ALGORITHMES DE FILTRAGE DE PETITE LONGUEUR ET LEUR UTILISATION EN FILTRAGE NON-RECURSIF

P. DUHAMEL, Z.J. MOU

CNET/PAB/RPE
38-40, rue du Général Leclerc
92131 Issy-les-Moulineaux
FRANCE

RESUME

Ce papier fournit les outils nécessaires à l'utilisation d'une nouvelle classe d'algorithmes rapides récemment proposée. Ces algorithmes permettent une réduction de la charge de calcul tout en gardant partiellement une structure de multiplication-accumulation.

Nous dérivons un ensemble d'algorithmes de petites longueurs, et donnons les règles permettant de les combiner. Leur efficacité est comparée avec celle des algorithmes classiques. Nous montrons ainsi que, quelque soit le critère choisi, il est possible d'améliorer le compromis charge de calcul/régularité de structure.

SUMMARY

This paper provides the basic tools required for an efficient use of the recently proposed fast FIR algorithms. These algorithms allow not only to reduce the arithmetic complexity but also maintain partially the multiply-accumulate structure, thus resulting in efficient implementation.

A set of basic algorithms is derived, together with some rules for combining them. Their efficiency is compared with that of classical schemes in the case of three different criteria, corresponding to various types of implementation. It is shown that this class of algorithms (which includes classical ones as special cases) allows to find the best tradeoff corresponding to any criterion.

1. Introduction

De nombreux algorithmes sont déjà connus pour accélérer l'implantation des filtres à réponse impulsionnelle finie (RIF). Le plus fréquent est basé sur l'utilisation de la Transformée de Fourier Rapide (TFR ou FFT) commme intermédiaire de calcul. Une autre solution classique est l'utilisation de la convolution apériodique. Ces algorithmes présentent les mêmes inconvénients : ils doivent travailler sur de grands blocs de données (de l'ordre de deux fois la taille du filtre) et perdent la structure de multiplication-accumulation (MAC) pour laquelle de nombreuses implantations sont optimisées. Le premier inconvénient impose un volume de matériel important dans les applications dédiées, alors que le deuxième est pénalisant pour les implantations sur processeur de traitement des signaux.

Nous avons proposé récemment [1], indépendamment d'autres auteurs [2,3], une nouvelle classe d'algorithmes rapides permettant de réduire la complexité arithmétique du filtrage RIF, tout en gardant en grande partie la structure MAC. Ces algorithmes permettent la recherche du meilleur compromis complexité arithmétique / régularité de structure pour une implantation donnée.

Après un bref rappel de la dérivation de ces algorithmes, nous montrons que les "briques de base" permettant de les construire sont des filtres de petite longueur à complexité de calcul réduite. Puis, nous donnons les exemples les plus utiles, et présentons la construction des algorithmes de longueurs composites.

Enfin, nous montrons que, quel que soit le critère choisi, la connaissance de l'ensemble des algorithmes de cette classe permet d'améliorer les algorithmes classiques.

2. Description générale

Considérons le filtrage d'une séquence $\{x_i\}$ par un filtre de longueur L dont les coefficients $\{h_i\}$ sont fixes :

(1)

$$y_n = \sum_{i=0}^{L-1} x_{n-i}\, h_i \qquad n = 0, 1, 2, \ldots, \infty$$

Il est bien connu que, par transformée en z, la convolution devient un produit de polynômes.

(2)

$$Y(z) = X(z)\, H(z)$$

où Y et X ont un degré infini, alors que H a un degré L-1.

Considérons maintenant séparément les termes polyphases des trois séquences en cause :

(3)

$$H_j(z) = \sum_{m=0}^{L/N-1} h_{mN+j}\, z^{-m} \quad ; j = 0,1,\ldots,N-1$$

$$X_k(z) = \sum_{m=0}^{\infty} x_{mN+k}\, z^{-m} \quad ; k = 0,1,\ldots,N-1$$

$$Y_i(z) = \sum_{m=0}^{\infty} y_{mN+i}\, z^{-m} \quad ; i = 0,1,\ldots,N-1$$

L'équation (2) devient alors :

(4)

$$\sum_{i=0}^{N-1} Y_i(z^N)\, z^{-i} = \sum_{j=0}^{N-1} H_j(z^N)\, z^{-j} \sum_{k=0}^{N-1} X_k(z^N)\, z^{-k}$$

et cette équation (4) est de la forme d'un produit de polynômes. Les deux polynômes à multiplier sont de degré N-1, et leurs coefficients sont eux-mêmes des polynômes, soit de degré fini, comme $\{H_i\}$, soit de degré infini, comme $\{Y_i\}$ et $\{X_i\}$.

