Sur l'emploi des équations de Chandrasekhar pour la factorisation rapide d'une matrice proche-de-Tœplitz

Guy DEMOMENT * et Roger REYNAUD **

* Laboratoire des Signaux et Systèmes (CNRS/ESE/UPS), Plateau de Moulon, 91192 Gif-sur-Yvette Cédex, France.
** Institut d'Electronique Fondamentale (CNRS/UPS), bât. 220, Université de Paris-Sud, 91405 Orsay Cédex, France.

* et ** Gréco "Traitement du Signal et Image"

RESUME

Le problème du calcul rapide des facteurs de Cholesky d'une matrice proche-de Tœplitz se rencontre fréquemment en traitement du signal. Sa résolution fait appel à des algorithmes de Levinson, de Cybenko, ou de Schur généralisés dont les propriétés sont maintenant bien connues. Nous proposons une approche différente, fondée sur l'emploi d'équations de Chandrasekhar, et dans laquelle le vecteur des coefficients de régression à estimer est considéré comme le vecteur d'état d'un problème de filtrage de Kalman. Nous exposons le principal avantage de la méthode qui réside dans sa double filiation avec les algorithmes de Levinson et Schur d'une part, et avec l'algorithme de Kalman d'autre part, ce qui permet d'utiliser un large éventail de possibilités pour résoudre le problème de la parallélisation des calculs. Nous examinons en particulier les structures de calcul liées aux récurrences sur le temps et sur l'ordre.

SUMMARY

Fast inversion of close-to-Tœplitz matrices is a problem that frequently occurs in signal processing. The generalized Levinson, Cybenko or Schur algorithms are well known solutions to this problem. We show that a different solution, based on Chandrasekhar factorization, may be inbedded in the same vector recursion involving a *J*-orthogonal transformation. These new recursions produce an interesting way to compute the generators of a close-to-Tœplitz matrix. A particular attention is devoted to the parallelization of the resulting algorithm.

1. INTRODUCTION

Le problème du calcul rapide des facteurs de Cholesky d'une matrice proche-de-Tœplitz se rencontre fréquemment en traitement du signal, dans des problèmes d'analyse ou de synthèse de signaux, ou, plus généralement, dans des problèmes de régression linéaire [6]. Les factorisations de type Levinson, Schur ou Cybenko en sont des solutions maintenant classiques [2]. Nous présentons dans cette communication une approche différente reposant sur une factorisation du type Chandrasekhar qui présente les caractéristiques suivantes:
- la complexité des calculs est comparable à celle des méthodes existantes;
- l'établissement des équations est simple et commun à toutes les variantes;
- elle apporte un éclairage supplémentaire sur les différences entre récurrences sur le temps et sur l'ordre;
- elle fait référence à des techniques récursives du type Kalman, ce qui permet d'élargir la gamme des solutions à retenir quand, lors de la mise en œuvre, on veut paralléliser le calcul.

2. L'ALGORITHME

2.1 *Le problème.*

On veut estimer les coefficients d'une régression linéaire apparaissant dans un modèle du type:

$$y(n) = \sum_{i=1}^{n} h_i\, x(n-i) + b(n) = \boldsymbol{h}^t \boldsymbol{x}_p(n) + b(n) \quad n = 1, 2, \ldots N \quad (2.1)$$

$$\boldsymbol{h}^t = [h_p, h_{p-1}, \ldots h_1] \qquad \boldsymbol{x}_p(n)^t = [x(n-p), \ldots x(n-1)] \quad (2.2)$$

où $\boldsymbol{h}$ est le vecteur des coefficients du modèle dont l'ordre est p, et où $b(n)$ est un b.b. gaussien centré de variance σ^2. Si l'on désigne par $e(n)$ l'erreur de prédiction de $y(n)$, les coefficients $\boldsymbol{h}$ choisis minimisent le critère suivant:

$$J(\boldsymbol{h}) = \sum_{n=1}^{N} \lambda^{N-p} e(n)^2 + \lambda^N \mu \, \|\boldsymbol{h}\|^2 \quad (2.3)$$

où λ est un coefficient de pondération exponentielle ($0<\lambda\leq 1$). On est ainsi conduit à résoudre l'équation normale:

$$\{ \sum_{n=1}^{N} \lambda^{N-p} \boldsymbol{x}_p(n) \boldsymbol{x}_p(n)^t + \lambda^N \mu \boldsymbol{I} \} \boldsymbol{h} = \sum_{n=1}^{N} \lambda^{N-p} \boldsymbol{x}_p(n) y(n) \quad (2.4)$$

