construction, puis résolution du problème interface ;
- 4) restitution des inconnues internes en parallèle de chaque sous-domaine.

Nous pouvons résumer le processus en utilisant pour les besoins de l'explication la matrice entièrement assemblée K . Nous la réordonnons par bloc, sous-domaine par sous-domaine, et nous présentons le système

$$\begin{pmatrix} K_{ii}^{(1)} & 0 & \cdots & 0 & K_{ib}^{(1)} \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & K_{ii}^{(N_s)} & K_{ib}^{(N_s)} \\ K_{bi}^{(1)} & \cdots & \cdots & K_{bi}^{(N_s)} & K_{bb} \end{pmatrix} \cdot \begin{pmatrix} u_i^{(1)} \\ \vdots \\ \vdots \\ u_i^{(N_s)} \\ u_b \end{pmatrix} = \begin{pmatrix} f_i^{(1)} \\ \vdots \\ \vdots \\ f_i^{(N_s)} \\ f_b \end{pmatrix} \quad [1]$$

avec

$$K_{bb} = \sum_{j=1}^{N_s} K_{bb}^{(j)} \quad [2]$$

L'indice i désigne des variables internes d'un sous-domaine et b les variables à l'interface avec d'autres sous-domaines. Un bloc $K_{ii}^{(j)}$ est le résultat de l'assemblage des contributions des nœuds internes du sous-domaine $SD^{(j)}$. Les termes d'un bloc

$K_{bi}^{(j)}$ (resp. $K_{ib}^{(j)}$) correspondent aux interactions entre les variables internes d'un sous-domaine $SD^{(j)}$ et les variables frontières. Le bloc K_{bb} correspond à l'assemblage des contributions de toutes les variables frontières [2]. C'est le seul bloc qui ne peut être factorisé avant l'assemblage de tous les sous-domaines.

En appliquant la méthode des sous-structures, présentée dans (Duff *et al.*, 1986) et (Escaig *et al.*, 1994), nous allons traiter le système [1] en factorisant partiellement des systèmes linéaires associés aux sous-domaines puis nous obtenons les inconnues u_b du problème interface à partir de

$$S \cdot u_b = \hat{f}_b \quad [3]$$

où S est la matrice du complément de Schur de la matrice initiale [1] et \hat{f}_b vecteur second membre correspondant.

Considérons d'abord un sous-domaine $SD^{(j)}$ et réordonnons en premier ses variables internes puis à la suite ses variables frontières. Pour chaque $SD^{(j)}$, nous construisons la matrice locale par blocs

$$\begin{pmatrix} K_{ii}^{(j)} & K_{ib}^{(j)} \\ K_{bi}^{(j)} & K_{bb}^{(j)} \end{pmatrix} = \begin{pmatrix} L_{ii}^{(j)} & 0 \\ L_{bi}^{(j)} & I \end{pmatrix} \cdot \begin{pmatrix} U_{ii}^{(j)} & U_{ib}^{(j)} \\ 0 & S_{bb}^{(j)} \end{pmatrix} \quad [4]$$

dont nous effectuons la factorisation partielle en utilisant une méthode basée sur la factorisation LU pour expliquer le processus.

Le bloc $K_{ii}^{(j)}$ correspond à des lignes et colonnes complètement sommées qui sont associées aux variables internes. Les blocs $K_{bb}^{(j)}$ correspondent aux variables qui ne peuvent être éliminées à ce stade car elles sont partagées par au moins un autre sous-domaine : les termes qui leur sont associés sont incomplets. Le bloc $S_{bb}^{(j)}$ désigne le complément de Schur local de $SD^{(j)}$ formé à l'issue de cette factorisation partielle. Nous construisons

$$S = \sum_{j=1}^{N_s} S_{bb}^{(j)} = \sum_{j=1}^{N_s} \left(K_{bb}^{(j)} - (L_{bi}^{(j)} \cdot U_{ib}^{(j)}) \right) \quad [5]$$

Nous remarquons immédiatement que

$$S = \sum_{j=1}^{N_s} \left(K_{bb}^{(j)} - K_{bi}^{(j)} \cdot (K_{ii}^{(j)})^{-1} \cdot K_{ib}^{(j)} \right) \quad [6]$$

qui est l'expression du complément de Schur global de la matrice K . De façon analogue, nous obtenons

$$\hat{f}_b = f_b - \sum_{j=1}^{N_s} L_{bi}^{(j)} \cdot (L_{ii}^{(j)})^{-1} \cdot f_i^{(j)} \quad [7]$$

que l'on construit aussi grâce aux factorisations partielles.

À ce stade un problème interface est formé. On construit la matrice S du problème interface [3] et on le résout. Les variables internes u_i de chaque bloc sont ensuite restituées en parallèle. Les factorisations partielles des sous-domaines permettent d'obtenir *in fine* le système interface [3] sans avoir à inverser aucune matrice (Duff *et al.*, 1986).

Dans le travail présenté nous utilisons une méthode frontale pour factoriser partiellement les matrices associées à chaque $SD^{(j)}$ et pour résoudre le problème interface.

Un unique problème interface peut être construit en assemblant, au sens élément fini, tous les compléments de Schur locaux. Malheureusement, le nombre de variables interfaces croît généralement avec le nombre de sous-domaines. Le temps de calcul du problème interface peut devenir important et la place mémoire requise pour le traiter peut dépasser les capacités d'une unique machine. Ce phénomène est aussi observé par Jennifer Scott pour le solveur *MP42* (Scott *et al.*, 1999). Nous pouvons utiliser une approche qui améliore le taux de parallélisme en organisant et structurant différemment les calculs de groupes $\left(K_{bb}^{(j)} - (L_{bi}^{(j)} \cdot U_{ib}^{(j)})\right)$.

En effet, une matrice de complément de Schur locale peut être vue comme la matrice élémentaire d'un « super-élément ». Deux super-éléments partageant des variables peuvent donc être assemblés dans une matrice frontale et partiellement factorisés. Les inconnues frontières sont repérées lors de la phase de partitionnement du graphe G_{elem} des éléments finis. Le nombre de sous-domaines auxquels une inconnue frontière appartient est connu. La mise à jour de cette information lors de l'assemblage de deux super-éléments permet de déterminer les inconnues frontières complètement sommées pour mener à bien la factorisation partielle. Un nouveau complément de Schur, local à ces deux super-éléments, est formé : il peut aussi être vu comme la matrice élémentaire d'un nouveau super-élément. Ce processus d'assemblage/élimination peut s'appliquer à plusieurs niveaux du calcul.

Nous définissons les tâches élémentaires suivantes :

- 1) factorisation partielle d'un sous-domaine initial $SD^{(j)}$;
- 2) assemblage d'au moins deux matrices élémentaires de super-éléments ;
- 3) factorisation partielle d'une matrice issue d'un tel assemblage.

Les deuxième et troisième points ne sont évidemment effectués que si les super-éléments ont des variables communes. Un ordre de traitement efficace des tâches élémentaires doit tenir compte de cette contrainte.

Nous regroupons en une seule tâche les opérations d'assemblage de deux super-éléments et l'élimination des variables interfaces uniquement communes. En effet, réaliser ces deux tâches élémentaires sur un même processeur permet d'éviter des communications.

Notre propos concernant l'équilibrage en volume de calcul, nous ne nous intéressons pas dans cet article à la recherche de la meilleure stratégie de parallélisation. La stratégie de parallélisation que nous considérons sera donc simple. Les tâches seront

organisées en arbre binaire et nous considérons qu'une seule tâche est affectée par processeur.

