国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
一九八六年第一期　总第五十三期　No.1 1986 Sum. 53

快速多项式变换(FPT)及其在计算机上的实现

蒋增荣　赵殿阳

提　要　本文首先推导了两种快速多项式(FPT)算法，所需加法次数均为

$$A_d = MN^2\log_2 N$$

然后讨论了FPT在计算机上的实现，给出了详细框图。在附录中给出了FPT的FORTRAN源程序。

一、引　　言

目前，多项式变换已成功地用来计算一維及多維数字卷积、多維离散富里叶变换（DFT）以及多项式的乘积、大整数的乘积[1]、[2]、[8]。应用多项式变换来计算二維数字卷积及二維DFT，与常用二維FFT法相比，所需的加法次数相同，乘法次数可减少30～40%，特别是FPT法具有高度的并行处理特性，从而节省了计算时间。1983年下半年，我们在CYBER-730计算机上用FPT法计算二維数字卷积和二維DFT（沒有考虑并行性），结果表明，计算时间减少了20～40%，若用并行处理，计算时间可望节省60～70%[4]、[5]，因此，国内外对多项式变换已日感兴趣，正积极将它应用于工业、农业及国防建设，以便取得更大的经济效益。

我们在[6]中已详细地讨论了当模$M(z)$是有理数域的可约或不可约多项式时，多项式变换存在的一系列条件，特别指出了当$N=2^t(t=1,2,\cdots)$，$M=2^m(m=-1,0,1\cdots)$时，如下一对变换是互逆变换，且具有循环卷积特性(CCP)：

$$\bar{A}_k(z) \equiv \sum_{n=0}^{N-1} A_n(z)\tilde{z}^{nk} \bmod (z^{MN}+1) \tag{1.1}$$

$$A_n(z) \equiv \frac{1}{N}\sum_{k=0}^{N-1} \bar{A}_k(z)\tilde{z}^{-nk} \bmod (z^{MN}+1) \tag{1.2}$$

其中$\tilde{z}=z^{2M}$，$\{A_n(z)\}$是已知的长为N的$MN-1$次多项式序列：

$$A_n(z) = \sum_{i=0}^{NM-1} A_n^{(i)} z^i (n=0,1,\cdots,N-1)$$

本文1984年8月26日收到

$$A_n^{(i)}=AR_n^{(i)}+iAI_n^{(i)}(n=0,1,\cdots,N-1,i=0,1,\cdots,NM-1)$$

计算(1.1)及(1.2)完全不要乘法，这是多项式变换获得许多应用的重要原因。直接计算(1.1)或(1.2)，需要$N(N-1)$对次数为$NM-1$的多项式加法，每对这样的多项式加法需NM次加法（复加），所以直接计算(1.1)或(1.2)式共需

$$A_d'=MN^2(N-1) \tag{1.3}$$

次加法，当N，M较大时，加法量是很大的。

本文首先推导(1.1)及(1.2)的快速算法（称为快速多项式变换—FPT），所需的加法次数是

$$A_d=MN^2\log_2 N \tag{1.4}$$

与(1.3)相比，运算量的减少是明显的。然后讨论如何在计算机上实现这种算法，在附录中给出了FPT的FORTRAN源程序。

由于

$$z^{2M}\cdot z^{2M(N-1)}\equiv 1\bmod(z^{MN}+1)$$

所以

$$(z)^{-1}\equiv(z^{2M})^{-1}\equiv z^{2M(N-1)}\bmod(z^{MN}+1) \tag{1.5}$$

所以在(1.2)中以$z^{2M(N-1)}$代替$(z)^{-1}$，便得到与(1.1)相同的结构，因此只需讨论正变换(1.1)的快速算法。

二、快速多项式变换(FPT)的推导

为了方便，这里讨论$m=-1\left(M=\frac{1}{2}\right)$的情况。这时(1.1)成为

$$\bar{A}_k(z)\equiv\sum_{n=0}^{N-1}A_n(z)z^{nk}\bmod(z^{\frac{N}{2}}+1) \tag{2.1}$$
$$(k=0,1,\cdots,N-1)$$

