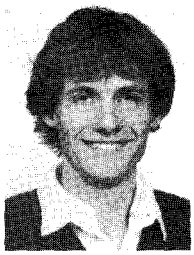
Architecture systolique de systèmes adaptatifs

Systolic implementation of adaptive systems



Pierre COMON

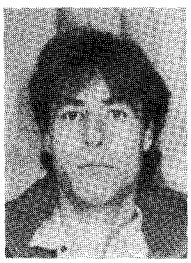
Laboratoire CEPHAG : Centre d'Études des Phénomènes Aléatoires et Géophysique de Grenoble, ENSIEG, BP n° 46, 38402 SAINT-MARTIN D'HERES Cédex.
Laboratory Information Systems, Department Electrical Engineering, Stanford University, STANFORD, CA 94305, USA.

Pierre Comon est né à Paris en 1959. Il a reçu le diplôme d'ingénieur de l'ENSIEG en 1982 et soutenu une Thèse de Doctorat en 1985. De 1982 à 1985, il a travaillé dans le département Aérospatial de la société Crouzet (Valence). Depuis 1981, il appartient au Laboratoire CEPHAG. Il est actuellement détaché au Information Systems Laboratory de l'université de Stanford, USA. Ces centres d'intérêt regroupent le filtrage adaptatif multivariable, l'élimination de bruit, l'analyse haute résolution et les réseaux systoliques.

Yves ROBERT

CNRS, Laboratoire TIM3 : Techniques pour l'Informatique, les Mathématiques, la Microélectronique et la Microscopie quantitative, Institut National Polytechnique de Grenoble, 38031 GRENOBLE Cédex.

Yves Robert, ancien élève de l'École Normale Supérieure de l'Enseignement Technique, agrégé de Mathématiques (1980), a obtenu une Thèse de troisième cycle en Mathématiques Appliquées à l'Université de Grenoble. Il a soutenu en 1986, une Thèse d'État sur l'algorithmique parallèle pour les réseaux d'automates, les architectures systoliques et les machines SIMD et MIMD. Ses recherches actuelles portent sur les algorithmes et architectures parallèles, systoliques et VLSI pour l'algèbre numérique linéaire et le traitement du signal.



Denis TRYSTRAM

Laboratoire CREDI (Création Réalisation Étude et Développement Informatiques), École Centrale de Paris, Grande-Voie-des-Vignes, 92295 CHÂTENAY-MALABRY Cédex.

Denis Trystram est ingénieur de l'ENSIMAG, il a soutenu une thèse de Docteur-Ingénieur en 1984 au laboratoire IMAG, sur l'algorithmique numérique des grands systèmes linéaires. Il est actuellement Assistant en Informatique à l'École Centrale de Paris où il dirige un groupe de travail sur le calcul parallèle.

RÉSUMÉ

Nous nous intéressons à la mise en œuvre sur des architectures VLSI hautement parallèles (réseaux systoliques) du calcul des projections intervenant dans le traitement des signaux numériques, et plus généralement dans les systèmes adaptatifs. Après avoir montré comment se pose ce genre de problème en traitement adaptatif du signal nous passons en revue diverses architectures systoliques (connues ou originales) qui permettent une résolution en temps réel.

MOTS CLÉS

Filtrage adaptatif, estimation de paramètres, projections, algorithmique numérique, réseaux systoliques, calcul parallèle.

SUMMARY

We are interested in the systolic computation of projection operators entering digital signal processing, or more generally adaptive systems. We first show how such computations arise when dealing with adaptive signal processing. Then we propose an overview of various systolic arrays (including original ones) achieving real time computations.

KEY WORDS

Adaptive filtering, parameter estimation, projections, numerical algorithms, systolic arrays, parallel computing.

Introduction

Le modèle systolique a été introduit en 1978 par Kung et Leiserson [20] pour répondre à la demande sans cesse croissante d'une plus grande puissance de calcul à moindre frais. Il s'est révélé être un outil très efficace pour la conception de processeurs intégrés spécialisés (citons pour référence [14, 19, 21, 22] parmi d'autres).

En un mot, une architecture systolique est agencée en forme de réseau composé d'un grand nombre de cellules élémentaires identiques et localement interconnectées. Chaque cellule reçoit des données provenant des cellules voisines, effectue un calcul simple, puis transmet les résultats, toujours aux cellules voisines, un temps de cycle plus tard. Pour fixer un ordre de grandeur, disons que chaque cellule a la complexité, au plus, d'un petit microprocesseur.

Les cellules évoluent en parallèle, en principe sous le contrôle d'une horloge globale : plusieurs calculs sont effectués simultanément sur le réseau et on peut résoudre plusieurs instances du même problème à la suite.

La dénomination « systolique » provient d'une analogie entre la circulation des flots de données dans le réseau et celle du sang dans le corps humain, l'horloge qui assure la synchronisation globale constituant le « cœur » du système.

Nous nous intéressons dans cet article à la réalisation systolique du calcul des projections intervenant dans le traitement des signaux numériques. Comme nous allons le montrer dans la première section, le problème général peut être posé de la manière suivante :

Calculer successivement $A_i^{-1} B_i$ pour $i = 1, 2, \ldots$, où les matrices A_i et B_i, respectivement de tailles $n \times n$ et $n \times p$, sont fonction de A_{i-1} et B_{i-1}.

Nous proposons tout d'abord, dans la section 2, une étude générale pour calculer $A^{-1} B$ par décomposition en sous-problèmes matriciels élémentaires. Nous faisons un tour d'horizon des meilleurs réseaux existants qui conduisent au produit $A^{-1} B$ par combinaisons des réseaux fondamentaux, après quoi nous proposons un réseau basé sur la décomposition de Gauss-Jordan qui permet de calculer directement $A^{-1} B$.

Enfin, nous donnons dans la section 3 la description de deux réseaux systoliques qui permettent de tenir compte des éléments de A_{i-1} et B_{i-1} dans le calcul de $A_i^{-1} B_i$. Le premier est basé sur la formule de Woodbury et le second est un réseau original qui inclut la mise à jour des matrices.

1. Projections en traitement du signal

Un système adaptatif peut être défini par un opérateur optimal, construit à partir d'un modèle d'observation et d'un critère d'optimisation (voir *fig.* 1). Un grand nombre d'opérateurs optimaux sont constitués de projections, notamment lorsque le critère est quadratique et le modèle linéaire. Dans le cadre de la modélisation et du traitement de signaux numériques observés sur une durée finie, les projections prennent une forme simple. Nous allons développer dans cette section quelques aspects de l'utilisation des projections dans le domaine du traitement adaptatif du signal.

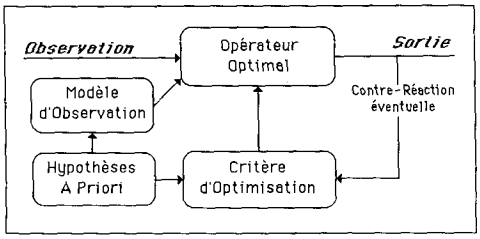


Fig. 1. – Schématisation d'un traitement adaptatif.

1.1. Projections

Considérons l'espace hilbertien L_p^2 des vecteurs déterministes de dimension p, muni du produit hermitien : $\langle x, y \rangle = x^t G^{-1} y$, où G est une matrice hermitienne $p \times p$ définie positive et où x^t désigne le vecteur transposé conjugué de x. Soit Y une matrice $p \times n$ formée de n vecteurs colonnes y_j. Cette matrice définit un sous-espace $L^2_{D(Y)}$ de dimension $D(Y) \leqq p$ engendré par la famille $\{y_1, \ldots, y_n\}$. En particulier, si Y est de rang plein, $D(Y) = \inf(p, n)$. La projection d'un vecteur x de L_p^2 sur $L^2_{D(Y)}$ est caractérisée simplement par le vecteur de L_n^2 :

$$(1) \qquad F_D(Y, x) = (Y^t G^{-1} Y)^{-1} Y^t G^{-1} x$$

où A^{-1} désigne l'inverse généralisée de A lorsque A est singulière [10]. En d'autres termes $F_D(Y, x)$ définit ici une application de L_p^2 dans L_n^2. Ce type d'expression est constamment rencontré; citons par exemple l'estimation par les moindres carrés généralisés de paramètres déterministes dans un modèle d'observation linéaire [28, 31] : coefficients de modèles AR [31, 3, 11], analyse spectrale paramétrique [12, 17], analyse haute résolution spatiale ou spectrale [12], ... Une

forme voisine est rencontrée si la projection est immergée dans L_p^2; le projecteur, que nous noterons $F_p(Y)$, est alors une application de L_p^2 dans lui-même :

$$(2)\quad F_p(Y)\,x = Y F_D(Y, x) = Y (Y^t G^{-1} Y)^{-1} Y^t G^{-1} x$$

Remarquons qu'alors le projecteur sur l'espace orthogonal s'écrit :

$$(3)\qquad F_p^0(Y) = Y\,[I - (Y^t G^{-1} Y)^{-1} Y^t G^{-1}]$$

Ces deux dernières expressions sont à la base de la théorie des filtres en treillis [17, 8].

