北京理工大學

第三讲 离散Fourier变换与快速算法

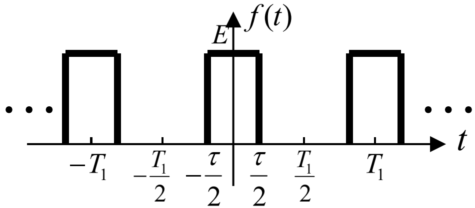
李炳照

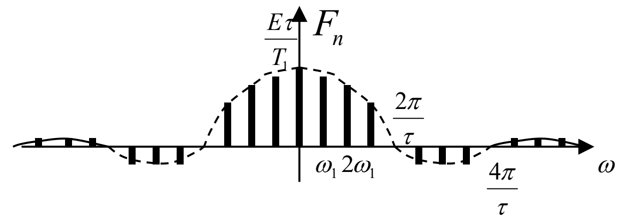
li_bingzhao@bit.edu.cn

2023-2024-2 学期 研究生课程 本硕博贯通课程

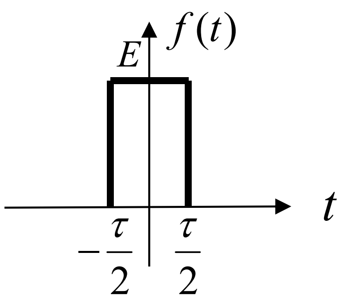
Fourier变换离散算法

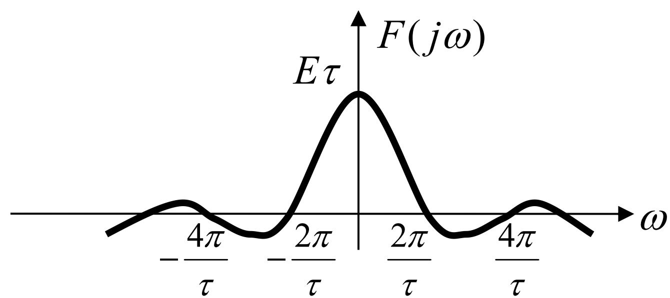
傅里叶级数：任何连续周期信号都可以由成谐波关系的正弦信号的加权和来表示。





傅里叶变换：任何连续非周期信号都可以由不全成谐波关系的正弦信号的加权积分来表示。





信号频谱的特点

- （1）周期信号的频频离散性 -------- 频谱是离散的而不是连续的，这种频谱称为离散频谱。

  非周期信号的频谱连续性。

- （2）周期信号的频谱谐波性 ----- 谱线出现在基波频率ω_1的整数倍上。非周期信号的频谱连续性。

- （3）频谱的收敛性 -------- 幅度谱的幅度随着 $n\to\infty$ 而逐渐衰减到零。

- （4）频谱图、相谱图。

从Fourier级数到变换

性质1（导数性质）$F\{f'(x)\}=i\omega F(\omega)$

性质2（积分性质）$F\left\{\int^{x} f(x)dx\right\}=\dfrac{1}{i\omega}F(\omega)$

性质3（相似性质）$F\{f(ax)\}=\dfrac{1}{a}F\left(\dfrac{\omega}{a}\right)$

性质4（延迟性质）$F\{f(x-x_0)\}=e^{-i\omega x}F(\omega)$

性质5（位移性质）$F\{e^{ix\omega}f(x)\}=F(\omega-\omega_0)$

性质6（卷积性质）$F\{f_1(x)*f_2(x)\}=2\pi F_1(\omega)\cdot F_2(\omega)$

Fourier变换离散算法

傅里叶首次提出“在有限区间上由任意图形定义的任意函数都可以表示为单纯的正弦和余弦之和”。

1804

傅里叶出版了《热的解析理论》

1822

快速傅里叶变换算法 (FFT) 被提出

1965

本 讲 目 录

1、Fourier变换离散算法

2、Fourier变换快速算法

Four ier变换的离散化:

是分析离散信号有用工具;

在信号处理的理论上有重要意义;