Si l'on oublie un moment que les coefficients du polynôme de degré N sont également des polynômes, on peut appliquer un algorithme rapide au calcul de ce produit. Il est en effet bien connu, depuis le travail de Winograd [4] que le produit de deux polynômes à N coefficients peut s'obtenir avec un minimum de 2N-1 multiplications (rappelons que ce minimum peut facilement être atteint

si N est petit, et que, si ce n'est pas le cas, on préfère utiliser des algorithmes sous-optimaux afin de réduire le nombre d'additions).

L'application de ces algorithmes rapides à l'équation (4) fournit un schéma nécessitant 2N-1 "produits", dont chacun est en fait le produit d'un polynôme de degré fini par un polynôme de degré infini, c'est-à-dire un filtrage de longueur L/N.

On peut préciser encore ce schéma général en remarquant que, écrit sous forme matricielle, cette équation (4) est de la forme du produit d'une matrice "pseudocirculante" et d'un vecteur. Cette équation a manifestement la forme de celle décrivant le fonctionnement d'un filtre RIF de longueur N, dont les coefficients sont $\{H_i\}$ et dont N sorties $\{Y_i\}$ sont calculées. Dans la suite, nous appellerons F(M,N) un algorithme calculant M sorties successives d'un filtre de longueur N.

Remarquons également que, si le filtre de l'équation (4) est calculé par un schéma de type FFT, et avec un choix approprié de N par rapport à L, on retombe sur les schémas classiques de calcul des filtres par FFT, utilisant des techniques de recouvrement ("overlap"). Ceci est expliqué en ref.[5], où nous montrons que tous les algorithmes rapides de filtrage (y compris ceux par FFT) se structurent de la façon suivante :

- décimation des signaux (N sur X et Y, K sur le filtre)
- évaluation des polynômes à N+K-1 "points d'interpolation" $\{\alpha_i\}$
- "produit terme à terme" (ou filtrage)
- reconstruction du polynôme produit et recouvrement.

tous les algorithmes différant par les choix de N, K, et $\{\alpha_i\}$. Le paragraphe suivant est consacré à la description des algorithmes les plus utiles.

3. Algorithmes de filtrage de petite longueur

Considérons tout d'abord le cas le plus simple: un algorithme du type F (2,2) avec $\{\alpha_i\} = \{ 0,1,\infty\}$:

(a1)

$$a_0 = x_0 \qquad b_0 = h_0$$
$$a_1 = x_0 + x_1 \qquad b_1 = h_0 + h_1$$
$$a_2 = x_1 \qquad b_2 = h_1$$
$$m_i = a_i\, b_i ; \qquad i = 0,1,2$$
$$y_0 = m_0 + z^{-2}\, m_2$$
$$y_1 = m_1 - m_0 - m_2$$

Reporté dans l'équation (4), cet algorithme donne le schéma de la Fig.1, où le problème du calcul de 2 sorties successives d'un filtre de longueur N est transformé en celui du calcul d'une sortie de trois filtres de longueur N/2, au prix de 4 additions par bloc de 2 sorties.

Nous avons donc maintenant une charge de calcul de 3/4 L multiplications par sortie (contre L initialement) et 3/4 L+ 1/2 additions par sortie (contre L-1 initialement).

Il est clair que les nombres d'opérations ont été réduits par un facteur 3/4 (excepté pour de très petits filtres). Bien sûr, comme les problèmes de taille N/2 sont encore des filtres RIF, on peut réitérer le processus et obtenir des gains plus conséquents. Ce problème sera traité au paragraphe 4.

