DIXIEME COLLOQUE SUR LE TRAITEMENT DU
SIGNAL ET SES APPLICATIONS

NICE du 20 au 24 MAI 1985

UN ALGORITHME DE TRANSFORMEE DE FOURIER RAPIDE A DOUBLE RADICAL. CET ALGORITHME EST-IL OPTIMAL ?

Pierre DUHAMEL

CNET-CRPE, 38-40, rue du Général Leclerc, 92131 ISSY-LES-MOULINEAUX CEDEX (France)

RESUME

Cet article décrit un algorithme de Transformée de Fourier rapide proposé récemment, qui présente des avantages sur le plan de la complexité de calcul, de l'occupation mémoire, et de la régularité de structure. Après avoir brièvement décrit l'algorithme à double radical dans le cas de signaux complexes, l'application aux cas de signaux réels est examinée, ainsi que le lien avec un algorithme optimal vis à vis du nombre de multiplications complexes non triviales.

SUMMARY

A detailed description of a recent algorithm for the fast computation of the DFT is presented. This algorithm has some advantages when considering computational complexity, "in place" computation, and regularity of its implementation. After briefly recalling the split-radix algorithm in the case of complex data, we show how it can be adapted to real data. Finally, we enlight the relationship between split-radix algorithm, and an algorithm with minimum number of non-trivial complex multiplication.

UN ALGORITHME DE TRANSFORMEE DE FOURIER RAPIDE A DOUBLE RADICAL. CET ALGORITHME EST-IL OPTIMAL ?

I - INTRODUCTION

Depuis l'article de COOLEY-TUCKEY [1], de nombreux travaux ont été effectués pour obtenir les algorithmes les plus rapides possibles pour le calcul de la Transformée de Fourier Discrète (TFD).
Les premiers proposés, dits à radical 2, 4, ou même 8, ont été des extensions directes de l'algorithme initial, à l'exception notable de [7] qui, bien que méconnu jusqu'à une période très récente, présentait une complexité de calcul plus faible que tous les algorithmes présentés jusqu'alors. En fait, cet algorithme demandait un nombre de multiplications et d'additions identique à celui des meilleurs algorithmes récemment proposés, mais avec une structure très peu régulière, compliquant à la fois sa programmation et une éventuelle réalisation matérielle.
Puis, les travaux de WINOGRAD sur la complexité des calculs, ont mis en évidence l'intérêt d'algorithmes rapides qui, cette fois ci n'étaient plus des puissances de 2, mais des produits de petites longueurs premières entre elles (choisies parmi 2,3,4,5,7,8,9,16, pour les plus courantes). Deux types d'algorithmes ont résulté de ces travaux : l'algorithme dit de WINOGRAD [5], et l'algorithme dit en facteurs premiers [4].
Parallèlement, les algorithmes rapides portant sur des longueurs $N=2^n$ ont été améliorés par l'introduction d'une série d'algorithmes à coefficients réels [2,3] qui présentaient la même complexité multiplicative que celui de YAVNE [7], c'est-à-dire la plus faible connue à ce jour, avec une structure plus régulière, mais également plus d'additions (cf tables 1 et 2).
Cependant, parmi tous ces algorithmes, les plus utilisés restaient ceux de la première génération, spécialement ceux à radical 2 et à radical 4, à cause de leur structure simple (en "papillon"), et de la possibilité de les programmer "en place", même s'ils apparaissaient plus coûteux en terme de complexité arithmétique que les algorithmes de WINOGRAD et en facteurs premiers.
Le champ de recherches sur les Transformées de Fourier Rapides (TFR) monodimensionnelles semblait presque clos, et l'on n'attendait plus d'amélioration sensible, quand trois nouveaux algorithmes [6,8,9] furent proposés, atteignant la même complexité multiplicative et additive que celui de YAVNE [12], mais avec une structure beaucoup plus simple.
Cet article décrit l'un de ces algorithmes : la TFR à double radical [6] qui semble cumuler les avantages des méthodes classiques et des méthodes récentes :

- un nombre minimum à la fois de multiplications et d'additions, parmi les algorithmes connus,
- la même régularité de programmation (ou de réalisation) que les algorithmes à radical 4,
- la même souplesse d'emploi que les algorithmes à radical 2 (utilisables pour toutes les puissances de 2 : $N=2^n$),
- pas de réarrangement des données à l'intérieur de l'algorithme,
- possibilité de programmation "en place" pour des données réelles ou réelles symétriques, avec une complexité arithmétique réduite (la plus faible parmi les algorithmes connus, ex aequo avec les plus récents),
- bon conditionnement numérique.

Dans cet article, nous présentons tout d'abord le principe de l'algorithme à double radical dans le cas de signaux complexes, et nous comparons sa complexité de calcul à celle d'autres algorithmes.
Puis, nous traitons l'application de cet algorithme à des signaux réels.
Enfin, nous faisons le lien entre cet algorithme et un algorithme basé sur une décomposition de la Transformée de Fourier Discrète en produits de polynômes.