Le terme $\mu \, \|\boldsymbol{h}\|^2$, qui introduit une "contrainte douce" sur la solution, est souvent là pour simplifier l'initialisation. Les algorithmes rapides qui résolvent ce problème exploitent les propriétés d'invariance par décalage des vecteurs $\boldsymbol{x}_p(n)$ successifs qui font que la matrice normale à inverser:

$$\boldsymbol{R}(N) = \sum_{n=1}^{N} \lambda^{N-p} \boldsymbol{x}_p(n) \boldsymbol{x}_p(n)^t + \lambda^N \mu \boldsymbol{I} \quad (2.5)$$

est proche-de-Tœplitz. Il existe ainsi des solutions qui effectuent des récurrences sur l'ordre (Levinson ou Cybenko

généralisés) [2] et des solutions qui effectuent des récurrences sur le temps (moindres carrés récursifs rapides) [7].

2.2 *La technique.*

On peut appliquer les techniques de factorisation de Chandrasekhar à ce problème bien que le modèle d'état obtenu en considérant $\boldsymbol{h}$ comme un vecteur d'état et (2.1) comme une équation d'observation ne soit pas strictement invariant. Il faut pour cela expliciter les propriétés d'invariance par décalage de $x_p(n)$ en immergeant (2.1) dans un modèle invariant. On définit pour cela un "vecteur-paramètre étendu" à m composantes:

$$\boldsymbol{h}_m(n) = [\, \mathbf{o}^t_{n-1},\, h_p,\, h_{p-1},\ldots,\, h_1,\, 0,\, 0,\ldots,\, 0]^t \qquad m=N+p \quad (2.6)$$

et le vecteur global des données:

$$x_m = [x(-p+1),\, x(-p+2),\ldots,\, x(0),\, x(1),\ldots,\, x(m-p)]^t \quad (2.7)$$

ce qui permet de réécrire (2.1) sous la forme équivalente suivante:

$$\boldsymbol{h}_m(n+1) = \boldsymbol{D}\, \boldsymbol{h}_m(n) \qquad n = 1,\, 2,\ldots N \quad (2.8)$$

$$y(n) = x^t_m\, \boldsymbol{h}_m(n) + b(n) \quad (2.9)$$

où $\boldsymbol{D}$ est l'opérateur de décalage.

Cette technique présente une analogie certaine avec celle introduite par Morf [11] pour développer des algorithmes des moindres carrés récursifs rapides. Mais au lieu d'utiliser des quantités auxiliaires dont la dimension est simplement augmentée d'une unité par rapport à la dimension p du modèle, afin de relier deux problèmes de dimension p à deux instants consécutifs, nous introduisons un vecteur-paramètre étendu de très grande dimension. Cet artifice permet d'expliciter la propriété d'invariance que l'on cherche à exploiter. L'avantage est qu'il permet ensuite d'utiliser les mêmes équations dans tous les cas.

La principale différence avec les méthodes des moindres carrés récursifs rapides est dans le vecteur-paramètre $\boldsymbol{h}$ qui est maintenant considéré comme une quantité aléatoire caractérisée par sa moyenne et sa matrice de covariance *a priori*. Nous nous retrouvons donc dans un cadre statistique bayésien qui justifie naturellement l'emploi des "contraintes douces".

