第5章 快速傅立叶变换（FFT）

主要讨论FFT算法以及具体的实现方法

§ 5.1 DFT 的运算特点和规律

一、直接计算DFT计算量 $(a+jb)(c+jd)=(ac-bd)+j(ad+bc)$

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn}=\sum_{n=0}^{N-1}\{(\mathrm{Re}x\cdot\mathrm{Re}W-\mathrm{Im}x\cdot\mathrm{Im}W)+j(\mathrm{Re}x\cdot\mathrm{Im}W+\mathrm{Im}x\cdot\mathrm{Re}W)\}$$

$k=0,1,\ldots N-1$

计算一个k值（如k=0）的$X(k)$运算量：

N次 复数乘　　(N-1)次 复数加

或4N次 实数乘　　2N+2(N-1)=4N-2次 实数加

全部计算N个X(k)（$k=0,1,\ldots N-1$）

N^2 次 复数乘　　N(N-1)次 复数加

或$4N^2$次 实数乘　　N(4N-2)次 实数加

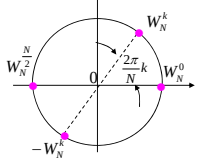
结论：直接计算N点DFT的计算量和N^2成正比

例如： 10点 DFT　　100次 复数乘

1024点 DFT　　1,048,576次 复数乘，

即100多万次复数乘运算

二、W_N 因子的特性



1.周期性
$$\begin{cases}W_N^{nk}=W_N^{n(N+k)}=W_N^{k(N+n)}\\ W_N^0=W_N^{rN}=1\end{cases}$$

2.对称性
$$\begin{cases}W_N^{-nk}=W_N^{n(N-k)}=W_N^{k(N-n)}\\ W_N^{(k+\frac{N}{2})}=-W_N^k\\ W_N^{\frac{N}{2}}=-W_N^0=-1\end{cases}$$

3.可约性
$$\begin{cases}W_N^{nk}=W_{mN}^{mnk}=W_{N/m}^{nk/m}\\ W_N^{N/2}=-1\\ W_N^{(k+N/2)}=-W_N^k\end{cases}$$

如何利用W_N因子的周期性和对称性等，导出一个高效的快速算法？

1965年 Cooley & Tukey 奠定FFT基础，把长序列短分解

使得乘法计算量由N^2次降为$\frac{N}{2}\log_2 N$次

以1024点为例，计算量降为5120次，仅为原来的 4.88%

4.正交性
$$\sum_{k=0}^{N-1}W_N^{kn}=\frac{1-W_N^{nN}}{1-W_N^n}=\frac{1-e^{-j\frac{2\pi}{N}nN}}{1-e^{-j\frac{2\pi}{N}n}}=\begin{cases}N & \text{当}n=rN,\ r\text{为整数}\\ 0 & \text{当}n\text{为其它}\end{cases}$$

三、 结论

快速傅里叶变换（FFT: Fast Fourier Transform）就是在这些特性基础上发展起来的：

（1）利用DFT旋转因子的对称性、周期性和可约性等特性，合并DFT运算中的某些项；

（2）将长序列分解为短序列，从而减少其运算量。

（3）因合并与分解方法的不同产生了多种FFT算法。

如果将序列$x(n)$的长度N取为2的整数幂次方$N=2^M$（M为正整数），这种FFT运算称为基--2FFT算法。基--2FFT算法可以按抽取方法的不同分为：

按时间抽取（DIT：Decimation-in-time）FFT算法

按频率抽取（DIF： Decimation-in-frequency）FFT算法。

§5.2 基2-FFT算法

§5.2.1 按时间抽取(DIT)的基2-FFT算法（库利-图基算法）

一、算法原理（时域奇偶分，频域前后分）

DFT计算公式为： $X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn}$

设*DFT*长度*N*，$N=2^M$，*M*为自然数，序列*x*(n)不够长可以补零为N，将长度为*N*的序列*x*(n)按时间顺序，奇偶分解为越来越短的子序列，称为按时间抽取(DIT：Decimation-In-Time)的FFT算法，也称Cooley-Tukey算法。

