MODÉLISATION MATHÉMATIQUE ET ANALYSE NUMÉRIQUE

YVES ROBERT
MAURICE TCHUENTE

Résolution systolique de systèmes linéaires denses

Modélisation mathématique et analyse numérique, tome 19, n° 2 (1985), p. 315-326

<http://www.numdam.org/item?id=M2AN_1985__19_2_315_0>

MATHEMATICAL MODELLING AND NUMERICAL ANALYSIS
MODÉLISATION MATHÉMATIQUE ET ANALYSE NUMÉRIQUE

(vol 19, n° 2, 1985, p 315 a 326)

RÉSOLUTION SYSTOLIQUE DE SYSTÈMES LINÉAIRES DENSES (*)

par Yves ROBERT et Maurice TCHUENTE (1)

Communiquée par F ROBERT

Résumé — *On présente deux algorithmes systoliques pour la résolution de systèmes linéaires denses Plutôt que de procéder en une etape de triangularisation suivie de la résolution d'un système triangulaire (comme les solutions systoliques déjà connues), nous utilisons directement la méthode de diagonalisation de Jordan Malgre son coût élevé en nombre d'opérations élémentaires (ce qui explique sa non-utilisation dans le cas séquentiel), cette methode conduit à des réseaux performants, et s'avère être la plus adaptée au calcul parallèle*

Abstract — *Two systolic arrays are proposed for the solution of dense linear systems Instead of triangularising and then solving a triangular system (as suggested in previous systolic solutions), the Jordan diagonalisation method is used In view of the better performances of the resulting networks, the Jordan method appears to be well-suited to parallel computation (contrarily to its high cost on sequential machines)*

1. INTRODUCTION

Les algorithmes numériques actuellement utilisés pour résoudre les problèmes qui se posent lors de l'étude en simulation du comportement de réacteurs nucléaires ou d'écoulements hydrodynamiques — pour ne reprendre que deux exemples dus à [16] — conduisent à une saturation quasi totale des machines séquentielles disponibles. Il est communément admis qu'il faut absolument, pour répondre aux besoins toujours croissants du Calcul Scientifique, gagner un facteur de cent au moins sur les vitesses de traitement, au cours de la prochaine décennie [3] : le parallélisme est à l'ordre du jour. Une première approche consiste à programmer au mieux les algorithmes séquentiels existants sur des machines vectorielles comme le CRAY I ou le CYBER

(*) Reçu en janvier 1984
(1) CNRS, Laboratoire TIM3, IMAG, BP 68, 38402 Saint Martin d'Hères Cedex.

205. Une autre démarche reconsidère les algorithmes usuels pour leur donner une structure à fort degré de parallélisme, adaptée aux exigences des nouvelles architectures d'ordinateurs (SIMD, MIMD, processeurs intégrés spécialisés, voir [6] pour cette classification).

La première approche est plus directement orientée vers la pratique (puisque les machines vectorielles sont d'ores et déjà largement répandues), tandis que la deuxième reste encore principalement du domaine de la recherche (il existe cependant des réalisations, comme l'ordinateur HEP de Denelcor, Inc de type MIMD, ou le processeur intégré systolique de l'Université de Carnegie Mellon).

Nous nous intéressons ici à la résolution de problèmes linéaires denses. En ce qui concerne des architectures de type SIMD ou MIMD, cette question a déjà fait l'objet de nombreuses études, comme en témoignent les abondantes bibliographies des articles de synthèse de Sameh [16] et Heller [10]. L'apparition de solutions intégrées est plus récente (1978), puisque c'est seulement depuis quelques années que les progrès de la technologie VLSI permettent d'entrevoir la réalisation de cartes spécialisées que l'on adjoindra à un calculateur dit « hôte » pour accélérer le traitement de certains types de problèmes comportant un grand volume de calculs.

Le modèle systolique introduit par Kung et Leiserson [13] permet de concevoir des circuits intégrés spécialisés performants ayant un facteur de régularité et de modularité élevé. En un mot, il s'agit de structures-réseaux composés de processeurs (ou cellules) simples, régulièrement agencés, et combinant les deux types classiques de parallélisme :

— dans le temps (pipe-line),

— dans l'espace : plusieurs calculs sont effectués simultanément sur le réseau.