Les projections prennent une forme similaire pour des signaux aléatoires. En effet, soit H_p^2 l'espace hilbertien des vecteurs aléatoires de dimension p, muni du produit hermitien : $\langle x, y \rangle = E\{x^t G^{-1} y\}$. La projection d'un vecteur x de H_p^2 sur le sous-espace engendré par la matrice aléatoire Y, revêt la forme suivante :

$$(4)\quad F_p(Y)\,x = Y F_D(Y, x) = Y E\{Y^t G^{-1} Y\}^{-1} E\{Y^t G^{-1} x\}$$

$F_D(Y, x)$ est ici un vecteur déterministe de L_n^2 représentant les coefficients de x dans sa décomposition sur la famille génératrice $\{y_1, \ldots, y_n\}$. Par exemple, ce vecteur correspondra au filtre linéaire optimal en détection et estimation bayesiennes; citons pour mémoire la théorie du filtrage de Wiener et de Kalman [16, 32]. L'expression (4) n'est utilisable dans la pratique que si les deux moments d'ordre deux y intervenant sont connus, ce qui limite fortement ses possibilités d'application. Ainsi, les matrices de covariance intervenant dans (4) sont en pratique remplacées par des matrices estimées à partir d'un ensemble d'observations $\{Y^\mu, x^\mu, 1 \leqq \mu \leqq M\}$ possédant les mêmes caractéristiques au second ordre, si bien que le projecteur à calculer prend la forme adaptative :

$$(5)\quad F_D(Y, x) = \left[\sum_{\mu=1}^{M} a_\mu Y^{\mu t} G^{-1} Y^\mu\right]^{-1} \times \left[\sum_{\mu=1}^{M} b_\mu Y^{\mu t} G^{-1} x^\mu\right]$$

Nous constatons donc que le calcul de produits tels que $A^{-1}B$, où A et B sont des matrices $p \times p$ et $p \times n$ respectivement, est omniprésent. Suivant le problème abordé, ces deux matrices peuvent avoir diverses formes particulières imposées : hermitiennes, symétriques réelles, Toeplitz, proches de Toeplitz, circulantes, Hankel, etc., ces structures pouvant être agencées par blocs dans le cas de signaux multivariables.

1.2. Structure des matrices

Limitons notre champ d'investigation à l'observation de processus localement stationnaires. Dans la pratique, les observations sont toujours de durée finie, et sont modélisées afin de pouvoir utiliser l'hypothèse de stationnarité au second ordre (au moins localement).

Cette modélisation, parfois implicite, a un impact important sur la structure des matrices de covariance figurant dans la formulation des projecteurs. En réalité, de manière non exhaustive, on peut grossièrement distinguer trois classes de modélisation, conduisant à trois types de structure de matrices de covariance :

1. les observations sont des réalisations infinies tronquées par une fenêtre d'observation; plusieurs types de fenêtres sont couramment utilisés à cet effet [17, 11, 8]. Ce type de modèle conduit à des matrices de covariance toeplitz symétriques par blocs dans le cas stationnaire;

2. les processus étudiés sont périodiques en moyenne quadratique, et les observations sont considérées comme étant des réalisations d'exactement une période de longueur. Les matrices de covariance de telles observations sont alors toeplitz circulantes par blocs. C'est cette modélisation qui est implicitement adoptée lorsqu'on exploite la Transformée de Fourier Discrète (TFD) des observations pour déterminer un traitement optimal [4];

3. Les observations sont directement délivrées dans le domaine fréquence ou vecteur d'onde. Les échantillons successifs sont par conséquent à valeurs complexes et théoriquement indépendants [4, 26]. Ceci permet, même si cette propriété n'est vérifiée dans la pratique qu'asymptotiquement, d'estimer facilement les matrices de covariance à partir d'un moyennage spectral ou spatial (périodogramme lissé) et d'en connaître les propriétés statistiques [4]. Dans ce type de modèle d'observation, les matrices de covariance de chaque échantillon sont hermitiennes, et la matrice de covariance de l'ensemble de l'observation est théoriquement hermitienne bloc-diagonale.

1.3. Types de récurrence

Afin de compléter cette caractérisation de l'acquisition des données, il faut définir le mode de succession des échantillons, dont tiendra nécessairement compte un traitement adaptatif en aval. Comme cela a été déjà souligné dans la littérature [8], il existe deux grandes familles de traitements adaptatifs :

1. les traitements dits « par blocs », dans lesquels les données arrivent par groupes disjoints qui sont traités séparément. L'estimation escomptée est alors calculée lors de chaque acquisition d'un nouveau bloc. En général, les observations spectrales sont délivrées de cette manière. Ces blocs peuvent éventuellement s'intersecter (recouvrement) [33] et être apodisés de fenêtres [4];

2. parallèlement, dans les traitements « récursifs », la sortie est recalculée à chaque acquisition d'un nouvel échantillon en fonction de la sortie calculée à l'instant précédent. Bien que ce soit malgré tout possible, les traitements récursifs ne fonctionnent habituellement pas conformément à une division par blocs.

Les classifications que nous venons de faire ci-dessus nous permettent maintenant de considérer les divers problèmes susceptibles d'être rencontrés :

1. Pour un traitement de type « par blocs », nous devrons calculer un nouveau projecteur à chaque acquisition d'un nouveau bloc de données. Si on se réfère à la pratique usuelle, ce type de traitement est souvent rencontré dans le domaine spectral. Des récurrences peuvent être construites en fonction du bloc précédent ou à l'intérieur même du bloc :

a. solution (bloc k) $=f\{$ solution (bloc $k-1$) $\}$;

b. solution (bloc k, échantillon j)
$=f\{$ solution (bloc k, échantillon $j-1$) $\}$.

Les matrices de covariance dans le premier cas (a) sont remises à jour par une modification de rang 1 :

$$(6) \qquad \begin{cases} \mathrm{B}(k)=(1-\alpha_k)\,\mathrm{B}(k-1)+\alpha_k u_k v_k^t; \\ 0<\alpha_k<1 \end{cases}$$

alors que dans le second (b), aucune relation particulière n'existe *a priori* entre les valeurs successives des covariances :

$$(7) \qquad \mathrm{B}(k,j)=\mathrm{B}(k,j-1)+\varepsilon(k,j)$$

En revanche, sous réserve de certaines propriétés de continuité de $\mathrm{B}(k,j)$ en fonction de j, l'innovation $\varepsilon(k,j)$ pourra être considérée de petite norme. Sinon, les covariances sont recalculées à chaque pas k à partir de moyennes :

$$(8) \qquad \mathrm{B}(k,j)=\sum_{m=1}^{\mathrm{M}}\sum_{q=-\mathrm{Q}}^{\mathrm{Q}} b_{k-m,j-q}\,u_{k-m,j}\,v_{k-m,j-q}^t$$

2. Pour les traitements « récursifs » sans division par blocs, le calcul des covariances se limite à :

$$\mathrm{B}(j)=\sum_{q=1}^{\mathrm{Q}} \mathrm{b}_{j-q}\,u_{j-q}\,v_{j-q}^t$$

et la récurrence s'écrit :

$$(9) \qquad \mathrm{B}(j)=f\{\mathrm{B}(j-1)\}=(1-\alpha_j)\,\mathrm{B}(j-1)+\alpha_j u_j v_j^t; \quad 0<\alpha_j<1$$

On peut retrouver les estimateurs classiques des matrices de covariance en jouant sur le coefficient α_k :
Si $\alpha_k=1/k$ et $\mathrm{B}(0)=0$ dans les récurrences (6) ou (9), l'estimateur de la covariance $\mathrm{B}(k)$ n'est autre que celui du périodogramme (ou corrélogramme) moyenné :

$$(10) \qquad \mathrm{B}(j)=\sum_{q=1}^{j} u_q v_q^t/j$$

En revanche, si $\alpha_k=\alpha$, le moyennage est de type exponentiel :

$$(11) \qquad \mathrm{B}(j)=\alpha\sum_{q=1}^{j}\alpha^{j-q}u_q v_q^t+(1-\alpha)^j\,\mathrm{B}(0)$$

1.4. Position du problème

Finalement, les calculs qui devront être réalisés seront ceux des projecteurs conformément aux expressions (1) à (3), dans lesquelles les matrices de covariance seront soit recalculées à chaque pas, soit calculées récursivement, à partir des transformations (6) à (9).
Nous avons déjà présenté dans [5] une implémentation systolique performante de projections dans le cas d'un calcul non récursif, et de matrices de covariance hermitiennes, ou symétriques réelles, sans structure imposée (le principe est rappelé en section 2.2). De plus, nous allons nous intéresser dans la section 3 aux calculs de projections régis par la récurrence exponentielle (9) et (11) :

$$(12) \qquad \begin{cases} \{\mathrm{F}_p(\mathrm{Y},x)\}_j=\mathrm{A}_j^{-1}\,\mathrm{B}_j \\ \mathrm{A}_j=(1-\alpha)\,\mathrm{A}_{j-1}+\alpha u_j u_j^t; \\ \mathrm{B}_j=(1-\alpha)\,\mathrm{B}_{j-1}+\alpha u_j v_j^t; \\ 0<\alpha<1 \end{cases}$$

où $u_j=\mathrm{Y}_j^t\mathrm{G}^{-1/2}$ et $v_j=x_j^t\mathrm{G}^{-1/2}$ sont donnés.
Dans un premier temps, aucune structure particulière ne sera imposée aux matrices de covariance si ce n'est l'hermicité (ou la symétrie) et la non-négativité. Elles ont une dimension dont l'ordre varie de 10 sur 10 à 100 sur 100. Dans la pratique, on peut envisager les réseaux permettant le calcul récursif des projections dans le cas des structures imposées (structure toeplitz symétrique par exemple). On ne gagne rien à introduire un réseau spécialisé sur un type particulier de structure, les réseaux proposés ici dans le cas général restent en effet d'une grande efficacité dans le cas des structures imposées. Il existe cependant des travaux traitant ces cas particuliers, nous nous contentons ici de renvoyer à la littérature correspondante [6, 7].