Définition 1. *Un arbre de calcul $\mathcal{A}^{N_s, y}$ comporte N_s feuilles et y niveaux*

La figure 3 illustre cette approche. Nous associons une tâche à un sommet de l'arbre de calcul. L'étiquette « cond » d'un sommet indique une tâche de factorisation partielle d'un sous-domaine $SD^{(j)}$ et une étiquette « inter » une tâche qui regroupe l'assemblage de deux super-éléments puis la factorisation partielle du système obtenu. Nous notons $q(l, i)$ une estimation du nombre d'opérations associées à une tâche numéro i du niveau l . Soit $L(l)$ l'ensemble des sommets de $\mathcal{A}^{N_s, y}$ au niveau l . Les feuilles dans $L(1)$ correspondent aux factorisations partielles de $SD^{(j)}$.

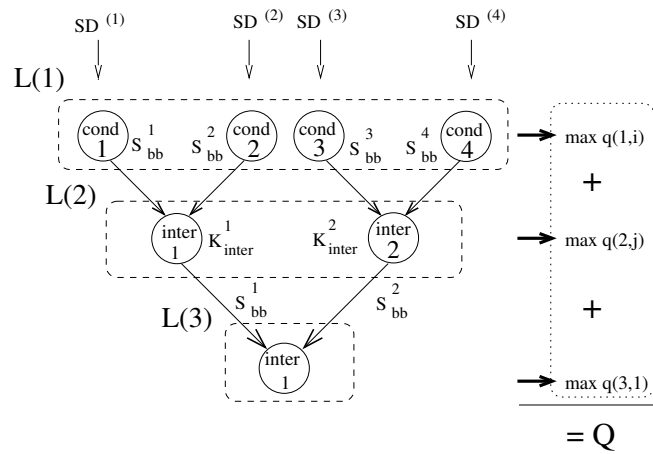


Figure 3. *L'arbre de calcul $\mathcal{A}^{4,3}$ et le principe de l'estimation des calculs*

Pour une tâche associée à un sommet feuille d'un arbre $\mathcal{A}^{N_s, y}$, la donnée d'entrée est le sous-domaine associé à la tâche, et le résultat est le complément de Schur obtenu par sa factorisation partielle.

Pour une tâche associée à un sommet interne d'un arbre $\mathcal{A}^{N_s, y}$, les données d'entrée sont les deux matrices de complément de Schur locaux (*i.e.* les super-éléments) issus des calculs des tâches précédentes. Ces matrices sont d'abord assemblées dans une matrice interface K_{inter} et les variables complètement sommées sont éliminées en utilisant [4] pour obtenir le nouveau complément de Schur S_{bb} .

Sur l'arbre de calcul $\mathcal{A}^{4,3}$ de la figure 3, les sous-domaines $SD^{(3)}$ et $SD^{(4)}$ sont partiellement factorisés par les tâches 3 et 4. Nous obtenons deux matrices de complément de Schur locaux S_{bb}^3 et S_{bb}^4 . Ces matrices élémentaires sont assemblées dans la matrice interface K_{inter}^2 . Les variables frontières associées uniquement aux sous-

domaines $SD^{(3)}$ et $SD^{(4)}$ correspondent à des lignes et colonnes complètement sommées de K_{inter}^2 . Nous obtenons S_{bb}^2 en les éliminant. À la dernière étape, le problème interface du niveau $L(3)$ est obtenu. Ce problème est résolu et les valeurs des variables interfaces associées sont obtenues. Leurs valeurs sont alors transmises au niveau $L(2)$, ce qui permet d'effectuer en parallèle les phases de restitution des autres variables interfaces. Puis les parties de u_b sont transmises au niveau $L(1)$ ce qui permet de déterminer les variables internes u_i de chaque sous-domaine.

Nous utilisons une approche de parallélisation à gros grain. Les deux tâches principales considérées sont la factorisation partielle des sous-domaines initiaux et le traitement des problèmes interfaces (assemblage puis factorisation partielle en multi-niveau). Les temps de communication et de restitution des variables sont considérés négligeables.

Notre objectif est de corriger une partition initiale du graphe G_{elem} associé au maillage par éléments finis de façon à obtenir une meilleure répartition du temps de calcul sur l'ensemble des sous-domaines. Pour atteindre cet objectif nous avons besoin d'une estimation du volume de calcul, c'est-à-dire du nombre d'opérations qu'effectue la méthode frontale sur un sous-domaine $SD^{(j)}$.

Nous proposons l'estimateur

$$\mathcal{E}(V_e^{(j)}, SD^{(j)}) = \alpha_1 \cdot \sum_{k \in assemblage} |F_k| + \alpha_2 \cdot \sum_{k \in elimination} |F_k|^2 \quad [8]$$

où $V_e^{(j)}$ est le vecteur de renumérotation des éléments finis du sous-domaine $SD^{(j)}$ et $|F_k|$ désigne la taille de la matrice frontale au cours des assemblages et éliminations élémentaires consécutifs. Le solveur utilisé assemble les matrices élémentaires des éléments finis et procède à l'élimination des variables complètement sommées à mesure qu'elles sont détectées : la numérotation des éléments finis induit l'évolution de la taille de la matrice frontale. \mathcal{E} correspond au nombre d'opérations d'assemblage et d'élimination. Il a été expérimentalement validé dans des travaux précédents (Denis *et al.*, 2003).

Les coefficients α_1 et α_2 peuvent être interprétés de deux façons. D'un point de vue algorithmique, ils sont homogènes à des nombres d'opérations élémentaires. Un processeur donné effectuera la quantité d'opérations \mathcal{E} en un temps qui dépend de ses caractéristiques. Du point de vue du temps de calcul, ils sont donc homogènes à des secondes par opération. Dans ce cas, $\mathcal{E}(V_e^{(j)}, SD^{(j)})$ peut donner une estimation du temps de calcul pour le sous-domaine $SD^{(j)}$ si l'on est capable d'estimer les coefficients α_1 et α_2 en secondes par opération. À partir d'une série de mesures, nous les avons déterminés par régression. Ces expérimentations ont été menées sur deux clusters de PC différents (Denis, 2003) et validées avec des sous-domaines n'appartenant pas à la base d'exemples. Pour les deux types de machines, nous obtenons moins de 10% de différence entre le temps de calcul estimé pour un sous-domaine $SD^{(j)}$ et le temps de calcul réel mesuré $T_{SD^{(j)}}$. Des effets de seuil semblent apparaître, le modèle ne permet pas de modéliser précisément les effets de cache matériel par exemple.

Toutefois, notre objectif n'était pas de prévoir exactement le temps de calcul mais de vérifier la corrélation entre le modèle et le temps de calcul mesuré. Le point essentiel pour le travail présenté est que le temps de calcul est monotone en fonction de $\mathcal{E}(V_e^{(j)}, SD^{(j)})$ même en fixant $\alpha_1 = \alpha_2 = 1$, c'est ce que nous utiliserons.

À ce stade, nous pouvons estimer les volumes d'opérations $q(1, i)$ des tâches de factorisation partielles du niveau $L(1)$ en utilisant [8], puis utiliser ces estimations pour guider la correction de la partition.

Nous utilisons un second estimateur (Denis *et al.*, 2003) qui compte le nombre d'opérations pour les assemblages de super-éléments et la factorisation partielle d'une matrice K_{inter}^u . L'assemblage de deux compléments de Schur locaux fournit une matrice dense. Pour estimer le nombre d'opérations, il suffit de compter le volume d'opérations d'élimination d'une méthode de type Gauss. Ce second estimateur permet d'évaluer les tâches notées *inter* sur la figure 3 qui correspondent à l'assemblage puis à la factorisation partielle de deux super-éléments (le niveau l est supérieur à 1).

