Implantation dans CESAR-LCPC d'un schéma multifrontal pour la résolution de systèmes linéaires de grande taille

Stéphane RIGOBERT
Laboratoire Central des Ponts et Chaussées

RÉSUMÉ

Un schéma multifrontal pour la résolution d'un système linéaire d'équations sous forme matricielle est présenté. Dans son principe, cette approche effectue une factorisation de Choleski de la matrice du premier membre représentée dans un format creux. En premier lieu, le motif de la matrice est analysé par l'algorithme du degré minimum ou la dissection emboîtée généralisée. Les informations retournées par la phase d'analyse sont utilisées pour guider la factorisation numérique. La solution est ensuite calculée à partir de la matrice des facteurs. La méthode multifrontale est implantée dans le code d'éléments finis CESAR-LCPC. Les temps de calcul obtenus sont comparés à ceux de l'algorithme existant dans ce code. L'influence de la méthode utilisée dans la phase d'analyse sur le temps de calcul lors de la factorisation et de la résolution est soulignée.

DOMAINE : *Sciences de l'ingénieur.*

ABSTRACT

IMPLEMENTATION WITHIN THE CESAR-LCPC CODE OF A MULTIFRONTAL SCHEME FOR SOLVING LARGE-DIMENSION LINEAR SYSTEMS

A multifrontal scheme for solving linear systems of equations in matrix form is presented herein. This approach has been based on Choleski factorization of the matrix on the left-hand side stored in sparse format. The matrix pattern will first be analyzed using either the minimum-degree algorithm or generalized nested dissection. The information provided by the analysis phase is then used to guide the numerical factorization stage. The solution of the linear system is to be obtained from coefficients of the factors. The multifrontal method has been implemented within the CESAR-LCPC finite element code and compared with the original algorithm in terms of computation time. The influence of the method used during analysis on the computation time in the numerical factorization and resolution phase will be underscored.

FIELD: *Engineering sciences.*

INTRODUCTION

Le logiciel CESAR-LCPC a été développé pour modéliser par la méthode des éléments finis une grande variété de problèmes rencontrés en génie civil. Ces problèmes peuvent être de mécanique, de diffusion, ou couplés par association de plusieurs phénomènes physiques. Leur mise en équation et leur discrétisation, qu'ils soient linéaires ou non, conduisent systématiquement à la résolution d'un système linéaire de la forme :

$$A \cdot x = b \qquad (1)$$

La matrice A d'ordre n formant le premier membre n'est en général que faiblement dense, d'autant plus que la configuration étudiée est tridimensionnelle. Le système d'équations (1) peut être résolu par une méthode directe, fondée sur l'élimination de Gauss, ou une méthode itérative, telle que le gradient conjugué ou la méthode GEMRES (GEneralized Minimum RESidual method), construisant une séquence de solutions approchées. CESAR-LCPC dispose à l'heure actuelle des deux types de méthodes pour la résolution de systèmes linéaires. L'utilisation du gradient conjugué (module LIGC) est toutefois limitée aux problèmes linéaires. Les méthodes itératives présentent l'avantage de ne pas modifier la matrice originale et d'utiliser moins d'espace mémoire qu'une méthode directe. La convergence rapide de ce type d'approche nécessite cependant un bon conditionnement de la matrice A, obtenu parfois au prix d'un préconditionnement de celle-ci. En outre, les gains en termes de calcul obtenus par l'emploi d'une méthode itérative en lieu et place d'une méthode directe s'estompent lorsque le nombre de seconds membres augmente. En raison de ces dernières considérations, on se focalisera dans cet article sur les méthodes directes.

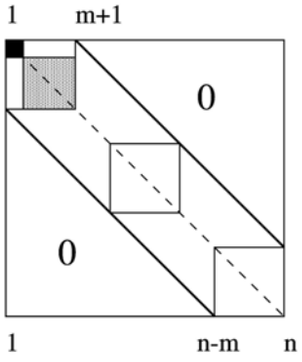


Figure 1
Exemple d'une matrice bande de largeur de bande m.

L'approche directe la plus immédiate pour résoudre le système d'équations (1) consiste à traiter A comme une matrice dense d'ordre n, tous les coefficients nuls ou non nuls de cette matrice étant pris en compte, et à utiliser une librairie standard de type LAPACK [1]. Cette technique montre toutefois vite ses limites en termes d'espace mémoire et de temps de calcul lorsque le nombre n d'inconnues devient grand. Une telle situation se produit, par exemple, dans l'étude de cas réalistes où un maillage fin est indispensable pour assurer la convergence du modèle. Une seconde approche consiste à réordonner les coefficients de la matrice A pour les contenir à l'intérieur d'une bande de largeur constante m comme le montre la figure 1. On parle alors de matrice bande. Le nombre d'opérations arithmétiques nécessaires pour la résolution du système est en $O(m^2n)$ si m est petit devant n, à comparer à $O(n^3)$ dans le cas dense. La technique utilisée actuellement dans CESAR est fondée sur l'exploitation du format dit *ligne-de-ciel*, terme utilisé pour désigner une matrice bande dont la largeur de bande est variable. Le dernier type de méthodes tire profit de la structure *creuse* de la matrice A, celle-ci ne comportant la plupart du temps qu'un petit nombre de coefficients non nuls.

En pratique, les performances des codes généraux pour matrices creuses peuvent être bien inférieures aux performances de pic des machines modernes sur lesquelles ils sont utilisés, leur implantation, inadaptée à l'architecture complexe de ces calculateurs, induisant des retards dans le traitement des données par le processeur. Des noyaux tels que le niveau 3 de la librairie BLAS [2], collection de sous-routines effectuant des opérations élémentaires sur des matrices, tirent profit de la hiérarchie de la mémoire et approchent ainsi la performance de pic du processeur. Cela vaut en particulier pour la routine _GEMM utilisée pour le calcul d'un produit de deux matrices. De fait, les méthodes creuses actuelles les plus performantes sont les méthodes frontales [3], multifrontales [4] et supernodales [5] structurées autour d'appels aux routines de la librairie BLAS.

On présente ici l'implantation dans CESAR d'un code pour matrices creuses fondé sur la méthode de Gauss multifrontale pour la résolution d'un système linéaire d'équations, symétrique défini positif. Le document est structuré comme suit. On rappelle d'abord le principe de la méthode de Gauss et on établit le lien avec la factorisation LU d'une matrice. On définit ensuite la notion de remplissage dans une matrice creuse et on présente les principales techniques utilisées pour le contenir. Enfin, la méthode multifrontale est introduite et les résultats relatifs à sa mise en œuvre dans CESAR sont présentés.