2. Réalisation systolique de $\mathrm{A}^{-1}\mathrm{B}$

2.1. Réseaux usuels de l'algèbre matricielle

Récemment, de nombreux réseaux systoliques ont été proposés pour résoudre les opérations fondamentales de l'algèbre linéaire (citons parmi les principaux les travaux de Gentleman et Kung [9], Kung et Leiserson [20], Hwang et Cheng [15], Kramer et Van Leeuwen [18], Li et Wah [23], Nash *et al* [25], Preparata et Vuillemin [27], Robert et Tchuente [29] etc.), le même réseau pouvant servir à différentes applications. Ils permettent de réaliser le produit de deux matrices, la triangularisation d'une matrice (décompositions LU et QR), l'inversion de matrices triangulaires, la multiplication de matrices et même la résolution de systèmes linéaires.
En ce qui concerne le problème particulier du calcul de $\mathrm{A}^{-1}\mathrm{B}$, on se ramène aux sous-problèmes élémentaires suivants : décomposition LU, inversion de matrices triangulaires et multiplication matricielle.

Plus précisément, on décompose la matrice A en LU, on inverse les matrices triangulaires L et U puis on les multiplie de façon à obtenir $U^{-1}L^{-1}$. Il ne reste alors plus qu'à postmultiplier par B pour obtenir le résultat.

Nous nous proposons dans cette section de faire un rapide tour d'horizon des réseaux systoliques élémentaires les plus performants pour chacun de ces problèmes élémentaires.

Multiplication matricielle

On présente ici un réseau systolique bidimensionnel qui réalise le produit de deux matrices denses A et B de taille n [27]. Comme il s'agit du premier réseau que nous décrivons, nous détaillons particulièrement cette présentation.

Le nombre de pas pour effectuer le calcul en séquentiel est n^3, et nous avons en vue un temps d'exécution linéaire (de l'ordre de n) sur un réseau de $O(n^2)$ cellules. Un produit de deux matrices de tailles n par n n'est jamais que le résultat de n^2 produits scalaires à n termes. Ainsi, pour calculer un élément du produit C, disons c_{ij}, on doit réaliser le produit scalaire de la ligne i de A et de la colonne j de B.

Une façon naturelle d'effectuer ce produit scalaire est illustrée sur les figures 2 et 3 (pour $n=3$). Ici, le réseau comprend une seule cellule, dont le registre interne est modifié par accumulations successives jusqu'à acquérir la valeur définitive du produit scalaire. On désigne par IPS (pour Inner Product Step), la cellule qui réalise 1 pas du produit scalaire (multiplication suivie d'une addition).

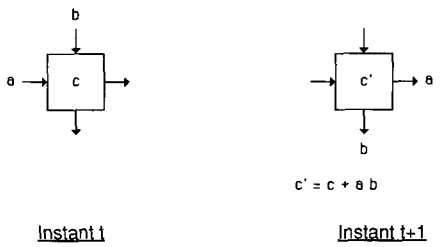


Fig. 2. – Cellule IPS.

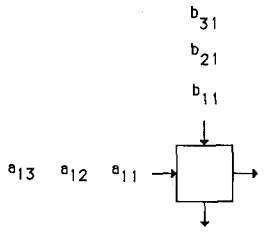


Fig. 3. – Calcul du produit scalaire.
Pour calculer l'élément c_{ij}, l'algorithme est donc le suivant :

$c_{ij} := 0;$
Pour $k := 1$ *à* n *faire*
$c_{ij} := c_{ij} + a_{ik} . b_{kj};$

Pour calculer la matrice C complète, nous allons utiliser n^2 cellules, une par coefficient, comme le montre la figure 4 sur des matrices 3×3. Le réseau obtenu fonctionne en temps $3n-2$. On peut s'en convaincre facilement en suivant le parcours des derniers coefficients a_{nn} et b_{nn}.

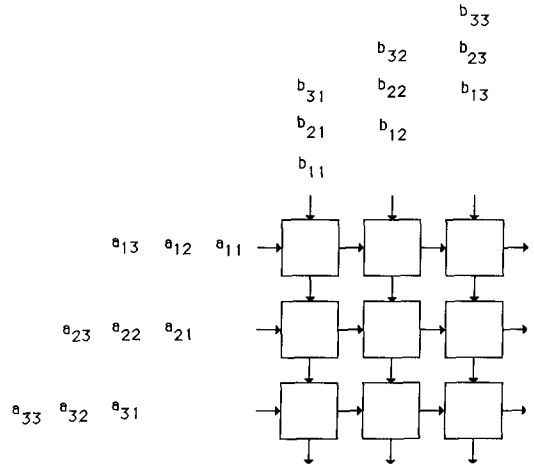


Fig. 4. – Produit de deux matrices denses.

Une difficulté cependant se pose. En effet, à la fin du calcul, les coefficients de la matrice C sont stockés dans le réseau. Il nous faut imaginer un mécanisme permettant de vider celui-ci. En effet, seules les cellules frontières peuvent communiquer avec l'hôte, il est donc exclu de lire directement et en parallèle (en un seul top d'horloge !) le contenu des registres internes des n^2 cellules. De même, l'instruction qui consiste à vider le réseau doit se propager de cellule en cellule, et ne peut être envoyée par un organe de contrôle central à toutes les cellules.

Plusieurs solutions peuvent être envisagées. Nous décrivons dans la suite un dispositif où le réseau se vide par la gauche, en utilisant le bus de circulation des coefficients a_{ij}. L'opération des cellules est maintenant commandée par un ou deux booléens de contrôle, qui circulent de la gauche vers la droite.

Informellement, le principe est le suivant : si la dernière accumulation dans une cellule K a lieu au temps t, au temps $t+1$ celle-ci envoie sur la droite le contenu c de son registre interne. Au temps $t+1$, sa voisine de droite K′ laisse passer inchangée la variable c en la transmettant à nouveau vers la cellule de droite K″. Au temps $t+2$, K′ envoie à son tour le contenu de son registre interne à K″. Le booléen contrôlant le vidage se propage donc à la vitesse 1/2, gagnant une nouvelle cellule tous les deux tops. Ce ralentissement est obtenu en insérant une bascule supplémentaire sur son parcours.

Nous pouvons maintenant décrire en détail ce que pourrait être le fonctionnement d'une cellule lors d'une mise en œuvre concrète. Tout d'abord, le contrôle des cellules doit coder les états suivants :

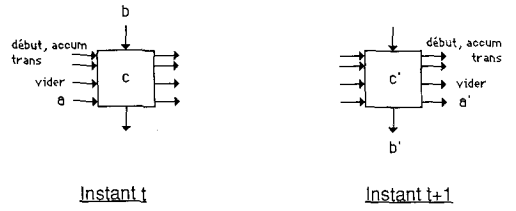
– *début :* commencer un nouveau calcul;

– *accum :* accumuler un pas de produit scalaire dans le registre interne (régime permanent);

— *trans :* transmettre inchangé un coefficient vers la droite (correspondant au registre interne d'une cellule plus à gauche);

— *vider :* transmettre le contenu du regitre interne vers la droite.

En fait, les trois variables *début*, *accum* et *trans* se propagent à la vitesse 1 et peuvent être codées sur deux bits, et un bit supplémentaire est nécessaire pour la variable *vider*, qui se propage à la vitesse 1/2. La figure suivante détaille le fonctionnement complet d'une cellule.




```
cas contrôle de
    début   : c' := a * b ; a' := a ; b' := b ;
    accum   : c' := c + a. b ; a' := a ; b' := b ;
    vider   : c' := nil ; a' := c ; b' := nil ;
    trans   : c' := c ; a' := a ; b' := nil ;
```

Fig. 5. — Contrôle des cellules du réseau.

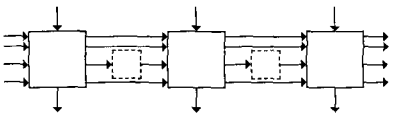


Fig. 6. — Circulation du booléen *vider*.

L'instruction *vider* doit arriver en même temps que l'instruction *trans*, avec priorité sur celle-ci, un top après le dernier coefficient significatif. Par exemple, la première cellule de la première ligne recevra cette instruction au top $n+1$.