Ces caractéristiques conduisent à des calculs en temps réel (i.e. où les sorties sont délivrées au même rythme que les entrées). Kung décrit en détail dans [12] les avantages du modèle systolique, et on peut se référer à [2] pour une introduction et un exemple d'application.

Dans le paragraphe suivant, nous rappelons brièvement les méthodes de triangularisation et de diagonalisation pour la résolution de systèmes linéaires denses, qui nécessitent $0(n^3)$ opérations, et comment ces méthodes ont été parallélisées sur des architectures MIMD comportant $0(n)$ processeurs pour obtenir une résolution en $0(n^2)$ unités de temps. Nous présentons alors de nouvelles architectures systoliques composées de $0(n^2)$ cellules et réalisant des temps de calcul linéaires, en les comparant aux architectures systoliques déjà connues (Ahmed *et al.* [1]; Gentleman et Kung [9]; Kung et Leiserson [13]).

2. RÉSOLUTION EN PARALLÈLE DE SYSTÈMES LINÉAIRES

Soit A une matrice carrée d'ordre n et $Ax = b$ un système linéaire à résoudre. La plupart des méthodes directes sont des algorithmes séquentiels qui procèdent en deux étapes : triangularisation de la matrice A bordée du vecteur colonne b par prémultiplication par des matrices convenables, puis résolution du système triangulaire obtenu. Formellement, on peut écrire :

Programme de triangularisation du système

Soit $A' = (A, b)$ matrice de taille $n \times (n + 1)$
pour $i := 1$ à n faire
pour $j := i + 1$ à n faire la tâche T_{ij} :

$$\begin{pmatrix}\text{ligne } i\\ \text{ligne } j\end{pmatrix} := M_{ij} \begin{pmatrix}\text{ligne } i\\ \text{ligne } j\end{pmatrix}$$

où M_{ij} est déterminée de manière à annuler le coefficient a_{ji}.

Par exemple, en choisissant

$$(*) \quad M_{ij} = \begin{pmatrix} 1 & 0 \\ l_{ji} & 1 \end{pmatrix}, \quad l_{ji} = -a_{ji}/a_{ii}$$

on retrouve la méthode de Gauss sans pivotage. En choisissant pour M_{ij} une matrice de rotation plane, on obtient la méthode de Givens. Toutes ces méthodes ont un coût de $n^3/3$ opérations élémentaires, mais il faut résoudre ensuite un système triangulaire.

Une autre possibilité est de diagonaliser A directement, en annulant à l'étape i tous les coefficients a_{ji}, $j \neq i$. Le nombre d'opérations élémentaires est alors $n^3/2$, et la solution du système est directement obtenue en effectuant n divisions. Le choix précédent (*) pour les M_{ij} conduit à la méthode de Jordan.

Tous ces algorithmes peuvent être parallélisés en recherchant le maximum de tâches indépendantes, que l'on mènera simultanément. Les contraintes de séquencement imposées par le programme précédent sont les suivantes :

(R) T_{ij} précède $T_{i+1,j}$ ($j > i$ pour la triangularisation, et $j \neq i$ pour la diagonalisation).

La fermeture transitive de la relation (R) conduit au graphe de parallélisme maximal. Par exemple, $T_{12}, T_{13}, \ldots, T_{1n}$ peuvent être exécutées en parallèle. Pour la triangularisation d'une matrice de taille n, Sameh et Kuck [17] intro-

duisent un algorithme parallèle (asymptotiquement optimal [4]) s'exécutant en temps linéaire sur une machine SIMD comportant $[n/2]$ processeurs, tandis que Lord *et al.* [14] décrivent une parallélisation optimale utilisant le même nombre de processeurs pour une architecture MIMD. Enfin, des réseaux systoliques ont été proposés par Gentleman et Kung [9], Ahmed *et al.* [1] pour résoudre ce problème, et seront décrits en détail ci-dessous.

3. RÉSEAUX SYSTOLIQUES POUR LA MÉTHODE DE JORDAN

Pour définir une architecture systolique réalisant la triangularisation ou la diagonalisation des matrices, il faut se doter de trois types de cellules :

— des cellules de génération, capables de calculer les coefficients des matrices M_{ij},

— des cellules de propagation, capables de recevoir, combiner et transmettre les informations en provenance des cellules voisines,

— éventuellement, des cellules de retard.

D'autre part, comme pour les machines SIMD ou MIMD, il faut définir un séquencement pour l'exécution des tâches élémentaires, et organiser le croisement des flots de données régis par le séquencement.

