第五章 快速傅里叶变换FFT

郑南宁 教授

本章主要内容

- FFT算法的基本原理
- 基2FFT算法的理论推导
- 按时间抽取的FFT算法
- 按频率抽取的FFT算法
- IDFT的快速运算方法
- 利用FFT计算线性卷积和线性相关

5.1 FFT算法的基本原理

矩阵方程 - DFT的计算量分析

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

上式表示N个方程的计算。若 N=4，上式可以展开为N个方程求解形式

$$\begin{aligned}
X(0)&=x(0)W^0+x(1)W^0+x(2)W^0+x(3)W^0\\
X(1)&=x(0)W^0+x(1)W^1+x(2)W^2+x(3)W^3\\
X(2)&=x(0)W^0+x(1)W^2+x(2)W^4+x(3)W^6\\
X(3)&=x(0)W^0+x(1)W^3+x(2)W^6+x(3)W^9
\end{aligned}$$

式中 $W=e^{-j(2\pi/N)}$。为方便起见，这里用W^{nk}替代W_N^{nk}。

可进一步表示成矩阵运算形式

$$\begin{bmatrix}X(0)\\X(1)\\X(2)\\X(3)\end{bmatrix}=\begin{bmatrix}W^0 & W^0 & W^0 & W^0\\ W^0 & W^1 & W^2 & W^3\\ W^0 & W^2 & W^4 & W^6\\ W^0 & W^3 & W^6 & W^9\end{bmatrix}\begin{bmatrix}x(0)\\x(1)\\x(2)\\x(3)\end{bmatrix}$$

把前面的矩阵形式重写如下

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \quad (5.1A)$$

或更紧凑地表示成

$$\boldsymbol{X(k) = W^{nk} x(n)}$$

考察式(**5.1**A)，可以看到，由于$\boldsymbol{W}$是复数，$\boldsymbol{x(n)}$也可能是复数，因此

1、每计算一个$X(k)$，需要 N 次复数乘法，N-1 次复数加法；

2、完成矩阵运算，即整个DFT运算，就需要 N^2 次复数乘法和 $N(N-1)$ 次复数加法；

3、每个复数乘法包含4次实数乘法和2次实数加法，每个复数加法包含2次实数加法。

例如：

当N=8，复乘次数=64，复加次数=56；

当N=1024，复乘次数 $(1024)^2$ =1048576，复加次数：1024 × （1024-1）=1047552

DFT的复数相乘与复数相加的运算复杂度都为O(N^2)，计算量大，难以做到实时处理

■ 直观上讨论如何减少计算DFT所需的乘法和加法次数

把矩阵(5.1A) 重写如下

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

(5.1A) 改写为

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^0 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \quad (5.1B)$$

直观推导（第一步）： 利用关系 $W^{nk} = W^{((nk))_N}$

因此，如 $N = 4, n = 2, k = 3$时，则 $W^6 = W^2$

这是因为

$$W^{nk} = W^6 = e^{\left(-j\frac{2\pi}{4}\right)(6)} = e^{-j3\pi}$$

$$= W^{((nk))_N} = W^2 = e^{\left(-j\frac{2\pi}{4}\right)(2)} = e^{-j\pi}$$

直观推导（第二步）

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^0 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad (5.1\text{C})$$

将式(5.1C)中的矩阵$[W^{nk}]$分解因子成如下形式

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad (5.1\text{D})$$

式(5.1D)与式(5.1C)比较，可以看到，式(5.1D)两个方阵相乘得到式(5.1C)的方阵，但其中第一行和第二行相互交换了，注意，考虑这里的行交换，需要改写式(5.1D)的列矢量$\boldsymbol{X(k)}$，改写后的矢量用$\overrightarrow{\boldsymbol{X(k)}}$表示：

$$\overrightarrow{\boldsymbol{X(k)}} = \begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} \qquad (5.1\text{E})$$

矩阵因子分解是FFT算法之所以有效的关键（注意其变换结果$\overrightarrow{\boldsymbol{X(k)}}$是乱序的）

重写式（5.1D）

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \quad (5.1\text{D})$$

承认上式的正确性（尽管结果是乱序的），就可以进一步讨论计算上述这个方程的乘法次数

首先设（先给出上式等号最右边两个矩阵相乘-中间计算步骤）

$$\begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \quad (5.1\text{F})$$

即，列矢量$\boldsymbol{x_1(n)}$等于式(5.1D)右边两个矩阵的乘积，元素$x_1(0)$的计算要用一次复数乘法和一次复数加法来确定，即

$$\boxed{x_1(0) = x(0) + W^0 x(2)} \quad (5.1\text{G})$$

为了推得普遍的结果，这里W^0不化为1

$\boldsymbol{x_1}(1)$的计算也是用一次复数乘法和一次复数加法来确定，但计算$x_1(2)$只要一次复数加法，这是因为，$W^0 = -W^2$，**因此**

$$\boxed{x_1(2) = x(0) + W^2 x(2) = x(0) - W^0 x(2)} \quad (5.1\text{H})$$

上式中的复数乘法$W^0 x(2)$已在计算$x_1(0)$时计算过了。

同理，计算$x_1(3)$也只要一次复数加法，不需要乘法。这样中间矢量 $\boldsymbol{x_1(n)}$ 只需要四次复数加法，不需作乘法。

下面继续完成式(5.1D)的计算（计算列矢量$x_2(0)$）

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad (5.1\text{D})$$

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} x_2(0) \\ x_2(1) \\ x_2(2) \\ x_2(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} \qquad (5.1\text{I})$$

$x_2(0)$项可用一次复数乘法和一次复数加法来确定，即

$$\boxed{x_2(0) = x_1(0) + W^0 x_1(1)} \qquad (5.1\text{J})$$

$x_2(1)$的计算，只要一次复数加法，因为，$W^0 = -W^2$。同样，$x_2(2)$只要一次复数乘法和一次复数加法，而$x_2(3)$的计算只要一次复数加法。

因此，用式(5.1D)计算$\overrightarrow{\boldsymbol{X(k)}}$，总共需4次复数乘法和8次复数加法，而直接计算DFT需要16次复数乘法和12次复数加法。

在前面计算式(5.1D)时，矩阵分解因子过程中，由于把零引入了被分解的矩阵，从而减少了乘法次数。

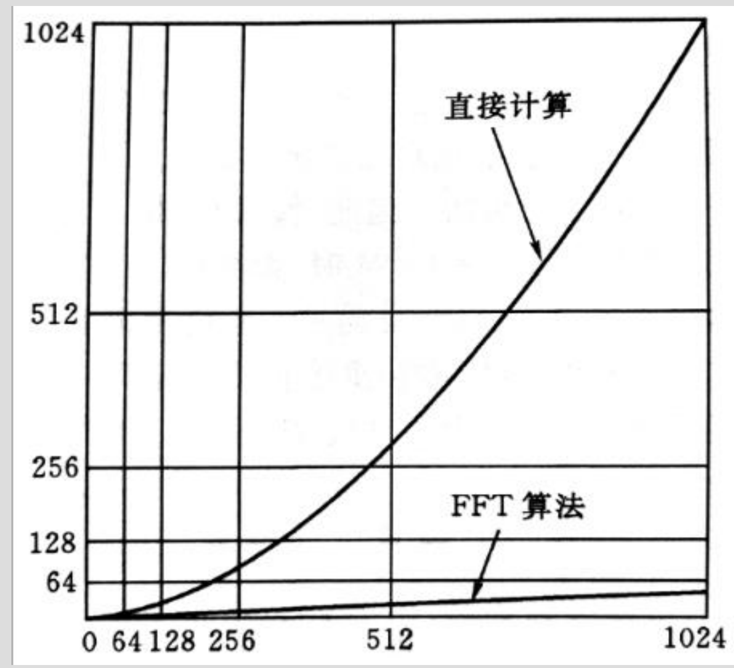
- 对于$N=2^M$ 的FFT算法，简单地讲，就是将N x N的矩阵分解为M 个N x N 矩阵（每一个矩阵的具有复数加法和复数加法次数最少的特性)
- 将大 N 的DFT分解为若干小点数的DFT的组合，使整个DFT的计算过程变成一系列迭代运算过程
- 引伸上面例子的结果，可以看到，N=2^M 的FFT算法只需要$N \cdot M/2$次复数乘法和 $N \cdot M$次复数加法，而直接计算DFT需要N^2次复数乘法和 $N(N-1)$次复数加法。

若只考虑乘法次数，直接计算DFT与FFT的计算时间之比的近似关系

$$\frac{N^2}{N \cdot M/2}=\frac{2N}{M}$$

对于N=1024=2^{10},可以使计算量减少到约200比1

计算复杂度从 $O(N^2)$ 下降为 $O(N\log(N))$



N点DFT和FFT所需乘法次数的比较

在前面计算式(5.1D)时, 矩阵分解因子过程中引进了一个差异，在计算式(5.1D)时，得到的是$\overline{\boldsymbol{X}(\boldsymbol{k})}$，而不是$\boldsymbol{X}(\boldsymbol{k})$, 即

$$\overline{\boldsymbol{X}(\boldsymbol{k})}=\begin{bmatrix}X(0)\\X(2)\\X(1)\\X(3)\end{bmatrix}\quad \text{代替了}\quad \boldsymbol{X}(\boldsymbol{k})=\begin{bmatrix}X(0)\\X(1)\\X(2)\\X(3)\end{bmatrix}\qquad (5.1\mathrm{K})$$

这种重新排列，是矩阵分解过程固有的。可直接对$\overline{\boldsymbol{X}(\boldsymbol{k})}$重新排序得到$\boldsymbol{X}(\boldsymbol{k})$

对结果重新排序

用相应的二进制数代替自变量k，重写$X(k)$

$$\begin{bmatrix}X(0)\\X(2)\\X(1)\\X(3)\end{bmatrix}\quad \text{变成}\quad \begin{bmatrix}X(00)\\X(10)\\X(01)\\X(11)\end{bmatrix}$$

将上面矩阵的二进制数位序反转

$$\overline{\boldsymbol{X}(\boldsymbol{k})}=\begin{bmatrix}X(00)\\X(10)\\X(01)\\X(11)\end{bmatrix}\quad \text{反转}\quad \begin{bmatrix}X(00)\\X(01)\\X(10)\\X(11)\end{bmatrix}=\boldsymbol{X}(\boldsymbol{k})$$

