MISE EN ŒUVRE DE L'ALGORITHME FTF STABILISE SUR UN PROCESSEUR SPECIALISE 16 BITS A VIRGULE FIXE

B. Joseph[1], P. Baylou[1], M. Najim[1], and D. Aboutajdine[2]

1. ENSERB. 351, Cours de la Libération 33405 Talence Cedex
2. ENSIAS-LEESA. Rabat. Maroc

Cette contribution porte sur l'implémentation de la version stabilisée de l'algorithme rapide FTF (Fast Transversal Filter) des moindres carrés récursif, présentée récemment (1988) par D.T.M.Slock et T.Kailath.
Après avoir rappelé les fondements de cette version, nous développons la méthodologie utilisée pour simuler le comportement de l'algorithme sur le processeur spécialisé de traitement du signal TS 68930 (SGS.Thomson), à virgule fixe sur 16 bits. Cette simulation est effectuée en langage évolué, sur des variables réelles, à virgule flottante, ce qui facilite la comparaison et l'interprétation des différents résultats obtenus. Ces résultats montrent, que cette nouvelle version de l'algorithme FTF est numériquement stable dans le temps et que ses performances, tant en ce qui concerne sa vitesse de convergence que sa capacité de poursuite, sont conservées .

The present paper deals with the implementation of a stabilized version of the Fast Transversal Filter, recently (1988) presented by D.T.M. Slock et T. Kailath. We present a method for implementing this algorithm on a 16 bits word digital signal processor: the TS 68930 of SGS.Thomson. The reported results show that the implemented algorithm is numerically stable and keeps its performances of convergence and tracking.

1. INTRODUCTION

Les techniques d'identification adaptatives en traitement numérique du signal sont aujourd'hui largement utilisées dans le domaine des communications numériques, tant pour l'égalisation que pour l'annulation d'échos. Les algorithmes adaptatifs du type gradient stochastique (LMS) [1] constituent une classe particulièrement populaire dans ce domaine, mais, en raison de leur lenteur de convergence, et de leur sous-optimalité, les techniques des moindres carrés récursifs, du filtre de Kalman, ou récemment, l'utilisation des approches de Mendel [2] sur l'estimation de l'entrée de systèmes linéaires par lissage [3], constituent une alternative à la technique du gradient stochastique. Les défauts de la méthode des moindres carrés récursifs sont la complexité de calcul et la divergence à plus ou moins long terme, qui pose le problème de l'instabilité numérique, qui est développé plus loin.

Les algorithmes des moindres carrés récursifs

T. Kailath et J.M. Cioffi présentent ces algorithmes de la manière suivante (figure1): soient les deux signaux d(T) et y(T) corrélés entre eux. On veut construire une estimation linéaire de l'échantillon courant d(T), à partir des échantillons d'entrée y(T), sous la forme:

$$\hat{d}(T) = \mathbf{w}^t(T/N) \cdot \mathbf{Y}(T/N)$$

où $\mathbf{Y}(T/N)$ et $\mathbf{w}(T/N)$ représentent respectivement les vecteurs colonne constitués des N échantillons les plus récents de y(T), et le vecteur paramètre des coefficients à estimer, soit:

$$\mathbf{Y}(T/N) = [\, y(T)\ y(T-1)\ \ldots\ldots\ y(T-N+1)\,]^t$$

On utilisera le critère suivant des moindres carrés:

$$\xi(T/N) = \sum_{t=0}^{T} \lambda^{T-t} \cdot [d(t) - \mathbf{w}^t(T/N) \cdot \mathbf{Y}(t/N)]^2$$

où λ est le facteur d'oubli. Les signaux sont supposés nuls, avant l'instant t=0.
En utilisant le filtre de Kalman, Godard [4] a déterminé l'estimation du vecteur paramètre $\mathbf{w}(T/N)$, suivant la relation:

$$\mathbf{w}(T/N) = \Phi^{-1}(T/N) \cdot \mathbf{p}(T/N)$$

où $\Phi(T/N)$ est la matrice d'autocorrélation définie par:

$$\Phi(T/N) = \sum_{t=0}^{T} \lambda^{T-t} \cdot \mathbf{Y}(t/N) \cdot \mathbf{Y}^t(t/N)$$

et $\mathbf{p}(T/N)$ le vecteur de corrélation, défini par:

$$\mathbf{p}(T/N) = \sum_{t=0}^{T} \lambda^{T-t} \cdot d(t) \cdot \mathbf{Y}(t/N)$$

