Algorithme de filtrage rapide pour les filtres à phase linéaire

Fast Linear Phase Filtering Algorithm

J. PRADO, ENST/Département SIGNAL, 46, rue Barrault, 75634 Paris Cedex 13, France

Résumé

Nous proposons une méthode simple d'implantation d'algorithme rapide de filtrage non récursif (RIF) à phase linéaire. Cet algorithme permet d'utiliser de manière conjointe la symétrie de la réponse impulsionnelle et l'existence de l'instruction de multiplication, addition et décalage disponible sur la plupart des processeurs numériques de traitement du signal actuels. Une étude comparative sur un calculateur personnel montre un gain de temps de calcul de l'ordre de 20 %.

Mots clés

Filtres RIF, phase linéaire.

Abstract

A simple method for the implementation of a fast algorithm of linear phase FIR digital filtering is proposed here. The algorithm allows to use both the symmetry of the impulse response and the disponibility, on most of the new DSPs, of the one cycle instruction multiplication, addition and delay (z^{-1}). A comparative study on a personnal computer shows that the gain of computation time is about 20 %.

Key words

FIR filters, linear phase.

1. Introduction

De nombreux algorithmes [1] ont été proposés dans la littérature dans le but de réduire le nombre d'opérations mises en jeu dans l'évaluation d'une convolution linéaire. Ces algorithmes sont bien adaptés au cas général, mais ne permettent pas de prendre en compte les symétries liées à la propriété de phase linéaire dans le cas des filtres à réponse impulsionnelle finie.

Nous proposons une méthode simple qui permette la prise en compte de cette symétrie.

2. Méthode

Rappelons tout d'abord l'équation de convolution linéaire dans le cas d'un filtre RIF :

$$y_n = \sum_{k=0}^{N-1} h_k x_{n-k} \cdot \tag{1}$$

L'hypothèse de filtre à phase linéaire se traduit par des relations de symétrie sur la réponse impulsionnelle que l'on peut résumer sous la forme suivante :

$$h_k = \pm h_{N-1-k} \cdot \tag{2}$$

Sans perte de généralité, nous supposerons que N est pair et que le filtre est symétrique, l'extension aux autres cas ne pose pas de problèmes particuliers. La façon la plus simple d'exploiter la symétrie de la réponse impulsionnelle se traduit par l'équation de filtrage suivante :

$$y_n = \sum_{k=0}^{N/2-1} h_k (x_{n-k} + x_{n-N+1+k}) \cdot \tag{3}$$

L'expression (3) montre que la convolution peut s'effectuer à l'aide de N/2 multiplications et (N – 1) additions. Cependant son implantation ne permet pas d'exploiter, de façon simple, la notion de décalage car les séquences x_{n-k} et $x_{n-N+1+k}$ doivent être décalées en sens opposés. Séparons (3) en deux termes faisant apparaître chacun une convolution sur N/2 points :

$$y_n = \sum_{k=0}^{N/2-1} h_k x_{n-k} + \sum_{k=0}^{N/2-1} h_k x_{n-N+1-k} \cdot \tag{4}$$

Nous remarquons que dans (4), chaque terme de la seconde somme a déjà été calculé comme élément de la première lors du calcul d'un échantillon de sortie du passé. Plus précisément le k-ième terme dans la seconde somme de (4) a été calculé comme k-ième terme de la première somme de l'échantillon de sortie y_{n-m}, où l'indice m est donné par :

$$m = N - 1 - 2k . \quad (5)$$

Ainsi, à chaque instant n, le k-ième terme de la première somme peut être mémorisé dans une somme partielle qui sera utilisée dans le calcul de l'échantillon de sortie futur $y_{n+N-1-2k}$. Il est alors possible d'écrire l'équation de convolution linéaire sous la forme :

$$y_n = \sum_{k=0}^{N/2-1} h_k x_{n-k} + py(p) . \quad (6)$$

Dans (6), $py(\ )$ correspond à une somme partielle calculée à l'aide de la remarque (5), et p un indice calculé comme n modulo N. A chaque instant n, la somme partielle qui correspond à la seconde somme apparaissant dans (4) peut être libérée et réutilisée pour l'échantillon futur de sortie y_{n+N}. Ainsi $py(\ )$ peut être stocké dans un tampon de taille N et le calcul d'un échantillon de sortie y_n nécessite seulement N/2 multiplications, N/2 additions plus N/2 additions pour l'actualisation des sommes partielles. La réduction de charge de calcul est compensée par un accroissement de la mémorisation nécessaire au même titre que dans [1].

