FFT의 改善
(Amelioration of FFT)

* 安致得 (Chi Deuk Ahn)
** 安之煥 (Jeeh Wan Ahn)
*** 安秀桔 (Sou Guil Ann)

要 約

DFT를 計算하기 위한 FFT의 計算過程에 包含된 redundancies를 分析하고, 이 redundancies를 最小로 減少시키는 한 方法을 提案하였다. 이를 利用한 FORTRAN Program에 의하여 DFT를 求하기 위한 FFT計算時間이 減縮됨을 보였다. 이러한 redundancy 除去는 다른 Discrete 變換 algorithm에도有用하게 適用될 수 있다.

ABSTRACT

Redundancies in the computational procedure of the FFT by which the DFT is calculated are analized and a modified algorithm to reduce redundancies is proposed.

An amended FFT FORTRAN program adopting the proposed algorithm shows that the time necessary for DFT computation is decreased about one-tenth. The proposed method can be applied to the most of other transformation algorithms

*서울大學校 電子工學科 大學院
**서울大學校 電子工學科 大學院
***正會員 서울大學校 電子工學科敎授

1. 序 論

DFT (Discrete Fourier Transform) 는 digital signal processing algorithm 및 system을 解析하고, 設計하며 실제 運用하는데 있어서 대단히 重要한 役割을 한다. Fourier 解析이 digital signal processing에서 그러한 廣範한 重要性을 갖는 理由中의 하나는 DFT를 計算하기 위한 效率的인 algorithm들이 存在하기 때문이다.

Cooley와 Tukey의 論文[1]이 發表된 以後 DFT를 計算하기 위한 많은 FFT (Fast Fourier Transform) algorithm들이 提案되었으며[2], 이들 algorithm들은 基本的으로 길이가 N인 sequence의 DFT를 連續的으로 더 작은 길이의 DFT로 分解하여 計算效率을 높이고 있다. 이러한 分解를 어떻게 適用하느냐에 따라 여러가지의 서로 다른 algorithm이 나타나게 되는 것이다.

計算對象인 time series가 實數일 때 sampling個數가 N이라면 역시 그 變換結果인 N個 周波數成分이 周波數領域에서 計算되는데 N個의 獨立된 數字만으로 2N個의 結果가 全部 一次 獨立일 수는 없어서 redundancy를 갖고 있다. 이를 除去하므로써 計算時間을 縮少시킬 수 있음이 指摘되었다[3]. 또,

FFT의 各 stage에서의 W_N^0 indexing도 實은 複素數計算을 할 必要가 없음을 追跡하였다.

本 論文에서는 第2章에서 FFT 計算過程에 內在하는 redundancy 를 考察하였으며, 이를 除去하는 方法을 第3章에 보였다. 第4章에서는 이러한 redundancies除去에 依한 decimation-in-time FORTRAN algorithm 을 利用하여 在來의 FFT algorithm에 依한 DFT 計算時間과 比較하고 그 結果를 解析하였으며, 마지막에 그 program을 보였다.

2. FFT 計算過程의 Redundancies

(1) Zero Frequency Redundancy

DFT의 基本計算은 다음 式으로 表示된다.

$$X(K)=\sum_{n=0}^{N-1} x(n)\, W_N^{Kn},\quad k=0.1,\cdots\cdots n-1 \cdots\cdots (1)$$

단, $x(n)$ 은 入力 data, N은 그의 個數이며 X(K)는 그의 DFT이다. 또 W_N 은

$$W_N=exp(-j2\pi/N) \cdots\cdots (2)$$
$$j=\sqrt{-1}$$

로서 discrete motion phasor[3]이다.

式(1)에 依하면 어떤 한 주파수 K에 대하여 transform된 sample $X(\kappa)$를 計算하기 위해서는 N번의 複素數積(complex multiplication)과 N−1번의 複素數加算 (complex addition)을 行해야 한다. 그러나, K=0 인 경우 式(1)은

$$X(o)=\sum_{n=0}^{N-1} x(n)\cdot 1 \cdots\cdots (3)$$

이 되므로, 複素數積을 行하지 않아도 됨을 알 수 있다. 이것은 式(1)을 다음과 같이 分解한 경우에도 마찬가지이다.