由于$N=2^t(t=1,2,\cdots)$，故设

$$k=2^{t-1}k_{t-1}+2^{t-2}k_{t-2}+\cdots+2k_1+k_0=(k_{t-1},k_{t-2},\cdots,k_1,k_0)$$
$$n=2^{t-1}n_{t-1}+2^{t-2}n_{t-2}+\cdots+2n_1+n_0=(n_{t-1},n_{t-2},\cdots,n_1,n_0)$$

其中$k_i,n_i=0,1(i=0,1,\cdots,t-1)$，注意到$z^N\equiv 1\bmod(z^{\frac{N}{2}}+1)$，有

$$\begin{aligned}z^{nk}&\equiv z^{(2^{t-1}k_{t-1}+\cdots+2k_1+k_0)\cdot(2^{t-1}n_{t-1}+\cdots+2n_1+n_0)}\\&\equiv z^{2^{t-1}n_{t-1}k_0}\cdot z^{2^{t-2}n_{t-2}(2k_1+k_0)}\cdot\cdots\cdot z^{2n_1(2^{t-2}k_{t-2}+\cdots+2k_1+k_0)}\\&\quad\cdot z^{n_0(2^{t-1}k_{t-1}+\cdots+2k_1+k_0)}\bmod(z^{\frac{N}{2}}+1)\end{aligned} \tag{2.2}$$

记

$$A'_{n_0,n_1,\cdots,n_{t-1}}(z)=A_{n_{t-1},\cdots,n_1,n_0}(z) \tag{2.3}$$

(2.1)式就成为

$$\bar{A}_{k_{t-1},\cdots,k_1,k_0}(z)\equiv\sum_{n_0=0}^{1}\sum_{n_1=0}^{1}\cdots\sum_{n_{t-1}=0}^{1}A'_{n_0,n_1,\cdots,n_{t-1}}(z)z^{2^{t-1}n_{n-1}k_0}$$

$$\cdot z^{2^{t-2}n_{t-2}(2k_1+k_0)}\cdot\cdots\cdot z^{2n_1(2^{t-2}k_{t-2}+\cdots+2k_1+k_0)}$$

$$\cdot z^{n_0(2^{t-1}k_{t-1}+\cdots+2k_1+k_0)}\bmod(z^{\frac{N}{2}}+1)$$

注意到$z^{2^{t-1}}=z^{\frac{N}{2}}\equiv-1\bmod(z^{\frac{N}{2}}+1)$，并令

$$(\mathrm{I})\begin{cases}\bar{A}^{(1)}_{n_0,n_1,\cdots,n_{t-2},k_0}(z)\equiv\sum\limits_{n_{t-1}=0}^{1}A'_{n_0,n_1,\cdots,n_{t-1}}(z)(-1)^{n_{t-1}k_0}\bmod(z^{\frac{N}{2}}+1)\\ \bar{A}^{(2)}_{n_0,n_1,\cdots,k_1,k_0}(z)\equiv\sum\limits_{n_{t-2}=0}^{1}\bar{A}^{(1)}_{n_0,n_1,\cdots,n_{t-2},k_0}(z)z^{2^{t-2}n_{t-2}k_0}(-1)^{n_{t-2}k_1}\bmod(z^{\frac{N}{2}}+1)\\ \vdots\\ \bar{A}^{(t-1)}_{n_0,k_{t-2},\cdots,k_1,k_0}(z)\equiv\sum\limits_{n_1=0}^{1}\bar{A}^{(t-2)}_{n_0,n_1,k_{t-3},\cdots,k_0}(z)z^{2n_1(2^{t-3}k_{t-3}+\cdots+2k_1+k_0)}\cdot(-1)^{n_1k_{t-2}}\bmod(z^{\frac{N}{2}}+1)\\ \bar{A}^{(t)}_{k_{t-1},\cdots,k_1,k_0}(z)\equiv\sum\limits_{n_0=0}^{1}\bar{A}^{(t-1)}_{n_0,k_{t-1},\cdots,k_1,k_0}(z)z^{n_0(2^{t-2}k_{t-2}+\cdots+2k_1+k_0)}\cdot(-1)^{n_0k_{t-1}}\bmod(z^{\frac{N}{2}}+1)\end{cases}$$