Ainsi, pour la triangularisation d'une matrice A de taille n, bordée du vecteur b dont les composantes sont notées $b_i = a_{i,n+1}$, Gentleman et Kung [9] définissent un réseau comportant $n(n+3)/2$ cellules, dont n cellules de génération (voir *fig.* 1). La matrice A est introduite par la partie supérieure du réseau, un port d'entrée recevant tous les éléments d'une même colonne. Lorsqu'à l'instant i, a_{1i} ($1 <= i <= n+1$) arrive dans le réseau, il est stocké dans le registre interne de la cellule qui le reçoit en entrée (i-ième cellule de la première ligne); l'instant d'après, cette cellule dispose de a_{1i}, a_{2i} comme données, et aussi de la matrice M_{21} générée par la cellule recevant a_{11} : elle dispose ainsi de tous les éléments pour modifier a_{2i} suivant la transformation M_{21} conduisant à l'annulation de a_{21}. Le même raisonnement s'applique pour les M_{i1}, $i >= 3$. De manière analogue, les cellules de la ligne j permettront l'annulation des coefficients a_{kj}, $k > j$, et la modification correspondante des éléments a_{1m}, 1 et $m >= j+1$.

A la fin de cette étape (au bout de $3n$ unités de temps, une unité correspondant au temps de cycle maximal d'une cellule), on obtient un système triangulaire $Tx = c$, la matrice $S = (T, c)$ est stockée dans le réseau selon le schéma de la figure 2. Gentleman et Kung proposent alors d'utiliser un réseau de n cellules défini dans [13] pour la résolution des systèmes triangulaires. Cette

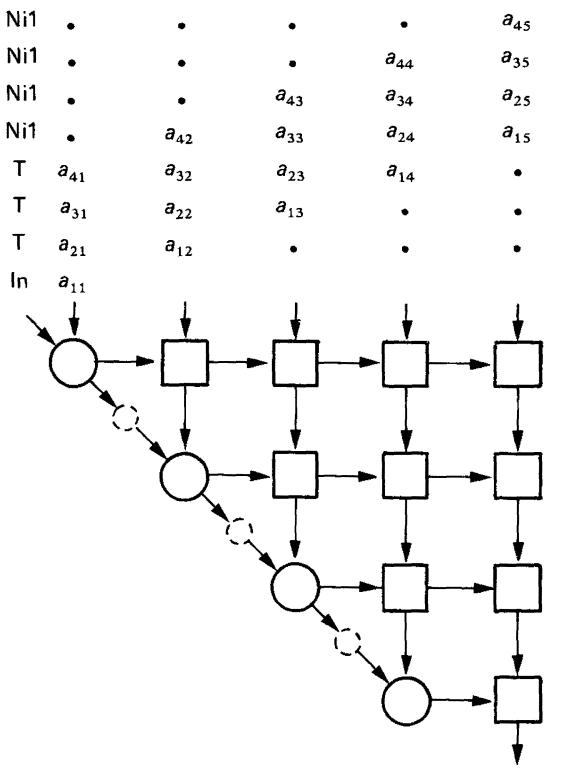


Figure 1*a*. — Structure générale du réseau.

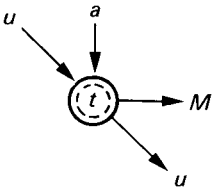
solution nécessite en plus l'emploi de $n(n-1)/2$ cellules de retard, et un temps de calcul additionnel de $2n$ tops d'horloge. Nous allons ici résoudre directement le système triangulaire obtenu par une deuxième phase d'activation du réseau.

On commence par envoyer un signal à la cellule contenant s_{11} : à la réception de ce signal, cette cellule génère la matrice

$$\begin{pmatrix} 0 & 0 \\ 1/s_{11} & 0 \end{pmatrix}.$$

Si les cellules de la première ligne continuent à fonctionner comme dans la phase 1, alors chaque élément s_{1i} sera divisé par s_{11} puis envoyé dans la cellule voisine du bas. Notons $s'_{1i} = s_{1i}/s_{11}$; étant donné que s'_{1i+1} est calculé un top d'horloge après s'_{1i}, tout se passe comme si on avait la situation de la figure 3. En conséquence, si toutes les cellules de génération fonctionnent