1、第一次分解：

将*x*(*n*)按奇、偶分成两组，可得各长为*N*/2的两个新序列

$$x(n)\longrightarrow\begin{cases}x_1(r)=x(2r)\\x_2(r)=x(2r+1)\end{cases}\qquad r=0,1,\ldots\frac{N}{2}-1$$

$x(n)$的DFT为 $X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn}=\sum_{n=偶数}x(n)W_N^{kn}+\sum_{n=奇数}x(n)W_N^{kn}$

$W_N^{2kr}=W_{N/2}^{kr}$

$$=\underbrace{\sum_{r=0}^{\frac{N}{2}-1}x(2r)W_N^{2kr}}_{n为偶数}+\underbrace{\sum_{r=0}^{\frac{N}{2}-1}x(2r+1)W_N^{k(2r+1)}}_{n为奇数}$$

$$X_1(k)=\sum_{r=0}^{N/2-1}x(2r)W_{N/2}^{kr}$$

$$X_2(k)=\sum_{r=0}^{N/2-1}x(2r+1)W_{N/2}^{kr}$$

$$=\underbrace{\sum_{r=0}^{\frac{N}{2}-1}x_1(r)W_{N/2}^{kr}}_{X_1(k)}+W_N^k\underbrace{\sum_{r=0}^{\frac{N}{2}-1}x_2(r)W_{N/2}^{kr}}_{X_2(k)}\qquad r=0,1,\ldots N/2-1$$

$$\therefore X(k)=X_1(k)+W_N^kX_2(k)\qquad k=0,1,\ldots N/2-1$$

X(*k*)的前半（N/2）部分　　*X*(*k*)后半（N/2）部分？$X(k+\frac{N}{2})$

X(*k*)后半（N/2）部分？ $k=0,1,\ldots,N/2-1$

- $X_1(k)$和$X_2(k)$的长度也是N/2;

$$X(k+\frac{N}{2})=X_1(k+\frac{N}{2})+W_N^{k+\frac{N}{2}}X_2(k+\frac{N}{2})=X_1(k)-W_N^kX_2(k)$$ ？

$$\because W_N^{(k+\frac{N}{2})}=W_N^{\frac{N}{2}}W_N^k=-W_N^k$$

求出*N*/2点DFT $X_1(k)$和$X_2(k)$ →

$$\begin{cases}X(k)=X_1(k)+W_N^kX_2(k)\\X(k+\frac{N}{2})=X_1(k)-W_N^kX_2(k)\end{cases}$$

这样，将一个*N*点DFT分解成2个*N*/2点DFT（$X_1(k)$和$X_2(k)$），只要求出N/2点的$X_1(k)$和$X_2(k)$，即可求出k=0,1,2,…，N-1区间全部X(*k*)值，这正是FFT能大量节省计算量的关键所在。

$x(n)$ 一次分解后：

$$\begin{cases}x_1(r)=x(2r)\\x_2(r)=x(2r+1)\end{cases}\qquad r=0,1,\ldots\frac{N}{2}-1$$

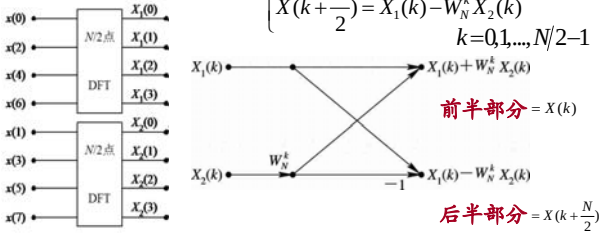
$$X_1(k)=\sum_{r=0}^{N/2-1}x_1(r)W_{N/2}^{kr}$$

$$X_2(k)=\sum_{r=0}^{N/2-1}x_2(r)W_{N/2}^{kr}$$

$k=0,1,\ldots N/2-1$　2个$\frac{N}{2}$点的D FT

$$\begin{cases}X(k)=X_1(k)+W_N^kX_2(k)\\X(k+\frac{N}{2})=X_1(k)-W_N^kX_2(k)\end{cases}$$