Méthode de Gauss et factorisation LU

Le système d'équations à résoudre, donné sous forme matricielle par l'équation (1), peut également s'écrire :

$$\begin{cases} a_{11}x_1 & +a_{12}x_2 & \cdots & +a_{1k}x_k & \cdots & +a_{1n}x_n = & b_1 \\ a_{21}x_1 & +a_{22}x_2 & \cdots & +a_{2k}x_k & \cdots & +a_{2n}x_n = & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots = & \vdots \\ a_{n1}x_1 & +a_{n2}x_2 & \cdots & +a_{nk}x_k & \cdots & +a_{nn}x_n = & b_n \end{cases} \quad (2)$$

avec a_{ij} coefficient de la $i^{ième}$ ligne et $j^{ième}$ colonne de la matrice A. Le but de la méthode de Gauss est d'aboutir en n étapes au système linéaire équivalent :

$$\begin{cases} a_{11}^n x_1 & +a_{12}^n x_2 & \cdots & +a_{1k}^n x_k & \cdots & +a_{1n}^n x_n = & b_1^n \\ & a_{22}^n x_2 & \cdots & +a_{2k}^n x_k & \cdots & +a_{2n}^n x_n = & b_2^n \\ & & \ddots & a_{kk}^n x_k & \cdots & \vdots \quad = & \vdots \\ & & & & \ddots & a_{nn}^n x_n = & b_n^n \end{cases} \tag{3}$$

Ce système d'équations est beaucoup plus facile à résoudre que le système original, car triangulaire supérieur. On l'écrit de façon équivalente $A^n.x = b^n$, avec A^n triangulaire supérieure.

Dans son principe, la méthode de Gauss consiste à éliminer à chaque étape k l'inconnue x_k des équations $k + 1$, ... n. Au début de l'étape k, le système linéaire noté, $A^{k-1}.x = b^{k-1}$, se présente sous la forme suivante :

$$\begin{cases} a_{11}^{k-1} x_1 & +a_{12}^{k-1} x_2 & \cdots & +a_{1k}^{k-1} x_k & \cdots & +a_{1n}^{k-1} x_n = & b_1^{k-1} \\ & a_{22}^{k-1} x_2 & \cdots & +a_{2k}^{k-1} x_k & \cdots & +a_{2n}^{k-1} x_n = & b_2^{k-1} \\ & & \ddots & \vdots & \cdots & \vdots \quad = & \vdots \\ & & & a_{kk}^{k-1} x_k & \cdots & +a_{kn}^{k-1} x_n = & b_k^{k-1} \\ & & & a_{k+1k}^{k-1} x_k & \cdots & +a_{k+1n}^{k-1} x_n = & b_{k+1}^{k-1} \\ & & & \vdots & \vdots & \vdots \quad = & \vdots \\ & & & a_{nk}^{k-1} x_k & \cdots & +a_{nn}^{k-1} x_n = & b_n^{k-1} \end{cases} \tag{4}$$

Les équations numérotées de k à n constituent le système réduit obtenu à l'issue des k-1 premières étapes de la factorisation. Il correspond à la sous-matrice de A^{k-1} formée des k-1 dernières lignes et colonnes et appelée *matrice réduite*.

On élimine l'inconnue x_k de l'équation i (i = k + 1 ,... n) en lui ajoutant l'équation k prémultipliée par $-a_{ik}^{k-1}/a_{kk}^{k-1}$. Cela suppose que le coefficient a_{kk}^{k-1}, *pivot de Gauss*, soit non nul. Le système obtenu à l'issue de cette opération s'écrit $A^k.x = b^k$. Il est relié au système précédent par les formules :

$$A^k = L^k \cdot A^{k-1} \tag{5}$$

$$b^k = L^k \cdot b^{k-1} \tag{6}$$

avec

$$L^k = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & & & \vdots \\ \vdots & 0 & 1 & \ddots & & \vdots \\ \vdots & \vdots & -\dfrac{a_{k+1k}^{k-1}}{a_{kk}^{k-1}} & 1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & 0 & 1 & 0 \\ 0 & 0 & -\dfrac{a_{k+1k}^{k-1}}{a_{kk}^{k-1}} & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

La matrice A^n triangulaire supérieure obtenue à la fin de la factorisation s'écrit donc :

$$A^n = \prod_{i=1}^{n} L^i \cdot A = G \cdot A \tag{8}$$

G est une matrice triangulaire inférieure car produit de matrices triangulaires inférieures, et son inverse l'est également. A est donc le produit d'une matrice triangulaire inférieure, G^{-1}, et d'une matrice triangulaire supérieure, A^n. Ces deux dernières matrices sont donc les matrices des facteurs LU de A, une telle décomposition, quand elle existe, étant unique.

D'un point de vue matriciel, la mise en œuvre de la méthode de Gauss revient donc à effectuer la factorisation LU de la matrice A. Lorsque la matrice est symétrique, la matrice U peut s'écrire comme le produit d'une matrice diagonale D et de la transposée de L, laquelle possède une diagonale unité. On parle alors de *factorisation de Crout*. Si la matrice est définie positive, il s'agit dans ce cas de trouver la matrice L qui, multipliée par sa transposée, donne la matrice A. On effectue dans ce cas une *factorisation de Choleski*.

LE REMPLISSAGE DANS LES MATRICES DES FACTEURS LU

Considérons une matrice A creuse, c'est-à-dire une matrice dont seuls certains coefficients sont non nuls. Les matrices L et U obtenues par décomposition LU de la matrice A comportent en général plus de coefficients à elles deux que A. En effet, certains coefficients de la matrice A ayant une valeur nulle au départ prennent une valeur non nulle au cours de la factorisation comme l'illustre la figure 2. Ce phénomène est appelé *remplissage* et joue un rôle très important. Plus il y a de nouveaux coefficients, plus le nombre d'opérations arithmétiques à réaliser pour factoriser A est importante. Afin de minimiser les ressources mémoire (ou disque) et le temps de calcul, il est donc primordial de limiter le remplissage.

La technique communément employée pour ce faire consiste à permuter les lignes et les colonnes de la matrice originale. Dans l'exemple donné sur la figure 3, on observe que les matrices des facteurs sont denses dans leurs parties triangulaires inférieures et supérieures respectivement. Si l'on permute la première et la dernière colonne ainsi que la première et la dernière ligne de la matrice originale, il n'y a plus remplissage dans les matrices des facteurs.