Il est bien connu, dans les implantations habituelles des filtres, que la transposition d'un graphe fournit une structure qui possède la même fonction de transfer. Bien que nous soyons ici dans un cas différent (multi - entrée, multi - sortie), c'est également le cas. Il est possible de montrer que les structures, fondées sur l'équation (4) sont des structures à recouvrement-addition (overlap-add), alors que les structures transposées sont à recouvrement-sauvegarde (overlap-save), bien connues dans les methodes par la transformée de Fourier. De plus, on peut démontrer que toutes deux ont exactement la même complexité arithmétique.

L'algorithme suivant est le transposé de l'algorithme (a1) et son utilisation dans un filtre de taille plus importante est donnée en Fig.2.

(a2)

$$a_0 = x_0 - x_1 \qquad b_0 = h_0$$
$$a_1 = x_0 \qquad b_1 = h_0 + h_1$$
$$a_2 = z^{-2}\, x_1 - x_0 \qquad b_2 = h_1$$
$$m_i = a_i b_i ; \qquad i = 0,1,2$$
$$y_0 = m_1 + m_2$$
$$y_1 = m_1 - m_0$$

On peut également dériver des algorithmes à radical plus élevé, qui doivent être plus efficaces, puisque le rapport d'amélioration de la complexité arithmétique $(2N-1)/N^2$ décroît.

Cependant, un algorithme de type F(3,3) nécessite l'utilisation de 5 points d'interpolation différents, c'est-à-dire un de plus que les plus simples: $\{\alpha_i\}=\{0,1,-1,?,\infty\}$ et les choix de complexité immédiatement supérieure $\{\pm 2,\pm 1/2\}$ augmentent le nombre d'additions et le bruit de calcul. Il est donc préférable d'utiliser dans ce cas un algorithme sous-optimal, fournissant un meilleur compromis entre les nombres de multiplications et d'additions.

Un tel algorithme peut s'obtenir par deux applications successives de l'algorithme (a1). Remarquons tout d'abord que (a1) est la traduction de l'équation suivante :

(5)

$$(x_0 + x_1\, z^{-1})\,(h_0 + h_1\, z^{-1})$$
$$= x_0\, h_0 + x_1\, h_1\, z^{-2} + [(x_0 + x_1)\,(h_0 + h_1) - x_0\, h_0 - x_1\, h_1]\, z^{-1}$$

et que la convolution apériodique de longueur 3 peut s'écrire :

(6)

$$[x_0 + (x_1 + x_2\, z^{-1})\, z^{-1}\,]\,[h_0 + (h_1 + h_2\, z^{-1})\, z^{-1}]$$
$$= (x_0 + Fz^{-1})\,(h_0 + Gz^{-1})$$
$$= x_0\, h_0 + [(x_0+F)(h_0+G) - x_0\, h_0 - FG]z^{-1} + FGz^{-2}$$

où l'algorithme (a1) est appliqué au calcul de $(x_0+F)(h_0+G)$ et de FG. Si, de plus, on fait attention à mettre en commun un certain nombre d'additions de "recouvrement", on arrive à l'algorithme suivant, qui nécessite 6 multiplications et 10 additions (Un algorithme optimal nécessiterait 5 multiplications et 22 additions).

(a3)

$$a_0 = x_0 \qquad b_0 = h_0$$
$$a_1 = x_1 \qquad b_1 = h_1$$
$$a_2 = x_2 \qquad b_2 = h_2$$
$$a_3 = x_0 + x_1 \qquad b_3 = h_0 + h_1$$
$$a_4 = x_1 + x_2 \qquad b_4 = h_1 + h_2$$
$$a_5 = x_0 + a_4 \qquad b_5 = h_0 + h_1 + h_2$$
$$m_i = a_i b_i ; \qquad i = 0,1,2,3,4,5$$
$$t_0 = m_0 - m_2\, z^{-3}$$
$$t_1 = m_3 - m_1$$
$$t_2 = m_4 - m_1$$
$$y_0 = t_0 + t_2\, z^{-3}$$
$$y_1 = t_1 - t_0$$
$$y_2 = m_5 - t_1 - t_2$$

L'utilisation de cet algorithme en tant que "radical" pour réduire la

charge de calcul d'un filtre plus grand est montrée en Fig.3. Son utilisation réduit d'environ 1/3 la charge de calcul initiale.

La réf.[5] fournit un certain nombre d'autres algorithmes.