Cellule de génération

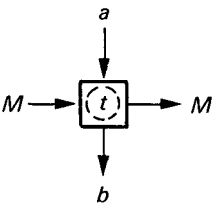


$a, u \in R$; $M \in M_{2\times 2}(R)$; $u \in \{$ Nil, In, Ga, Pv, Gi $\}$ où In, Ga, Pv, Gi sont respectivement les abréviations de Initialisation, Gauss, Pivots-voisins et Givens :

si u = Nil alors $M = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ sinon
si u = In alors $M = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ sinon
si u = Ga alors $M = \begin{pmatrix} 1 & 0 \\ -a/t & 1 \end{pmatrix}$ sinon
si u = Pv alors
début si $abs(a) > abs(t)$ alors $M = \begin{pmatrix} 1 & -t/a \\ 0 & 1 \end{pmatrix}$ sinon
si $a = t = 0$ alors $M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ sinon
$M = \begin{pmatrix} 1 & 0 \\ -a/t & 1 \end{pmatrix}$
fin
sinon
si u = Gi alors
début si $a = t = 0$ alors $M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ sinon
$M = \begin{pmatrix} a/a^2 + t^2 & -t/a^2 + t^2 \\ t/a^2 + t^2 & a/a^2 + t^2 \end{pmatrix}$
fin,
$t = m_{11}.t + m_{12}.a$,

Cellule de combinaison

$$\begin{pmatrix} t \\ b \end{pmatrix} := M.\begin{pmatrix} t \\ a \end{pmatrix}.$$



Cellule de retard

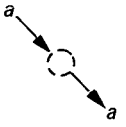


Figure 1*b*. — Structure des cellules.

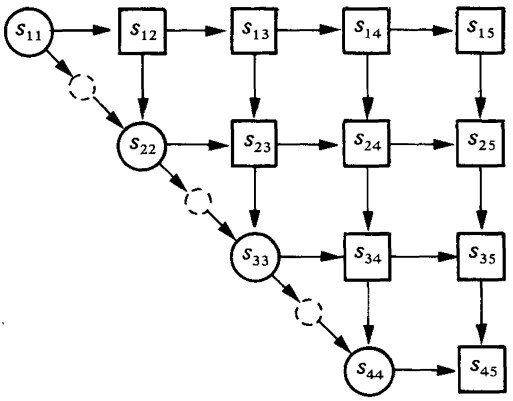


Figure 2.

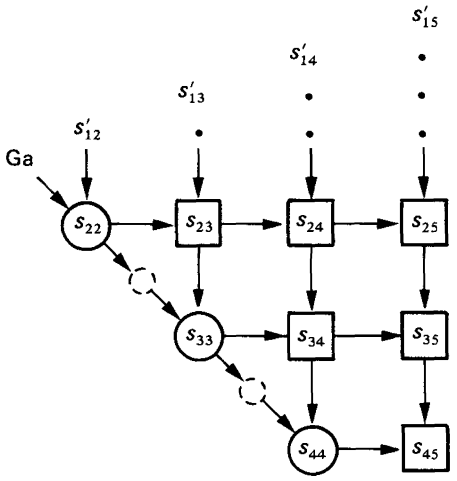


Figure 3.

comme dans la phase 1, mais en générant cette fois-ci les matrices 2×2 correspondant exclusivement à l'élimination de Gauss, alors on peut éliminer successivement s'_{12}, s'_{13}, ..., s'_{1n} ; à la sortie du réseau, on aura $s'_{1n+1} = x_1$. Le même raisonnement est valable pour toutes les autres lignes. De plus :

— au cours de cette deuxième phase, les traitements correspondant aux lignes 1, 2, ..., n peuvent être pipelinés,

— la phase 1 et la phase 2 peuvent également être pipelinées.

