SIMULATION SUR ORDINATEUR DES EFFETS DE QUANTIFICATION DANS LES ALGORITHMES DE FILTRAGE ADAPTATIFS

R. ALCANTARA, J. PRADO et C. GUEGUEN

ENST - SYC
46, Rue Barrault
75634 Paris Cedex 13

J. BOUDY et G. FAVIER

LASSY
41, Bd. Napoléon III
06041 Nice Cedex

RESUME

Lors de l'implantation en arithmétique entière d'un algorithme, les effets de quantification dûs à l'utilisation d'une longueur de mot machine finie peuvent changer considérablement les résultats attendus en précision infinie. L'étude de ces effets numériques peut s'avérer très complexe dans le cas des algorithmes adaptatifs et les résultats obtenus sont souvent trop approximatifs pour une implantation sur microprocesseur ou sur un circuit spécialisé. Dans la pratique on utilise différentes techniques de simulation des effets de quantification de la virgule fixe. Dans cet article, une approche générale du problème de l'implantation en arithmétique entière d'algorithmes, en particulier des algorithmes des moindres carrés récursifs (MCR), est développée. La comparaison des performances des algorithmes MCR transverses et treillis, dans le contexte de la prédiction linéaire adaptative, est effectuée d'une part en termes du nombre de bits nécessaires pour obtenir le meilleur comportement numérique, d'autre part en termes de difficulté d'implantation en virgule fixe.

SUMMARY

When an algorithm is implemented in fixed point arithmetic, the quantization effects using a finite word length can considerably modify the expected infinite precision results. The study of these numerical effects can be difficult when using coefficient update algorithms and the results can generally not be carried over to microprocessor or to other special purpose hardware. Different techniques to simulate the quantization effects of fixed point arithmetic operations have been widely used in practice. In this paper, a general approach to the problem of implementing algorithms on finite word length machines is proposed, in particular the implementation of the recursive least squares (RLS) algorithms is discussed. The numerical performance of the transversal and lattice RLS algorithms, in the context of adaptive linear prediction, is studied in terms of the bit number necessary to obtain a good numerical behavior and in terms of the difficulty of the fixed point implementation.

1. INTRODUCTION

Tout algorithme est développé sous l'hypothèse que l'on dispose d'une arithmétique en précision infinie pour son implantation. Cependant, le calcul sur ordinateur des opérations arithmétiques élémentaires est toujours accompagné d'erreurs de quantification dues à la longueur finie des mots machine utilisés, ceci indépendamment du type de la représentation, en virgule flottante ou en virgule fixe, choisi. Lors de l'implantation sur ordinateur d'un algorithme adaptatif, ces erreurs numériques peuvent changer considérablement les résultats attendus en précision infinie.

L'étude de ces effets de quantification peut permettre soit d'évaluer la dégradation des performances des algorithmes, compte tenu d'une longueur de mot machine fixée, soit de définir une architecture de calcul spécialisée pour un type d'application donnée. Deux approches sont possibles pour effectuer cette étude. D'une part une approche théorique d'analyse de la propagation des erreurs de quantification, troncature ou arrondi, considérant ces erreurs comme des variables aléatoires avec une loi de probabilité connue a priori, voir par exemple [1] - [3]. Dans le cas des algorithmes adaptatifs cette approche peut s'avérer très complexe et les résultats obtenus sont souvent trop approximatifs pour être utilisables lors d'une implantation sur microprocesseur ou sur un circuit VLSI spécialisé. D'autre part une approche empirique consistant à évaluer expérimentalement les erreurs numériques, à partir d'une simulation de la quantification des variables, à l'aide de logiciels informatiques, [4] - [11]. Cette dernière approche permet de rendre systématique les tâches à effectuer pour passer de l'étape du développement d'un algorithme à l'étape de son implantation en vraie grandeur. Nous nous intéressons dans cet article à la deuxième approche du problème.