$k=0,1,\ldots,N/2-1$



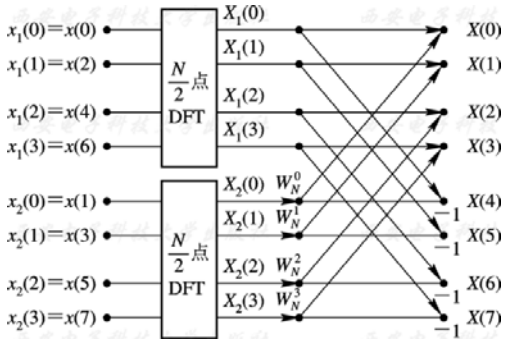


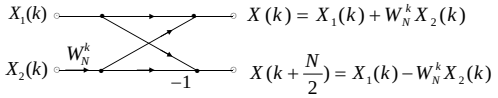
图 5.1 按时间抽取将一个*N*点DFT分解为两个*N*/2点DFT（*N*=8）

2、蝶形运算

一个蝶形运算：几次复数乘？几次复数加？

一次复数乘、两次复数加

计算一个$X(k)$的运算信号流图：



用下面的蝶形图也可清楚地说明这种运算。

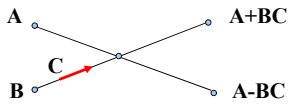


图5.2 DIT-FFT 蝶形运算符号

A: $X_1(k)$

B: $X_2(k)$

C: W_N^k

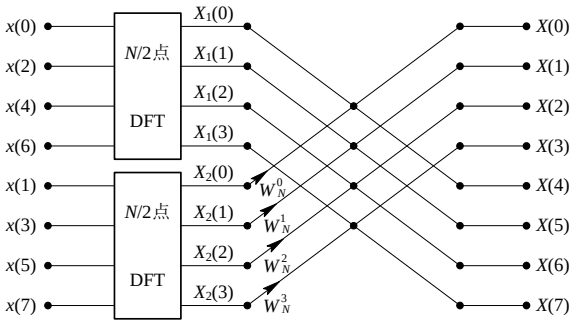


图5.3 N点DFT的一次时域抽取分解图（N=8）

一次时域分解后计算一个N点DFT的计算量？

一个N点$X(k)$ $\begin{cases} X_1(k),X_2(k) & \text{两个N/2点DFT} \quad \text{复数乘 } 2\times(N/2)^2 \quad \text{复数加 } 2\cdot\frac{N}{2}(\frac{N}{2}-1) \\ \text{N/2个 碟形运算} & \text{复数乘 } N/2 \quad \text{复数加 } 2\cdot\frac{N}{2} \end{cases}$

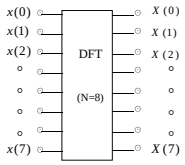
计算一个N点DFT:

复数乘 $2\times(N/2)^2+N/2=N(N+1)/2$

复数加 $2\cdot\frac{N}{2}(\frac{N}{2}-1)+2\cdot\frac{N}{2}=N^2/2$

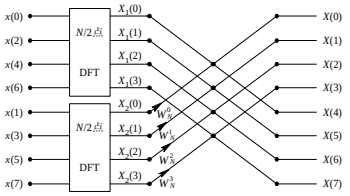
运算量减少近一半

以$N=8$为例,经过一次分解计算量比较



直接计算

需要8×8次复数乘、8×7次复数加



36次复数乘

32次复数加

3、进一步分解（第二次分解）

将$x_1(r)$按奇偶分解成两个**N/4**长的子序列 $x_3(l)$ 和$x_4(l)$

$$x_1(r) \rightarrow \begin{cases} x_3(l)=x_1(2l) \\ x_4(l)=x_1(2l+1) \end{cases} \quad l=0,1,\ldots\frac{N}{4}-1$$

$$X_1(k)=\sum_{l=0}^{N/4-1}x_1(2l)W_{N/2}^{2kl}+\sum_{l=0}^{N/4-1}x_1(2l+1)W_{N/2}^{k(2l+1)}$$