Le nombre d'opérations par itération est proportionnel à N^2, ce qui rend difficile l'utilisation de cet algorithme en temps réel, pour un filtre d'ordre élevé.
Godard a déduit, de l'algorithme précédent, une relation récursive pour le calcul du vecteur paramètre $\mathbf{w}(T/N)$:

$$\mathbf{w}(T/N) = \mathbf{w}(T-1/N) + \varepsilon^p(T/N) \cdot \Phi^{-1}(T/N) \cdot \mathbf{Y}(T/N)$$

où $\varepsilon^p(T/N)$ est l'erreur d'estimation à l'instant T, calculée avec le vecteur d'estimation à l'instant T-1, soit:

$$\varepsilon^p(T/N) = d(T) - \mathbf{w}^t(T-1/N).\ \mathbf{Y}(T/N)$$

Morf, Ljung, et Falconer [5] ont développé une version dite "algorithme de Kalman rapide", pour la mise à jour du vecteur "gain de Kalman", défini par la relation suivante:

$$\mathbf{c}(T/N) = \Phi^{-1}(T/N).\ \mathbf{Y}(T/N)$$

Le nombre d'opérations par itération est de l'ordre de 10N.
J.M. Cioffi et T. Kailath [6] ont proposé une autre version de cet algorithme, où le nombre d'opérations par itération est réduit à 7N. Ces deux derniers algorithmes, [5] et [6], utilisent les filtres prédicteurs progressif et rétrograde, représentés respectivement par les vecteurs $\mathbf{a}(T/N)$ et $\mathbf{b}(T/N)$. Leur entrée est le signal y(T), et leurs sorties respectives sont les erreurs de prédiction progressives et rétrogrades e(T/N) et r(T/N), dont les énergies respectives $\alpha(T/N)$ et $\beta(T/N)$ sont minimales. Les expressions des erreurs de prédiction sont:

$$\begin{aligned} e(T/N) &= y(T) - \mathbf{a}^t(T/N)\ .\ \mathbf{Y}(T-1/N) \\ r(T/N) &= y(T-N) - \mathbf{b}^t(T/N)\ .\ \mathbf{Y}(T-1/N) \end{aligned}$$

Ils utilisent également la quantité suivante:

$$\gamma(T/N) = 1 + \mathbf{c}^t(T/N)\ .\ \mathbf{Y}(T/N)$$

qui permet de calculer directement les erreurs de prédiction a posteriori, à partir des erreurs de prédiction a priori.
Ces deux algorithmes présentent cependant une instabilité numérique au cours du temps.
Plusieurs travaux récents [7], [8], et [9], proposent des versions stabilisées de cet algorithme. Ainsi, J.L. Botto [9] analyse, dans un premier temps, le système de propagation des erreurs numériques, en définissant d'abord le vecteur d'état $\chi(T)$ de la manière suivante:

$$\chi(T) = [\Delta\alpha(T/N)/\alpha(T/N) \quad \Delta\gamma(T/N)/\gamma(T/N) - \Delta\alpha(T/N)/\alpha(T/N)]^t$$

où Δx est l'écart numérique entre la valeur théorique x et sa valeur calculée x.
L'évolution temporelle du vecteur d'état $\chi(T)$ montre que le système est instable. Le procédé de stabilisation consiste à introduire des redondances dans le calcul de la variable $r^p(T/N)$ erreur de prédiction rétrograde a priori.
Gilloire et Benallal [8] utilisent un vecteur d'état différent, pour l'analyse du système de propagation des erreurs, soit:

$$\mathbf{Z}(T) = [d\mathbf{a}^t(T/N)\ d\alpha(T/N)\ d\mathbf{c}^t(T/N)\ d\gamma(T/N)\ d\mathbf{b}^t(T/N)\ d\beta(T/N)]$$

où $d\,x$ est l'approximation du premier ordre de l'erreur sur la variable théorique x.
La redondance, vue précédemment, pour le calcul de $r^p(T/N)$ est utilisée pour le calcul des variables $\gamma(T/N)$, $\beta(T/N)$, et $\mathbf{b}(T/N)$ de l'algorithme, et permet de contrôler la stabilité numérique de ces mêmes variables critiques. Les simulations de cette version [8], réalisées en virgule flottante, n'ont pas montré d'instabilité numérique.
D.T.M. Slock et T. Kailath [7] utilisent le même vecteur d'état que Gilloire et Benallal. Par contre, ils ajoutent d'autres redondances, notamment pour le calcul de la variable $\gamma(T/N)$. Analysons de plus près l'approche de leurs travaux.