En utilisant ces deux estimateurs, nous évaluons le nombre maximum d'opérations $q(l, i)$ pour chaque niveau $L(i)$ comme illustré dans la figure 3. En procédant ainsi, la somme

$$Q(\mathcal{A}^{N_s, y}, \mathcal{P}) = \sum_{l=1, y \text{ et } i \in L(l)} \max q(l, i) \quad [9]$$

fournit une estimation du volume de calcul requis par la méthode. Dans le cas idéal, toutes les tâches d'un même niveau ont le même nombre d'opérations, Q est alors une borne inférieure. Nous utiliserons [9] pour effectuer des comparaisons avec le temps de calcul global mesuré T_{glob} .

Pour nos expérimentations dans le cas des calculs multiniveaux nous avons considéré des arbres binaires complets (N_s est une puissance de 2) ce qui nous permet d'utiliser les outils de partitionnement multi-niveaux (Karypis *et al.*, 1998). La stratégie de placement des tâches est simple. Dans un premier temps une tâche unique de $L(1)$ est affecté par processeur. Dès qu'une tâche de factorisation est terminée elle envoie le complément au processeur qui n'a pas fini de factoriser partiellement le sous-domaine voisin dans l'arbre. Une fois sa propre factorisation terminée, il assemble le complément de Schur obtenu localement avec le complément de Schur reçu puis effectue la factorisation. Le processus se répète jusqu'à la résolution du problème d'interface qui permet de lancer le processus de restitution des variables locales des sous-domaines.

4. Principe de l'heuristique d'équilibrage

À partir d'une partition initiale \mathcal{P}_{init} , calculée en utilisant l'outil de partitionnement multiniveau (Karypis *et al.*, 1998) nous évaluons le volume d'opérations de chaque sous-domaine en utilisant [8]. Le sous-domaine $SD^{(j)}$ qui possède le plus gros volume estimé d'opérations est sélectionné. Nous notons $SD^{(max)}$ ce sous-domaine et $q^{(max)}$ l'estimation de son volume de calcul. Nous détectons ensuite tous les sous-domaines voisins de $SD^{(max)}$. Nous notons \mathcal{N}_{max} l'ensemble des indices de ces

sous-domaines. Nous réalisons ensuite l'agrégation virtuelle de cet ensemble de sous-domaines que nous notons $SD^{(\mathcal{N}_{max})}$. Pour effectuer l'équilibrage nous procédons au transfert d'éléments finis du sous-domaine $SD^{(max)}$ vers les sous-domaines de $SD^{(\mathcal{N}_{max})}$. Pour cela il faut déterminer le nombre d'éléments finis à transférer et les identifier.

Dans un premier temps, nous déterminons $q_{moy}^{\mathcal{N}_{max}}$ le volume moyen d'opérations des sous-domaines de l'ensemble \mathcal{N}_{max} .

Le volume q_{trans} correspond au volume de calcul à transférer :

$$q_{trans} = \frac{1}{2} \left(q^{(max)} - q_{moy}^{\mathcal{N}_{max}} \right) \quad [10]$$

Il est la moitié de la différence entre le nombre maximum d'opérations estimé et $q_{moy}^{\mathcal{N}_{max}}$ afin d'équilibrer le volume de calcul entre le sous-domaine sélectionné et le sous-domaine virtuel $SD^{(\mathcal{N}_{max})}$. Supposons que \mathcal{N}_{max} ne contienne qu'un seul sous-domaine noté $SD^{(v)}$ d'estimation q^v . Dans ce cas $SD^{(max)}$ possède une unique frontière avec ce voisin. À l'issue du transfert du volume de calculs nous devrions avoir pour $SD^{(max)}$: $q^{(max)} - \frac{1}{2} (q^{(max)} - q^v)$, et pour $SD^{(v)}$: $q^v + \frac{1}{2} (q^{(max)} - q^v)$, c'est-à-dire l'équilibre. La situation est en réalité plus complexe car le transfert s'effectue vers plusieurs sous-domaines. De plus, nous transférons des éléments finis et non des volumes de calculs. En pratique, ce choix pour le calcul de q_{trans} donne de bons résultats.

À ce stade, il faut corrélérer q_{trans} avec le nombre d'éléments finis à transférer m_t . Pour déterminer m_t , nous divisons q_{trans} par le nombre moyen d'opérations par élément fini pour le sous-domaine $SD^{(max)}$. Il s'agit d'une approximation qui ne tient pas compte de la numérotation des éléments finis.

Nous effectuons ensuite le transfert de m_t éléments finis. Nous obtenons une nouvelle partition du domaine à partir de laquelle nous itérons.

L'heuristique se résume ainsi :

- 1) pour chaque sous-domaine $SD^{(j)}$ calculer la numérotation optimisée $V_e^{(j)}$;
- 2) pour chaque sous-domaine $SD^{(j)}$ évaluer $\mathcal{E}(V_e^{(j)}, SD^{(j)})$;
- 3) sélectionner $SD^{(max)}$ ayant l'estimation maximale du nombre d'opérations ;
- 4) déterminer les sous-domaines voisins de $SD^{(max)}$;
- 5) agréger virtuellement ces sous-domaines dans $SD^{(\mathcal{N}_{max})}$;
- 6) calculer le volume moyen d'opérations de ces sous-domaines ;
- 7) calculer le volume d'opérations à transférer ;
- 8) calculer le nombre m_t d'éléments finis à transférer ;
- 9) sélectionner les m_t éléments finis au sein de $SD^{(max)}$;
- 10) transférer ces éléments finis de $SD^{(max)}$ vers les sous-domaines réels.

À chaque itération les éléments finis des sous-domaines obtenus sont renumérotés pour réduire la taille du front.

Les éléments finis à transférer sont choisis au voisinage de la frontière commune de façon à limiter l'accroissement de la taille de l'interface. Les figures 4, 5, 6 illustrent le processus.

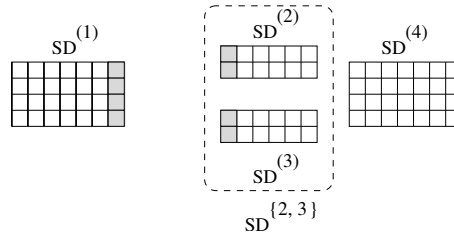


Figure 4. La partition initiale du domaine en 4 sous-domaines

Le domaine de la figure 4 est initialement partitionné en 4 sous-domaines. Nous estimons que $SD^{(1)}$ a le nombre maximum d'opérations. La partie grisée montre les éléments finis proches de la frontière entre $SD^{(1)}$ et le sous-domaine virtuel composé de $SD^{(2)}$ et $SD^{(3)}$ (nous avons $\mathcal{N}_{max} = \{2, 3\}$).

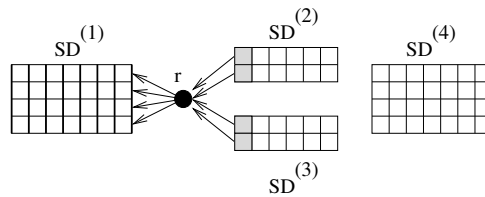


Figure 5. Initialisation de la racine r de la structure en niveau de façon à sélectionner les éléments finis à transférer de $SD^{(1)}$ vers le sous-domaine virtuellement agrégé $SD^{\{2,3\}}$

Nous calculons une structure en niveaux à travers $SD^{(1)}$ à partir des éléments frontières que possède le sous-domaine virtuel $[SD^{(2)}, SD^{(3)}]$ avec $SD^{(1)}$. Nous initialisons le calcul de la structure en niveau avec tous ses éléments finis frontières.