4. Algorithmes à longueurs composites

Dans ce paragraphe, nous allons rechercher les meilleurs manières de combiner ces algorithmes de petites longueurs afin de minimiser différents critères.

Tout d'abord, étudions la décomposition d'un filtre de longueur $L=N_1N_2$ L_2, par $F(N_1,N_1)$, suivi de $F(N_2,N_2)$ (chacun des $F(N_i,N_i)$ a un coût de M_i multiplications et A_i additions).

La complexité globale est alors la suivante :

N_2A_1 adds + M_1 [M_2 filtres de longueur L_2 + A_2 adds]

C'est à dire :

(7)

$$M = M_1 M_2 L_2$$

(8)

$$A = N_2 A_1 + M_1A_2 + M_1 M_2 (L_2 - 1)$$

Nous voyons que, si le nombre de multiplications a une expression symétrique en $F(N_1,N_1)$ et $F(N_2,N_2)$, ce n'est pas le cas du nombre d'additions, qui est donc sensible à l'ordre dans lequel les $F(N_i, N_i)$ sont appliqués.

Il est aisé de voir que la minimisation du nombre d'additions s'effectue en caractérisant les algorithmes par la quantité $(M_i-N_i)/A_i$ et en appliquant les algorithmes par $(M_i-N_i)/A_i$ croissants. Soit Q cette quantité.

On trouve aisément que : Q[F(1,N)]=1; Q[F(2,2)]=0.25; Q[F(3,3)]=0.3; Q[F(5,5)]=0.175 (cf. Tableau I). F(1,N) étant la méthode directe de calcul des filtres, on voit que la meilleure performance est obtenue en le plaçant au centre des algorithmes, ce qui était supposé implicitement jusqu'ici. Cette methode fournit l'ordre des $F(N_i,N_i)$ minimisant le nombre d'additions pour une décomposition donnée.

Cependant, lorsque l'on désire implanter un filtre de longueur L, il est très peu probable que ce soit une décomposition maximale qui soit souhaitable, pour un type d'implantation donnée.

Dans ce qui suit, nous n'allons pas nous attacher à optimiser les décompositions dans un cas précis, mais plutôt chercher à minimiser un certain nombre de critères différents, correspondant chacun à un type d'implantation.

Un premier critère, classique en complexité de calcul est le nombre de multiplications. Le tableau I fournit les algorithmes minimisant ce nombre de multiplications pour différentes longueurs de filtres. On observe que ce minimum est obtenu pour une décomposition maximale de L. Ce tableau fournit également une comparaison avec les méthodes classiques basées sur FFT, tous les signaux étant supposés réels. Pour une comparaison objective, la FFT utilisée est la plus efficace connue [6,7].

On voit que l'approche proposée est plus efficace que les FFT jusquà L = 36, et compétitive jusqu'à L = 64, ce qui recouvre les longueurs les plus utiles.

Néanmoins, dans de nombreuses implantations, ce ne sont plus les temps de multiplications qui sont dominants. En particulier, dans un calculateur d'usage général, le temps d'une addition est comparable à celui d'une multiplication, et un critère plus approprié est celui de la somme du nombre d'opérations (= M + A).

Le tableau II fournit les algorithmes minimisant le nombre total d'opérations nécessaires au calcul d'un filtre de longueur L. La méthode proposée est ici plus efficace que la FFT jusqu'à L = 64.

Le critère précédent était bien adapté pour des implantations sur calculateur général, où toutes les opérations sont effectuées en séquence. Néanmoins, les processeurs de traitement du signal (DSP) posent encore un autre problème. En effet, ces processeurs spécialisés effectuent en général une multiplication - accumulation complète (MAC) en un seul temps de cycle, si bien que la MAC devrait être considérée comme une seule opération, qui n'est pas plus coûteuse qu'une addition. On voit alors qu'un critère approprié est la minimisation du nombre total du MAC et d'additions individuelles.