$$X(\kappa)=\sum_{n=0}^{N-1}\{(Re[x(n)\cdot Re[W_N^{Kn}] - I_m[x(n)]\cdot I_m[W_N^{Kn}]) + j(Re[x(n)]\cdot I_m[W_N^{Kn}] + I_m[x(n)]\cdot Re[W_N^{Kn}])\},$$

$$\kappa=0,\ 1,\cdots\cdots,\ N-1 \cdots\cdots (4)$$

즉, 式(4)에서 $k=0$인 경우 다음과 같은 實數積(real multiplication)을 行하지 않아도 된다.

$$X(o)=\sum_{n=0}^{N-1}\{(Re[x(n)]\cdot 1) + j(I_m[x(n)]\cdot 1)\} \cdots\cdots (5)$$

따라서, $\kappa=0$인 경우 transform된 sample $X(o)$는 式(3) 또는 式(5)와 같이 複素數加算(實際로 實數加算)만으로 얻어질 수 있다.

(2) 出力 Data Redundancy

出力 Data Redundancy는 특별히 入力 data가 實數列인 경우에 나타나는 redundancy로서, 式(1)로부터 $x(n)$이 實數인 경우 $N/2<K\leq N$ 사이의 line spectrum $X(\kappa)$ 가 redundancy 가 된다[3]. 즉,

$$X(\kappa)=X^*(N-K),\quad N/2<K\leq N \cdots (6)$$

가 된다. 따라서, $N/2<K\leq N$ 사이의 line spectrum을 式(1)에 의해 計算하지 않고, 式(6)에 의하여 $0\leq K\leq N/2$ 에 對한 line spectrum의 虛數項의 부호를 바꾸어 주므로 해서 求할 수 있다.

3. Redundancies 除去에 依한 FFT 計算時間의 減縮 및 그의 解析

入力 data sample數 N이 2의 冪數인 경우, 즉

$$N=2^M,\ M\text{은 整數} \cdots\cdots (7)$$

일때, 第1圖의 decimation-in-time FFT algonthm의 基本計算은 다음 式으로 주어진다[4].

$$X_{l+1}(p)=X_l(p)+W_N^r X_l(q)$$
$$X_{l+1}(q)=X_l(p)-W_N^r X_l(q)$$

$l = 1, 2, \cdots\cdots, M-1, M$ ..... (8)

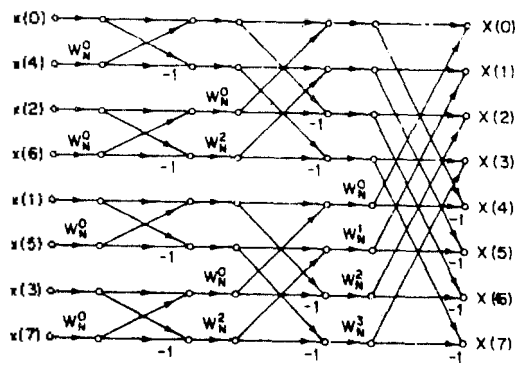
式(8)에 의한 基本計算을 " butterfly computation "이라 하며, 第2圖에 있다. 여기서 $X_l(\cdot)$ 및 $X_{l+1}(\cdot)$은 各各 $(l+1)$ stage에서의 入力과 出力이라 생각할 수 있으며, in-place computation이 可能하다. 第2圖에 의하면 한 butterfly에서 要求되는 複素數積의 個數는 1이고, 複素數加算의 個數는 2이다. 한편, 第1圖에 의하면 各 stage에는 N/2個의 butterfly가 있고, 全體stage 數는 $M = \log_2 N$個이므로, 要求되는 總複素數積의 個數는

$$\frac{N}{2} \cdot \log_2 N = \frac{NM}{2} \quad \cdots\cdots (9)$$

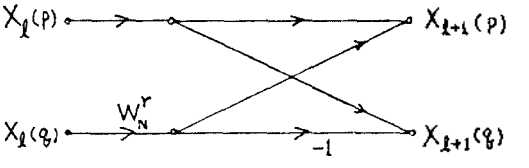
가 되며, 要하는 總複素數加算의 個數는

$$2 \cdot \frac{N}{2} \cdot \log_2 N = NM \quad \cdots\cdots (10)$$

가 됨을 알 수 있다.