II - L'ALGORITHME A RADICAL DOUBLE

Cet algorithme trouve son origine dans une observation très simple :
Le graphe de fluence d'un algorithme à radical 2 et entrelacement temporel peut se transformer de manière évidente en graphe de fluence d'un algorithme à radical 4 uniquement en changeant les exposants de la racine de l'unité servant de coefficients multiplicateurs (les "Twiddle factors"). Ce faisant, il apparaît assez vite que, à chaque étage de l'algorithme, un radical 4 est plus intéressant pour les termes impairs, et un radical 2 pour les termes pairs de la TFD.
L'algorithme à radical double est donc basé sur la décomposition suivante :

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{nk} \qquad (1)$$

la TFD à calculer (autre notation : $X_k = TFD(k,N, x_n)$).
Un étage de l'algorithme à entrelacement temporel décompose (1) en :

$$\begin{cases} X_{2k} = \sum_{n=0}^{N/2-1} \left(x_n + x_{n+\frac{N}{2}} \right) W_N^{2nk} \\ X_{4k+1} = \sum_{n=0}^{N/4-1} \left[\left(x_n - x_{n+\frac{N}{2}} \right) + j \left(x_{n+\frac{N}{4}} - x_{n+3\frac{N}{4}} \right) \right] W_N^{n} W_N^{4nk} \\ X_{4k+3} = \sum_{n=0}^{N/4-1} \left[\left(x_n - x_{n+\frac{N}{2}} \right) - j \left(x_{n+\frac{N}{4}} - x_{n+3\frac{N}{4}} \right) \right] W_N^{3n} W_N^{4nk} \end{cases} \qquad (2)$$

Le premier étage d'un algorithme à radical double et entrelacement temporel remplace donc le calcul d'une TFD de longueur N par celui d'une TFD de longueur moitié, et deux TFD de longueur N/4, au prix de (N/2-4) multiplications complexes générales (3 multiplications et 3 additions réelles) et de 2 multiplications par $\sqrt[8]{1}$ (2 mult. et 2 add. réelles).
On obtient l'algorithme complet par application successive de cette décomposition, jusqu'au dernier étage, où quelques "papillons" usuels à radical 2 sont nécessaires (le graphe d'une TFR à radical double de longueur 32 est donné en fig. 1).

III - L'ALGORITHME A RADICAL DOUBLE APPLIQUE A DES DONNEES COMPLEXES

L'algorithme à radical double est une combinaison de décompositions à radical 2 et 4 (localement, à chaque étape). Intuitivement, on pourrait se dire que la complexité de calcul résultante doit être intermédiaire entre les complexités des algorithmes à radical 2 et 4. En fait, nous allons voir maintenant que les nombres de multiplications et d'additions intervenant dans cet algorithme sont plus faibles que ceux des deux algorithmes initiaux.

III.1. Complexité de calcul

Soit M_n^c (resp. A_n^c) le nombre de multiplications (resp. additions) réelles non triviales nécessaires au calcul d'une TFD complexe par l'algorithme à double radical. D'après l'équation (2), on a :

$$M_n^c = M_{n-1}^c + 2 M_{n-2}^c + 3.2^{n-1} - 8 \qquad (3)$$

$$M_n^c = 2^n (n-3) + 4 \qquad (4)$$

D'autre part, si l'on oublie pour un instant les additions nécessaires à l'évaluation des produits complexes, le nombre restant d'additions réelles peut facilement être évalué à 2.2^{n+1} puisque, à chacun des 4 étages, un nouveau point est généré par une addition complexe. Ainsi, comme le nombre d'additions réelles nécessaires au calcul des produits complexes est égal au nombre de multiplications réelles :

$$A_n^c = n.\, 2^{n+1} + M_n^c \qquad (5)$$

$$A_n^c = 3.2^n (n-1) + 4 \qquad (6)$$

UN ALGORITHME DE TRANSFORMEE DE FOURIER RAPIDE A DOUBLE RADICAL. CET ALGORITHME EST-IL OPTIMAL ?

II.2. Comparaison avec d'autres algorithmes

Les quantités correspondant aux nombres de multiplications et d'additions intervenant dans un algorithme à double radical sont comparées dans les tables 1 et 2 avec les complexités arithmétiques d'algorithmes classiques (à radical 2,4,8, à facteurs réels) et d'algorithmes plus récents, par WANG [9], VETTERLI-NUSSBAUMER [8], et MARTENS [10].

L'observation de ces tables montre que l'algorithme à double radical demande le nombre le plus faible à la fois de multiplications et d'additions, exaequo avec les algorithmes FFCT [8] et RCFA [10].