在运算方法上起核心作用，谱分析、卷积、相关等都可以通DFT在计算机上实现。

离散Four ier变换是现代信号处理桥梁

Fourier变换离散算法

$$F(j\omega)=\int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

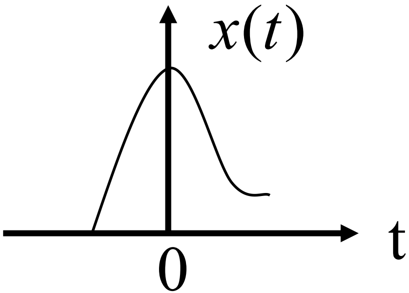
•--------$f(t)$ 的傅里叶变换

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}d\omega$$

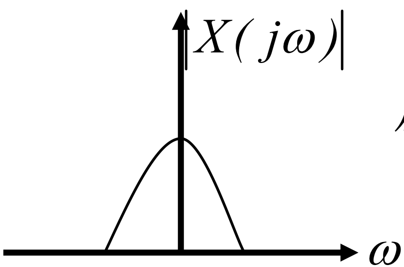
•--------$f(t)$ 的逆傅里叶变换

如何离散化？

1. 连续时间、连续频率的傅氏变换–傅氏变换。



$$正: X(j\omega)=\int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$$



$$反: x(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}d\omega$$

Fourier变换离散算法

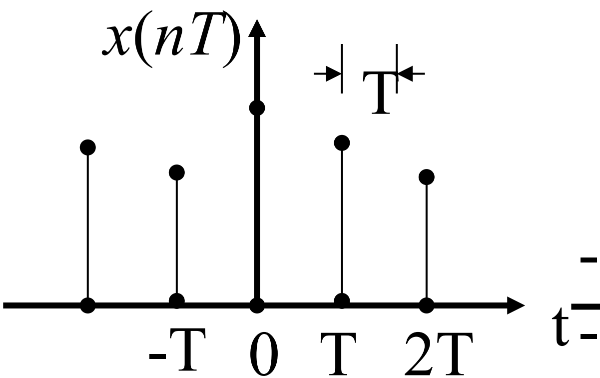
时域信号	频域信号
连续	非周期
非周期的	连续

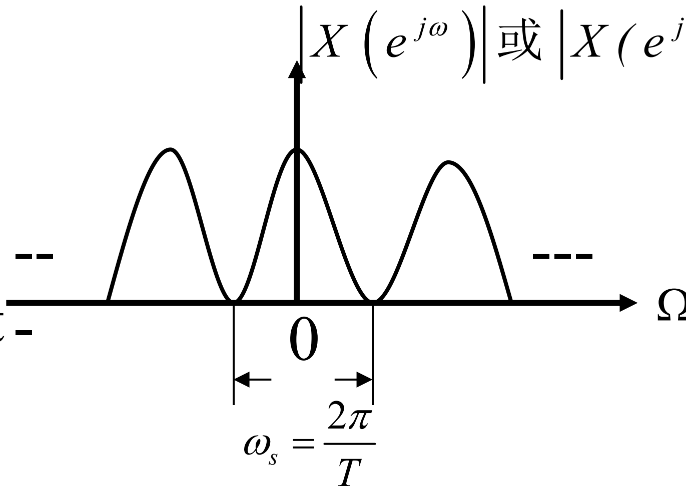
对称性：

时域连续，则频域非周期

反之亦然。

2、离散时间、连续频率的傅氏变换--序列的傅氏变换





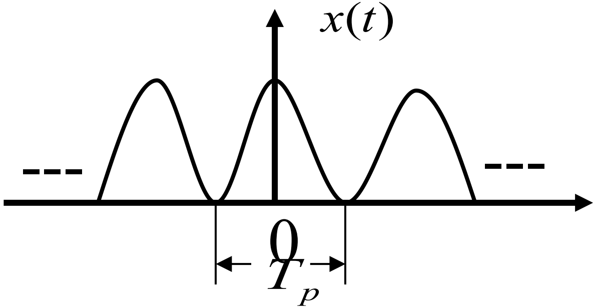
$$正：X(e^{j\omega T}) = \sum_{n=-\infty}^{\infty} x(nT)e^{-jn\omega T}$$