第1圖 N=8인 경우의 FFT計算圖
Fig. 3 FFT Signal Flow Graph when N=8



第2圖 버터플라이 演算
Fig2. Butterfly Computation

第1圖에서 式(3)에 依한 X(0)을 求하는 경우 各 stage에서 W_N^0 indexing 수,

$$W_N^0 = W_N^{Kn}\Big|_{K=0} = 1 + j \cdot 0 \quad \cdots\cdots (11)$$

複素數積을 遂行하고 있다. 이것은 바로 2-(1)節의 Zero Frequency Redundancy에 해당하는 것으로서, 이 Zero Frequency Redundancy는 各 stage에서의 複素數 W_N^0 indexing을 行하지 않으므로 써서 除去할 수 있다. 이때 l번째 stage에서 除去되는 W_N^0 複素數積의 個數는 $N/2^l$이고, 全體 stage 數는 M이므로, 減少되는 總複素數積의 個數 $\triangle_m$는

$$\triangle_m = N-1 \doteqdot N \quad \cdots\cdots (12)$$

가 된다. 따라서 減少되는 複素數積의 相對的인 比率 $\triangle_{rm}$은, 式(9) 및 式(12)로 부터,

$$\triangle_{rm} = \frac{\triangle_m}{NM/2} \times 100 = \frac{200}{M}\,[\%] \quad \cdots\cdots (13)$$

가 된다. 第3圖에 Zero Frequency Redundancy를 除去한 경우의 複素數積의 絶對減少數 및 相對減少率을 보였다. 第3圖에 의하면 入力 data sample數 N가 커질수록 減少되는 複素數積의 數도 比例하여 커지나(式(12)) 그 相對減少比率은 漸次로 減少함을 알 수 있다(式(13)).

2-(2)節의 出力 Data Redundancy는 第1圖의 마지막 stage를 별도로 遂行시키므로 해서, 즉 出力을 $0 \le K \le N/2$ 까지만 計算하므로해서 除去시킬 수 있다. 이 경우에 出力 data sample의 printing時間은 半으로 줄어든다. 그러나, 마지막 stage에서 $0 \le K \le N/2$ 까지만을 計算한 다음 complex conjugate시키므로 해서 $N/2+1 \le K \le N-1$

사이의 line spectrum을 求할 수 있으며,이 경우는 半을 완전히 除去한 경우보다 時間이 더 걸리나, 在來의 FFT 보다는 약간 덜 걸린다.

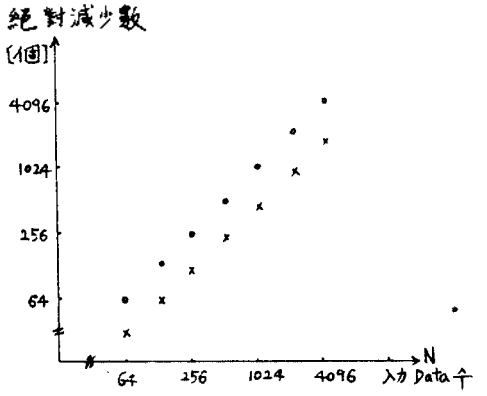
出力 Data Redundancy를 除去하는 경우, 減少되는 複素數積은 없으며, 단지 複素數加算이 約

$$\triangle_a = \frac{N}{2} \quad \cdots\cdots (14)$$

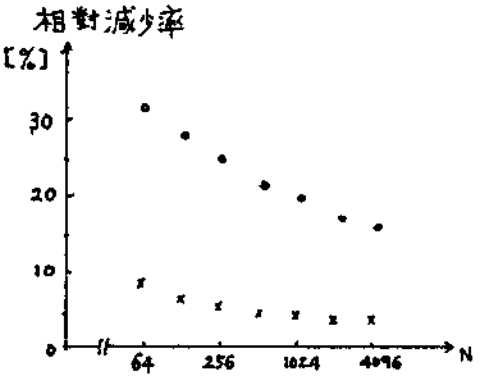
個 줄어든다. 따라서 複素數加算의 相對減少比率은 式(14)와 式(10)에 의해

$$\triangle_{ra} = \frac{50}{M} [\%] \quad \cdots\cdots (15)$$

가 된다. 出力 data redundancy를 除去하는 경우의 複素數加算의 絶對減少數 및 相對減少率이 第3圖에 있다.



(a) O : 複素數積의 減少數 $\triangle_m$ (式 12)
X : 複素數加算 減少數 $\triangle_a$ (式 14)



(b) O : 複素數積의 減少率 $\triangle_{rm}$ (式 13)
× : 複素數加算 減少率 $\triangle_{ra}$ (式 15)

第3圖 Redundancies 除去에 依한 演算의 絶對減少數 및 相對減少率

Fig 3. The number of the eliminated computational operation and the relative reduction rate

4. 結果 및 檢討

Redundancies를 除去한 새로운 algorithm에 依한 FFT 計算時間과 在來의 FFT에 依한 計算時間이 第1表에 比較되어 있다. 여기서 採択한 FFT algorithm은 decimation-in-time FFT algorithm이며, Zero Frequency Redundancy를 除去한 修正된 program을 附錄에 보였다.