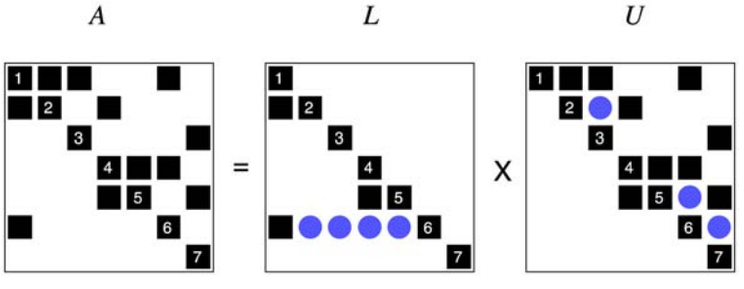
Ces permutations des lignes et des colonnes reviennent à pré- et post-multiplier la matrice A par une matrice de permutation P sur les lignes et Q sur les colonnes. Les coefficients de la matrice P (resp. Q) sont nuls sauf si les lignes i et j (resp. les colonnes i et j) sont permutées, auquel cas P_{ij} (resp. Q_{ji}) vaut 1. Dans le cas de l'exemple de la figure 3, P est donnée par :

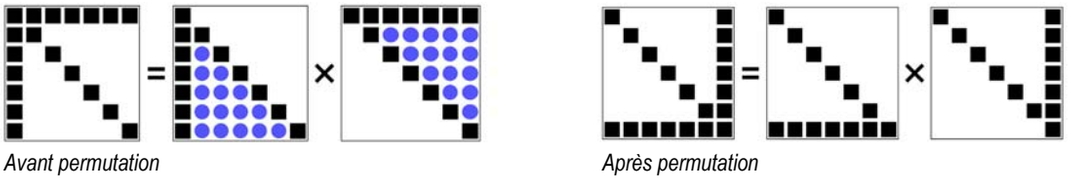
$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (9)$$

et $Q = P^T$. La minimisation du remplissage consiste donc à trouver les matrices P et Q optimales. On factorise alors la matrice P.A.Q. Dans le cas d'une matrice symétrique, on effectue une permutation symétrique des lignes et des colonnes ($Q = P^T$) de façon à préserver la symétrie de la matrice originale. La recherche des matrices P et Q optimales étant un problème NP-*complet* [6], des méthodes

Figure 2

Remplissage dans une matrice creuse :
■ correspond aux positions de coefficients non nuls dans la matrice originale.
● correspond à des coefficients dus au remplissage.





Avant permutation *Après permutation*

Figure 3

Minimisation du remplissage dans une matrice creuse par permutation de lignes et de colonnes dans la matrice originale.

heuristiques sont utilisées en pratique. On présente dans la suite deux de ces techniques conçues pour le cas de matrices symétriques possédant des coefficients sur la diagonale : l'algorithme du degré minimum approché [7] et la dissection emboîtée généralisée [8, 9].

L'algorithme du degré minimum approché

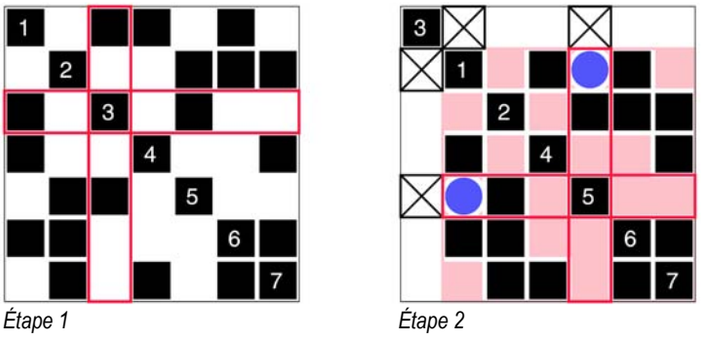
Cette approche réalise une factorisation symbolique de la matrice A pour déterminer l'ordre des lignes et des colonnes, ou *séquence de pivotage*, minimisant le remplissage. À chaque étape, le pivot de Gauss est choisi sur la diagonale de la matrice *réduite* formée des lignes et des colonnes du système restant à factoriser. Cela permet de conserver le caractère symétrique de cette matrice tout au long de la factorisation.

La stratégie est la suivante. À l'étape k, on choisit la colonne i de la matrice réduite ayant le moins de coefficients, coefficient diagonal exclu. Cette grandeur est appelée *degré externe* de la colonne i. Le degré externe de la ligne i est égal au degré externe de la colonne i, la matrice réduite étant symétrique, tout comme la matrice originale. Par analogie avec la méthode de Gauss décrite dans la section 2, le choix de la colonne ayant le degré minimum revient à éliminer l'inconnue x_i des n-k équations restantes, l'équation i ayant le nombre minimal de termes.

Dans la suite, B et C étant deux ensembles, on note B/C l'ensemble de composantes de B n'appartenant pas à C. La structure de la colonne pivot i est donnée par la liste d'indices de ligne L_i. Le problème étant symétrique, toutes les colonnes dont l'indice figure dans $L_i \setminus \{i\}$, c'est-à-dire toutes les colonnes de L_i sauf la colonne i, doivent être mises à jour pour prendre en compte un éventuel remplissage et leur degré recalculé. En pratique, cette opération, coûteuse en temps et en espace, n'est pas effectuée. On garde juste une trace de la relation de dépendance entre la colonne i et les colonnes de L_i. Ainsi, la structure de la matrice réduite au cours de la factorisation est représentée de façon implicite à l'aide du motif original de cette matrice, du motif des colonnes de la matrice L des facteurs (ou plus précisément d'un certain nombre d'entre elles), et des relations de dépendance entre ces dernières colonnes et les colonnes de la matrice réduite. L'ensemble de ces données constitue le *graphe quotient* de la matrice A. L'espace requis pour le stockage de ce graphe est inférieur à celui nécessaire pour représenter la structure de la matrice A originale, ce qui constitue l'un des avantages de l'algorithme du degré minimum. L'utilisation du graphe quotient est l'une des principales améliorations de l'algorithme original de Tiney et Walker.

La structure d'une colonne de la matrice réduite affectée par le choix de la colonne *i* comme pivot est accessible, mais n'est calculée que lorsque cette colonne devient pivot à son tour. Son degré exact n'est donc pas calculé. On calcule une borne supérieure de ce degré proche de sa véritable valeur. Cette opération de mise à jour des degrés est la plus coûteuse en temps, moins toutefois que le calcul du degré exact. La technique d'évaluation du degré constitue l'une des différences entre les diverses variantes de l'algorithme du degré minimum comme MMD [10] ou AMD [11].

La figure 4 présente un exemple de mise en œuvre de la stratégie du degré minimum sur une matrice d'ordre 7. À la première étape, la colonne 3 de degré 2 est choisie comme pivot et modifie la structure des colonnes 1 et 5. La structure de la colonne 3 est stockée ainsi que le lien entre cette colonne et les colonnes 1 et 5, et les degrés sont mis à jour. La colonne 5 est la colonne de plus petit degré dans la



Étape 1 *Étape 2*

Figure 4
Principe de l'algorithme du degré minimum approché.

matrice réduite ; elle est donc choisie comme pivot à l'étape 2. Son motif L_5 est constitué et le degré des colonnes de $L_5 \setminus \{5\}$ est mis à jour.

