IMPLANTATION DE L'ALGORITHME DES MOINDRES CARRÉS RAPIDES SUR LE PROCESSEUR SPÉCIALISÉ DSP56001

Patrice NUS, Vladimir BOCHEV
Patrick SIBILLE

Centre de Recherche en Automatique de Nancy, CRAN-CNRS URA 821
Université Henri Poincaré, Nancy 1
11 rue de l'Université, 88100 Saint DIE, France

RÉSUMÉ

Les processeurs spécialisés DSP à virgule fixe sont fréquemment utilisés en filtrage numérique ou adaptatif. Dans cette dernière classe de filtre, on trouve en particulier l'algorithme des moindres carrés rapides "MCR" dont l'implantation sur ce type de calculateur demande d'une part une version stabilisée et d'autre part des méthodes ou artifices pour éviter les dépassements dans les calculs et améliorer la rapidité du traitement. Dans cet article, on présente les méthodes utilisées dans l'implantation de cet l'algorithme sur le processeur DSP56001. Pour un compromis entre précision et rapidité de calcul, cette implantation a nécessité l'utilisation des nombres au format mixte et le développement d'un circuit de conversion de format.

ABSTRACT

Fixed point DSPs are frequently used in digital or adaptive filtering. In this last class of filter, we find the fast recursive least squares algorithm or fast Kalman algorithm. To implement it on the fixed point DSP56001 processor, a stabilised version of this algorithm is used. Moreover, to eliminate the overflow problems frequently occurring in adaptive filtering and to improve the computing time, particular techniques are also necessary. In this paper, the implementation of the stabilised fast recursive least squares algorithm is presented. Mixed format numbers are used and associated with a simple hardware aid for speeding up this particular computing situation.

1. INTRODUCTION

Le filtrage adaptatif est à la base de l'opération de modélisation des systèmes. Le filtre adaptatif est généralement réalisé à l'aide d'une structure numérique dont les caractéristiques fréquentielles sont variables de façon à les adapter à celles du signal représentatif du système inconnu. L'opération d'identification peut se représenter sous la forme suivante :

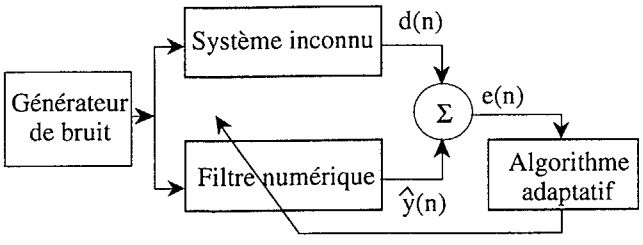

Figure 1 : Opération d'identification

Le filtre adaptatif et le système inconnu sont excités en parallèle et le signal de sortie du système inconnu (signal désiré) d(n) est comparé au signal estimé $\hat{y}(n)$ du filtre adaptatif. Le signal d'erreur e(n), représentatif de la différence entre les deux systèmes, est ensuite traité par l'algorithme adaptatif qui se charge de modifier les coefficients du filtre numérique.

L'algorithme adaptatif constitue la partie la plus importante de l'opération d'identification car il est responsable de la minimisation de l'erreur e(n) et de la qualité de l'estimation des paramètres du modèle du système inconnu.
Parmi les nombreux algorithmes possibles, nous nous sommes intéressés à ceux conduisant à une estimation en temps réel des paramètres d'un modèle autorégressif AR et plus particulièrement aux algorithmes rapides notamment celui des moindres carrés rapides "MCR" appelé encore Kalman rapide.

L'objectif de cet article est de présenter les techniques utilisées dans l'implantation de cet l'algorithme sur un processeur en virgule fixe, le DSP56001. Nous commencerons par rappeler l'algorithme des moindres carrés rapides stabilisé [1], avant d'aborder la partie implantation sur le calculateur spécialisé.

2. ALGORITHME MCR

L'algorithme des moindres carrés rapides est fondé sur l'exploitation des erreurs de prédiction a priori. Il utilise un modèle autorégressif AR où l'échantillon courant est exprimé linéairement en fonction d'un nombre fini d'échantillons précédents et du bruit e(n).

Le signal de sortie du filtre numérique adaptatif $\hat{y}(n)$ s'écrit :

$$\hat{y}(n) = \hat{H}_N^t(n).X_N(n) \qquad (2.1)$$

où $\hat{H}_N^t(n) = [h_0(n)\ h_1(n) \ldots h_{N-1}(n)]$ est le vecteur des N coefficients et $X_N(n)$ le vecteur des N échantillons d'entrée qui s'écrit $X_N^t(n) = [x(n)\ x(n-1) \ldots x(n+1-N)]$.