以上讨论了 N =4的 FFT算法，对于$N>4$ 的FFT算法按式(5.1D)的方式描述矩阵因子分解过程比较麻烦，后面将用图解的方式讨论式(5.1D)的计算过程, 用这种形式导出计算机的流程图

■ 减少DFT计算量的依据

DFT的公式

$$\boxed{X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk}, \qquad k=0,1,\cdots,N-1}$$

改进DFT计算效率的多数方法利用了W_N^{nk}的周期性和对称性，即

W_N^{nk}的周期性 $\quad W_N^{nk}=W_N^{(n+N)k}=W_N^{n(k+N)}$

W_N^{nk}的共轭对称性 $\quad W_N^{-nk}=(W_N^{nk})^*=W_N^{k(N-n)}$

且由于$W_N^{N/2}=e^{-j\pi}=-1$，有

$$\boxed{W_N^{(k+N/2)}=-W_N^{k}}$$

同时利用W_N^{nk}的对称性和周期性可以大大减少DFT的计算量

举例 对实数序列$x(n)$，利用权重的对称性，则可将其DFT公式中含有 n 和 N-n 的项组合在一起

$$\underline{x(n)W_N^{nk}+x(N-n)W_N^{(N-n)k}}$$

$$=x(n)W_N^{nk}+x(N-n)\left(W_N^{nk}\right)^*$$

$$=\underline{[x(n)+x(N-n)]\mathrm{Re}\left(W_N^{nk}\right)+j[x(n)-x(N-n)]\mathrm{Im}\left(W_N^{nk}\right)}$$

原式需**4**次实数乘法

变形以后需**2**次实数乘法

对其它项也做类似的组合处理，计算DFT的乘法次数就减少一半

■ 实数序列的FFT算法

通常考虑时间的实函数，而频率函数通常是复的，因此要设计一个即能减少计算DFT，又能计算IDFT的程序，就要假设一个复序列$x(n)$的FFT

$$X(k)=\sum_{n=0}^{N-1}[x_r(n)+jx_i(n)]e^{-j2\pi nk/N}$$

因为，根据复数共轭的反变换公式可以得到：

$$x(n)=\frac{1}{N}\left[\sum_{k=0}^{N-1}\left[\mathrm{Re}\big(X(k)\big)+j\mathrm{Im}(X(k))\right]^{*}e^{-j2\pi nk/N}\right]^{*}$$

由于上面两式中，包含着共同的因子$e^{-j2\pi nk/N}$，因此，用同一个程序就可以计算DFT和它的反变换。

■ 利用复时间函数的虚部，使实函数的FFT计算有更高的效率

□ 同时计算两个实序列的FFT

我们希望用下列复序列的形式：

$$\boxed{v(n) = x(n) + jy(n)}$$

同时计算两个实序列$x(n)$和$y(n)$的DFT 。也就是说，$v(n)$是由两个实函数组成，其中一个实函数作为虚部。

按DFT的线性性质，$v(n)$的DFT由下式给出：

$$\begin{aligned}
V(k) &= X(k) + jY(k) \\
&= \left[X_R(k) + jX_I(k)\right] + j\left[Y_R(k) + jY_I(k)\right] \\
&= \underline{\left[X_R(k) - Y_I(k)\right]} + j\underline{\left[X_I(k) + Y_R(k)\right]} \\
&= V_R(k) + jV_I(k)
\end{aligned}$$

重写如下 $V_R(k)=\left[X_R(k)-Y_I(k)\right],\quad V_I(k)=\left[X_I(k)+Y_R(k)\right]$

由上式可以看到，求得的$V(k)$的实部和虚部既含有的$X(k)$成分，也有$Y(k)$的成分，可以把$V_R(k)$和$V_I(k)$分解成奇偶序列，即：

$$V(k)=\left\{\left[\underbrace{\frac{V_R(k)+V_R(N-k)}{2}}_{\text{偶序列}}\right]+\left[\underbrace{\frac{V_R(k)-V_R(N-k)}{2}}_{\text{奇序列}}\right]\right\}$$
$$+j\left\{\left[\underbrace{\frac{V_I(k)+V_I(N-k)}{2}}_{\text{偶序列}}\right]+\left[\underbrace{\frac{V_I(k)-V_I(N-k)}{2}}_{\text{奇序列}}\right]\right\}$$

根据DFT性质：

若$w(n)$是实序列，则 $W(k)=\underbrace{W_R(k)}_{\text{偶序列}}+j\underbrace{W_I(k)}_{\text{奇序列}}$

若$w(n)$是纯虚序列，则 $W(k)=\underbrace{W_R(k)}_{\text{奇序列}}+j\underbrace{W_I(k)}_{\text{偶序列}}$

由上述的性质可以得出两个序列$X(k)$和$Y(k)$如下

$$X(k)=\left[\frac{V_{\mathrm{R}}(k)+V_{\mathrm{R}}(N-k)}{2}\right]+j\left[\frac{V_{\mathrm{I}}(k)-V_{\mathrm{I}}(N-k)}{2}\right] \quad \text{(A)}$$

$$jY(k)=\left[\frac{V_{\mathrm{R}}(k)-V_{\mathrm{R}}(N-k)}{2}\right]+j\left[\frac{V_{\mathrm{I}}(k)+V_{\mathrm{I}}(N-k)}{2}\right]$$

或者

$$Y(k)=\left[\frac{V_{\mathrm{I}}(k)+V_{\mathrm{I}}(N-k)}{2}\right]-j\left[\frac{V_{\mathrm{R}}(k)-V_{\mathrm{R}}(N-k)}{2}\right] \quad \text{(B)}$$

做一次N点复序列的DFT，就能同时把两个N点实序列的DFT求出来，因为求得$V(k)=V_{\mathrm{R}}(k)+jV_{\mathrm{I}}(k)$后，按式(A)、(B)即可组合出$X(k)$和$Y(k)$，显然使运算效率提高一倍。

□ 同时计算两个实序列的FFT的步骤

1. 函数$x(n)$和$y(n)$是实序列，$n = 0,1,\cdots,N-1$。

2. 构成复序列：$v(n) = x(n) + jy(n)$，$\quad n = 0,1,\cdots,N-1$

3. 计算

$$V(k) = \sum_{n=0}^{N-1} v(n)e^{-j2\pi nk/N} = V_R(k) + jV_I(k), \quad k = 0,1,\cdots,N-1$$

式中$V_R(k)$和$jV_I(k)$分别为$V(k)$的实部和虚部。

4. 计算

$$\boxed{\begin{cases} X(k) = \frac{1}{2}\left[V_{\mathrm{R}}(k) + V_{\mathrm{R}}(N-k)\right] + j\frac{1}{2}\left[V_{\mathrm{I}}(k) - V_{\mathrm{I}}(N-k)\right] \\ Y(k) = \frac{1}{2}\left[V_{\mathrm{I}}(k) + V_{\mathrm{I}}(N-k)\right] - j\frac{1}{2}\left[V_{\mathrm{R}}(k) - V_{\mathrm{R}}(N-k)\right] \end{cases}} \quad k = 0,1,\cdots,N-1$$

这里$X(k)$和$Y(k)$分别为$x(n)$和$y(n)$的DFT。

□ 用N点变换计算$2N$点实序列的FFT

考虑一个用$2N$个样本点描述的序列$x(n)$，其$2N$点的DFT变换为

$$X(k)=\sum_{n=0}^{2N-1}x(n)W_{2N}^{kn},\quad k=0,1,2,\ldots,2N-1$$

我们希望用N点的DFT来计算序列$x(n)$的DFT

也就是说，我们希望把$2N$点的序列$x(n)$分解为两个N点的序列。但是，序列$x(n)$不能简单地分成两半，而要按n的奇偶分为两组，即

$$\begin{cases}x(2r)=u(r)\\x(2r+1)=v(r)\end{cases},\quad r=0,1,2,\ldots,N-1$$

式中序列$u(r)$等于$x(r)$的偶数号样本点，而$v(r)$等于奇数号样本点（注意，$u(r)$和$v(r)$并不是由$x(n)$分解所得的偶函数和奇函数）

这样就可以把长度为$2N$的实序列$x(n)$分解为两个N点实序列（奇偶两组）计算：

$$X(k)=\sum_{n=0}^{2N-1}x(n)e^{-j2\pi kn/2N}=\sum_{r=0}^{N-1}x(2r)e^{-j2\pi k(2r)/2N}+\sum_{r=0}^{N-1}x(2r+1)e^{-j2\pi k(2r+1)/2N}$$

$$=\sum_{r=0}^{N-1}x(2r)e^{-j2\pi k(r)/N}+e^{-j\pi k/N}\sum_{r=0}^{N-1}x(2r+1)e^{-j2\pi k(r)/N}$$

$$=\sum_{r=0}^{N-1}u(r)e^{-j2\pi k(r)/N}+e^{-j\pi k/N}\sum_{r=0}^{N-1}v(r)e^{-j2\pi k(r)/N}$$

$$=U(k)+e^{-j\pi k/N}V(k)\quad,\quad k=0,1,2,\dots,2N-1$$

其中$U(k)$、$V(k)$是两个长度为N的实序列$u(r)$、$v(r)$的DFT，可以通过前面讨论的“同时计算两个实序列的FFT的方法”实现。

虽然上式中的k是从0到$2N-1$，但由于$U(k+N)=U(k)$、$V(k+N)=V(k)$，只需计算$0\le n\le N-1$的$U(k)$和$V(k)$，又由于$N\le k\le 2N-1$时，有$e^{-j\pi k/N}=-e^{-j\pi(k+N)/N}$，所以当$N\le k\le 2N-1$时，令$k=N+l$，则

$$\boxed{\begin{aligned}X(k)=X(N+l)&=U(l)+e^{-j\pi(l+N)/N}V(l)\\&=U(l)-e^{-j\pi k/N}V(l),\qquad l=0,1,2,\dots,N-1\end{aligned}}$$

5.2 基2FFT算法的理论推导

记号的定义

考虑DFT变换式（$N=2^{\gamma}$）

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk}, \qquad k=0,1,\cdots,N-1 \qquad (5.2A)$$