En effet si l'on compare l'algorithme à radical double aux algorithmes "classiques" on peut facilement se rendre compte que tous les algorithmes en cause ont exactement le même nombre de coefficients multiplicateurs, si l'on cumule le nombre de multiplications par j, 1, W_N^k. L'avantage essentiel de cet algorithme par rapport aux autres réside dans le fait qu'il maximise le nombre de multiplications par $j = \sqrt{-1}$.

Si l'on compare cet algorithme aux autres algorithmes récents, on se rend compte que l'algorithme de WANG [9] est moins intéressant du point de vue de la complexité de calcul. En ce qui concerne les algorithmes qui demandent le même nombre de multiplications et d'additions, l'algorithme de VETTERLI-NUSSBAUMER [8] présente une structure moins régulière, faisant intervenir des réarrangements des données à chaque étape de calcul. Quant à l'algorithme de MARTENS [10], il s'agit en fait exactement d'un algorithme à racine double et entrelacement temporel. L'approche présentée ici a cependant deux avantages :

. une meilleure compréhension de ce que devrait être la structure élementaire de calcul, permettant ainsi de réduire le nombre d'appels mémoire par rapport à la présentation qui a conduit à RCFA.

. la possibilité d'étudier les deux types d'algorithmes à entrelacement fréquentiel et temporel, qui ne sont plus équivalents au niveau régularité de structure pour des signaux réels ou réels symétriques [13].

III.3. Structure dans le cas de signaux complexes

L'utilisation de la décomposition donnée en (2) conduit naturellement à un algorithme "sur place", obtenu par l'utilisation répétitive de la structure élementaire de calcul données en fig. (2), que l'on appellera "papillon" par extension du terme habituellement utilisé pour les algorithmes à radical 2.

Il est également facile de voir que le nombre minimal d'opérations réelles est obtenu à l'aide de seulement 4 types de "papillon" différents : le cas général (6 mult. réelles), donné en fig. (2), plus deux cas particuliers : k=0 (pas de mult) et k=N/8 (4 mult réelles). Le quatrième type de papillon est celui habituellement utilisé dans l'algorithme à radical 2 (sans mult) pour calculer des TFR de longueur 2.

La programmation de cet algorithme (fig. 1) est fondée sur l'observation du fait que, à chaque étage n°i, les papillons sont appliqués de manière répétitive à des blocs de longueur $N/2^i$.

Le calcul d'une TFR de 1024 points par l'intermédiaire de ce programme prend 92,2 ms, contre 100,3 ms pour le même calcul par le programme compact à radical 4 de MORRIS [14] (moyennes sur 200 calculs sur un Honeywell-Bull DPS8).

Pour d'autres applications que la programmation de TFR sur gros calculateur (par ex. réalisation matérielle, ou nécessité de réduire soit l'occupation mémoire, soit les pré-calculs), il peut être utile de disposer d'un algorithme encore plus régulier. On peut obtenir un tel algorithme en permutant les sorties de la cellule de calcul élémentaire, comme indiqué en fig. 3.

Dans ce cas, le diagramme de la TFR à double radical de longueur 32 de la fig. 1 se transforme en celui de la fig. 4, où les adresses des débuts de bloc de chaque étape apparaissent de manière périodique.

Cependant, cette permutation présente deux inconvénients :

- Les échantillons de sortie de l'algorithme ne sont plus dans un ordre correspondant à l'inversion de la représentation binaire des indices, comme dans les algorithmes habituels, et l'ordre obtenu ne permet plus une remise en ordre naturel dans le même vecteur.

- L'algorithme obtenu demande plus de lectures/écritures en mémoire (2 lectures et 2 écritures supplémentaires par bloc élementaire de calcul).

Effectivement, on peut constater que le programme résultant est plus lent (96.7 ms pour une TFR de longueur 1024, dans les mêmes conditions que précédemment). Néanmoins, la régularité du diagramme résultant peut être intéressante dans le cas d'une réalisation matérielle.

Dans la suite de cet article, comme nous désirons établir des algorithmes de TFR permettant des calculs "en place" pour des signaux réels, nous aurons besoin d'une représentation schématique de la progression de l'algorithme dans les différentes étapes, de la signification des résultats intermédiaires, et de l'endroit où ceux-ci sont stockés. Cette représentation est fournie en fig. (5a) pour l'algorithme à double radical et entrelacement temporel sur données complexes de longueur 16 :

A l'étage 0, les papillons sont appliqués à un bloc de longueur N, bloc qui sert à calculer chacun des échantillons X_k, k=0, --- 15. Après cette première étape, nous avons maintenant un bloc de longueur N/2, appartenant à l'étage 1, qui sert à calculer les échantillons pairs X_{2k}, k=0, ---- 7, et 2 blocs de longueur N/4, appartenant donc à l'étage 2, utilisés pour le calcul de X_{4k+1}, k=0, --- 3 et X_{4k+3}, k=0, --- 3 respectivement. Ce processus est ensuite appliqué successivement à tous les blocs de longueur $N/2^i$ apparaissant à l'étage n°i, i=0, --- n-1.

