Sur la résolution de systèmes linéaires issus de la méthode des éléments finis par une machine "multiprocesseur"

A. Lichnewsky

IRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France

Tél. 954 90 20

Rapports de Recherche

N° 119

SUR LA RÉSOLUTION DE SYSTÈMES LINÉAIRES ISSUS DE LA MÉTHODE DES ÉLÉMENTS FINIS PAR UNE MACHINE "MULTIPROCESSEUR"

Alain LICHNEWSKY

Février 1982

INRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. 954 90 20

Rapports de Recherche

N° 119

SUR LA RÉSOLUTION DE SYSTÈMES LINÉAIRES ISSUS DE LA MÉTHODE DES ÉLÉMENTS FINIS PAR UNE MACHINE "MULTIPROCESSEUR"

Alain LICHNEWSKY

Février 1982

SUR LA RESOLUTION DE SYSTEMES LINEAIRES ISSUS DE LA METHODE DES ELEMENTS FINIS PAR UNE MACHINE "MULTIPROCESSEUR"

Alain LICHNEWSKY

+++++

Abstract :

The Conjugate Gradient Method, with incomplete factoring preconditionning is well suited for the solution of large linear systems with ill-conditionned matrices. (Arising, for instance, in the finite element discretization in a general domain of R^2 or R^3). However, this method needs to be adapted to run efficiently on a parallel computer. It is shown that the parallelism occuring in this method can be increased without altering the rate of convergence. The speed-up thus obtained is evaluated for several types of domains of finite element discretization.

++++++++

Résumé :

La méthode du Gradient Conjugué Préconditionnée par factorisation incomplète a de très bonnes performances pour la résolution de systèmes linéaires de grande taille associés à des matrices mal conditionnées. (Issues, par exemple, de discrétisation par la méthode des éléments finis dans un domaine quelconque).
Cette méthode ne se prête pas de manière évidente à l'utilisation de Calculateurs parallèles. On montre qu'il est possible d'accroitre le taux de parallélisme inhérent à la méthode sans modifier sensiblement sa vitesse de convergence. On évalue l'accélération ainsi obtenue pour divers types de domaines discrétisées par éléments finis.

INTRODUCTION

Le marché des calculateurs scientifiques adaptés à la résolution de problèmes issus, par exemple, de l'approximation de solutions d'équations aux dérivées partielles, semble devoir être dominé par des machines capables d'un taux de parallélisme élevé. Ceci implique que l'efficacité des programmes exécutés par ces machines sera de plus en plus liée à leur possibilité de tirer parti de ce parallélisme existant au niveau matériel [A][B][C].

La résolution de systèmes linéaires apparait comme une phase cruciale de la mise en oeuvre de nombreux schémas numériques visant à approcher la solution de problèmes stationnaires ou même d'évolution.
Les systèmes linéaires

$$(1) \qquad Ax = b \qquad x,b \in R^N \; ; \; A \in M_N(\mathbb{R})$$

issus de la méthode des éléments finis comprennent en général, une matrice A très creuse et dont la structure n'est pas régulière. A priori le calcul d'une composante de x nécessite la connaissance de toutes les composantes de b ce qui implique une forte activité du réseau d'interconnexion des processeurs. L'étude de (1) lorsque la matrice A possède une structure particulière a fourni plusieurs méthodes itératives ou directes permettant une excellente utilisation du parallélisme inhérent au (multi)-processeur [E] , [F], [G], [H].

Le but de ce travail est de montrer qu'il est possible de parvenir à un taux de parallélisme satisfaisant en combinant la méthode des "dissections emboitées" [I] , [J] et une méthode itérative du type "Gradient Conjugué avec

préconditionnement" [K],[L].
On obtient ainsi une méthode applicable à un système (1) général. Il n'est pas évident que l'on ait conservé les propriétés de convergence rapide de la méthode du Gradient Conjugué. Nous montrons en effectuant des comparaisons, sur des cas représentatifs mais simples, que le nombre d'itérations nécessaires n'augmente pas de façon trop sensible.

Afin de permettre d'apprécier l'efficacité pratique de la méthode nous donnons des évaluations de l'accélération obtenue pour l'éxécution d'une itération de l'algorithme sur 2^{ν} processeurs. Ceci montre que la méthode du Gradient Conjugué préconditionné par la décomposition incomplète de Choleski fournit des accélérations intéressantes dans un contexte multiprocesseur. Toutefois, ce bon rendement n'est obtenu que lorsque certaines relations existent entre le nombre de processeurs, le nombre d'inconnues et les caractéristiques géométriques du domaine discrétisé par éléments finis.

En conclusion la méthode que nous proposons apparait efficace dans des situations réalistes pour un nombre de processeurs de l'ordre de 4 à 16 pour un nombre d'inconnues de l'ordre de 10^4 à 10^5 ou plus.

Le fait d'avoir à travailler sur des systèmes (1) mal conditionnés parait augmenter la compétitivité de l'algorithme face à des méthodes itératives se prêtant mieux au parallélisme (méthodes de relaxation éventuellement dans leur version "chaotique" [X])[K, V].
Dans une situation à très grand nombre de processeurs, nos estimations ne sont pas favorables et la comparaison des méthodes envisageables nous semble un sujet très "ouvert", posant aussi bien des problèmes numériques que combinatoires. ∎

I. STRUCTURE DE SYSTEMES MATRICIELS ISSUS DE LA METHODE DES ELEMENTS FINIS

Les méthodes d'éléments finis fournissent des matrices extrémement creuses dont la structure est cependant intimement liée à la géométrie de la discrétisation. Un outil d'analyse très puissant est d'associer à la matrice A un graphe G traduisant la position des éléments non nuls. Il est ainsi possible de montrer que la complexité des algorithmes de résolution directe de

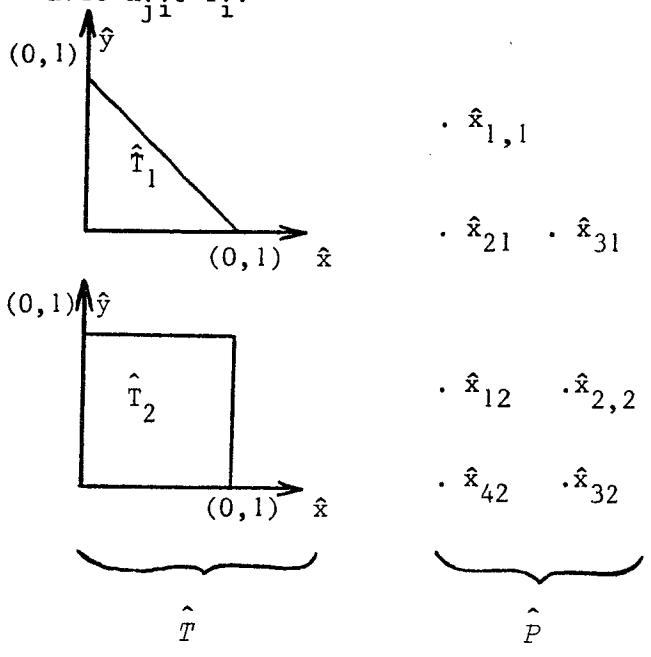
(2) $\quad Ax = b$

est liée à la taille des séparateurs S de G. (cf. [J]). Nous allons utiliser ici ces propriétés de séparation pour accroitre les possibilités de parallélisme dans les algorithmes. Les performances des algorithmes que nous proposons sont liées à l'existence des "petits" séparateurs, leur nombre de sommets $|S|$ influençant le volume des données à transmettre entre les processeurs ainsi que la "taille" des portions de programme qui ne permettent pas un parallélisme optimal.

I.1. Rappels sur les éléments finis

Ce paragraphe est un bref rappel des notions et des résultats dont nous aurons besoin par la suite, que nous avons regroupé pour la commodité du lecteur. On trouvera une exposition plus complète dans [J] , [M] ,[O].

Afin de définir une discrétisation de type éléments finis, nous nous donnons tout d'abord une famille $\hat{T} = \bigcup \hat{T}_i$ d'éléments de référence (polygones simples dans R^2, polyèdres dans le cas de R^3). A chaque élément $\hat{T}_i$ correspond d'une part un espace $\hat{V}_i$ de dimension finie de fonctions $\hat{\phi}_{j,i}$ de $\hat{T}_i$ dans $\mathbb{R}$ et d'autre part un ensemble fini de points particuliers $\hat{P}_i = \bigcup_j (\hat{u}_{ji})$ avec $\hat{x}_{ji} \in \hat{T}_i$.



Exemple 1

Afin de paramétrer $\hat{V}_i$ nous nous donnons un ensemble de couples formés d'un point dans $\hat{P}_i$ et d'un multi-indice de dérivation : $\hat{D}_i = \bigcup_{k=1}^{K} (\hat{x}_k, \alpha_k)$ (les degrés de liberté).
Nous faisons l'hypothèse que l'application linéaire :

$$(3) \qquad \hat{V}_i \to \mathbb{R}^K : \hat{\phi} \to \left(\left(\frac{\partial^{K^1}}{\partial x_{\alpha_k}} \phi \right)(\hat{x}_k) \right)_{k=1}^{K}$$

est une bijection : les degrés de liberté fournissent un paramétrage de $\hat{V}_i$.

Nous considérons à présent un ouvert Ω de $\mathbb{R}^2$ ou $\mathbb{R}^3$ que nous supposons décomposé en éléments T_i : $\bar{\Omega} = \bigcup T_i$. Chaque T_i est l'image par une bijection O_i de l'un des éléments de référence $\hat{T}_{\hat{i}}$; de plus cette décomposition obéit aux règles suivantes :

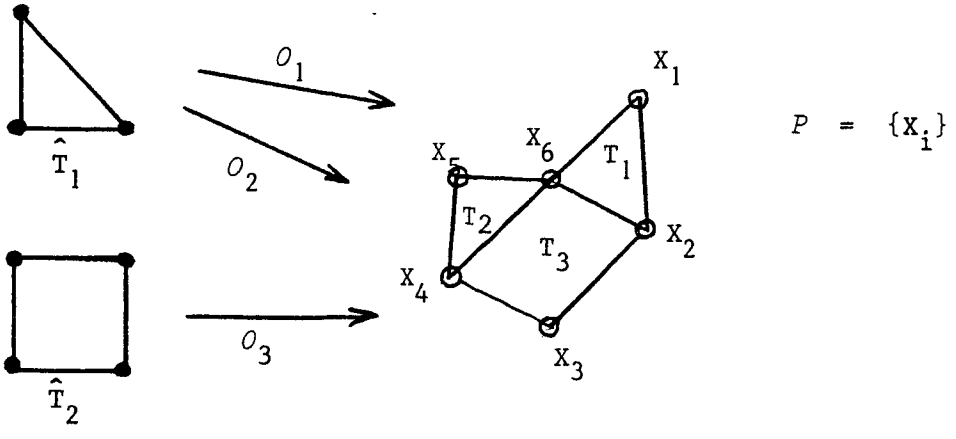
$$(4) \qquad \text{si } i \neq j : \quad T_i \cap T_j = \begin{cases} \emptyset \\ \\ \text{l'image par } O_i \text{ d'un sommet, d'un côté ou} \\ \text{d'une face de } \hat{T}_{\hat{i}} \end{cases}$$

L'espace V des approximations est un espace de fonctions de $\bar{\Omega}$ dans $\mathbb{R}$ tel que si $\phi \in V$ alors :

$$(5) \qquad \text{. pour tout } T_i \in T : \quad \phi_{|T_i} \circ O_i = \hat{\phi} \in \hat{V}_i \ .$$

Afin de paramétrer V nous utilisons ici encore des degrés de liberté portés par une famille P de points particuliers, images des points de $\hat{P}$:

$$(6) \qquad \text{si } x \in P \text{ et } x \in T_i \text{ alors } \hat{x} = O_i^{-1}(X) \in \hat{P}_1$$



Exemple 2

Soit $= \bigcup_i D_i = \bigcup_i (x_i, \ _i)$ l'ensemble des degrés de liberté, l'application linéaire

$$(7) \qquad V \xrightarrow{\Psi} \mathbb{R}^{|D|} \ : \ \phi \ \rightarrow \ ((\frac{\partial^{|\alpha_i|}}{\partial x_{\alpha_i}} \phi)(X_i))$$

fournit un paramétrage de V. Nous utiliserons comme base privilégiée de V l'image B par Ψ^{-1} de la base canonique de $\mathbb{R}^{|D|}$: un élément de B sera dit "fonction de base". De plus cette construction fait correspondre une fonction de base ϕ_i à chaque degré de liberté D_i.

La condition que nous imposons à présent est à l'origine de la structure creuse des matrices que nous allons étudier.

Soit ϕ_i une fonction de base correspondant au degré de liberté $D_i = (X_i, \alpha_i)$ alors

$$(8) \qquad \text{support } (\phi_i) \ \subset \ \{\bigcup T_k \mid \ x_i \in \ T_k \ \}$$

Nous allons maintenant considérer des systèmes linéaires de $N = |D|$ équations à N inconnues

$$(9) \qquad Ax = b$$

où la matrice A vérifie les conditions

$$\left\{ \begin{array}{l} \underline{\text{A est symétrique, définie positive}} \\ A = (a_{ij}) \ \underline{\text{et si}} \ a_{ij} \neq 0, \ \underline{\text{alors il existe un élément}} \ T_k \in T \\ \qquad \underline{\text{tel que}} \ X_i \in T_k \ \underline{\text{et}} \ X_j \in T_k \quad (1) \end{array} \right.$$

Nous avons ainsi fait le lien entre le domaine Ω , sa triangulation et la structure de la matrice obtenue en mettant en oeuvre la méthode des éléments finis.

Par la suite, nous considérerons comme a priori non nuls tous les coefficients a_{ij} satisfaisant la condition (10-2), bien que des simplifications supplémentaires soient possibles dans des cas particuliers.

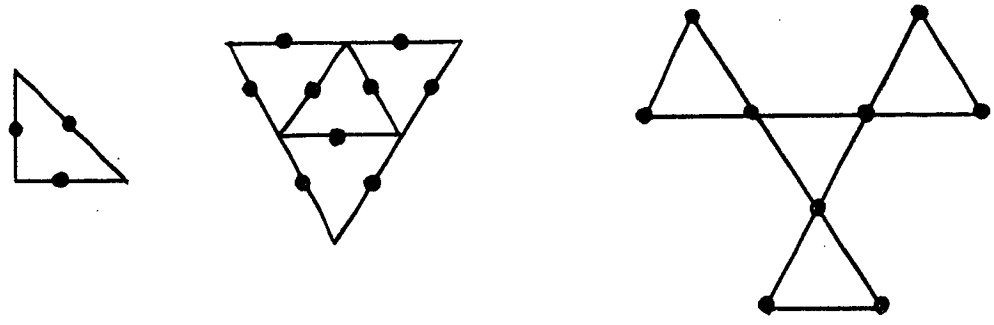
---

(1) X_i (resp. X_j) supportant le degré de liberté D_i (resp. D_j).

I.2. Utilisation de graphes pour représenter les matrices creuses, relations entre parallélisme de l'algorithme et séparateurs du graphe.

A la matrice A intervenant en (9) nous associons un graphe $G(A) = (V,E)$; un sommet $v_i \in V$ correspondant à chaque degré de liberté $D_i \in D$. Les arêtes de G(A) repèrent les éléments non nuls de A :

(11) $\qquad e = (v_i, v_j) \in G(A) \leftrightarrow a_{ij} \neq 0$



$\hat{T}$ $\qquad$ T $\qquad$ G(A)

Exemple 3

Les propriétés du graphe G(A) sont directement liées au coût des calculs nécessaires pour résoudre (2) par des méthodes directes, d'élimination [O][M][J]. Nous allons montrer maintenant que la propriété d'admettre de petits séparateurs pour G(A) autorise la résolution du système (2) en parallèle.

De façon plus précise nous effectuons une renumérotation des inconnues de (2) associée à la permutation P de (1,,N), ce qui transforme la matrice A en la matrice $A' = PA\,P^T$. ([1])

Nous notons Π la bijection de $V \to (1,,N)$ qui associe à chaque sommet v_i de G l'étiquette P(i).

On a alors

(12) $\qquad a'_{ij} \neq 0 \leftrightarrow \{v_{\Pi^{-1}(i)}, v_{\Pi^{-1}(j)}\} \in G(A)$

([1]) Nous utilisons la même notation pour la permutation et pour la matrice correspondant à la permutation des vecteurs de la base canonique de R^N.

Soit en repérant les sommets par leurs étiquettes notées comme indice supérieur :

$$(13) \qquad a'_{ij} \neq 0 \quad \leftrightarrow \quad \{v^i, v^j\} \quad \in \quad G(A)$$

Supposons que l'on puisse trouver une permutation P et une décomposition de $R^N = R^{n_1} \times R^{n_2} \times R^{n_3}$ tels que la matrice A' admette la représentation par blocs

$$(14) \qquad A' = \begin{pmatrix} A'_{11} & 0 & A'_{13} \\ 0 & A'_{22} & A'_{23} \\ A'_{31} & A'_{32} & A'_{33} \end{pmatrix} \qquad \begin{pmatrix} A'_{11} & 0 & A'^{T}_{31} \\ 0 & A'_{12} & A'^{T}_{32} \\ A'_{31} & A'_{32} & A'_{33} \end{pmatrix}$$

La matrice A' est aussi définie positive et symétrique si bien qu'elle admet une décomposition de Choleski $A' = LL^T$ où L est une matrice triangulaire inférieure :

$$(15) \qquad L = \begin{pmatrix} L_{11} & 0 & 0 \\ 0 & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix}$$

Etant donné que nous utiliserons par la suite des décompositions "incomplètes" il n'y a pas lieu de se préoccuper du remplissage de la matrice L, qui est lié à la permutation P.