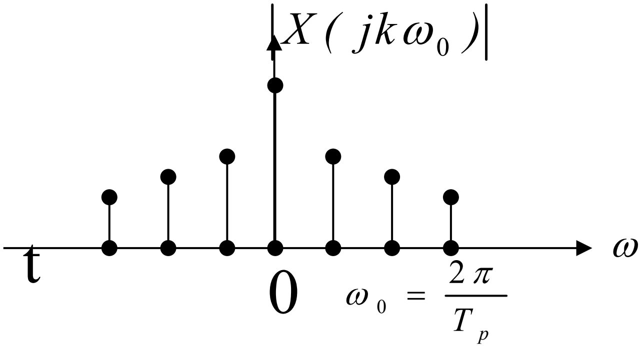
$$反：x(nT) = \frac{1}{\omega_s}\int_{-\omega_s/2}^{\omega_s/2} X(e^{j\omega T})e^{jn\omega T}d\omega$$

*时域抽样间隔为 T,频域的周期为 $\omega_s = \dfrac{2\pi}{T}$

时域信号	频域信号
离散的	周期的
非周期的	连续的

3、连续时间、离散频率傅里叶变换–傅氏级数





$$正:X(jk\omega_0) = \frac{1}{T_p}\int_{-T_p/2}^{T_p/2} x(t)e^{-jk\omega_0 t}dt$$

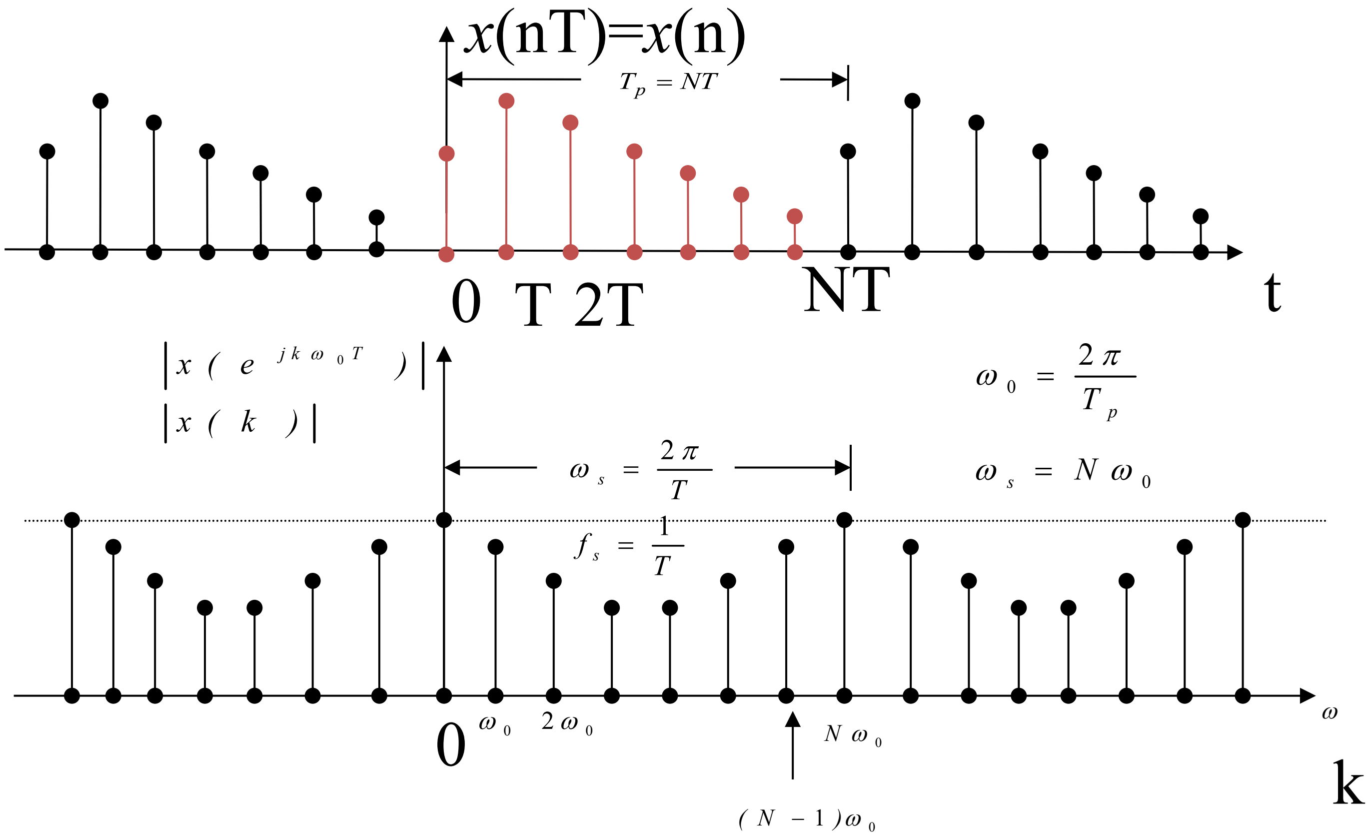
$$反:x(t) = \sum_{k=-\infty}^{\infty} X(jk\omega_0)e^{jk\omega_0 t}$$