IV - L'ALGORITHME A RADICAL DOUBLE SUR DES DONNEES A VALEUR REELLE

Il est facile de réduire la complexité de calcul dans un algorithme à radical double.

IV.1. Complexité arithmétique

En effet, quand la séquence x_n est réelle, X_k et X_{N-k} sont complexes conjugués. Reporté dans l'équation (2), ceci signifie qu'il est inutile de calculer à la fois $\{X_{4k+1}\}$ et $\{X_{4k+3}\}$, car :

$$(7) \qquad X_{4k+3} = X_{N-(4k'+1)} = X^*_{4k'+1}$$

On voit donc qu'un algorithme à radical double et entrelacement temporel décompose une TFD réelle de longueur 2^n en une TFD réelle de longueur 2^{n-1} (pour calculer X_{2k}), plus une TFD complexe de longueur 2^{n-2} (pour calculer X_{4k+i}), au prix de $3.2^{n-2}-4$ multiplications réelles, $3.2^{n-2}-4$ additions réelles dues aux coefficients multiplicatifs et 2^n additions.

On obtient donc ainsi la complexité arithmétique d'une TFD réelle à entrelacement temporel :

$$(8) \qquad M_n^r = M_{n-1}^r + M_{n-2}^c + 3(2^{n-2}-2)+2$$

et, avec $M_1^c = 0$, $M_2^r = 0$, il vient :

$$(9) \qquad M_n^r = 2^{n-1}(n-3)+2 = M_n^c / 2$$

de même

$$A_n^r = A_{n-1}^r + A_{n-2}^c + 2^n + 3.2^{n-2} - 4$$

$$(10) \qquad = A_{n-1}^r + 2^{n-2}(3n-2)$$

et, avec $A_1^r = 2$, il vient :

$$(11) \qquad A_n^r = (3n-5)2^{n-1} + 4 = A_n^c/2 - 2^n + 2$$

Une comparaison avec les complexités d'autres algorithmes publiés est fournie dans le tableau (3) où le nombre d'opérations de [11] a été corrigé pour prendre

l'algorithme de multiplication complexe à 3 multiplications et 3 additions réelles. Ici encore, on voit que l'algorithme à radical double a la même complexité que l'algorithme FFCT, soit un nombre d'opérations nettement plus faible que l'algorithme de BERGLAND [11].

IV.2. Structure dans le cas de signaux à valeur réelle
Si nous appliquons une décomposition de type entrelacement temporel à une séquence réelle, comme montré en fig. (5b), le bloc de l'étage 0 est réel conduisant après application de la décomposition à un bloc de longueur N/2 à l'étage 1, qui est également réel. Les deux autres blocs de longueur N/4 appartenant à l'étage 2 sont complexes, mais comme déjà indiqué au paragraphe précédent, il n'est pas nécessaire de calculer X_{4k+3}, de telle sorte que les endroits correspondant du diagramme peuvent être utilisés pour stocker la partie imaginaire du bloc servant à calculer X_{4k+1}. Ce processus est répété jusqu'à ce que la transformée totale soit obtenue.
Remarquons que, puisque nous devons appliquer la décomposition à des blocs de données réelles (le 1er bloc de chaque étage) mais aussi à des blocs de données complexes, le nombre de papillons nécessaires pour une réalisation à nombre minimum d'opérations est maintenant 8.
Remarquons cependant que, parmi les 8 "papillons" différents nécessaires au calcul de la TFR réelle à entrelacement temporel, plusieurs sont rarement utilisés, ce qui permet de trouver facilement des réalisations sous-optimales avec des performances satisfaisantes.
Il est bien sûr également possible d'accroître la régularité de ces algorithmes, comme dans le cas des données complexes, par une permutation des sorties des "papillons".
A ce point de l'exposé, un certain nombre de remarques s'imposent :
Nous disposons, avec l'algorithme à double radical, d'une méthode de calcul de la TFD de signaux complexes, réels, réels symétriques (mais aussi de Transformées en cosinus discrètes, et de TFD impaires, utilisées dans le calcul des TFD à 2 dimensions par transformées polynomiales) demandant dans chaque cas la complexité arithmétique la plus faible connue.
D'autre part, d'autres algorithmes existent, qui demandent exactement le même nombre de multiplications réelles. Ces algorithmes sont fondés sur des principes différents (FFCT, coefficients réels).
Une question se pose alors naturellement :
Puisque ces méthodes arrivent toutes au même nombre (le plus faible connu) de multiplications réelles par des voies différentes, ne seraient-elles pas optimales ? (au sens du nombre minimum de multiplications).
La suite de cet article fournit les premiers éléments de réponse à cette question, à partir de résultats déjà partiellement publiés en [12] :
Après avoir fourni une décomposition de la TFD de longueur 2^n en produits de polynômes, nous évaluons le nombre minimum de multiplications complexes nécessaires au calcul de cet ensemble de produits de polynômes. Nous montrons ensuite le lien existant entre cet algorithme "optimal" et l'algorithme à double radical.