La forme (15) nous permet de conduire les calculs de la résolution de $A'x = b$ en parallèle :

On résoud successivement : $Ly = b$ puis $L^T x = y$ suivant l'algorithme :

proc DIRECT

for each i = 1 to 2 pardo

$Y_i = L_{ii}^{-1} \; b_i \; ; \quad s_i = L_{3i} \; Y_i$

odpar ;

$t = b_3 - s_1 - s_2$;

$Y_3 = L_{33}^{-1} \; t \; ; \quad X_3 = (L_{33}^T)^{-1} \; Y_3$

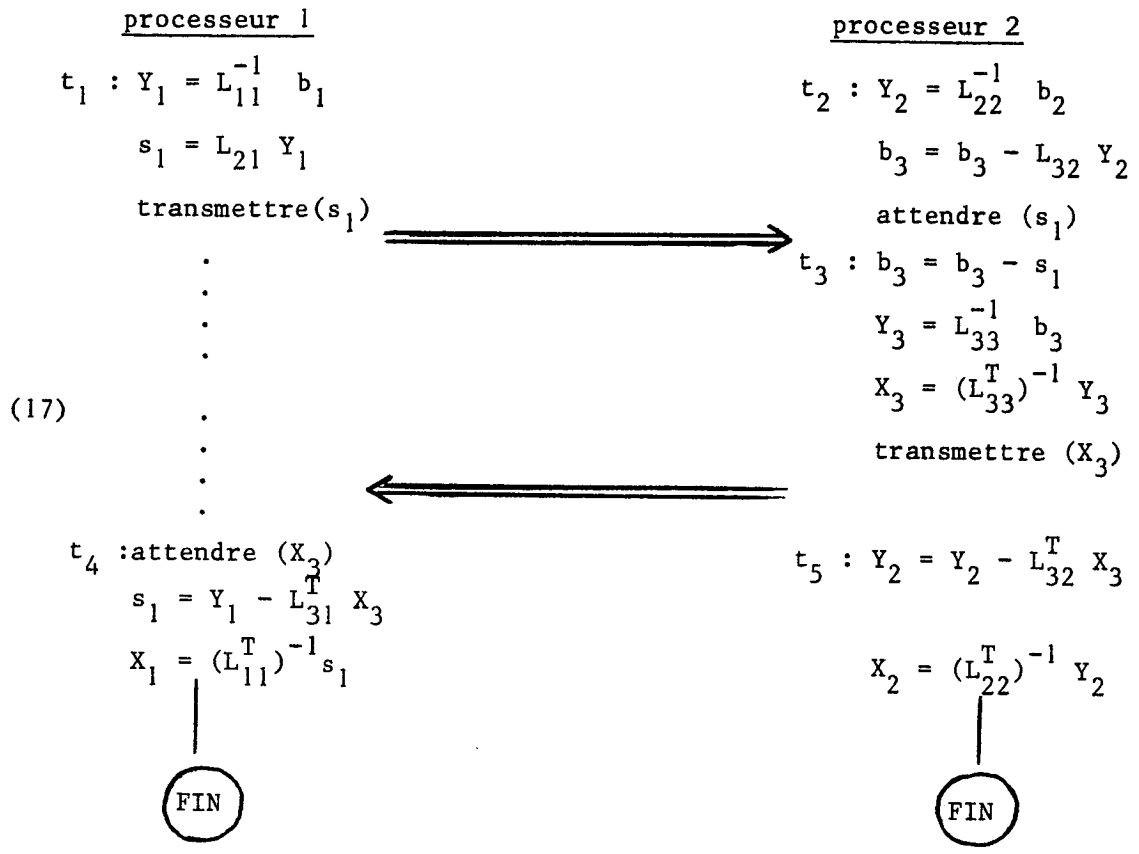
for each i = 1 to 2 pardo

$s_i = Y_i - L_{3i}^T X_3$; $X_i = (L_{ii}^T)^{-1} s_i$

odpar ;

corp DIRECT

Dans le cas biprocesseur on aboutit donc au schéma :



Il apparait sur cette décomposition de l'algorithme que les principales pertes d'efficacité sont dues :

(a) à la transmission de vecteurs s_1 et X_3 de dimension n_3 ,

(b) à l'attente du processeur 1 durant t_3.

Pour les minimiser, il y a lieu de prendre n_3 aussi petit que possible ; pour minimiser les attentes, il convient de s'assurer que les temps d'exécution de t_1 et t_2 (resp. t_4 et t_5) soient voisins. On aboutit ainsi aux contraintes :

$$(18)\quad \begin{cases} N = n_1 + n_2 + n_3 & n_1 >> n_3 \quad n_2 >> n_3 \\ n_1 \simeq n_2 \end{cases}$$

Cette renumérotation donnant la structure particulière (14) admet une interprétation en termes de graphes.

Notons :

$$V_1 = \Pi^{-1}(\{1,,n_1\})$$
$$V_2 = \Pi^{-1}(\{n_1+1,\ldots,n_1+n_2\})$$
$$V_3 = \Pi^{-1}(\{n_1+n_2+1,\ldots,N\})$$

et soient G_i les sous-graphes de G induits par les ensembles de sommets V_i (i=1,,3). Le fait que les blocs A'_{21} et A'_{12} soient nuls équivaut à ce qu'aucun sommet de V_1 n'est adjacent à un sommet de V_2. Ceci nous amène à introduire les définitions suivantes.

Définitions :

Soit G = (V,E) un graphe ; l'ensemble $S \subset V$ sépare G en V_1 et V_2.

(1) V admet une partition en V_1, V_2, S

(2) aucun sommet dans V_1 n'est adjacent à un sommet dans V_2.

Dans les évaluations que nous ferons par la suite, il sera nécessaire de pouvoir séparer récursivement les sous-graphes ainsi obtenus, ceci nous conduit à introduire la définition suivante :

Définition :

Une classe S de graphes a la propriété de f(n)-séparation si

(i) Si G_2 est un sous-graphe de $G_1 \in S$ alors $G_2 \in S$

(ii) Il existe des constantes $\alpha \in [1/2,1[$, $\beta > 0$ telles que si $G = (V,E) \in S$ et $|V| = n$ alors il admet une séparation en V_1 et V_2 par S avec

$$(19)\quad |V_1| \leq \alpha n , \quad |V_2| \leq \alpha n , \quad |S| \leq \beta f(n)$$

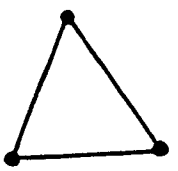
En précisant les inégalités (18) cette définition va nous permettre l'évaluation d'algorithmes basés sur l'utilisation répétée de la séparation (cf. (16), (17)).
Dans les cas pratiques, il sera en général nécessaire de considérer les constantes exactes intervenant en (18), pour un séparateur S "bien choisi".

I.3. Exemples de résultats de séparation applicable avec les éléments finis.

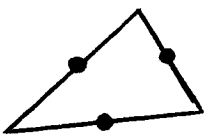
a) le cas d'un maillage bidimensionnel

(a-1) éléments tels que le graphe G(A) soit plan

par exemple : éléments "P_1" et "P_1-non-conforme"



P_1



"P_1-non-conforme"

Exemple 4

Pour des données au bord de type Dirichlet ou Neumann on obtient un graphe G(A) plan et le résultat suivant est applicable (cf.[M][J][S]) ([1])

Théorème 1 (R.J. LIPTON et R.E. TARJAN[S])

Supposons que G(A) = (V,E) soit un graphe plan. Alors G peut être séparé en V_1 et V_2 par S avec :

$$|V_1| \ , \ |V_2| \leq 1/2 \ |V|$$

$$|S| \leq (2\sqrt{2}/(1-\sqrt{2}/3))\sqrt{|V|}$$

(a-2) cas d'éléments finis plans "généraux"

Supposons que la "triangulation" T soit ainsi faite que chaque sommet soit support d'un degré de liberté au plus et que les conditions aux limites soient du type Dirichlet ou Neumann.

Il est alors possible de se ramener à un graphe $\tilde{G}(\tilde{A})$ ayant la

([1]) Il convient de se méfier de l'influence de conditions aux limites de type "transmission, périodicité..." qui peuvent modifier la topologie de G(A). Le choix d'un séparateur est quand même possible mais le résultat général ne s'applique pas toujours.

propriété suivante : $\tilde{G}$ peut être obtenu à partir d'un graphe plan $\tilde{\tilde{G}}$ en ajoutant certaines diagonales de chaque face.

Preuve :

1) Construction de $\tilde{G}$ et $\tilde{\tilde{G}}$.

Considérons tout d'abord le cas des $\hat{x}_{ji} \in \partial\hat{T}_i$ qui supportent plusieurs degrés de libertés : $\bigcup_k (\hat{x}_{ji}, \alpha_k)$. De tels points sont, par hypothèse, sur un côté de $\hat{T}_i$ et il est possible de leur associer une famille de points voisins $\hat{x}'_{jik} \in \partial\hat{T}_i$ chaque point correspondant à un degré de liberté unique : $(\hat{x}_{ji}, \alpha_k) \leftrightarrow \hat{x}_{jik}$.

Il est aussi possible d'effectuer cette décomposition de manière cohérente sur l'ensemble du domaine $\bar{\Omega} = \bigcup T_i$.

Soit $X \in T_i \cap T_j$ un point supportant les degrés de liberté $\bigcup_K (X, \alpha_K)$, alors on a

$$(20) \quad \begin{cases} T_i = O_i(\hat{T}_{\hat{i}}) \quad ; \quad T_j = O_j(\hat{T}_{\hat{j}}) \quad ; \\ X = O_i(\hat{X}_i) = O_j(\hat{X}_j) \end{cases}$$

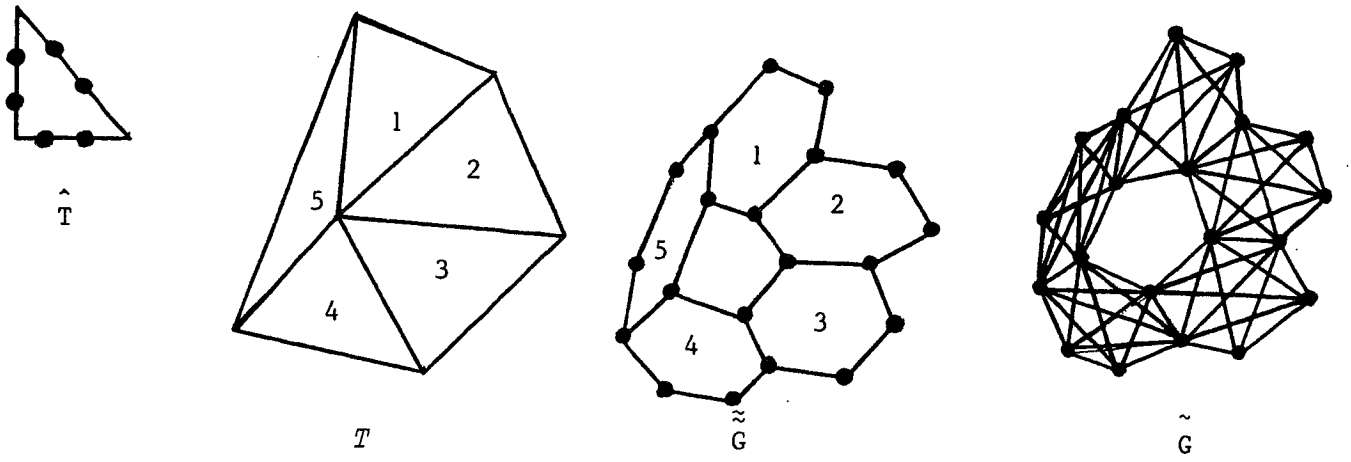
$\hat{X}_i$ et $\hat{X}_j$ sont support des degrés de libertés $\bigcup_k (\hat{X}_i, \beta_k)$ et $\bigcup_k (\hat{X}_j, \gamma_K)$. Par décomposition cohérente nous entendons que

$$(21) \qquad X_K = O_i(\hat{X}_{ik}) = O_j(\hat{X}_{jk})$$

Ceci étant, nous associons à chaque élément de référence $\hat{T}_i$ le polygone convexe $\hat{T}'_i$ obtenu en joignant les $\hat{X}_{ji} \in \partial\hat{T}_i$ et supportant des degrés de libertés simples ainsi que les $\hat{X}'_{jik}$. Le graphe plan $\tilde{\tilde{G}}$ est obtenu en prenant :

$$(22) \qquad \tilde{\tilde{G}} = \bigcup_i O_i(\hat{T}'_i)$$

Il est à noter que $\tilde{G}$ peut avoir plus de faces que T ne contient d'éléments ainsi que le montre la figure suivante :



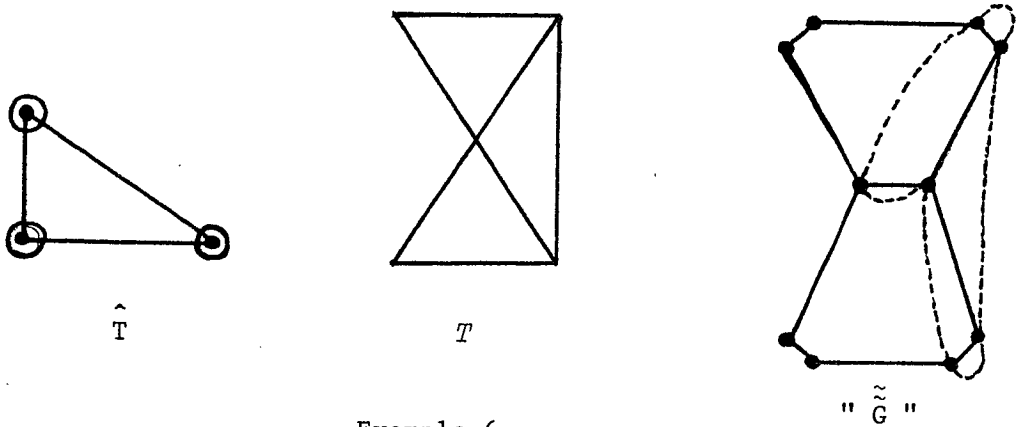
$\hat{T}$ $\quad T$ $\quad \tilde{\tilde{G}}$ $\quad \tilde{G}$

Exemple 5

Le graphe $\tilde{G}$ est obtenu à partir de $\tilde{\tilde{G}}$ en le complétant par toutes les diagonales des faces $O_i(\hat{T}_i')$.

Remarque :

L'hypothèse faite sur les sommets de T a pour but d'éviter que $\tilde{\tilde{G}}$ ne soit non-plan comme dans la situation suivante :



$\hat{T}$ $\quad T$ $\quad$ " $\tilde{\tilde{G}}$ "

Exemple 6

2) La matrice $\tilde{A}$.

Considérons d'abord le cas où tous les degrés de libertés sont supportés par le bord des éléments. On prend alors $\tilde{A} = A$, les relations (8) et (10) assurant que $G(A) \leq \tilde{G}$. Les différences entre $G(A)$ et $\tilde{G}$ sont dues au fait que certains termes de la matrice A sont nuls, le fait de travailler avec $\tilde{G}$ consiste simplement à ne pas chercher à tirer parti de ce fait et à stocker ces coefficients nuls.

Considérons maintenant le cas où certains degrés de liberté sont supportés par des points à l'intérieur des éléments. En utilisant le fait que la matrice A est symétrique et définie positive il est possible d'éliminer ces inconnues par la méthode de Gauss, on obtient ainsi une matrice $\tilde{A}$ symétrique et définie positive. Si nous comparons $\tilde{A}$ et la matrice A' obtenue à partir de A en supprimant les lignes et les colonnes correspondant aux inconnues éliminées, les seules différences touchent les termes a'_{ij} correspondant à des arêtes $(v_i, v_j) \in G(\tilde{A})$ avec :

(23) v_i et v_j appartiennent à un même élément $O_K(\hat{T}_1)$

Ceci achève la démonstration avec (10) et (8).

Remarque :

Il est à noter que la transformation faisant passer de A à $\tilde{A}$ conserve d'autres propriétés comme celle d'être une M-matrice.

□

L'intérêt de la construction de $\tilde{A}$ est qu'il est possible de lui appliquer le résultat de séparation ci-après ; en outre $\tilde{A}$ est symétrique définie positive, comme A ce qui sera important pour la suite.

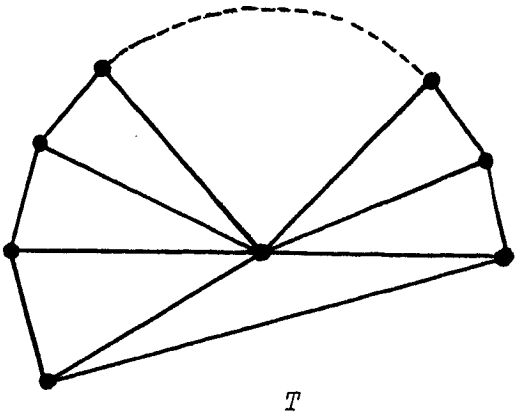
Théorème 2 (R.J. LIPTON, R.E. TARJAN S)

Soit $\tilde{G}$ un graphe obtenu en ajoutant au graphe plan $\tilde{\tilde{G}}$ certaines diagonales de faces. Supposons que les faces de $\tilde{\tilde{G}}$ ayant fait l'objet de ces modifications aient au plus k sommets chacune.
Alors $\tilde{G}$ peut être séparé en V_1 et V_2 par S avec

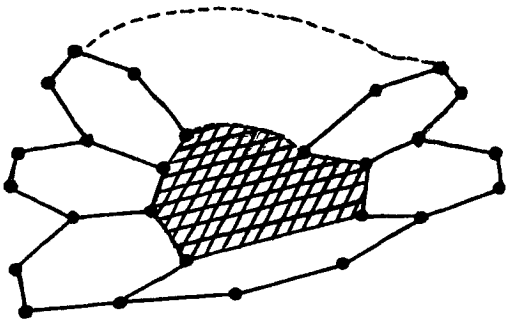
$$|V_1|, |V_2| \leq 2/3 |G|$$

$$|S| \leq 4 [k/2] \sqrt{|G|}$$

Preuve : La démonstration est essentiellement celle de [S] . La définition de $\tilde{G}$ et $\tilde{\tilde{G}}$ a été légèrement modifiée (sans conséquences pour la preuve) afin de d'appliquer à des variantes de l'exemple 5 ci-après, dans lesquelles il peut s'introduire des faces ne correspondant à aucun élément fini et comportant un nombre arbitraire de sommets :



T



$\tilde{\tilde{G}}$

■

II.APPLICATION DES OUTILS DE SEPARATION A LA METHODE DU GRADIENT CONJUGUE AVEC PRECONDITIONNEMENT.

II.1.Description de la méthode du Gradient Conjugué.

Soit A une matrice symétrique, définie positive ; afin de résoudre le système (9) nous appliquons la méthode du Gradient Conjugué au système

$$(2.1) \qquad (L^{-1}A(L^T)^{-1})\, L^T X = L^{-1}b$$

où $M = LL^T$ est inversible. On aboutit ainsi au système linéaire

$$(2.2) \qquad B\,\tilde{\tilde{x}} = \tilde{b}$$

et la matrice $B = L^{-1}A(L^T)^{-1}$ est symétrique définie positive. L'intérêt du préconditionnement, c'est-à-dire de l'introduction de (2.2) est d'obtenir une vitesse de convergence satisfaisante dans le cas de systèmes de grande taille.

L'algorithme du Gradient Conjugué s'écrit maintenant :

Gradient Conjugué :

$u^o = \ldots$;
$\tilde{b} = L^{-1}b$;
$r^o = L^{-1}A(L^T)^{-1}\, u^o - \tilde{b}$;
$d^o = -r^o$; $\alpha^o = \langle r^o, r^o \rangle$;
$n = 0$;

Boucle : $n = n+1$; /* nombre d'itérations */

$\lambda^n = \alpha^{n-1} / \langle d^{n-1}, L^{-1}A(L^T)^{-1} d^{n-1}\rangle$;

$u^n = u^{n-1} + \lambda^n d^{n-1}$;

$r^n = L^{-1}A(L^T)^{-1} u^n - \tilde{b}$,

$\beta^n = \langle r^n, r^n \rangle$;

Si $\beta^n \leq \varepsilon$ aller à Fin ;

$\gamma^n = \beta^n / \alpha^{n-1}$; $\alpha^n = \beta^n$;

$d^n = - r^n + \gamma^n d^{n-1}$

aller a Boucle ;

Fin $x = L^{-T} u^n$

On le met sous une forme plus adaptée au calcul en posant $\tilde{u} = (L^T)^{-1}u$, $\tilde{s} = Lr$, $\tilde{r} = (L^T)^{-1}r$ et $\tilde{d} = (L^T)^{-1} d$. On obtient alors :

Gradient Conjugué-préconditionné :

$u^o = \ldots$

$s^o = A u^o - b$; $r^o = (L^{-T} L^{-1})s^o$;

$d^o = -r^o$;

$\alpha^o = \langle s^o, r^o \rangle$; $n = 0$;

Boucle : $n = n+1$ /* nombre d'itérations */

$\lambda^n = \alpha^{n-1} / \langle d^{n-1}, A d^{n-1} \rangle$;

$u^n = u^{n-1} + \lambda^n d^{n-1}$;

$s^n = s^{n-1} + \lambda^n A d^{n-1}$;

$r^n = (L^{-T}L^{-1}) s^n$

$\beta^n = \langle r^n, s^n \rangle$; $\gamma^n = \beta^n / \alpha^{n-1}$; $\alpha^n = \beta^n$;

si $\beta^n \leq \varepsilon$ aller à Fin ;

$d^n = -r^n + \gamma^n d^{n-1}$;

aller à Boucle ;

Fin : $x = u^n$;

La nature des opérations vectorielles et matricielles s'explicite enfin dans la description suivante de l'algorithme :

Gradient conjugué-préconditionné :

vecteurs : u, s, r, d ;

matrices-creuses : A ;

matrices-creuses triangulaires : L ;

$u = u_o$; $s = Au - b$; $r = (L^T)^{-1} L^{-1} s$;

$d = -r$; $= \langle s,r \rangle$;

Boucle : $r = Ad$;

$\lambda = \alpha / \langle d,r \rangle$;

(2.3) $u = u + \lambda d$; $s = s + \lambda r$;

$r = (L^T)^{-1} L^{-1} s$;

$\beta = \langle r,s \rangle$;

si $\beta \leq \varepsilon$ aller à Fin ;

$\gamma = \beta / \alpha$; $\alpha = \beta$;

$d = -r + \gamma d$;

aller à Boucle ;

Fin : $X = u$

Ce qui fait apparaitre que par itération il est nécessaire d'effectuer :

(2.4)
- Produits matrice vecteur : 1
- Résolutions de systèmes triangulaires : 2
- Produits scalaires de vecteurs : 2
- Additions vectorielles : 3

L'utilisation d'un préconditionnement impose donc la résolution supplémentaire de 2 systèmes linéaires triangulaires.
Dans le cas de l'utilisation de calculateurs séquentiels, le coût de ces résolutions peut être considéré comme proportionnel au nombre de coefficients non nuls de L.
Ceci ne suffit pas dans le cas de l'utilisation d'un calculateur parallèle, nous allons utiliser les méthodes de partitionnement exposées au §.I.2 pour résoudre les difficultés soulevées par la résolution de ces systèmes. Il est à noter que les autres opérations mentionnées en (2.4) peuvent être effectuées de manière efficace par un calculateur parallèle (cf. [T]).