就有

$$\bar{A}_k(z)\equiv\bar{A}^{(t)}_k(z)\bmod(z^{\frac{N}{2}}+1)\quad(k=0,1,\cdots,N-1)\tag{2.4}$$

方程组(Ⅰ)就是多项式变换(2.1)的迭代算法—FPT，首先根据(2.3)式算出$\{A'_m(z)\}$，它与$\{A_m(z)\}$仅在排列次序上有所不同，然后根据(Ⅰ)依次算出$\bar{A}^{(1)}_k(z)$，$\bar{A}^{(2)}_k(z)$，…，$\bar{A}^{(t)}_k(z)$，由(2.4)式便得到$\bar{A}_k(z)$．每次迭代都要计算N对次数为$\frac{N}{2}-1$的多项式加法，共迭代$t=\log_2N$次，所以本算法所需的加法次数是

$$A_d=N\log_2N\cdot\frac{N}{2}=\frac{1}{2}N^2\log_2N\tag{2.5}$$

由方程组(Ⅰ)可画出FPT流程图。例如$N=8$，这时(Ⅰ)就成为

$$A'_{n_0,n_1,n_2}(z)=A_{n_2,n_1,n_0}(z)$$

$$\bar{A}^{(1)}_{n_0,n_1,k_0}(z)\equiv\sum_{n_2=0}^{1}A'_{n_0,n_1,n_2}(z)(-1)^{n_2k_0}\bmod(z^4+1)$$

$$\bar{A}^{(2)}_{n_0,k_1,k_0}(z)\equiv\sum_{n_1=0}^{1}\bar{A}^{(1)}_{n_0,n_1,k_0}(z)z^{2k_0n_1}(-1)^{n_1k_1}\bmod(z^4+1)$$

$$\bar{A}^{(3)}_{k_2,k_1,k_0}(z)\equiv\sum_{n_0=0}^{1}\bar{A}^{(2)}_{n_0,k_1,k_0}(z)z^{n_0(2k_1+k_0)}(-1)^{n_0k_0}\bmod(z^4+1)$$

$$\bar{A}_{k_2,k_1,k_0}(z)\equiv\bar{A}^{(3)}_{k_2,k_1,k_0}(z)\bmod(z^4+1)$$

流程图如图1所示。

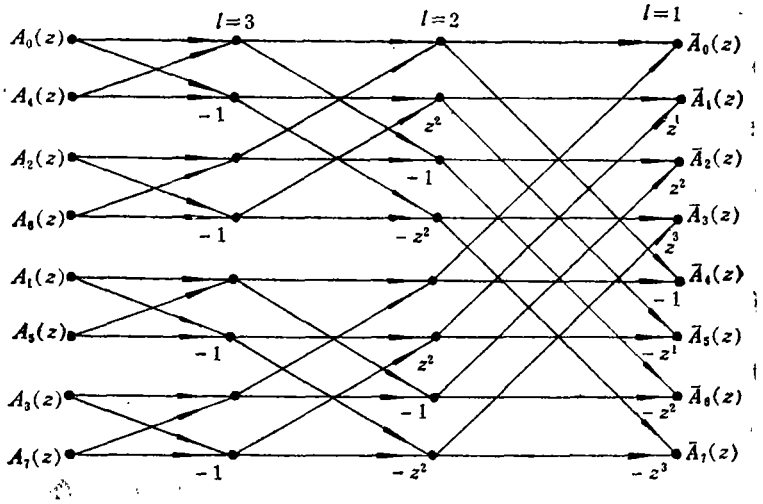

图 1 $N=8$ FPT流程图

若将(2.2)式改写一下，还可得到另一种快速算法。

$$z^{nk}\equiv z^{(2^{t-1}n_{t-1}+\cdots+2n_1+n_0)(2^{t-1}k_{t-1}+\cdots+2k_1+k_0)}$$