On définit également l'erreur d'estimation ou la différence entre le signal désiré et estimé, soit $e(n) = d(n) - \hat{y}(n)$ qu'il s'agit de minimiser, d'où le critère ou énergie de prédiction E(n) :

$$E(n) = \sum_{p=1}^{n} w^{n-p}\left[d(p) - \hat{H}_N^t(n).X_N(p)\right]^2 \qquad (2.2)$$

où w est un facteur d'oubli introduit pour donner à l'algorithme une capacité de poursuite. La dérivation de ce critère conduit à l'estimation récursive du vecteur des coefficients à l'instant n+1 donné selon :

$$\hat{H}_N(n+1) = \hat{H}_N(n) + G_N(n+1).e(n+1) \qquad (2.3)$$

où $G_N(n+1)$ est le gain d'adaptation exprimé selon $G_N(n) = R_N^{-1}(n).X_N(n)$ avec $R_N^{-1}(n)$ l'inverse de la matrice d'autocorrélation des échantillons d'entrée et $e(n+1) = d(n+1) - \hat{H}_N^t(n).X_N(n+1)$ l'erreur de prédiction a priori.

La suite du problème consiste à calculer le vecteur gain qui fait intervenir une inversion de matrice. Cette inversion peut se faire à l'aide du lemme d'inversion [2], par la factorisation (Cholesky, Schur) [3], ou par une approximation par δ.I (gradient exact) [1]. Dans le cadre des moindres carrés rapides, on cherche à exprimer le vecteur gain à travers un vecteur gain de taille N+1. Cette approche conduit à introduire les vecteurs de prédiction avant et arrière respectivement $A_N(n)$ et $B_N(n)$. Ils permettent de prendre en compte l'information gagnée par l'arrivée et la perte d'un échantillon (respectivement x(n+1) et x(n+1-N)), sous la forme des erreurs de prédiction avant et arrière a priori, $e_a(n)$ et $e_b(n)$, et a posteriori $\varepsilon_a(n)$ et $\varepsilon_b(n)$. Les énergies correspondantes à ces erreurs, soit $E_a(n)$ et $E_b(n)$ sont également à minimiser ce qui conduit à la construction de la matrice $R_{N+1}(n+1)$ et au vecteur gain $G_{N+1}(n+1)$. Le partitionnement de ce vecteur gain en un vecteur $M_N(n+1)$ de taille N, et d'un scalaire m(n+1), permet alors de définir le vecteur gain d'adaptation de taille N intervenant dans l'équation récursive (2.3).

Stabilisation de l'algorithme

Des problèmes de stabilité numérique ont été constatés pour tous les algorithmes des moindres carrés et plus particulièrement pour les formes rapides, du fait de l'élimination des calculs redondants. Cette instabilité est encore plus marquée lorsque l'implantation est effectuée sur des calculateurs dont la précision est limitée, notamment le DSP56001.

La principale source d'instabilité de l'algorithme MCR a pour origine l'accumulation des erreurs d'arrondi accentuée par une mise à jour asymétrique des vecteurs de prédiction $A_N(n)$ et $B_N(n)$. La méthode de stabilisation utilisée [4] propose d'introduire dans l'algorithme une contre-réaction permettant de contrôler le prédicteur arrière $B_N(n)$. Cette contre-réaction fait intervenir l'image de l'accumulation d'erreur d'arrondi, calculée à partir d'une autre formulation de l'erreur de prédiction arrière, soit $e_b'(n)$. Une fraction de la différence entre $e_b(n)$ et $e_b'(n)$ (pondération par la constante μ) est ensuite injectée dans l'algorithme à l'aide du prédicteur arrière $B_N(n)$.

Les équations de l'algorithme des moindres carrés rapides fondé sur les erreurs de prédictions a priori sont résumées ci-après où N est l'ordre du filtre, w le facteur d'oubli, μ le taux de contre-réaction pour la stabilisation, E_a l'énergie de prédiction avant initialisée à une faible valeur, x(n+1) le signal d'entrée et d(n+1) le signal de référence. A noter que les vecteurs A_N, B_N et H_N sont initialisés à zéro au démarrage des calculs.