À chaque étape de la factorisation, l'algorithme choisit la colonne de la matrice réduite ayant le moins de coefficients, sans véritable stratégie lorsque plusieurs colonnes sont en « compétition ». De ce fait, ce choix est optimal à un moment donné (pour réduire le nombre de coefficients dans une colonne de la matrice L), mais son influence sur la structure de la matrice réduite n'est pas prise en compte. L'algorithme du degré minimum est ainsi une méthode dite *locale*. Par opposition, la dissection emboîtée généralisée que l'on présente dans le paragraphe suivant vise à réduire le remplissage par une analyse de la structure de la matrice A. Il s'agit là d'une méthode *globale.*

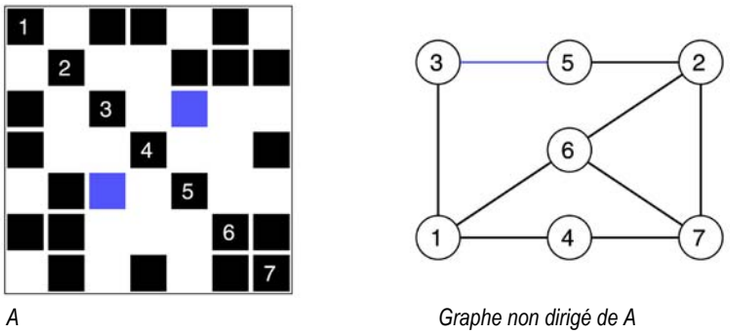
La dissection emboîtée généralisée

La méthode de la dissection emboîtée généralisée est, comme l'algorithme du degré minimum, conçue pour les matrices symétriques n'ayant aucun coefficient nul sur leur diagonale. Elle s'appuie sur un *graphe non dirigé* représentant la structure de la matrice à factoriser. Le graphe G(V,E) d'une matrice symétrique A est constitué d'un ensemble V = {1, 2,..., n-1, n} de nœuds et d'un ensemble E de liaisons entre les nœuds de V.< i, j > est une liaison de E si et seulement si A_{ij} est non nul. Dans le cas d'une matrice symétrique, si A_{ij} est non nul, alors A_{ji} l'est également. Dans ce cas, une seule liaison est nécessaire ; elle est dite *non-dirigée*. La figure 5 donne un exemple de graphe non dirigé pour une matrice symétrique.

Le principe de la dissection emboîtée généralisée repose sur une bissection récursive du graphe G(V,E) de la matrice A. Une première bissection est réalisée en sélectionnant un ensemble de nœuds formant un *séparateur*. Ce séparateur, composé d'un nombre minimal de nœuds, est ensuite retiré du graphe original et génère une partition de celui-ci en deux sous-graphes non connectés. Un nœud d'une composante de cette partition n'est lié dans le graphe original qu'à des nœuds appartenant à la même composante que lui ou au séparateur. Chacune des deux partitions est ensuite bissectée de façon récursive suivant le même principe, jusqu'à ce que les sous-graphes générés successivement soient de taille suffisamment petite. La figure 6 présente un exemple de bissection récursive d'un graphe. Les nœuds du séparateur 1 partitionnent le graphe original en deux composantes, 1 et 2. La composante 2 est bissectée en deux composantes 3 et 4 en soustrayant de ce graphe les nœuds formant le séparateur 2. La composante 3 est bissectée à son tour à l'aide du séparateur 3, générant les composantes 5 et 6.

Une fois le graphe partitionné, les nœuds des différentes composantes et des séparateurs successifs sont réordonnés, ce qui revient à choisir un ordre de permutation des colonnes (et des lignes) des sous-matrices de la matrice originale associées à ces nœuds. Cette opération s'effectue suivant deux principes. Tout d'abord, les nœuds d'un séparateur sont réordonnés après les nœuds des deux composantes qu'ils génèrent. Ensuite, les nœuds de chaque composante sont réordonnés par une méthode locale.

La figure 7 présente la mise en œuvre du premier principe considérant la partition du graphe donnée par la figure 6. Les blocs diagonaux principaux de la matrice permutée correspondent au séparateur 1 et aux composantes 1 et 2. Les coefficients hors diagonale correspondent aux liaisons entre les nœuds du séparateur et les nœuds des deux sous-graphes qu'il génère. Le bloc diagonal



A — *Graphe non dirigé de A*

Figure 5

Graphe non dirigé d'une matrice symétrique. La liaison <3,5> (—) du graphe représente les coefficients A_{35} et A_{53}.

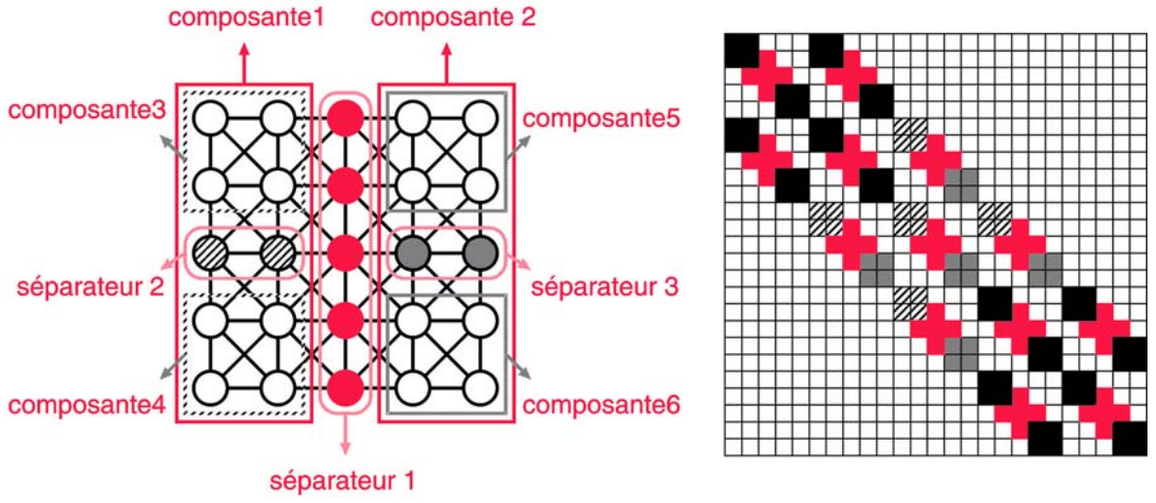


Figure 6

Partition du graphe d'une matrice. Les coefficients ■, ■ et ▨ dans la matrice correspondent à des liaisons dans le graphe entre les nœuds ●, ● et ◍ des séparateurs et les nœuds des composantes successives qu'ils génèrent.

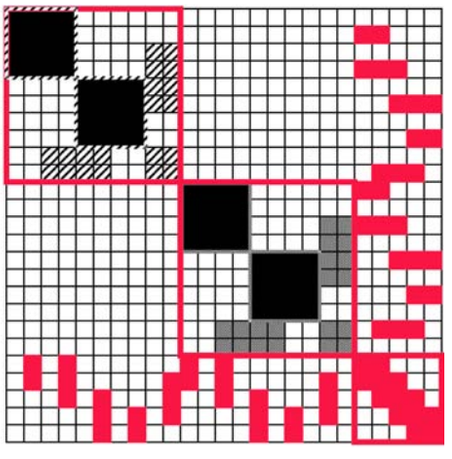


Figure 7

Matrice réordonnée suivant le principe de la bissection emboîtée. Chaque bloc diagonal correspond à un sous-graphe généré par bissection récursive.

supérieur gauche sur la figure 7 comprend lui-même trois blocs diagonaux correspondant au séparateur 2 et aux composantes 3 et 4. Les coefficients non diagonaux à l'intérieur de ce bloc principal représentent les liaisons entre les nœuds du séparateur 2 et des nœuds des composantes 3 et 4.