*时域周期为T_p,

频域谱线间隔为$2\pi/T_p$

时域信号	频域信号
连续的	非周期的
周期的	离散的

4、离散时间、离散频率的傅氏变换--DFT



由上述分析可知，要想在时域和频域都是离散的，那么两域必须是周期的。

时域信号	频域信号
离散的	周期的
周期的	离散的

*时域是周期为T_p函数，频域的离散间隔为$\Omega_0 = \dfrac{2\pi}{T_p}$;

时域的离散间隔为T,频域的周期为$\Omega_s = \dfrac{2\pi}{T}$.

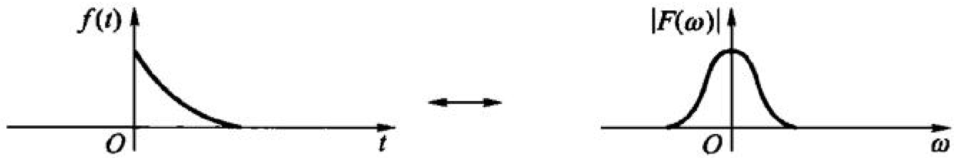
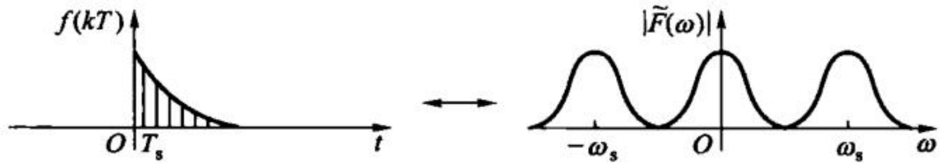
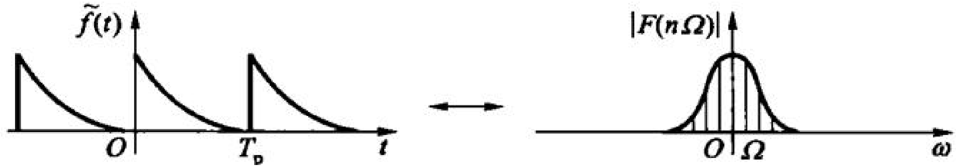
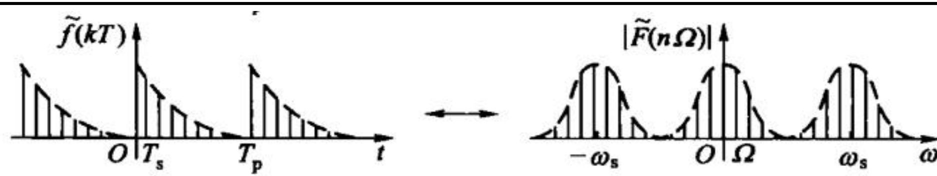
四、离散时间、离散频率的傅氏变换--DFT

$$X(e^{j\frac{2\pi}{N}k}) = \sum_{n=0}^{N-1} x(nT)e^{-j\frac{2\pi}{N}nk}$$

$$x(nT) = \frac{1}{N}\sum_{k=0}^{N-1} X(e^{j\frac{2\pi}{N}k})e^{j\frac{2\pi}{N}nk}$$