II.2. Choix du préconditionnement - Méthode de la décomposition incomplète de Choleski.

L'objet du préconditionnement est d'obtenir une vitesse de convergence suffisante pour l'algorithme (2.3) ; cette vitesse de convergence est directement liée au spectre de la matrice $M^{-1}A$. Parmi les très nombreuses méthodes possibles (cf. [R]), la décomposition incomplète de Choleski et ses variantes que nous ne décrivons pas ici (cf. [L]) se révèle particulièrement efficaces.

Soit $G(A) = (V_A, E_A)$ le graphe associé à la matrice A, nous nous donnons a priori le graphe $G_I = (V_A, E_I)$ associé à la décomposition incomplète. On cherche une décomposition de la forme :

$$(2.4)\quad \begin{cases} 1)\ G_I \text{ est le graphe associé à la matrice} \\ \quad L + L^T \ :\ L_{ij} \neq 0 \ \rightarrow (v_i, v_j) \in E_I \\ \\ 2)\ A = L\,L^T + R \quad \text{où} \\ \quad R_{ij} \neq 0 \ \rightarrow \ (v_i, v_j) \notin E_I \qquad (1) \end{cases}$$

où la matrice L est triangulaire inférieure (2)

La matrice R indique l'erreur que l'on commet en négligeant a priori certains termes de la décomposition usuelle de Choleski. Lorsqu' on choisit pour graphe G_I le graphe de remplissage pour l'élimination des inconnues dans l'ordre naturel G^* (cf. [J]), R = 0 et on effectue la décomposition complète de Choleski.

En pratique, il n'est pas nécessaire de calculer la matrice R, qui peut faire l'objet d'un remplissage important : il suffit d'identifier un par un les coefficients de L à l'aide des équations

(1) On rappelle que les arêtes de G_I ne sont pas orientées.

(2) Pour une numérotation des inconnues fixée donnée par l'application Π (§1.2).

(2.5) $A_{ij} = (L\,L^T)_{ij}$ pour $(v_i,v_j) \in E_I$ où $i = j$

Contrairement à la décomposition de Choleski complète (usuelle) d'une matrice symétrique définie positive, il n'est pas évident que la décomposition (2.4) soit possible. Diverses conditions ont été établies qui assurent de cette possibilité, une opération de translation consistant à décomposer la matrice A + αI au lieu de A (pour une constante α bien choisie) permet de s'y ramener. cf. [L])(Il est à noter que seule importe, en définitive, la vitesse de convergence de (2.3)). Ceci est illustré par les résultats suivants :

Théorème 3 :

Si A est une matrice à diagonale dominante et définie positive, alors la décomposition incomplète de Choleski est possible (1) pour tout graphe G_I.

Théorème 4 : (cf. [K])

Si A est une M-matrice symétrique la décomposition incomplète de Choleski est possible pour tout graphe G_I.

Remarquons aussi que les hypothèses des Théorèmes 3 et 4 ne sont pas invalidées par une permutation de la base de R^N transformant A en PAP^T.

II.3. Utilisation de Méthodes de séparation pour accroitre le parallélisme.

Nous supposons maintenant que le graphe G(A) possède la propriété de f(.)-séparation. (2) En exploitant ce fait nous allons mettre A sous une forme généralisant (14) et permettant la conduite des calculs en parallèle avec 2^ν processeurs ($\nu > 1$).

Comme au § I.2 nous construisons une permutation P de la base canonique de R^N en associant à chaque sommet $v_i \in G(A)$ une étiquette $\Pi(i)$. Pour cela nous allons construire une famille de sous graphes de G(A) induits par les ensembles de sommets V_α (où α sera un multi-indice de longueur

(1) En fait on établit aussi la "stabilité" de cette décomposition : les coefficients diagonaux ne s'approchent pas de 0.

(2) La fonction f(.) n'a pas d'importance pour les considérations développées dans ce paragraphe.

$\leq \nu+1$). En partant de l'ensemble des sommets de G(A) : V_o nous définissons récursivement les V_α comme suit :

(2.6)
(1) si longueur (α) = ν ne plus décomposer V_α
(2) si α(longueur (α)) = 2 alors $V_\alpha = V_{(\alpha,o)}$
(3) sinon soit G_α le sous-graphe de G(A) induit par V_α . Il est possible de le séparer en U et W par S en utilisant la propriété de f(.)-séparation.

Nous posons alors :

(2.6bis)
$$V_{(\alpha,o)} = U$$
$$V_{(\alpha,1)} = W$$
$$V_{(\alpha,2)} = S$$

et nous appliquons récursivement cet algorithme aux V_α, que nous venons de former.

En utilisant sur les multi-indices que nous venons de former la relation d'ordre lexicographique notée " <<", nous imposons à l'application $\Pi : V \rightarrow (1,N)$ les conditions :

$$(2.7) \qquad \alpha << \bar{\alpha} \qquad \rightarrow \qquad \Pi(V_\alpha) < \Pi(V_{\bar{\alpha}})$$

Il est important de remarquer que les relations (2.7) sont compatibles avec l'ordre induit par la méthode de dissections emboitées qui revient simplement à poursuivre (2.6) jusqu'à ce que les V_α soient assez petits, puis à utiliser une numérotation respectant (2.7). Ceci fera que les considérations sur le parallélisme que nous développons sont applicables sans modification aux méthodes directes de type décomposition L U ou de Choleski combinées avec la méthode des dissections emboitées généralisées.

Afin de pouvoir étudier, dans l'esprit de [u] , les transferts d'information entre les processeurs nous associons aussi à chaque sommet $v_i \in G(A)$ un "numéro de processeur" $p(i) \in \{0,\ldots,2^\nu-1\}$.

Détermination du "numéro de processeur"

(2.8)

(a) Soit α le multi-indice de longueur $\nu+1$ tel que $v \in V_\alpha$

(b) $\alpha = (\alpha_\nu, \ldots, \alpha_o)$. On définit $\bar{\alpha} \in (\{0,1\})^{\nu+1}$ tel que $\bar{\alpha}_j = \alpha_j$ [modulo 2] pour $j = 0,\ldots,\nu$

(c) $p(v) = \sum_{j=0}^{\nu} 2^j \alpha_j$

On remarque que $\alpha_\nu = 0$ si bien que $p(v) \in [0, 2^\nu - 1]$; le fait que $\bar{\alpha}$ soit la représentation binaire de $p(v)$ nous sera utile.

Décomposition par blocs de la matrice permutée $A' = PAP^T$

Comme au paragraphe I.2, une représentation par bloc des matrices et des vecteurs fait apparaitre le "découplage" des équations. Nous allons définir des décompositions à plusieurs niveaux de manière à faire apparaitre plus ou moins "finement" la structure de A'.

Décomposition de niveau μ $(1 \leq \mu \leq \nu+1)$

Soit $L_\mu = (\alpha^o, \ldots, \alpha^k)$ la liste des multi-indices apparus en (2.6) et de longueur μ , rangés en ordre croissant. On décompose

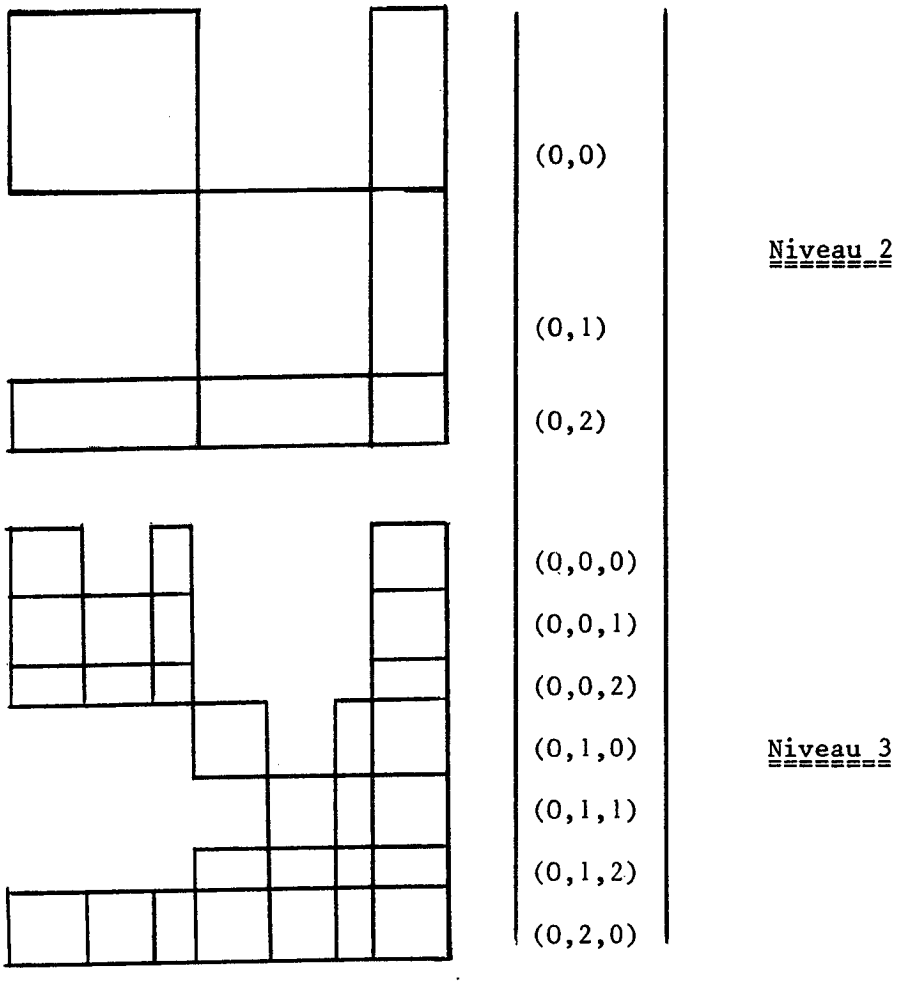
$$\mathbb{R}^N = \mathbb{R}^{|V_{\alpha^o}|} \oplus \ldots \oplus \mathbb{R}^{|V_{\alpha^k}|}$$

et on utilise la décomposition par bloc induite sur les matrices, que l'on désignera par "décomposition par blocs de niveau μ ". On note $X \in R^N$

$$X = (X_{\alpha^o}, \ldots, X_{\alpha^k})$$

les inconnues X_{α^ℓ} correspondant aux degrés de liberté dans V_{α^ℓ}. On indexe de la même façon la décomposition par bloc des matrices.

Décompositions par bloc de la matrice A' de niveau 2 et 3



Exemple 8 multi-indices

Les seuls blocs non nuls de A' sont ceux caractérisés par le

Lemme 1 : Considérons la décomposition par blocs de niveau μ de A'. Soient α et β deux multi-indices de $L\mu$; alors $A'_{\alpha\beta}$ et $A'_{\beta\alpha}$ pourraient être non nuls seulement si l'une des alternatives suivantes est vraie

(i) $\alpha = \beta$

(ii) $\exists$ $1 \leq \quad \leq \mu-1$ tel que

$$\alpha_{(\nu:\nu- \ +1)} = \beta_{(\nu:\ \nu- \ +1)} \text{ et } (\alpha_{\nu-1} = 2 \text{ ou } \beta_{\nu-1} = 2) \quad \square$$

Preuve :

La preuve se fait par récurrence sur les niveaux.
Au niveau 1 il n'y a rien à démontrer ; supposons la propriété vraie jusqu'au niveau μ . Soit $A'_{\alpha\beta}$ un bloc de la décomposition de niveau $\mu+1$ notons $\bar{\alpha} = \alpha_{(\mu:\ 1)}$ et $\bar{\beta} = \beta_{(\mu:1)}$.
Si $\alpha = \beta$ le bloc $A'_{\alpha\beta}$ est situé sur la diagonale et est non nul d'après (10). Si $\alpha \neq \beta$ deux éventualités doivent être considérées :

(a) $\bar{\alpha} = \bar{\beta}$. α et β peuvent avoir les trois valeurs
$\gamma = (\bar{\alpha},0)$; $\delta = (\bar{\alpha},1)$, $\varepsilon = (\bar{\alpha},2)$
En revenant à la construction effectuée en (2.6) on s'aperçoit qu'alors V_ε sépare $V_{\bar{\alpha}}$ en V_γ et V_δ et que les seuls blocs non nuls peuvent être d'après $(12)^{\bar{\alpha}}$:

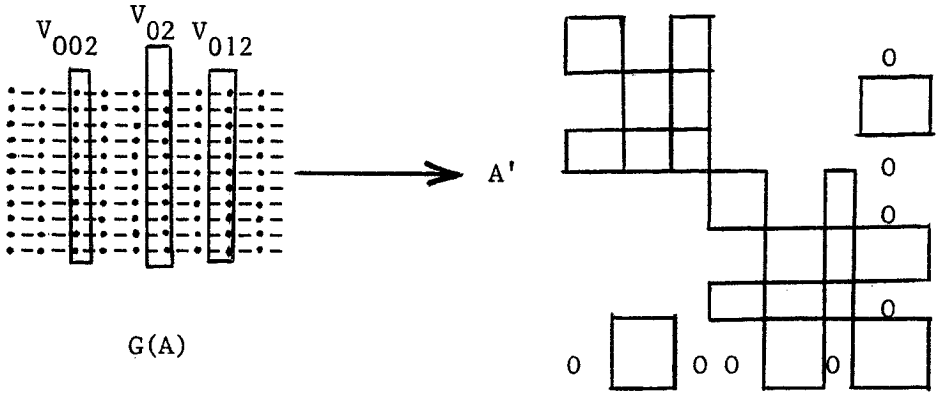
$$A'_{\varepsilon\gamma} \ , \ A'_{\varepsilon\delta} \ , \ A'_{\gamma\varepsilon} \text{ et } A'_{\delta\varepsilon} \ ;$$

dans ce cas la caractérisation est vraie.

(b) $\bar{\alpha} \neq \bar{\beta}$. pour que $A'_{\alpha\beta}$ soit non nul il est nécessaire que $A'_{\bar{\alpha}\bar{\beta}} \neq 0$ et la caractérisation (ii) est vraie par l'hypothèse de récurrence.

Ceci achève la preuve. $\square$

L'exemple suivant montre que suivant l'agencement des séparateurs certains blocs supplémentaires peuvent être nuls.



Exemple 9 : (Les blocs nuls supplémentaires sont notés 0).

II.4. Algorithme de factorisation incomplète

Nous allons factoriser la matrice A' selon la forme décrite en (2.4) pour un graphe G_I quelconque ([1]).
La factorisation est obtenue en utilisant l'algorithme ALG-1 ci-dessous qui exploite les possibilités de parallélisme introduites par la décomposition en dissection emboitées.

Remarque :

Dans la description ALG-1 nous utilisons la procédure "Identifier-IC" qui effectue l'identification d'un bloc de la matrice A' en respectant G_I. Tel que nous l'utilisons dans ALG-1, ce sous-algorithme est réalisable si l'on suppose simplement que A' admet une décomposition incomplète en deux matrices L et L^T inversibles (ce qui n'est pas une restriction).

([1]) Les inconnues étant permutées suivant la permutation Π construite au §.2.3.

```
FACTORISATION INCOMPLETE
   BEGIN
        Identifier-IC : PROC (sous-matrice ; equation) ;
            / * Identification un par un des coefficients d'une
                matrice, méthodes :"LL^T-"Incomplet"" */
        END ;
        FOR NIV =   ν+1  STEP-1   UNTIL  NIV=1
            DO
                IF NIV =  ν+1
                 THEN FOR EACH   α ∈ Bin_{ν+1} {0,...,2^ν-1} ∩ L_{ν+1}
                    PARDO
                     Identifier-IC(L_αα  ; "L_αα L^T_αα  =  A_αα ")
                    ODPAR
                 ELSE FOR EACH α ∈ {α∈ L_{ν+1} | α_{ν-NIV} = 2}
                    PARDO
                     SET   C(α,NIV)={β∈L_{ν+1} | β=α ou β_{(ν:ν-NIV+1)} =α_{(ν:ν-NIV+1)}}
                     DO    β= Min (C)  REPEAT ( β= Min (γ ∈ C,γ > β))
                                       UNTIL ( β= α)
                     Identifier-IC (L_αβ ; "L_αβ L^T_ββ  = A_αβ -Σ  L_αγ L^T_γβ")
                                                          γ_γ ∈ C(α,NIV)
                                                           γ ≤ β
                     END
                     ODPAR
        END
END FACTORISATION-INCOMPLETE ;
```

Algorithme ALG-1

Preuve de ALG-1

Soit α un multi-indice de $L_{\nu+1}$, nous définissons la fonction

$$\text{(2.9)} \qquad \text{diss}(\alpha) = \begin{cases} -1 & \text{si} \quad \alpha_k \neq 2 \qquad \forall k \\ k & \text{si} \quad \alpha_k = 2 \end{cases}$$

L'image $\text{diss}(\alpha_{\nu+1})$ est $\{-1,\ldots,\nu-1\}$. Nous allons montrer par récurrence sur k que l'algorithme ALG-1 revient à identifier simultanément les lignes de L correspondant à $\text{diss}^{-1}(k)$ pour chaque k de -1 à $\nu-1$.

a) $k = -1$

Si $\alpha \in \text{diss}^{-1}(-1)$ alors on ne peut avoir simultanément, d'après le lemme 1 $\beta \leq \alpha$ et $A_{\alpha\beta} \neq 0$ que si $\beta = \alpha$. Ceci fait que le seul bloc $L_{\alpha\beta}$ que l'on peut identifier est $L_{\alpha\alpha}$, à l'aide de l'équation

$$(2.10) \qquad A'_{\alpha\alpha} = L_{\alpha\alpha} L^T_{\alpha\alpha} + R_{\alpha\alpha}$$

b) Supposons calculées les lignes correspondant aux indices de $\text{diss}^{-1}(-1,,k-1)$. Soit $\alpha \in \text{diss}^{-1}(k)$.
Du fait du lemme 1 les seuls blocs $A'_{\alpha\beta}$ avec

$$(2.11) \qquad \beta \leq \alpha \quad \text{et} \quad A'_{\alpha\beta} \neq 0$$

sont ceux qui vérifient

$$(2.12) \qquad \beta \in C(\alpha, \nu-k) = \{\beta \mid (\beta=\alpha) \text{ ou } \alpha_{(\nu:k+1)} = \beta_{(\nu:k+1)}\}$$

En particulier $\beta \in C(\alpha,\nu-h)$ entraine soit $\alpha=\beta$ soit $\beta \in \text{diss}^{-1}(-1,,\ldots,k-1)$ ce qui va permettre d'utiliser l'hypothèse de récurrence.
De plus si $\gamma \in C(\alpha,\nu-k)$ et $\delta \in L_{\nu+1}$ alors

$$(2.13) \qquad \gamma \leq \delta \leq \alpha \quad \text{entraine} \quad \delta \in C(\alpha,\nu-k)$$

Ceci montre que les seuls blocs $L_{\alpha\beta}$, non nuls a priori, sont ceux pour lesquels $\beta \in C(\alpha,\nu-k)$: il suffit donc de les identifier.
Pour calculer $L_{\alpha\beta}$ on part de

$$(2.14) \qquad A'_{\alpha\beta} = \sum_{\substack{\gamma \leq \alpha \\ \gamma \leq \beta}} L_{\alpha\gamma} L^T_{\gamma\beta} + R_{\alpha\beta}$$

En supprimant de cette sommation les produits nuls :

$$(2.15) \qquad A'_{\alpha\beta} = \sum_{\substack{\gamma \in C(\alpha,\nu-k) \\ \gamma \leq \beta}} L_{\alpha\gamma} (L_{\beta\gamma})^T + R_{\alpha\beta} .$$

Ceci montre qu'il est possible d'identifier les $L_{\alpha\beta}$ en considérant (2.15) pour β décrivant $C(\alpha,\nu-k) - \{\alpha\}$ en ordre croissant. On remarque en particulier

que les $(L_{\beta\gamma})^T$ ont déjà été calculés car $\beta \in diss^{-1}(-1,,k-1)$.
Le calcul de $L_{\alpha\alpha}$ part de l'équation

$$(2.16) \qquad A'_{\alpha\alpha} = \sum_{\gamma \in C(\alpha,\nu-k)} L_{\alpha\gamma} (L_{\gamma\alpha})^T + R_{\alpha\alpha}$$

Les $L_{\alpha\gamma}$ pour $\gamma \in C(\alpha,\nu-k)-\{\alpha\}$ viennent d'être calculés ce qui permet cette dernière identification.
Ceci achève la preuve par récurrence de la validité de l'algorithme ALG-1.