2. LA STABILISATION DE L'ALGORITHME FTF

D.T.M. Slock et T. Kailath montrent d'abord, à partir de l'analyse de la propagation des erreurs, que l'algorithme FTF diverge de manière exponentielle, au cours du temps. En introduisant des redondances dans le calcul de plusieurs variables, ils peuvent alors contrôler la propagation des erreurs dans le temps. La complexité de l'algorithme FTF stabilisé obtenu reste proportionnelle à N, qui est l'ordre du filtre adaptatif utilisé.

2.1. Analyse de la propagation des erreurs

Kailath et Slock considèrent les algorithmes adaptatifs numériques comme des systèmes dynamiques non-linéaires:

$$\Theta(T) = f(\Theta(T-1),\ d(T),\ \mathbf{Y}(T/N)) \qquad (1)$$

où $\Theta(T)$ est le vecteur des variables de l'algorithme calculées récursivement, et $\mathbf{Y}(T/N)$ est le vecteur ligne des N échantillons les plus récents du signal d'entrée y(T).
L'implémentation de l'algorithme sur un processeur entraîne une perte de précision des variables calculées. L'algorithme va donc évoluer avec des variables approchées, représentées par le vecteur $\hat{\Theta}(T)$. Son comportement peut être représenté par:

$$\hat{\Theta}(T) = f(\hat{\Theta}(T-1),\ d(T),\ \mathbf{Y}(T/N)) + V(T) \qquad (2)$$

où V(T) représente le vecteur bruit des erreurs d'arrondis.
Soit le vecteur $\Delta\Theta(T) = \hat{\Theta}(T) - \Theta(T)$. Linéarisons le système (2) autour de la trajectoire exacte de $\Theta(T)$. Nous obtenons:

$$\Delta\Theta(T) = \Delta\Theta(T-1)\ .\ F(T) + V(T) \qquad (3)$$

où

$$F(T) = \nabla_{\Theta(T)} f(\Theta, d(T), Y(T/N))$$

Si le système (3) est stable, alors l'algorithme sera stable, autour de sa trajectoire exacte $\Theta(T)$. Mais, s'il est instable, l'algorithme divergera.
L'intérêt majeur est l'étude de la propagation du vecteur erreur $\Delta\Theta(T)$, c'est-à-dire du système:

$$\Delta\Theta(T) = \Delta\Theta(T-1)\ .\ F(T) \qquad (4)$$

Le vecteur $\Theta(T)$ est choisi de la manière suivante:

$$\Theta(T) = [-\mathbf{a}^t(T/N)\ \alpha(T/N)\ -\mathbf{b}^t(T/N)\ \beta(T/N)\ \tilde{\mathbf{c}}^t(T/N)\ \gamma^{-1}(T/N)]$$

Les variables $\alpha(T/N)$ et $\beta(T/N)$ sont les énergies des erreurs de prédictions progressives e(T/N), et rétrogrades r(T/N), et $\gamma^{-1}(T/N)$ est défini à partir du vecteur "gain de Kalman" modifié, telles que:

$$\begin{aligned} e(T/N) &= \mathbf{A}^t(T/N).\ \mathbf{Y}(T/N+1) \\ r(T/N) &= \mathbf{B}^t(T/N).\ \mathbf{Y}(T/N+1) \\ \gamma^{-1}(T/N) &= 1 - \tilde{\mathbf{c}}^t(T/N)\ .\ \mathbf{Y}(T/N) \end{aligned}$$

où $\mathbf{A}(T/N)$ et $\mathbf{B}(T/N)$ sont définis respectivement par $\mathbf{A}=[1\ -\mathbf{a}^t]^t$ et $\mathbf{B}=[-\mathbf{b}^t\ 1]^t$. On trouvera en [7] l'analyse complète du système (4), dans le cas de la version stabilisée de l'algorithme FTF.