这里$W_N=e^{-j2\pi/N}$。需要把整数n和k表成二进制数；也就是说，如果假设$N=4$，那么$\gamma=2$，n和k可用两位二进制数表示：

$$k=0,1,2,3 \text{ 或者 } k=(k_1,k_0)=00,01,10,11$$

$$n=0,1,2,3 \text{ 或者 } n=(n_1,n_0)=00,01,10,11$$

把上面的k和n写成如下紧凑形式：

$$\boxed{k=2k_1+k_0, \quad n=2n_1+n_0} \qquad (5.2B)$$

这里k_0,k_1,n_0和n_1只能取值0和1。式 (5.2B) 只是把相应的十进制数k写成二进制数的一种方法。

当$N=4$时，可把式(5.2A)改写为（为方便起见，用W^{nk}替代W_N^{nk}）

$$X(k_1,k_0)=\sum_{n_0=0}^{1}\sum_{n_1=0}^{1}x(n_1,n_0)W^{(2n_1+n_0)(2k_1+k_0)} \tag{5.2C}$$

注意，为了计算二进制表示的n的所有位，公式(5.2A) 中单个求和号现在必须用γ个求和号来代替。

W^p的因子分解

现在研究W^p这一项。由于$W^{a+b}=W^a\cdot W^b$，所以

$$\begin{aligned}W^{(2n_1+n_0)(2k_1+k_0)}&=W^{2n_1(2k_1+k_0)}\cdot W^{n_0(2k_1+k_0)}\\&=[W^{4n_1k_1}]W^{2n_1k_0}W^{n_0(2k_1+k_0)}=W^{2n_1k_0}W^{n_0(2k_1+k_0)}\end{aligned} \tag{5.2D}$$

注意，方括号中的项等于1， 因为

$$W^{4n_1k_1}=[W^4]^{n_1k_1}=\left[e^{-j2\pi4/4}\right]^{n_1k_1}=[1]^{n_1k_1}=1 \tag{5.2E}$$

这样，公式(5. 2C)就可以写成下面的形式

$$X(k_1,k_0)=\sum_{n_0=0}^{1}\left[\sum_{n_1=0}^{1}x(n_1,n_0)W^{2n_1k_0}\right]W^{n_0(2k_1+k_0)} \qquad (5.2F)$$

这个公式代表了FFT算法的基础。为了说明这一点，下面分别研究公式(5.2F)中各个求和式。首先重写上式方括号内的求和式为

$$x_1(k_0,n_0)=\sum_{n_1=0}^{1}x(n_1,n_0)W^{2n_1k_0} \qquad (5.2G)$$

由上式可得

$$\begin{cases}x_1(0,0)=x(0,0)+x(1,0)W^0\\x_1(0,1)=x(0,1)+x(1,1)W^0\\x_1(1,0)=x(0,0)+x(1,0)W^2\\x_1(1,1)=x(0,1)+x(1,1)W^2\end{cases} \qquad (5.2H)$$

把式(5.2H)重写

$$\begin{cases} x_1(0,0) = x(0,0) + x(1,0)W^0 \\ x_1(0,1) = x(0,1) + x(1,1)W^0 \\ x_1(1,0) = x(0,0) + x(1,0)W^2 \\ x_1(1,1) = x(0,1) + x(1,1)W^2 \end{cases} \qquad (5.2H)$$

如果把式(5.2H)重写成矩阵形式，有

$$\begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0,0) \\ x(0,1) \\ x(1,0) \\ x(1,1) \end{bmatrix} \qquad (5.2I)$$

注意到上式正好是5.1节中导出的矩阵方程(5.1F)

$$\begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad (5.1F)$$

但式(5.2I)中的n是用二进制表示的。

因此，式(5.2F)方括号

$$X(k_1,k_0)=\sum_{n_0=0}^{1}\left[\sum_{n_1=0}^{1}x(n_1,n_0)W^{2n_1k_0}\right]W^{n_0(2k_1+k_0)} \quad (5.2\mathrm{F})$$

里边的求和式代表前面5.1节直观推导DFT计算次数时的式(5.1D)的第一个矩阵因子

$$\begin{bmatrix}X(0)\\X(2)\\X(1)\\X(3)\end{bmatrix}=\begin{bmatrix}1 & W^0 & 0 & 0\\1 & W^2 & 0 & 0\\0 & 0 & 1 & W^1\\0 & 0 & 1 & W^3\end{bmatrix}\begin{bmatrix}1 & 0 & W^0 & 0\\0 & 1 & 0 & W^0\\1 & 0 & W^2 & 0\\0 & 1 & 0 & W^2\end{bmatrix}\begin{bmatrix}x(0)\\x(1)\\x(2)\\x(3)\end{bmatrix} \quad (5.1\mathrm{D})$$

同样，如果把式(5.2F)

$$X(k_1,k_0)=\sum_{n_0=0}^{1}\left[\sum_{n_1=0}^{1}x(n_1,n_0)W^{2n_1k_0}\right]W^{n_0(2k_1+k_0)} \qquad (5.2\text{F})$$

的外层的求和写成

$$x_2(k_0,k_1)=\sum_{n_0=0}^{1}x_1(k_0,n_0)W^{n_0(2k_1+k_0)} \qquad (5.2\text{J})$$

进一步写成矩阵形式，得到

$$\begin{bmatrix}x_2(0,0)\\x_2(0,1)\\x_2(1,0)\\x_2(1,1)\end{bmatrix}=\begin{bmatrix}1 & W^0 & 0 & 0\\1 & W^2 & 0 & 0\\0 & 0 & 1 & W^1\\0 & 0 & 1 & W^3\end{bmatrix}\begin{bmatrix}x_1(0,0)\\x_1(0,1)\\x_1(1,0)\\x_1(1,1)\end{bmatrix} \qquad (5.2\text{K})$$

将式(5. 2K)重写

$$\begin{bmatrix} x_2(0,0) \\ x_2(0,1) \\ x_2(1,0) \\ x_2(1,1) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} \quad (5.2K)$$

这就是矩阵方程(5.1F)。所以式(5.2F)外层的求和定义了前面5.1节直观推导DFT计算次数时的式(5.1D)的第二个矩阵因子

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \quad (5.1D)$$

从公式（5.2F）和（5.2J），有(见下页)

从公式(5.2F)和(5.2J)，有

$$X(k_1,k_0)=\sum_{n_0=0}^{1}\left[\sum_{n_1=0}^{1}x(n_1,n_0)W^{2n_1k_0}\right]W^{n_0(2k_1+k_0)} \tag{5.2F}$$

$$x_2(k_0,k_1)=\sum_{n_0=0}^{1}x_1(k_0,n_0)W^{n_0(2k_1+k_0)} \tag{5.2J}$$

于是

$$X(k_1,k_0)=x_2(k_0,k_1) \tag{5.2L}$$

也就是说，由外层求和得到的最后结果$x_2(k_0,k_1)$与我们所要求的$X(k_1,k_0)$，它们的位序是倒序的。

如果把公式(5.2G)、(5.2J)和(5.2L)相联立，即

$$
\begin{cases}
x_1(k_0,n_0)=\sum_{n_1=0}^{1} x(n_1,n_0)W^{2n_1k_0} \\
x_2(k_0,k_1)=\sum_{n_0=0}^{1} x_1(k_0,n_0)W^{n_0(2k_1+k_0)} \\
X(k_1,k_0)=x_2(k_0,k_1)
\end{cases}
\tag{5.2M}
$$

上面方程组(5.2M)便是库利−图基最初为$N=4$列出的FFT算法。在这一组方程中，第二个方程是由第一个方程计算的，所以称它们是递归的。

5.3 按时间抽取的FFT算法

■ 算法原理（基2-FFT）

先将$x(n)$按 n 的奇偶分为两组，设N=2^{M}，不足时补零，n 用变量$2r$ 表示，即

n 为偶数： $x(2r)=x_1(r),\qquad r=0,1,\cdots,\frac{N}{2}-1$

n 为奇数： $x(2r+1)=x_2(r),\qquad r=0,1,\cdots,\frac{N}{2}-1$

因此有

$$X(k)=\mathrm{DFT}[x(n)]=\sum_{n=0}^{N-1}x(n)W_N^{nk}$$

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

（n为偶数） （n为奇数）

$$=\sum_{r=0}^{\frac{N}{2}-1}x(2r)W_N^{2rk}+\sum_{r=0}^{\frac{N}{2}-1}x(2r+1)W_N^{(2r+1)k}$$

$$=\sum_{r=0}^{\frac{N}{2}-1}x_1(r)(W_N^2)^{rk}+W_N^k\sum_{r=0}^{\frac{N}{2}-1}x_2(r)(W_N^2)^{rk}$$

由于$W_N^2 = e^{-j\frac{2\pi}{N}2} = e^{-j2\pi/(\frac{N}{2})} = W_{N/2}$，上式可表示为：

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_{\frac{N}{2}}^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x_2(r) W_{\frac{N}{2}}^{rk} = X_1(k) + W_N^k X_2(k)$$

式中 $k = 0, \cdots, \frac{N}{2} - 1$

其中

$$\begin{cases} X_1(k) = \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_{\frac{N}{2}}^{rk} = \mathrm{DFT}\left(x_1(r)\right) \\ X_2(k) = \sum_{r=0}^{\frac{N}{2}-1} x_2(r) W_{\frac{N}{2}}^{rk} = \mathrm{DFT}\left(x_2(r)\right) \end{cases}$$

- **结论**

1、$X_1(k)$，$X_2(k)$均为$N/2$点的DFT；

2、$X(k)=X_1(k)+ W_N^k X_2(k)$只能确定出$X(k)$的$k=0,1,\cdots,\frac{N}{2}-1$的值，即$X(k)$前一半的结果。

■ X(k)后一半的计算

根据W_N^{nk}的周期性有 $W_{N/2}^{r(k+N/2)}=W_{N/2}^{rk}$，所以有：

$$X_1(\frac{N}{2}+k)=\sum_{r=0}^{\frac{N}{2}-1}x_1(r)W_{\frac{N}{2}}^{r(\frac{N}{2}+k)}=\sum_{r=0}^{\frac{N}{2}-1}x_1(r)W_{\frac{N}{2}}^{rk}=X_1(k)$$

同理有 $X_2\left(\frac{N}{2}+k\right)=X_2(k)$ $\quad k=0,\cdots,\frac{N}{2}-1$

这就是说，$X_1(k)$、$X_2(k)$的后一半，分别等于其前一半的值

又由于 $W_N^{(N/2+k)}=W_N^{N/2}W_N^k=-W_N^k$，且$X_1(k)$, $X_2(k)$以$N/2$为周期，故有：

$$X(k+\frac{N}{2})=X_1(k+\frac{N}{2})+W_N^{k+\frac{N}{2}}X_2(k+\frac{N}{2})=X_1(k)-W_N^kX_2(k)\ , \quad k=0,1,\cdots,\frac{N}{2}-1$$