Nous venons ainsi de démontrer le

Théorème 5

Supposons que la matrice A' admette une décomposition de Choleski incomplète. Alors cette décomposition est donnée par l'algorithme ALG-1.

En prenant le cas particulier du graphe $G^*(A')$ on obtient le

Corollaire :

Supposons la matrice A symétrique définie positive. Alors la décomposition de Choleski de A' est donnée par l'algorithme ALG-1. ∎

II.5. Résolution des systèmes triangulaires issus de la factorisation incomplète.

Nous considérons maintenant la résolution de systèmes linéaires de la forme

$$(2.17) \qquad L\,L^T x = b$$

où les inconnues ont été permutées suivant la permutation construite au §. 2.3 et où L est une matrice triangulaire inférieure dont la structure de remplissage est donnée par (2.4).

Nous considérerons successivement la résolution de

$$(2.18) \qquad L\,y = b \qquad \text{("Descente")}$$

$$(2.19) \qquad L^T x = y \qquad \text{("Remontée").}$$

La résolution de ces systèmes est obtenue en utilisant respectivement les algorithmes ALG-2 et ALG-3 ci-dessous qui exploitent les possibilités de parallélisme introduites par la décomposition en dissection emboitées.

DESCENTE :

FOR NIV = $\nu+1$ STEP -1 UNTIL NIV=1

DO

IF NIV = $\nu+1$

THEN FOR-EACH $\alpha \in B_{\nu+1}\{0,,2\nu-1\} \cap L_{\nu+1}$

PARDO

$Y_\alpha = (L_{\alpha\alpha})^{-1} b_\alpha$

ODPAR

ELSE FOR-EACH $\alpha \in \{\alpha \in L_{\nu+1} \mid \alpha_{\nu-NIV} = 2\}$

PARDO

FOR-EACH $\beta \in B(\alpha,NIV) = \{\beta \in L_{\nu+1} \mid \beta \neq \alpha ,$

$\beta_{(\nu:\ \nu-NIV+1)} = \alpha_{(\nu:\ \nu-NIV+1)}\}$

PARDO

$Z_{\alpha\beta} = L_{\alpha\beta} Y_\beta$

ODPAR ;

$Y_\alpha = L_{\alpha\alpha}^{-1} (b_\alpha - \sum_{\beta \in B(\alpha,NIV)} z_{\alpha\beta})$

ODPAR

END

ALG-2

REMONTEE :

FOR NIV=1 STEP 1 UNTIL NIV = $\nu+1$

DO

IF NIV = $\nu+1$

THEN FOR-EACH $\alpha \in Bin_{\nu+1}\{0,\ldots,2^\nu-1\} \cap L_{\nu+1}$

PARDO $X_\alpha = (L_{\alpha\alpha}^T)^{-1} Y_\alpha$ ODPAR

ELSE FOR-EACH $\alpha \in \{\alpha \in L_{\nu+1} \mid \alpha_{\nu-NIV} = 2\}$

PARDO

$X_\alpha = (L_{\alpha\alpha}^T)^{-1} Y_\alpha$;

FOR EACH $\beta \in B(\alpha,NIV)$ PARDO $Y_\beta = Y_\beta - L_{\beta\alpha}^T X_\alpha$ ODPAR

ODPAR

END

ALG-3

Preuve de ALG-2 :

Cette démonstration utilise les mêmes notations que la preuve de ALG-1 ; elle revient à montrer que l'algorithme calcule simultanément les composantes de y : y_α pour $\alpha \in diss^{-1}(k)$ et ce pour k variant de -1 à $\nu-1$.

a) Prenons $k = -1$ et $\alpha \in diss^{-1}(-1)$. Le seul bloc $L_{\alpha\beta} \neq 0$ est $L_{\alpha\alpha}$ d'après le lemme 1 et en utilisant le caractère triangulaire inférieur de L. On a donc l'équation permettant le calcul de Y_α

$$L_{\alpha\alpha}\ Y_\alpha = b_\alpha \qquad (2.20)$$

b) Supposons déjà calculés les Y_α pour $\alpha \in diss^{-1}(-1,,k-1)$ et soit $\alpha \in diss^{-1}(k)$. Le lemme 1 montre que les seuls blocs $L_{\alpha\beta} \neq 0$ sont ceux qui vérifient :

$$\beta \in C(\alpha,\nu-k) = \{\beta | \alpha_{(\nu:k+1)} = \beta_{(\nu:k+1)}\} \cup \{\alpha\} \quad .$$

De plus $\beta \in C(\alpha,\nu-k)$ et $\beta \neq \alpha$ entrainent : $\beta \in diss^{-1}(-1,,k-1)$ ce qui montre que les Y_β correspondants ont déjà été calculés.
On a donc

$$\sum_{\beta \in C(\alpha,\nu-k)} L_{\alpha\beta}\ Y_\beta = b_\alpha$$

ce qui peut se mettre sous la forme

$$L_{\alpha\alpha}\ Y_\alpha = b_\alpha - \sum_{\substack{\beta \in C(\alpha,\nu-k) \\ \beta \neq \alpha}} L_{\alpha\nu}\ Y_\beta \qquad (2.21)$$

et permet le calcul de Y_α , le terme de droite étant connu.

Ceci achève la preuve par récurrence. ∎

Preuve de ALG-3

L'algorithme de remontée ALG-3 revient à calculer simultanément les X_α pour $\alpha \in diss^{-1}(k)$ et en faisant varier k de $\nu-1$ à -1.

1) : Première étape. Soit $\alpha \in diss^{-1}(\nu-1)$. Le lemme 1 montre que le seul bloc $L^T_{\alpha\beta} \neq 0$ est $L^T_{\alpha\alpha}$, et on peut donc calculer X_α à partir de :

$$L^T_{\alpha\alpha} \; X_\alpha = Y_\alpha \tag{2.22}$$

2) $(\nu-k)$-ième étape : On suppose déjà calculés les X_α correspondant à $\alpha \in diss^{-1}(\nu-1,..,k+1)$. D'après le lemme 1 les multi-indices $\beta \neq \alpha$ tels que $L^T_{\alpha\beta} \neq 0$ sont caractérisés par :

$$\begin{cases} \beta \in diss^{-1}(\ell) \quad \ell \in [\nu-1,\ldots,k+1] \quad \text{et} \\ \alpha_{(\nu:\ell+1)} = \beta_{(\nu:\ell+1)} \end{cases} \tag{2.23}$$

Les indices vérifiant (2.23) peuvent aussi être décrits par :

$$\ell \in [\nu-1,\ldots,k+1] \qquad \beta = (\alpha_{(\nu:\ell+1)},2,(0,\ldots,0)) \tag{2.24}$$

Ceci permet d'écrire :

$$L^T_{\alpha\alpha} \; X_\alpha = Y_\alpha - \sum_{\ell=\nu-1}^{k+1} L^T_{\alpha\beta(\ell)} \; X_{\beta(\ell)} \tag{2.25}$$

$$\beta(\ell) = (\alpha_{(\nu:\ell+1)},2,(0,\ldots,0))$$

Les X_β du terme de droite ayant été calculés aux étapes précédentes, il est possible de calculer X_α à l'aide de (2.25). Qui plus est un terme de la sommation est calculé à chacune des étapes précédentes : ceci permet d'accumuler cette somme au fur et à mesure des étapes précédentes.
Ceci rend possible l'accumulation des termes $Y_\beta = Y_\beta - L^T_{\beta\alpha} \; X_\alpha$ dans l'algorithme ALG-3 : à l'intérieur d'une même étape (boucle séquentielle sur NIV) une composante Y_β de Y subit au plus 1 accumulation et il n'y a ni conflit ni partage d'une même composante Y_β .

Ceci achève la preuve par récurrence de l'algorithme ALG-3. ∎

```
DESCENTE-VARIANTE :      /* UTILISER EN LIEU ET PLACE DE DESCENTE  */
  FOR NIV =   ν+1  STEP -1  UNTIL NIV = 1
   DO
     IF NIV =   ν+1
     THEN FOR EACH       α ∈ Bin ν+1 {0,,2^ν-1 } ∩ L ν+1
     PARDO
            Y_α = (L_αα)^-1 b_α   ;
            FOR EACH    γ ∈ {γ ∈ L ν+1 | ∃ k γ_ν-k = 2 et α(ν:ν-k) =γ(ν:ν-k)}
            PARDO
            (ARBITRE) : b_γ = b_γ - L_γα Y_α
            ODPAR
            ELSE FOR EACH  α ∈ {α ∈ L ν +1| α_ν-NIV = 2}
     ODPAR
     PARDO
     Y_α =  L_αα^-1 b_α ;
     FOR EACH      γ ∈ {γ|γ ∈ L ν+1 et (∃ k ≤ NIV  γ_ν-k = 2  et
     PARDO                                  α(ν:ν-k) = γ(ν: ν-k))}
     (ARBITRE) : b_γ  =  b_γ - L_γα Y_α
     ODPAR
     ODPAR
   END
```

ALG-4

La variante ALG-4 de l'algorithme de "descente" ALG-2 a pour but de répartir le calcul des sommations

$$b_\alpha - \sum_{\beta \in B} z_{\alpha\beta}$$

intervenant dans ALG-2. Une difficulté supplémentaire apparait : on peut être amené à tenter plusieurs sommations sur un même b_γ simultanément. Pour permettre l'implémentation de cette variante il y a lieu d'employer une méthode

d'arbitrage de ces conflits d'accès symbolisée par les préfixes "(ARBITRE)" :. L'évaluation de l'intérêt de cette variante dépend de façon essentielle du coût de cet arbitrage et ne sera pas discutée ici.

Nous venons de prouver le

Théorème 6 :

Considérons le système linéaire $LL^T Y = b$ où L est inversible, de structure donnée par (2.4). Supposons les inconnues permutées selon la construction "par dissection emboitées" indiquée au §. 2.3.
Alors la résolution du système linéaire peut être effectuée en employant les algorithmes ALG-2 et ALG-3.

Ce résultat nous permet donc d'introduire un taux de parallélisme intéressant dans la résolution de systèmes linéaires issus de décompositions de Choleski incomplètes ou non.

II.5. Répartition des tâches et transferts d'information entre les processus.

Le parallélisme que nous venons de mettre en évidence dans les algorithmes de décomposition (incomplète) et de résolution reste à exploiter en affectant les tâches aux processeurs, et en les synchronisant entre elles. La réalisation automatique et optimale de cette répartition ne sera pas abordée ici ; nous nous contenterons d'indiquer un mode de répartition d'une mise en pratique aisée et tirant parti de la structure récursive des algorithmes.

Nous supposons donc maintenant que nous disposons de 2^ν processeurs dotés chacun d'une mémoire locale.
La matrice A' et la matrice L ([1]) ont la structure décrite en §. 2.2. Soit $A_{\alpha\beta}$ un bloc de A' ; il est affecté à la mémoire locale du processeur p avec : (cf. II.8 pour la définition de p(.))

$$(2.26)\quad \begin{cases} \text{si } \alpha \leq \beta & p = p(\alpha) \\ \text{si } \alpha \geq \beta & p = p(\beta) \end{cases}$$

([1]) Il n'est pas nécessaire de représenter la transposée L^T de L.

La même règle est utilisée pour représenter L ([1]).
Les vecteurs b, y et x intervenant sont aussi décomposés par blocs et la règle d'affectation est

(2.27) v_α est affecté à $p = p(\alpha)$

pour v = b, y ou x.

Ceci étant, nous pouvons écrire l'algorithme que doit effectuer un processeur p quelconque et expliciter les transferts d'information.

La résolution du système $LL^T x = b$ revient à faire exécuter par chacun des processeurs le programme :

```
RESOL : PROC (np)  ;   /* np : numéro de processeur  */
          DESCENTE (np) ;
          REMONTEE (np) ;
          END RESOL     ;
```

Tous les aspects liés à la transmission des données et à la synchronisation étant décrite dans ALG-5 et ALG-6.

([1]) La généralisation au cas d'une décomposition LU est immédiate et convient aussi.

```
DESCENTE :
   PROC (np) ;                          /*np : numéro de processeur  */
      λ = Bin_{ν+1}(np)
     FOR NIV =  ν+1   STEP -1  UNTIL NIV = 1
        DO ;
         IF (NIV = ν+1)  THEN   y_λ = (L_λλ)^{-1} b_λ    ;
                         ELSE
                  DO  ;
                   α= (λ_{ν:ν+1-NIV},2,(0,...,0)) ; B = {B(α,NIV)|p(β) = np}  ;
                  z_{α,np} = Σ_{β∈B} L_αβ  Y_β     ;

                  IF p(α) ≠ np THEN SEND z_{α,np} to p(α)  ;
                               ELSE
                  DO  ;
                      FOR-EACH m ∈{m ≠ np|  (Bin_{ν+1}(m))_{ν:ν+1-NIV} = λ }
                               DO  ;
                                RECEIVE Z_{α,m} from    m   ;
                                b_α =  b_α -  z_{α,m}

                               END ;
                            Y_α  =  (L_αα)^{-1} b_α
                  END ;

                  END ;

         END ;

END DESCENTE  ;
```

ALG-5

```
REMONTEE :
      PROC (np) ;
          λ = Bin_{ν+1}(np)  ;
        FOR NIV = 1  STEP 1  UNTIL  NIV =  ν+1
        DO ;
            IF NIV <  ν+1  THEN α = (λ_{ν:ν+1-NIV} 2,(0)*)  ;
                           ELSE     α =  λ                  ;
            IF NIV >   1   THEN
            DO  ;
              β = (α_{ν:ν+1 - NIV}),2,(0)*)  ;
              IF p(β) ≠ np THEN RECEIVE X_β  FROM p(β) ;
              FOR-EACH  γ ∈ {{α}  ,{ (λ_{ν:ν+1-NIV}) 0*)}}
                    DO  ;
                    Y_γ  =  Y_γ  - (L^T_γβ) X_β
                    END  ;
            END  ;
              IF p(α) = np THEN X_α = (L^T_αα)^-1 Y_α  ;
            IF NIV  <  ν+1
                    THEN FOR-EACH pє {p=p(γ) et  γ_{ν:ν-NIV+1}  =  λ_{ν:ν+1-NIV}}
                       DO  ;
                       SEND  X_α  TO  p
                       END  ;
       END ;
END REMONTEE  ;
```

ALG-6

II.6. Répartition des tâches : l'algorithme du gradient conjugué préconditionné

Dans ce qui suit on a partitionné les matrices et les vecteurs suivant les méthodes exposées au paragraphe II.3. On a réparti entre les processeurs la matrice L et les vecteurs u, s, r et d suivant (2.26) et (2.27). La répartition des blocs de A' suivant (2.26) est utilisée pour le calcul de sa décomposition incomplète par l'algorithme ALG-1. Pour le calcul de produits du type Au nous utilisons une autre répartition de A'.

Lemme :

Supposons que la matrice A' vérifie les hypothèses du Lemme 1. Supposons de plus le bloc

(2.28) $A'_{\alpha\beta}$ affecté au processeur $p(\alpha)$.

Alors le produit A'u peut être calculé par l'algorithme ALG-7 ci-après.

Note : Si on stocke A' de manière redondante pour utiliser simultanément les répartition (2.26) et (2.28) seuls les blocs $A'_{\alpha\beta}$ avec $\alpha > \beta$ et $p(\alpha) \neq p(\beta)$ devront être stockés 2 fois.
Ceci nécessite une place mémoire supplémentaire majorée par $k \sum_i |S_i|$ (où $\sum_i S_i$ représente l'union des séparateurs intervenant en II.6 et k est le nombre maximum de coefficients non nuls sur une ligne de A).