D'un point de vue pratique, nous appliquons un algorithme de type *BFS* (Cormen *et al.*, 1992) sur le graphe des éléments finis du sous-domaine $SD^{(1)}$ en initialisant sa file avec les sommets des éléments finis frontières, ce qui définit le niveau 0 de la structure. Nous obtenons un arbre couvrant où le niveau l contient les éléments finis qui sont à une distance de l arêtes du niveau 0. Cette façon de procéder peut s'interpréter comme l'utilisation d'un sommet virtuel r qui connecte $SD^{(1)}$ et le sous-domaine virtuel comme le montre la figure 5 pour laquelle $m_t=4$. Nous transférons ensuite

les éléments finis sélectionnés vers les sous-domaines voisins comme le montre la figure 6.

En effet, le processus de marquage de l'algorithme basé sur le *BFS* est initialisé avec les éléments finis de $SD^{(\mathcal{N}_{max})}$ qui sont à la frontière avec $SD^{(max)}$. Ces éléments finis appartiennent à un unique sous-domaine compte tenu de la décomposition du graphe des éléments finis G_{elem} . Chaque élément fini marqué au sein de $SD^{(max)}$ possède un ancêtre unique car un sommet ne peut être marqué qu'une seule fois durant le processus (Cormen *et al.*, 1992). L'affectation d'un élément fini marqué de $SD^{(max)}$ vers un unique sous-domaine de l'ensemble \mathcal{N}_{max} est donc immédiate.

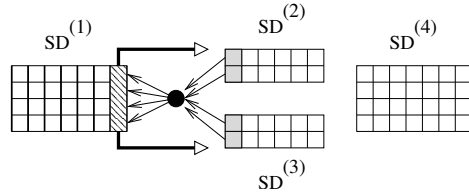


Figure 6. Transfert des m_t éléments finis du sous-domaine de $SD^{(1)}$ vers $SD^{(2)}$ et $SD^{(3)}$

Cette stratégie de sélection permet de limiter l'accroissement de la frontière qu'inclut la migration d'éléments finis.

Le paramètre d'entrée noté *Limite_glob* fixe le nombre d'itérations de l'algorithme d'équilibrage. Nous relevons la partition qui fournit le meilleur équilibrage en volume de calcul estimé ainsi que les vecteurs de renumérotation $V_e^{(j)}$ calculés.

5. Protocole expérimental et critères

Les expérimentations ont été menées sur un cluster de PC composé de 10 bi-processeurs Athlon 2200+, 1 Go, sous LINUX Red Hat 7.1, avec un réseau 1 Gbit/s Ethernet cuivre. Nous avons utilisé le solveur à front multiple de *SIC* et l'outil de partitionnement multiniveaux de (Karypis *et al.*, 1998) pour construire la partition initiale du domaine. Les données utilisées pour les expérimentations sont issues du projet *PARASOL* (PARASOL, 2003) et de la collection de maillages (Rassineux, 2004). Le Tableau 1 présente les caractéristiques de notre jeu test. La colonne *nombre de degrés de liberté* donne le nombre de lignes de la matrice assemblée. Les fichiers de données du projet *PARASOL* sont au format *RSE* (*Real Symmetric Elemental*) ce qui nous a obligé de reconstruire un graphe G_{elem} correspondant au système. Les données des systèmes élémentaires ne permettent pas de construire ce graphe de manière unique. Dans cette reconstruction nous avons pris un certain nombre d'hypothèses. Notamment nous avons considéré un nœud par degré de liberté.

La méthode de partitionnement de graphe multiniveaux est appliquée sur le graphe G_{elem} . Les N_s sous-domaines $SD^{(j)}$ de cette partition initiale \mathcal{P}_{init} sont renuméro-

Label	nombre d'éléments finis nelt	nombre de nœuds nnt	nombre de degrés de liberté ndlt	source
C1	42 689	11 569	34 707	(Rassineux, 2004)
FU	36 570	8 374	25 122	(Rassineux, 2004)
M4	55 498	27 824	166 824	(Rassineux, 2004)
SU	18 171	4 839	14 517	(Rassineux, 2004)
M_T1	5 328	97 578	97 578	(PARASOL, 2003)
SHIP_003	45 464	121 728	121 728	(PARASOL, 2003)
SHIPSEC8	35 280	114 919	114 919	(PARASOL, 2003)
THREAD	2 176	29 736	29 736	(PARASOL, 2003)

Tableau 1. Les données et maillages du jeu test

tés en utilisant une méthode adaptée au solveur utilisé (Boufflet *et al.*, 2001) : nous obtenons les N_s ordres d'assemblage $V_e^{(j)}$. Nous utilisons la même méthode pour renuméroter les sous-domaines dans l'heuristique. Nous évaluons $Q(\mathcal{A}^{N_s, y}, \mathcal{P}_{init})$ [9]. Puis nous effectuons la résolution parallèle, et mesurons le temps de résolution global de la méthode T_{glob}^{init} pour \mathcal{P}_{init} et les temps de condensation $T_{SD^{(j)}}$ de chaque $SD^{(j)}$.

Nous appliquons ensuite l'heuristique d'équilibrage. La meilleure solution rencontrée fournit la partition corrigée en volume de calcul \mathcal{P}_c et les N_s vecteurs $V_e^{(j)}$. Nous relevons T_{equi} le temps d'exécution de l'heuristique d'équilibrage. Nous évaluons $Q(\mathcal{A}^{N_s, y}, \mathcal{P}_c)$. Nous effectuons la résolution parallèle, et nous relevons T_{glob}^c le temps de résolution global de la méthode pour \mathcal{P}_c et les temps de factorisation $T_{SD^{(j)}}$ de chaque $SD^{(j)}$.

Le gain $g_{T_{glob}}$ en % permet de mesurer l'intérêt de l'approche

$$g_{T_{glob}} = \frac{T_{glob}^{init} - T_{glob}^c}{T_{glob}^{init}} \quad [11]$$

mais il ne tient pas compte du temps d'exécution de l'heuristique.

Le gain g_{eff} en % le prend en compte, il met en évidence le bénéfice pratique que l'on peut escompter

$$g_{eff} = \frac{T_{glob}^{init} - T_{glob}^c - T_{equi}}{T_{glob}^{init}} \quad [12]$$

Le facteur d'équilibrage $\Delta^{mesu}(\mathcal{P})$ d'une partition \mathcal{P} mesure l'équilibre en temps de calcul des tâches de factorisation

$$\Delta^{mesu}(\mathcal{P}) = \frac{1}{N_s} \frac{\sum_{j \in 1}^{N_s} T_{SD(j)}}{\max_j T_{SD(j)}} \quad [13]$$

Le facteur $\Delta^{mesu}(\mathcal{P})$ est borné supérieurement par 1. Nous l'exprimerons donc en % dans les tableaux. Plus il est proche de 1, plus les tâches de factorisation de la partition \mathcal{P} sont équilibrées en volume de calcul.

Le facteur d'équilibrage Δ_{moy}^{mesu} est une moyenne des facteurs d'équilibrage Δ^{mesu} sur une série d'expérimentations.

Trois arbres de tâches sont utilisés dans les expérimentations. Ils sont répertoriés dans le tableau 2. Nous rappelons que nous affectons un seul sous-domaine par pro-

arbre	nombre de sous-domaines N_s	nombre de niveaux y	label
$\mathcal{A}^{2,2}$	2	2	A
$\mathcal{A}^{4,3}$	4	3	B
$\mathcal{A}^{8,4}$	8	4	C

Tableau 2. Descriptif des arbres de calculs

cesseur. Nous nous sommes limités à un nombre de sous-domaines égal à 8 sur une machine à 10 processeurs.

Le code de l'heuristique est séquentiel et écrit en langage C. Il constitue un module indépendant du solveur. Le coût algorithmique le plus important est dû aux renumérotations de tous les sous-domaines qui s'effectuent entre chaque itération de la boucle d'équilibrage.