$$\equiv z^{k_0(2^{t-1}n_{t-1}+\cdots+2n_1+n_0)}\cdot z^{2k_1(2^{t-2}n_{t-2}+\cdots+2n_1+n_0)}$$

$$\cdot\cdots\cdot z^{2^{t-2}k_{t-2}(2n_1+n_0)}\cdot z^{2^{t-1}k_{t-1}n_0}\bmod(z^{\frac{N}{2}}+1) \tag{2.6}$$

于是

$$\bar{A}_{k_{t-1},\cdots,k_1,k_0}(z)\equiv\sum_{n_0=0}^{1}\sum_{n_1=0}^{1}\cdots\sum_{n_{t-2}=0}^{1}\sum_{n_{t-1}=0}^{1}A_{n_{t-1},n_{t-2},\cdots,n_1,n_0}(z)$$

$$\cdot z^{k_0(2^{t-1}n_{t-1}+\cdots+2n_1+n_0)}\cdot z^{2k_1(2^{t-2}n_{t-2}+\cdots+2k_1+k_0)}$$

$$\cdot\cdots\cdot z^{2^{t-2}k_{t-2}(2n_1+n_0)}\cdot z^{2^{t-1}k_{t-1}n_0}\bmod(z^{\frac{N}{2}}+1)$$

令

$$(\mathrm{I})\quad\begin{cases}A^{(1)}_{k_0,n_{t-2},\cdots,n_1,n_0}(z)\equiv\sum_{n_{t-1}=0}^{1}A_{n_{t-1},n_{t-2},\cdots,n_1,n_0}(z)(-1)^{k_0n_{t-1}}\\ \qquad\cdot z^{k_0(2^{t-2}n_{t-2}+\cdots+2n_1+n_0)}\bmod(z^{\frac{N}{2}}+1)\\ A^{(2)}_{k_0,k_1,\cdots,n_1,n_0}(z)\equiv\sum_{n_{t-2}=0}^{1}A^{(1)}_{k_0,n_{t-2},\cdots,n_1,n_0}(z)(-1)^{k_1n_{t-2}}\\ \qquad\cdot z^{2k_1(2^{t-3}n_{t-3}+\cdots+2n_1+n_0)}\bmod(z^{\frac{N}{2}}+1)\\ \vdots\\ A^{(t-1)}_{k_0,k_1,\cdots,k_{t-2},n_0}(z)\equiv\sum_{n_1=0}^{1}A^{(t-2)}_{k_0,k_1,\cdots,n_1,n_0}(z)(-1)^{k_{t-2}n_1}\cdot z^{2^{t-2}k_{t-2}n_0}\\ \qquad\bmod(z^{\frac{N}{2}}+1)\\ A^{(t)}_{k_0,k_1,\cdots,k_{t-1}}(z)\equiv\sum_{n_0=0}^{1}A^{(t-1)}_{k_0,k_1,\cdots,k_{t-2},n_0}(-1)^{k_{t-1}n_0}\bmod(z^{\frac{N}{2}}+1)\end{cases}$$

就有

$$\bar{A}_{k_{t-1},k_{t-2},\cdots,k_1,k_0}(z)\equiv A^{(t)}_{k_0,k_1,\cdots,k_{t-2},k_{t-1}}(z)\bmod(z^{\frac{N}{2}}+1) \tag{2.7}$$