Les figures 5 et 6 illustrent la communication entre deux cellules voisines d'une même ligne. On peut imaginer que la première cellule de chaque ligne transmette les instructions de contrôle qu'elle reçoit à sa voisine de dessous, puisque cette dernière effectue les mêmes opérations qu'elle avec un retard d'un top. Ceci permet de ne faire communiquer, en matière de contrôle, que la première cellule en haut à gauche avec l'extérieur, réduisant ainsi le nombre de ports d'entrée-sortie nécessaires.

Le premier coefficient c_{11} est délivré à la droite du réseau juste après a_{1n}, et un nouvel élément suit tous les tops. Cette phase de vidage requiert donc n tops supplémentaires, et le temps total d'exécution devient égal à $4n-2$.

Décomposition LU

De nombreux réseaux systoliques ont été proposés pour la triangularisation d'une matrice dense [1, 9]... Gentleman et Kung proposent un réseau orthogonal [9] qui permet d'effectuer la décomposition LU d'une matrice.

La décomposition d'une matrice en produit LU de matrices triangulaires est basée sur l'élimination de Gauss [10]. La décomposition LU est utilisée pour la résolution des systèmes linéaires $Ax=b$. Le second membre b subit les mêmes transformations que la matrice A, on posera à cet effet $A=(A, b)$ [matrice de taille $n\times(n+1)$]. On peut schématiser l'algorithme général de triangularisation de la manière suivante :

Pour $k := 1$ *à* $n-1$ *faire*
  Pour $i := k+1$ *à* n *faire*

$$\begin{bmatrix}\text{ligne } k\\ \text{ligne } i\end{bmatrix} := M_{ik}\begin{bmatrix}\text{ligne } k\\ \text{ligne } i\end{bmatrix}$$

La matrice M_{ik} est choisie de manière à annuler le coefficient en position (i, k). L'algorithme procède en $n-1$ étapes. A l'étape k, la k-ième ligne sert de pivot. On la combine avec toutes les lignes inférieures pour annuler les éléments de la k-ième colonne situés sous la diagonale.

Le choix des matrices M_{ik} dépend des propriétés de la matrice du système. En effet, il est exclus d'introduire des techniques de pivotage, même partiel, car cela aurait pour effet de détruire la régularité du réseau. Aussi emploie-t-on la méthode de Gauss (sans pivotage) pour des matrices symétriques définies positives ou à diagonales dominantes [10]. Dans le cas général, il faut recourir à des matrices de factorisations orthogonales (matrices de Givens) pour des raisons de stabilité. L'algorithme est le même dans les deux cas, au choix de la matrice M_{ik} près. De notre point de vue, il importe seulement d'organiser les rencontres *ad hoc* entre les lignes de la matrice, indépendamment de la nature des combinaisons réalisées par les processeurs.

Gentleman et Kung [9] utilisent un réseau triangulaire de processeurs connectés orthogonalement, reproduit figure 7 pour $n=4$. Le nombre total de cellules est égal à $n[(n+1)/2+1]$. La matrice A entre dans le réseau colonne par colonne. Plus précisément, le processeur P_{1k} reçoit les coefficients de la colonne k, un nouvel élément à chaque top d'horloge, à partir du temps k. Ce format d'entrée est également représenté figure 7.

D'une façon informelle, disons que la k-ième ligne du réseau est destinée à effectuer l'étape k de l'algorithme. Pour cela, on procède en deux temps. D'abord, il faut stocker la k-ième ligne de la matrice dans le réseau, un coefficient dans le registre interne de chaque cellule. Les lignes suivantes traversent alors les processeurs $P_{k1}, \ldots, P_{k,n+2-k}$, et effectuent au passage une combinaison avec la ligne pivot k. A cet effet, les matrices $M_{k+1,k}, M_{k+2,k}, \ldots, M_{nk}$ sont générées par la cellule ronde et transmises à toutes les autres cellules de la ligne. Bien sûr, ces deux phases sont pipelinées, si bien que la cellule P_{k1} génère les premières matrices M_{ik} alors que les derniers coefficients de la ligne pivot ne sont pas encore stockés.

Examinons en détail le fonctionnement de la première ligne du réseau. Tout d'abord, la première ligne de la matrice A est stockée dans les registres internes des

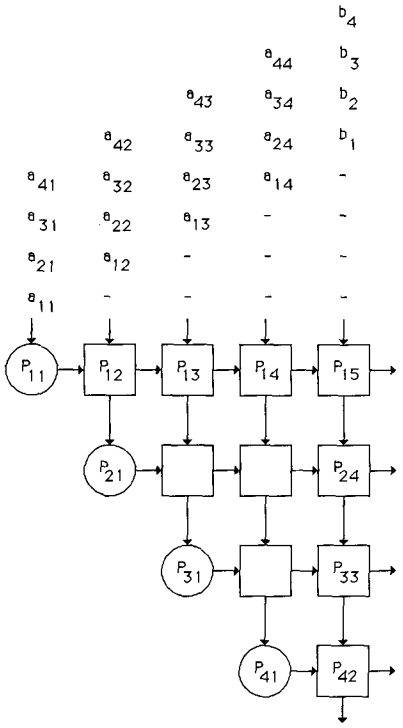


Fig. 7. – Triangularisation d'une matrice dense.

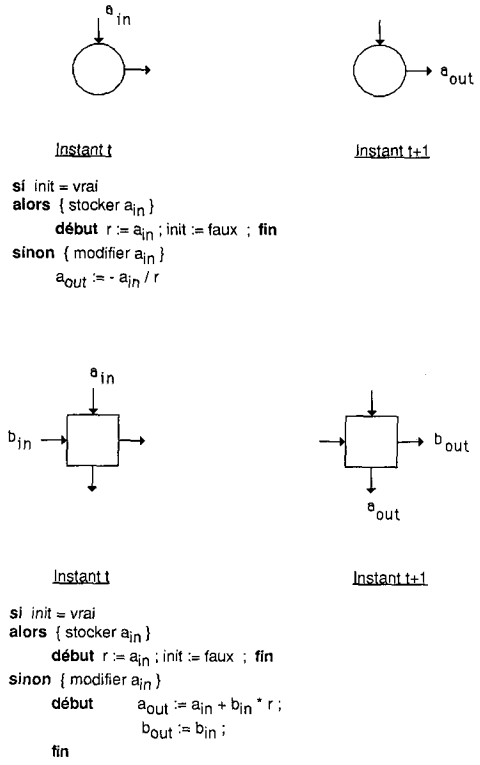


Fig. 8. – Programmation des cellules pour l'élimination de Gauss.

cellules : l'élément a_{1k} est stocké dans le processeur P_{1k} au temps k. Quand une ligne numéro $i\,(i \geqq 2)$ est lue par le réseau, elle est combinée avec la ligne 1 de manière à annuler l'élément a_{i1}. La matrice M_{i1} est générée par la cellule P_{11} au temps i, puis elle est envoyée aux autres cellules de la ligne. Plus précisément, la cellule $P_{1k}\,(k \leqq 2)$ effectue au temps $i+k-1$, la transformation suivante :

$$(a_{1k},\, a_{ik})^t := M_{i1} \,.\, (a_{1k},\, a_{ik})^t$$

On procède de la même manière pour toutes les lignes k.

Il y a deux types de cellules dans le réseau de la figure 7 : les cellules représentées par des ronds et celles représentées par des carrés.

Plus précisément, la k-ième ligne du réseau est composée de :

– la cellule P_{k1} (cellule ronde), qui génère les matrices M_{ik} pour $i > k$;

– toutes les autres (carrées), qui appliquent les transformations M_{ik}.

La programmation détaillée des cellules est décrite ci-dessous dans la figure 8 dans le cas de l'élimination de Gauss pour une matrice 4×4.

Le programme des processeurs est l'unique chose à modifier pour traiter le cas d'une factorisation orthogonale avec des matrices de Givens. Le flot des données et l'organisation des rencontres entre celles-ci reste inchangé, quel que soit l'algorithme de triangularisation utilisé. Dans la pratique, il serait facile de combiner les deux approches Gauss et Givens pour obtenir des cellules programmables, le choix de l'algorithme dépendrait alors d'un bit de contrôle. Bien mieux, on peut imaginer des variantes mixant les deux stratégies selon la taille du pivot rencontré. Cependant, le synchronisme total limite l'intérêt de telles stratégies, puisque le réseau fonctionne au rythme du processeur le plus lent.

L'évolution des opérations de la première ligne du réseau est résumée dans les figures suivantes sur un exemple avec $n=4$. Dans la table de la figure 9, on note M_{i1} pour la génération de la matrice d'élimination. De même, on note a_{1j} le contenu des registres internes des cellules dans la figure 9. $a_{ij}^{(k)}$ est la valeur de a_{ij} après k étapes d'élimination.