L'application du second principe consiste à réordonner les lignes et les colonnes des blocs diagonaux, correspondant chacun à une composante du graphe, afin de minimiser le remplissage dans ce bloc. La méthode utilisée est locale. En pratique, on utilise les variantes MMD ou AMD de l'algorithme du degré minimum. La permutation des lignes et des colonnes et d'un bloc diagonal correspondant à une composante d'une partition n'occasionne aucun remplissage dans le bloc diagonal associé à la seconde composante de cette même partition.

Pour de nombreux problèmes, l'ordre de permutation retourné par la dissection emboîtée généralisée est de meilleure qualité que celui obtenu à l'aide de l'algorithme du degré minimum, dans le sens où il conduit à un remplissage moins important, mais demande plus de temps pour être calculé.

LA MÉTHODE MULTIFRONTALE

On considère dans la suite une matrice A symétrique définie positive. On suppose que les lignes et les colonnes ont été permutées de façon symétrique suivant l'ordre donné par l'algorithme du degré minimum ou la méthode de la dissection emboîtée généralisée. Dans la suite, A fait référence à la matrice permutée. On suppose que cette matrice n'est pas diagonale par blocs, auquel cas on traite successivement chacun des blocs diagonaux.

La méthode multifrontale que nous allons utiliser [4] pour effectuer la factorisation de Choleski de A tire ses performances du fait que les opérations de factorisation sont effectuées dans de petites matrices denses, permettant l'utilisation des routines des niveaux 2 et 3 de la librairie BLAS dont les performances de pics approchent celles du processeur.

Dans cette méthode, la matrice réduite au début de l'étape k de la factorisation s'écrit sous la forme :

$$A^k = A'_k + \sum_{j=1,\ldots,k-1} E_j \qquad (10)$$

où E_j est une matrice dense, appelée *élément généré*, contenant des coefficients de mise à jour permettant de prendre en compte la $j^{ième}$ étape de la factorisation. A'_k contient les coefficients originaux de la matrice réduite permutée restant à l'issue des k-1 premières étapes.

Les opérations de factorisation de la $k^{ième}$ étape portent sur la colonne k. Supposons, sans perte de généralité, que cette colonne, de structure L_k, n'est pas affectée par les k-1 premières étapes, de sorte que les coefficients de cette colonne sont ceux de la matrice réduite originale. Les coefficients de la ligne et de la colonne k sont alors assemblés dans une matrice frontale carrée F, dense, dont l'ordre est la taille de la liste L_k. On associe à la matrice frontale la liste L_k d'indices de ligne et de colonne. On écrit :

$$F_k = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & 0 \end{bmatrix} \qquad (11)$$

avec F_{11}, scalaire, correspondant à A'_{kk} . F_{21} est un vecteur-colonne contenant les coefficients non nuls de la colonne k hors diagonale. F_{12} est un vecteur-ligne contenant les coefficients non nuls de la ligne k. Par symétrie, $F_{12} = F_{21}^T$. La formation de la $k^{ième}$ colonne de la matrice L des facteurs est réalisée en divisant F_{21} par $\sqrt{F_{11}}$, cette opération étant possible car le pivot F_{11} est toujours strictement positif dans le cas d'une matrice définie positive. Les colonnes dont l'indice figure dans $L_k \setminus \{k\}$ doivent être mises à jour pour prendre en compte les opérations de factorisation relatives à l'étape k. Plus précisément, si i est l'indice de ligne de L_k associé au coefficient i_1 de F_{21} et j est l'indice de L_k associé au coefficient j_1 de F_{12}, on doit effectuer :

$$A'_{ij} \leftarrow A'_{ij} - \frac{(F_{21})_{i_1}(F_{12})_{j_1}}{F_{11}} \qquad (12)$$

L'ensemble des produits intervenant dans cette opération est formé en calculant, à l'aide d'une routine du niveau 3 de BLAS, la matrice dense $S = -F_{21}F_{11}^{-1}F_{12}$ appelée *complément de Schur*. L'assemblage de cette matrice dans la matrice réduite n'est pas effectué immédiatement, mais retardé jusqu'à ce qu'une colonne affectée par la $k^{ième}$ étape de la factorisation soit choisie comme pivot. Le complément de Schur est stocké, formant ainsi l'élément généré E_k, et associé à la liste d'indices $L_k \setminus \{k\}$. Il est assemblé dans la matrice frontale correspondant à l'élimination de la variable p, p étant la première colonne de la liste $L_k \setminus \{k\}$ à être choisie comme pivot. On dit que p est le *parent* de k, et inversement que k est un *enfant* de p. On a $L_k \subseteq L_p$. Une variable peut avoir plusieurs enfants car une même colonne peut être affectée par plusieurs étapes de la factorisation. Il faut alors assembler plusieurs éléments générés dans la matrice frontale correspondante. La matrice frontale associée au choix de la colonne p comme pivot s'écrit :

$$F_p = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \qquad (13)$$

Les coefficients non nuls de la matrice F_{22} correspondent à des lignes et des colonnes parmi lesquelles aucun pivot ne sera choisi (elles sont dites *partiellement sommées*) et proviennent de l'assemblage d'un ou de plusieurs éléments générés dans F_p. Les opérations de factorisation lors de la $p^{ième}$ étape sont menées de la même façon que précédemment.

Chaque étape de la méthode multifrontale est donc la succession des opérations suivantes : formation du motif de la ligne et de la colonne pivot, formation de la matrice frontale, assemblage des coefficients originaux et des éléments générés dans la matrice frontale, réalisation des opérations d'élimination au sein de la matrice frontale et stockage de l'élément généré.

La factorisation d'une matrice par un schéma multifrontal peut être représentée par un *arbre d'assemblage*. Chaque *nœud* de l'arbre correspond aux opérations de factorisation dans une matrice frontale. Chaque *branche* de l'arbre correspond à l'opération d'assemblage d'un élément généré dans sa matrice frontale parent. Cet arbre, utilisé pour guider la factorisation numérique de A, est constitué au cours d'une phase de factorisation symbolique de la matrice permutée. Celle-ci est menée conjointement à la recherche de la séquence optimale de permutation des lignes et des colonnes dans le cas de l'algorithme du degré minimum, ou postérieurement à celle-ci lorsque la dissection emboîtée généralisée est utilisée.