2.2. La stabilisation de l'algorithme FTF

L'idée est de calculer certaines variables de deux manières différentes. Si la précision du calculateur était infinie, les deux calculs conduiraient au même résultat. Lorsqu'on implante l'algorithme sur un processeur, on note une différence entre les deux résultats, due aux erreurs numériques de troncatures et d'arrondis. Ces signaux différences sont des signaux de sortie pour le système de propagation des erreurs. On peut donc les envoyer à travers une boucle de retour, afin de contrôler la propagation des erreurs. La difficulté consiste à déterminer la structure de cette boucle d'asservissement. Slock et Kailath utilisent la différence entre les deux résultats de la variable, pour le calcul d'autres variables qui en dépendent.
Il y a deux possibilités pour le calcul de $r^p(T/N)$, erreur de prédiction rétrograde a priori. Soit on utilise un produit de convolution, puisque $r^p(T/N)$ est le résultat d'un filtrage linéaire; on la note $r^p{}_c(T/N)$:

$$r^p{}_c(T/N) = \mathbf{B}^t(T-1/N).\ \mathbf{Y}(T/N+1)$$

ou bien, on calcule $r^p(T/N)$ de la même façon, que dans l'algorithme FTF instable [6] , soit :

$$r^p{}_s(T/N) = -\lambda.\ \beta(T-1/N).\ [\tilde{\mathbf{c}}(T/N+1)]_{N+1}$$

Le calcul de $[\tilde{\mathbf{c}}(T/N+1)]_{N+1}$ peut s'effectuer de deux manières, soit:

$$[\tilde{\mathbf{c}}_c(T/N+1)]_{N+1} = -\lambda.\ \beta^{-1}(T-1/N).\ r^p{}_c(T/N)$$

soit, en utilisant la relation de mise à jour du vecteur "gain de Kalman" pour sa $(N+1)^{ième}$ composante:

$$[\tilde{\mathbf{c}}_s(T/N+1)]_{N+1} = [\tilde{\mathbf{c}}(T-1/N)]_N + [\tilde{\mathbf{c}}(T/N+1)]_1.\ [\mathbf{A}(T-1/N)]_N$$

De même, $\gamma^{-1}(T/N)$ peut être calculé de deux façons différentes: soit comme la sortie d'un filtre linéaire:

$$\gamma_c^{-1}(T/N) = 1 - \tilde{\mathbf{c}}^t(T/N).\ Y(T/N)$$

soit, en utilisant les deux formules de mise à jour de $\gamma(T/N)$ suivantes:

$$\gamma^{-1}(T/N+1) = \gamma^{-1}(T-1/N) - [\tilde{\mathbf{c}}(T/N+1)]_1.\ e^p(T/N)$$

et

$$\gamma_S^{-1}(T/N) = \gamma^{-1}(T/N+1) + [\tilde{c}(T/N+1)]_{N+1} \cdot r^P(T/N)$$

Il a été introduit, dans ce nouvel algorithme, deux produits de convolution, ce qui ajoute 2N additions et multiplications supplémentaires par itération.
Les variables définitives sont calculées sous la forme d'une combinaison des deux résultats obtenus pour leur calcul, soit:

$$r^P(T/N) = r^P_S(T/N) + K.\,(\,r^P_C(T/N) - r^P_S(T/N)\,)$$

où K est le gain de la boucle d'asservissement.
Le même type de combinaison est calculé pour les variables $\gamma^{-1}(T/N)$ et $[\tilde{c}(T/N+1)]_{N+1}$.
Dans l'algorithme, la variable $r^P(T/N)$ est utilisée pour le calcul de différentes variables ; selon la variable où intervient la valeur $r^P(T/N)$, Slock et Kailath utilisent différentes valeurs de gain K, ceci afin de mieux contrôler le système (4) de propagation des erreurs. L'algorithme FTF stabilisé du tableau1 prend en compte les différentes boucles d'asservissement. Si tous les Ki (pour i=1, 5) sont nuls, nous retrouvons l'algorithme FTF (7N) instable.