A la fin des $n-1$ étapes de l'algorithme, la matrice triangulaire (U, b') est stockée dans le réseau de la manière suivante :

$$\begin{aligned} u_{11}u_{12}u_{13}u_{14}\,b'_1 \\ u_{22}u_{23}u_{24}\,b'_2 \\ u_{33}u_{34}\,b'_3 \\ u_{44}\,b'_4 \end{aligned}$$

Il reste à vider le réseau. La philosophie générale du modèle impose un vidage systolique, par propagation d'une variable de contrôle. De plus, on voudrait le vidage du réseau de la phase de calcul, de manière à éviter n tops supplémentaires. Nous adoptons un processus de vidage où chaque cellule envoie le contenu de son registre interne à sa voisine de droite quand elle a fini de calculer.

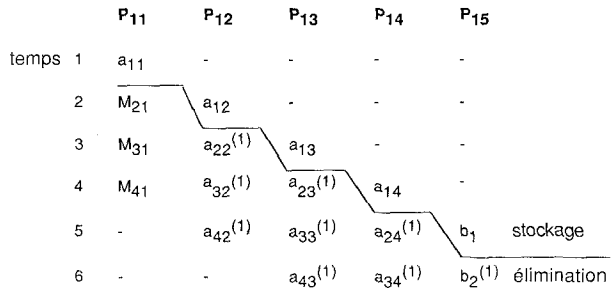


Fig. 9. – Opérations de la première ligne du réseau.

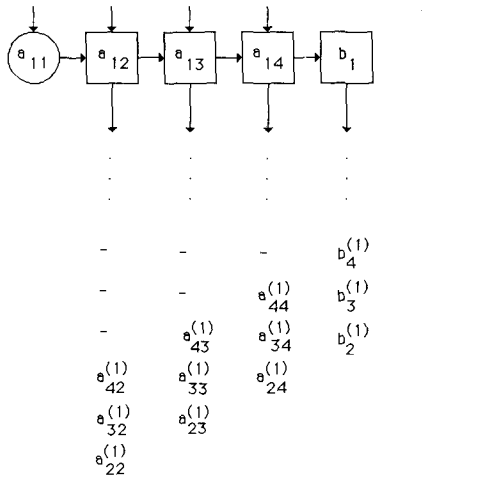


Fig. 10. – Visualisation de la première ligne du réseau.

Dans l'exemple précédent avec $n=4$, au temps 4 la cellule P_{11} travaille pour la dernière fois. P_{11} reste inactive durant un top puis envoie u_{11} vers la droite. P_{12} finit son travail au temps 5 et transmet son résultat au temps 6. Il transmet u_{11} vers la droite au temps 7, puis envoie le contenu de son propre registre u_{12} au temps 8. Le processus se poursuit, et u_{1j} est délivré en sortie par la cellule P_{15} au temps $9+i$ $(1 \leqq i \leqq n+1)$. De même, on commence à vider la deuxième ligne du réseau au temps 8. On voit que les premières lignes du réseau sont vidées alors que les dernières continuent à opérer: c'est là le «pipeline» que nous désirions.

Dans le cas général, le dernier calcul à lieu au temps $2i+3$ dans la cellule P_{i2}, et $u_{i,\,i+k}$ $(1 \leqq i \leqq n,$ $0 \leqq k \leqq n-i+1)$ est délivré en sortie du réseau au temps $2n+i+k+1$. Enfin, remarquons que l'on peut simplifier le vidage dans le cas de l'élimination de Gauss. Comme nous l'avons déjà remarqué, les $u_{i,\,i+k}$ ne seront plus modifiés une fois stockés, et peuvent donc être envoyés immédiatement vers le bas lors de leur stockage.

Le temps de calcul total pour effectuer la décomposition LU d'une matrice de taille n est donc $3n+2$ sur ce réseau. Dans le cas de l'élimination de Gauss, on obtient également la décomposition LU de A: les coefficients de la matrice L se déduisent directement (par changement de signe [10]) des matrices M_{ik}, qui sont délivrées en sortie à la droite du réseau lors du processus d'élimination.

Inversion de matrices triangulaires

Le réseau orthogonal précédent permet d'inverser une matrice triangulaire dans un temps $3n$. Li et Wah proposent un réseau de cellules connectées hexagonalement [23] qui calcule l'inverse en un temps asymptotique de $2n$ seulement. Cette performance (la meilleure connue) a été obtenue à l'aide d'une méthodologie de synthèse systématique des réseaux.

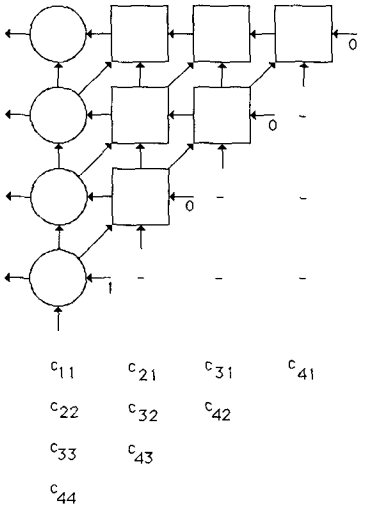


Fig. 11. – Inversion d'une matrice triangulaire.

Le réseau est représenté dans la figure 11 pour une matrice triangulaire inférieure C de taille 4×4. Comme le montre la figure, le flot des entrées se fait diagonale par diagonale au rythme de 1 coefficient tous les tops d'horloge. Le réseau est composé de deux types de cellules, dont l'action est décrite à la figure 12.

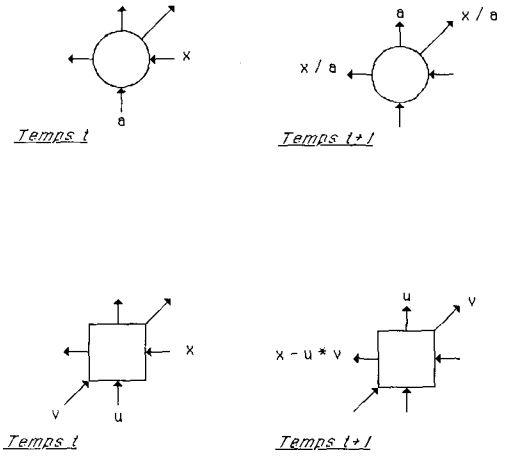


Fig. 12. – Opérations des cellules pour l'inversion.

Plutôt que de proposer une explication détaillée du fonctionnement de ce réseau, nous invitons le lecteur à suivre lui-même le processus d'élimination sur le réseau.

Bilan

On peut maintenant évaluer le temps de calcul global du produit $A^{-1}B$ à partir des réseaux élémentaires précédents pour une matrice A de taille $n \times n$. On obtient asymptotiquement la triangularisation en $3n$ unités de temps, l'inversion d'une matrice triangulaire en $2n$ et la multiplication en $4n$. Soit $11n$ unités de temps pour obtenir la matrice A^{-1}, donc $15n$ au total pour obtenir $A^{-1}B$. De plus, le flot des entrées n'est pas compatible d'une application à l'autre (entrée des matrices par colonnes ou par diagonales). Il faudrait donc en toute rigueur rajouter au temps de calcul, le temps de réordonnancement des données, et le coût d'un stockage intermédiaire en mémoire de l'architecture hôte.

Il est possible d'utiliser d'autres réseaux élémentaires pour effectuer le calcul de $A^{-1}B$, comme le réseau de Kramer et Van Leewen qui calcule l'inverse d'une matrice. Mais cette opération doit être suivie d'une multiplication sur un autre réseau, et la même remarque s'applique. A ce sujet, nous renvoyons à la discussion de Moraga [24] qui montre tout l'intérêt d'une approche basée sur un seul réseau, comme celui que nous décrivons maintenant.

2.2. Calcul direct de $A^{-1}B$

Le calcul direct de $A^{-1}B$ peut être vu comme la résolution de p systèmes linéaires de même matrice:

$$A x_i = b_i, \qquad 1 \leqq i \leqq p$$

On note B la matrice de taille $n \times p$ formée des p vecteurs seconds membres b_i et C la matrice (A, B) de taille $n \times (n+p)$.

Nous proposons ci-dessous un réseau orthogonal de $n(n+1)$ cellules qui permet le calcul de $A^{-1}B$ en temps $4n+p$. C'est le réseau transposé de celui décrit dans [5].

Le calcul de $A^{-1}B$ s'effectue à partir de la diagonalisation de Jordan. Comme dans le cas de la triangularisation, si la matrice A n'a pas les propriétés de stabilité requises, il faut mixer l'algorithme d'élimination avec un algorithme de factorisation orthogonale. Supposons pour l'instant que A est non singulière, et préoccupons nous exclusivement de l'implantation systolique de la méthode, sans souci numérique. L'algorithme vise à réduire la matrice $C=(A, B)$ en la matrice $(I_n, A^{-1}B)$ en la prémultipliant par n matrices élémentaires $J_1, J_2, \ldots, J_n$ (matrices de Jordan).