Le calcul récurrent de la solution est effectué classiquement par le filtre de Kalman suivant:

$$\hat{\boldsymbol{h}}_m(n+1/n) = \boldsymbol{D}\, \hat{\boldsymbol{h}}_m(n/n-1) + k_m(n)\, r(n)^{-1}\, [y(n) - x^t_m\, \hat{\boldsymbol{h}}_m(n/n-1)] \quad (2.10)$$

Mais la mise à jour du vecteur-gain $k_m(n)$ est faite à l'aide d'une factorisation de Chandrasekhar. Le détail des calculs est donné dans [3]. On part de la factorisation:

$$\delta \boldsymbol{P}_m(n+1) \triangleq \boldsymbol{P}_m(n+1/n) - \boldsymbol{P}_m(n/n-1) = \boldsymbol{B}(n)\, \boldsymbol{M}(n)\, \boldsymbol{B}(n)^t$$

de l'incrément de la matrice $\boldsymbol{P}_m(n+1/n)$ de covariance de l'erreur de prédiction à un pas de $\boldsymbol{h}_m(n+1)$ pour obtenir:

$$\left[\begin{array}{c|c} r(n+1) & 0 \\ \hline k_m(n+1) & B(n+1) \\ \hline 0 & \lambda^{-1} R^r(n+1) \end{array}\right] = \left[\begin{array}{c|c} r(n) & x^t_m\, B(n) \\ \hline k_m(n) & D\, B(n) \\ \hline B(n)^t\, x_m & R^r(n) \end{array}\right] \Theta_n$$

$$\Theta_n = \left[\begin{array}{c|c} I_1 & -r(n)^{-1}\, x^t_m\, B(n) \\ \hline -R^r(n)^{-1}\, B(n)^t\, x_m & I_\alpha \end{array}\right] \quad (2.11)$$

où Θ_n est une matrice J-orthogonale et où α est le rang de:

$$\delta \boldsymbol{P}_m(2) = \boldsymbol{D}\, \boldsymbol{P}_m(1/0)\, \boldsymbol{D}^t - k_m(1)\, r(1)^{-1}\, k_m(1)^t - \boldsymbol{P}_m(1/0)$$

L'initialisation se fait selon:

$$\hat{\boldsymbol{h}}_m(1/0) = [0,\ldots,\, 0]^t \quad (2.12)$$

$$r(1) = x^t_m\, \boldsymbol{P}_m(1/0)\, x_m + \sigma^2 \quad (2.13)$$

$$k_m(1) = \boldsymbol{D}\, \boldsymbol{P}_m(1/0)\, x_m \quad (2.14)$$

$$\boldsymbol{P}_m(1/0) = \left[\begin{array}{c|c} \boldsymbol{P}(1/0) & 0 \\ \hline 0 & 0 \end{array}\right] \quad (2.15)$$

où $\boldsymbol{P}(1/0) = \sigma^2/\mu \boldsymbol{I}$ par exemple.

Bien que l'algorithme (2.11) porte sur des quantités étendues, on peut vérifier qu'à chaque récurrence le nombre de coordonnées non-nulles du vecteur-gain $k_m(n)$ reste égal à p. Une écriture condensée de l'algorithme est obtenue par projection, en ne faisant apparaître que les composantes intervenant effectivement à chaque récurrence dans le calcul [3].

2.3 *Conditions initiales et rang de déplacement.*

Le volume des calculs est directement conditionné par α, c'est-à-dire à la fois par le rang de déplacement de la matrice de covariance *a priori* $\boldsymbol{P}_m(1/0)$ et par le gain initial . C'est par l'intermédiaire de $k_m(1)$, et donc de $x_p(1)$, que les différences entre versions "préfenêtrée" et "covariance" apparaissent. Cette complexité passe de $6p$ à $9p$, $12p$ et $15p$ selon que le signal est préfenêtré ou non et que les paramètres sont corrélés *a priori* ou non. Mais ce qui est remarquable, c'est qu'on utilise le même algorithme dans tous les cas.