Le tableau III fournit une description des algorithmes minimisant un tel critère. Bien sûr, ce critère n'est qu'une mesure grossière de l'efficacité d'un algorithme, puisqu'il ne prend en compte ni les temps d'initialisation de boucle, ni les temps d'accès mémoire. Ce tableau II montre cependant que ce type de critère, prenant au moins partiellement en compte la structure des DSP's, peut être amélioré par notre approche. Un filtre de longueur 64 peut s'implanter avec la moitié du nombre total d'opérations (MAC + additions), comparé à l'algorithme habituel, alors que les FFT sont inefficaces pour ce type d'implantation. De plus, cette amélioration est obtenue pour une taille de bloc de seulement 8 points.

5. Conclusion

Nous avons présenté une nouvelle classe d'algorithmes rapides pour l'implantation des filtres non-récursifs, ainsi que l'ensemble des outils nécessaires à leur utilisation.

A titre d'exemple, nous avons montré que, quelque soit le critère choisi (même si l'on prend en compte la structure d'implantation), la méthode proposée permettait d'obtenir des algorithmes plus efficaces que ceux connus à ce jour.

Références

[1]Z.J.Mou, P.Duhamel, "Fast FIR filtering: algorithms and implementations", Signal Processing, Dec. 1987, pp.377-384.

[2]M.Vetterli, "Running FIR and IIR filtering using multirate filter banks", IEEE Trans. ASSP, Vol. 36, N°5, May 1988, pp.730-738.

[3]H.K.Kwan, M.T.Tsim, "High speed 1-D FIR digital filtering architecture using polynomial convolution", in Proc. IEEE ICASSP, Dallas, USA, April, 1987, pp.1863-1866.

[4]S.Winograd, "Arithmetic complexity of computations", CBMS-NSF Reginal Conf. Series in Applied Mathematics, SIAM Publications No. 33, 1980.

[5]Z.J.Mou, P.Duhamel, "Short-length FIR filters and their use in fast non-recursive filtering", soumis à IEEE Trans. on CAS.

[6]P.Duhamel, M.Vetterli, "Improved Fourier and Hartley transform algorithms: application to cyclic convolution of real data", IEEE Trans. ASSP, Vol. ASSP-35, N°6, June 1988, pp.818-824.

[7]H.V. Sorensen, D.L. Jones, M.T. Heideman, C.S. Burrus, "Real-valued fast Fourier transform algorithms", IEEE Trans. ASSP, Vol. ASSP-35, N°6, June 1988, pp.849-863.

Tableau.I Complexité arithmétique d'un algorithme F(N,N) par RIF de petite longueur et par FFT.

	(par RIF de petite longeur)				*(par FFT)*		*(RIF direct)*	
N	M	/N	A	/N	M_{FFT}/N	A_{FFT}/N	M/N	A/N
2	3	1.5	4	2			2	1
3	6	2	10	3.3			3	2
4	9	2.25	20	5	3.75	10.75	4	3
5	12	2.4	40	8			5	4
6	18	3	42	7	5.5	13.83	6	5
8	27	3.38	76	9.5	5.38	16.38	8	7
9	36	4	90	10			9	8
10	36	3.6	128	12.8			10	9
12	54	4.5	150	12.5			12	11
15	72	4.8	240	16	7.47	23.53	15	14
16	81	5.06	260	16.25	7.19	22.19	16	15
18	108	6	306	17			18	17
20	108	5.4	400	20			20	19
24	162	6.75	498	20.75			24	23
25	144	5.76	680	27.2			25	24
27	216	8	630	23.33			27	26
30	216	7.2	744	24.8	7.50	27.63	30	29
32	243	7.59	844	26.38	9.09	28.09	32	31
36	324	9	990	27.5	7.53	30.11	36	35
60	648	10.8	2280	38	7.58	32.58	60	59
64	729	11.39	2660	41.56	10.95	34.05	64	63
128	2187	17.09	8236	64.34	13.02	40.02	128	127
256	6561	25.63	25220	98.52	13.61	46.01	256	255
512	19683	38.44	76684	149.77	17.01	52.01	512	511
1024	59049	57.67	232100	226.66	19.00	58.00	1024	1023

Tableau.II Somme minimum d'opérations (M+A). (mxn) signifie une decomposition par un algorithme rapide F(**m**,**m**), en respectant l'ordre optimal, suivi d'un filtre de longueur n. (FFT) signifie les methodes basées sur FFT.