■

ALG-7

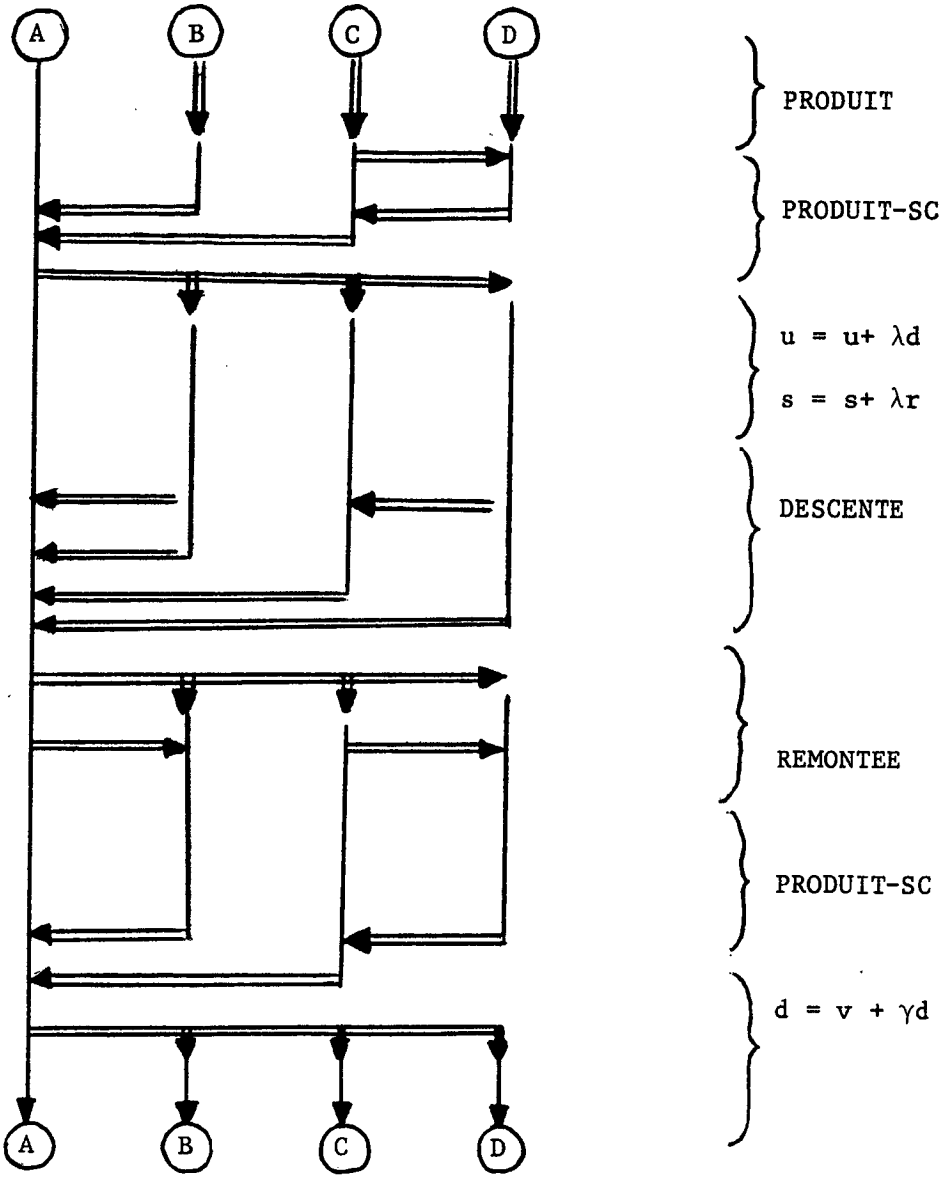
```
PRODUIT : PROC(np,u,v)   ;          /* v = Au        */
                                    /* np = numéro de processeur   */
 SET   F (x) = {α | A'αx ≠ 0 d'après lemme 1}   ;
 FOR EACH β ∈ p^-1(np)
      DO ;
      FOR EACH    α ∈ F(β)
        DO ;
         IF  p(α) ≠ np  THEN SEND (u(β)) TO p(α)   ;
        END ;
     END ;
FOR EACH β ∈ ∪ F(α)
      DO ; α∈ p^-1(np)
        IF p(β) ≠ np THEN RECEIVE u(β) FROM p(β)   ;

      END ;
FOR EACH   α ∈ p^-1(np)
     DO
     vα   = 0  ;
     FOR EACH  β   ∈ F(α)
         DO  ;
         vα  =   vα + A'αβ  vβ   ;
         END  ;
      END ;
END PRODUIT  ;
```

La tâche d'un processeur dans l'algorithme du Gradient Conjugué préconditionné peut maintenant être explicitée complètement

```
GCCI(np)
    FOR EACH  α ∈ p^-1(np) u_α  = (u_o)_α   ;
    PRODUIT (np, u, s)  ;
    FOR EACH  α ∈ p^-1(np) s_α = s_α - b_α  ;
    DESCENTE (np, s, u)  ;
    REMONTEE (np, u, r)
    FOR EACH  α ∈ p^-1(np) d_α = -r_α  ;
    PRODUIT-SC (np, d, r, η)  ;
BOUCLE : PRODUIT(np, d, r)  ;
    PRODUIT-SC(np, d, r, λ)  ;
    λ = η / λ
    FOR EACH α ∈ p^-1(np) DO ; u_α = u_α + λ d_α , s_α = s_α + λ r_α , END ;
    DESCENTE (np, s, u)
    REMONTEE (np, u, r)
    PRODUIT-SC (np, r , s, ξ)
    IF ξ ≤ ε   GO TO FIN
    γ = ξ / η   ;   η = ξ
    FOR EACH  α ∈ p^-1(np)   d_α = - r_α + γ d_α  ;
    GO TO BOUCLE
FIN  ;
    END GCCI  ;
```

La synchronisation des processeurs est entièrement décrite par le mécanisme de transmission SEND-RECEIVE et intervient dans les procédures PRODUIT, PRODUIT-SC, DESCENTE, REMONTEE.
Du fait de la présence de produits scalaires on a synchronisé complètement les processeurs 3 fois par boucle d'itération.
Dans le cas de 4 processeurs on aboutit au schéma suivant pour la boucle principale de l'algorithme du Gradient Conjugué préconditionné. (Sur cette figure on a simplement fait apparaitre les transferts de données ; on n'a pas cherché à montrer l'importance relative des tâches).


A
B
C
D
PRODUIT
PRODUIT-SC
u = u+ λd
s = s+ λr
DESCENTE
REMONTEE
PRODUIT-SC
d = v + γd
A
B
C
D

(2.29)

II.7. Estimation de l'accélération

Nous faisons cette estimation dans un modèle extrémement simplifié :

- le volume de transmission est estimé comme proportionnel à la somme du nombre de variables flottantes transférées d'un processeur à un autre. (Si la même variable est transmise à p destinataires on a donc compté p transmissions).

- le volume des calculs pour un processeur est estimé proportionnel au nombre d'opérations flottantes à effectuer par ce processeur.

- le temps de calcul pour un processeur est la somme du volume des calculs qu'il effectue et du temps décompté en nombre d'opérations flottantes) passé à attendre d'autres processeurs

- le temps total permettant de calculer l'accélération est la somme :

a) du maximum des temps de calculs.

b) du temps de transmissions estimé comme une fonction linéaire du nombre de transferts et du volume de transmissions.

Concernant la structure de la matrice A nous faisons les hypothèses :

- que le graphe G(A) satisfait une propriété de séparation (19) avec $f(x) = x^{\gamma}$

- qu'à un sommet de G(A) ne parviennent pas plus de k-1 arêtes.

Ces hypothèses sont vérifiées dans le cas d'éléments finis plans généraux (cf. § I.3) et aussi pour certains maillages tridimensionnels particuliers.

Dans ces estimations, les quantités que nous définissons maintenant interviennent naturellement.

$$\Sigma = \sum_{\{\mu \in L_{\nu+1} \mid \exists j \ \mu_j = 2\}} |V_\mu| \qquad (2.30)$$

$$(2.31)\qquad \tilde{\Sigma} = \sum_{j=0}^{\nu-1} (\nu - j) \sum_{\{\mu \in L_{\nu+1} \mid \mu_j = 2\}} |V_\mu|$$

$$(2.32)\qquad \approx{\Sigma} = \sum_{j=0}^{\nu-1} 2^{\nu-j} \sum_{\{\mu \in L_{\nu+1} \mid \mu_j = 2\}} |\ _\mu|$$

$$(2.33)\qquad \Sigma_p = \max_q \sum_{\mu \in \{p^{-1}(q) \cap \{\mu \in L_{\nu+1} \mid \exists\, j\ \mu_j = 2\}\}} |V_\mu|$$

$$(2.34)\qquad \sigma_p = \max_q \sum_{\mu \in p^{-1}(q)} |V_\mu|$$

Ces notations représentent respectivement :

Σ : la taille totale des séparateurs

$\tilde{\Sigma}$: la somme des tailles des séparateurs pondérées par leur niveau

Σ_p : la taille maximum des séparateurs affectés à un même processeur.

Σ_p : le nombre maximum d'inconnues affectées à un même processeur.

Nous commençons par majorer ces quantités en fonction des paramètres

ν : nombre de niveaux de dissection

α, β, γ : caractérisation des propriétés du graphe G(A)

$N = |V|$: nombre d'inconnues.

Lemme :

Sous les hypothèses ci-dessus, on a les majorations :

$$(2.35)\qquad \Sigma \leq \beta\, N^\gamma\, \frac{(2\alpha^\gamma)^\nu - 1}{2\alpha^\gamma - 1}$$

$$(2.36)\qquad \tilde{\Sigma} \leq \beta\, N^\gamma\, \frac{(2\alpha^\gamma)^{\nu+1} - 2(\nu+1)\alpha^\gamma + \nu}{(2\alpha^\gamma - 1)^2}$$

$$(2.37)\qquad \Sigma_p \leq \beta\, N^\gamma\, \frac{1-(\alpha^\gamma)^\nu}{1 - \alpha^\gamma}$$

$$(2.38)\qquad \sigma_p \leq N\alpha^\nu + \beta\, N^\gamma\, \frac{1 - (\alpha^\gamma)^\nu}{1 - \alpha^\gamma}$$

$$(2.39) \qquad \tilde{\Sigma} \leq \beta N^{\gamma} \; 2^{\nu} \; \left(\frac{1- \alpha^{\gamma\nu}}{1 -\alpha^{\gamma}} \right)$$

Lemme :

Le temps de calcul au cours d'une itération dans la boucle de l'algorithme GCCI est majoré par :

$$M\text{-op} = \sigma_p(12+ 4k) + \Sigma_p (6k-2) + 2^{\nu+1}$$

Le nombre de variables flottantes transférées est majoré par

$$M\text{-tr} = 3 \tilde{\Sigma} + 2^{\nu+2}$$

Le nombre de transferts est majoré par :

$$M\text{-tr} = 3 \nu 2^{\nu} + 2^{\nu+2}$$

Dans le cas mono-processeur on utilise la

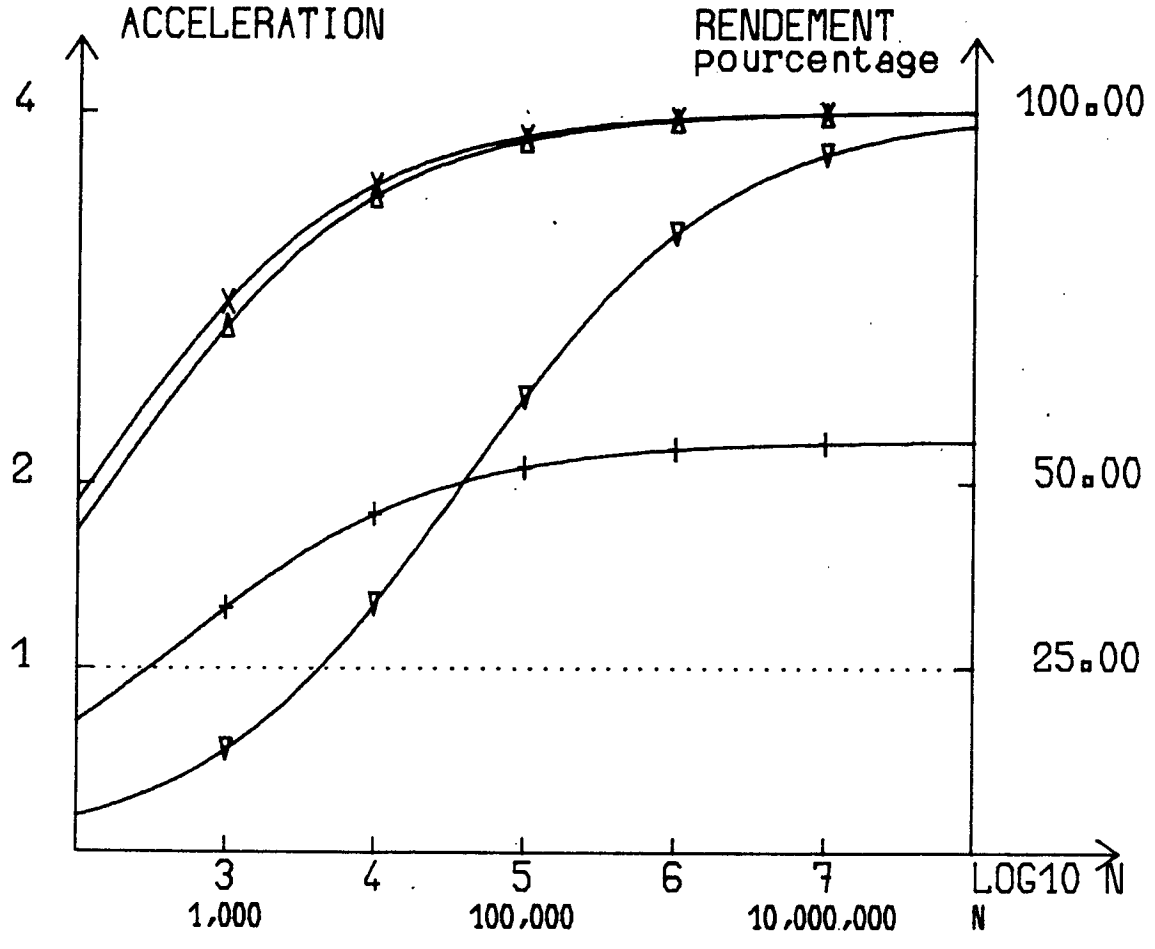
Référence : Le nombre d'opérations flottantes à réaliser au cours d'une itération dans l'algorithme GCCI mono-processeur est

$$M\text{-op-mono} = (12 + 4k)N$$

Note : Cette référence est approximative ; on a considéré que toutes les lignes de la matrice ont exactement k coefficients non nuls alors que ceci n'est, en toute rigueur qu'une majoration.

Les figures (2.1), (2.2), (2.3) donnent les accélérations auxquelles on aboutit pour divers maillages bidimensionnels suivant ν et le nombre d'inconnues N. Tous les domaines considérés satisfont une propriété de séparation avec

F I G 2 . 1

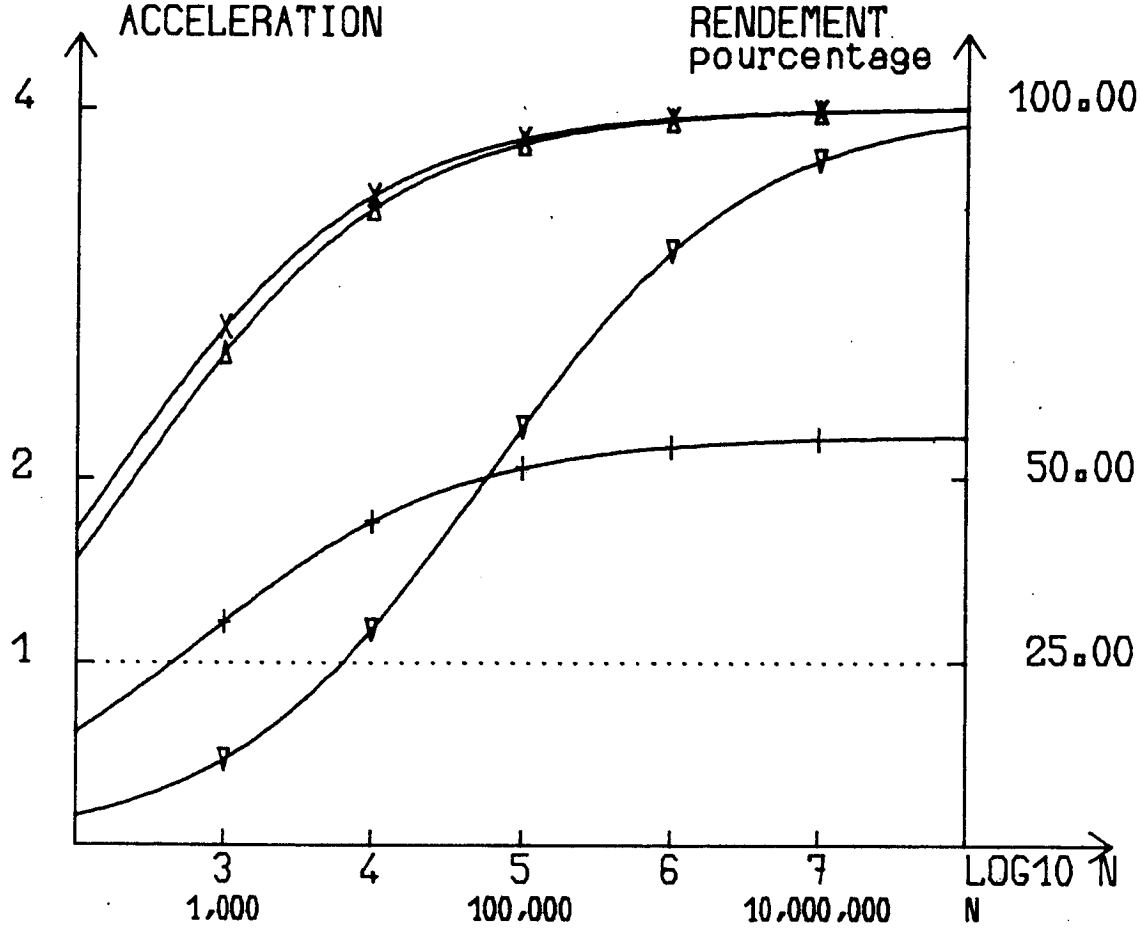


MAILLAGES BIDIMENSIONNELS 4 PROCESSEURS

variation de l acceleration suivant le domaine
TRANSFERTS : MEMOIRE PARTAGEE

	NU	ALPHA	BETA	GAMMA	K	TRINI	TRVIT
Δ	2	0.50	1.00	0.50	5	0.0	0.0
∇	2	0.50	15.40	0.50	5	0.0	0.0
+	2	0.67	2.83	0.50	5	0.0	0.0
X	2	0.50	0.85	0.50	5	0.0	0.0

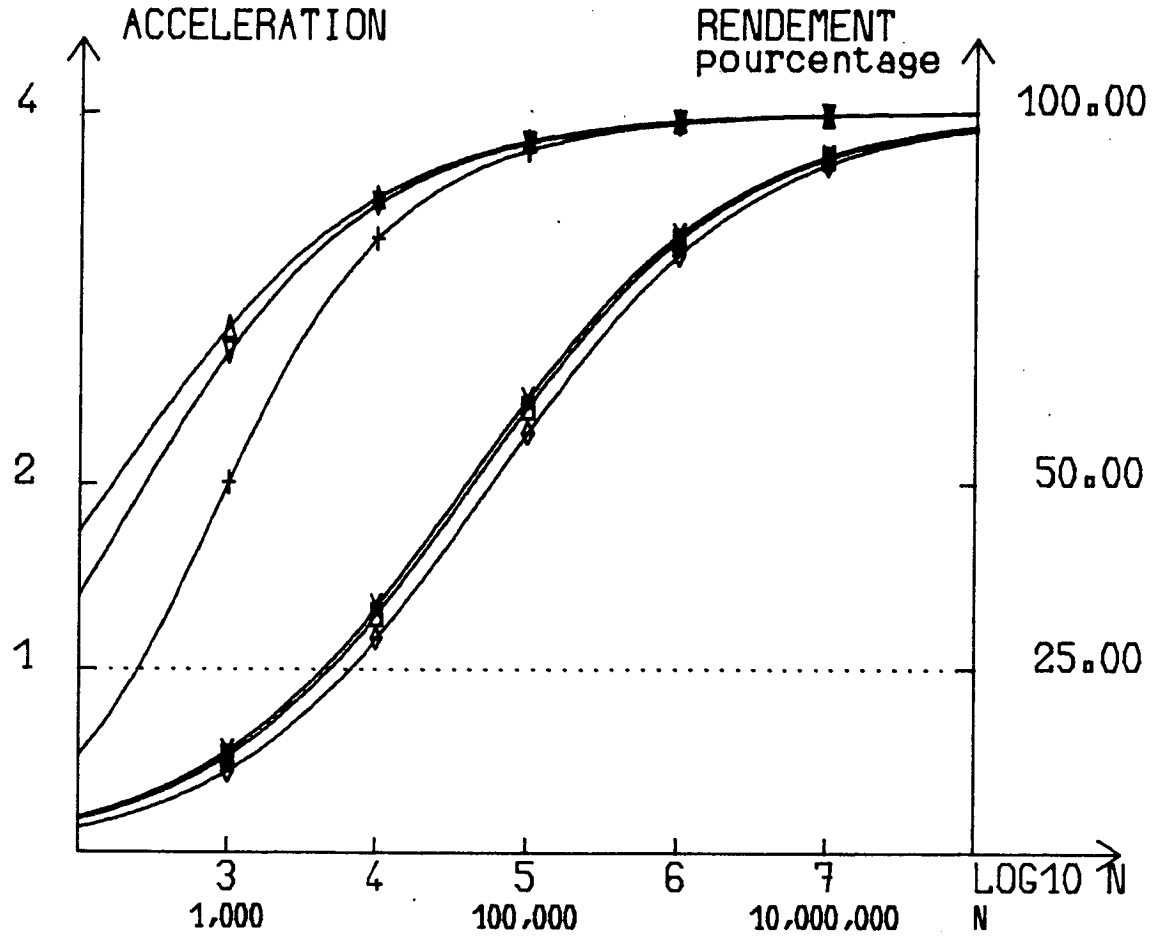
F I G 2 . 1bis



MAILLAGES BIDIMENSIONNELS 4 PROCESSEURS
element fini de plus haut degre
variation de l acceleration suivant le domaine
TRANSFERTS : MEMOIRE PARTAGEE