第1表에 依하면, 入力 data sample數 N이 커질수록 減少되는 時間量도 따라서 커지지만, 그 相對的인 比는 줄어드는 것을 알 수

入力 sample 數N	Decimation-in-time FFT[4]	Zero Freq. Redundancy를 除去한 FFT
64	0.32 초	0.27초 16%
128	0.65 〃	0.60 〃 9〃
256	1.45 〃	1.32 〃 9〃
1024	6.83 〃	6.35 〃 7〃
4096	31.70 〃	29.85 〃 6〃

入力 sample 數N	出力Data Redundancy를 除去한 FFT	兩Redundancy를 모두 除去한 FFT
64	0.30초 6 %	0.25초 22%
128	0.63 〃 3 〃	0.57 〃 12 〃
256	1.40 〃 3 〃	1.28 〃 12 〃
1024	6.60 〃 3 〃	6.15 〃 10 〃
4096	30.82 〃 3 〃	28.97 〃 9 〃

註) 여기서 %는 相對的인 時間減少率이며, 使用된 計算機는 서울大學校本部에 設置된 IBM360 이다.

第1表. Redundcies除去에 依한 FFT 計算時間의 比較

Table 1. The comparison of computing times for the redundancy eliminated FFT.

있으며, 이 結果는 第3章에서 解析한 結果와 一致한다(第3圖 參照). 한편, 計算所要時間은 그 絶對量이 重要하므로 redundancies를 除去한 FFT algorithm의 效率性은 第1表를 通하여 立證된다.

5. 結 論

FFT計算過程內에 包含된 redundancies를 分析하고, 이 redundancies를 除去하므로 해서 FFT algorithm의 效率性을 더욱 높였다.

Zero Frequency Redundancy가 除去되는 경우, 入力 data 個數에 相當하는 만큼의 複素數積이 減少되며, 특히 入力 data가 實數列인 경우는 出力 Data Redundancy 까지도 除去될 수 있으므로 FFT計算時間이 더욱 減縮된다.

參 考 文 獻

[1] J.W.Cooley and J.W.Tukey, "An Algorithm for the Machine Computation of Complex Fourier Series," Math. Comp.. 19 , Apr.,1965.

[2] W.T.Cochran et al., "What is the Fast Fourier Transform?," IEEE Trans. Audio Electroacoust., vol. Au-15. June 1967.

[3] 安秀桔, "DFT 및 FFT에 있어서의 Redundancies와 그의 除去에 依한 Fourier變換高速化", 電子工學會誌 第14卷6號 1978年1月.

[4] L.R. Rabiner and B. Gold, "Theory and Application of Digital Signal Processing", Prentice-Hall. 1975.

附 錄

Zero Frequency Redundancy를 除去한 FFT FORTRAN Program

```
1. SUBROUTINE FFT(X,N,M)
2. COMPLEX X(N), U,W,T
3. NV2=N/2
4. NM1= N-1
5. J=I
6. DO 7 I=I, NM1
7. IF( I.GE. J ) GO TO 5
8. T= X( J )
9. X(J)= X(I)
10. X(I)= T
11. 5 K=NV2
12. 6 IF(K.GE.J) GO TO 7
13. J= J-K
14. K= K/2
15. GO TO 6
16. 7 J=J+K
17. PI=3. 14159265359
18. DO 30 I=1, N, 2
19. IP= I+ 1
20. T= X (IP)
21. X(IP )= X(I)-T
22. 30 X(I)=X(I)+T
23. DO 20 L=2,M
24. LE= 2**L
25. LE1 =LE/2
26. DO 40 I=I, N, LE
27. IP=I+LE1
28. T= X (IP )
29. X(IP)=X(I)-T
30. 40 X(I)= X(I)+T
31. U= ( 1.0 , 0.0)
32. W= ( MPLX ( Cos ( PI/LE1 ), -SIN
    (PI/LE1))
33. DO 20 J=2. LE1
34. U= U × W
35. DO 20 I= J, N, LE
36. IP= I+LE1
37. T= X ( IP)*U
38. X(IP)=X(I)-T
39. 20X(I)=X(I)+T
40. RETURN
41. END
```