On obtient finalement le réseau de la figure 4. Le temps total de calcul de la solution x du système linéaire est de $4\,n$ tops d'horloge (au lieu de $5\,n$ pour

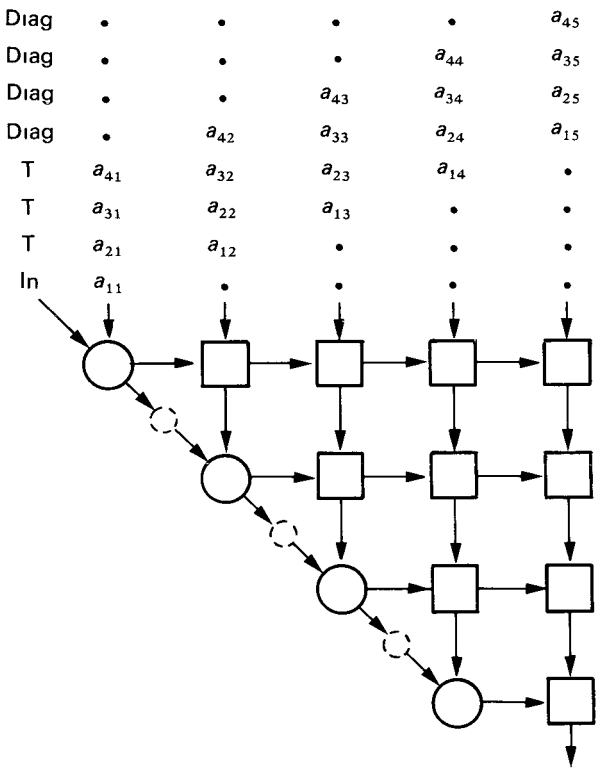
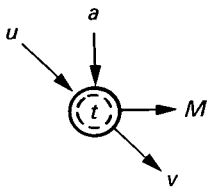


Figure 4*a*. — Structure générale du réseau.

Cellule de génération

$u \in$ { Nil, In, Ga, Gi, Pv, Diag }. Diag est le signal de diagonalisation. La cellule exécute en plus l'instruction suivante :



si u = Diag alors
début si $t \neq 0$ alors
début $t = 0$,
v = Nil,
$M = \begin{pmatrix} 1/t & 0 \\ 0 & 0 \end{pmatrix}$
fin
sinon
v = Diag
fin
sinon
$v = u$,

Figure 4*b*. — Structure de la nouvelle cellule de génération.

[9]) et nous n'avons utilisé que les $n(n + 3)/2$ cellules du réseau de triangularisation.

A condition de doubler le nombre de cellules, il est possible de diagonaliser la matrice A en seulement $4\,n$ tops d'horloge, en employant directement la méthode de Jordan. Compte tenu du fait que dans l'algorithme, toutes les colonnes de la matrice jouent maintenant des rôles similaires (on doit annuler n coefficients dans chacune d'elles), l'idée-clé est de préparer un réseau rectangulaire de $n(n + 2)$ cellules (dont n de génération et n de retard) dans lequel les informations correspondant aux matrices M_{ij} seront stockées dans les cellules, alors que les coefficients de la matrice circuleront ligne par ligne dans le réseau. En ne conservant que la partie triangulaire inférieure de ce nouveau réseau, on obtient le réseau de triangularisation que présentent Ahmed *et al.* dans [1].

Cette deuxième solution pour diagonaliser A est décrite figure 5 (noter que les éléments de la matrice diagonale obtenue sont stockés dans la dernière ligne du réseau). On vérifie bien que le système $Ax = b$ est maintenant résolu en $4\,n$ tops d'horloge, le prix à payer pour l'emploi de cette méthode étant le doublement de la surface.

Remarques

A. *Choix de la méthode de triangularisation*

Si la matrice A n'est pas symétrique définie positive ou à diagonale dominante, il peut être préférable, pour des raisons de stabilité, d'opter pour la méthode de Givens. Les cellules de propagation doivent alors calculer des racines carrées, et leur temps de cycle devient nettement supérieur à celui des autres cellules. Deux remèdes à cette situation peuvent être envisagés :

— l'emploi de la méthode de Givens sans racines carrées, due à Gentleman [8]. Gentleman et Kung [9] discutent cette solution, qui complique le réseau :

— l'emploi de processeurs spécialisés, de type Cordic [1], pour lesquels multiplication et extraction de racine carrée sont du même ordre de complexité.

Notons que le dernier réseau proposé peut réaliser sans difficulté la triangularisation par la méthode de Givens : il suffit pour cela de programmer les cellules du réseau, celles de la partie supérieure réalisant des transformations de Givens et celles de la partie inférieure des transformations de Gauss.