根据方程组（Ⅰ）依次迭代算出 $A^{(1)}_m(z)$，$A^{(2)}_m(z),\cdots,A^{(t)}_m(z)$，最后由(2.7)式便有

$$\bar{A}_k(z)\equiv A^{(t)}_m(z)\bmod(z^{\frac{N}{2}}+1)$$

其中 k 是m的二进制逆序所对应的整数：

$$k=2^{t-1}m_0+2^{t-2}m_1+\cdots+2m_{t-2}+m_{t-1}$$

$$m=2^{t-1}m_{t-1}+2^{t-2}m_{t-2}+\cdots+2m_1+m_0$$

这种快速算法所需的加法次数也如(2.5)式所示。

当$N=8$时，（Ⅱ）成为

$$A^{(1)}_{k_0,n_1,n_0}(z)\equiv\sum_{n_2=0}^{1}A_{n_2,n_1,n_0}(z)(-1)^{k_0n_2}\cdot z^{k_0(2n_1+n_0)}\bmod(z^4+1)$$

$$A^{(2)}_{k_0,k_1,n_0}(z)\equiv\sum_{n_1=0}^{1}A^{(1)}_{k_0,n_1,n_0}(z)(-1)^{k_1n_1}\cdot z^{2k_1n_0}\bmod(z^4+1)$$

$$A^{(3)}_{k_0,k_1,k_2}(z)\equiv\sum_{n_0=0}^{1}A^{(2)}_{k_0,k_1,n_0}(z)(-1)^{k_2n_0}\bmod(z^4+1)$$

$$\bar{A}_{k_2,k_1,k_0}(z)\equiv A^{(3)}_{k_0,k_1,k_2}(z)\bmod(z^4+1)$$

流程图如图 2 所示。

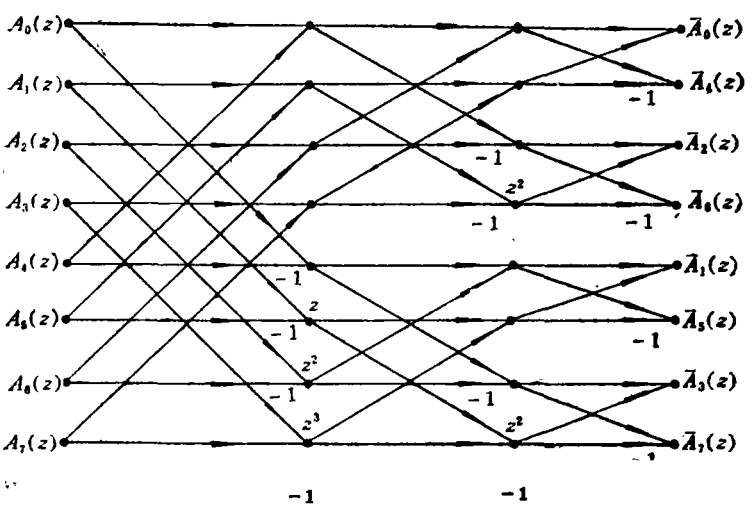

图 2 $N=8$ FPT 的流程图

将(2.2)式变成其它形式，还可得到其它形式的快速算法，读者不难自行推导。

对于模为$z^{NM}+1$、根为z^{2M}、长为N的一般形式的多项式变换(1.1)式，其快速算法可与上述算法一样建立起来，只要在图 1 （图 2）的流程图中，各中间步骤的乘积因子以z^{2M}代换z，就得到(1.1)式的快速算法流程图。这时所需要的加法次数为

$$A_d=MN^2\log_2N \tag{2.8}$$

三、快速多项式变换在计算机上的实现

以图1表示的快速算法为例来说明FPT在计算机上的实现。由于FPT和FFT形式上极为相似，在下面的叙述中，采取FFT的有关术语。

在图1中，以 l 表示各纵列的编号，从左到右依次编为 $l=t,\cdots,2,1(N=2^t)$，每一纵列对偶结点的距离为 $N_2=N/2^l$，每对对偶结点均可用下述一对公式表示：

$$A_{k_1}(z)\equiv B_{k_1}(z)+z^{2^{l-1}k}B_{k_2}(z)\bmod(z^{\frac{N}{2}}+1) \tag{3.1}$$

$$A_{k_2}(z)\equiv B_{k_1}(z)-z^{2^{l-1}k}B_{k_2}(z)\bmod(z^{\frac{N}{2}}+1) \tag{3.2}$$