6. Résultats expérimentaux

Deux séries d'expérimentations sont présentées. L'objectif de la première est de mettre en évidence les gains obtenus sur le temps T_{glob} en corrigeant en volume de calculs une décomposition de domaine initiale. Le temps T_{equi} d'exécution de l'heuristique n'est pas considéré et $Limite_{glob} = 100$. La deuxième série vise à mettre en évidence le gain pratique que l'on peut escompter sur T_{glob} en prenant en compte le temps T_{equi} et en effectuant peu d'itérations. Dans cette série nous avons fixé $Limite_{glob} = 10$.

6.1. Validation expérimentale

Les résultats obtenus sur les maillages (Rassineux, 2004) sont exposés dans le tableau 3 et ceux obtenus sur les exemples (PARASOL, 2003) sont présentés dans le tableau 4. Pour chaque expérimentation, nous reportons sur les cinq premières colonnes : le label de l'exemple (tableau 1), le type d'arbre de calcul (tableau 2), le type de partition (\mathcal{P}_{init} pour une partition issue de METIS et \mathcal{P}_c pour cette partition corrigée), l'estimateur du volume de calcul $Q(\mathcal{A}^{N_s, y}, \mathcal{P})$ [9], et le temps global T_{glob} de résolution en secondes. La sixième colonne donne le ratio T_{glob}/Q . Si l'estimation $Q(\mathcal{A}^{N_s, y}, \mathcal{P})$ était parfaite et si le temps T_{glob} correspondait exactement à ce volume de calculs estimé nous aurions une constante. En pratique ce n'est pas le cas, mais cela donne une indication de la qualité de l'estimation du volume de calculs global en tenant compte des aléas de l'architecture. La septième colonne contient le gain $g_{T_{glob}}$ obtenu sur T_{glob} en comparant les résultats obtenus avec la partition initiale \mathcal{P}_{init} et avec la partition corrigée \mathcal{P}_c . La huitième colonne contient le facteur d'équilibrage Δ_{mesu} .

Sur l'ensemble des expérimentations nous pouvons constater que les valeurs de Q et de T_{glob} sont moins élevées pour les partitions corrigées \mathcal{P}_c . Prenons par exemple le maillage $M4$ avec l'arbre de calcul A et une partition initiale. La valeur de Q est de $1,44.10^{11}$ et le temps de résolution global est $T_{glob} = 1220s$. Avec la partition corrigée nous obtenons $Q = 3,91.10^{10}$ et $T_{glob} = 395.3s$.

Nous présentons sous forme graphique (Fig. 7 à 10) l'évolution des mesures et des critères pour le maillage $C1$ et la donnée SHIPSEC8.

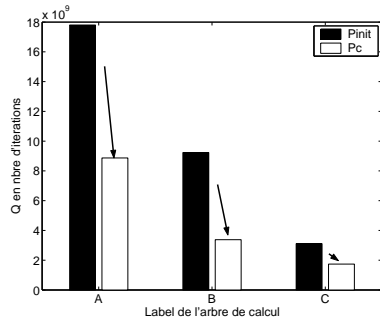


Figure 7. Evolution de $Q(\mathcal{A}^{N_s, y}, \mathcal{P})$ pour les partitions \mathcal{P}_{init} et \mathcal{P}_c , maillage C1, $Limite_{glob} = 100$, pour les trois types d'arbre

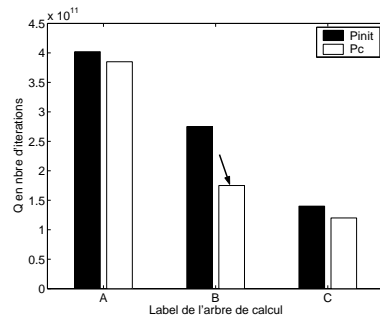


Figure 8. Evolution de $Q(\mathcal{A}^{N_s, y}, \mathcal{P})$ pour les partitions \mathcal{P}_{init} et \mathcal{P}_c , donnée SHIPSEC8, $Limite_{glob} = 100$, pour les trois types d'arbre

Les figures 7 et 8 présentent l'estimateur en volume de calcul $Q(\mathcal{A}^{N_s, y}, \mathcal{P})$. Les différences entre les hauteurs des barres noires et blanches indiquent les espoirs de gains pour chaque type d'arbre de calcul (tableau 2).

Label	Arbre	Partition	Q	T_{glob} (s)	$\frac{T_{glob}}{Q}$ ($\times 10^{-9}$)	$g_{T_{glob}}$ (%)	Δ^{mesu} (%)
C1	A	\mathcal{P}_{init}	$1,78.10^{10}$	129,9	7,30		80
	A	\mathcal{P}_c	$8,87.10^9$	84,2	9,49	35,2	88
	B	\mathcal{P}_{init}	$9,23.10^9$	99,4	10,7		54
	B	\mathcal{P}_c	$3,38.10^9$	30,1	8,91	69,7	90
	C	\mathcal{P}_{init}	$3,11.10^9$	24,7	7,94		54
	C	\mathcal{P}_c	$1,74.10^9$	15,4	8,85	37,7	90
FU	A	\mathcal{P}_{init}	$6,71.10^9$	58	8,64		84
	A	\mathcal{P}_c	$4,56.10^9$	42	9,21	27,6	97
	B	\mathcal{P}_{init}	$5,61.10^9$	43,8	7,72		71
	B	\mathcal{P}_c	$3,25.10^9$	30,1	9,26	31,3	89
	C	\mathcal{P}_{init}	$3,54.10^9$	22,4	6,33		69
	C	\mathcal{P}_c	$2,15.10^9$	19	8,84	15,2	79
SU	A	\mathcal{P}_{init}	$2,67.10^9$	21,1	7,90		88
	A	\mathcal{P}_c	$1,19.10^9$	9,9	8,32	53,1	73
	B	\mathcal{P}_{init}	$1,20.10^9$	9,9	8,25		50
	B	\mathcal{P}_c	$9,28.10^8$	4,1	4,42	58,6	80
	C	\mathcal{P}_{init}	$7,75.10^8$	6,7	8,65		44
	C	\mathcal{P}_c	$4,05.10^8$	3,8	9,38	43,8	73
M4	A	\mathcal{P}_{init}	$1,44.10^{11}$	1220	8,47		71
	A	\mathcal{P}_c	$3,91.10^{10}$	395,3	10,1	67,6	99
	B	\mathcal{P}_{init}	$1,04.10^{11}$	891,6	8,57		65
	B	\mathcal{P}_c	$4,81.10^{10}$	406,6	8,45	54,4	67
	C	\mathcal{P}_{init}	$6,13.10^{10}$	448,3	7,31		59
	C	\mathcal{P}_c	$2,26.10^{10}$	204,5	9,05	54,4	93

Tableau 3. Résultats de la première série d'expérimentations, obtenus sur les maillages (Rassineux, 2004) Les lignes \mathcal{P}_c montrent le résultat après correction, à comparer aux lignes \mathcal{P}_{init} de la partition initiale. Plus le % de la colonne $g_{T_{glob}}$ est élevé meilleur est le gain. La colonne Q correspond à l'estimation du volume de calcul $Q(\mathcal{A}^{N_s, y}, \mathcal{P})$. Le ratio T_{glob}/Q est une indication de la qualité de l'estimation par rapport au temps de calcul réel. Δ^{mesu} donne la valeur du facteur d'équilibrage

Les figures 9 et 10 montrent respectivement l'évolution du temps de résolution global T_{glob} pour le maillage C1 et pour la donnée SHIPSEC8. Il s'agit des gains effectifs que nous avons mesurés sur T_{glob} .

Nous constatons que les graphiques pris deux à deux ont des tendances similaires. Cette corrélation est confirmée dans l'ensemble de nos expérimentations.