DFT是现代信号处理的核心变换

Fourier变换离散算法

时域	频域	
连续	**连续**	
非周期	非周期	
离散	**连续**	
非周期	周期	
连续	**离散**	
周期	非周期	
离散	**离散**	
周期	周期	

本讲目录

1、Fourier变换离散化

2、Fourier变换快速算法

四、离散时间、离散频率的傅氏变换--DFT

$$X(e^{j\frac{2\pi}{N}k}) = \sum_{n=0}^{N-1} x(nT)e^{-j\frac{2\pi}{N}nk}$$

$$W_N = e^{-j\frac{2\pi}{N}}$$

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

k=0， ， …， N-1

1、DFT运算的特点

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

```
function Xk = DFT(xn,N)
     Xk = zeros(1,N);
     for k=1:N
          sn =0.0;
          for n=1:N
               sn = sn+xn(n)*exp(-j*2*pi*n*k/N);
          end
          Xk(k) = sn;
      end
end
```

1、DFT运算的特点

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

```
function dft(x)
    N = length(x)
    X = zeros(complex(Float64), N)
    for k = 0:N-1
        for n = 0:N-1
            X[k+1] = X[k+1] + x[n+1] * exp(-
2im*pi*k*n/N)
        end
        X[k+1] = X[k+1] / N  # 归一化
    end
    return X
end

x = LinRange(-5, 5, 20);
X= dft(x)
```

DFT运算的mworks代码

```
function dft(x)
    N = length(x)
    X = zeros(complex(Float64), N)
    for k = 0:N-1
        for n = 0:N-1
            X[k+1] = X[k+1] + x[n+1]
 * exp(-2im*pi*k*n/N)
        end
        X[k+1] = X[k+1] / N  # 归一化
    end
    return X
end

x = LinRange(-5, 5, 20);
X= dft(x)
```

DFT的运算量

设复序列*x*(*n*) 长度为*N*点，其DFT为

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

（1）计算一个*X*(*k*) 值的运算量

复数乘法次数：*N*

复数加法次数：*N*－1

（2）计算全部N个$X(k)$ 值的运算量

复数乘法次数：N^2

复数加法次数：$N(N-1)$

（3）对应的实数运算量

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk}=\sum_{n=0}^{N-1}[\mathrm{Re}\,x(n)+j\,\mathrm{Im}\,x(n)][\mathrm{Re}\,W_N^{nk}+j\,\mathrm{Im}\,W_N^{nk}]$$

$$=\sum_{n=0}^{N-1}\{[\mathrm{Re}\,x(n)\cdot\mathrm{Re}\,W_N^{nk}-\mathrm{Im}\,x(n)\cdot\mathrm{Im}\,W_N^{nk}]$$

$$+j[\mathrm{Re}\,x(n)\cdot\mathrm{Im}\,W_N^{nk}+\mathrm{Im}\,x(n)\cdot\mathrm{Re}\,W_N^{nk}]\}$$

一次复数乘法： 4次实数乘法＋2次实数加法

一个$X(k)$： $4N$次实数乘法 ＋

$2N+2(N-1)=2(2N-1)$次实数加法

所以整个N点DFT运算共需要：

实数乘法次数：$4N^2$

实数加法次数：$N\times2(2N-1)=2N(2N-1)$

N点DFT的复数乘法次数举例

N	N^2	N	N^2
2	4	64	4049
4	16	128	16384
8	64	256	65 536
16	256	512	262 144
32	1028	1024	1 048 576

结论：当N很大时，其运算量很大，对实时性很强的信号处理来说，要求计算速度快，因此需要改进DFT的计算方法，以大大减少运算次数。

Fourier变换离散算法

主要原理是利用系数 W_N^{nk} 的以下特性对DFT进行分解：

（1）对称性

$$(W_N^{nk})^* = W_N^{-nk} = W_N^{k(N-n)}$$

（2）周期性

$$W_N^{(n+N)k} = W_N^{n(k+N)} = W_N^{nk}$$

（3）可约性

$$W_{mN}^{mnk} = W_N^{nk} \qquad W_N^{nk} = W_{N/m}^{nk/m}$$

另外，

$$W_N^{N/2} = -1 \qquad W_N^{(k+N/2)} = -W_N^k$$

FFT算法原理(安时间抽取)