Calcul du gain d'adaptation

$$\begin{aligned}
e_a(n+1) &= x(n+1) - \hat{A}_N^t(n).X_N(n) \\
\hat{A}_N^t(n+1) &= \hat{A}_N^t(n) + G_N(n).e_a(n+1) \\
\varepsilon_a(n+1) &= x(n+1) - \hat{A}_N^t(n+1).X_N(n) \\
E_a(n+1) &= wE_a(n) + e_a(n+1).\varepsilon_a(n+1) \\
G_{N+1}(n+1) &= \begin{bmatrix} 0 \\ G_N(n) \end{bmatrix} + \begin{bmatrix} 1 \\ -\hat{A}_N(n+1) \end{bmatrix} \frac{\varepsilon_a(n+1)}{E_a(n+1)} \\
&= \begin{bmatrix} M_N(n+1) \\ m(n+1) \end{bmatrix} \\
e_b(n+1) &= x(n+1-N) - \hat{B}_N^t(n).X_N(n+1) \\
e_b'(n+1) &= w^{-N}.E_a(n+1).m(n+1) \\
\Delta e_b(n+1) &= e_b(n+1) - e_b'(n+1) \\
G_N(n+1) &= \frac{M_N(n+1) + m(n+1).\hat{B}_N(n)}{1 - m(n+1)e_b(n+1)} \\
\hat{B}_N(n+1) &= \hat{B}_N(n) + G_N(n+1).\left[e_b(n+1) + \mu.\Delta e_b(n+1)\right]
\end{aligned}$$

Filtrage

$$\begin{aligned}
e(n+1) &= d(n+1) - \hat{H}_N^t(n).X_N(n+1) \\
\hat{H}_N(n+1) &= \hat{H}_N(n) + G_N(n+1).e(n+1)
\end{aligned}$$

Ce sont ces différentes équations qui constituent l'algorithme MCR et qui sont implantées sur le DSP56001 dans l'ordre ci-dessus.

3. IMPLANTATION

La principale difficulté rencontrée lors de l'implantation de l'algorithme MCR sur le DSP56001 est relative à la gamme arithmétique $[-1, +1-2^{-23}]$ imposée par le codage en virgule fixe sur 24 bits de ce processeur. En effet, des variables de

l'algorithme sont susceptibles de dépasser ces limites notamment lors de la phase transitoire d'adaptation.
Il nous faut donc user d'un artifice afin d'élargir cette gamme arithmétique sans augmenter de manière importante le temps d'exécution de l'algorithme.

La méthode utilisée pour cette implantation sur le DSP56001, repose sur l'exploitation des nombres au format mixte [5]. Un nombre codé dans le format mixte est constitué du bit de signe "S", d'une partie entière "e" et d'une partie fractionnaire "f" que nous avons choisi respectivement pour notre application de 1, 7 et 16 bits, soit au total 24 bits. La gamme arithmétique permet alors de coder un nombre R dans les limites $-128 \leq R \leq +127.99999$. Ce format réduit les risques de dépassement tout en présentant néanmoins une précision suffisante (16 bits). Cependant, les opérations arithmétiques dans ce format mixte entraînent des calculs supplémentaires notamment à la suite de l'opération de multiplication schématisée sur la figure suivante :

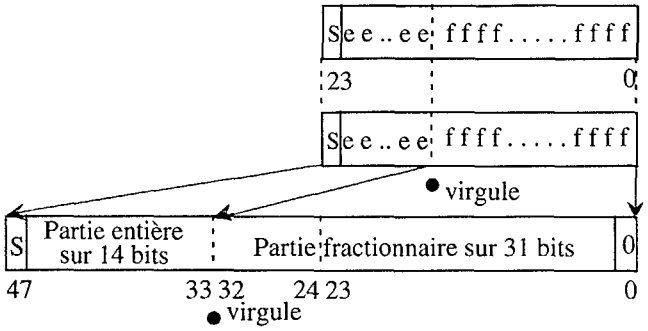

Figure 2 : Multiplication de 2 nombres au format mixte

Le résultat de la multiplication est constitué d'une partie fractionnaire codée sur 32 bits dont le premier bit est à zéro, et d'une partie entière sur 14 bits. Pour parvenir au résultat désiré dans le format mixte que nous avons adopté, il faut décaler le résultat de 7 bits vers la gauche en préservant le bit de signe et ne prélever que les 24 bits de poids forts. Le DSP56001 ne disposant pas d'une fonction de décalage par bloc, celle-ci doit être réalisée par 7 décalages successifs entraînant 7 cycles machine à chaque multiplication. Il en résulte que la durée de calcul de l'algorithme est importante puisque la complexité obtenue est alors de 115.N + 206 cycles machine (N est l'ordre du filtre), limitant ainsi la fréquence d'utilisation.