En pratique, plusieurs inconnues sont éliminées dans une même matrice frontale, conduisant à l'assemblage de plusieurs lignes et colonnes pivots. Typiquement, cette situation se rencontre lorsque deux colonnes p_1 et p_2 ont le même motif, soit $L_{p1} \setminus \{p_1\} = L_{p2}$. La liste d'indices associée à la matrice frontale est ainsi la même, hors lignes et colonnes pivots, que l'on élimine ces deux inconnues dans deux matrices frontales différentes ou au sein de la même matrice frontale. Le choix de la seconde option n'introduit aucun remplissage supplémentaire dans la matrice L des facteurs. Les variables p_1 et p_2 constituent une *supervariable*. L'intérêt de la formation de supervariables provient du fait que F_{21} (resp. F_{12}) comporte alors plusieurs colonnes (resp. lignes) et que le calcul du complément de Schur, un produit entre deux matrices, est effectué à l'aide de la routine _GEMM du niveau 3 de BLAS. Les performances de cette routine, en termes de nombre d'opérations sur les flottants à la seconde (ou *flops*), sont d'autant plus grandes que les matrices F_{21} et F_{12} comportent un nombre important de lignes et de colonnes, dans une certaine limite dépendant des caractéristiques matérielles de la machine. On a donc intérêt à former le plus grand nombre possible de supervariables. La détection des supervariables s'effectue au cours de la factorisation symbolique.

La figure 8 détaille la factorisation d'une matrice symétrique définie positive d'ordre 7 par la méthode multifrontale exposée dans ce paragraphe. L'ordre de pivotage est l'ordre naturel des

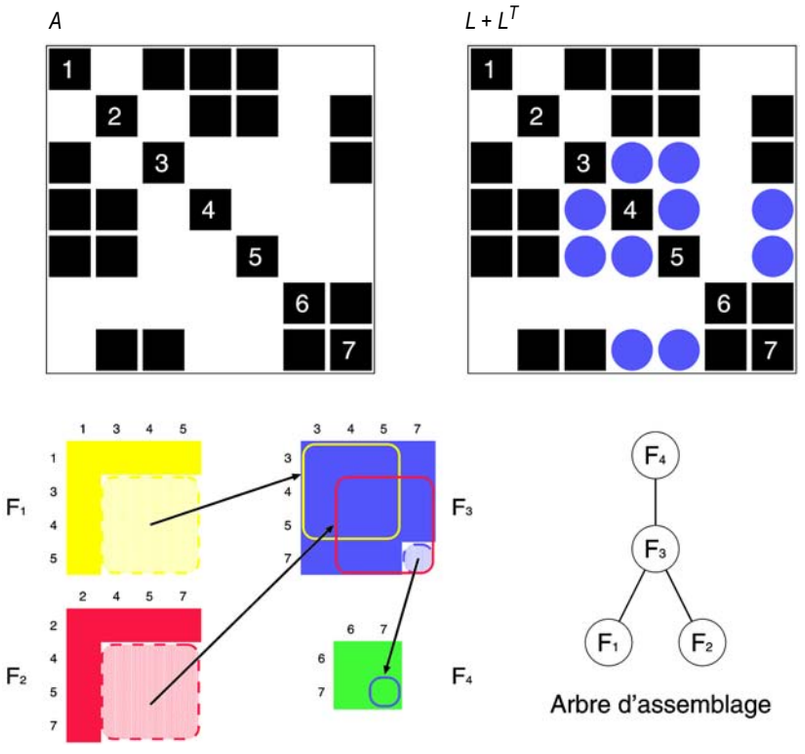


Figure 8
Étapes de la factorisation d'une matrice symétrique définie positive d'ordre 7 par la méthode multifrontale.

inconnues. À la première étape, l'inconnue 1 est choisie comme pivot. Les coefficients non nuls originaux de la ligne et la colonne 1 sont assemblés dans la matrice frontale F_1. La première colonne des facteurs est formée, le complément de Schur calculé, et l'élément généré est placé sur la pile. La deuxième étape se passe comme la première, puisque la colonne 2 est choisie comme pivot et qu'elle n'est pas affectée par les opérations d'élimination relatives au premier pivot. À l'issue de ces deux premières étapes, le remplissage se produisant dans les lignes et les colonnes 3, 4 et 5 donne à celles-ci une structure identique. Les trois inconnues correspondant à ces colonnes sont éliminées au cours de la troisième étape. On forme la matrice frontale F_3 dans laquelle on assemble les coefficients originaux des lignes et des colonnes 3, 4 et 5 de la matrice réduite. La colonne 3 est la première colonne de la liste $L_1 \setminus \{1\}$ à être choisie comme pivot. L'élément généré constitué à la première étape doit être assemblé dans F_3. De la même façon, la colonne 4 est la première colonne de la liste $L_2 \setminus \{2\}$ à être choisie comme pivot nécessitant l'assemblage de l'élément généré de l'étape 2 dans F_3. Les inconnues 6 et 7 sont éliminées dans la dernière matrice frontale. L'élément généré de l'étape 3 est assemblé dans F_4 pour mettre à jour la colonne 7. L'arbre d'assemblage sur la figure 8 récapitule l'ensemble du processus d'assemblage et d'élimination.

Les coefficients de la matrice L des facteurs ayant été calculés, la solution de l'équation (1) est obtenue par résolutions successives de $Ly = b$ et $L^T x = y$ en deux phases, dites de *descente* et de *remontée*.

RÉSULTATS NUMÉRIQUES

Le schéma multifrontal présenté dans les sections précédentes a été implanté dans le module LINE de CESAR, qui modélise des problèmes d'élasticité linéaire. Ce code de résolution comporte trois étapes.

❶ Une analyse du motif de la matrice originale est réalisée afin de trouver une séquence de permutation symétrique des lignes et des colonnes permettant de minimiser le remplissage. Cette étape est réalisée par l'algorithme du degré minimum approché ou par la méthode de la dissection emboîtée. Le premier algorithme a été programmé suivant la description donnée par Davis et *al.* [11]. L'implantation de la dissection emboîtée généralisée utilisée est celle proposée dans la librairie METIS [9]. Lorsque cette dernière méthode est utilisée, une factorisation symbolique de la matrice permutée est effectuée pour constituer l'arbre d'assemblage guidant la factorisation numérique.

❷ La factorisation numérique de la matrice permutée est ensuite réalisée à l'aide des informations retournées par la phase d'analyse.

❸ La solution est calculée à partir des coefficients de la matrice des facteurs.

L'algorithme original utilisé dans CESAR pour la résolution d'un système linéaire d'équations est conçu pour des matrices stockées en format ligne de ciel. Comme le schéma multifrontal, il réalise une factorisation de Choleski de la matrice du premier membre du système symétrique et calcule la solution à partir des coefficients de la matrice des facteurs. En revanche, l'étape d'analyse est absente. Plus précisément, elle est menée dans le pré-processeur de CESAR sur la base d'un algorithme minimisant la largeur de bande moyenne de la matrice à factoriser. Dans la suite, on compare les temps de calcul de cet algorithme, que l'on désigne par le terme « solveur ligne de ciel », et ceux obtenus à l'aide de l'approche proposée dans cet article et appelée « solveur multifrontal » en spécifiant dans ce dernier cas si l'on utilise l'algorithme du degré minimum ou la dissection emboîtée généralisée dans la phase d'analyse.

Cette étude comparative est menée pour différents cas. Les caractéristiques des matrices à factoriser correspondant à chaque cas sont données dans le tableau I. Toutes sont symétriques définies positives. Les systèmes d'équations à résoudre sont de grande taille puisque le nombre d'inconnues dépasse 50 000. La matrice à factoriser pour le cas *inter2* est une permutation de celle du cas *inter*. La largeur de bande moyenne est moins importante pour la première des deux matrices.