2.3. Choix optimal des K_i

Le choix des valeurs des Ki est d'une grande importance pour le contrôle de la stabilité numérique des variables. L'étude de Kailath et Slock conduit aux résultats suivants.
K_5 est d'une importance seconde.
K_4 est aussi non nécessaire pour la stabilisation, mais ajoute seulement une liberté supplémentaire dans le développement de la dynamique du système d'erreurs.
$K_3 = 0$ mène à un algorithme 8N, et à une évolution aléatoire pour $\Delta\gamma^{-1}(T/N)$; ceci est confirmé par les simulations, qui montrent que les erreurs croissent parfaitement de façon linéaire avec le temps, sur des millions d'itérations. L'explosion finale peut prendre un temps très long, mais elle se produira. La solution proposée par J.L. Botto [9] correspond à cet algorithme 8N (où le paramètre ρ est relié à K_1).
En général, les Ki optimaux sont fonction de λ, N, et des caractéristiques du signal d'entrée y(T). Le choix optimal pour K_4 apparaît être un problème délicat. Plusieurs valeurs de K_4, significativement différentes de zéro, semblent faire, que les valeurs optimales pour les autres Ki dépendent largement des caractéristiques du signal. Aussi, $K_4 = 1$ mène à des performances moins bonnes que $K_4 = 0$, ou même à l'instabilité, pour des signaux où $B_N \neq 0$. Nous choisirons $K_4 \approx 0$, et ainsi les valeurs optimales de K_1, K_2, K_3 sont alors pratiquement indépendantes du signal.
Afin de déterminer les paramètres Ki optimaux, lorsque le signal d'entrée est un bruit blanc, Slock et Kailath ont mis au point un programme de calcul, qui minimise la somme suivante:

$$\sum_{T=0}^{T1} \{\, (r^P_S(T/N) - r^P_C(T/N))^2 + (\,\gamma_S^{-1}(T/N) - \gamma_C^{-1}(T/N))^2 \,\}$$

où $T1 = 10^5$ est fonction de N.
$K_4=0$ et $K_5=1$ ont été fixés, et nous cherchons, à déterminer K_1, K_2, et K_3. Les tests ont été effectués pour $N=2^i$, où i=0,1,....7, et $\lambda=1-(j/10N)$, où j=0,1,....10. Les résultats indiquent, que la stabilité peut être atteinte pour tout N, et $\lambda \in [1-1/2N,1]$. Le critère n'est pas très sensible à la variation de K_2, et K_3; le choix de $K_3=1$ sera convenable dans tous les cas. Pour $\lambda=0.98$ et N=20, les valeurs de K_2 seront choisies dans l'intervalle [0.6,10], et $K_2=2.5$ conviendra dans tous les cas.
Par contre, la fonction critère est beaucoup plus sensible à la variation de K_1. La valeur $K_1=1.6$ donne de bons résultats. Nous donnons sur la figure2 les résultats de nos simulations, en virgule flottante sur 32 bits, pour les valeurs suivantes des Ki:

$$K_1=1.6,\ K_2=2.5,\ K_3=1,\ K_4=0,\ K_5=1.$$

Nous abordons maintenant, le problème de la simulation du comportement de cet algorithme sur le processeur spécialisé.

3. SIMULATION DU COMPORTEMENT DU PROCESSEUR SPECIALISE SUR UN CALCULATEUR A VIRGULE FLOTTANTE

Nous avons simulé le comportement du processeur spécialisé sur un calculateur à virgule flottante (MicrovaxII); ceci est une première étape afin de déterminer le codage des différentes variables, avant l'implémentation de l'algorithme FTF stabilisé [7] sur le processeur spécialisé. Elle permet ainsi de montrer, que cette version reste numériquement stable, lorsque ces variables sont représentées sur des mots de 16 bits, à virgule fixe.
Le processeur spécialisé est muni d'un multiplieur complexe et d'une unité arithmétique et logique ULA, qui, organisés en une structure "pipeline", opèrent parallèlement en un seul cycle d'horloge.
Rappelons brièvement, que sur un processeur spécialisé, chaque variable x est représentée par le contenu x_r d'un mot 16 bits, à un facteur multiplicatif près 2^m, soit:

$$x = 2^m \cdot x_r$$

où m entier positif, ou négatif.

La valeur de m sera fixée par avance, en fonction de l'intervalle de variation de la variable, déterminé lors des simulations en virgule flottante.
Pour simuler des variables 16 bits à virgule fixe sur des mots de 32 bits à virgule flottante, nous introduisons des opérateurs arithmétiques d'arrondi et de troncature.