Pour réduire de façon significative cette complexité numérique, on propose de remplacer les décalages successifs réalisés par le DSP, par un système de conversion externe qui permet de sélectionner automatiquement les bits nécessaires au format mixte [6]. Pour ce faire, 4 bascules D sont utilisées pour mémoriser le résultat de la multiplication et la lecture par le DSP de cette mémoire, permet de restituer le bon format par sélection automatique des bits pertinents.

Le schéma électrique de ce système est présenté sur la figure suivante :

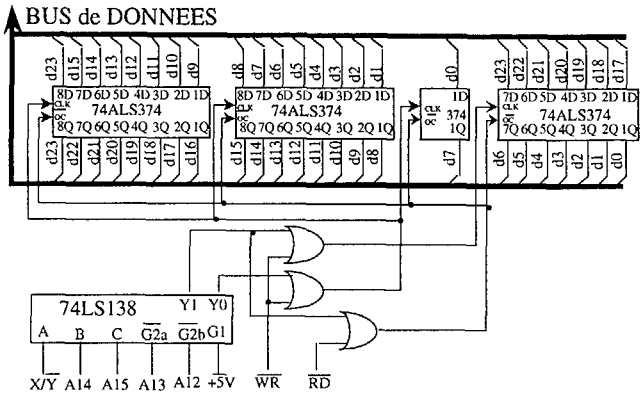

Figure 3 : Système de conversion

Ce circuit permet de réduire les 7 opérations de décalage à seulement 2 transferts entre l'accumulateur du processeur qui contient le résultat brut de la multiplication et les bascules D.
Dans l'exemple ci-après, on montre premièrement l'opération de multiplication de 2 nombres contenus dans les registres x0 et x1 du DSP56001 et la mise au format mixte de manière classique par 7 décalages successifs :

```
                                        mots prog.   Nb cycles
mpy     x0,x1,a   ; x0.x1--> a          1
rep     #7        ; répéter 7 fois      1            2
asl     a         ; le décalage sur a   1            7
```

Ce programme assembleur devient le suivant lorsque l'on exploite le système externe de conversion :

```
                                        mots prog.   Nb cycles
mpy     x0,x1,a   ; x0.x1--> a          1
move    a,l:(r1)  ; transfert dans D    1            2
move    x:(r1),a  ; lecture D           1            1
```

Ainsi, les 9 cycles machine du premier programme, se réduisent à 3 cycles pour le second, soit un gain de vitesse pour l'opération de mise au format de l'ordre de 60 %. La complexité de calcul de l'algorithme devient alors de 55.N + 206 lorsque l'on utilise le système de conversion, ce qui autorise une fréquence d'échantillonnage maximale de 8500 Hz, pour un temps de cycle du processeur de 90 ns et un ordre N de 20.

Organisation de la mémoire du DSP56001

La rapidité de calcul du programme, écrit en assembleur, repose sur une gestion optimale des pointeurs du processeur. Ainsi chaque vecteur de l'algorithme est mémorisé dans une mémoire de taille modulo N ou N+1, à laquelle on associe un pointeur choisi judicieusement, afin de permettre le transfert des données en parallèle sur les champs mémoires X et Y du DSP. L'organisation adoptée est résumée dans le tableau suivant :

Pointeur	Taille	Champ mémoire	Définition
R0	N+1	X	X(n) et X(n+1)
R2	N	X	G et m
R4	N	Y	A
R5	N	Y	B
R6	N	Y	H

Tableau 1 : Définition des zones mémoires du DSP

Quant au matériel utilisé, il se compose du DSP56001 (on utilise les champs mémoires internes), d'un convertisseur analogique numérique, du système de conversion de format et d'une liaison série (disponible dans le DSP) chargée de transmettre vers l'extérieur les coefficients calculés.