2.4 *Interprétation.*

La forme (2.11) est intéressante car elle montre que, comme pour les algorithmes de Levinson ou de Schur généralisés, la récurrence de base fait intervenir une même transformation J-orthogonale, et ce quel que soit le type de récurrence. En effet, les équations de Levinson généralisées exprimées en dimension fixe [2] peuvent s'écrire:

$$\left[\begin{array}{c|c} \sigma^2_{i+1} & 0 \\ \hline A_{i+1} & B_{i+1} \\ \hline 0 & \sigma^2_{i+1} \Delta_{i+1} \end{array}\right] = \left[\begin{array}{c|c} \sigma^2_i & -\sigma^2_i K^t_i \\ \hline D\, A_i & B_i \\ \hline -\sigma^2_i K_i & \sigma^2_i \Delta_i \end{array}\right] \Psi_k$$

$$\Psi_k = \left[\begin{array}{c|c} I_1 & K^t_i \\ \hline \Delta^{-1}_i K_i & I_{\alpha+1} \end{array}\right] \quad (2.16)$$

où la matrice Ψ_k est aussi une matrice J-orthogonale.

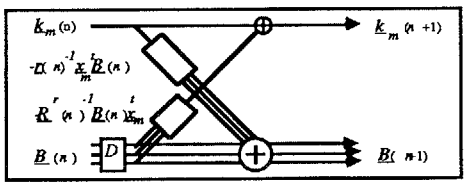

Figure 1 - Treillis vectoriel du type Chandrasekhar.

La récurrence de base d'une factorisation de Chandrasekhar est donc elle aussi du type "treillis vectoriel" (*c.f.* figure 1). Elle n'est donc pas à l'abri des problèmes d'instabilité numérique liés à ces transformations. Mais la différence la plus importante tient au fait que l'algorithme (2.11) choisit comme premier vecteur générateur de l'inverse de la matrice normale non pas le vecteur des coefficients de prédiction directe comme habituellement, mais le vecteur-gain de Kalman! Ceci est en fait logique dans un problème de régression dont le but est

avant tout de calculer ce vecteur-gain pour ensuite mettre à jour le paramètre h selon (2.10). Une conséquence en est que le facteur $B(n)$ de l'incrément de covariance qui apparaît dans ces équations de Chandrasekhar est en fait une matrice constituée des α vecteurs générateurs de $R(N)^{-1}$. La multiplicité des formes possibles des algorithmes de Chandrasekhar n'est que le reflet du manque d'unicité de la décomposition d'une matrice à l'aide de ses générateurs [4].

2.5 *Récurrence sur le temps ou récurrence sur l'ordre?*

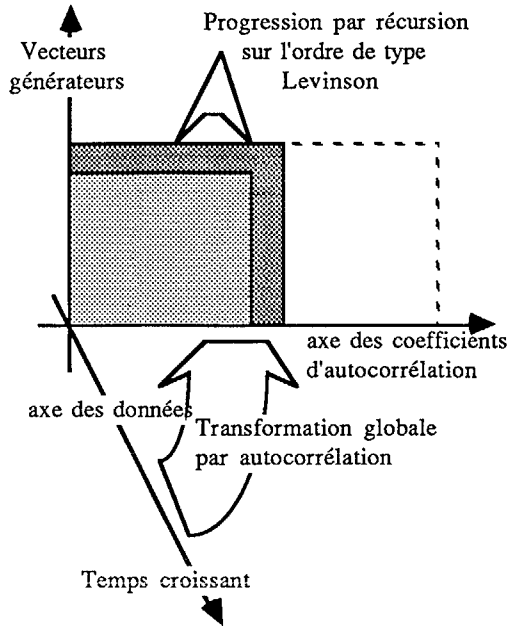

Figure 2 - Récurrence sur l'ordre.