$$=X_3(k)+W_{N/2}^kX_4(k) \qquad k=0,1,\ldots\frac{N}{4}-1$$

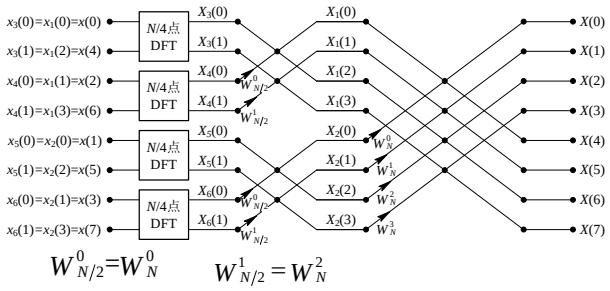
于是 $\begin{cases} X_1(k) & =X_3(k)+W_{N/2}^kX_4(k) \\ X_1(k+N/4) & =X_3(k)-W_{N/2}^kX_4(k) \end{cases}$ $k=0,1,\ldots\frac{N}{4}-1$

同样将$x_2(r)$按奇偶分解成两个$N/4$长的子序列 $x_5(l)$和 $x_6(l)$

$$x_2(r) \rightarrow \begin{cases} x_5(l)=x_2(2l) \\ x_6(l)=x_2(2l+1) \end{cases} \quad l=0,1,\ldots\frac{N}{4}-1$$

$$\begin{cases} X_2(k) & =X_5(k)+W_{N/2}^kX_6(k) \\ X_2(k+N/4) & =X_5(k)-W_{N/2}^kX_6(k) \end{cases} \quad k=0,1,\ldots\frac{N}{4}-1$$

以$N=8$为例,经过第二次分解



$W_{N/2}^0=W_N^0$ $W_{N/2}^1=W_N^2$

图5.4 8点DFT的二次时域抽取分解图（N=8）

4、N=8点DFT的第三次分解（2点DFT的计算）

当N=8，最后剩下的是4个N/4=2的2点DFT，

2点DFT也可以用蝶型运算来完成，以 $X_3(k)$ 为例:

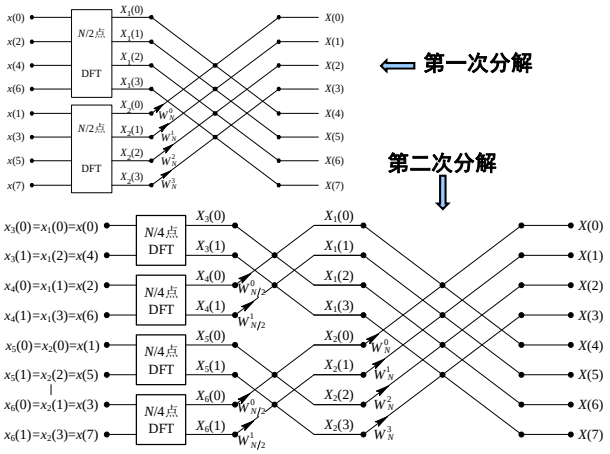
$$X_3(k)=\sum_{l=0}^{N/4-1}x_3(l)W_{N/4}^{lk}=\sum_{l=0}^{1}x_3(l)W_{N/4}^{lk},k=0,1$$

$$\because x_3(0)=x(0),x_3(1)=x(4),W_{N/4}^1=W_{N/4}^1W_{N/4}^0=-W_{N/4}^0$$

$$\therefore\begin{cases}X_3(0)=x_3(0)W_{N/4}^0+W_{N/4}^0x_3(1)=x(0)+W_N^0x(4)\\X_3(1)=x_3(0)W_{N/4}^0+W_{N/4}^1x_3(1)=x(0)-W_N^0x(4)\end{cases}$$

$$\begin{cases}X_3(0)=x(0)+x(4)\\X_3(1)=x(0)-x(4)\end{cases}$$ 同理 $$\begin{cases}X_4(0)=x(2)+x(6)\\X_4(1)=x(2)-x(6)\end{cases}$$

2点的DFT只有加减运算



第三次分解:

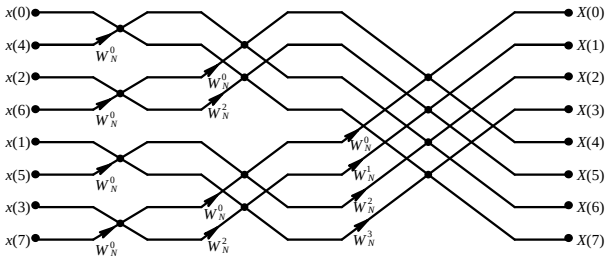


图5.5 按时间抽取基2-FFT算法流图（N=8）

第一次次分解，将N点的DFT分解成两个$N/2$点的DFT和$N/2$个蝶形运算

第二次分解，将每个$N/2$点的DFT分解成两个$N/4$点的DFT和$N/4$个蝶形运算

依次类推，经过M次分解，最后将N点DFT分解成$N/2$个2点DFT

$N=2^M$点DFT ⟹ M次分解 ⟹ $N/2$个2点DFT

按照这种方法不断划分下去，直到最后剩下的是2点只有加减运算的DFT为止。

二、算法讨论

1. 级的概念

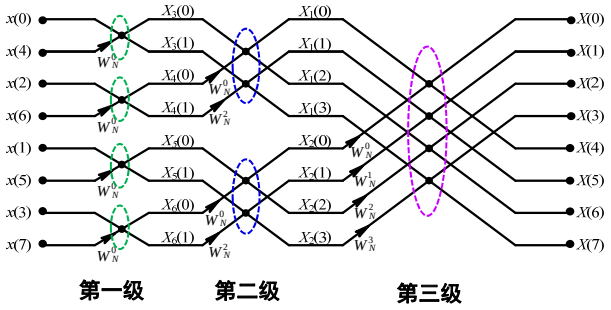
此算法以2为基，写作$N=2^M$（不够长，补零延伸）

上述DIT-FFT算法过程，将N点DFT先分成两个$N/2$点DFT，再……直至N/2个两点DFT。每分解一次，称为一“级”运算。$N=2^M$点DFT可以分成M级，从左到右依次是1，2，…，M级，每级有N/2个蝶形

$$M=\log_2 N$$

每级有多少个蝶形?每级多少次复数乘法和复数加法?

N/2个蝶形，N/2次复数乘，N次复数加



2. DIT-FFT与DFT算法运算量比较

FFT: 全部“蝶形”数= $\frac{N}{2}M=\frac{N}{2}\log_2 N$

运算量 { 复数乘法次数 $\frac{N}{2}M$ 在N值很大时，
复数加法次数 NM 都 <<N^2，十分高效

DFT:复数乘法次数 N^2,

复数加法次数 $N(N-1)$

例如，$N=2^{10}=1024$时

运算效率：$\frac{N^2}{(N/2)\log_2 N}=\frac{1048576}{5120}=204.8$

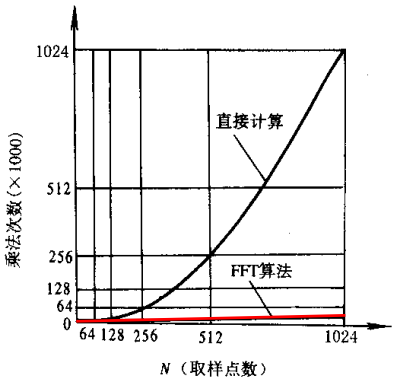
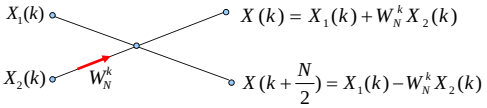
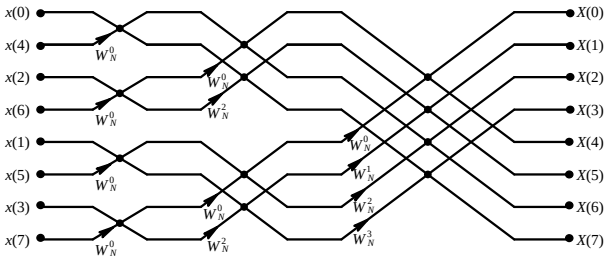


