第3卷 第3期 电 子 学 通 讯 Vol. 3, No. 3
1981 年 7 月 JOURNAL OF ELECTRONICS July, 1981

快速傅氏变换的一个改进算法*

姜 遵 富
(中国科学院电子学研究所)

我们知道，对于 $N=2^r$ 点的复数离散序列，快速算法能以 $\frac{N}{2}\cdot r$ 次复数乘法实现其傅氏变换，比 N^2 次复数乘法的实现方法，运算次数大为减少，速度大为提高. 这对数值计算和数字信号处理技术都有重大的意义.

快速傅氏变换 (FFT) 出现虽仅有 15 年[1]，但已有了很大的发展，在实践中已得到广泛的应用. 近年来，在科学技术的许多领域中，日益迫切地要求实现实时地进行数字信号处理，这就促使人们继续不断地就进一步减少乘法运算次数、精练计算机算法程序、探索新的 FFT 的有效算法等问题进行深入的研究.

1975 年 Winograd[2]曾提出一种用最少乘法次数卷积来实现 FFT 的方法，被誉为 FFT 的一次实质性突破. 但，WFTA[3,4] 算法程序过于复杂，普适性较差，最终速度提高仅 20% 左右[5]，至今仍在进行研究中，尚未在工程技术中普遍采用.

在本文中，我们给出 FFT 基 2 算法的一个改进算法，将表明对于 $N=2^r$ 点复数离散序列的傅氏变换，能以 $\frac{N}{2}(r-3)$ 次复数乘法运算次数来实现，与普通 FFT 相比，速度提高率为 $3/(r-3)$. 当 $r=12$, $N=4096$ 时，速度提高率为 33%，比 WFTA 好一些.

我们知道，复数离散傅氏变换定义为

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk} \tag{1}$$

式中，$W_N=e^{-j\frac{2\pi}{N}}$，$N=2^r$，而 n 和 k 以二进制形式表示为：

$$\left.\begin{aligned} n&=2^{r-1}n_{r-1}+2^{r-2}n_{r-2}+\cdots+n_0\\ k&=2^{r-1}k_{r-1}+2^{r-2}k_{r-2}+\cdots+k_0\end{aligned}\right\} \tag{2}$$

由此得：$X(k_{r-1},k_{r-2},\cdots,k_0)=\sum_{n_0=0}^{1}\sum_{n_1=0}^{1}\cdots\sum_{n_{r-1}=0}^{1}x(n_{r-1},n_{r-2},\cdots,n_0)W_N^{2^{r-1}k_0n_{r-1}}$

$$\cdot W_N^{2^{r-2}n_{r-2}(2k_1+k_0)}\cdot W_N^{n_0(2^{r-1}k_{r-1}+2^{r-2}k_{r-2}+\cdots+k_0)} \tag{3}$$

以递归形式表示即为

* 1980 年 4 月 2 日收到.

$$
\left.\begin{aligned}
&x_1(k_0, n_{\gamma-2}, \cdots, n_0) = \sum_{n_{\gamma-1}=0}^{1} x(n_{\gamma-1}, n_{\gamma-2}, \cdots, n_0) W_N^{2^{\gamma-1} k_0 n_{\gamma-1}} \\
&x_2(k_0, k_1, n_{\gamma-3}, \cdots, n_0) = \sum_{n_{\gamma-2}=0}^{1} x_1(k_0, n_{\gamma-2}, \cdots, n_0) W_N^{2^{\gamma-2} n_{\gamma-2}(2k_1+k_0)} \\
&\qquad\vdots \\
&x_i(k_0, \cdots, k_{i-1}, n_{\gamma-i-1}, \cdots, n_0) = \sum_{n_{\gamma-i}=0}^{1} x_{i-1}(k_0, \cdots, k_{i-2}, n_{\gamma-i}, \cdots, n_0) \\
&\qquad \cdot W_N^{2^{\gamma-i} n_{\gamma-i}(2^{i-1}k_{i-1}+\cdots+k_0)} \\
&\qquad\vdots \\
&x_\gamma(k_0, \cdots, k_{\gamma-1}) = \sum_{n_0=0}^{1} x_{\gamma-1}(k_0, \cdots, k_{\gamma-2}, n_0) W_N^{n_0(2^{\gamma-1}k_{\gamma-1}+\cdots+k_0)} \\
&X(k_{\gamma-1}, k_{\gamma-2}, \cdots, k_0) = x_\gamma(k_0, k_1, \cdots, k_{\gamma-1})
\end{aligned}\right\} \tag{4}
$$

因为

$$
\left.\begin{aligned}
W_N^{2^{\gamma-1}} &= W_2 = -1 \\
W_N^{2^{\gamma-2}} &= W_4 = -j \\
W_N^{2^{\gamma-3}} &= W_8 = \frac{\sqrt{2}}{2}(1-j)
\end{aligned}\right\} \tag{5}
$$