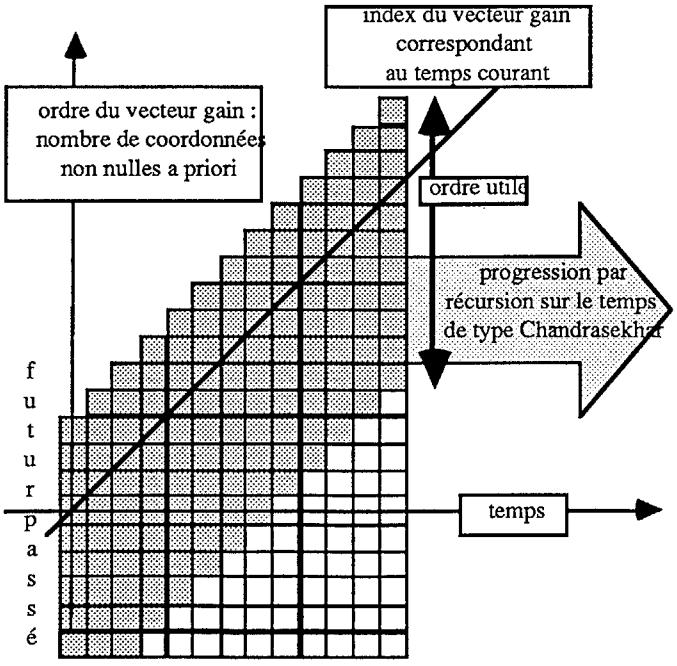

Figure 3 - Récurrence sur le temps.

L'écriture de ces algorithmes en dimension fixe montre que la différence principale n'est pas tellement dans le type de récurrence utilisé, mais plutôt dans la nature des quantités calculées. Il subsiste malgré tout une différence quant à la manière d'explorer les données. Dans le cas d'un algorithme de Levinson par exemple, on leur fait subir une première transformation pour les convertir en coefficients de la matrice normale $R(N)$. Ces coefficients sont ensuite traités séquentiellement et à cette récurrence sur le temps pour les covariances correspond une récurrence sur l'ordre des vecteurs générateurs de $R(N)^{-1}$ (*c.f.* figure 2). Les effets de bords sont donc connus avant la première récurrence.

Dans la récurrence du type Chandrasekhar, les données sont explorées directement et séquentiellement. Mais cette récurrence sur le temps s'accompagne aussi d'une récurrence sur l'ordre des vecteurs étendus (*c.f.* figure 3). Cette récurrence disparait (sauf dans le cas préfenêtré) lorsque l'on extrait, par projection, les vecteurs générateurs de $R(N)^{-1}$.

On remarque par ailleurs que, dans cette récurrence de Chandrasekhar, le nombre minimal de vecteurs générateurs que l'on doit prendre est 3. Ceci ne doit pas surprendre puisque les conditions finales ne sont pas connues à l'initialisation. On vérifie par contre qu'en cas de postfenêtrage l'un des vecteurs générateurs s'annulle. C'est justement le gain de Kalman.

3. LA MISE EN OEUVRE

Les problèmes de mise en œuvre se posent à deux niveaux: le premier concerne le choix du matériel, le deuxième porte sur le partitionnement des calculs qu'implique l'algorithme sur les différentes ressources matérielles. En fait ces différents choix sont fortement interactifs, et nous avons abordé l'étude en considérant trois options potentielles correspondant à des gammes de matériel différentes.

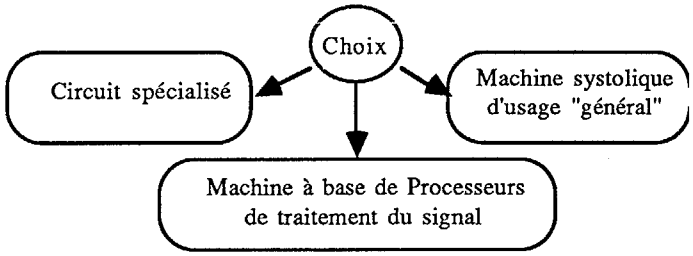

Figure 4 - Options potentielles.

3.1 *Approche intégrée spécialisée.*

Dans la première solution envisagée [1,8,10,12] l'algorithme est implanté sur un réseau systolique spécialisé. Pour un algorithme de Levinson généralisé, le nombre d'entrées des différentes cellules élémentaires est 4 (*c.f.* figure 5). Cette dimension traduit la prise en compte d'invariants au niveau du modèle [9], ce qui diminue le volume de calcul effectué dans une cellule élémentaire dès l'initialisation du problème. Ce volume reste constant au cours des itérations, ce qui est conforme à la notion de rang de déplacement d'une matrice proche-de-Tœplitz [6].