Nous obtenons des gains qui peuvent atteindre 70%. Pour une décomposition à la fois équilibrée en volume de données et en volume de calculs (THREAD, tableau 4), le gain est nul comme attendu. Toutes les autres expérimentations permettent d'ob-

Label	Arbre	Partition	Q	T_{glob} (s)	$\frac{T_{glob}}{Q}$ ($\times 10^{-9}$)	$g_{T_{glob}}$ (%)	Δ^{mesu} (%)
M_T1	A	\mathcal{P}_{init}	$1,17.10^{11}$	94,4	0,81		96
	A	\mathcal{P}_c	$1,00.10^{11}$	89,6	0,90	5,1	98
	B	\mathcal{P}_{init}	$1,09.10^{11}$	66,7	0,61		71
	B	\mathcal{P}_c	$8,05.10^{10}$	59,1	0,73	11,4	80
	C	\mathcal{P}_{init}	$4,26.10^{10}$	43,0	1,01		54
	C	\mathcal{P}_c	$3,15.10^{10}$	36,9	1,17	14,2	72
SHIP_003	A	\mathcal{P}_{init}	$9,26.10^{10}$	73,8	0,80		82
	A	\mathcal{P}_c	$8,20.10^{10}$	53,6	0,65	27,4	97
	B	\mathcal{P}_{init}	$4,98.10^{10}$	34,9	0,7		89
	B	\mathcal{P}_c	$4,98.10^{10}$	34,9	0,7	0	89
	C	\mathcal{P}_{init}	-	-	-		-
	C	\mathcal{P}_c	-	-	-		-
SHIPSEC8	A	\mathcal{P}_{init}	$4,02.10^{11}$	266,6	0,66		81
	A	\mathcal{P}_c	$3,85.10^{11}$	239,1	0,62	10,3	89
	B	\mathcal{P}_{init}	$2,75.10^{11}$	160,5	0,58		71
	B	\mathcal{P}_c	$1,75.10^{11}$	112,7	0,64	29,8	89
	C	\mathcal{P}_{init}	$1,40.10^{11}$	108,3	0,77		71
	C	\mathcal{P}_c	$1,20.10^{11}$	101,1	0,84	6,6	79
THREAD	A	\mathcal{P}_{init}	$9,26.10^{10}$	17,8	0,19		99
	A	\mathcal{P}_c	$9,26.10^{10}$	17,8	0,19	0	99
	B	\mathcal{P}_{init}	$8,17.10^9$	15,3	1,87		89
	B	\mathcal{P}_c	$8,17.10^9$	15,3	1,87	0	89
	C	\mathcal{P}_{init}	$7,51.10^9$	17,5	2,33		65
	C	\mathcal{P}_c	$7,51.10^9$	17,5	2,33	0	65

Tableau 4. Résultats de la première série d'expérimentations, obtenus sur les données (PARASOL, 2003). Les lignes \mathcal{P}_c montrent le résultat après correction, à comparer aux lignes \mathcal{P}_{init} de la partition initiale. Plus le % de la colonne $g_{T_{glob}}$ est élevé, meilleur est le gain. La colonne Q correspond à l'estimation du volume de calcul $Q(\mathcal{A}^{N_s,y}, \mathcal{P})$. Le ratio T_{glob}/Q est une indication de la qualité de l'estimation par rapport au temps de calcul réel. Δ^{mesu} donne la valeur du facteur d'équilibrage

tenir un gain sur au moins une des décompositions initiales, ce qui confirme qu'une partition équilibrée en volume de données ne l'est pas forcément en volume de calculs.

Les figures 11 et 12 reportent les facteurs d'équilibrage Δ^{mesu} obtenus pour le maillage $C1$ et la donnée $SHIPSEC8$. Nous constatons sur notre jeu test que des partitions équilibrées en volume de calcul conduisent à des facteurs d'équilibrage mesurés nettement améliorés. Ce qui semble justifier notre modélisation du solveur.

Le tableau 5 répertorie le facteur d'équilibrage moyen en fonction de l'arbre de calcul. Nous observons que le facteur d'équilibrage moyen Δ_{moy}^{mesu} des partitions \mathcal{P}_{init} diminue lorsque le nombre de sous-domaines augmente aussi bien pour \mathcal{P}_{init} que

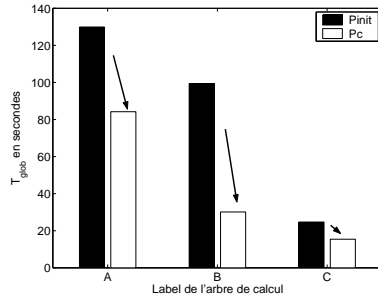


Figure 9. Evolution de T_{glob} pour les partitions \mathcal{P}_{init} et \mathcal{P}_c , maillage C1, $Limite_{glob} = 100$, pour les trois types d'arbre

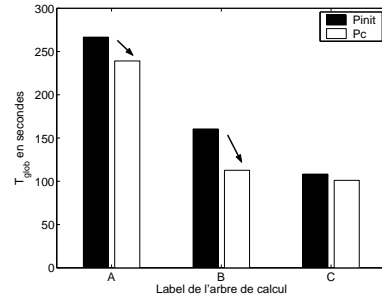


Figure 10. Evolution de T_{glob} pour les partitions \mathcal{P}_{init} et \mathcal{P}_c , donnée SHIPSEC8, $Limite_{glob} = 100$, pour les trois types d'arbre

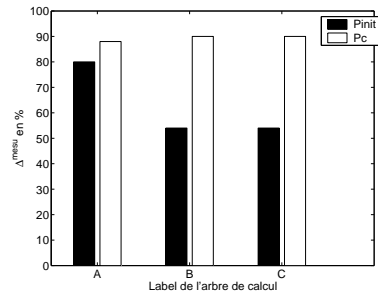


Figure 11. Evolution de Δ^{mesu} pour les partitions \mathcal{P}_{init} et \mathcal{P}_c , maillage C1, $Limite_{glob} = 100$, pour les trois types d'arbre

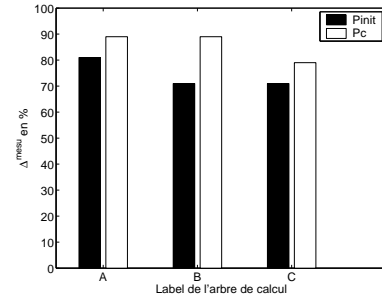


Figure 12. Evolution de Δ^{mesu} pour les partitions \mathcal{P}_{init} et \mathcal{P}_c , donnée SHIPSEC8, $Limite_{glob} = 100$, pour les trois types d'arbre

pour \mathcal{P}_c . La cinquième colonne du tableau 5 donne le pourcentage moyen d'amélioration de l'équilibrage. Elle montre que l'heuristique améliore proportionnellement mieux l'équilibrage quand le nombre de sous-domaines augmente.