	NU	ALPHA	BETA	GAMMA	K	TRINI	TRVIT
Δ	2	0.50	1.00	0.50	19	0.0	0.0
∇	2	0.50	15.40	0.50	19	0.0	0.0
+	2	0.67	2.83	0.50	19	0.0	0.0
X	2	0.50	0.85	0.50	19	0.0	0.0

F I G 2.2

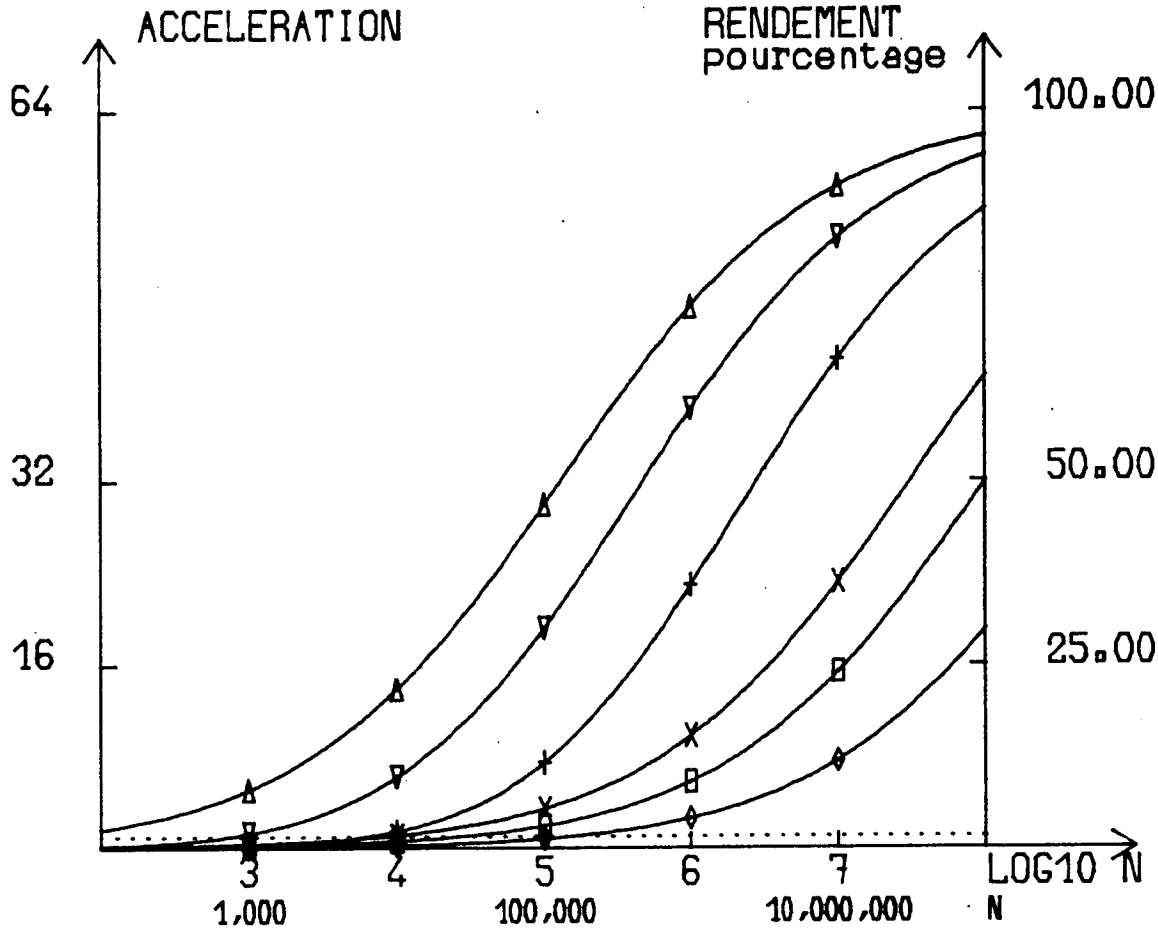


MAILLAGES BIDIMENSIONNELS 4 PROCESSEURS

variation de l acceleration suivant le domaine et les temps de transfert

	NU	ALPHA	BETA	GAMMA	K	TRINI	TRVIT
△	2	0.50	1.00	0.50	5	0.0	0.0
▽	2	0.50	1.00	0.50	5	10.0	0.3
+	2	0.50	1.00	0.50	5	100.0	1.0
X	2	0.50	15.50	0.50	5	0.0	0.0
□	2	0.50	15.50	0.50	5	10.0	0.3
◇	2	0.50	15.50	0.50	5	100.0	1.0

F I G 2.3



MAILLAGES BIDIMENSIONNELS 64 PROCESSEURS

variation de l acceleration suivant le domaine et les temps de transfert

	NU	ALPHA	BETA	GAMMA	K	TRINI	TRVIT
Δ	6	0.50	1.00	0.50	5	0.0	0.0
∇	6	0.50	1.00	0.50	5	10.0	0.3
+	6	0.50	1.00	0.50	5	100.0	1.0
X	6	0.50	15.50	0.50	5	0.0	0.0
□	6	0.50	15.50	0.50	5	10.0	0.3
◇	6	0.50	15.50	0.50	5	100.0	1.0

α	β	γ	Remarques
0.5	1.	0.5	maillage régulier du carré
0.5	15,4	0.5	graphe G(A) plan quelconque
0.67	2,83	0.5	graphe G(A) plan quelconque
0.5	0.85	0.5	maillage régulier du carré avec "trou" (valable pour $\nu \leq 3$)

On a aussi considéré sur ces figures l'effet des variations de la vitesse relative des transmissions de données entre les processeurs en faisant intervenir deux paramètres

tr-init : temps d'initialisation d'une transmission

tr-vit : temps nécessaire à la transmission d'une variable flottante après initialisation.

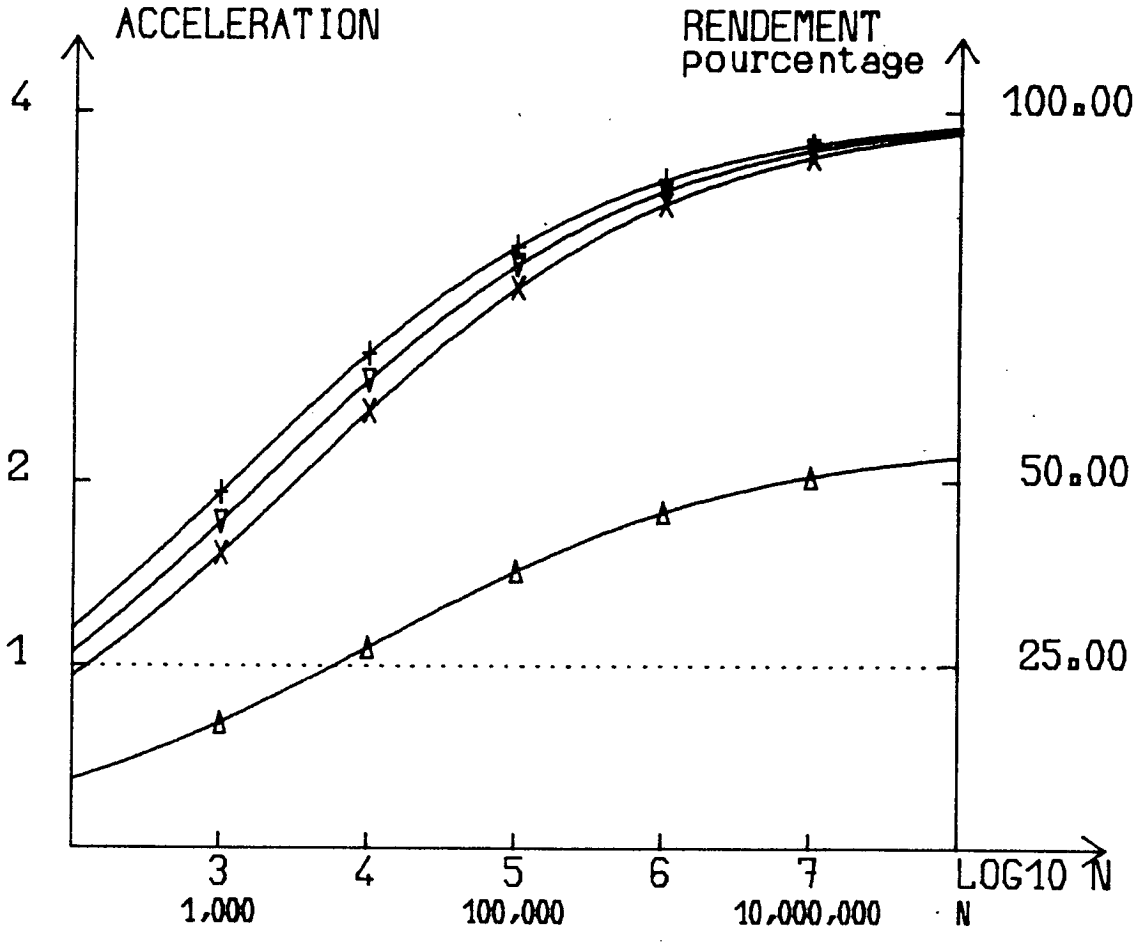
(ces paramètres sont évalués par rapport au temps unité nécessaire pour effectuer une opération flottante).

Les figures (2.4) et (2.5) illustrent le cas de domaines tridimensionnels. Faute de résultats généraux nous nous sommes contentés d'envisager des domaines ayant des propriétés comparables au maillage d'un cube par des éléments cubiques.

α	β	γ	Remarques
0.5	1.0	0.667	cube
0.5	0.85	0.667	cube "troué"
0.67	3.00	0.667	domaine moins "favorable"

Les figures 2.6 à 2.8 donnent l'évolution du rendement : accélération/nombre de processeurs en fonction du nombre de processeurs.

F I G 2.4

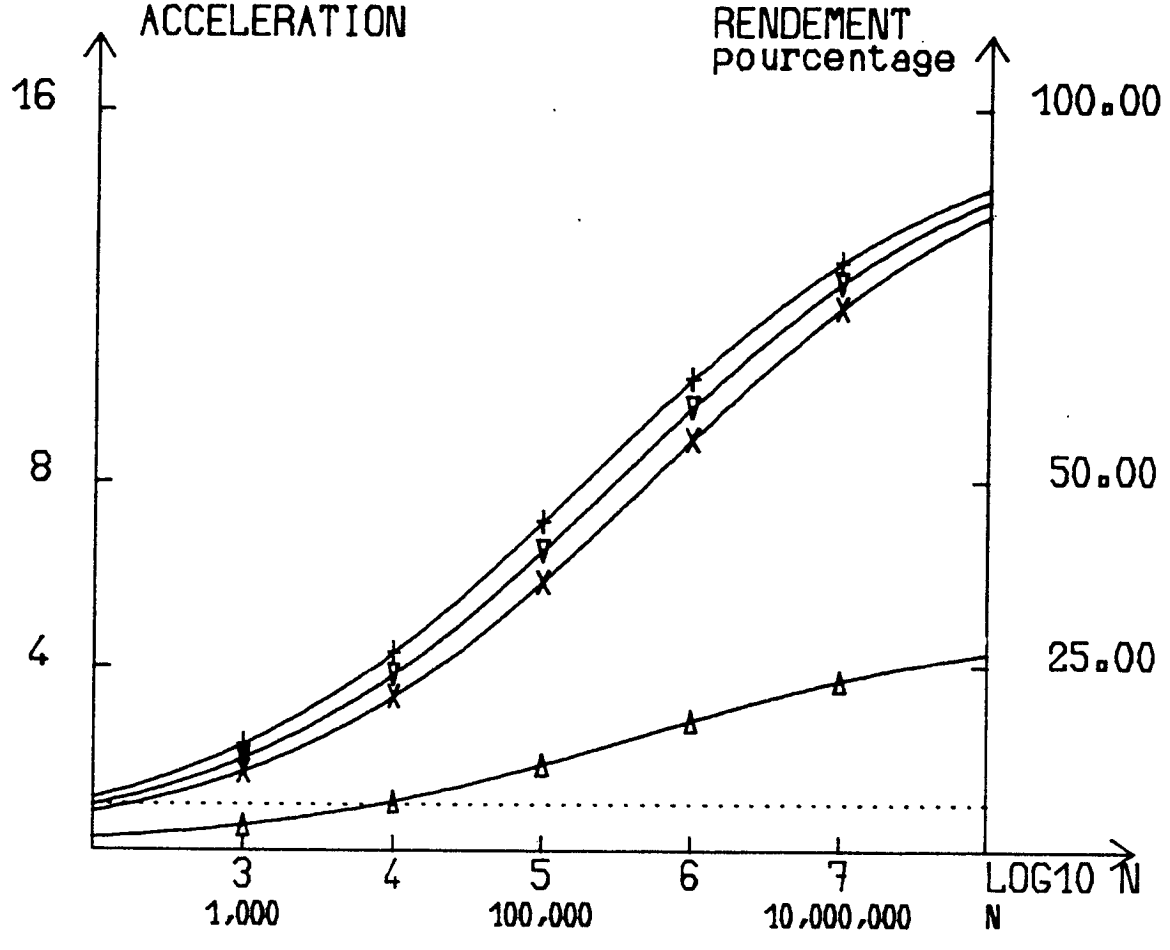


MAILLAGES TRIDIMENSIONNELS 4 PROCESSEURS

variation de l acceleration suivant le domaine
TRANSFERTS : MEMOIRE PARTAGEE

	NU	ALPHA	BETA	GAMMA	K	TRINI	TRVIT
Δ	2	0.67	3.00	0.66	7	0.0	0.0
∇	2	0.50	1.00	0.66	7	0.0	0.0
+	2	0.50	0.85	0.66	7	0.0	0.0
X	2	0.50	1.00	0.66	40	0.0	0.0

F I G 2 . 5

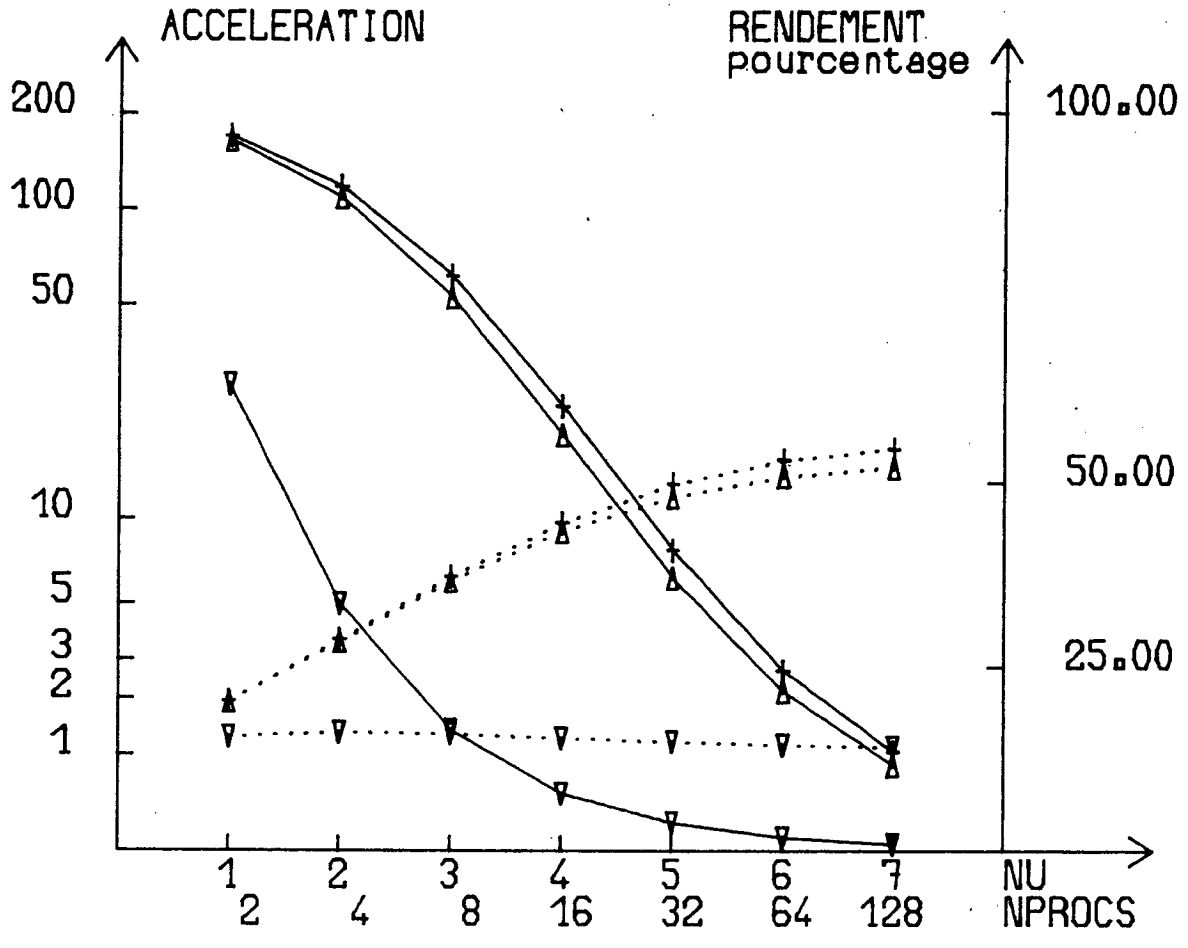


MAILLAGES TRIDIMENSIONNELS 16 PROCESSEURS

variation de l acceleration suivant le domaine
TRANSFERTS : MEMOIRE PARTAGEE

	NU	ALPHA	BETA	GAMMA	K	TRINI	TRVIT
Δ	4	0.67	3.00	0.66	7	0.0	0.0
∇	4	0.50	1.00	0.66	7	0.0	0.0
+	4	0.50	0.85	0.66	7	0.0	0.0
X	4	0.50	1.00	0.66	40	0.0	0.0