Posons $C_k = J_k . C_{k-1}$, la matrice obtenue après k étapes de l'algorithme, en partant de $C_0 = C$. Soit C_k^0 la matrice de taille $n \times (n+p-k)$ formée des $n+p-k$ dernières colonnes de C_k. On choisit J_k de telle sorte que les k premières colonnes de C_k soient celles de I_n (matrice identité d'ordre n). C_k a donc la structure suivante:

$$C_k = [\text{les } k \text{ premières colonnes de } I_n, C_k^0]$$

En particulier, C_n^0 est la matrice $A^{-1}B$ cherchée. La matrice J_k diffère seulement de l'identité par sa k-ième colonne, que l'on note:

$$[c_{1\,k}^{(k)} \ldots c_{nk}^{(k)}]^t$$

Les coefficients de la matrice C_k^0 sont notés $(c_{ij}^{(k)})$, $1 \leqq i \leqq n$, $k+1 \leqq j \leqq n+p$. La valeur des $c_{ij}^{(k)}$, $1 \leqq i \leqq n$, $k \leqq j \leqq n+p$, $1 \leqq k \leqq n$, est calculée récursivement par l'algorithme suivant, partant de $c_{ij}^{(0)} = c_{ij}$:

pour $k := 1$ *à* n *faire*
  début
    { calcul de J_k }
(A) $c_{kk}^{(k)} := 1/c_{kk}^{(k-1)}$;
    pour $i := 1$ *à* n, $i \neq k$, *faire*
(B) $c_{ik}^{(k)} := -c_{ik}^{(k-1)} * c_{kk}^{(k)}$;
    { calcul de C_k^0 }
    pour $j := k+1$ *à* $n+p$ *faire*
      début
        pour $i := 1$ *à* n, $i \neq k$, *faire*
(C) $c_{ij}^{(k)} := c_{ij}^{(k-1)} + c_{ik}^{(k)} * c_{kj}^{(k-1)}$;
(D) $c_{kj}^{(k)} := c_{kk}^{(k)} * c_{kj}^{(k-1)}$;
      fin;
fin;

Attention, ici $c_{ik}^{(k)}$ désigne un élément de la k-ième colonne de J_k, tandis que $c_{ij}^{(k)}$ avec $k < j$ se rapporte à la matrice C_{k2}^0.

Pour implanter cet algorithme sur un réseau systolique, nous utilisons un réseau orthogonal à $n(n+1)$

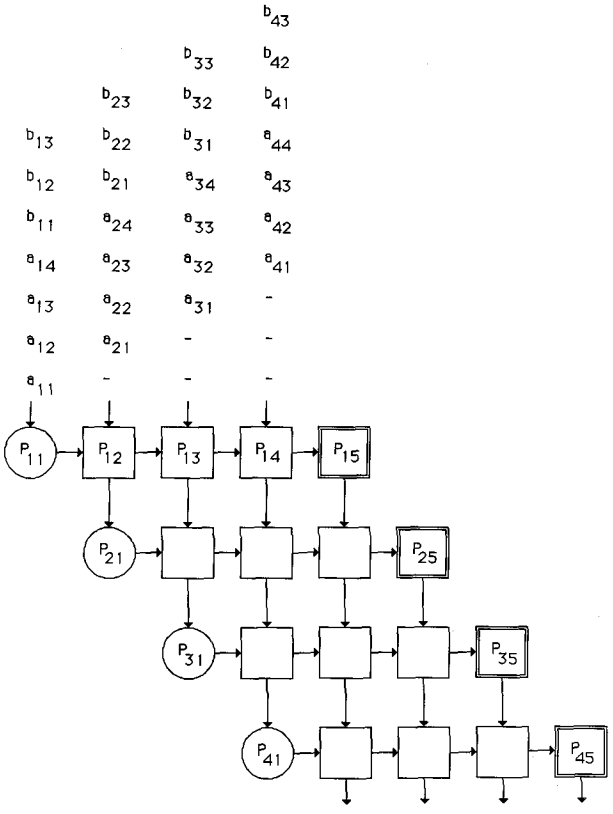


Fig. 13. – Calcul direct de $A^{-1}B$.

cellules. Le réseau se compose de n lignes, chaque ligne k comprenant $n+1$ processeurs numérotés de gauche à droite $P_{k,1}, \ldots, P_{k,n+1}$. Il est décrit dans la figure 13 sur un exemple de taille $n=4$ et $p=3$. La matrice A entre dans le réseau ligne par ligne, suivie de la matrice B. Plus précisément, la k-ième ligne de la matrice C est en entrée de la cellule P_{1k}, un nouvel élément tous les tops d'horloge, à partir du temps k. Ce format d'entrée est également décrit figure 13.

Avant de décrire en détail le fonctionnement de ce réseau, il convient de donner un mot d'explication informelle: dans le réseau de Gentleman et Kung, les lignes de la matrice sont stockées dans les processeurs, et les matrices M_{ik} circulent à travers le réseau. Ici, le fonctionnement est dual: les lignes circulent, et les matrices M_{ik} sont stockées dans les cellules. En effet, dans l'algorithme de Jordan, chaque ligne doit, quand elle devient la ligne pivot, rencontrer toutes les autres, et non plus seulement celles d'indices supérieurs. Plutôt que de faire repartir à travers le réseau une ligne déjà stockée, comme nous l'avons fait précédemment, il est préférable ici de faire circuler toutes les lignes de la matrice C. Ceci explique aussi que la matrice A entre dans le réseau sous forme transposée par rapport au réseau précédent de Gentleman et Kung.

Il y a n étapes dans l'algorithme de Jordan. Nous implantons chacune d'entre elles sur une ligne du réseau. Plus précisément, la k-ième instance de la boucle externe est réalisée par la k-ième ligne du réseau, qui reçoit la matrice C^0_{k-1} de la ligne $k-1$ et délivre la matrice C^0_k à la ligne $k+1$.

Chaque étape k comporte deux opérations distinctes: d'abord la génération de la matrice J_k, puis le calcul du produit $C^0_k = J_k C^0_{k-1}$. L'opération des cellules est définie en conséquence: les cellules de la ligne k calculent et stockent les coefficients $c^{(k)}_{ik}$, puis effectuent des calculs IPS.

Une fois les n matrices élémentaires J_k calculées et stockées, le réseau est initialisé. Il agit alors comme un opérateur linéaire qui transforme toute matrice B en $A^{-1}B$.

Les opérations des cellules sont décrites dans la figure 14. Il y a trois types de cellules:

– type 1: les cellules rondes calculent l'inverse de leur première entrée valide, suivant l'instruction (A) dans l'algorithme. Ensuite, elles agissent simplement comme des cellules de délai;

– type 2: les cellules carrées initialisent d'abord leur registre interne en stockant (après modification) leur première entrée valide [instruction (B)]. Puis elles agissent comme des cellules IPS [instruction (C)];

– type 3: les cellules à double carré opèrent en fait comme les cellules carrées, mais leur registre interne n'est pas initialisé de la même façon (voir *fig.* 14). Elles effectuent l'instruction (D).

Dans la k-ième ligne du réseau, le processeur de gauche $P_{k,1}$ est de type 1, et le processeur de droite $P_{k,n+1}$ est de type 3. Tous les autres processeurs sont de type 2. Comme pour le réseau de triangularisation, l'action des cellules est controlée par une variable

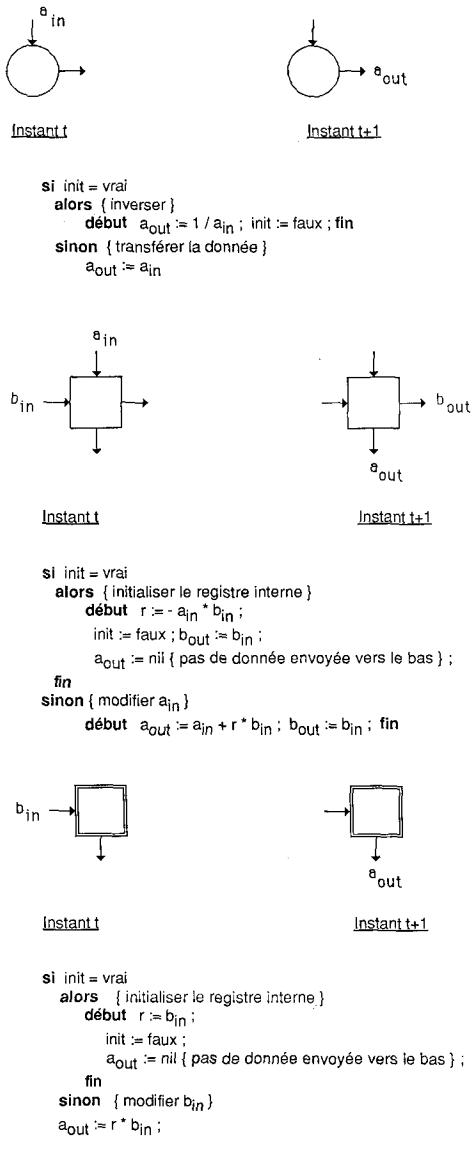


Fig. 14. – Opérations des cellules pour la diagonalisation de Jordan.

booléenne «init» qui repère si la première donnée que reçoit la cellule est valide. La même simplification a été adoptée: «init» (initialisé à la valeur «vrai») est supposé résider dans les cellules au début de l'exécution. Comme P_{kj} opère pour la première fois au temps $3k+j-3$, un signal en entrée au temps 1 de P_{11}, circulant à vitesse 1 vers la droite et 1/2 vers le bas, fait l'affaire.