V - DECOMPOSITION D'UNE TFD DE LONGUEUR 2^n EN PRODUITS DE POLYNOMES

Nous ne rappelons ici que les grandes étapes de cette décomposition, en renvoyant le lecteur à [12] pour plus de détails.
Considérons tout d'abord un étage de décomposition à double radical et entrelacement temporel (cf eq. (2)). Cet étage décompose une TFD de longueur 2^n en 3 parties :

- une TFD de longueur 2^{n-1}
- $U_i\,(2^n, hr)$ $\quad i=0, --- 2^{n-2}-1$
- $V_i\,(2^n, h'r)$ $\quad i=0, --- 2^{n-2}-1$

où

(12) $$U_i(2^n,h_r) = \sum_{r=0}^{2^{n-2}-1} h_r\, w_N^{\,r}\, W_N^{\,4ri}$$

(12) $$V_i(2^n,h'_r) = \sum_{r=0}^{2^{n-2}-1} h'_r\, w_N^{\,3r}\, W_N^{\,4ri}$$

et h_r et h'_r sont des combinaisons linéaires des échantillons initiaux, de période 2^{n-2} :

(13) $$h_{r+2^{n-2}} = h_r$$

Remarquons tout d'abord que U_i et V_i sont des fonctions du même type :

(14) $$V_i(2^n,h'_r) = -j\, U_{2^{n-2}-i-1}\,(2^n\,,\, h''_r)$$

avec $h''_r = h'_{-r}$ $\quad r = 1,2, --- 2^{n-2}-1$

$h''_0 = j\, h_0$

Une conséquence directe de l'équation (14) est que, si l'on trouve une décompositon en produits de polynômes de $U_i(2^n,h_r)$, on en aura également une pour la TFD initiale.

Or :

$$U_i(2^n,hr) = \sum_{r=0}^{2^{n-2}-1} h_r\, W_N^{\,r}\, W_N^{\,4ri}$$

(15) $$= \sum_{r=0}^{2^{n-3}-1} h_{2r}\, W_N^{\,2r}\, W_N^{\,8ri} + \sum_{r=0}^{2^{n-4}-1} h_{4r+1}\, W_N^{\,4r+1}\, W_N^{\,4(4r+1)i} + \sum_{r=0}^{2^{n-4}-1} h_{4r+3}\, W_N^{\,4r+3}\, W_N^{\,4(4r+3)i}$$

$$= U_i\,(2^{n-1}, h_{2r}) + S_i\,(W_N\,,\, h_{4r+1}) + T_i\,(W_N\,,\, h_{4r+3})$$

Ici encore, T_i est une fonction du type S_i :

(16) $$Ti\,(W_N\,,\, h_{4r+3}) = j\, S_i\,(W_N^{\,-1}\,,\, h_{-4r-1})$$

et après avoir remarqué que

(17) $$S_{i+2^{n-4}}\,(W_N\,,\, h_{4r+1}) = j\, S_i\,(W_N\,,\, h_{4r+1})$$

et que $$S_i\,(W_N\,,\, h_{4r+1}) = \sum_{r=0}^{2^{n-4}-1} h_{4r+1}\, W_N^{\,(4r+1)\,(4k+1)}$$

peut s'écrire sous forme d'un produit de polynômes modulo $z^{2^{n-4}} \pm j$, on voit que U_i peut se calculer par un ensemble de produits de polynômes, et donc qu'il en est de même pour la TFD initiale (cf. [12]).
Pour résumer, une TFD de longueur 2^n sera obtenue par le calcul de 4 produits de polynôme de longueur 2^{n-4}, plus 8 produits de polynôme de longueur 2^{n-5}, etc...

V.I. Complexité de calcul

Comme $z^{2^n} \pm j$ est un polynôme irréductible dans Q[j] ($\triangleq$ le corps des nombres rationnels étendu par $j = \sqrt{-1}$), un produit de polynômes modulo $z^{2^n} \pm j$ se calcule avec un minimum de $2^{n+1}-1$ multiplications complexes.
En sommant le nombre de multiplications nécessaires au calcul de $U_i(2^n,h_r)$, on obtient sa complexité

(18) $$\mu \left\{U_i(2^n,h_r)\right\} = 2^{n-1} - 2n - 3$$

Un autre calcul simple nous permet ensuite d'obtenir la complexité multiplicative d'une TFD de longueur 2^n par cet algorithme :

UN ALGORITHME DE TRANSFORMEE DE FOURIER RAPIDE A DOUBLE RADICAL. CET ALGORITHME EST-IL OPTIMAL ?