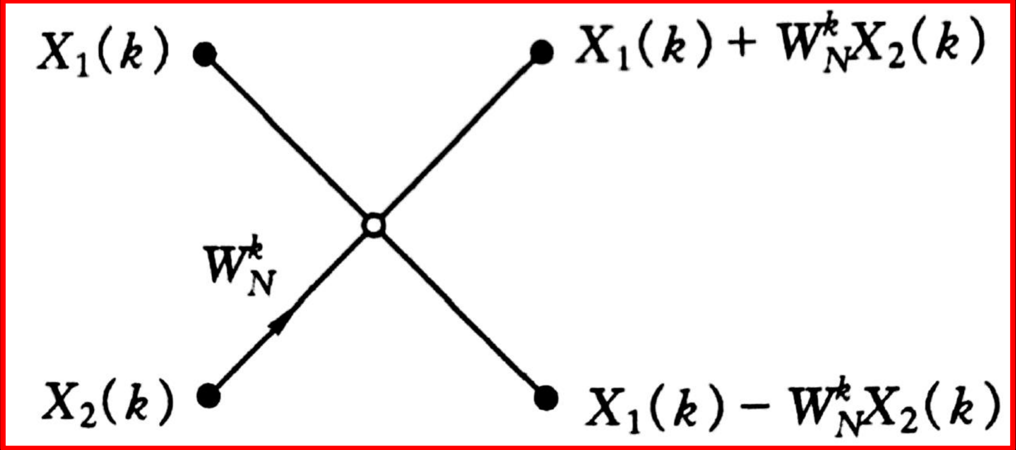
可见，$X(k)$的后一半，也完全由$X_1(k)$、$X_2(k)$ 所确定。所以，N点的DFT可由两个$N/2$点的DFT来计算。

■ 蝶形运算

由$X_1(k)$、$X_2(k)$来表示$X(k)$的运算如下

$$\begin{cases} X(k) = X_1(k) + W_N^k X_2(k) \\ X(k+\dfrac{N}{2}) = X_1(k) - W_N^k X_2(k) \end{cases} \quad (k = 0, 1, \cdots, \tfrac{N}{2} - 1)$$

实现上式运算的流图称作**蝶形运算**



举例

N=8序列的DFT，可以分解为两个$N/2$=4点的DFT，具体方法如下：

1、n为偶数时的序列$x_1(n)$为

$$x_1(0)=x(0),\ x_1(1)=x(2),\ x_1(2)=x(4), x_1(3)=x(6)$$

进行$N/2$=4点的DFT得$X_1(k)$

$$X_1(k)=\sum_{r=0}^{3}x_1(r)W_4^{rk}=\sum_{r=0}^{3}x(2r)W_4^{rk}, k=0,1,2,3$$

2、n为奇数时的序列$x_2(n)$为

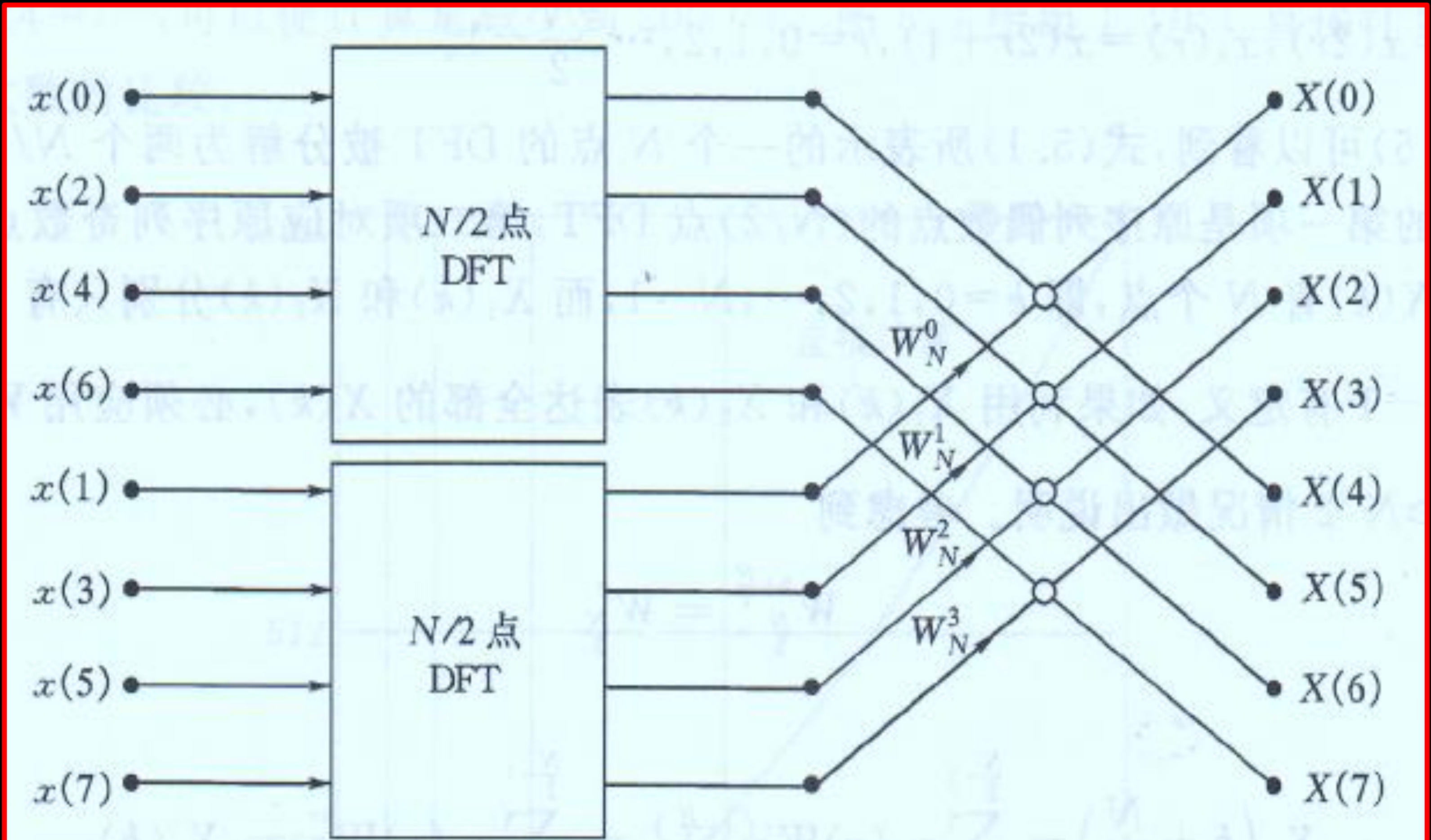
$$x_2(0)=x(1),\ x_2(1)=x(3),\ x_2(2)=x(5), x_2(3)=x(7)$$

进行$N/2$=4点的DFT得$X_2(k)$

$$X_2(k)=\sum_{r=0}^{3}x_2(r)W_4^{rk}=\sum_{r=0}^{3}x(2r+1)W_4^{rk}, k=0,1,2,3$$

根据蝶形运算，即可由4点$X_1(k)$和4点$X_2(k)$来计算全部N=8 点$X(k)$

3、对$X_1(k)$和$X_2(k)$进行蝶形运算，前半部为$X(0)$到$X(3)$，后半部分为$X(4)$到$X(7)$，整个过程如下图所示：



■ 迭代奇偶分组

按照奇偶分组的基本思想，不妨对每个$N/2$的序列进一步奇偶分组，从而进一步减少运算量。假设序列总长$N=2^L$,就可以这样分解L层。当不满足时，可以补零操作。

1、对前面的偶序列$x_1(r),\ r=0,\cdots,\frac{N}{2}-1$进一步奇偶数分解，即

$$x_1(2l)=x_3(l),\ l=0,\cdots,\tfrac{N}{4}-1$$

$$x_1(2l+1)=x_4(l),\ l=0,\cdots,\tfrac{N}{4}-1$$

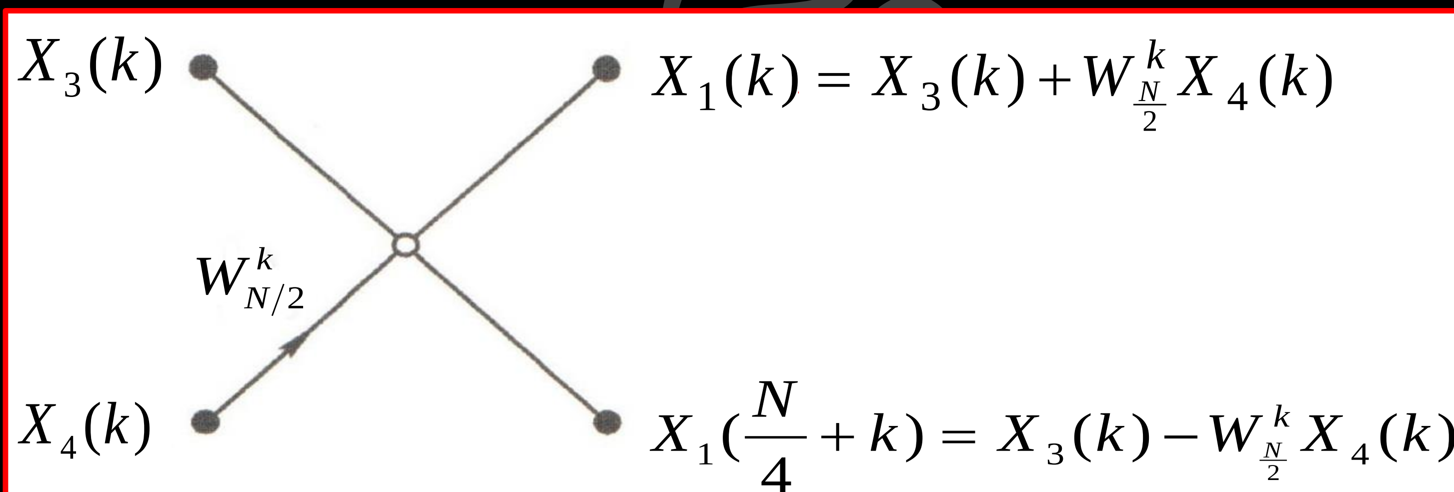
分别进行$N/4$点的DFT，得到：

$$\begin{cases} X_3(k)=\sum_{l=0}^{\frac{N}{4}-1}x_3(l)W_{N/4}^{lk},\quad k=0,\cdots,\dfrac{N}{4}-1 & \text{(偶数序列中再取偶)} \\ X_4(k)=\sum_{l=0}^{\frac{N}{4}-1}x_4(l)W_{N/4}^{lk},\quad k=0,\cdots,\dfrac{N}{4}-1 & \text{(偶数序列中再取奇)} \end{cases}$$