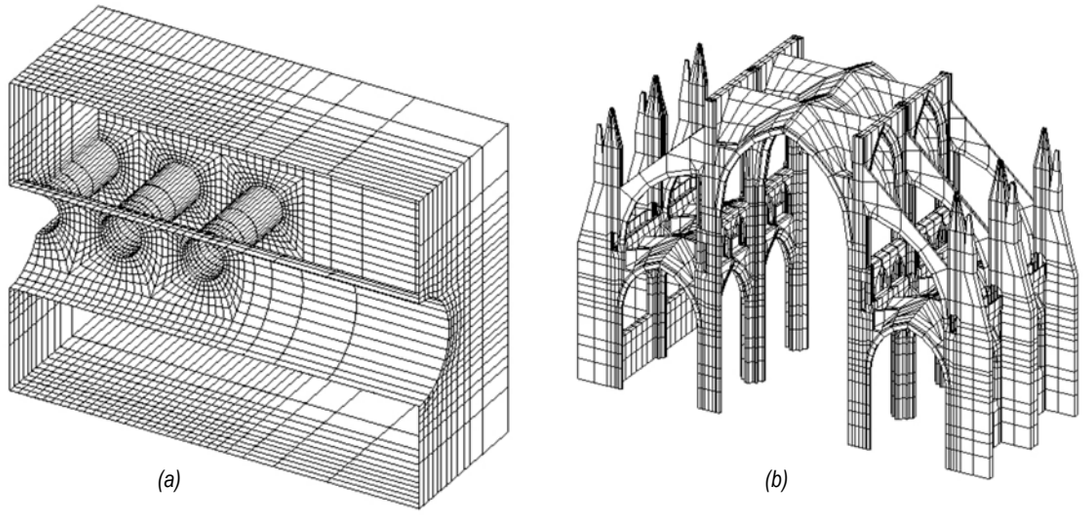
Le code CESAR a été compilé en 64 Bits sur une station Sun Fire 880 à quatre processeurs cadencés à 750 MHz. Cette machine possède 16 Go de mémoire vive, 1 Mo de cache L2 et 64 k de cache L1. Le code est séquentiel et n'utilise donc qu'un processeur sur les quatre disponibles. Les routines BLAS utilisées sont celles de la librairie SunPerformance livrée avec la machine. Ces routines sont nécessaires pour le solveur multifrontal.

Il a été choisi de faire tourner le solveur ligne de ciel avec le strict minimum d'espace mémoire. Cette option nécessite l'emploi de fichiers en accès direct pour le stockage de la matrice, mais conduit au temps de factorisation minimal, au détriment toutefois du temps passé dans la phase de résolution. Comme on le verra, la factorisation de la matrice est l'étape qui prend le plus de temps. Le solveur multifrontal n'utilise aucun fichier de stockage. Le calcul s'effectue donc totalement en mémoire dans ce cas.

TABLEAU I

Cas d'étude. *N* est la dimension du problème. nnz est le nombre de coefficients dans la partie triangulaire inférieure stricte de la matrice. Toutes les matrices sont symétriques définies positives

Cas	N	nnz	Origine
Stras	51021	1523547	Cathédrale de Strasbourg (Fig. 9b)
Cube16	52190	4037267	Cube 3D. Maillage 16*16*16 éléments H20
Ramses	74442	5193430	Tombeau de Ramsès
Inter	138645	10678528	Intersection de deux galeries dans un massif
Inter2	138645	10678528	Idem Inter
Tr10	198501	11916382	Galerie dans un massif
Galmu	249559	19372284	Intersection de plusieurs galeries dans un massif (Fig. 9a)



(a) (b)

Figure 9

Maillages correspondant au cas galmu (a) et stras (b) (cf. tableau I).

Le tableau II donne les temps CPU passés dans chaque étape du calcul pour le solveur ligne de ciel (SKY) et le solveur multifrontal utilisant soit l'algorithme du degré minimum (AMD), soit la méthode de la dissection emboîtée généralisée (GND). Les temps inférieurs à deux fois le meilleur temps (en gras) sont soulignés. Le temps nécessaire pour mener la phase d'analyse (solveur multifrontal uniquement) est inclus dans le temps d'assemblage. On observe que ce dernier est toujours plus important pour le solveur ligne de ciel, quand bien même une étape de moins est réalisée. Cela s'explique par le fait que le solveur multifrontal utilise une représentation de la matrice dans un format creux (CSC pour Compressed Sparse Column dit aussi format Harwell-Bœing). De ce fait, le nombre de coefficients à assembler est bien moins important que dans le cas d'un format ligne de ciel puisque seuls les coefficients non nuls sont pris en compte.

Le tableau II indique que l'algorithme du degré minimum est toujours nettement plus rapide que la dissection emboîtée généralisée. Cependant, cette méthode locale produit plus de remplissage dans la matrice des facteurs que la dissection emboîtée généralisée, nécessitant d'effectuer un plus grand nombre d'opérations arithmétiques lors de la factorisation numérique et la résolution. Il en découle un temps de calcul plus important pour ces deux phases.

Comme on l'a déjà écrit, la matrice du cas *inter2* est une permutation de celle du cas *inter* et se caractérise par une largeur de bande réduite. Cela a une influence directe sur le temps de factorisation du solveur ligne de ciel (divisé par 4) et dans une moindre mesure sur le temps de résolution (temps divisé par 1,7). Le solveur multifrontal est moins sensible à la permutation des lignes et des colonnes

TABLEAU II

Temps de calcul pour les différentes étapes de la factorisation. Comparaison entre le solveur ligne de ciel (SKY) et le solveur multifrontal utilisant soit l'algorithme du degré minimum (AMD), soit la dissection emboîtée généralisée (GND)

	Assemblage*			Factorisation			Résolution		
	AMD	**GND**	**SKY**	**AMD**	**GND**	**SKY**	**AMD**	**GND**	**SKY**
Stras	**8.3**	19.2	*17.8*	*20.1*	**14.8**	369.7	*1.0*	**0.9**	8.6
Cube16	**14.8**	37.3	*27.1*	757	**208**	2 382	6.2	**2.0**	13.5
Ramses	**26.6**	57.8	53.5	1 125	**310**	14 108	*10.0*	**5.5**	58.5
Inter	**61.0**	140.5	263.5	2 227	**618**	100 190	*18.2*	**10.0**	187
Inter2	**63.4**	138.2	202.9	2 965	**595**	25 135	21	**9.7**	109.8
Tr10	**52.2**	149.3	130.8	4 319	**1 165**	27 642	*28.8*	**15.3**	141.6
Galmu	**139.4**	279.3	438	11 093	**1 764**	139 365	*40.6*	**25.6**	312

* *Dans le cas de la méthode multifrontale, ce temps comprend à la fois le temps d'assemblage et le temps passé dans la recherche de la permutation optimale des lignes et des colonnes ainsi que la factorisation symbolique de la matrice permutée.*

puisqu'une phase d'analyse précède la factorisation numérique. Toutefois, on note que le cas *inter2* demande plus de temps que le cas *inter* (x 1,33 pour la factorisation et x 1,15 pour la résolution) lorsque l'algorithme du degré minimum est utilisé pour minimiser le remplissage. Cette versatilité toute relative des résultats illustre les manques de cette dernière méthode en terme de stratégie globale de pivotage. Dans le cas du solveur mutltifrontal GND, les temps de calcul sont quasiment identiques.