(19) $\mu \left\{ DFT_{2^n} \right\} = 2^{n+1} - 2n^2 + 4n - 8$

Tout semble indiquer que ce nombre de multiplications complexes non triviales est l'optimum (cf. th. 3 de [16]) car les deux conditions essentielles pour que les complexités multiplicatives des produits de polynômes puissent être additionnées sont ici réunies : indépendance linéaire des variables intervenant dans les produits, ainsi que des constantes multiplicatives.
Naturellement, comme les algorithmes optimaux de calcul des produits de polynômes deviennent beaucoup trop coûteux en additions pour des degrés supérieurs à 4, cette décomposition ne paraît utile pratiquement que pour les longueurs correspondantes, c.a.d. N 64.
Cependant, comme nous avons maintenant l'optimum, nous pouvons comparer les complexités des différents algorithmes connus au minimum absolu de multiplications complexes non triviales nécessaires au calcul d'une TFD de longueur 2^n. Ceci est effectué dans le tableau 4. Il nous est donc facile de voir que l'algorithme à radical double est, parmi les algorithmes comparés, celui qui suit l'optimum le plus longtemps, jusqu'à N=64, qui est précisément la longueur au-delà de laquelle on ne saurait plus utiliser les algorithmes optimaux de produits de polynômes sans conséquences sur le nombre d'additions. C'est ce qui motive la conjecture suivant laquelle l'algorithme à radical double serait optimal dans une sous-classe convenablement choisie.

V.2. Liaison avec l'algorithme à radical double

La première étape du raisonnement aboutissant à la description de la TFD de longueur 2^n en produits de polynômes était une décomposition à radical double et entrelacement temporel (cf. eq. (12)). Il est également facile de voir que la deuxième étape de ce raisonnement, explicitée par l'éq. (15) est aussi une décomposition à radical double, mais cette fois-ci, à entrelacement fréquentiel, appliquée à la séquence $h_i W_N^i$.
Comme les décompositions à radical double ont la même complexité l'une que l'autre, les algorithmes à radical double auront la même complexité que l'algorithme de décomposition en produits de polynômes, pourvu que ces produits soient calculés, au lieu de l'algorithme à nombre minimum de multiplication, à l'aide de :

(20) $$S_i(W_N, h_{4r+1}) = W_N^{4i} \sum_{r=0}^{2^{n-4}-1} (h_{4r+1} W_N^{4r+1}) W_N^{16ri}$$

($2^{n-4}-1$ mult ; 2^{n-4} mult ; TFD longueur 2^{n-4})

qui est l'expression exacte des deux décompositions successives à entrelacements temporel et fréquentiel. On voit que l'éq. (20) aboutira à un calcul du produit de polynôme avec un nombre minimum de multiplications $(2^{n-3}-1)$ tant que la TFD ne coûtera pas de multiplications, c'est-à-dire tant que $2^{n-4} \leqslant 4$, ce qui correspond aux remarques déjà faites au paragraphe précédent.

VI - CONCLUSION

Dans cet article nous avons décrit des algorithmes de Transformée de Fourier rapide à radical double appliqués à des données complexes, réelles, ou réelles et symétriques. Ces algorithmes présentent l'avantage de pouvoir être effectués en place, de demander le nombre le plus faible connu d'opérations arithmétiques, et de ne pas nécessiter de réorganisation des données en cours de calcul.
De plus, la comparaison des complexités multiplicatives (nombre de multiplications complexes non triviales) de cet algorithme et d'un algorithme basé sur des décompositions en produits de polynômes permet de conjecturer les domaines d'optimalité suivants (cf. [13] pour Transformées en cosinus et transformées de Fourier impaires) :

TFD impaire :	$N \leqslant 64$
TFD réelle	$N \leqslant 64$
TFD réelle-symétrique	$N \leqslant 64$
TFD symétrie hermitienne	$N \leqslant 64$
Transformée en Cosinus	$N \leqslant 16$
Transformée de Fourier impaire	$N \leqslant 32$
Transformée de Fourier à 2 dimensions par transformée polynomiale	$N \times N \leqslant 64 \times 64$