On considère souvent que la précision des opérations arithmétiques en virgule flottante est infinie. La vitesse de calcul, le prix et la disponibilité des unités de calcul sont des critères qui conduisent souvent à des réalisations en virgule fixe sur des processeurs travaillant sur un nombre réduit de bits. Les simulations des effets de quantification de la virgule fixe peuventt être réalisées à l'aide de la représentation en nombre entier ou en virgule flottante d'un calculateur. Dans le cas d'une représentation en nombre entier, on peut soit définir ses propres opérateurs en langage évolué (+, -, * et /), [9], soit utiliser ceux de la machine sur laquelle est effectuée la simulation, en gérant la position de la virgule [8] et [10]. Lorsque les variables sont codées en virgule flottante, on utilise directement les opérateurs de la machine en gérant la dynamique et la précision [4], [6] et [11]. Le choix d'une technique plutôt qu'une autre est déterminé par la simplicité d'utilisation, par le temps d'exécution nécessaire et par la facilité de validation des résultats. Indépendemment de la technique utilisée, les résultats numériques obtenus devront être identiques pour une longueur de mot nachine fixée. Les différentes techniques de simulation des effets de quantification, et plus particulièrement la simulation à l'aide de la représentation des nombres entiers en virgule flottante, sont discutés brièvement dans la section 2.

La caractéristique de récursivité des algorithmes adaptatifs et la dynamique inconnue de certaines variables intérmediaires (étroitement liée aux conditions initiales et aux caractéristiques des signaux à analyser) rend considérablement difficile le problème de la mise à l'échelle des variables. Par ailleurs, la programmation directe des algorithmes adaptatifs en langage machine est assez ardue, ceci à cause de la structure complexe de ce type d'algorithme. Dans la section 3 nous proposons une approche systématique de l'implantation en arithmétique entière de ce type d'algorithme, [11]. L'aspect

automatique de cette approche simplifie le passage de l'écriture de l'algorithme à son implantation. A l'aide de la méthodologie proposée, dans la section 4 nous présentons les résultats de l'implantation en virgule fixe des algorithmes des moindres carrés récursifs (MCR) transverses et en treillis, dans le contexte de la prédiction linéaire adaptative. La comparaison des performances pour les deux structures est effectuée d'une part en termes du nombre de bits nécessaires pour obtenir le meilleur comportement numérique, d'autre part en termes de la difficulté de l'implantation en virgule fixe de ces algorithmes (mise à l'échelle des variables, programmation en langage machine, etc.). Finalement, dans la section 5 nous présentons les conclusions de notre étude.

2. SIMULATION DES EFFETS DE QUANTIFICATION

L'intérêt principal des simulations en virgule fixe, à l'aide d'un langage évolué, est de pouvoir verifier à l'avance et de manière simple, le comportement d'un algorithme lors de son implantation sur un processeur à longueur de mot machine finie. Une technique pour réaliser ces simulations est de définir les opérations arithmétiques telles qu'elles seront réalisées sur le processeur cible, tout en respectant les caractéristiques de son architecture. Malgré la fidélité des résultats de cette technique, son principal inconvénient est le temps d'exécution de la simulation, très important lorsque l'on travaille en intéractif. Une autre possibilité est de représenter toutes les variables comme des grandeurs entieres afin de pouvoir utiliser les opérations arithmétiques du calculateur hôte. Dans les deux derniers cas, la position de la virgule doit être gérée par l'utilisateur, de la même façon que dans la programmation en langage machine. La simulation des effets de quantification peut aussi être réalisée à l'aide de la représentation en virgule flottante. Dans ce cas l'utilisateur doit uniquement donner la longueur de mot machine et la précision désirée pour chacune des variables. L'intérêt que nous voyons à cette approche est de n'avoir à développer qu'un seul logiciel permettant la simulation en nombre entier ou en précision "infinie". Le principe de la simulation est très simple et indépendant du processeur utilisé pour l'implantation en virgule fixe. Pour illustrer ceci, rappelons la représentation d'un nombre réel, x, sur Q_L bits en complément à 2:

$$\frac{x}{\max(x)} = -b_0 + \sum_{i=1}^{Q_L} b_i 2^{-i} \qquad (1)$$

où $b_i = 0$ où 1 et le pas de quantification, q, est donné par:

$$q = \frac{\max(x)}{2^{Q_L - 1}} \qquad (2)$$

Toute valeur réelle est composée d'une partie entière, d'une partie fractionnaire et d'un signe, alors nous pouvons respectivement leur associer Q_e, Q_f et 1 bits, on aura donc:

$$Q_L = Q_e + Q_f + 1 \qquad (3)$$

et la représentation entière de la variable x quantifiée sera:

$$X_L = \text{Int}\ (x * 2^{Q_{fx}}) \qquad (4)$$

où Int (z) = partie entière de z (réel) et Q_{fx} est le nombre de bits de la partie fractionnaire de x. Si la simulation est réalisée de cette façon, toutes les variables dans l'algorithme devront être représentées comme dans l'équation (4). Prenons maintenant la représentation en flottant sur L bits du nombre réel x:

$$x_L = \pm m\ b^e \qquad (5)$$

où m est la mantisse, b est la base de la représentation de nombres et e est l'exposant. Dans ce cas, il suffit de travailler uniquement sur la mantisse, codée en virgule fixe, pour simuler la troncature et l'arrondi de x:

$$x_{tr} = [\ \text{Int}\ (\ (x_L + n * 2^{-(Q_{fx}+1)})\ ) * 2^{Q_{fx}})\ ]\ * 2^{-Q_{fx}} \qquad (6)$$

x_{tr} est tronqué si n = 0 et arrondi si n = 1. Le modèle donné par l'équation (6) est équivalent à celui utilisé dans [4] et [6], dans lequel si $\max(x) \leq 2^{Q_{ex}}$, alors $q = \frac{1}{2^{Q_{fx}}}$, où Q_{ex} est le nombre de bits de la partie entière de x. Nous avons opté pour la représentation des variables utilisant le nombre de bits de la partie entière et de la partie fractionnaire, plutôt que le pas de quantification q, car elle est plus proche de la façon de procéder de l'utilisateur. Quelques détails supplémentaires de cette approche ainsi que les programmes de la simulation de la troncature et de l'arrondi (en langage FORTRAN) peuvent être consultés dans [11]. Une fois définis les opérateurs de quantification, dans la section 3 nous allons présenter une méthodologie pour l'implantation en virgule fixe.

3. METHODOLOGIE DE L'IMPLANTATION

Les principaux problèmes de l'implantation en virgule fixe des algorithmes adaptatifs sont: l'estimation de la dynamique et la mise à l'échelle des variables, la détermination du nombre de bits nécessaires pour avoir le fonctionnement le plus proche de la réalisation en virgule flottante (précision "infinie") et le choix des conditions initiales optimales. Puisque nous ne disposons pas de relations analytiques pour la solution des problèmes énoncés ci-dessus, on est amené à procéder d'une façon expérimentale pour les résoudre. Après avoir estimé les valeurs maximum des variables (la "dynamique") avec un nombre important de simulations en virgule flottante, nous avons réalisé les mêmes simulations en virgule fixe, de manière à vérifier les premiers résultats et afin d'estimer le nombre de bits nécessaires pour avoir un bon comportement numérique. Ceci pour différentes valeurs des conditions initiales. La dynamique et la précision des variables ayant été estimées à l'aide des simulations en virgule flottante et en virgule fixe, la tâche de l'implantation en langage machine se résume uniquement au calcul des décalages à faire et au calcul de la position du bit à arrondir après chaque opération du type addition/multiplication/stockage. Cette phase de l'implantation reste toujours un travail très ardu, car elle est faite manuellement et il faut la réaliser pour toutes les équations de l'algorithme à implanter. Bien sûr cette tâche peut être rendue automatique à l'aide d'un logiciel qui permet le passage du programme en virgule fixe (avec tous les paramètres précalculés) au programme en langage machine.

Pour illustrer la procédure énoncée précédemment, prenons par exemple le calul de la relation suivante:

$$e(t) = x(t) + A^T(t)\, Y(t) \qquad (7)$$

où e(t) et x(t) sont des scalaires ; A(t) et Y(t) sont des vecteurs de dimension p, T = transposé. La relation (7) en langage FORTRAN (virgule flottante) s'écrit de la façon suivante:

```
EFLX=XFLX
DO I = 1,p
EFLX = EFLX + AFLX(I) * YFLX(I)
END DO
```

où EFLX = e(t), XFLX = x(t), AFLX(I=1,p) = A(t) et YFLX(I=1,p) = Y(t) sont définis comme des variables réelles en simple précision.

Supposons maintenant que l'on veuille implanter

l'équation (7) en virgule fixe sur des longueurs de mot finis. Si toutes les variables ont une dynamique différente, il faudra calculer le pas de quantification de chacune en fonction de leur valeur maximum. Une façon équivalente est de prévoir le nombre de bits nécessaires pour coder la partie entière de chaque variable (et ainsi éviter les saturations) et de prendre les bits restants pour la partie fractionnaire (longueur de mot - (nombre de bits partie entière + 1 bit de signe)). A l'aide d'un opérateur de quantification (d'après la relation (6)), le programme en virgule fixe pour l'équation (7) sera:

```
CALL QUANTR(XFLX,QLX,QTX,NTRX,RX)
EFLX=XFLX
DO I = 1,p
EFLX = EFLX + AFLX(I) * YFLX(I)
END DO
CALL QUANTR(EFLX,QLE,QTE,NTRE,RE)
```