Décrivons en détail l'action de la première ligne du réseau. Au temps 1, le processeur P_{11} calcule $c^{(1)}_{11} := 1/c^{(0)}_{11}$ et agit ensuite comme une cellule délai. Au temps 2, P_{12} initialise son registre interne, avec le calcul $c^{(1)}_{21} := -c^{(0)}_{21} * c^{(1)}_{11}$. P_{12} agit alors comme une cellule IPS: au temps 3, il modifie $c^{(0)}_{22}$ en $c^{(1)}_{22} := c^{(0)}_{22} + c^{(1)}_{21} * c^{(0)}_{12}$, et de même avec c_{23}, c_{24}, ...

P_{13}, P_{14}, P_{1n} opèrent comme P_{12}, commençant respectivement au temps 3, 4, ..., n. Au temps $n+1$,

$c_{11}^{(1)}$ gagne le processeur $P_{1,n+1}$ et se stocke dans son registre interne. De $n+2$ jusqu'à la fin de l'exécution, $P_{1,n+1}$ évalue la première ligne de C_1^0: à $n+2$, il calcule $c_{12}^{(1)} = c_{11}^{(1)} * c_{12}^{(0)}$, et de même avec c_{13}, c_{14}, ..., suivant l'instruction (D).

Les figures 15 et 16 suivantes résument, pour $n=4$ et $p=3$, l'action de la première ligne du réseau. On note que cette première ligne délivre la matrice C_1^0, de taille 4×6. Dans le cas général, C_1^0 est de taille $n\times(n+p-1)$, une colonne a été perdue en traversant la première ligne du réseau.

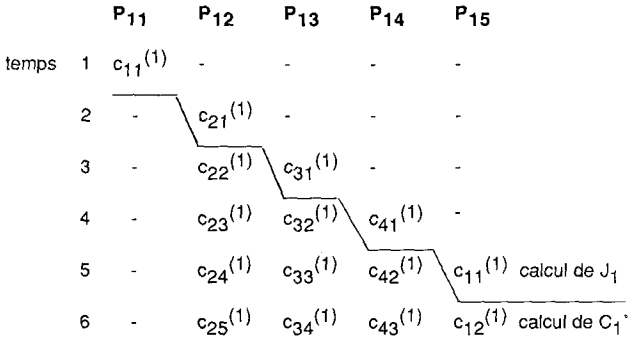


Fig. 15. – Opération de la première ligne du réseau.

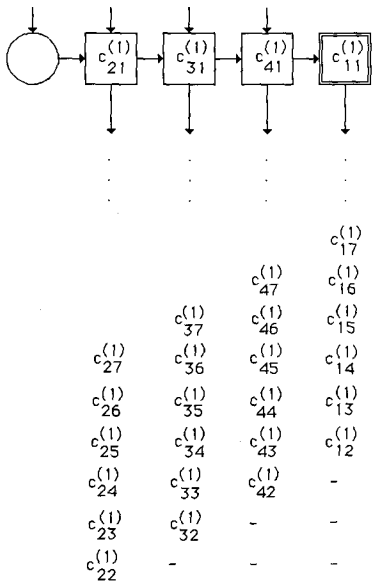


Fig. 16. – Visulaisation de la première ligne du réseau.

De façon similaire, la k-ième ligne du réseau effectue le calcul $C_k^0 = J_k C_{k-1}$. La matrice en entrée C_{k-1}^0 est de taille $n\times(n+p-k+1)$, et la matrice en sortie C_k^0 est de taille $n\times(n+p-k)$. Ainsi, après avoir traversé tout le réseau, une ligne de $n+p$ coefficients devient une ligne contenant p coefficients seulement. $C_0 = C_0^0 = C$ est en entrée de la première ligne, C_1^0 en entrée de la seconde et C_{n-1}^0 en entrée de la n-ième. Finalement, le réseau délivre en sortie la matrice $C_n^0 = A^{-1} B$.

Pour bien comprendre le fonctionnement de ce réseau, il est important de noter que les lignes de la matrice initiale sont permutées circulairement en traversant le réseau, de manière à ce que la ligne pivot à l'étape k puisse être combinée avec toutes les autres dans les cellules $P_{k1}, \ldots, P_{k,n+1}$.

Résumons les performances du réseau que nous venons de présenter: on calcule $A^{-1} B$ en $4n+p-2$ tops d'horloge sur un réseau de $n(n+1)$ cellules où A et B sont des matrices denses de tailles respectives $n\times n$ et $n\times p$. Ces chiffres sont à comparer au bilan de la section précédente.

3. Application au filtrage adaptatif

3.1. Calcul de l'inverse d'après Woodbury

La décomposition de Woodbury permet de calculer l'inverse d'une somme de matrices à partir des matrices elles-mêmes et de leurs inverses [10], p. 3 et donc en particulier de calculer l'inverse d'une matrice ayant subi une modification de rang k.

Plus précisément dans ce cas, si U et V désignent deux matrices de taille $n\times k$, alors sous certaines hypothèses de régularité [10], l'inverse de la matrice $M_{i+1} = (M_i + UV^t)$ peut être obtenu par la formule suivante:

$$M_{i+1}^{-1} = M_i^{-1} - M_i^{-1} U (I_k + V^t M_i^{-1} U)^{-1} V^t M_i^{-1}$$

Dans le cas d'une modification de rang 1, on obtient plus simplement:

$$M_{i+1}^{-1} = M_i^{-1} - (M_i^{-1} u v^t M_i^{-1})/(1 + v^t M_i^{-1} u)$$

(u et v désignant des vecteurs colonnes).

Dans le problème qui nous occupe ici, on cherche à obtenir le produit $A_{i+1}^{-1} B_{i+1}$, en fonction du produit $A_i^{-1} B_i$ au pas précédent. A_{i+1} et B_{i+1} sont des matrices respectivement de tailles $n\times n$ et $n\times p$, elles sont obtenues par modifications de rang 1 de la manière suivante:

$$A_{i+1} = (1-\alpha) A_i + \alpha u^{(i)} u^{(i)t}$$
$$B_{i+1} = (1-\alpha) B_i + \alpha u^{(i)} v^{(i)t}$$

où α est un réel appartenant à $]0, 1[$, $u^{(i)}$ et $v^{(i)}$ sont respectivement des vecteurs de tailles n et p.

On applique alors la formule de Woodbury pour calculer A_{i+1}^{-1} en fonction de A_i^{-1}. La formule complète s'écrit:

$$A_{i+1}^{-1} = [A_i^{-1} - \alpha A_i^{-1} u^{(i)} u^{(i)t} A_i^{-1} / (1 - \alpha + \alpha u^{(i)t} A_i^{-1} u^{(i)})]/(1-\alpha)$$

Ce calcul complexe comporte quatre étapes indépendantes qui s'organisent de la manière suivante:

– une première étape où l'on calcule le produit $d^{(i)} = u^{(i)t} A_i^{-1}$;

– le calcul de la matrice $E_i = \alpha\, d^{(i)t} d^{(i)}$ mené simultanément avec le calcul du coefficient $f_i = 1 - \alpha + \alpha\, u^{(i)t} d^{(i)}$;
– division $G_i = E_i / f_i$;
– calcul de $(A_i^{-1} - G_i)/(1-\alpha)$.

Bien que les calculs ci-dessus soient simples et admettent bien une réalisation de type « pipeline », ils se prêtent mal à une implantation systolique efficace. En effet, on peut utiliser des réseaux connus pour réaliser les opérations élémentaires de chacune des quatre étapes, mais on rencontre alors les mêmes problèmes que précédemment – format d'entrées/sorties incompatibles, stockages intermédiaires – auxquels vient s'ajouter la nécessité de dupliquer certaines données.

3.2. Réseau adaptatif

Nous proposons dans cette section un réseau systolique qui permet de calculer directement le produit $A_{k+1}^{-1} B_{k+1}$, en fonction de A_k et B_k au pas précédent en tenant compte des modifications de rang 1. Partant du réseau décrit au chapitre 2.2, il suffit de lui adjoindre en entrée un réseau de préconditionnement.

Étudions en détail le passage de l'étape i à l'étape $i+1$. Tout d'abord, pour simplifier l'écriture, on note u_i l'élément courant du vecteur u ($i = 1, \ldots, n$), en omettant l'exposant i de l'étape (de même pour v).

Le réseau de préconditionnement a pour fonction de modifier les valeurs courantes des coefficients a_{ij} et b_{ij} selon les formules:

$$a_{ij} := (1-\alpha)\, a_{ij} + \alpha\, u_i\, u_j$$
$$b_{ij} := (1-\alpha)\, b_{ij} + \alpha\, u_i\, v_j$$

On adopte le schéma de fonctionnement suivant (voir *fig.* 17): chaque ligne de la matrice courante (A, B) circule à travers un anneau composé de $n+p-1$ registres et d'une cellule du réseau de préconditionnement. Lors de son passage dans le réseau de préconditionnement, chaque coefficient est modifié selon les formules précédentes. La nouvelle valeur, qui retourne circuler dans l'anneau, est également envoyée en entrée du réseau de calcul de la section 2.2. Après une phase d'initialisation du réseau, qui correspond au premier calcul de $A_1^{-1} B_1$, une nouvelle matrice $A_k^{-1} B_k$ est délivrée tous les $n+p$ tops.