其中 $N=2^t$，$l=t,\cdots,2,1$，$N_2=N/2^l$，$k_1=k+2iN_2$，$k_2=k_1+N_2$，$k=0,1,\cdots,N_2-1$，$i=0,1,\cdots,2^{l-1}-1$. 对于编号为 $l(t>l\geqslant 1)$ 的纵列，(3.1)及(3.2)式右边 $B_{k_1}(z)$ 和 $B_{k_2}(z)$ 是上一纵列结点 k_1 和 k_2 处已经计算出的值，左边 $A_{k_1}(z)$ 和 $A_{k_2}(z)$ 是当前纵列对偶结点 k_1 和 k_2 处的值，当 $l=t$ 时，$B_{k_1}(z)$ 及 $B_{k_2}(z)$ 即为给定多项式序列 $\{A_n^t(z)\}$ 的第 k_1,k_2 个值，当 $l=1$ 时，$A_{k_1}(z)$ 与 $A_{k_2}(z)$ 即为所求值 $\bar{A}_{k_1}(z)$ 与 $\bar{A}_{k_2}(z)$。因此，只要依次按 $l=t,\cdots,2,1$，$i=0,1,\cdots,2^{l-1}-1$，$k=0,1,\cdots,N_2-1$ 计算(3.1)及(3.2)式，最后便得到结果。

下面讨论(3.1)及(3.2)式的实现，设

$$B_i(z)=\sum_{j=0}^{\frac{N}{2}-1}B(i,j)z^j \tag{3.3}$$

$$A_i(z)=\sum_{j=0}^{\frac{N}{2}-1}A(i,j)z^j$$

于是(3.1)和(3.2)式便成为

$$\sum_{j=0}^{\frac{N}{2}-1}A(k_1,j)z^j\equiv\sum_{j=0}^{\frac{N}{2}-1}B(k_1,j)z^j+z^{2^{l-1}k}\sum_{j=0}^{\frac{N}{2}-1}B(k_2,j)z^j\bmod(z^{\frac{N}{2}}+1)$$

$$\sum_{j=0}^{\frac{N}{2}-1}A(k_2,j)z^j\equiv\sum_{j=0}^{\frac{N}{2}-1}B(k_1,j)z^j-z^{2^{l-1}k}\sum_{j=0}^{\frac{N}{2}-1}B(k_2,j)z^j\bmod(z^{\frac{N}{2}}+1)$$

注意到 $z^N\equiv 1\bmod(z^{\frac{N}{2}}+1)$，$z^{\frac{N}{2}}\equiv -1\bmod(z^{\frac{N}{2}}+1)$，便有

当 $k=0$ 时，

$$\begin{cases}A(k_1,j)=B(k_1,j)+B(k_2,j)\\A(k_2,j)=B(k_1,j)-B(k_2,j)\end{cases}\quad(j=0,1,\cdots,\frac{N}{2}-1) \tag{3.4}$$

当 $k\neq 0$ 时，

$$\begin{cases}A(k_1,j)=B(k_1,j)-B(k_2,\frac{N}{2}-2^{l-1}k+j)\\A(k_2,j)=B(k_1,j)+B(k_2,\frac{N}{2}-2^{l-1}k+j)\end{cases}\quad(j=0,1,\cdots,2^{l-1}k-1) \tag{3.5}$$

$$\begin{cases}A(k_1,j)=B(k_1,j)+B(k_2,j-2^{l-1}k)\\A(k_2,j)=B(k_1,j)-B(k_2,j-2^{l-1}k)\end{cases}\quad(j=2^{l-1}k,\cdots,\frac{N}{2}-1) \tag{3.6}$$

只要按(3.4)～(3.6)计算，就实现了(3.1)及(3.2)式。

最后要说明如何由 $\{A_n(z)\}$求$\{A'_m(z)\}$。如前所述，设 $n=2^{t-1}n_{t-1}+\cdots+2n_1+n_0$（$n_i$ 不难求得），则$m=2^{t-1}n_0+\cdots+2n_{t-2}+n_{t-1}$，$m$ 求得后，如果$m>n$，则将$A_m(z)$与$A_n(z)$互换，否则不动，这样就求得了$\{A'_m(z)\}$。