3. Implantation 1

L'implantation découle directement des équations (5) et (6). Nous en donnons une description en langage symbolique, dans laquelle $xs(\ )$ est une pile de N/2 échantillons d'entrées et $py(\ )$ est un tampon de taille N contenant les sommes partielles du second membre de la convolution donnée par (4).

```
p = 0
loop :
  xs(0) = input (x_n)
  y_n = py(p)
  py(p) = 0
  for k = N/2 - 1 to 1 step - 1
    partial = h(k)* xs(k)        /*multiplication*/
    y_n = y_n + partial          /*accumulation*/
    xs(k) = xs(k - 1)            /*décalage*/
    ind = (p + N - 1 - 2 k) mod N
    py(ind) = py(ind) + partial
  next k
  partial = h(0)* xs(0)
  y_n = y_n + partial
  ind = (p + N - 1) mod N
  py(ind) = py(ind) + partial
  p = (p + 1) mod N
goto loop
```

La boucle la plus interne sur k peut être implantée efficacement sur des processeurs de signaux possédant du parallélisme et la possibilité de répéter un bloc d'instructions.

Dans cette implantation, le calcul modulo N de certains indices peut être pénalisant, exception faite du cas où N est une puissance de 2. En effet si $N = 2^m$, le calcul modulo N est équivalent à un ET logique avec N – 1.

4. Implantation 2

A l'examen du calcul de $py(\ )$, on remarque que lorsque l'indice n est pair on ne calcule que les éléments de $py(\ )$ qui correspondent aux échantillons futurs de sortie d'indice impair, et réciproquement quand l'indice n est impair, on actualise les éléments de $py(\ )$ qui correspondent aux échantillons futurs de sortie d'indice pair. Ainsi, on peut utiliser alternativement deux tampons de longueur N/2, en remplacement du tampon de longueur N. Ces tampons peuvent être contrôlés comme deux FIFO dans lesquelles le sommet de l'une (élément d'incide 0) est la somme partielle utilisée pour évaluer la sortie courante pendant que l'autre est actualisée et décalée du bas vers le haut. De cette façon il est possible d'éviter l'indexation modulo N. L'implantation correspondante est la suivante :

```
loop :
  xs(0) = input (x_n)          /*n pair*/
  p = 0
  y_n = py0(0)
  for k = N/2 - 1 to 1 step - 1
    partial = h(k)* xs(k)        /*multiplication*/
    y_n = y_n + partial          /*accumulation*/
    xs(k) = xs(k - 1)            /*décalage*/
    py1(p) = py1(p + 1) + partial
    p = p + 1
  next k
  py1(p) = h(0)* xs(0)
  y_n = y_n + py1(p)           /*sortie échantillon pair*/
  xs(0) = input (x_n)          /*n est impair*/
  p = 0
  y_n = py1(0)
  for k = N/2 - 1 to 1 step - 1
    partial = h(k)* xs(k)        /*multiplication*/
    y_n = y_n + partial          /*accumulation*/
    xs(k) = xs(k - 1)            /*décalage*/
    py0(p) = py0(p + 1) + partial
    p = p + 1
  next k
  py0(p) = h(0)* xs(0)
  y_n = y_n + py0(p)           /*sortie échantillon impair*/
goto loop
```

Le nombre total d'opérations pour un échantillon de sortie correspond à N/2 multiplications et N – 1 additions comme pour (3) mais la structure permet une meilleure gestion des différents tampons. Une simulation écrite en langage C, sur un calculateur personnel sans parallélisme, des équations (1) et (3), et des deux méthodes proposées, pour différentes longueurs de filtres montre que, si (1) est choisi comme référence de temps de calcul, la méthode 1 donnent un gain de 10 % et la méthode 2 un gain de 20 %. L'évaluation sur un processeur de signal, dont l'architec-

ture est généralement optimisée pour réaliser la convolution directe, va dépendre essentiellement du degré de parallélisme et des possibilités d'adressage de celui-ci. On peut cependant estimer, comme pour tout algorithme dit rapide, obtenir un gain de temps à partir d'une certaine taille N.

5. Conclusion

Nous avons présenté deux méthodes simples d'implantation de filtre RIF à phase linéaire. Ces méthodes réduisent le nombre d'opérations nécessaires à l'évaluation d'une convolution avec comme contrepartie un accroissement de la taille mémoire des données à conserver. L'algorithme est d'une complexité équivalente à celle de [1] et s'en différencie par l'exploitation des symétries de la réponse impulsionnelle. Le gain essentiel, par rapport à l'exploitation directe de la symétrie des coefficients, se situe dans la gestion du décalage temporel.

Manuscrit reçu le 15 février 1993.

BIBLIOGRAPHIE

[1] Z. J. Mou and P. Duhamel, « Fast FIR Filtering : Algorithms and Implementations », *Signal Processing*, Vol. 13, n° 4, December 1987, pp. 377-384.

[2] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*, Addison-Wesley, Reading, MA, 1985.

[3] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983.