图5.6 FFT算法与直接计算DFT所需乘法次数的比较曲线

三. FFT算法的特点

1、蝶形运算





2 原位计算：

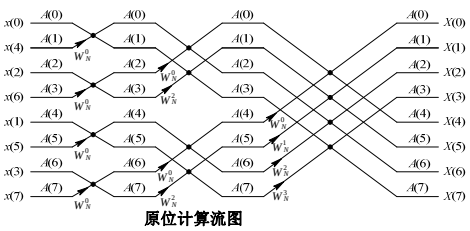
$N=2^M$点的FFT共进行M级运算，每级由$N/2$个蝶形运算组成

设 N个存贮单元 A(0) A(1) A(2) A(3) A(4) A(5) A(6) A(7)

存入数据 x(0) x(4) x(2) x(6) x(1) x(5) x(3) x(7)

每次运算结果存入原输入数据占用的存贮单元

这种利用同一存贮单元存贮蝶形计算输入输出数据的方法称为原位（同址）计算



原位计算流图

这样整个N点FFT运算只需N个存储单元

原位计算可节省大量内存，使设备成本降低。

3 码位倒置：

由DIT－FFT流图可以看出，变换后的输出 $X(k)$ 依照自然顺序排列，输入序列$x(n)$不再是原来的自然顺序，这是由于对$x(n)$作奇偶抽取所产生的。

对N＝8，其自然序号n是 0，1，2，3， 4，5，6，7

$x(n)$第一次按奇偶分： 0，2，4，6 | 1，3，5，7

偶数、奇数组再作奇偶分： 0，4 | 2，6 | 1，5 | 3，7

把这种序号的排列顺序称之为倒位序

序号n用二进制数表示$n_2n_1n_0$

顺位序 n	二进制顺序码 $n_2n_1n_0$	二进制倒置码 $n_0n_1n_2$	倒位序 $\hat{n}$
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

先按自然顺序输入 → 变址运算 → 将顺序码的二进制位倒置 → 倒位序

n	$x(n)$	$X(k)$	k
0	000	000	0
4	100	001	1
2	010	010	2
6	110	011	3
1	001	100	4
5	101	101	5
3	011	110	6
7	111	111	7

第十六次作业：

5.1 画出N=4点的按时间抽取基2-FFT算法的完整流图。

5.2； 5.3 ；

§ 5.2.2 按频率抽取(DIF)的基2-FFT算法（桑德-图基算法）

一、算法原理（时域前后分，频域奇偶分）

设*DFT*长度N，$N=2^M$，M为自然数，序列$x(n)$不够长可以补零为N，将长度为N的序列$x(n)$分成前后两段：

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn}=\sum_{n=0}^{\frac{N}{2}-1}x(n)W_N^{kn}+\sum_{n=\frac{N}{2}}^{N-1}x(n)W_N^{kn}$$

令后者的$n=m+N/2$，得：

$$\sum_{n=\frac{N}{2}}^{N-1}x(n)W_N^{kn}=\sum_{m=0}^{\frac{N}{2}-1}x(m+\frac{N}{2})W_N^{km}W_N^{k\frac{N}{2}}$$

$$\underline{\underline{m=n}}\sum_{n=0}^{\frac{N}{2}-1}W_N^{k\frac{N}{2}}x(n+\frac{N}{2})W_N^{kn}$$

$$\therefore\ X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn}=\sum_{n=0}^{\frac{N}{2}-1}[x(n)+W_N^{k\frac{N}{2}}x(n+\frac{N}{2})]W_N^{kn}$$

$$k=0,1,\cdots,N-1$$

$$W_N^{k\frac{N}{2}}=(-1)^k=\begin{cases}1 & k\text{为偶数}\\ -1 & k\text{为奇数}\end{cases}$$