Dans ce cas, EFLX, XFLX, AFLX(I), et YFLX(I) seront déclarés comme des variables réelles en simple ou double précision. QLX et QLE sont les longueurs de mot sur lesquels sont représentés x(t) et e(t) respectivement; QTX et QTE sont respectivement les nombres de bits des parties fractionnaires de x(t) et e(t). NTRX et NTRE sont des paramètres pour le choix de la troncature ou de l'arrondi; RX et RE sont des compteurs du nombre de saturations. Les valeurs numériques des variables seront équivalentes à celles d'une représentation en virgule fixe complément à 2. Avant chaque opération arithmétique les opérandes sont supposés codés sur le nombre de bits fixé préalablement. La mise à l'échelle des variables, obtenue à partir de la simulation en virgule fixe, pourra être directement utilisée pour l'implantation sur processeur de la relation (7). Pour faciliter ceci, nous pourrons définir les opérateurs nécessaires à l'aide des macro intructions du langage machine utilisé, voir [11].

De l'exemple précédent, nous pouvons remarquer que la seule différence entre l'implantation en virgule flottante et la simulation en virgule fixe est le fait d'ajouter le sous-programme de quantification. En outre, à la différence de l'implantation en langage machine, dans le cas de la simulation en virgule fixe on n'a pas besoin de faire le cadrage de la virgule après chaque opération arithmétique. La procédure énoncée précédemment sera utilisée dans la section suivante pour l'implantation en virgule fixe des algorithmes des moindres carrés récursifs.

4. IMPLANTATION DES ALGORITHMES MCR

Nous allons présenter les résultats de l'implantation en virgule fixe des algorithmes des moindres carrés récursifs (MCR) dans le contexte de la prédiction linéaire. La comparaison des performances en arithmétique entière de la version transversale rapide, [12], et de la version en treillis a posteriori, [13], est effectuée en utilisant comme critères le nombre de bits nécessaires pour avoir un bon comportement numérique et la difficulté de l'implantation en virgule fixe.

Les performances des algorithmes MCR sont très liées aux caractéristiques des signaux analysés, c'est pourquoi, dans notre étude nous avons fait des simulations avec différents types de données afin de pouvoir conclure plus largement sur le fonctionnement des algorithmes. Nous limitons notre discussion aux résultats obtenus sur les données engendrées par le modèle auto régressif d'ordre 10 dont les pôles correspondants ont été fixés aux valeurs suivantes, (module;argument): (0,85;0,1 π), (0,89;0,3 π), (0,87;0,5 π), (0,89;0,7 π) et (0,20;0,7 π). La variance du bruit gaussien en entrée est égale à un.

La mise à l'échelle et les conditions initiales des variables ont été déterminées expérimentalement à l'aide des simulations en virgule flottante. Pour ceci, plusieurs valeurs de l'énergie résiduelle initiale (δ) et du facteur d'oubli (λ) ont été essayées. Indépendemment de la structure utilisée, deux remarques importantes peuvent être faites à ce sujet. Premièrement, la dynamique de certaines variables est inconnue, non bornée et variable au cours du temps. Ainsi, pour un ensemble de conditions initiales, le nombre de bits nécessaires pour coder les variables en régime transitoire ne sera pas le même que celui utilisé en régime permanent. A titre d'exemple, dans le tableau 1 on montre les valeurs maximum de l'énergie résiduelle et des intercorrelations dans le cas de la structure treillis, on peut constater que les dynamiques à l'ordre 0 sont très différentes de celles à l'ordre 10. Pour la structure transversale voir la référence [11], où une étude comparative du temps d'exécution, sur processeur de signaux spécialisé, est aussi donné. Deuxièmement, la dynamique des variables et le temps de convergence sont étroitement liées au choix des conditions initiales. Pour avoir un taux de convergence élevé, il est nécessaire d'initialiser l'énergie résiduelle à des valeurs très faibles et de prendre un facteur d'oubli légèrement inférieur à l'unité, de manière à éviter un biais trop élevé sur les paramètres (ceci indépendamment des caractéristiques de stationnarité du signal analysé). Sous les mêmes conditions initiales (δ et λ), la vitesse de convergence des algorithmes transverse et en treillis a été identique, ce fait est montré dans la figure 1. Pour toutes les figures on a af =δ, L=λ, TRA=transverse, TRE=treillis, fx=fixe et fl=flottante. Il est évident que le choix des valeurs initiales des variables est limité par la longueur de mot du processeur utilisé, et donc dans la pratique un compromis entre la complexité de calcul, le taux de convergence, le biais de l'estimation des paramètres et le bon comportement numérique sera nécessaire lors de l'implantation en virgule fixe. D'après nos simulations, les valeurs "optimales" de δ et λ ont été respectivement 0,01 et 0,99 - 0,995.