F I G 2.6

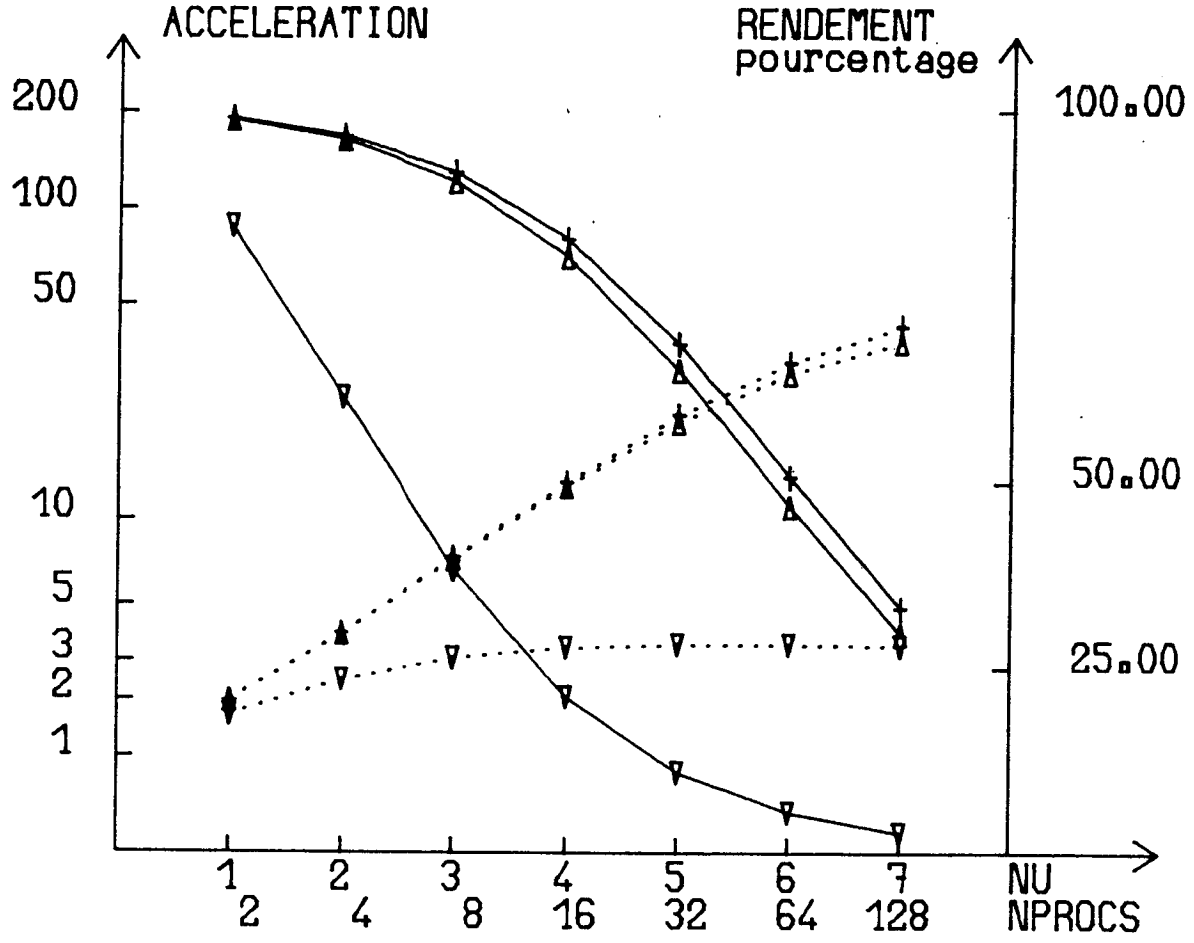


MAILLAGES BIDIMENSIONNELS

variation de l acceleration suivant le nombre de processeurs
TRANSFERTS : MEMOIRE PARTAGEE

	NVAR	ALPHA	BETA	GAMMA	K	TRINI	TRVIT
Δ	10000	0.50	1.00	0.50	5	0.0	0.0
∇	10000	0.50	15.40	0.50	5	0.0	0.0
+	10000	0.50	0.85	0.50	5	0.0	0.0

F I G 2 . 7

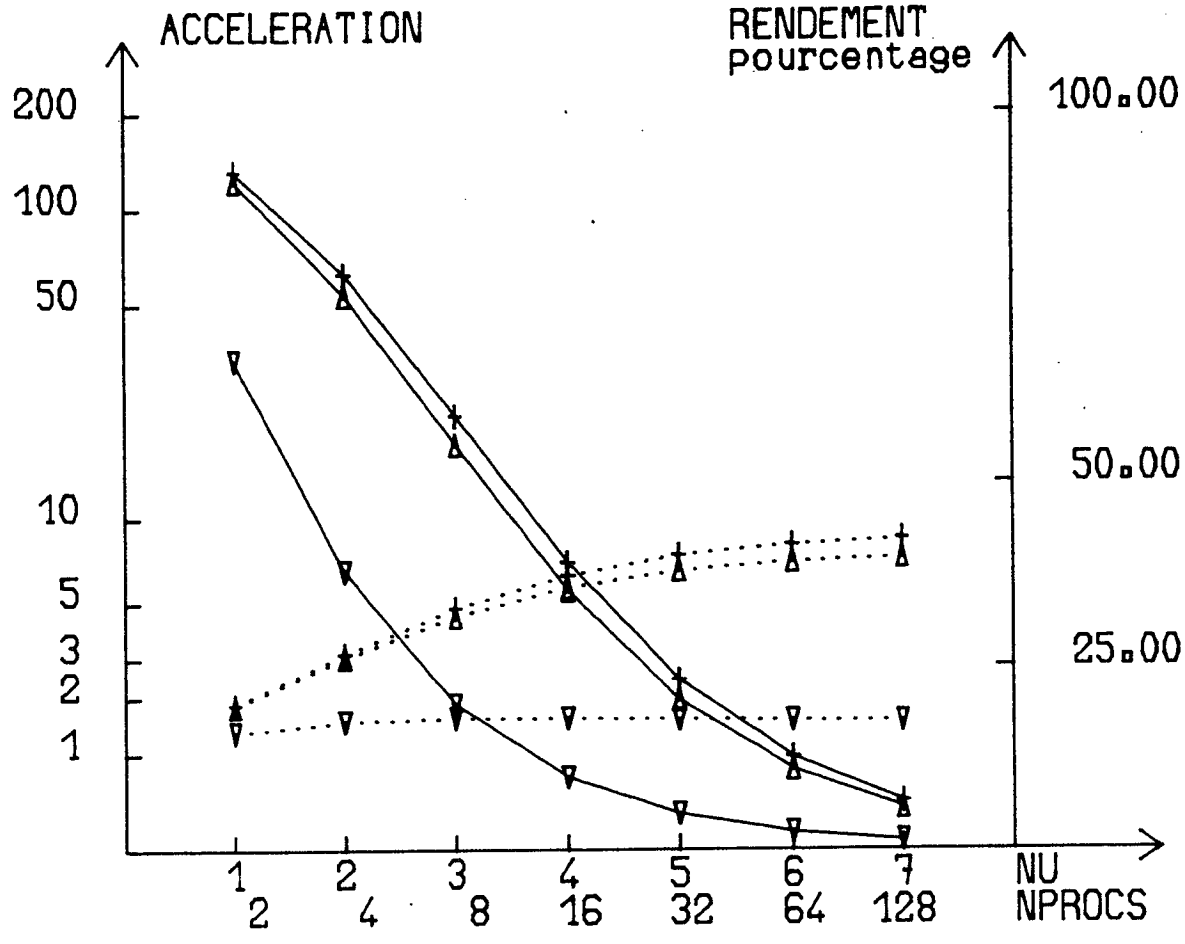


MAILLAGES BIDIMENSIONNELS

variation de l acceleration suivant le nombre de processeurs
TRANSFERTS : MEMOIRE PARTAGEE

	NVAR	ALPHA	BETA	GAMMA	K	TRINI	TRVIT
Δ	100000	0.50	1.00	0.50	5	0.0	0.0
∇	100000	0.50	15.40	0.50	5	0.0	0.0
+	100000	0.50	0.85	0.50	5	0.0	0.0

F I G 2.8



MAILLAGES TRIDIMENSIONNELS

variation de l acceleration suivant le nombre de processeurs
TRANSFERTS : MEMOIRE PARTAGEE

	NVAR	ALPHA	BETA	GAMMA	K	TRINI	TRVIT
Δ	100000	0.50	1.00	0.67	30	0.0	0.0
∇	100000	0.50	5.00	0.67	30	0.0	0.0
+	100000	0.50	0.85	0.67	30	0.0	0.0

III. EXPERIMENTATION NUMERIQUE

Nous venons de montrer que les méthodes de séparations étaient souhaitables en ce sens qu'elles permettaient d'accroitre le taux de parallélisme dans la méthode du Gradient Conjugué avec préconditionnement. Il importe maintenant de vérifier que ceci a pu être réalisé sans détruire les propriétés de convergence très rapide de la méthode (obtenues précisément grâce à ce préconditionnement). cf [K,L,V] .

A cet effet, nous avons considéré les matrices issues de la résolution par une méthode d'éléments finis "Q1" du problème elliptique

$$\begin{aligned} &-\Delta u + u = f \quad \text{dans} \quad \Omega =]0,1[\times]0,1[\\ &u|_{\partial\Omega} = 0 \quad \text{sur} \quad \partial\Omega \end{aligned} \tag{3.1}$$

ainsi que de diverses variantes obtenues par modification des conditions aux limites.

Au premier niveau de dissection nous avons utilisé le séparateur S pour diviser le domaine en 4 sous-domaines suivant la figure 3.1

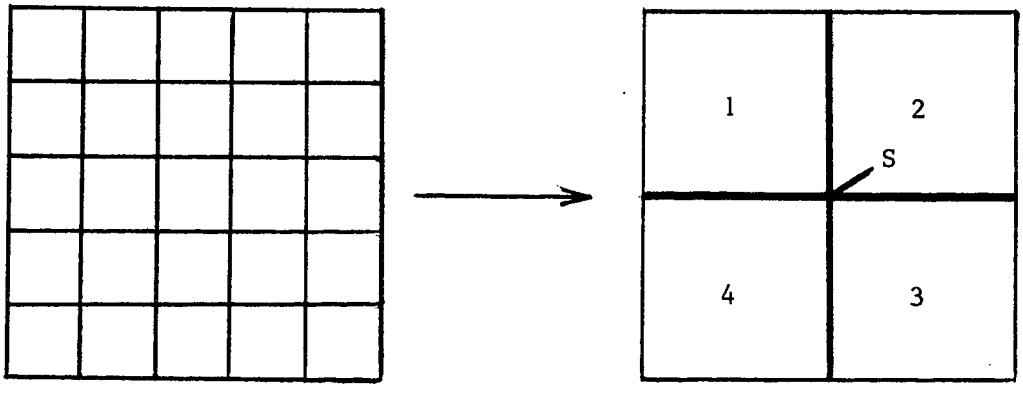


Figure 3.1.

Et pour atteindre le niveau ν de dissection poursuivi récursivement de la même façon dans chacun des 4 sous-domaines. Ceci fait qu'avec ν dissections nous avons introduit 4^{ν} processus au sens du § II.5/II.6/II/7 ([1])

([1]) Cette variante nous est apparue commode lors de l'expérimentation étant donné le domaine choisi.

III.1. Influence de la dissection sur la vitesse de convergence.

La figure 3.2 donne l'évolution du résidu $- \log_{10} \| Ax-b \|_2$ pour le système issu de (3.1) suivant le niveau de dissection $\nu = 0, 2$ ($1, 16$ sous-domaines).

La figure 3.3 illustre cette évolution pour le système issu du problème

$$(3.2) \quad \begin{cases} - \Delta u + u = f \quad \text{dans} \quad \Omega =]0,1[^2 \\ u|_{\{x=0\} \cap \partial\Omega} = 0 \qquad \frac{\partial u}{\partial \nu} = 0 \text{ sur } \partial\Omega \cap \{u \neq 0\} \end{cases}$$

La figure 3.4 permet de juger de l'effet d'une dissection pour le système issu du problème

$$(3.3) \quad \begin{cases} - \Delta u + u = f \quad \text{dans} \quad \Omega \\ \frac{\partial u}{\partial n} = 0 \quad \text{sur} \quad \partial\Omega \end{cases}$$

III.2. Influence de la dissection sur les propriétés spectrales de la matrice $L^{-1} A^1 (L^T)^{-1}$.

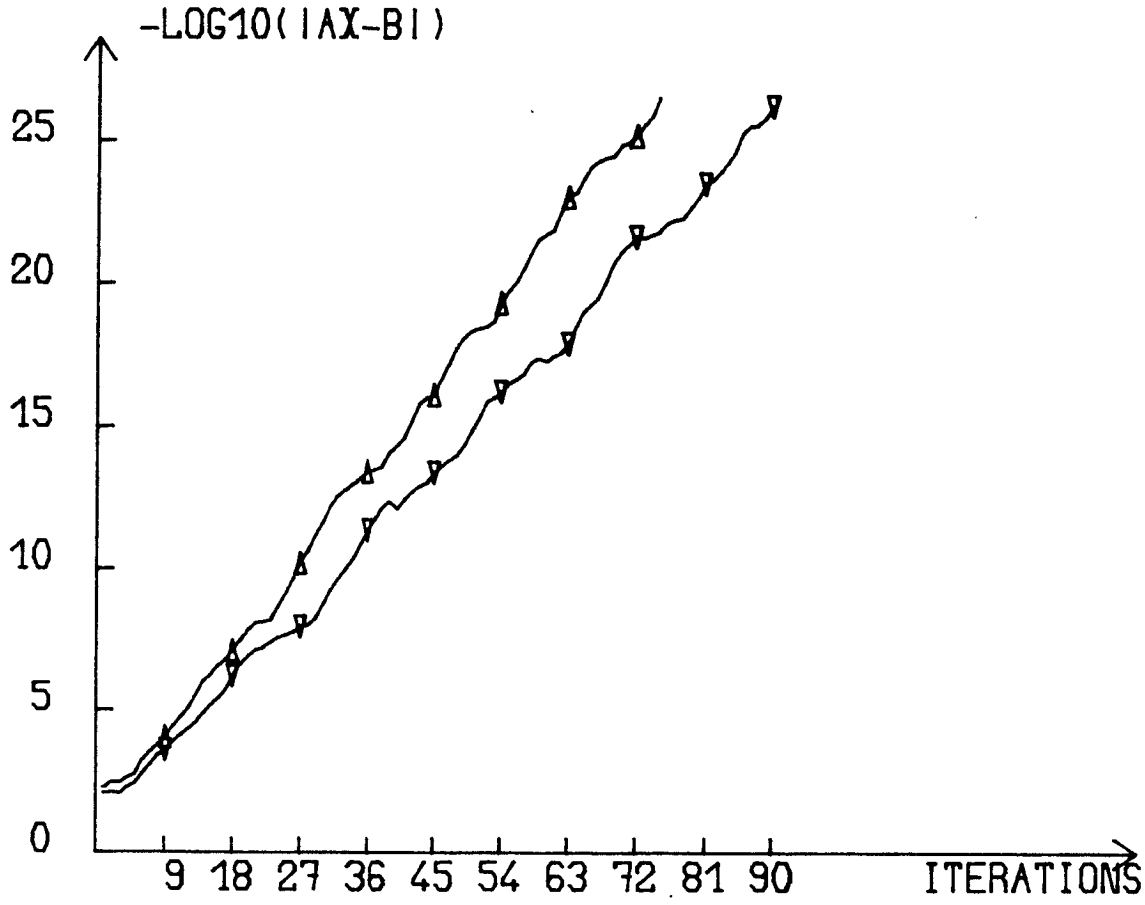
Comme nous l'avons indiqué au paragraphe II.1 le préconditionnement revient à appliquer la méthode du Gradient Conjugué à la matrice

$$(3.4) \qquad B = L^{-1} A' (L^T)^{-1}$$

où A' est déduite de A par la renumérotation (permutation) liée à la dissection et où LL^T est la décomposition de Choleski incomplète de A'. Dans la mesure où l'efficacité de la méthode avec préconditionnement est liée au spectre de B (qui varie du fait de la dissection) il est intéressant de montrer son évolution, même sur des exemples aussi simples que ceux du paragraphe précédent (cf. [W]).

La figure 3.5 montre la variation du spectre au fil des dissections pour le problème issu de (3.1).

FIG 3.2

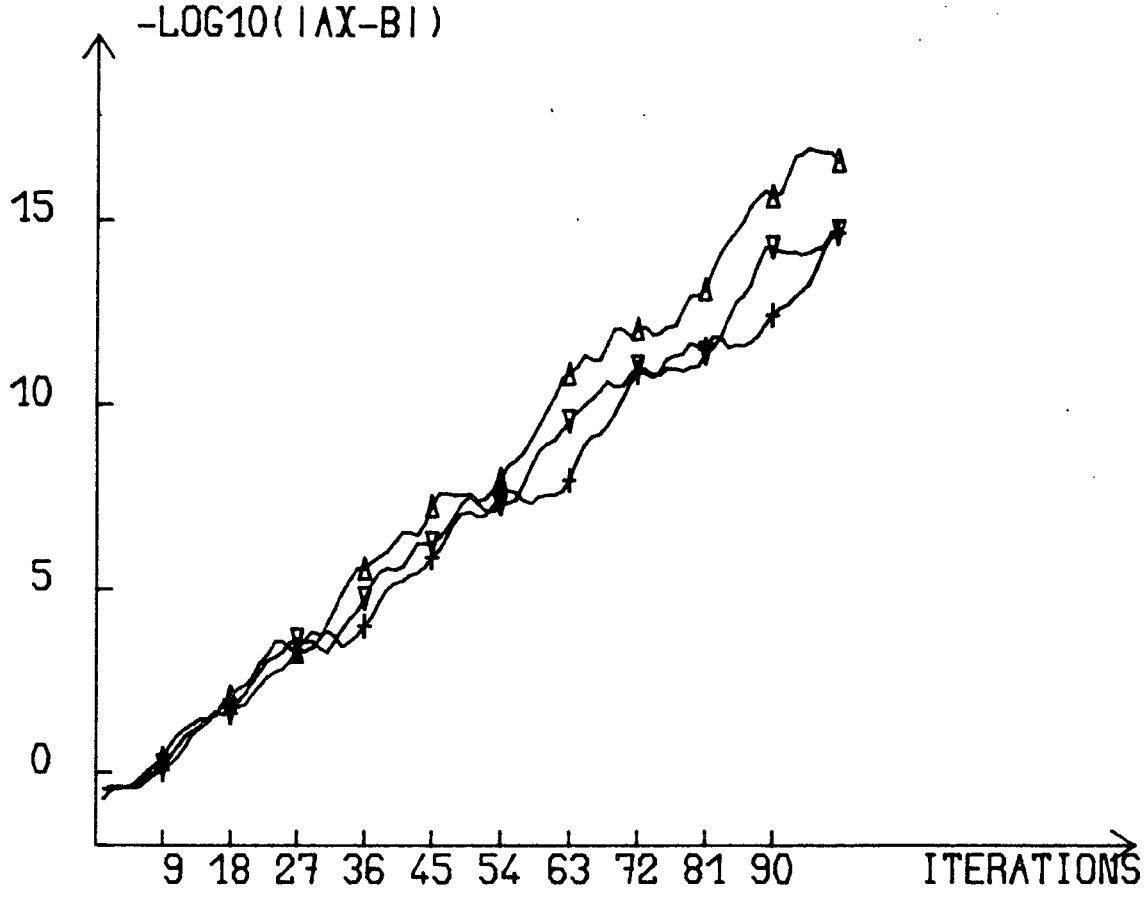


MODIFICATION DE LA CONVERGENCE PAR DISSECTION

PROBLEME DE DIRICHLET
MATRICE PRECONDITIONNEE PAR CHOLESKI INCOMPLET
DOMAINE CARRE EL.Q1 VARIABLES= 961

△ NIVEAU DE DISSECTION 0 (MONO-PROC.)
▽ NIVEAU DE DISSECTION 2 (16 PROCS.)