4. STABILITÉ DE L'ALGORITHME

La méthode de stabilisation utilisée a pour objectif d'éviter la divergence à long terme de l'algorithme. Pour visualiser la robustesse de l'algorithme implanté sur le DSP56001, nous avons choisi comme indicateur de stabilité, la variable de vraisemblance $\theta(n)$ représentant le rapport $\varepsilon_a(n) / e_a(n)$ des erreurs de prédiction avant a posteriori et a priori. Cette variable est comprise entre 0 et 1 si l'algorithme reste stable [1]. Le signal d'excitation x(n) est un bruit blanc gaussien centré de variance unité et les conditions de fonctionnement, à la limite de la stabilité sont : w = 0.958, l'énergie initiale $E_a(0)$ = 5e-4 et l'ordre du filtre est fixé à N = 10.

Les deux courbes présentées ci-après illustrent respectivement le comportement de l'algorithme sans stabilisation et avec stabilisation.

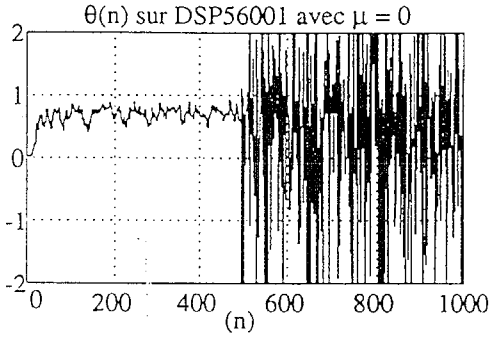

Figure 4 : Algorithme MCR sans stabilisation

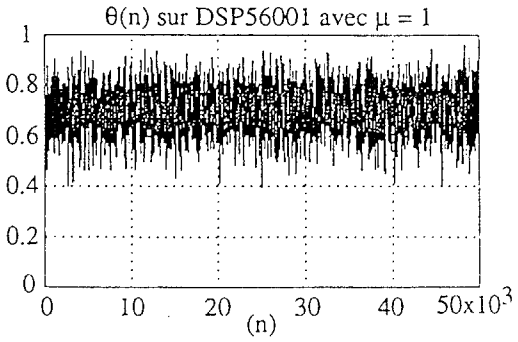

Figure 5 : Algorithme MCR avec stabilisation

La figure 4 montre que, sans contrôle des erreurs d'arrondi, (la constante de contre-réaction vaut $\mu = 0$) la divergence se produit rapidement après seulement 500 itérations. Par contre, lorsque la stabilisation est active (figure 5), aucun signe de divergence n'est observé pendant les 50000 itérations de l'algorithme, ce qui correspond à un fonctionnement pendant 10 secondes (fréquence d'échantillonnage de 5 kHz).

5. CONCLUSION

L'implantation de l'algorithme des moindres carrés rapides sur le DSP56001 nécessite l'utilisation des nombres au format mixte ainsi qu'un système de conversion. Cette approche permet, d'une part d'élargir la gamme arithmétique afin de réduire les risques de dépassement sur certaines variables de l'algorithme, et d'autre part d'accélérer le temps de traitement. Le fonctionnement de l'algorithme stabilisé sur le processeur est tout à fait convenable, on notera néanmoins que la complexité de calcul (55.N + 206) reste élevée réduisant ainsi la fréquence d'échantillonnage possible et par la même les domaines d'application. Il faut cependant préciser que la vitesse des processeurs de la famille DSP56K devient de plus en plus élevée et que le dernier calculateur commercialisé, le DSP56002, fonctionne avec un temps de cycle pouvant aller jusque 40 ns, ce qui permet de doubler la fréquence d'échantillonnage. (Le programme sur les DSP56001 et DSP56002 est disponible auprès du premier auteur).

BIBLIOGRAPHIE

[1] M. BELLANGER, "Analyse des signaux et filtrage numérique adaptatif". Masson, Collection technique et scientifique des télécommunications, 1989.

[2] T. KAILATH, "Linear systems". Prentice Hall, Englewood Cliffs, New Jersey, 1980.

[3] S. L. MARPLE, "Digital spectral analysis with applications". Prentice Hall, Englewood Cliffs, New Jersey, 1987.

[4] BENALLAL, GILLOIRE, "A new method to stabilize fast RLS algorithms based on a first-order model of the propagation of numerical errors". IEEE Proc. ICASSP, New-York, 1988.

[5] A. CHRYSAFIS, S. LANSDOWNE, "Fractional and integer arithmetic using the DSP56000 family of general-purpose digital signal processors". Note d'application APR3/D, Motorola Inc., 1991.

[6] P. NUS, V. BOCHEV, "A Hardware add-on for mixed format multiplication on the DSP56001". EDN Magazine, DI# 1713, USA, 1995.