B. *Résolution de plusieurs systèmes*

Si on a p systèmes linéaires $Ax_i = b_i$, $1 <= i <= p$, à résoudre avec la même matrice A :

— pour la première solution, il est nécessaire d'ajouter $p - 1$ colonnes à la droite du réseau, pour un temps total de résolution de $4\,n + p$ tops d'horloge,

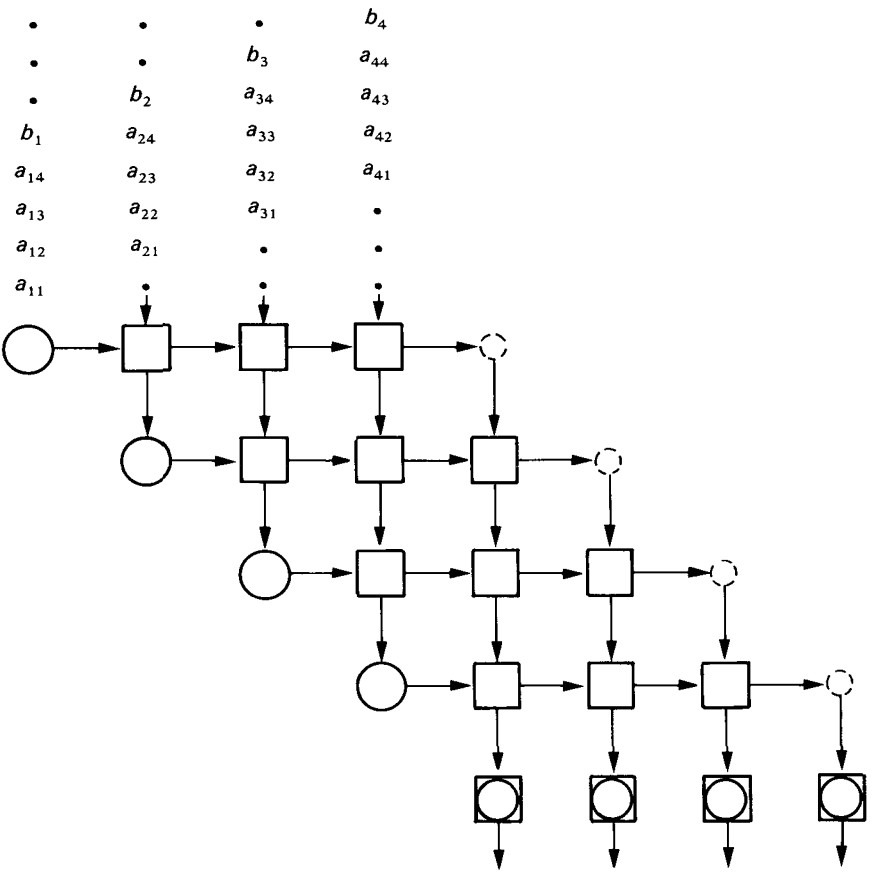


Figure 5*a*. — Structure générale du réseau.

Cellule de division

t est initialisé à 0.

si $t = 0$ alors $t := x$
sinon
$y := x/t$;

Figure 5*b*. — Structure de la cellule de division.

— pour la deuxième solution, il suffit de compléter chaque ligne j entrant dans le réseau par les j-ièmes composantes des vecteurs $b_2, ..., b_p$: le nombre de cellules reste inchangé, et le temps de calcul est encore de $4n + p$. Cette solution s'avère donc préférable si p est grand en particulier, le calcul de l'inverse d'une matrice peut se faire en temps $5n$ sur le réseau de la figure 5.

C. *Cas où la matrice du système est symétrique*

Dans le cas des matrices symétriques, Delosme [5] propose un réseau de $n^2/2$ cellules pour résoudre le système $Ax = b$; le facteur de Choleski L de la matrice $A = L^t L$ est tout d'abord déterminé en $2n$ tops d'horloge (par utilisation d'une extension de l'algorithme de Schur-Levinson). Toutes les composantes du produit $L^{-1} b = c$ sont alors disponibles au top suivant, et renvoyées dans le réseau qui délivre le vecteur solution x après n tops d'horloge supplémentaires; le système est ainsi résolu en temps total $3n + 1$. Ce réseau réalise donc le meilleur compromis temps-surface dans le cas symétrique.

D. *Cas des matrices bandes*

Plusieurs réseaux ont été proposés pour la triangularisation des matrices bandes ($a_{ij} = 0$ si $i - j > p$ ou $j - i > q$) : voir [9, 11, 13, 15]. Tous ont l'intérêt d'utiliser un nombre de cellules dépendant uniquement de la largeur $w = p + q - 1$ de la bande, et non de la taille de la matrice. Aucun ne semble cependant pouvoir se prêter à la résolution directe d'un système linéaire à structure bande.