Vis à vis de la stabilité numérique des algorithmes MCR, nous avons constaté que si la mise à l'echelle des variables et les conditions initiales sont "bien" choisies, leur comportement en virgule fixe et en virgule flottante sera identique. Cependant les instabilités numériques sont plus fréquentes lorsque la longueur de mot machine utilisée est trop réduite, ceci est encore plus vraie pour la structure transversale. Dans nos simulations la longueur limite à été de 12 bits. La figure 2 montre la convergence des algorithmes en virgule flottante et en virgule fixe 16 bits pour $\delta = 0,01$ et $\lambda = 0,995$. Considérant l'ergodicité, le logarithme de l'erreur quadratique moyenne est donnée figure 3. On peut remarquer qu'il est légèrement plus faible dans le cas de la structure treillis. Du point de vue numérique, en régime permanent la version en treillis s'est avérée plus robuste que la version transversale, cependant l'algorithme en treillis utilise un temps de calcul par itération plus élevé.

5. CONCLUSIONS.

Une méthodologie d'implantation en virgule fixe, indépendante de l'algorithme et du processeur utilisés, a été proposée. Une étude comparative des performances numériques des algorithmes MCR transverse et treillis a été aussi présentée. Nos résultats ont mis en évidence les liens entre la mise à l'échelle des variables, le choix des conditions initiales, la convergence et la stabilité numérique des algorithmes. Lorsque les algorithmes transverse et en treillis sont utilisés dans les mêmes conditions, leur comportement numérique en virgule fixe sur 16 bits est similaire. Cependant, le pilotage du facteur d'oubli et de l'énergie résiduelle sera nécessaire afin d'éviter d'éventuelles divergences en régime permanent, surtout lors de l'utilisation de l'algorithme transversal qui est numériquement moins robuste que la version en treillis.

6. REFERENCES

[1] C. Samson and V. U. Reddy, "Fixed point error analysis of the normalized ladder algorithms," IEEE Trans. on ASSP, Vol. 31, No. 5, pp. 1177-1191, October 1983.

[2] S. Ljung, "Fast algorithms for integral equations and least-squares identification problems," Ph.D. dissertation, Linkoping Univ., Sweden, 1983.

[3] C. Caraiscos and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," IEEE Trans. on ASSP, Vol. 32, No. 1, pp. 34-41, February 1984.

[4] P. M. Warren and L. J. Griffiths, "Finite word length effects in the LMS adaptive algorithm," in Proc. ASILOMAR Conf. on Circuits, Syst. and Comp., Pacific Grove, CA, pp. 17-21, November 1982.

[5] F. Ling and J. G. Proakis, "Numerical accuracy and stability: two problems of adaptive estimation algorithms caused by round-off error," in Proc. ICASSP 84, paper 30.3.1, San Diego, 1984.

[6] M. Cresp, "Etude comparative d'algorithmes en treillis," Thèse de Troisième Cycle, Université de Nice, France, November 1984.

[7] G. Favier, "Algorithmes en treillis adaptatifs. Etude comparative de la complexité numérique, des propriétés de convergence et des effets de quantification," Ann. Télécommun., FRA, tome 41, No. 5-6, pp. 305-321, Mai-Juin 1986.

[8] J. F. Agnel, "Elimination de fouillis en radar et détection séquencielle des cibles: une approche par la reconnaisance des formes et le filtrage treillis," Thèse de Docteur Ingénieur, ENST Paris, France, Mars 1985.

[9] K. Kassapoglou and P. Hulliger, "Implementation of recursive least squares identification algorithms on the TMS 320," EUSIPCO-86, The Hague, The Netherlands, September 1986.

[10] R. Alcantara, J. Prado and C. Gueguen, "Fixed point implementation of the fast Kalman algorithm: using a TMS320 microprocessor," EUSIPCO-86, The Hague, The Netherlands, September 1986.