3.1. L'opérateur de troncature

Avec la représentation réelle simple précision, une variable x_l est représentée, a priori, avec 24 bits significatifs. Afin que cette représentation corresponde à celle adoptée sur le processeur spécialisé, nous opérons sur x_l la transformation:

$$\mathrm{Int}\,(\,x_l \cdot 2^{15-m}\,) \cdot 2^{-(15-m)}$$

où Int(x) est la partie entière de x.
Cette transformation est la troncature de la variable x_l.

3.2. L'opérateur d'arrondi

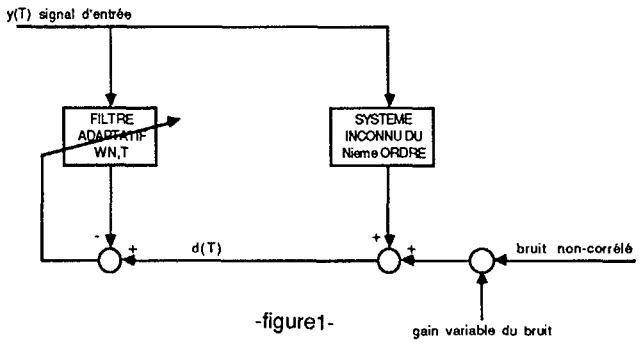
L'arrondi d'une variable x_l s'effectue en lui ajoutant la valeur $2^{-(15-m+1)}$ avant la troncature.
Nous disposons à ce stade, de tous les outils arithmétiques nécessaires pour la simulation. Afin de réaliser le codage le mieux adapté à chacune des variables, nous devons tester leur dynamique avec beaucoup d'attention.

3.3. La dynamique des variables

L'objet de cette étude est d'évaluer l'intervalle de variation de chacune des variables de l'algorithme, afin d'associer à chaque variable, la valeur de m la plus adéquate. En effet, plus m sera choisi faible, et plus élevée sera la précision de cette variable au cours des calculs. Par contre, les risques "d'overflow" augmentent parallèlement. Un compromis pour le choix de m doit assurer une bonne précision de la variable, mais aussi, une représentation correcte de sa dynamique.
Afin d'obtenir un codage convenable des variables, nous avons utilisé deux méthodes:
1) D'abord, nous pouvons limiter, autant que possible, la variation temporelle de la dynamique de chacune des variables. Ceci est rendu possible, en modifiant la valeur δ, énergie initiale des erreurs de prédiction progressives $\alpha(T/n)$ (cf.table1). Lors des simulations en virgule flottante, nous avons déterminé expérimentalement la valeur optimale de δ, notée δ_{opt}, qui correspond à la vitesse de convergence de l'algorithme la plus élevée, et qui minimise l'écart quadratique moyen durant la phase permanente. Cependant, avec cette valeur δ_{opt}, la dynamique des variables varie énormément de la phase d'initialisation, à la phase permanente. Nous notons qu'un choix de δ, supérieur à δ_{opt}, diminue la variation de la dynamique, facilitant par là-même, le codage des variables. Nous analyserons, dans le quatrième paragraphe, l'influence de δ sur les performances de l'algorithme.
2) La deuxième méthode consiste à partager le temps en plusieurs phases, afin qu'à l'intérieur d'une même phase, la dynamique des variables varie peu avec le temps. Le codage des variables sera différent suivant la phase, de manière à conserver toujours un maximum de bits significatifs (sur les 16 de disponibles) pour la représentation de chacune des variables. Aussi, ce nombre de phases doit être de l'ordre de 2 ou 3, afin de ne pas compliquer exagérément la programmation finale sur le processeur spécialisé.
Ce découpage en phases doit être opéré avec soin, car la précision des variables en dépend.
Nous reviendrons sur ce point, dans le paragraphe suivant, qui énonce l'ensemble des résultats obtenus, lors des différentes simulations.

4. RESULTATS EXPERIMENTAUX

Dans ce paragraphe, nous présentons quelques résultats expérimentaux, afin de vérifier la dépendance de la dynamique des variables par rapport au choix de δ, valeur initiale de l'énergie des erreurs de prédiction progressives, mais aussi, le comportement numériquement stable de la version stabilisée de l'algorithme FTF, implémentée sur un processeur à virgule fixe sur 16 bits.
La figure1 montre la configuration utilisée pour tester les algorithmes adaptatifs; c'est le cas de l'annulation d'écho auto-adaptative.