CONCLUSION

Pour la résolution des systèmes linéaires, l'emploi de la méthode de Jordan conduit à des réseaux systoliques plus performants que ceux obtenus par l'utilisation d'une méthode de triangularisation suivie de la résolution d'un système triangulaire. Cette conclusion s'inscrit à l'opposé du critère séquentiel qui au vu du nombre d'opérations, conduit à préférer la deuxième méthode (pour un coût de l'ordre de $n^3/3$) à la première (nécessitant $n^3/2$ transformations élémentaires); elle illustre bien le fait que les critères d'évaluation des algorithmes, traditionnellement basés sur les opérations arithmétiques et ignorant les temps de communication et la structure des données, doivent être révisés en calcul parallèle. Dans le domaine particulier de l'implantation VLSI, les bons algorithmes ne sont pas ceux qui nécessitent le moins de calculs : comme il est souligné dans [7], les calculs sont peu coûteux en VLSI, ce sont les communications qui déterminent les performances.

REFFRENCES

[1] H. M. AHMED, J. M. DELOSME, M. MORF, *Highly concurrent computing structures for matrix arithmetic and signal processing.* Computer magazine, January 1982, pp. 65-82.

[2] F. ANDRÉ, P. FRISON, P. QUINTON, *Algorithmes systoliques : de la théorie à la pratique,* Rapport de Recherche INRIA n° 214, 1983.

[3] A. BOSSAVIT, *Préface des actes du colloque AFCET-GAMNI-ISINA,* 17-18 mars 1983, Paris, Bulletin de la direction des études et recherches EDF, série C, vol. 1, 1983.

[4] M. COSNARD, Y. ROBERT, *Complexité de la factorisation QR en parallèle,* C. R. Acad. Sc. Paris, t. 297, Série I, pp. 137-139 (septembre 1983).

[5] J. M. DELOSME, *Algorithms for finite shift-rank processes,* Ph. D., Technical Report M735-22, September 1982, Stanford Electronics Laboratories.

[6] M. FLYNN, *Some computer organisations and their effectiveness,* IEEE Trans. on Computers C21, 9 (1972), pp. 948-960.

[7] M. J. FOSTER, H. T. KUNG, *The design of special-purpose VLSI chips,* IEEE Computer 13, 1 (January 1980), pp. 26-40.

[8] W. M. GENTLEMAN, *Least squares computation by Givens transformations without square roots,* J. Inst. Math. Appl. 12 (1973) pp. 329-336.

[9] W. M. GENTLEMAN, H. T. KUNG, *Matrix triangularisation by systolic arrays,* Proc. SPIE 298, Real-time Signal Processing IV, San Diego, California, 1981.

[10] D. HELLER, *A survey of parallel algorithms in numerical linear algebra,* Siam Review 20, pp. 740-777, 1978.

[11] D. HELLER, I. IPSEN, *Systolic networks for orthogonal equivalence transformations and their applications,* Proc. 1982 Conf. Advanced Research in VLSI, pp. 113-122, MIT 1982.

[12] H. T. KUNG, *Why systolic architectures,* IEEE Computer 15, 1 (January 1982), pp. 37-46.

[13] H. T. KUNG, C. E. LEISERSON, *Systolic Arrays for (VLSI), in the proceedings of the Symposium on sparse matrix computations and their applications,* Knoxville, 1978.

[14] R. E. LORD, J. S. KOWALIK, S. P. KUMAR, *Solving linear algebraic equations on an MIMD computer,* J. ACM 30 (1), pp. 103-117, 1983.

[15] L. MELKEMI, M. TCHUENTE, *Systolic arrays for connectivity and triangularisation problems,* to appear in Proc. « Dynamical Systems and Cellular Automata », J. Demongeot, E. Coles et M. Tchuente eds., Academic Press, 1985.

[16] A. SAMEH, *Numerical parallel algorithms — a survey,* in « High Speed Computer and Algorithm Organization », D. Kuck, D. Lawrie and A. Sameh eds., pp. 207-228, Academic Press, 1977.

[17] A. SAMEH, D. KUCK, *On stable parallel system solvers,* J. ACM 25 (1), pp. 81-91, 1978.