故(4)式可写成

$$
\left.\begin{aligned}
&x_1(k_0, n_{\gamma-2}, \cdots, n_0) = \sum_{n_{\gamma-1}=0}^{1} x(n_{\gamma-1}, \cdots, n_0) W_2^{k_0 n_{\gamma-1}} \\
&\qquad = \sum_{n_{\gamma-1}=0}^{1} x(n_{\gamma-1}, \cdots, n_0)(-1)^{k_0 n_{\gamma-1}} \\
&x_2(k_0, k_1, n_{\gamma-3}, \cdots, n_0) = \sum_{n_{\gamma-2}=0}^{1} x_1(k_0, n_{\gamma-2}, \cdots, n_0) W_2^{k_1 n_{\gamma-2}} W_4^{k_0 n_{\gamma-2}} \\
&\qquad = \sum_{n_{\gamma-2}=0}^{1} x_1(k_0, n_{\gamma-2}, \cdots, n_0)(-1)^{k_1 n_{\gamma-2}}(-j)^{k_0 n_{\gamma-2}} \\
&x_3(k_0, k_1, k_2, n_{\gamma-4}, \cdots, n_0) = \sum_{n_{\gamma-3}=0}^{1} x_2(k_0, k_1, n_{\gamma-3}, \cdots, n_0) W_2^{k_2 n_{\gamma-3}} \cdot W_4^{k_1 n_{\gamma-3}} \cdot W_8^{k_0 n_{\gamma-3}} \\
&\qquad = \sum_{n_{\gamma-3}=0}^{1} x_2(k_0, k_1, n_{\gamma-3}, \cdots, n_0)(-1)^{k_2 n_{\gamma-3}}(-j)^{k_1 n_{\gamma-3}} \cdot \left[\frac{\sqrt{2}}{2}(1-j)\right]^{k_0 n_{\gamma-3}} \\
&\qquad\vdots \\
&x_i(k_0, \cdots, k_{i-1}, n_{\gamma-i-1}, \cdots, n_0) = \sum_{n_{\gamma-i}=0}^{1} x_{i-1}(k_0, \cdots, k_{i-2}, n_{\gamma-i}, \cdots, n_0) W_2^{k_{i-1} n_{\gamma-i}} \\
&\qquad \cdot W_4^{k_{i-2} n_{\gamma-i}} \cdot W_8^{k_{i-3} n_{\gamma-i}} W_N^{2^{\gamma-i} n_{\gamma-i}(2^{i-4}k_{i-4}+\cdots+k_0)}
\end{aligned}\right\} \tag{6}
$$

$$
\left.
\begin{aligned}
&= \sum_{n_{\gamma-i}=0}^{1} x_{i-1}(k_0,\cdots,k_{i-2},n_{\gamma-i},\cdots,n_0)(-1)^{k_{i-1}n_{\gamma-i}}(-j)^{k_{i-2}n_{\gamma-i}} \\
&\quad\cdot\left[\frac{\sqrt{2}}{2}(1-j)\right]^{k_{i-3}n_{\gamma-i}} W_N^{2^{\gamma-i}n_{\gamma-i}(2^{i-4}k_{i-4}+\cdots+k_0)} \\
&\quad\vdots \\
x_\gamma(k_0,k_1,\cdots,k_{\gamma-1}) &= \sum_{n_0=0}^{1} x_{\gamma-1}(k_0k_1,\cdots,k_{\gamma-2},n_0)(-1)^{k_{\gamma-1}n_0}(-j)^{k_{\gamma-2}n_0} \\
&\quad\cdot\left[\frac{\sqrt{2}}{2}(1-j)\right]^{k_{\gamma-3}n_0} W_N^{n_0(2^{\gamma-4}k_{\gamma-4}+\cdots+k_0)}
\end{aligned}
\right\}
$$

若按（6）式所表示的递归方程编写计算机程序，x_1、x_2 可不要做乘法运算．在 x_3 中，若令 $k_0=0$ 则得

$$x_3(0,k_1,k_2,n_{\gamma-4},\cdots,n_0)=\sum_{n_{\gamma-3}=0}^{1} x_2(0,k_1,n_{\gamma-3},\cdots,n_0)(-1)^{k_2n_{\gamma-3}}(-j)^{k_1n_{\gamma-3}} \tag{7}$$

共有 $2^{\gamma-1}$ 个方程也不要做乘法运算．若 $k_0=1$，则在 $2^{\gamma-1}$ 个方程中仅需要对 $2^{\gamma-2}$ 个方程做二次实数乘运算即能完成复数列的复乘运算．

在 x_i 中，对于

$$k_0,\ k_1,\cdots,\ k_{i-3}=0$$

有

$$x_i(0,\cdots,0,k_{i-2},k_{i-1},n_{\gamma-i-1},\cdots,n_0)=\sum_{n_{\gamma-i}=0}^{1}\cdot$$

$$x_{i-1}(0,\cdots,0,k_{i-2},n_{\gamma-i},\cdots,n_0)(-1)^{k_{i-1}n_{\gamma-i}}\cdot(-j)^{k_{i-2}n_{\gamma-i}} \tag{8}$$