- 按时间抽取基-2FFT算法与直接计算DFT运算量的比较
- 按时间抽取的FFT算法的特点
- 按时间抽取FFT算法的其它形式流程图

FFT算法的基本思想

- 引入：多项式的乘法计算

系数表示法：

$$P_1(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$$

$$P_2(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_k x^k$$

两个k阶多项式相乘，结果为___2k___阶多项式。

$$P_1(x)P_2(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{2k} x^{2k}$$

一般的计算方法：乘法分配律　　$O(k^2)$

2. FFT算法的基本思想

- 引入：多项式的乘法计算

点值表示法：任一k阶多项式可由<u>k+1</u>个点唯一确定。

$$P_1(x):\{(x_0,P_1(x_0)),(x_1,P_1(x_1)),\cdots,(x_k,P_1(x_k)),\cdots\}$$

$$P_2(x):\{(x_0,P_2(x_0)),(x_1,P_2(x_1)),\cdots,(x_k,P_2(x_k)),\cdots\}$$

若已知多项式的点表示法，那么只需挑出<u>2k+1</u>个点，然后将函数值一对一相乘，从而得到乘出来的多项式的点值表示。

$$O(k)$$

2. FFT算法的基本思想

- 引入：多项式的乘法计算

问题：

$$P_1(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$$

如何转化？

$$P_1(x):\ \{(x_0, P_1(x_0)), (x_1, P_1(x_1)), \cdots, (x_k, P_1(x_k))\}$$

暴力方法：在x轴上找n(≥k+1)个点，一个一个计算函数值

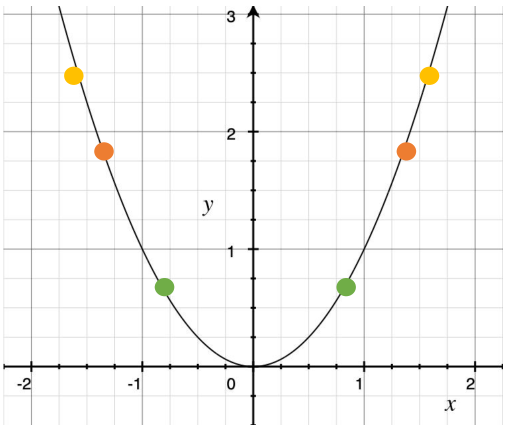
每个点计算都是O(k)，加起来还是O(nk)≥O(k^2)

2. FFT算法的基本思想

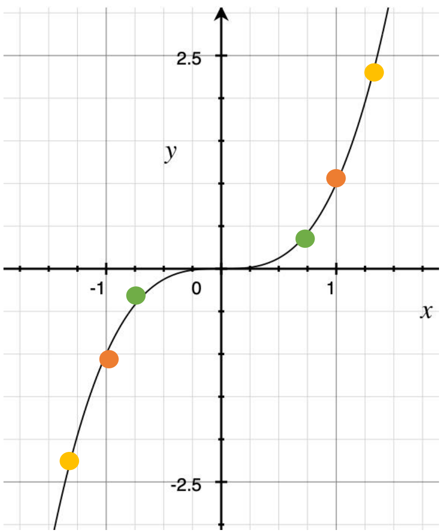
- 引入：多项式的乘法计算

减少计算量：

$$P(x) = x^2$$



$$P(x) = x^3$$



偶/奇函数----计算量减半

FFT算法的基本思想

- 引入：多项式的乘法计算

解决思路： 不妨设k为偶数

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$$

$$P(x) = \underbrace{(a_0 + a_2 x^2 + \cdots + a_k x^k)}_{P_e(x^2)} + x\underbrace{(a_1 + a^3 x^2 + \cdots + a_{k-1} x^{k-2})}_{P_o(x^2)}$$