将$X(k)$分解成偶数组和奇数组：$\begin{cases}k=2r\\ k=2r+1\end{cases}$，$r=0,1,\cdots,\frac{N}{2}-1$

$$X(2r)=\sum_{n=0}^{N/2-1}[x(n)+x(n+\frac{N}{2})]W_N^{2rn}=\sum_{n=0}^{N/2-1}x_1(n)W_{N/2}^{rn}=X_1(k)$$

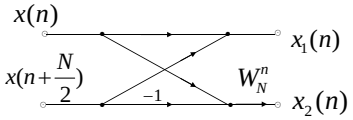
$$X(2r+1)=\sum_{n=0}^{N/2-1}[x(n)-x(n+\frac{N}{2})]W_N^{(2r+1)n}$$

$$=\sum_{n=0}^{N/2-1}\{[x(n)-x(n+\frac{N}{2})]W_N^n\}W_N^{2rn}=\sum_{n=0}^{N/2-1}x_2(n)W_{N/2}^{rn}=X_2(k)$$

$$k=0,1,\ldots.\frac{N}{2}-1$$

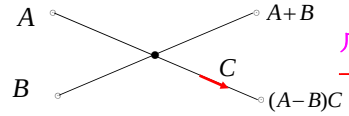
二、蝶形运算单元

其中 $\begin{cases}x_1(n)=x(n)+x(n+\frac{N}{2})\\ x_2(n)=[x(n)-x(n+\frac{N}{2})]W_N^n\end{cases}$ $n=0,1,\ldots.\frac{N}{2}-1$



$n=0,1,..\frac{N}{2}-1$

运算流图



一个蝶形运算：
几次复数乘？几次复数加？
一次复数乘、两次复数加

图5.7 DIF-FFT蝶形运算单元

第一次分解：

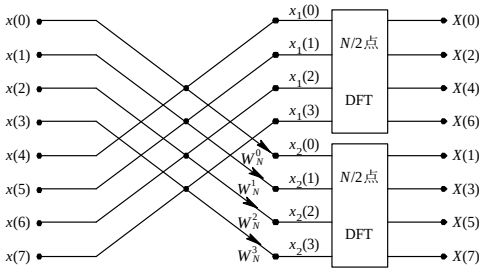


图5.8 DIF—FFT一次分解运算流图(N=8)

同理，第二次分解：将$x_1(n)$、$x_2(n)$再进行前后分解做N/4的DFT

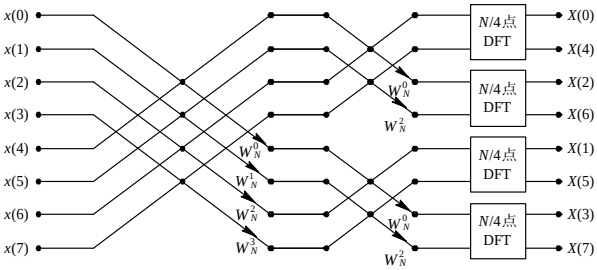


图5.9 DIF—FFT二次分解运算流图(N=8)

第三次分解：

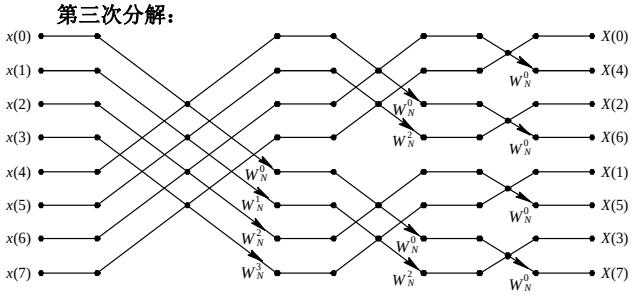


图5.10 DIF—FFT运算流图(N=8)

1.蝶形单元； 2.原位计算；3.级的概念；4.码位倒置；

DIF-FFT和DIT-FFT具有一样的运算量：

$\frac{N}{2}\log_2 N$ 次复数乘法　　$N\log_2 N$ 次复数加法

§ 5.3 离散傅立叶反变换(IDFT)的快速计算方法

由定义：

$$\begin{cases} x(n) = IDFT[X(k)] = \dfrac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-kn} \\ X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n)W_N^{kn} \end{cases}$$