REFERENCES

[1] J.W. COOLEY, J.W. TUKEY : "An algorithm for machine computation of complex Fourier series". Math. Comput., 1965, vol. 19, pp. 297-301.
[2] C.M. RADER, N.M. BRENNER : "A new principle for fast Fourier Transformation". IEEE trans. ASSP., 1976, vol. 24, pp. 264-265.
[3] R.D. PREUSS : "Very fast computation of the Radix-2, Discrete Fourier Transform". IEEE Trans. ASSP., 1982, vol. 30, pp. 595-607.
[4] C.S. BURRUS, P.W. ESCHENBACHER : "An In-place, In-order Prime Factor FFT Algorithm". IEEE trans. on ASSP., 1981, vol. 29, pp. 806-817.
[5] S. WINOGRAD : "On computing the Discrete Fourier Transform". Math. Comput., vol. 32, pp. 175-199, Jan. 1978.
[6] P. DUHAMEL, H. HOLLMANN : "Split-radix FFT algorithm". Electronics Letters, vol. 20, n°1, pp. 14-16, Jan. 1984.
[7] R. YAVNE : "An economical method for calculating the Discrete Fourier Transform". AFIPS Proc., vol. 33, pp. 115-125, 1968, Fall Joint Computer Conference, Washington.
[8] M. VETTERLI, H.J. NUSSBAUMER : "Simple FFT and DCT algorithms with reduced number of operations". Signal Processing, vol. 6, n° 4, pp. 267-278, juillet 1984.
[9] Z. WANG : "Fast Algorithms for the Discrete W Transforms and the Discrete Fourier Transform". IEEE Trans. on ASSP., vol. ASSP-32, n°4, pp. 803-816, août 1984.
[10] J.B. MARTENS : "Recursive cyclotomic factorization - A new algorithm for calculating the Discrete Fourier Transform". IEEE Trans. on ASSP., vol. ASSP-32, n° 2, pp. 750-761, Avril 1984.
[11] G.D. BERGLAND : "A fast Fourier Transform algorithm for real-valued series". Commun. ACM., vol. 11, pp. 703-710, Oct. 1968.
[12] P. DUHAMEL, H. HOLLMANN : "Existence of a 2^n-FFT algorithm with a number of multiplications lower than 2^{n+1}". Electronics Letters, vol. 20, n° 17, pp. 690-692, août 1984.
[13] P. DUHAMEL : "Implementation of "split-radix" FFT algorithms for complex, real, and real-symmetric data". Soumis pour publication dans IEEE Trans. on ASSP.
[14] L.R. MORRIS : "Digital Signal Processing Software", Ottawa, Canada, DSPSW, Inc, 1982, 1983.
[15] M. VETTERLI : "FFT's of signals with symmetries and application to autocorrelation computation". Soumis pour présentation à MELECON'85. Madrid.
[16] L. AUSLANDER, S. WINOGRAD : "The multiplicative complexity of certain semi-linear systems defined by polynomials". Advances in Applied Mathematics, vol. 1, pp. 257-299, 1980.

N	radix 2	radix 4	radix 8	real factor {2,3,4}	WANG [14]	FFCT [13]	RCFA [15]	split radix
16	24	20		20	20	20	20	20
32	88			68	68	68	68	68
64	264	208	204	196	204	196	196	196
128	712			516	564	516	516	516
256	1800	1392		1284	1468	1284	1284	1284
512	4360		3204	3076	3652	3076	3076	3076
1024	10248	7856		7172	8780	7172	7172	7172
2048				16388	20564	16388	16388	16388

Table I : number of non-trivial real multiplications to compute a length-N complex DFT

UN ALGORITHME DE TRANSFORMEE DE FOURIER RAPIDE A DOUBLE RADICAL. CET ALGORITHME EST-IL OPTIMAL ?

N	Multiplications			Additions		
	Bergland	FFCT	split radix	Bergland	FFCT	split radix
16	12	10	10	60	60	60
32	44	34	34	172	164	164
64	132	98	98	572	420	420
128	356	258	258	1124	1028	1028
256	900	642	642	2692	2436	2436
512	2180	1538	1538	6276	5636	5636
1024	5124	3586	3586	14340	12804	12804
2048	11780	8194	8194	32260	28676	28676

Table 3 : Arithmetic complexity for the computation of DFT's on real data.

$N=2^n$	radix 2 $(n-3)\,2^{n-1}+2$	radix 4 $\frac{(9n-26)\,2^{n-3}+4}{3}$	split-radix $\frac{(3n-8)2^n-(-1)^n}{9}$	this paper $2^{n+1}-2^{n/2}+4n-6$
8	2		2	2
16	10	8	8	8
32	34		26	26
64	98	76	72	72
128	258		186	178
256	642	492	456	408
512	1536		1082	890
1024	3586	2732	2504	1880

Table 4 : number of non-trivial complex multiplications involved in the 2^n FFT algorithm.