N	Decomposition	M+A	/N	taille de bloc
2	directe	6	3	1
3	directe	15	5	1
4	(2x2)	26	6.5	2
5	directe	45	9	1
6	(3x2)	56	9.33	3
8	(2x2x2)	94	11.75	4
9	(3x3)	120	13.33	3
10	(5x2)	152	15.2	5
12	(2x3x2)	192	16	6
15	(5x3)	300	20	5
16	(2x2x2x2)	314	19.63	8
18	(2x3x3)	396	22	6
20	(5x2x2)	472	23.6	10
24	(2x2x3x2)	624	26	12
25	(5x5)	740	29.6	5
27	(3x3x3)	810	30	9
30	(5x3x2)	912	30.4	10
32	(2x2x2x2x2)	1006	31.44	16
36	(2x2x3x3)	1260	35	12
60	(5x2x3x2)	2784	46.4	30
64	(FFT)	2880	45.00	64
128	(FFT)	6790	53.05	128
256	(FFT)	15262	59.62	256
512	(FFT)	35334	69.01	512
1024	(FFT)	78854	77.01	1024

Tableau III Nombre minimum de MACs (Longueur de produits scalaires + pré/post additions). (mxn) signifie une decomposition par un algorithme rapide F(**m**,**m**), en respectant l'ordre optimal, suivi d'un filtre de longueur n. (FFT) signifie les methodes basées sur FFT.

N	Decomposition	MAC+adds	/N	taille de bloc
2	directe	4	2	1
3	directe	9	3	1
4	directe	16	4	1
5	directe	25	5	1
6	directe	36	6	1
8	directe	64	8	1
9	directe	81	9	1
10	(2x5)	95	9.5	2
12	(2x6)	132	11	2
15	(3x5)	200	13.33	3
16	(2x8)	224	14	2
18	(2x9)	279	15.5	2
20	(2x2x5)	325	16.25	4
24	(2x2x6)	444	18.5	4
25	(5x5)	500	20	5
27	(3x9)	576	21.33	3
32	(2x2x8)	736	23	4
36	(2x2x9)	909	25.25	4
60	(5x2x6)	2064	34.4	10
64	(2^3x8)	2336	36.5	8
128	(2^4x8)	7264	56.75	16
256	(2^5x8)	22304	87.13	32
512	(2^6x8)	67936	132.69	64
1024	(2^7x8)	205856	201.03	128

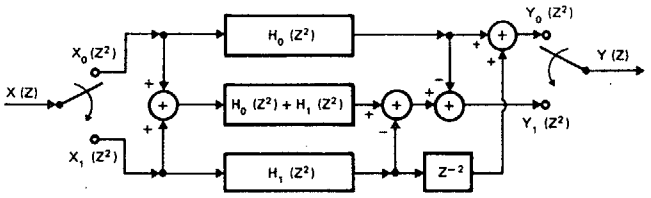

Fig.1 Filtrage par F(2,2)--(a1).

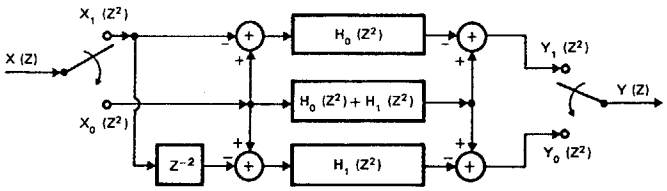

Fig.2 Filtrage par F(2,2)--(a2) = (a1) transposé.

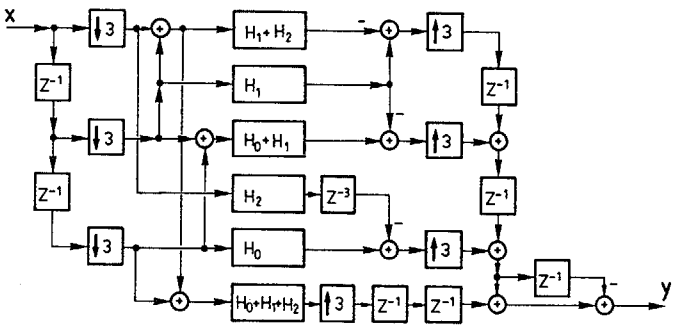

Fig.3 Filtrage par F(3,3)--(a3).