Quelle que soit la méthode retenue pour la minimisation du remplissage, le solveur multifrontal s'avère être beaucoup plus rapide que le solveur ligne de ciel. Les gains sur le temps de factorisation et de résolution permis par l'utilisation de cette méthode sont reportés dans le tableau III, considérant l'utilisation de la dissection emboîtée dans la phase d'analyse. On rappelle que le solveur ligne de ciel fonctionne avec un minimum d'espace mémoire, ce qui conduit à écrire les coefficients de la matrice des facteurs sur le disque. Cette option est caractéristique des conditions d'utilisation de CESAR pour la résolution de systèmes de grande taille. Elle permet de réduire le temps passé dans la phase de factorisation. Le solveur ligne de ciel est par contre pénalisé dans la phase de résolution puisque le calcul de la solution nécessite de lire deux fois sur le disque les coefficients de la matrice des facteurs. Notons que même en retranchant les temps d'entrée/sortie, le solveur multifrontal conserve l'avantage dans la phase de résolution.

La méthode multifrontale tire ses performances du fait que le calcul du complément de Schur est une opération de type produit de matrices, effectué à l'aide des routines du niveau 3 de la librairie BLAS.

TABLEAU III

Rapport des temps de calcul pour les phases de factorisation et de résolution. Comparaison entre le solveur ligne de ciel et le solveur multifrontal GND

	$\frac{T_{sky}^{fac}}{T_{gnd}^{fac}}$	$\frac{T_{sky}^{resol}}{T_{gnd}^{resol}}$
Stras	20,1	9,5
Cube16	11,4	6
Ramses	45,5	10,0
Inter	162	18,7
Inter2	42,2	11,3
Tr10	23,7	9,2
Galmu	79	12,2

Il est donc important de disposer d'une version de ces routines permettant de tirer profit de l'architecture mémoire de la machine par le biais de techniques de programmation telles que le déroulage de boucles et le *blocking* (le fait de travailler sur des blocs d'une matrice plutôt que la matrice entière). Pour le cas *ramses* étudié précédemment, l'utilisation de routines BLAS « basiques » (disponibles sur NetLiB) en lieu et place de celles contenues dans la librairie SunPerformance fait passer le temps de factorisation de 310 à 1 142 secondes. L'optimisation des routines du niveau 3 de BLAS est donc d'une grande importance pour l'amélioration des performances de codes de résolution de systèmes linéaires et fait l'objet de nombreux travaux concrétisés par la disponibilité de librairies optimisées [12, 13].

CONCLUSION

Un schéma multifrontal a été présenté pour la résolution d'un système linéaire symétrique défini positif d'équations, écrit sous forme matricielle. La méthode exposée consiste en une factorisation de Choleski de la matrice du premier membre et un calcul de la solution à partir de la matrice des facteurs. Elle tire ses performances du fait que les opérations de factorisation sont effectuées dans de petites matrices denses à l'aide des routines de la librairie BLAS. La résolution du problème comprend plusieurs étapes. Une analyse de la structure de la matrice à factoriser est menée afin de trouver une séquence de permutation symétrique des lignes et des colonnes de cette matrice minimisant le remplissage. Une factorisation symbolique de la matrice permutée est effectuée pour constituer l'arbre d'assemblage guidant la factorisation numérique. La factorisation numérique est alors réalisée, suivie du calcul de la solution. Ce schéma multifrontal a été implanté dans le module LINE de CESAR. Les résultats obtenus montrent que cette approche est nettement plus rapide pour la résolution de problèmes de grande taille que l'algorithme existant dans ce code. L'influence du choix de la méthode de minimisation du remplissage sur le temps de calcul a été soulignée. L'importance de disposer d'une version optimisée de BLAS pour tirer profit de l'architecture mémoire des machines modernes a été également montrée.

RÉFÉRENCES BIBLIOGRAPHIQUES

[1] DEMMEL J.W., LAPACK : A portable linear algebra library for supercomputers, *1989 IEEE Control Systems Society Workshop on Computer-Aided Control System Design*, December **1989.**

[2] DONGARRA J.J., DU CROZ J., DUFF I.S., HAMMARLING S., A set of level 3 Basic Linear Algebra Subprograms, *ACM Trans. MAth. Soft.* **16, 1990,** pp. 1-17.

[3] DUFF I.S., SCOTT J.A., MA42, a new frontal code for solving sparse unsymmetric systems, *Technical Report RAL 93-06*4, Rutherford Appleton Laboratory, **1993.**

[4] DUFF I.S., REID J.K., MA57 – a new code for the solution of sparse symmetric positive definite and indefinite systems, *Techical Report RAL 2002-024*, Rutherford Appleton Laboratory, **2002.**

[5] DEMMEL J.W, EISENSTAT S.C., GILBERT J.R., LI X.S., LIU J.W.H., A supernodal approach to sparse partial pivoting, *Technical Report UCB-CSD-95-883*, Computer Science Division, U.C. Berkeley, Berkeley, California, **1995.**

[6] YANNAKAKIS M., Computing the minimum fill is NP-complete, *SIAM J. Alg. Disc. Math.* **2**, **1981,** pp. 77-79.

[7] TINEY W.F., WALKER J.W., Direct solution of sparse network equations by optimally ordered triangular factorization, *Proceedings of IEEE*, **55**, **1967,** pp. 1801-1809.

[8] HENDRICKSON N., LELAND R., A multilevel algorithm for partitioning graphs, *Technical Report sand 93-1301*, Sandia National Laboratory, **1993.**

[9] KARYPIS G., KUMAR V., A fast and high quality multilevel scheme for partionning irregular graphs, *SIAM J. Sci. Comp.*, **20**, **1999,** pp. 259-192.

[10] LIU J.W.H., Modification of the minimum degree algorithm by multiple elimination, *ACM Trans. Math. Soft.*, **11, 1985,** pp. 141-153.

[11] DAVIS T.A., AMESTOY P., DUFF I.S., An approximate minimum degree ordering algorithm, *Technical Report TR-94-039*, University of Florida, Computer and Information Science Department, **1994.**

[12] GOTO K., VAN DE GEIJN R., On reducing TLB misses in matrix multiplication, *FLAME working note 9. Technical Report TR 2002-55*, The University of Texas at Austin, Department of Computer Science, **November 2002.**

[13] WHALEY R. C., PETITET A., DONGARA J. J., Automated empirical optimization of software and the ATLAS project, *Technical Report LAPACK working note 147*, Computer Science Department, The University of Tenessee, **September 2000.**