$$[\pm x_1, \pm x_2, \cdots, \pm x_{k/2}]$$

$$P(x_i) = P_e(x_i^2) + xP_o(x_i^2) \quad P(-x_i) = P_e(x_i^2) - xP_o(x_i^2)$$

$$[x_1^2, x_2^2, \cdots, x_{k/2}^2]$$

递归？

FFT算法的基本思想

考察DFT与IDFT的运算发现，利用以下两个特性可减少运算量：

$$w_N^{nk} = e^{-j\frac{2\pi}{N}nk}$$

（1）折半引理 $$w_N^{2nk} = w_{N/2}^{nk}$$

（2）消去引理 $$w_N^{n(k+N/2)} = (-1)^n w_N^{nk}$$ $$w_N^{k(n+N/2)} = (-1)^k w_N^{kn}$$

$$w_N^{k+N/2} = -w_N^k$$

FFT算法正是基于这样的基本思想发展起来的。它有多种形式，但基本上可分为两类：时间抽取法和频率抽取法。

按时间抽取的FFT（N点DFT运算的分解）

先从一个特殊情况开始，假定N是2的整数次方，即 $N=2^M$，M为正整数。

首先将序列x(n)分解为两组，一组为偶数项，一组为奇数项：

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

$$x(n):\ \{x_0, x_1, x_2, x_3, \cdots, x_{N-2}, x_{N-1}\}$$

$$\{x_0, x_2, x_4, \cdots, x_{N-2}\} \qquad \{x_1, x_3, x_5, \cdots, x_{N-1}\}$$

偶数样本

$$x_1(r) = x(2r)$$

奇数样本

$$x_2(r) = x(2r+1)$$

N=2³=8 的例子: $X(k)=\sum_{n=0}^{7} x(n)W_8^{nk}$

$$\{x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7\}$$

X(k)

X(0) X(1) X(2) X(3) X(4) X(5) X(6) X(7)

N=4
k=0,1,2,3,4,5,6,7

$$\{x_0,x_2,x_4,x_6\}$$

G(k)

G(0) G(1) G(2) G(3)

$$\{x_1,x_3,x_5,x_7\}$$

H(k)

H(0) H(1) H(2) H(3)

N=4
k=0,1,2,3

$$\{x_0,x_4\}$$

A(k)

A(0) A(1)

$$\{x_2,x_6\}$$

B(k)

B(0) B(1)

$$\{x_1,x_5\}$$

C(k)

C(0) C(1)

$$\{x_3,x_7\}$$

D(k)

D(0) D(1)

N=2
k=0,1

FFT算法原理

设$N=2^L$，将$x(n)$按 n 的奇偶分为两组：

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

则

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

$$=\sum_{\substack{n=0 \\ n\text{为偶数}}}^{N-1}x(n)W_N^{nk}+\sum_{\substack{n=0 \\ n\text{为奇数}}}^{N-1}x(n)W_N^{nk}$$

$$= \sum_{\substack{n=0 \\ n\text{为偶数}}}^{N-1} x(n)W_N^{nk} + \sum_{\substack{n=0 \\ n\text{为奇数}}}^{N-1} x(n)W_N^{nk}$$

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

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

式中，$X_1(k)$和$X_2(k)$分别是$x_1(n)$和$x_2(n)$的$N/2$的DFT。

另外，式中k的取值范围是：0，1， …，N/2－1 。

因此，$X(k)=X_1(k)+W_N^k X_2(k)$ 只能计算出X(k)的前一半值。

后一半X(k) 值， N/2， N/2 ＋1， …， N？

利用 $W_{N/2}^{r(N/2+k)} = W_{N/2}^{rk}$

可得到

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

同理可得

$$X_2(\frac{N}{2}+k)=X_2(k)$$

考虑到

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

及前半部分$X(k)$

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

k=0，1，…，N/2－1

因此可得后半部分$X(k)$

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

$$= X_1(k) - W_N^k X_2(k)$$

k=0，1，…，N/2－1

蝶形运算

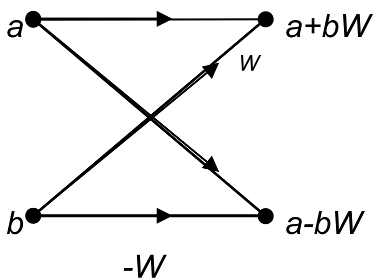
$$\left.\begin{aligned} X(k) &= X_1(k) + W_N^k X_2(k) \\ X(k) &= X_1(k) - W_N^k X_2(k) \end{aligned}\right\} \Rightarrow \text{蝶形运算式}$$