[11] R. Alcantara, "Implantation d'algorithmes rapides sur des processeurs de traitement de signal," Thèse de Docteur Ingénieur, ENST Paris, France, Septembre 1986.

[12] L. Ljung, M. Morf and D. Falconer, "Fast calculation of gain matrices for recursive estimation schemes," Int. J. Control, Vol. 27, No. 1, pp. 1-19, January 1978.

[13] D. T. L. Lee, M. Morf and B. Friedlander, "Recursive least squares ladder estimation algorithms," IEEE Trans. ASSP, Vol. 29, No. 3, June 1981.

Tableau 1 Valeurs maximum des variables dans l'algorithme MCR treillis (t=500)

δ	λ	R_0^εmax	$R_0^\varepsilon(t)$	$R_{10}^\varepsilon(t)$	$\Delta_1(t)$	$\Delta_{10}(t)$
0,00	0,96	5,2	3,0	0,7	1,2	0,10
1,00	0,98	8,9	6,1	1,6	2,8	0,05
0,01	0,99	15,5	12,8	3,5	6,7	-0,02
0,01	1,00	61,0	61,0	18,5	33,4	-0,50

$R_p^\varepsilon(t)$ = énergie résiduelle directe , $R_0^\varepsilon(0)=\delta$
$\Delta_p(t)$ = intercorrelation

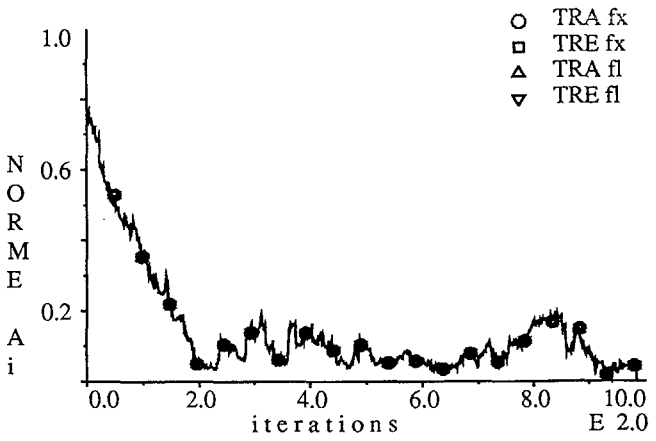

Figure 2 Convergence des algorithmes MCR en virgule flotante et fixe 16 bits

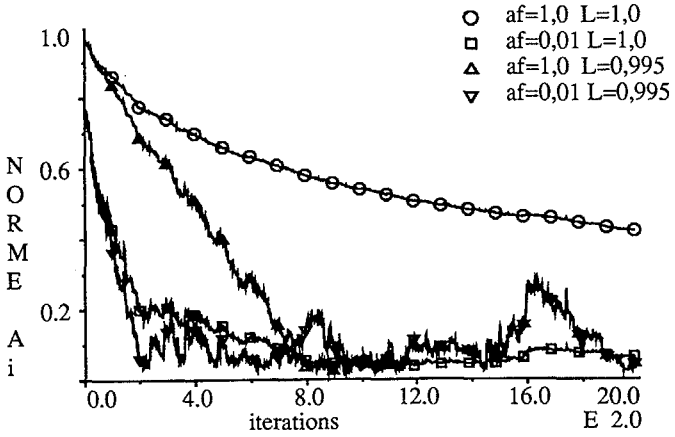

Figure 1 Convergence des algorithmes MCR transverse et en treillis en virgule flottante

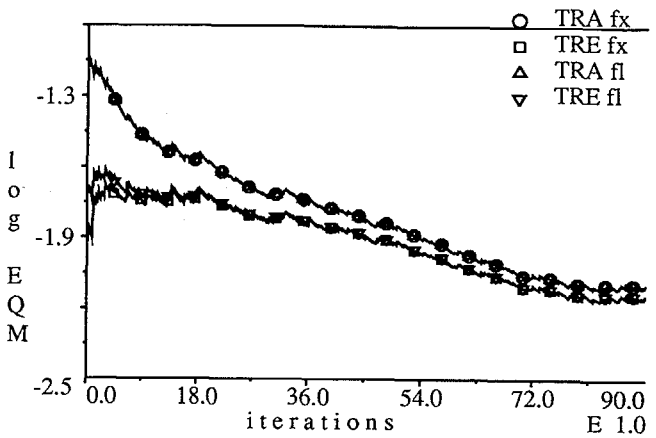

Figure 3 Erreur quadratique moyenne en virgule flottante et fixe 16 bits