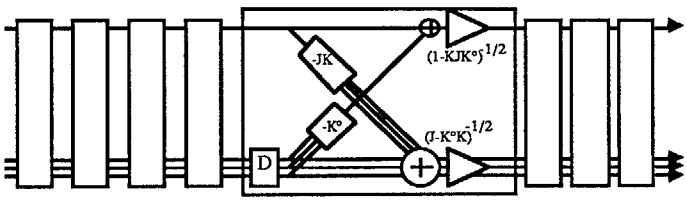

Figure 5 - Structure à n-pôles en treillis cascadés.

Le réseau est formé de cellules à coefficients constants dans le cas où la matrice de covariance du signal ou la matrice normale $R(N)$ sont connues à l'avance. Ces coefficients peuvent alors être calculés par anticipation. Mais, dans le cas contraire qui

nous intéresse ici, nous devons résoudre le problème de partitionnement suivant: la machine doit pouvoir s'adapter à des réalisations différentes des signaux $x(n)$ et $y(n)$, et les coefficients de réflexion concernant une cellule nouvelle à ajouter à la cascade de treillis doivent être calculés en ligne. Or ce calcul fait intervenir des grandeurs externes à la cellule, *i.e.* des grandeurs qui ne sont pas à proximité immédiate au regard de la structure en treillis. A ce niveau la nécessaire interaction ressource matérielle - partitionnement de l'algorithme transparaît dans la nécessité d'obtenir un calcul automatique en ligne (par opposition avec un calcul *a priori*) et passe par l'emploi de nombreux bus de données ou par l'emploi de mémoires multi-accès. Les caractéristiques de cette solution sont:
- Utilisation d'une conception descendante visant à obtenir une solution très adaptée mais peu versatile, avec un coût de développement élevé, ce qui explique l'absence de produits commerciaux.
- Une telle solution est destinée à une application fortement intégrée avec une arithmétique le plus fréquemment bit-série et cordic. Son développement passe par l'utilisation d'un compilateur de silicium dédié à la génération d'une architecture systolique. Cet objectif est habituellement beaucoup trop ambitieux vis-à-vis des investissements en jeu.

3.2 *Approche "haut-niveau".*

Dans cette approche, la mise en œuvre de l'algorithme est faite sur une machine systolique d'usage général (connection machine) ou vectorielle (Cray1). Le coût d'achat conduit en général à un rejet de cette solution pour beaucoup d'applications, mais les problèmes de partitionnement restent entiers [***5***]. Pour utiliser un pourcentage élevé de la puissance de la machine, il faut en effet limiter par partitionnement l'échange des données entre une mémorisation locale et la mémorisation globale. Les trois partitionnements habituellement proposés sont:
- par vecteurs ==> machine vectorielle,
- par bloc 2D ==> machine systolique gros grain,
- par cellule treillis ==> machines intégrées.

Le but est bien d'amener les données devant être traitées de façon conjointe devant une même unité de traitement. Mais nous manquons actuellement d'une méthode de recherche d' un partitionnement "naturel" (dans le sens langage naturel) qui prenne en compte les interprétations géométrique ou combinatoire des transformations qui entrent en jeu.

3.3 *Machine à base de quelques circuits PTS*

La solution d'une machine multi-processeurs de signal (PTS) est souvent retenue sur les critères suivants: coût des composants, coût de conception, prise en compte du savoir faire acquis, viabilité de la solution finale. La parallélisation se résume alors à trouver un découpage de l'algorithme pour l'implantation sur 3 ou 4 PTS.

Si l'on arrive à obtenir une forme treillis où tous les calculs ne nécessitent qu'un accès à des données locales pour chaque PTS, alors nous sommes en présence de la forme la plus souple, qui autorise une communication de type message (faible débit vis-à-vis de la puissance de traitement interne du PTS) et qui permet de choisir un nombre de cellules arbitraire par PTS. Dans une telle structure, le rapport charge de communication sur charge de traitement varie en sens inverse du nombre de cellules affectées par circuit PTS.