FIG 3.3

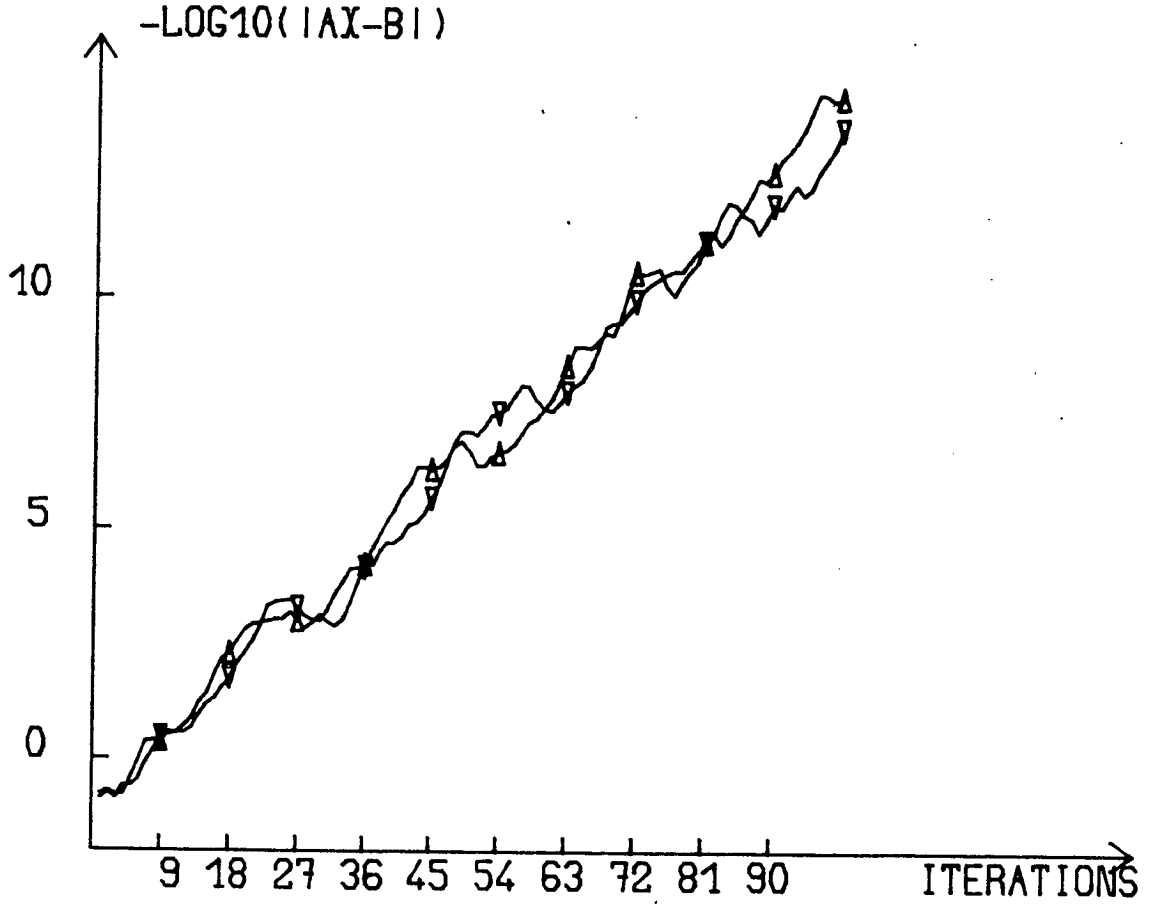


MODIFICATION DE LA CONVERGENCE PAR DISSECTION

PROBLEME DE DIRICHLET/NEUMANN
MATRICE PRECONDITIONNEE PAR CHOLESKI INCOMPLET
DOMAINE CARRE EL.Q1 VARIABLES= 1066

△ NIVEAU DE DISSECTION 0 (MONO-PROC.)
▽ NIVEAU DE DISSECTION 1 (4 PROCS.)
\+ NIVEAU DE DISSECTION 2 (16 PROCS.)

FIG 3.4

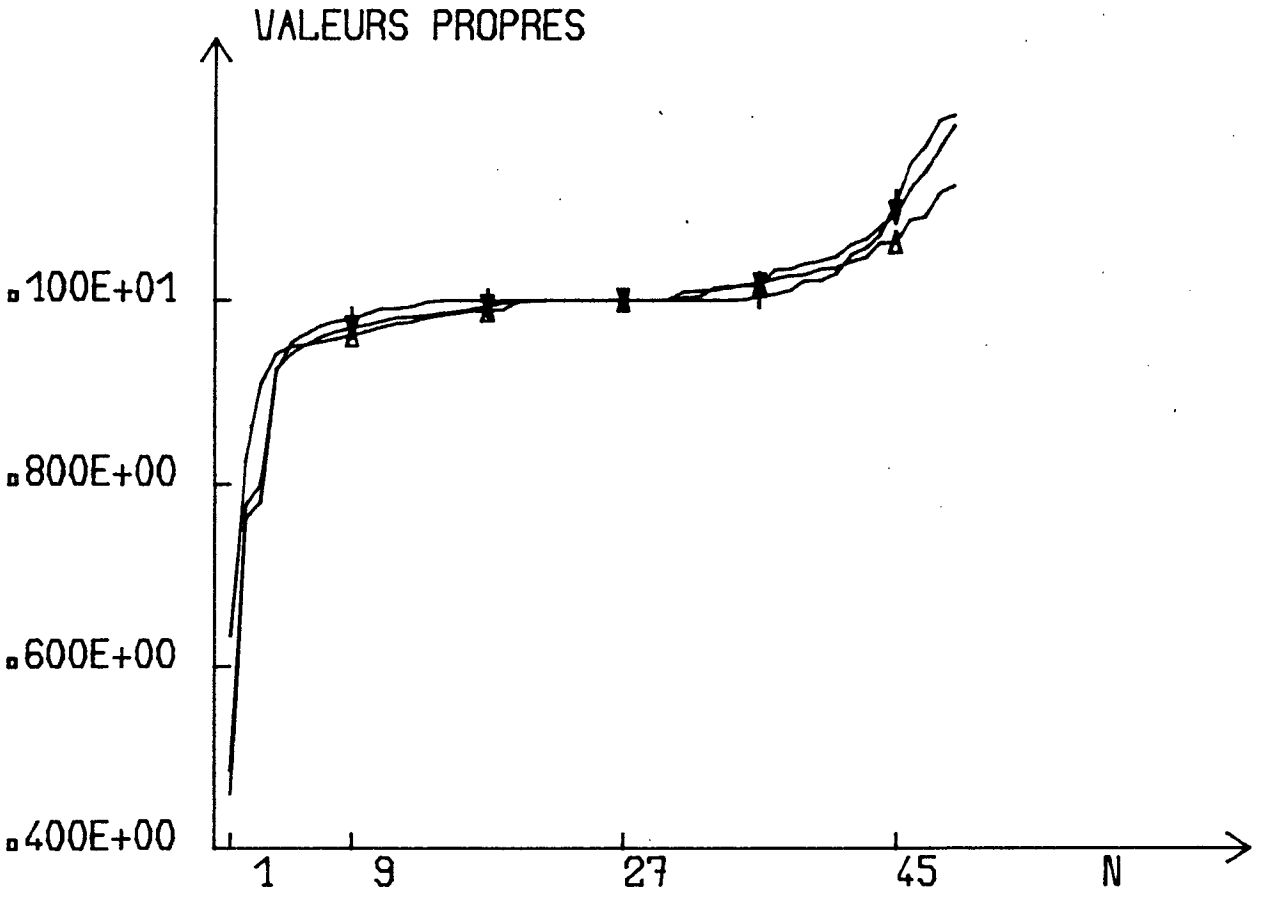


MODIFICATION DE LA CONVERGENCE PAR DISSECTION

PROBLEME DE NEUMANN
MATRICE PRECONDITIONNEE PAR CHOLESKI INCOMPLET
DOMAINE CARRE EL.Q1 VARIABLES= 1089

△ NIVEAU DE DISSECTION 0 (MONO-PROC.)
▽ NIVEAU DE DISSECTION 1 (4 PROCS.)

FIG 3.5-1

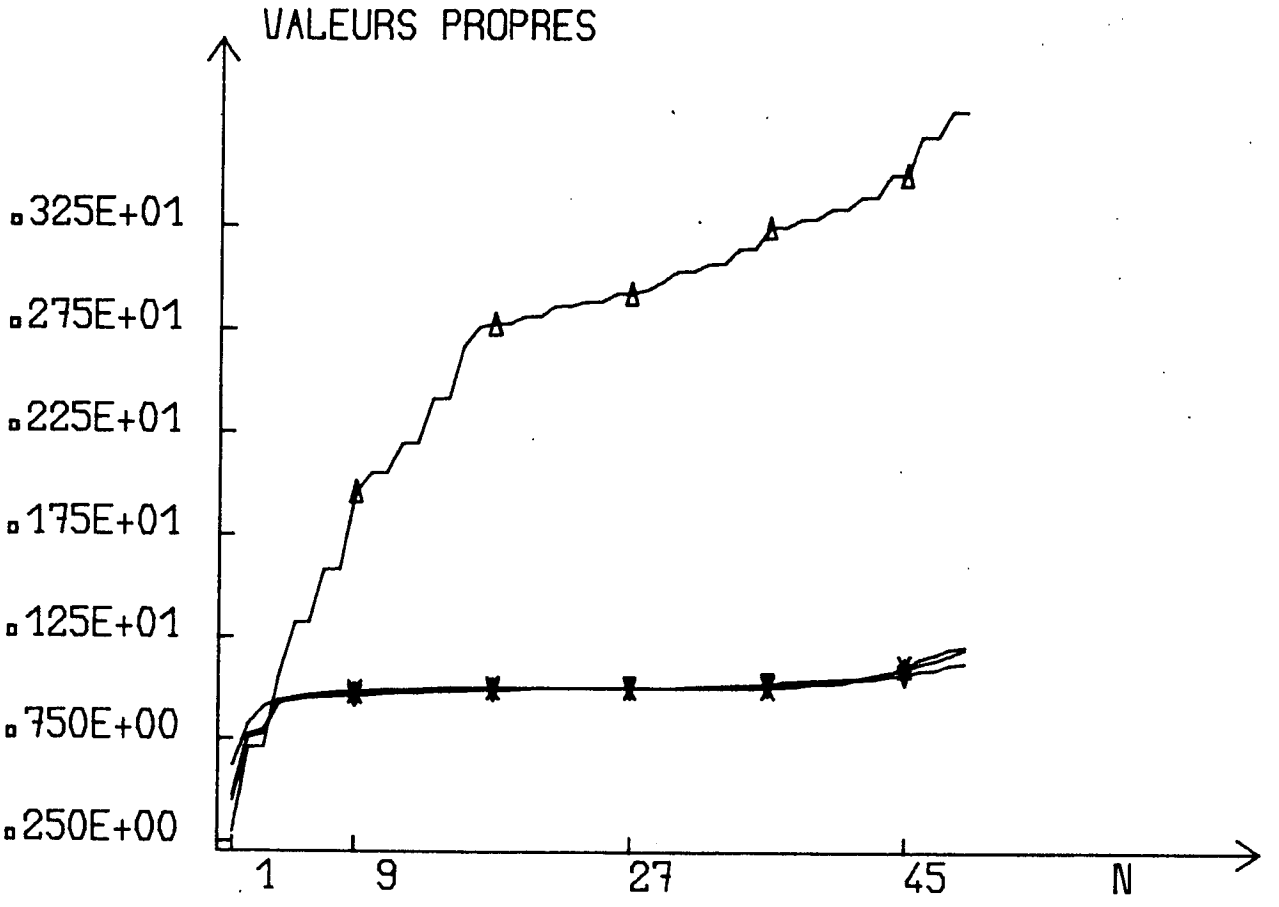


MODIFICATION DU SPECTRE PAR DISSECTION

PROBLEME DE DIRICHLET
MATRICE PRECONDITIONNEE PAR CHOLESKI INCOMPLET
DOMAINE CARRE EL.Q1 VARIABLES=49

△ NIVEAU DE DISSECTION 0 (MONO-PROC.)
▽ NIVEAU DE DISSECTION 1 (4 PROCS.)
+ NIVEAU DE DISSECTION 2 (16 PROCS.)

FIG 3.5-2



INFLUENCE DU PRECONDITIONNEMENT SUR LE SPECTRE. INFLUENCE DES PERMUTATIONS PROBLEME DE DIRICHLET (LAPLACIEN)

DOMAINE CARRE EL.Q1 VARIABLES=49

Δ PAS DE PRECONDITIONNEMENT (INDEP.PERMUTATION)
∇ NIVEAU DE DISSECTION 0 (MONO. PROC)
\+ NIVEAU DE DISSECTION 1 (4 PROCS.)
X NIVEAU DE DISSECTION 2 (16 PROCS.)

IV. CONCLUSION

L'utilisation de la dissection emboitée ne semble pas remettre en cause la vitesse de convergence de la méthode GCCI, ainsi que le montrent les essais du paragraphe III. Par contre les accélérations et rendements varient fortement en fonction des paramètres suivants :

- nombre de processeurs
- caractéristiques géométriques du domaine discrétisé (*)
- nombre d'inconnues.

Les estimations que nous donnons au paragraphe 2 montrent que cette méthode ne devient intéressante que pour des problèmes de grande taille (plus de 10^4 inconnues pour des géométries bidimensionnelles (cf. fig. 2.1 et 2.2). L'intérêt de la résolution de problèmes de telle(s) taille dans le futur proche est fondamental dans certains domaines d'applications (cf. [Y]).

Il est tout aussi important de comparer les méthodes numériques entre elles. Plusieurs méthodes de relaxation (ponctuelles, par blocs, chaotiques) et de décomposition (directions alternées...) permettent d'introduire naturellement un fort taux de parallélisme. Toutefois, dans les situations décrites par D.S. KERSHAW [V] , MEIJERINK et VAN DER VORST [K] et MANTEUFFEL [L] le bilan apparaît favorable à la méthode que nous venons d'introduire.

V. REMERCIEMENTS

L'auteur remercie L. LOTH et J. LAMINIE qui l'ont aidé à effectuer les essais numériques, ainsi que Mme BARNY pour le soin apporté à la réalisation de ce document.

(*) Le cas d'un maillage régulier a été étudié indépendamment par G. RODRIGUE et D. WOLITZER. Ces auteurs montrent qu'il est possible d'adapter la méthode du Gradient Conjugué avec préconditionnement de façon a bien utiliser le parallélisme SIMD ou vectoriel. Une autre alternative est étudiée dans [BB] .

BIBLIOGRAPHIE

[A] E. DICK GIROUS. "A large mathematical model implementation on the star-100 computers", pp. 287-298, in KUCK-LAWRIE-SAMEH [ ]

[B] R.G. VOIGT. "The influence of vector computer architecture on numerical algorithms"., pp. 229-244 in KUCK-LAWRIE-SAMEH [ ]

[C] D.J. KUCK. "A survey of parallel machine organization and programming". ACM Surveyx, Vol. 9, n° 1, pp. 29-60, 1977.

[D] R.D. RICHTMYER and K.W. MORTON. "Difference methods for initial value problems", Wiley 1957.

[E] B.L. BUZBEE, G.H. GOLUB, and J.A. HOWELL. "Vectorization for the Cray-1 of some methods for solving elliptic difference elevations"., pp. 255-271, in KUCK-LAWRIE-SAMEH

[F] A. GEORGE, W. POOLE and R. VOIGT. "Analysis of dissection algorithms for vector computers", Report n° 76-17 - ICASE-NASA Langley Research Center, Hampton.

[G] V. SCHUMANN Ed. "Computers, Fast elliptic solvers and applications".

[H] P.N. SWARZTRAUBER.

[I] A. GEORGE, W.G. POOLE and R.G. VOIGT. "Incomplete nested dissection for solving n by n grid problems" SIAM J. Numer. Anal. Vol. 15, 1978.

[J] R.J. LIPTON, D.J. ROSE and R.E. TARJAN. "Generalized nested dissection" SIAM J. Numer. Anal. Vol. 16, 1979.

[K] J.A. MEIJERINK and H.A. VAN DER VORST. "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix". Math. Comp., Vol. 31, 1977.

[L] T.A. MANTEUFFEL. "Shifted incomplete Choleski factorization", Report SAND/78/8226, Sandia Laboratories, Calif., May 1978.

[M] R.E. TARJAN. "Graph theory and Gaussian elimination" in sparse matrix computations, J.R. BUNCH and D.J. ROSE Eds., Acad. Press, New York (1976).

[N] J.W. HUANG and O. WING. "Optimal parallel triangulation of a sparse matrix", IEEE Trans. on circuits and systems, sept. 79.

[O] J.A. GEORGE. "Solution of linear systems of equations : direct methods for finite element problems", in sparse matrix Techniques, V.A. BARKER ed., Springer, (lect. notes in Maths. n° 572), 1977.

[P] G. STRANG and G.J. FIX. "An analysis of the finite element method", Prentice Hall, 1973.

[Q] R. GLOWINSKI, P.L. LIONS, J. PERIAUX. "Applications of domain decomposition techniques to the numerical solution of non linear problems in fluid dynamics", 1980.

[R] M. BONNET and G. MEURANT. "Résolution des systèmes matriciels linéaires et non linéaires ; application à une équation de diffusion non linéaire", note CEA-N-2059 et note CEA-N-2159.

[S] R.J. LIPTON and R.E. TARJAN. "A separator theorem for planar graphes", SIAM J. Appl. Math., 1979.

[T] D. HELLER. "A survey of parallel algorithms in numerical linear algebra", SIAM Review, 1976.

[U] E. GELENBE, A. LICHNEWSKY, et A. STAPHYLOPATIS. "Experience with the parallel solution of partial differential equation on a distributed computer system", à paraître.

[V] D.S. KERSHAW. "The incomplete Choleski conjugate gradient method for the iterative solution of systems in linear equations".

[W] O. AXELSSON. "A Class of iterative methods for finite element equations", comp. meth. Appl. Mech. Engag., 9, n° 2 (1976), pp. 123-138.

[X] G.M. BAUDET. "The Design and Analysis of Algorithms for Asynchronous Multiprocessors, Thèse, Université Carnegie Mellon, 1978.

[Y] D.R. CHAPMAN. "Computational aerodynamics development and outlook", AIAA Journal, Vol. 17, n° 12, pp. 1293-1313, D.J. KUCK, D.M. LAWRIE and A.H. SAMEH Ed., "High Speed Computer and Algorithm Organization", Academic Press, 1977.

[Z] G. RODRIGUE et D.WOLITZER. "Incomplete block cyclic reduction", IMACS, Montréal, 1981.

[AA] J.R. GILBERT. "Graph Separator Theorems and Sparse Gaussian Elimination", Ph. D. Thesis, Stanford, 1980.

[BB] J. ERHEL, A. LICHNEWSKY, F. THOMASSET. "Finite Element and Parallel Processing", in Symp. on Vectors and Parallel Processors, Rome, 1982. "Finite Elements and Parallel Processing", Rome 1982.

NOTATIONS :

Pour manipuler les multi-indices nous utilisons les conventions suivantes :

α : multi-indice de longueur $k = |\alpha|$ représente

$$\alpha = (\alpha_{k-1}, , \alpha_0)$$

Soient m et n avec $|\alpha| - 1 \geq m > n > 0$ alors

$$\alpha_{(m:n)} = (\alpha_m, \ldots, \alpha_n) \quad \text{et} \quad |\alpha_{(m:n)}| = m - n + 1$$

Soient $\alpha = (\alpha_m, \ldots, \alpha_0)$ et $\beta = (\beta_n, \ldots, \beta_0)$

alors

$$(\alpha, \beta) = (\alpha_m, \ldots, \alpha_0, \beta_n, \ldots, \beta_0) .$$