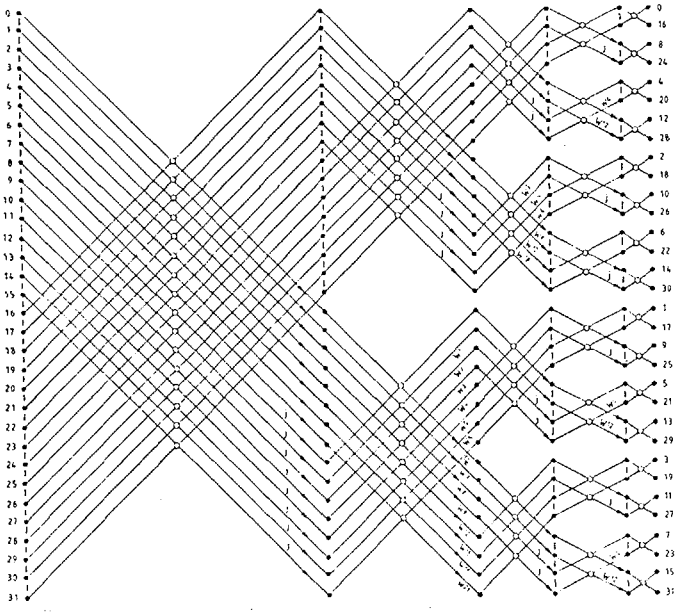

Fig. 1 : length 32 split-radix diagram

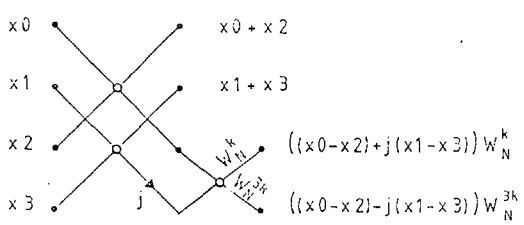

Fig. 2 : the "butterfly" used in the split-radix algorithm

N	radix 2	radix 4	radix 8	Rader Brenner	WANG	FFCT	RCLA	split radix
16	152	148		148	148	148	148	148
32	408			424	388	388	388	388
64	1032	976	972	1104	972	964	964	964
128	2504			2720	2356	2308	2308	2308
256	5896	5488		6464	5566	5380	5380	5380
512	13566		12420	14976	12868	12292	12292	12292
1024	30728	28336		34048	29266	27652	27652	27652
2048				76288	65620	61444	61444	61444

Table : 2 number of real additions to compute a length-N complex DFT.

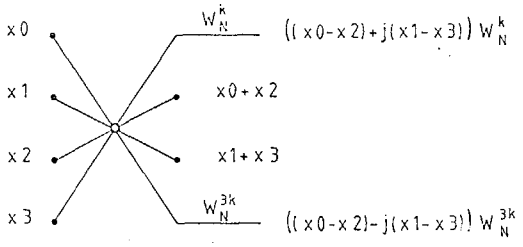

Fig. 3 : butterfly with permuted outputs for a more regular "split-radix" algorithm

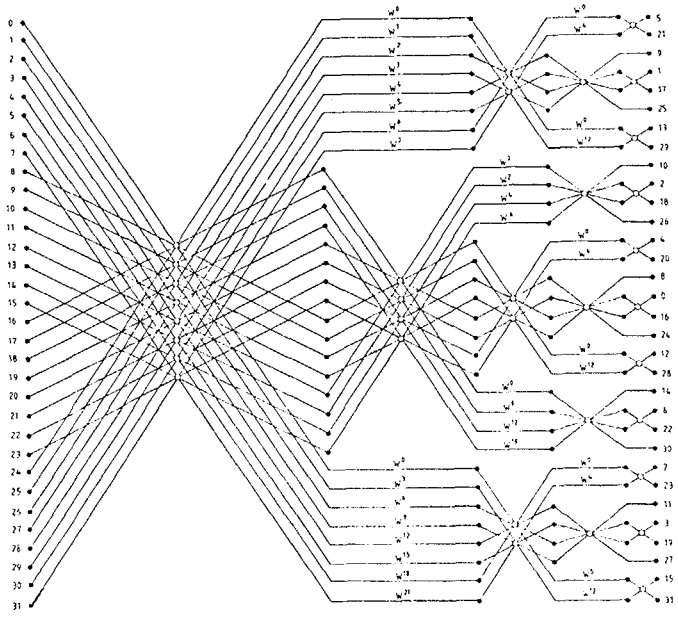

Fig. 4 : symmetric "split-radix" diagram.

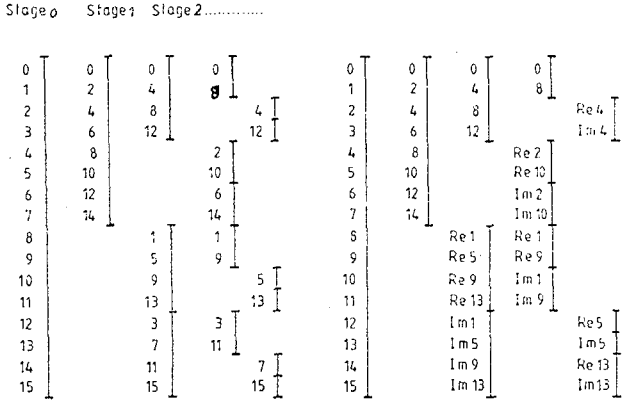

Fig. 5 : schematic representation of a 16-point split radix algorithm a) on complex data b) on real data