Mais il arrive fréquemment qu'une telle formulation soit impossible, et c'est le cas de l'algorithme précédemment décrit. La solution doit alors passer par une gestion de la concurrence entre les divers PTS pour accéder à une ressource commune de stockage, imposant alors soit un système de mémoire globale multi-accès, soit un système de mémoire-cache. Pour obtenir les améliorations escomptées du parallélisme, il faut alors disposer d'un système d'exploitation très efficace.

Le découpage des vecteurs de l'algorithme en blocs peut permettre de conserver une certaine forme de localité des données et donc de retomber sur des structures plus souples de gestion du parallélisme du type communication. Mais nous manquons là aussi d'une méthode de partitionnement "naturel" qui permette d'implanter un tel algorithme sur une machine faiblement parallèle.

4. CONCLUSION

La parallélisation d'un algorithme récursif en temps tel que celui décrit plus haut soulève le problème du choix d'une implantation impliquant un minimum de traitement global vis-à-vis des données. Ce problème est réglé dans les algorithmes récursifs sur l'ordre qui commencent par effectuer une transformation globale par bloc sur les données (dépendant du temps). On pourrait penser que la situation est la même pour notre algorithme qui introduit une information comparable au niveau de l'*a priori* concernant α. Mais cette information n'est pas équivalente, il s'agit d'une contrainte qui va être propagée au cours du temps et qui conserve de ce fait un caractère global. La transformation J-orthogonale correspond alors à une forme treillis vectoriel, qui ne peut se transformer en treillis scalaire à cause des dépendances vis-à-vis des données à venir.

La seule solution qui semble envisageable consiste à partitionner chaque vecteur en un nombre de sous-vecteurs égal au nombre d'unités de traitement et à reconstruire l'algorithme.

5. BIBLIOGRAPHIE

[1] P.K. Behera, "Implementation of Chandrasekhar filter using systolic arrays", in *Signal Processing IV*, J.L. Lacoume *et al.* eds, Elsevier, 1988, pp.899-902.

[2] C. Demeure and L.L. Scharf, "Vector algorithms for computing QR and Cholesky factors of Close-to-Tœplitz matrices", *Proc. IEEE ICASSP 87*, Dallas, 1987, pp.1851-54.

[3] G. Demoment, "Equations de Chandrasekhar et algorithmes rapides pour le traitement du signal et de l'image" à paraître dans *Traitement du signal*, vol. 6, 1989.

[4] G. Demoment et R. Reynaud "Equations de Schur et de Chandrasekhar généralisées pour des problèmes généraux d'estimation linéaire", *11ème Colloque GRETSI*, Nice, 1987.

[5] D.E. Foulser and R. Schreiber, "The SAXPY Matrix-1: A general purpose systolic computer", *Computer*, vol.**20**, pp35-43, 1987.

[6] C. Guéguen, "An introduction to displacement ranks and related fast algorithms", Lecture Notes, *NATO Summer School on Signal Processing*, Les Houches, 1985.

[7] S. Haykin, "Adaptative filter theory", Prentice Hall, 1986.

[8] S.Y. Kung, "VLSI signal processing from transversal filtering to concurrent array processing", in *VLSI and Modern Signal Processing*, S.Y. Kung *et al.* eds, Prentice-Hall, 1985.

[9] S.Y. Kung and J.N. Hwang, "An efficient triarray systolic design for real-time Kalman filtering", *Proc. IEEE ICASSP88*, New-York, 1988, pp.2041-2044.

[10]R.A. Lincoln and K. Yao, "Efficient systolic Kalman filtering by dependance graph mapping", in *VLSI Signal Processing III*, R.W. Brodersen & H.S. Moskovitz eds, IEEE Press, 1988, pp.396-407.

[11]M. Morf, "Fast algorithms for multivariate systems", Ph.D. Dissertation, Stanford University, 1974.

[12]T-Y. Sung and Y-H. Hu, "Parallel VLSI implementation of the Kalman filter", *IEEE Trans.*, vol. **AES-23**, pp.215-224, 1987.