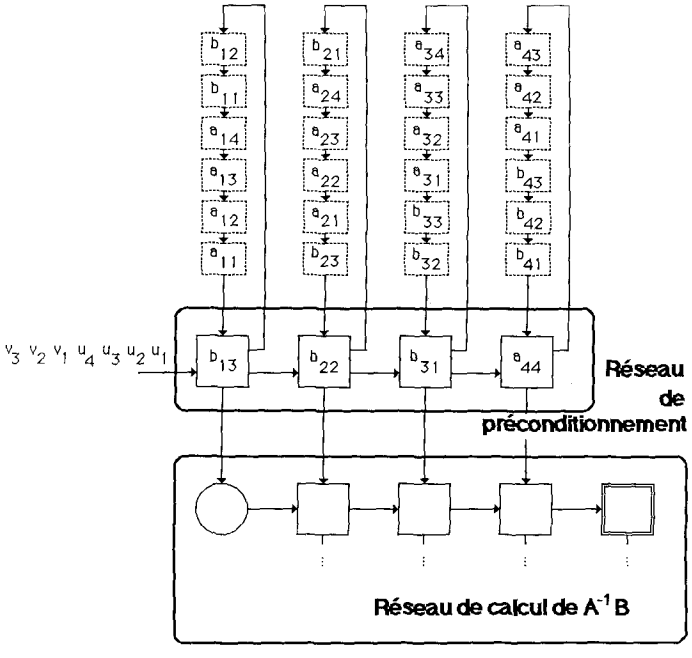


Fig. 17. – Fonctionnement du réseau de préconditionnement.

Le réseau de préconditionnement se compose de n cellules, une pour traiter chaque ligne de la matrice (A, B). Pour simplifier la description des cellules (*fig.* 18), on a supposé que l'élément u_i et le coefficient α résident dans la i-ième cellule du réseau. Pour s'affranchir de cette hypothèse, il suffit d'adjoindre un port vertical à chaque cellule i pour recevoir l'élément u_i (en même temps que $a_{i\,1}$) et de faire circuler α à travers le réseau, de la gauche vers la droite. Au passage, remarquons que cette solution permet de choisir une valeur différente de α à chaque étape, permettant ainsi de traiter le cas de durées de stationnarité variables.

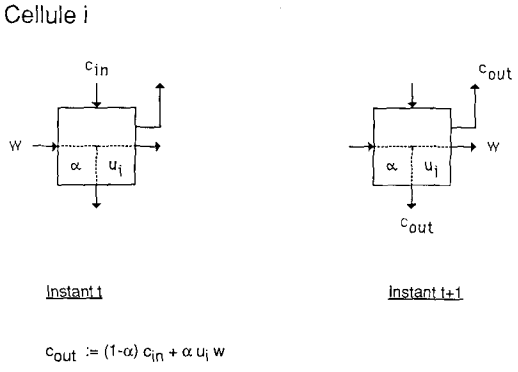


Fig. 18. – Opération des cellules du réseau de préconditionnement.

4. Conclusion

Deux caractéristiques dominantes du traitement du signal sont les suivantes: des débits d'entrées/sorties très rapides et une énorme masse de données en mémoire. Une puissance de calcul de l'ordre de la centaine de mégaflops est souvent nécessaire pour obtenir un résultat en temps-réel. Heureusement, les algorithmes utilisés possèdent pour la plupart des propriétés de régularité, répétitivité et localité, qui permettent d'envisager des solutions VLSI pour obtenir de telles performances à un coût raisonnable [21]. L'exemple du filtrage adaptatif que nous traitons dans cet article en est une bonne illustration.

Manuscrit reçu le 1er *novembre* 1986.

BIBLIOGRAPHIE

[1] H. M. AHMED, J. M. DELOSME et M. M. MORF, Highly concurrent computing structures for matrix arithmetic and signal processing, *IEEE Computer*, 15, n° 1, 1982, p. 65-82.

[2] M. BELLANGER, *Traitement Numérique du Signal*, Masson, 1980.

[3] A. BLANC-LAPIERRE et B. PICINBONO, *Fonctions aléatoires*, Masson, 1981.

[4] D. R. BRILLINGER, *Time series, data analysis, and theory*, Holden Day, 1981.

[5] P. COMON et Y. ROBERT, *A systolic array for computing* BA^{-1}, Rapport de Recherche TIM3/IMAG, Grenoble, n° 591, 1986.

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

[7] J. M. DELOSME et I. C. F. IPSEN, An illustration of a methodology for the construction of efficient systolic architectures in VLSI, *Proc. Second Int. Symposium on VLSI technology, systems and applications*, Taipei, Taiwan, 1985, p. 268-273.

[8] B. FRIEDLANDER, Lattice filter for adaptive processing, *Proceedings of the IEEE*, 70, n° 8, 1982, p. 829-867.

[9] W. M. GENTLEMAN et H. T. KUNG, Matrix triangularisation by systolic arrays, *Proc. SPIE* 298, *Real-time Signal Processing IV*, San Diego, California, 1981, p. 19-26.

[10] G. H. GOLUB et C. F. VAN LOAN, *Matrix computations*, The Johns Hopkins University Press, 1983.

[11] C. GUEGUEN, An Introduction to Displacement Ranks. Les Houches, *Summer School, Session XLV "Signal Processing"*, Aug. 12-Sept. 6, 1985, DURANI, LACOUME et STORA éd., North Holland (à paraître).

[12] S. HAYKIN éd., *Non Linear Methods for Spectral Analysis*, 1979, Springer Verlag.

[13] H. L. HONIG et D. G. MESSERSCHMITT. *Adaptive Filters; Structures, Algorithms and Applications*, 1984, Kluwer Academic Publishers.

[14] K. HWANG et F. BRIGGS, *Computer architecture and parallel processing*, Mac Graw Hill, 1984.

[15] K. HWANG et Y. H. CHENG, Partitioned matrix algorithm for VLSI arithmetic systems, *IEEE Trans. Computers*, 31, n° 12, 1982, p. 1215-12224.

[16] H. JAZWINSKI, *Stochastic processes and Filtering Theory*, 1970, Academic Press.

[17] T. KAILATH éd., *Modern Signal Processing*, 1985, Springer Verlag.

[18] M. R. KRAMER et J. VAN LEEUWEN Jr, *Systolic computation and VLSI, Foundations of Computer Science IV*, J. W. DEBAKKER et J. VAN LEEUWEN éd., 1983, p. 75-103.

[19] H. T. KUNG, Why systolic architectures, *IEEE Computer*, 15, n° 1, 1982, p. 37-46.

[20] H. T. KUNG et C. E. LEISERSON, Systolic arrays for (VLSI), *Proc. of the Symposium on Sparse Matrices Computations*, I. S. DUFF *et al.* éd., Knoxville, Tenn., 1978, p. 256-282.

[21] S. Y. KUNG, VLSI array processors, *IEEE ASSP Magazine*, 2, n° 3, 1985, p. 4-22.

[22] S. Y. KUNG, H. J. WHITEHOUSE et T. KAILATH éd., *VLSI and Modern Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1985.

[23] G. H. LI et B. W. WAH, The design of optimal systolic arrays, *IEEE Trans. Computers*, 34, n° 1, 1985, p. 66-77.

[24] C. MORAGA, On a case of symbiosis between systolic arrays, *Integration the VLSI Journal*, 2, 1984, p. 243-253.

[25] J. G. NASH, S. HANSEN et G. R. NUDD, *VLSI processor arrays for matrix manipulation, VLSI Systems & Computations*, Rockville, MA, H. T. KUNG *et al.* éd., Computer Science Press, 1981, p. 367-378.

[26] B. PICINBONO, Signaux Déterministes et Aléatoires: Analyse et Modes Houches, *Summer School, Session XLV "Signal Processing"* Aug. 12-Sept. 6, 1985, DURANI, LACOUME et STORA, éd., North Holland (à paraître).

[27] F. P. PREPARATA et J. VUILLEMIN, Area-time optimal VLSI networks for multiplying matrices, *Information Processing Letters*, 11, n° 2, 1980, p. 77-80.

[28] C. R. RAO, *Linear Statistical Inference and its Applications*, 1973, Wiley.

[29] Y. ROBERT et M. TCHUENTE, Résolution systolique de systèmes linéaires denses, *RAIRO Modélisation et Analyse Numérique*, 19, n° 2, 1985, p. 315-326.

[30] R. SCHREIBER et P. KUEKES, *Systolic linear algebra machines*, in [22].

[31] S. A. TRETTER, *Introduction to Discrete-Time Signal Processing*, 1976, Wiley.

[32] H. L. VAN TREES, *Detection, estimation and modulation theory*, 1, Wiley, 1968.

[33] P. D. WELCH, The Use of FFT for the Estimation of Power Spectra, *IEEE Trans. Audio Electro.*, 15, n° 2, June 1967, p. 70-73.