-figure1-

Le filtre adaptatif a la même entrée y(T) que le système discret inconnu. La réponse désirée d(T) est la sortie du système inconnu, et le niveau variable du bruit détermine l'écart quadratique moyen minimal. Nous choisissons la réponse impulsionnelle du système inconnu, telle que:

$$w(T) = (0.96)^T \quad \text{pour } T = 0,1,2, \ldots\ldots\ N-1$$

et le rapport signal/bruit est :

$$10 \log_{10} \frac{E\,[y2(T)]}{E\,[\eta^2(T)]} = 40 \text{ db}$$

L'ordre du filtre adaptatif est N=100, et la valeur du facteur d'oubli est λ=0.997.
Le critère de performance utilisée est l'erreur de sortie a priori pour l'estimation du signal désiré d(T), notée $\varepsilon^p(T/N)$:

$$\varepsilon^p(T/N) = d(T) - \mathbf{w}^t(T\text{-}1/N).\ \mathbf{Y}(T/N)$$

Le vecteur ligne $\mathbf{w}(T\text{-}1/N)$ est le vecteur estimation d'ordre N, calculé à l'instant T-1.

Expérience1. Nous voulons tester les variations de la vitesse de convergence, en fonction de δ, valeur initiale de l'énergie des erreurs de prédiction progressives. Nous réalisons la simulation en virgule flottante sur 32 bits. Nous notons, à partir des courbes de la figure2, que la valeur choisie δ=0.01 donne les meilleurs résultats, bien que le choix de δ=1. donne des résultats encore très convenables.

Expérience 2. Dans les mêmes conditions que l'expérience1, nous testons la dynamique des variables $\gamma^{-1}(T/N)$, et $\tilde{\mathbf{c}}(T/N)$ sur les trois intervalles temporels suivants: [0,120], [121,200], et [201,+∞]. La table1 récapitule les résultats obtenus, pour différentes valeurs de δ. Nous notons que, lorsque δ croît, la dynamique des variables varie moins dans le temps.

Expérience3. Après avoir choisi δ=1., afin de limiter la variation temporelle de la dynamique des variables, nous déterminons le codage associé à chacune des variables. L'utilisation des opérateurs de troncature et d'arrondi nous permet d'effectuer la simulation en virgule fixe. Nous avons testé sur un million d'itérations, la version stabilisée [7] de l'algorithme FTF, implémentée sur un processeur à virgule fixe sur 16 bits; aucune instabilité numérique n'a été détectée. La figure3 montre une légère diminution des performances de cet algorithme implémenté en virgule fixe par rapport à son implémentation en virgule flottante, en ce qui concerne sa vitesse de convergence et l'écart quadratique moyen résiduel.

Conclusion:
Nous avons constaté, que le comportement de la version stabilisée de l'algorithme FTF [7] est numériquement stable, même si l'implémentation est réalisée sur un processeur à virgule fixe 16 bits. Cette étude constitue une première étape pour la réalisation d'annuleurs d'échos auto-adaptatifs à partir de cet algorithme performant, dans le cadre de la conception de modems pour la transmission de données bilatérale simultanée sur les lignes téléphoniques deux fils quelconques.

Remerciements:
Les auteurs tiennent à remercier Digital Equipement-CDG pour les facilités mises à leur disposition pour mener à bien cette étude.