共有 $2^{\gamma-(i-2)}$ 个方程不要做乘法运算．若

$$k_0,\ k_1,\cdots,\ k_{i-4}=0 \tag{9}$$

而

$$k_{i-3}=1 \tag{10}$$

则

$$x_i(0,\cdots,1,k_{i-2},k_{i-1},n_{\gamma-i-1},\cdots,n_0)=\sum_{n_{\gamma-i}=0}^{1}\cdot$$

$$x_{i-1}(0,\cdots,1,k_{i-2},n_{\gamma-i},\cdots,n_0)(-1)^{k_{i-1}n_{\gamma-i}}\cdot(-j)^{k_{i-2}n_{\gamma-i}}\left[\frac{\sqrt{2}}{2}(1-j)\right]^{n_{\gamma-i}} \tag{11}$$

需要做二次实数乘运算实现复数乘法．(11)式代表 $2^{\gamma-(i-2)}$ 个方程，但仅有一半数量的方程要实际做乘法，故要做 $2^{\gamma-(i-1)}$ 次复数乘法，但只要二次实数乘即能完成．当 k_0 至 k_{i-4} 不全为 0 时，共有 $(2^\gamma-2^{\gamma-i+3})$ 个方程，其中仅有一半要做乘法，故要做的复数乘法为 $2^{\gamma-1}[1-2^{-(i-3)}]$ 次．当 $i=\gamma$ 时，即最后一次运算需做复乘法情形为：

需二次实数乘完成的复数乘 2 次，

需四次实数乘完成的复数乘 $2^{\gamma-1}-4$ 次．

显然，按(6)式实现整个数列 FFT 要进行复数乘运算的次数为

$$M = \sum_{i=3}^{\gamma} m_i = \sum_{i=3}^{\gamma} 2^{\gamma-1}[1 - 2^{-i+2}] = 2^{\gamma-1}(\gamma - 3) + 2 \tag{12}$$

其中有

$$M_1 = \sum_{i=3}^{\gamma} 2^{\gamma-i+1} = 2^{\gamma-1} - 2 \tag{13}$$

次复数乘法用二次实数乘运算即可完成．因此，需四次实数乘的复数运算次数为

$$M_2 = 2^{\gamma-1}(\gamma - 4) + 4 \tag{14}$$

如果以(12)式与普通 FFT 的 $\frac{N}{2}\gamma$ 比较，当 N 较大时，提高变换速度近似等于

$$\eta = \frac{3}{\gamma - 3} \tag{15}$$

对于不同的 γ 值，按(12)式和公式 $2^{\gamma-1}\cdot\gamma$ 分别计算复数乘运算次数，以及本改进算法相对于 FFT 的速度提高率 η 见表 1．

表 1

γ	3	4	5	6	7	8	9	10	11	12	13
$2^{\gamma-1}\gamma$	12	32	90	192	448	1024	2404	5120	11264	24576	53248
$2^{\gamma-1}(\gamma-3)+2$	2	10	34	98	258	642	1538	3586	8194	18434	40962
η	5	2.2	1.57	0.96	0.68	0.59	0.55	0.44	0.375	0.33	0.30

表 1 中 η 值是实际比值，在 $\gamma \geqslant 10$ 时，其值与(15)式相近．由表 1 中的数字可见，本文算法在 γ 较小时速度提高较大，如 $\gamma = 6$ 时，速度提高近 1 倍；$\gamma = 12$，$N = 4096$ 时，速度仍提高 33%，实际上比 WFTA 只提高 20% 左右要好一些．如果在改进的本算法中计及二次实数乘部分，速度提高率 η 当能更大一些．

由于对任意的 γ 数值，(6)式中的 x_1，x_2 皆成立，故可单独编程序．从 x_3 到 x_γ 的计算可按通式 x_i 编写程序．此时首先对 $k_0, k_1, \cdots, k_{i-3} = 0$ 的前 $2^{\gamma-i+2}$ 个方程仅用加法运算程序即可，而后的要用乘法运算的方程另编程序．其中尚可将二次实数乘和四次实数乘的部分分开来编写．可见程序中将增加一些语句，但不会很多．在调用子程序让机器执行过程中，要花费时间，但表中没有计及二次实数乘完成的复数乘运算，节省的时间可与之相抵消．因此，实际达到的速度提高率仍能近于表 1 中所列的数值．本改进算法还能编写出与普通 FFT 同样灵活和通用的程序来．

参　考　文　献

[1] J. Cooley and J. Tukey, Math. Comp t., **19**(1965), 297.
[2] S. Winograd, Proc. Nat, Acad. Sci. U. S. A., **73**(1976), 100.
[3] H. F. Silverman, IEEE Trans. on Assp **ASSP-25**(1977), 242.
[4] V. P. Kolba and T. W. Parks, IEEE Trans. on Assp, **ASSP-25**(1977), 281.
[5] 田中公男、青山友纪，情报处理，**19**(1978)，1091.

AN IMPROVED ALGORITHM FOR FFT

Jiang Zun-fu

(Institute of Electronics, Academia Sinica)

An improved algorithm for FFT is proposed. The Fourier transforms of a complex discrete sequence with $N = 2^\gamma$ points are performed by the operation of $(\gamma-3)$ $N/2$ times of multiplication. The speed of the algorithm is faster than the conventional FFT. The speed up ratio is about $3/(\gamma - 3)$.