两者作比较，得到启发,IDFT是否也可以应用FFT？

方法一：

修改DFT运算中的各个系数W_N^{kn} → W_N^{-kn},最后乘以1/N

为防止溢出将常数 $\frac{1}{N}$ 分解 $(\frac{1}{2})^M$ 分散到各级中，即每一级都乘以因子$\frac{1}{2}$

不足：需修改FFT的程序和参数

方法二：直接调用FFT子程序计算IFFT的方法：

优点：利用共轭变换的方法,不修改FFT的程序和参数.

$$\because x(n) = IDFT[X(k)] = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-kn}$$

$$\therefore \quad x^*(n) = \frac{1}{N}\sum_{k=0}^{N-1} X^*(k)W_N^{kn}$$

$$\therefore x(n) = \left[x^*(n)\right]^* = \frac{1}{N}[\sum_{k=0}^{N-1} X^*(k)W_N^{kn}]^* = \frac{1}{N}\{DFT[X^*(k)]\}^*$$

即

$X(k)$ → 取共轭 → $X^*(k)$ → FFT → 取共轭 → 乘1/N → $x(n)$

不足：做两次复共轭运算

§ 5.4 实序列的FFT算法

法1 设$x_1(n)$、$x_2(n)$为两个N点实序列，用一个N点FFT同时计算两个N点实序列的DFT，一个作为实部，另一个作为虚部，计算完后再把输出按奇、偶、虚、实特性加以分离，具体步骤如下：

步骤1：构造复序列y (n)：$y(n) = x_1(n) + jx_2(n)$

步骤2：求y (n)的N点FFT，记为Y(k)：Y(k)=FFT[y(n)]

步骤3：根据对称性求出X_1 (k)和X_2 (k)

$$X_1(k) = Y_{ep}(k) = \frac{Y(k)+Y^*(N-k)}{2}$$

$$X_2(k) = Y_{op}(k) = \frac{Y(k)-Y^*(N-k)}{2j}$$

法2 若只有一个N点实序列 $x(n)$，可以将其分解为两个N/2点实序列，记为 $x_1(r)$ 和 $x_2(r)$，然后仿照方法1计算两个N/2的X_1 (k)和 X_2 (k)，最后根据对称性求出X (k)。具体步骤如下：

步骤1：将N点实序列 $x(n)$分解成两个N/2点的实序列，分解方法采用按时间奇偶抽取方法：

$$\left.\begin{aligned} x_1(r) &= x(2r) \\ x_2(r) &= x(2r+1) \end{aligned}\right\} r = 0,1,2\cdots,N/2-1$$

步骤2：构造复序列y (r)：$y(r) = x_1(r) + jx_2(r)$

$$r = 0,1,2\cdots,N/2-1$$

步骤3：求y (r)的N/2点FFT，记为Y(k)：

$$Y(k)=FFT[y(r)]_{N/2}$$

步骤4：根据对称性求出X_1 (k)和X_2 (k)

$$X_1(k)=Y_{ep}(k)=\frac{Y(k)+Y^*(\frac{N}{2}-k)}{2}$$

$$X_2(k)=Y_{op}(k)=\frac{Y(k)-Y^*(\frac{N}{2}-k)}{2j}$$

步骤5：按DIT-FFT算法的碟形公式求出X (k)

$$X(k) = X_1(k) + W_N^k X_2(k)$$

$$X(k+\frac{N}{2}) = X_1(k) - W_N^k X_2(k)$$

$$k = 0,1,\ldots,N/2-1$$

第5章小结

DFT与FFT快速算法的运算量比较

FFT 算法的基本特点：

1.蝶形单元； 2.原位计算；3.级的概念；4.码位倒置；

会画DIT-FFT与DIF-FFT的4、8点运算流图。

第十七次作业：

5.1画出N=4点的按频率抽取基2-FFT算法的完整流图。

5.4；5.5

补充题：

采用FFT算法，可用循环卷积完成线性卷积。计算线性卷积$x(n)*h(n)$，试写采用快速卷积的计算步骤（注意说明点数）。