Références:
(1) . A.M. Alvarez : "Echo canceller design and implementation of a CCITT V32 Modem: Software solution" *à paraître dans Proc. ICASSP 1989, Glascow,UK.*
(2) . J.M. Mendel : "White noise estimator for seismic data processing in oil exploration" *IEEE Trans.on Aut. Contr. vol AC-22, pp 694-706, Oct 1977.*
(3) . S. Prasad and S.S. Pathak : "An approch to recursive equalization via white sequence estimation techniques" *IEEE Trans. on Communications, vol. COM 35, n°10, pp 1037-1040, Oct 1987.*
(4) . D. Godard, "Channel equalization using Kalman filter for fast data transmission," *IBM J. of Res. and Dev., pp 267-273,May 1974.*
(5) . L. Ljung, M. Morf, and D. Falconer, " Fast calculation of gain matrices for recursive estimation schemes", *Int. J. Control, vol 27, n°1, pp.1-19, 1978.*
(6) . J.M. Cioffi and T. Kailath : "Fast, recursive least-squares transversal filters for adaptive filtering ".*IEEE.Trans.on ASSP,vol32, n°2, 1984,p304-337.*
(7) . D.T.M. Slock and T. Kailath : "Numerically Stable Fast Recursive Least Squares Transversal Filters", *Proc. of IEEE-ICASSP-88, New York, April 1988, pp. 1365-1368, soumis dans une version plus complète à IEEE ASSP, May 1988.*
(8) . A. Benallal and A. Gilloire : "A new method to stabilize fast RLS algorithms based on a first-order model of the propagation of numerical errors " *Proc. of IEEE-ICASSP-88, New York, April 1988, pp.1373-1376.*
(9) . J. L. Botto and G.V. Moustakides : "Stabilisation of fast recursive least-squares transversal filters for adaptive filtering ". *Proc. of IEEE-ICASSP-87, Dallas, Texas, April 1987, pp.403-407.*
(10) . R. Alcantara : "Implémentation d'algorithmes rapides sur des processeurs de traitement du signal " . *Thèse de Docteur-Ingénieur ENST.1986* .
(11) . "User's manual: TS68930 Architecture and Software" *doc. SGS.Thomson.87.*
(12) . B. Bazareh, J.C. Michalina and A. Picco : "A VLSI Signal Processor with complex arithmetic capability" *IEEE. Trans. on Circuits and Systems, vol CAS-35, pp 495-505, May 1988.*

		[0, 120]	[121, 200]	[201, +∞[
δ=0.01	$\gamma^{-1}(T/N)$	2163	155	3
	$\tilde{\mathbf{c}}(T/N)$	197	18	0.1
δ=1.	$\gamma^{-1}(T/N)$	39.	22.	3.
	$\tilde{\mathbf{c}}(T/N)$	2.	1.7	0.09

-tableau1-

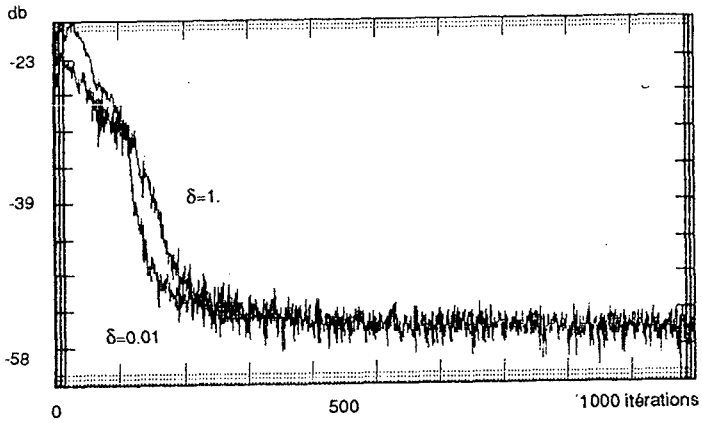


figure2: Erreurs $\varepsilon^p(T/N)$ moyennées sur 10 réalisations pour l'algorithme FTF stabilisé [7]

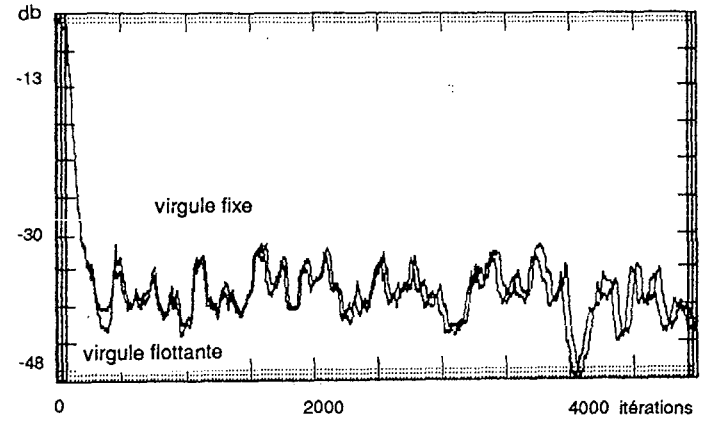


figure3: Puissance de l'erreur normalisée, pour l'algorithme FTF stabilisé [7] implémenté en virgule flottante et en virgule fixe 16 bits