利用蝶形运算，可用$N/4$点的$X_3(k)$和$X_4(k)$计算$N/2$点的$X_1(k)$，即

$$\begin{cases} X_1(k) = X_3(k) + W_{\frac{N}{2}}^{k} X_4(k) \\ X_1(\frac{N}{4}+k) = X_3(k) - W_{\frac{N}{2}}^{k} X_4(k) \end{cases} \qquad k = 0,1,\ldots,\frac{N}{4}-1$$

其蝶形运算流图如下



2、同样对前面的奇序列$x_2(r)$，$r=0,\cdots,\frac{N}{2}-1$进一步奇偶数分解为$N/4$的序列$x_5(l)$和$x_6(l)$，并分别进行$N/4$点的DFT如下

$$\begin{cases} X_5(k)=\sum_{l=0}^{N/4-1} x_2(2l)W_{N/4}^{lk}=\sum_{l=0}^{N/4-1} x_5(l)W_{N/4}^{lk} & \text{(奇数序列中再取偶序号子序列)} \\ X_6(k)=\sum_{l=0}^{N/4-1} x_2(2l+1)W_{N/4}^{lk}=\sum_{l=0}^{N/4-1} x_6(l)W_{N/4}^{lk} & \text{(奇数序列中再取奇序号子序列)} \end{cases}$$

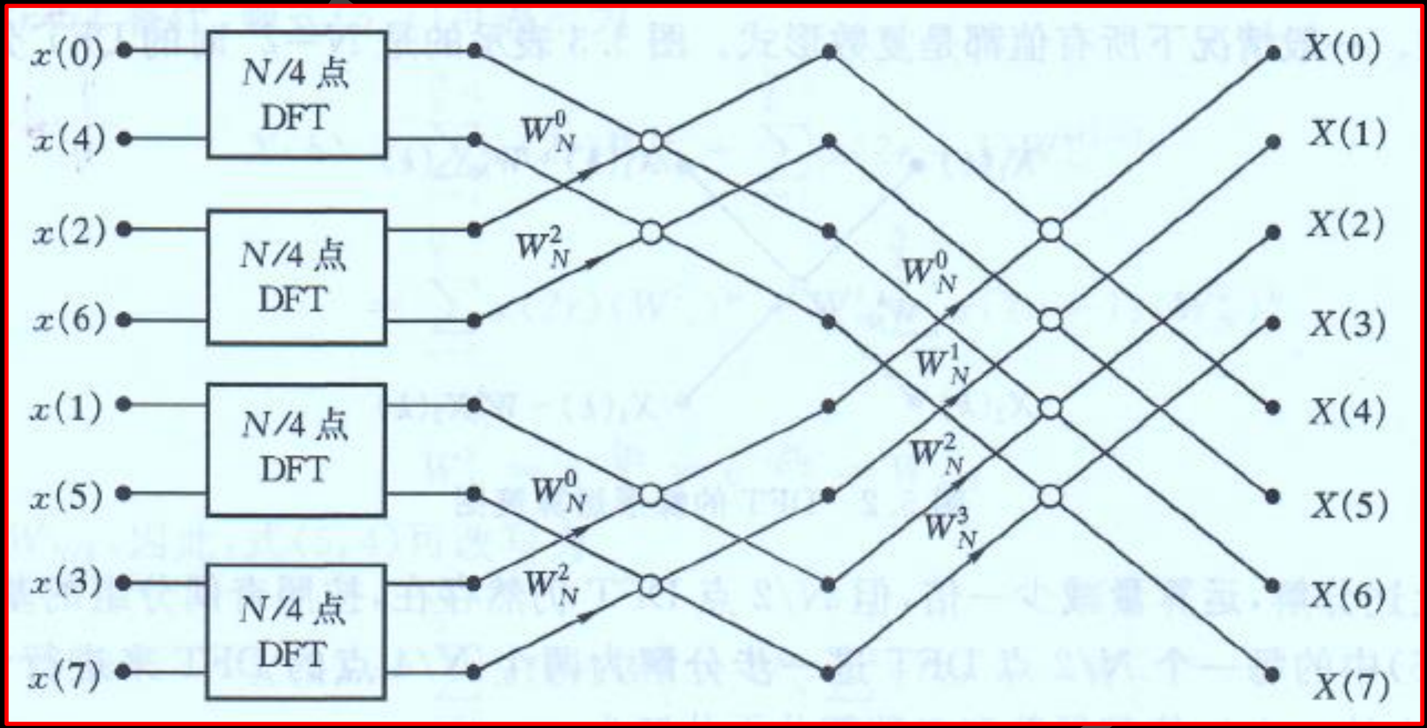
由$X_5(k)$、$X_6(k)$进行蝶形运算计算$N/2$点的$X_2(k)$，得到

$$\begin{cases} X_2(k)=X_5(k)+W_{N/2}^{k}X_6(k)\ ;k=0,1,\ldots,\frac{N}{4}-1 \\ X_2(\frac{N}{4}+k)=X_5(k)-W_{N/2}^{k}X_6(k)\ ;k=0,1,\cdots,\frac{N}{4}-1 \end{cases}$$

所以，N点的DFT分解为四个$N/4$点的DFT来计算。

3、两级蝶形计算流图如下

(为减少参数个数，设 $W_{N/2}^{0}=W_{N}^{0}, W_{N/2}^{1}=W_{N}^{2}$)



经过上述两级蝶形运算，用四个$N/4$点DFT计算N点序列的DFT，其运算量可再减少约一半，即为**N点DFT计算量的1/4**

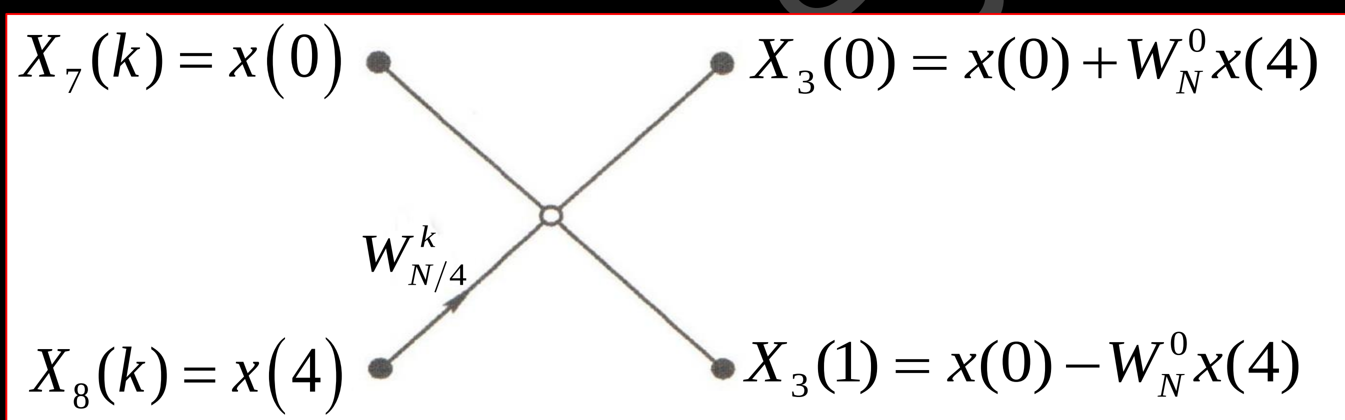
4、最后一级——2点DFT的蝶形运算

对于$N=2^3=8$时DFT，经过2级迭代，即为$N/4$点即为两点DFT。

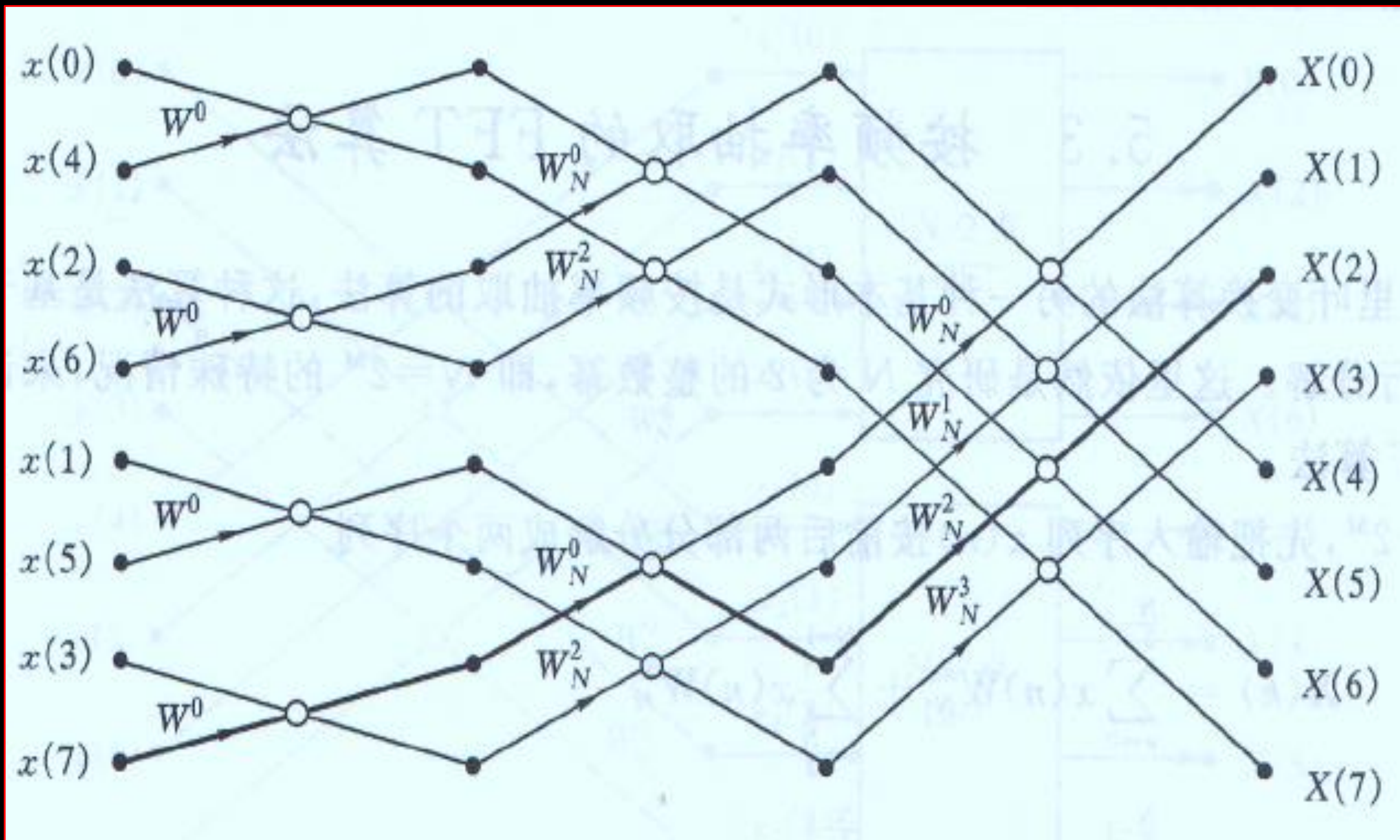
以 $X_3(k)=\sum_{l=0}^{1} x_3(l)W_2^{lk}$，$k=0,1$ 为例，即