La figure 13 montre l'évolution du facteur d'équilibrage Δ_{moy}^{mesu} (tableau 5) en fonction de $\text{Log}(N_s)$ pour les partitions équilibrées en volume de données et en volume de calculs. Dans la figure 2, nous présentons les tendances pour le calcul avec un seul problème interface. Les points de la figure 13 concernent le calcul en multi-niveau du problème interface. Les courbes pointillées montrant la tendance de l'estimation des facteurs d'équilibrages pour des partitions équilibrées en volume de données et équilibrées en volume de calculs sont superposées aux résultats. Nous observons que les comportements sont corrélés comme attendu. En moyenne, le facteur d'équilibrage évolue comme annoncé dans les motivations.

arbre de calcul	N_s : nombre de sous- domaines	partition	Δ_{moy}^{mesu} en %	amélioration moyenne en %
A	2	\mathcal{P}_{init}	85,2	
A	2	\mathcal{P}_c	92,4	8,4
B	4	\mathcal{P}_{init}	70,0	
B	4	\mathcal{P}_c	84,2	20,3
C	8	\mathcal{P}_{init}	59,4	
C	8	\mathcal{P}_c	71,6	20,5

Tableau 5. Valeur moyenne Δ_{moy}^{mesu} du facteur d'équilibrage pour les maillages et les données du tableau 1

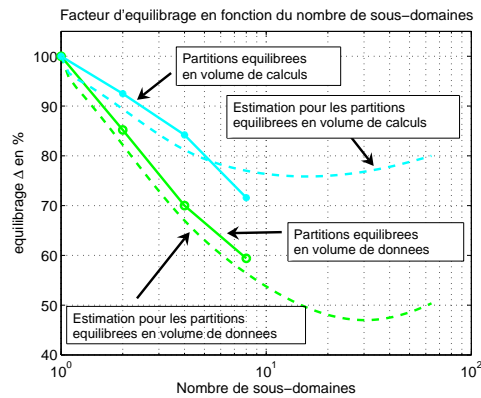


Figure 13. Facteur d'équilibrage Δ_{moy}^{mesu} en fonction du nombre de sous-domaines pour les valeurs du tableau 5

Les estimateurs permettent d'évaluer des volumes de calculs. Néanmoins, ils ne mesurent pas exactement le volume réel et le calcul effectif est soumis à des fluctuations du comportement de l'architecture matérielle que nous ne pouvons maîtriser. Les éléments finis sont transférés en se basant sur ces estimations, ce qui peut engendrer des comportements moins réguliers pour un maillage particulier. C'est ce que nous pouvons constater en examinant les colonnes T_{glob}/Q des tableaux 3 et 4. Pour le premier la valeur moyenne du ratio s'établit à 8,42 et l'écart type est de 1,95. Pour le second nous ne disposons pas du maillage par éléments finis pour mener à bien nos expérimentations. Un graphe G_{elem} a été reconstruit à partir du format RSE en petites matrices élémentaires. Nous avons supposé un degré de liberté par nœud ce qui n'est pas forcément le cas. Les résultats obtenus sont de moins bonne qualité que les résultats sur les maillages (Rassineux, 2004).

Les valeurs des facteurs d'équilibrage mesurés Δ^{mesu} reportés dans les tableaux 3 et 4 sont, soit conservées, soit améliorées : équilibrer en volume de calculs ne peut que faire diminuer le temps de calcul global et les gains les plus intéressants sont obtenus pour de faibles nombres de processeurs.

D'autres expérimentations ont été menées sur un cluster de Pentium III à 333Mhz, 384 Mo de mémoire centrale, reliés par un réseau ethernet 100 Mb (Denis, 2003). Elles confirment les résultats obtenus.

6.2. Expérimentations en vue d'un usage pratique de l'approche

La méthode de renumérotation utilisée dans notre version actuelle de l'algorithme d'équilibrage est trop longue pour les données du projet *Parasol*. En effet, dans le format *RSE*, nous ne disposons pas du maillage par éléments finis explicite mais des matrices élémentaires. Nous reconstruisons un graphe G_{elem} en supposant qu'un nœud correspond à un degré de liberté (*ddl*). Nous considérons qu'un élément fini est connecté à un autre s'ils possèdent un *ddl* commun. Le degré moyen d'un sommet de ce graphe est donc très important. Ce qui induit que le coût de la renumérotation effectuée pour tous les sous-domaines à chaque itération est conséquent. La deuxième série d'expérimentations a donc été effectuée sur les maillages explicites (Rassineux, 2004). Les résultats obtenus sont présentés dans le tableau 6.

Pour chaque expérimentation, nous reportons sur les cinq premières colonnes : le label de l'exemple (tableau 1), le type d'arbre de calcul (tableau 2), le type de partition (initiale ou corrigée), le temps de calcul T_{equi} pour obtenir la partition corrigée, et le temps de résolution global T_{glob} en secondes. La sixième colonne contient le gain g_{eff} réellement obtenu sur T_{glob} en comparant les résultats obtenus entre \mathcal{P}_{init} et \mathcal{P}_c en tenant compte du temps de calcul de l'heuristique. La septième colonne contient le facteur d'équilibrage Δ^{mesu} .

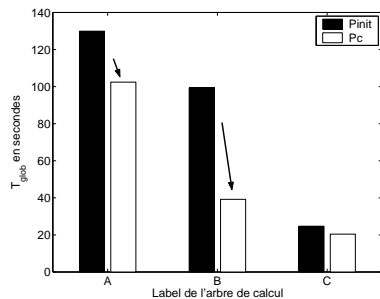


Figure 14. Evolution de T_{glob} pour les partitions \mathcal{P}_{init} et \mathcal{P}_c , maillage C1, $Limite_{glob} = 10$, pour les trois types d'arbre

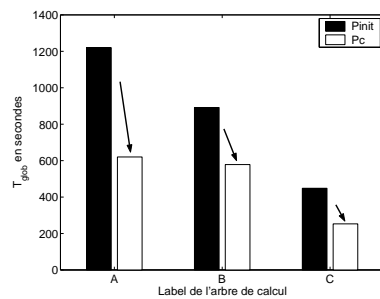


Figure 15. Evolution de T_{glob} pour les partitions \mathcal{P}_{init} et \mathcal{P}_c , maillage M4, $Limite_{glob} = 10$, pour les trois types d'arbre

Label	Arbre $\mathcal{A}^{N_s, y}$	Partition	$T_{equi}(s)$	$T_{glob}(s)$	$g_{eff}(\%)$	$\Delta^{mesu}(\%)$
C1	A	\mathcal{P}_{init}	0	129,9		80
	A	\mathcal{P}_c	4,8	102,4	17,5	88
	B	\mathcal{P}_{init}	0	99,4		54
	B	\mathcal{P}_c	12,7	39,2	48,2	80
	C	\mathcal{P}_{init}	0	24,7		54
	C	\mathcal{P}_c	34,4	20,4	-121,8	69
FU	A	\mathcal{P}_{init}	0	58		84
	A	\mathcal{P}_c	3,6	43,5	18,8	90
	B	\mathcal{P}_{init}	0	43,8		71
	B	\mathcal{P}_c	6,6	30,5	12,3	92
	C	\mathcal{P}_{init}	0	22,4		69
	C	\mathcal{P}_c	19,5	16,5	-60,7	90
SU	A	\mathcal{P}_{init}	0	21,1		84
	A	\mathcal{P}_c	1,7	9,8	45,5	73
	B	\mathcal{P}_{init}	0	9,9		50
	B	\mathcal{P}_c	3,5	7,8	-14,1	79
	C	\mathcal{P}_{init}	0	6,7		44
	C	\mathcal{P}_c	9,79	5,7	-131,1	48
M4	A	\mathcal{P}_{init}	0	1220		71
	A	\mathcal{P}_c	3,4	620	48,9	66
	B	\mathcal{P}_{init}	0	891,6		65
	B	\mathcal{P}_c	6,7	578,4	34,6	62
	C	\mathcal{P}_{init}	0	448,3		59
	C	\mathcal{P}_c	12,7	252,9	40,7	78

Tableau 6. Résultats de la deuxième série d'expérimentations, obtenus sur les maillages (Rassineux, 2004). Les lignes \mathcal{P}_c montrent le résultat après correction, à comparer aux lignes \mathcal{P}_{init} de la partition initiale. Plus le % de la colonne $g_{eff}(\%)$ est élevé meilleur est le gain effectif qui tient compte du temps de calcul de l'heuristique. La colonne Q correspond à l'estimation du volume de calcul $Q(\mathcal{A}^{N_s, y}, \mathcal{P})$. Δ^{mesu} donne la valeur du facteur d'équilibrage

Les figures 14 et 15 montrent l'évolution du temps de résolution global T_{glob} en fonction du type d'arbre de calcul pour les maillages $C1$ et $M4$. Prenons par exemple le maillage $M4$ décomposé en huit sous-domaines. Le temps de résolution avant équilibrage est de 448.3 s. La méthode d'équilibrage fournit en 12.7 s une partition \mathcal{P}_c dont le temps de résolution après équilibrage est de 252.9 s : on obtient un gain effectif de 40.7 %.