综上所述，可得到FPT的算法框图，由图 3 所示。

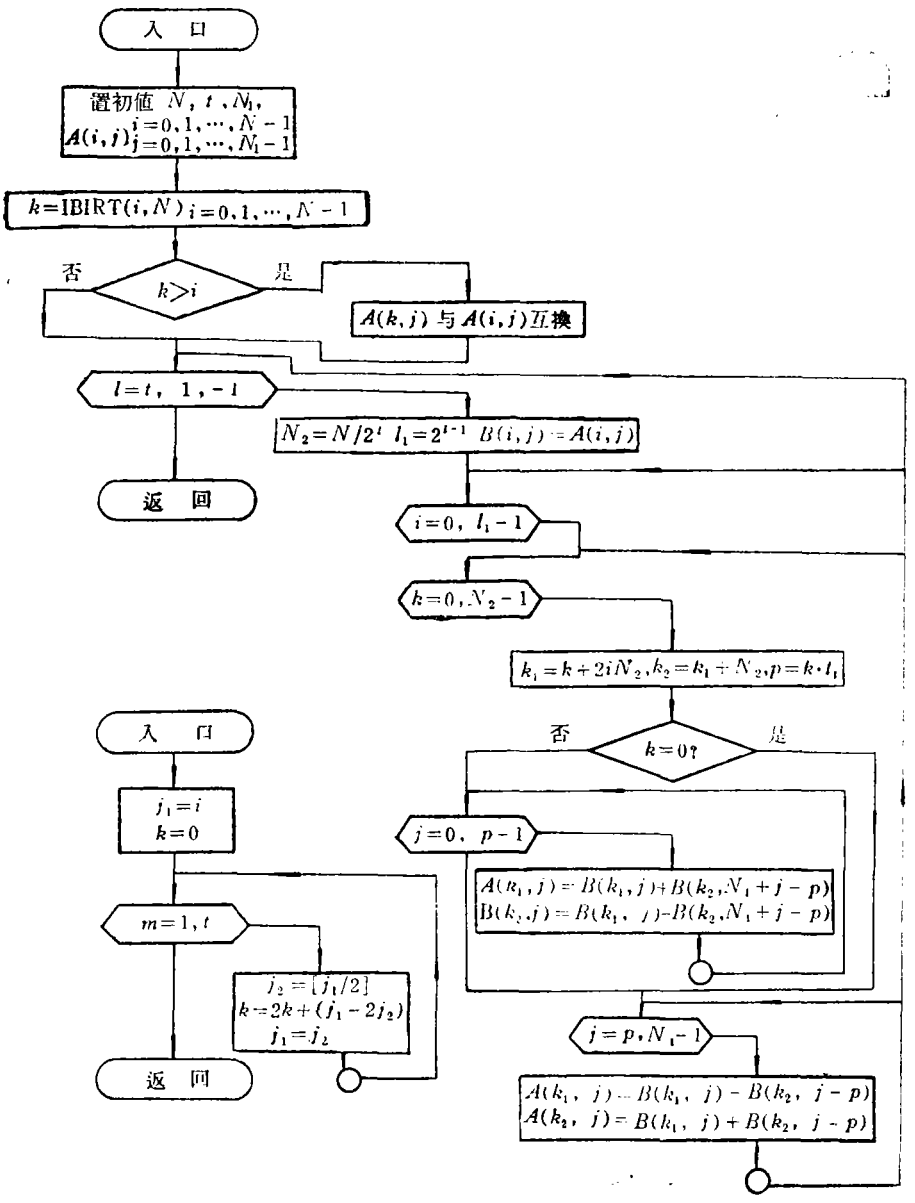

图 3 FPT计算框图

附录中给出了FPT的FORTRAN源程序，这个程序仅实现上述计算过程，不是标准的，很多地方可以改进。

参 考 文 献

[1] H.J. Nussbaumer, P.Quandalle, Fast Computation of Discrete Fourier Transforms Using Polynomial Transforms, IEEE. Trans. Acoust., Speech, Singnal Processing, Vol. ASSP-27, №.2, April 1979.