$$\begin{cases} X_3(0)=x_7(0)+W_{N/4}^{0}x_8(0)=x(0)+W_N^0 x(4) \\ X_3(1)=x_7(0)-W_{N/4}^{0}x_8(0)=x(0)-W_N^0 x(4) \end{cases}$$

2点DFT仍可以用蝶形运算分解为单点DFT，进一步减少运算量，即



5、最终，得到$N=2^M=8$点DFT分解为$M=3$级迭代的蝶形运算流图如下：

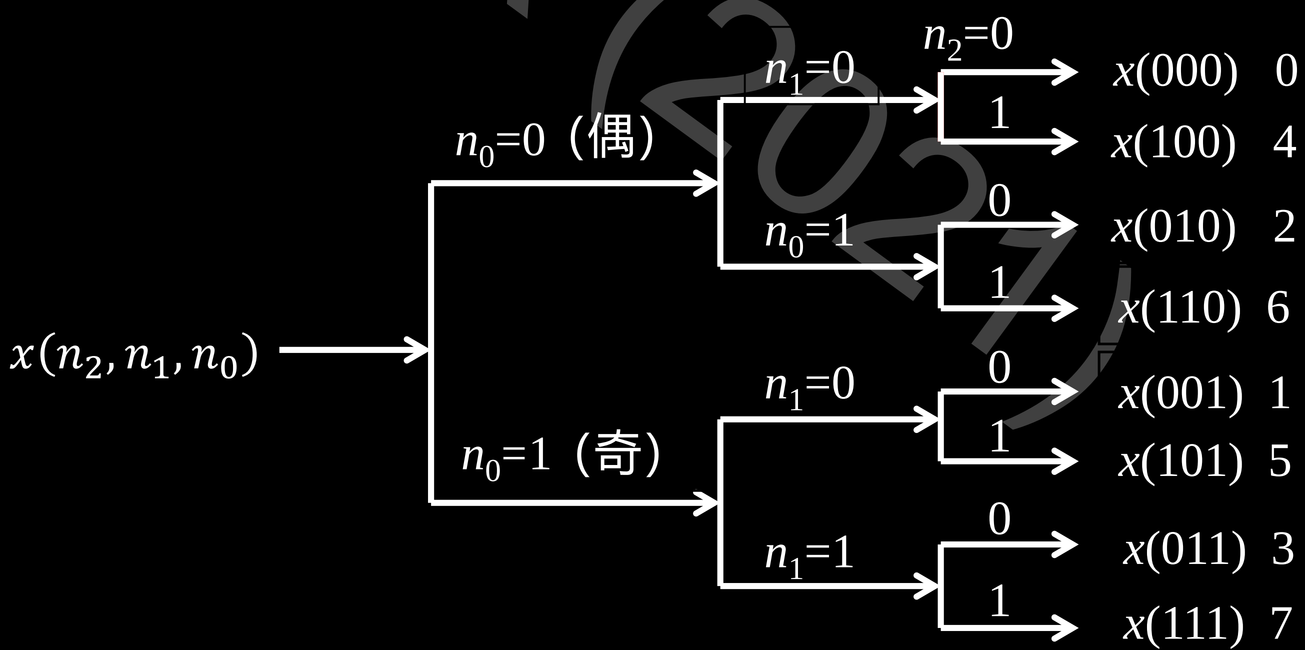


■ 倒位序的实现

由上图可知，按时域抽取FFT的输出$X(k)$按自然顺序排列在存储单元，而输入是按以下顺序排列在存储单元：

$$x(0),x(4),x(2),x(6);\ x(1),x(5),x(3),x(7)$$

这种顺序称作**倒位序**，它是由奇偶分组造成的。



■ 码位倒置与自然序号的关系

对$N=8=2^3$的输入序列，序号二进制为$(n_2, n_1, n_0)_2$，其倒位序二进制为$(n_0, n_1, n_2)_2$。

码位倒置与自然序号的关系

自然顺序	二进制$n_2n_1n_0$	倒位序二进制$n_0n_1n_2$	倒位顺序
0	0 0 0	0 0 0	0
1	0 0 1	1 0 0	4
2	0 1 0	0 1 0	2
3	0 1 1	1 1 0	6
4	1 0 0	0 0 1	1
5	1 0 1	1 0 1	5
6	1 1 0	0 1 1	3
7	1 1 1	1 1 1	7

5.4 按频率抽取的FFT算法

■ 算法原理（基2FFT）

1、把$N=2^M$的输入序列$x(n)$按前后两部分分解为2个$N/2$长的短序列， $x(n)$的DFT可以计算为

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

$$=\underline{\sum_{n=0}^{N/2-1}x(n)W_N^{nk}+\sum_{n=N/2}^{N-1}x(n)W_N^{nk}}$$

前后分组

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

$$=\sum_{n=0}^{N/2-1}\left[x(n)+x(n+\frac{N}{2})W_N^{(N/2)k}\right]W_N^{nk}$$

由于 $W_N^{\frac{N}{2}k} = \left(W_N^{\frac{N}{2}}\right)^k = (-1)^k$

因此 $X(k) = \sum_{n=0}^{N/2-1}\left[x(n) + (-1)^k x(n+\frac{N}{2})\right]W_N^{nk} \quad k = 0,1,\ldots,N-1$

2、将N点DFT输出序列$X(k)$按k 的奇偶分组分为两个$N/2$点DFT

当k为偶数，即 $k=2r$ 时，有 $(-1)^k = 1$

当k为奇数，即 $k=2r+1$ 时，有 $(-1)^k = -1$

这样，$X(k)$ 可分为两部分：

k为偶数时

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

$$= \sum_{n=0}^{N/2-1}\underline{\left[x(n) + x(n+\frac{N}{2})\right]W_{N/2}^{nr}} \qquad 0 \le r \le \frac{N}{2} - 1$$

k为奇数时

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

$$= \sum_{n=0}^{N/2-1}\underline{\left\{\left[x(n) - x(n+\frac{N}{2})\right]W_N^{n}\right\}W_{N/2}^{nr}}, \quad 0 \le r \le \frac{N}{2}\text{-}1$$

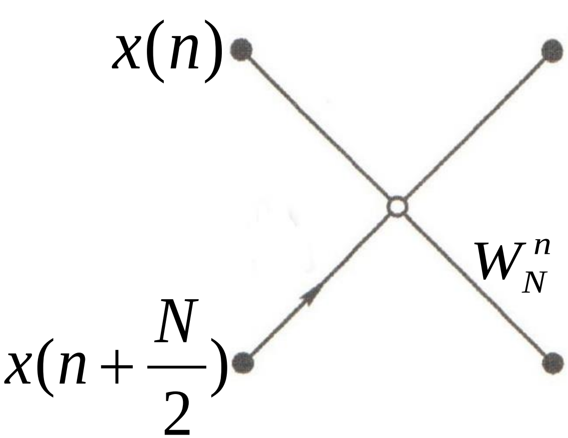
上面两式均满足$N/2$点**DFT**的定义式

3、蝶形运算

$$\begin{cases} X(2r)=\displaystyle\sum_{n=0}^{\frac{N}{2}-1}\underbrace{\left[x(n)+x\left(n+\frac{N}{2}\right)\right]}_{x_1(n)}W_{\frac{N}{2}}^{nr} \\ X(2r+1)=\displaystyle\sum_{n=0}^{\frac{N}{2}-1}\underbrace{\left\{\left[x(n)-x\left(n+\frac{N}{2}\right)\right]W_N^n\right\}}_{x_2(n)}W_{\frac{N}{2}}^{nr} \end{cases}$$

$$0 \le n \le \frac{N}{2}-1$$

对$x(n)$和$x(n+\frac{N}{2})$进行如下蝶形运算



$$x(n)+x\left(n+\frac{N}{2}\right)=x_1(n)$$

先形成了两个子序列

$$\left[x(n)-x\left(n+\frac{N}{2}\right)\right]W_N^n=x_2(n)$$

4、N=8时的计算流图（先形成子序列，得到两个4点的DFT运算）

$$X(2r)=\sum_{n=0}^{\frac{N}{2}-1}\left[x(n)+x(n+\frac{N}{2})\right]W_{\frac{N}{2}}^{nr}$$