Pour certains essais le temps d'exécution de l'heuristique est plus important que la diminution du temps de résolution global. Le gain obtenu n'est pas, dans ce cas,

suffisant pour compenser le temps d'exécution de la méthode d'équilibrage. La valeur de g_{eff} ne représente alors pas un gain mais une perte, correspondant à des valeurs négatives dans le tableau 6. Par exemple, pour le maillage SU décomposé en huit sous-domaines, le temps de résolution avant équilibrage est de 6.7 s. La méthode d'équilibrage fournit en 9.79s une partition \mathcal{P}_c . Le temps de la résolution après équilibrage est de 5.7 s.

Le tableau 6 montre que le gain effectif g_{eff} est négatif pour les tests avec des tailles réduites de sous-domaines. L'usage de l'heuristique est intéressant quand la décomposition du graphe d'éléments finis G_{elem} fournit des sous-domaines de taille suffisamment importante.

7. Conclusion

Nous avons proposé une heuristique qui corrige une décomposition de domaine initialement équilibrée en volume de données de façon à équilibrer les volumes estimés d'opérations des sous-domaines. Pour cela nous avons modélisé le comportement algorithmique du solveur parallèle à front multiple. L'heuristique est dépendante du solveur utilisé, mais nous pensons que la démarche est applicable à d'autres solveurs directs basés sur la méthode du complément de *Schur* et la décomposition de domaine. Le transfert d'éléments finis peut faire augmenter considérablement le nombre de nœuds interfaces et, en conséquence, le temps de calcul du problème interface. La primitive de transfert proposée permet de limiter ce problème. Avec cette approche de parallélisation à gros grain, les résultats préliminaires obtenus améliorent le temps de calcul global. Pour utiliser pratiquement la version actuelle de l'heuristique, il est souhaitable de disposer du maillage par éléments finis explicite. Il faut que les sous-domaines possèdent un volume de calcul suffisamment important tout en restant économiques à renuméroter pour que l'équilibrage soit rentable. Deux perspectives d'amélioration de la méthode d'équilibrage permettraient de diminuer son temps d'exécution. Une première possibilité d'amélioration concerne la numérotation des sous-domaines qui est modifiée à chaque itération. On observe que les numéros varient uniquement près de l'interface. En numérotant en dernier les strates d'éléments finis près de la frontière d'un sous-domaine nous espérons pouvoir mettre au point une technique de numérotation qui modifie uniquement l'ordre des dernières couches. La seconde possibilité concerne la parallélisation de l'heuristique. À partir d'une affectation aux processeurs de la décomposition initiale en sous-domaines nous pouvons paralléliser les évaluations des volumes d'opérations et les renumérotations des sous-domaines.

8. Bibliographie

Amestoy P.R., Duff I.S., L'Excellent J-Y., Koster J., « A fully asynchronous multifrontal solver using distributed dynamic scheduling », *SIAM J. Matrix Anal. Appl.*, vol. 23, 2001, p. 15-41.

- Boufflet J.P., Breitkopf P., Denis C., Rassineux A., Vayssade M., « Optimal Element Numbering Schemes for Direct Solution of Mechanical Problems using Domain Decomposition Method », *4th ECCOMAS Solid Mechanics Conference*, Espagne, 2000.
- Boufflet J.P., Breitkopf P., Denis C., Rassineux A., Vayssade M., « Renumérotation des Eléments Finis par Sous-domaines pour un Solveur Parallèle Multifrontal », *5ème Colloque National en Calcul des Structures*, vol. 2, Giens France, 2001, p. 699-706.
- Breitkopf P., Escaig Y., « Object oriented approach and distributed finite element simulations », *Revue Européenne des Eléments Finis*, vol. 7, n° 5, 1998, p. 609–626.
- Cormen T., Leiserson C., Rivest R., *Introduction to algorithms*, The MIT Press, 1992.
- Denis C., Equilibrage en volume de calcul de la méthode parallèle à fronts multiples : application au logiciel SIC, Thèse de doctorat, Université de Technologie de Compiègne, 2003.
- Denis C., Boufflet J.P., Breitkopf P., Rassineux A., Vayssade M., « Equilibrage en Volume de Calcul pour un Solveur Parallèle Multi-niveau », *Actes du 6ème Colloque National en Calcul des Structures*, Giens France, 2003, p. 349-356.
- Duff I., Erisman A., Reid J., *Direct Methods for Sparse Matrices*, Monographs on Numerical Analysis, Clarendon Press - Oxford, 1986.
- Duff I., Scott J.A., MA42 – A New Frontal Code for Solving Sparse Unsymmetric Systems, Technical Report n° RAL 93-064, 1993, Chilton Oxon England.
- Escaig Y., Décomposition de Domaines Multiniveaux et Traitements Distribués pour la Résolution de Problèmes de Grande Taille, Thèse de doctorat, Université de Technologie de Compiègne, 1992.
- Escaig Y., Vayssade M., Touzot G., « Une méthode de décomposition de domaines multifrontale multiniveaux » *Revue Européenne des Eléments Finis*, vol. 3, n° 3, 1994, p. 311-337.
- Escaig Y., Touzot G., « Application des méthodes de décomposition de domaines au calcul parallèle », *Revue Européenne des Eléments Finis*, vol. 6, n°5, 1997, p. 589-609.
- Hendrickson B., Leland R., The *Chaco* user's guide, Sandia National Laboratories, Technical Report n° SAND93–2339, 1995, Albuquerque.
- Hendrickson B., « Graph Partitioning and Parallel Solvers : Has the Emperor No Clothes ? », *Actes de Irregular'98, Lecture Notes in Computer Science*, Springer, vol. 1457, 1998, p. 218-225.
- Hendrickson B., « Load balancing fictions, falsehoods and fallacies », *Applied Mathematical Modelling*, vol. 25, 2000, p. 99-108.
- Irons B.M., « A Frontal Solution Program for Finite Element Analysis », *Int. J. for Num. Meth. in Eng.*, vol. 2, 1970, p. 5-32.
- Karypis G., Kumar V., METIS : A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Technical Report, 1998, University of Minnesota, Department of Computer Science.
- Lassignardie J.F., Application du calcul distribué en mécanique des solides et des structures, Thèse de doctorat, INSA de Rouen, 2001.

- Negre S., Optimisation de la méthode multifrontale en vue de sa parallélisation, Thèse de doctorat, Université de Technologie de Compiègne, 1997.
- PARASOL, URL du projet : <http://www.parallab.uib.no/parasol>, 2003.
- Pellegrini F., SCOTCH 3.1 User's guide, Rapport Technique, LaBRI URA CNRS 1304, 1997, Université Bordeaux I, France.
- Preis R., The PARTY Graph Partitioning - Library User manual Version 1.99, Technical Report, 1998, Universität Paderborn, Germany.
- Rassineux A., URL : <http://www.utc.fr/~rassineu>, 2004.
- Scott J., The Design of a Parallel Frontal Solver, Technical Report RAL-TR99-075, 1999, Rutherford Appleton Laboratory, UK.
- Walshaw C., The *jostle* user manual : Version 2.2, Technical Report, 2000, School of Computing & Mathematical Sciences, University of Greenwich, London, UK.