[2] 蒋增荣，用快速多项式变换(FPT)计算二维离散富里叶变换，高等学校计算数学学报，1984年第三期。

[3] 蒋增荣，多项式变换计算多项式乘积，湖南数学年刊，1982年第1、2合期。

[4] 蒋增荣，赵殿阳，二维DFT的FPT算法及其在计算机上的实现，国防科技大学学报，1984年第二期。

[5] 蒋增荣，二维数字卷积的FPT算法及其计算机上的实现，高等学校计算数学学报，1985年第三期。

[6] 蒋增荣，多项式变换及其在卷积计算中的应用，计算数学，第五卷第三期，1983.8.

附 录

这里给出多项式变换

$\bar{A}_k(z)=\sum_{n=0}^{N-1}A_n(z)(\tilde{z})^{nk}\bmod(z^M+1)(k=0,1,\cdots,N-1)$，其中$\tilde{z}=Z^{2\cdot M/N}$.快速算法的FORTRAN源程序，系数是复的。

```
      SUBROUTINE FPT(N,NU,M,AR,AI,BR,BI)
      DIMENSION AR (N,M) , AI (N,M) ,BR (N,M) ,BI (N,M)
      DO 50 I=1,N
      K=IBIRT (I-1,N) +1
      IF (K.GT.I)  GOTO 50
      DO 40 J=1,M
      TR=AR (I,J)
      TI=AI (I,J)
      AR (I,J) =AR (K,J)
      AI (I,J) =AI (K,J)
      AR (K,J) =TR
      AI (K,J) =TI
   40 CONTINUE
   50 CONTINUE
      DO 100 IL=1,NU
      L=NU-IL+1
      N2=N/2**L
      L1=2**(L-1)
      DO 60 I=1,N
      DO 60 J=1,M
      BR(I,J)=AR(I,J)
      BI(I,J)=AI(I,J)
   60 CONTINUE
```

```
      DO 100 II=1,L1
      DO 100 K=1,N2
      I=II-1
      K1=K+2*I*N2
      K2=K1+N2
      IP=L1*(K-1)*2*M/N
      IP1=IP+1
      IF(K.EQ.1) GOTO 95
      DO 90 J=1,IP
      NN=M+J-IP
      AR(K2,J)=BR(K1,J)+BR(K2,NN)
      AI(K2, J)=BI(K1, J)+BI(K2, NN)
      AR(K1, J)=BR(K1, J)-BR(K2, NN)
      AI(K1, J)=BI(K1, J)-BI(K2, NN)
   90 CONTINUE
   95 DO 100 J=IP1, M
      MM=J-IP
      AR(K2, J)=BR(K1, J)-BR(K2, MM)
      AI(K2, J)=BI(K1, J)-BI(K2, MM)
      AR(K1, J)=BR(K1, J)+BR(K2, MM)
      AI(K1, J)=BI(K1, J)+BI(K2, MM)
  100 CONTINUE
      RETURN
      END
      FUNCTION IBIRT(I, NU)
      J1=I
      IBIRT=0
      DO 10 M=1, NU
      J2=J1/2
      IBIRT=IBIRT*2+(J1-J2*2)
      J1=J2
   10 CONTINUE
      RETURN
      END
```

Fast Polynomial Transforms and It's Implemetation on General Computer

Jiang Zengrong, Zhao Dianyang

Abstract

In this paper, two algorithms of Fast Polynomial Transforms are developed. The number of times of addition needed for them is

$$A_d = MN^2 \log_2 N$$

Then, the implementation of FPT on general computer is discussed and a detialed block diagram given. A FORTRAN program is also given in the appendix.