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

$$x_1(n)=x(n)+x(n+\frac{N}{2})$$

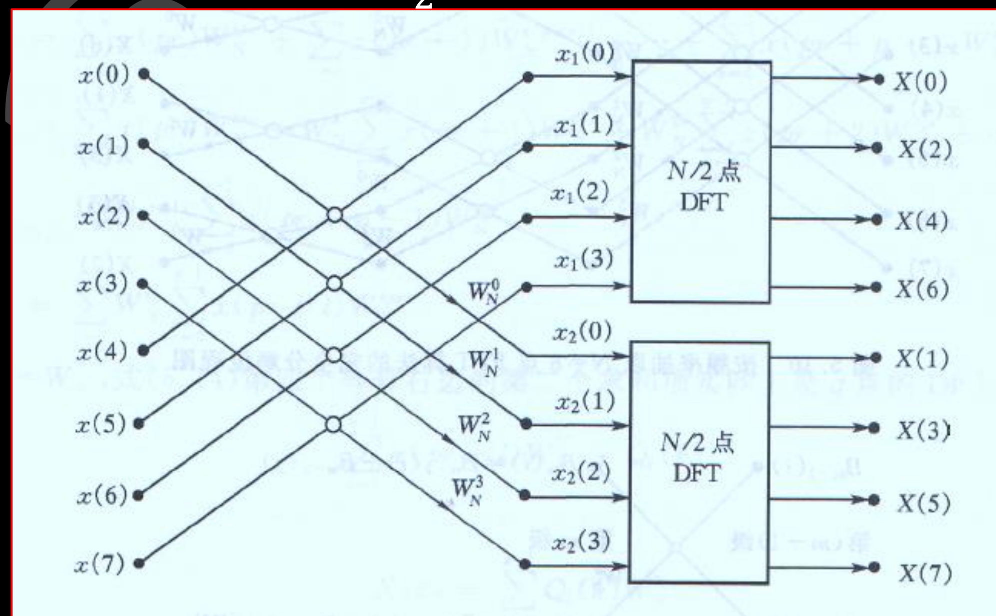
$$x_2(n)=[x(n)-x\left(n+\frac{N}{2}\right)]\,W_N^n$$

子序列 $0\le n\le\frac{N}{2}-1$

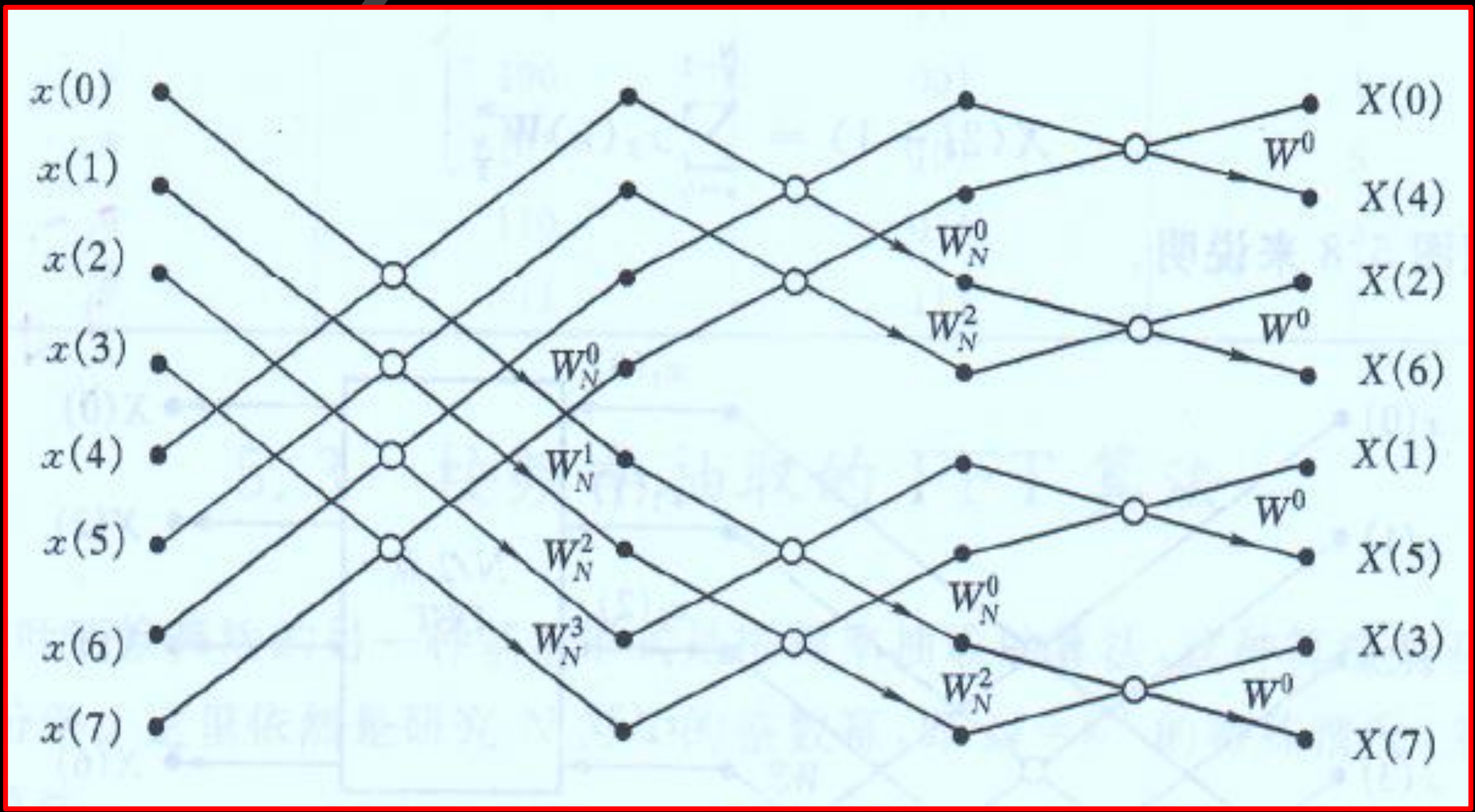
因为 $W_N^2=W_{\frac{N}{2}}$， 于是，由上面两组公式得到两个$\frac{N}{2}$点的DFT运算，即

$$X(2r)=\sum_{n=0}^{\frac{N}{2}-1}x_1(n)\,W_{\frac{N}{2}}^{rn}$$

$$X(2r+1)=\sum_{n=0}^{\frac{N}{2}-1}x_2(n)W_{\frac{N}{2}}^{rn}$$



5、按照时域抽取FFT的思路，再将N/2点DFT按k的奇偶进一步分解为两个N/4点的DFT，如此迭代进行下去，直至分解为单点DFT。得到N=8时按频域抽取FFT的完整计算流图：



- 权函数W_N^p的确定

设迭代蝶形运算的次数为M ($1 \le l \le M$)次，计算序列的长度为$N= 2^M$，l表示计算阵列的第l列。

权函数W_N^p的计算主要是确定p值，p值计算的一种方法如下：

1、把p值表示成M位的二进制数$(p)_2$，其中M应满足：$M=\log_2 N$，N为处理点数；

2、将$(p)_2$右移$(M-l)$位，并把左边的空位补零，结果依然为M位；

3、将移位补零的M位二进制数进行比特倒置；

4、倒置后的二进制数转换成十进制数即得到p值。

- 一节点的权函数是W_N^p，其对偶节点的权函数必然为$W_N^{p+N/2}$，而且$W_N^p = -W_N^{p+N/2}$，所以对偶节点可按下式计算

$$\begin{cases} x_l(k) = x_{l-1}(k) + W_N^p x_{l-1}(k + N/2^l) \\ x_l(k + N/2^l) = x_{l-1}(k) - W_N^p x_{l-1}(k + N/2^l) \end{cases}$$

■ 两种FFT的主要异同

1、不同点

倒位序不同：按时域抽取FFT的输入为倒位序，输出为自然顺序；按频域抽取FFT的输入为自然顺序，输出为倒位序。

蝶形运算形式不同。

2、相同点

运算量相同，均为$(N/2)\text{Log}_2 N$次复乘，$N\log_2 N$次复加；两种FFT形式上具有左右（输入、输出）对称性。

5.5 IDFT的快速运算方法

■ 算法原理

$$X(k) = \mathrm{DFT}[x(n)] = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

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

比较两式可知，只要**DFT**的每个系数W_N^{nk}换成W_N^{-nk}，最后再乘以常数**1/*N***就可以得到**IDFT**的快速算法——**IFFT**。

可以将常数**1/*N***分配到每一级蝶形运算中， $1/N=(1/2)^L$，即每级蝶形运算均乘以1/2。

■ 不改FFT程序直接实现IFFT

由于 $[W_N^{-nk}]^* = W_N^{nk}$, $[A \cdot B]^* = A^* \cdot B^*$

对IDFT取共轭，得到

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

因此有

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

算法步骤：

1、先对$X(k)$取共轭，即将$X(k)$的虚部乘-1；

2、直接利用FFT程序计算其DFT；

3、对计算结果再取一次共轭；

4、最后再乘以常数$1/N$，即得$x(n)$。

从而，FFT、IFFT可共用同一个子程序。

5.6 利用FFT计算线性卷积和线性相关

■ 利用FFT计算线性卷积

$$\boxed{y(n)=x(n)*h(n)=\sum_{m=0}^{L-1}x(m)h(n-m)}$$

其中，$x(n)$长度为L，$h(n)$长度为M， $L\geq M$。

步骤：

1、分别对$x(n)$、$h(n)$补零点至长度至少为$N=M+L-1$点

2、用FFT求 $H(k)=\mathrm{FFT}[h(n)]$

3、用FFT求 $X(k)=\mathrm{FFT}[x(n)]$

4、求 $Y(k)=X(k)H(k)$

5、用IFFT求 $y(n)=\mathrm{IFFT}[Y(k)]$

■ 利用FFT计算线性相关

$$r_{xy}(m)=\sum_{n=0}^{L-1}x(n+m)y^*(n)=\sum_{n=0}^{L-1}x(n)y^*(n-m)$$
$$=x(m)*y^*(-m)$$ **(把线性相关转换到卷积表示)**

其中，$x(n)$长度为L，$y(n)$长度为M， $L\geq M$。

步骤：

1、将$x(n)$、$y(n)$补零点至长度至少为$N=M+L-1$点

2、用FFT求 $X(k)$=FFT[$x(n)$]

3、用FFT求 $Y(k)$=FFT[$y(n)$]

4、求 $R_{xy}(k)=X(k)H(k)^*$

5、用IFFT求 $r_{xy}(n)$=IFFT[$R_{xy}(k)$]

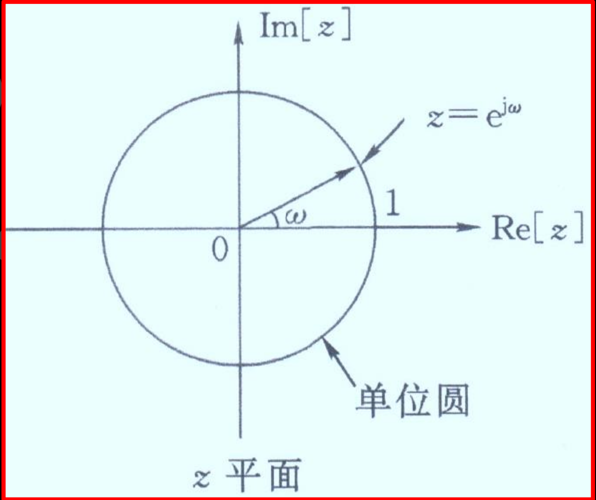
5.7 Chirp-z变换

■ 回顾：DFT与标准z变换的关系

z变换 $$Z[x(n)] = X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}$$

DFT $$X(k) = X(z)\big|_{z=e^{j\frac{2\pi}{N}k}=W_N^{-k}} = X\left(e^{j\frac{2\pi}{N}k}\right) = \sum_{n=0}^{N-1} x(n)W_N^{kn}$$

在z变换中，$z = re^{j\omega}$，取单位圆 r=1，对单位圆进行等间隔采样 $\omega = \frac{2\pi}{N}k$，并取结果的主值区间 k=[0,N-1]，得$z = e^{j\frac{2\pi}{N}k} = W_N^{-k}$，即得DFT。



在实际中，有时对一个时间序列的某个频率分段感兴趣，就会要求这个分段的采样频率与其它分段不一样。

■ Chirp-z变换的定义

$$Z[x(n)] = X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$$

在z变换中，使z沿一段螺线作等角采样，即采样点：

$$z_k = AW^{-k}, \qquad k = 0, \cdots, M-1$$

其中：

参数含义：

A_0——起始采样点的矢量半径

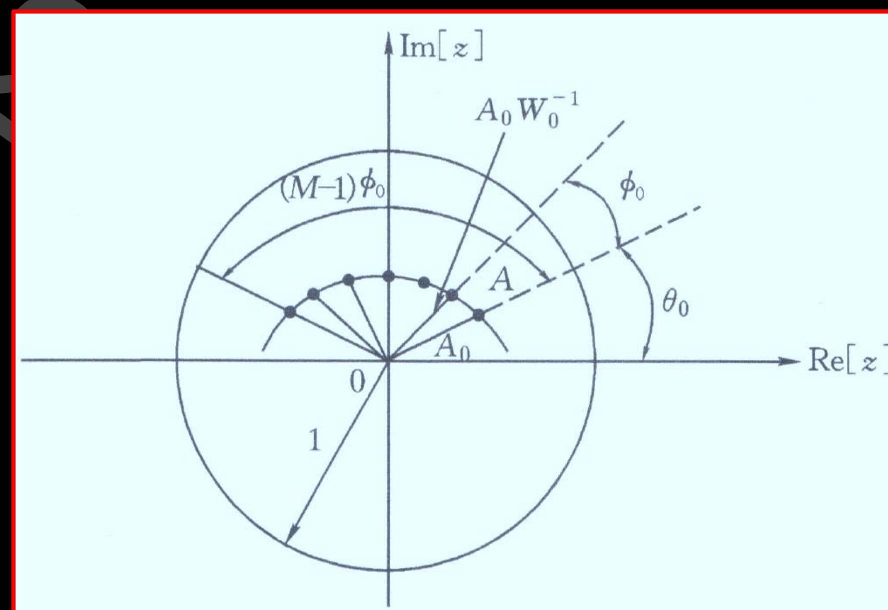
θ_0——起始采样点的相角

ϕ_0——两相邻采样点间的角度差

W_0——螺线的伸展率，

$W_0 > 1$时，螺线内缩；

$W_0 < 1$时，螺线外伸。



对Chirp-z变换 $z_k = AW^{-k}$， $k = 0, \cdots, M-1$，可以看出当取$A = 1$, $W = e^{-j\frac{2\pi}{N}}$时有$z_k = e^{j\frac{2\pi}{N}k}$, $k = 0, \cdots, M-1$，Chirp-z变换即退化成为DFT。

■ **Chirp-z变换的优势**

1、可以只计算单位圆上感兴趣的一小段频谱的采样，而非整个单位圆，这特别适用于高分辨率窄带信号；

2、可以计算远离单位圆的任意点处的频谱，特别适用于语音及雷达信号。

■ Chirp-z变换的快速算法

CZT的定义

$$X(z_k)=\sum_{n=0}^{N-1}x(n)z_k^{-n}=\sum_{n=0}^{N-1}x(n)A^{-n}W^{nk},$$

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

代入

Bluestein等式

$$kn=\frac{1}{2}\left[n^2+k^2-(k-n)^2\right]$$

得到

$$X(z_k)=W^{\frac{k^2}{2}}\sum_{n=0}^{N-1}\left[x(n)A^{-n}W^{\frac{n^2}{2}}\right]W^{-\frac{(k-n)^2}{2}}$$

令

$$g(n)=x(n)A^{-n}W^{\frac{n^2}{2}}$$

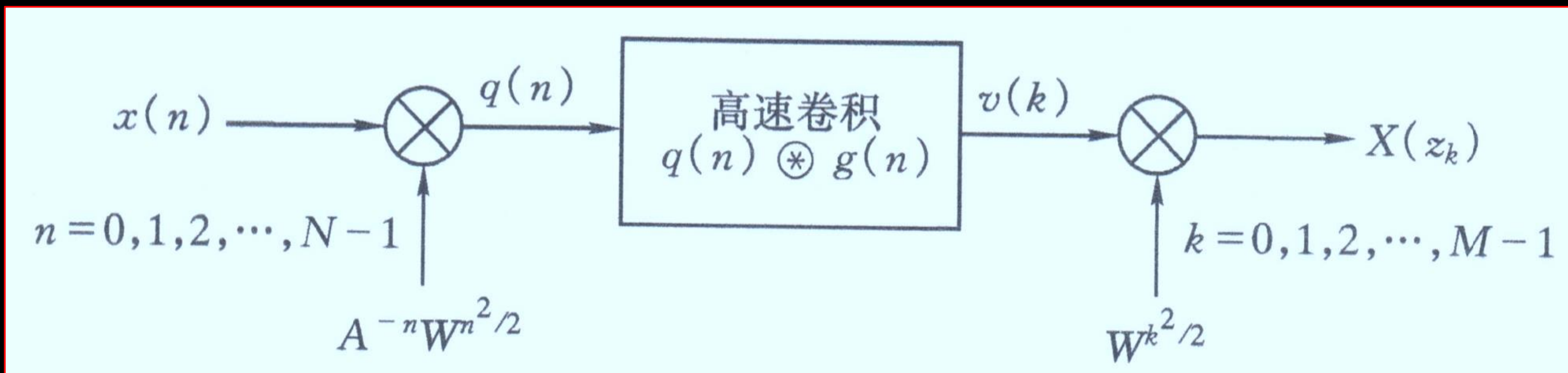
$$h(n)=W^{-\frac{n^2}{2}}$$

得到

$$X(z_k)=W^{\frac{k^2}{2}}\sum_{n=0}^{N-1}g(n)h(k-n)=W^{\frac{k^2}{2}}[g(k)*h(k)]$$

用FFT求解线性卷积即可实现CZT的快速计算

基于卷积计算CZT变换的运算过程



卷积的长度$L \geq N+M-1$，若取$L=N+M-1$，则有

$$q(n)=\begin{cases} x(n)A^{-n}W^{n^2/2}, & n=0,1,2,\ldots,N-1 \\ 0, & n=N,N+1,\ldots,L-1 \end{cases}$$

$$g(n)=\begin{cases} W^{-n^2/2}, & 0 \leq n \leq M-1 \\ W^{-(L-n)^2/2}, & L-N+1 \leq n < L \\ \text{任意值}, & \text{其他}n \end{cases}$$

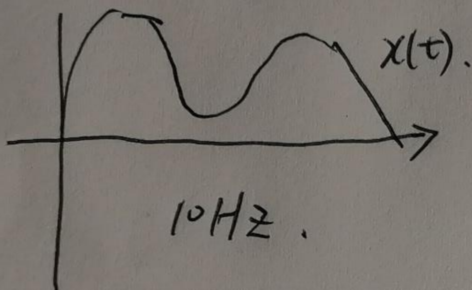
本章小结

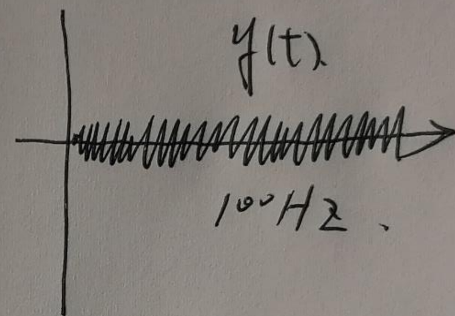
- FFT算法的基本原理
- 基2 FFT算法的理论推导
- 按时间抽取的FFT算法
- 按频率抽取的FFT算法
- IDFT的快速运算方法
- FFT在线性卷积、相关中的应用

调制. 解调. $x(t)$. 低频 10HZ. $y(t)$. 高频 100HZ.

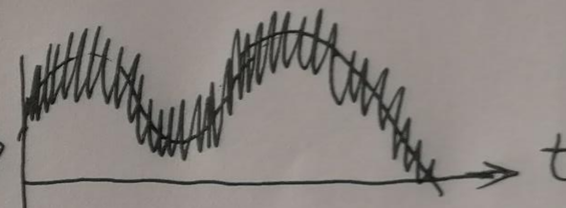
Case 1. $x(t)+y(t) \Rightarrow X(e^{jw})+Y(e^{jw})$.

时域.



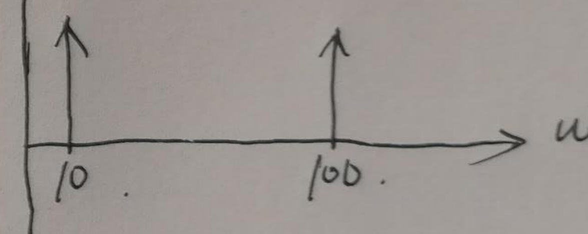


$\Rightarrow$



频域.

$\delta(\omega-100)$.



Case 2. $x(t)\cdot y(t) \Rightarrow X(e^{jw}) * Y(e^{jw})$.

$\Rightarrow$