因此，只要求出2个N/2点的DFT，即X_1(k)和X_2(k)，再经过蝶形运算就可求出全部X(k)的值，运算量大大减少。

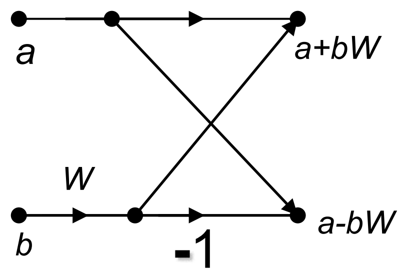
蝶形信号流图

将*a*和b 合成*X*(*k*)运算可归结为：

$$\begin{cases} a - bW \\ a + bW \end{cases}$$

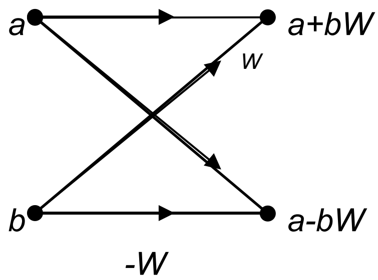


(a)

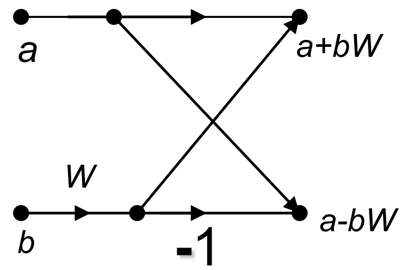


(b)

蝶形运算的简化



(a)

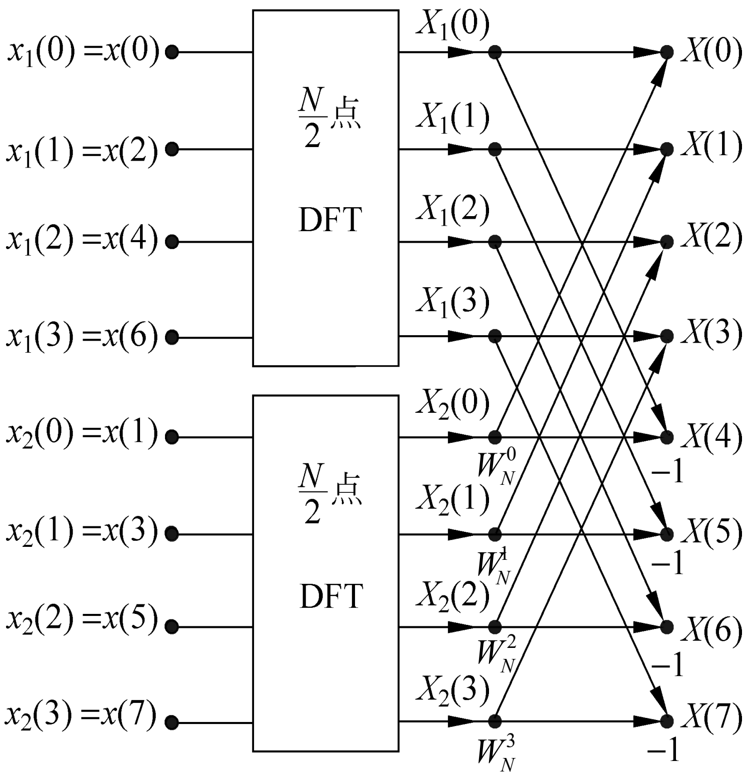


(b)

图 (a)为实现这一运算的一般方法，它需要两次乘法、两次加减法。考虑到-*bW*和*bW*两个乘法仅相差一负号，可将图 (a)简化成图 (b)，此时仅需一次乘法、两次加减法。

图 (b)的运算结构像一蝴蝶通常称作蝶形运算结构简称蝶形结，采用这种表示法，就可以将以上所讨论的分解过程用流图表示。

Fourier变换离散算法



以N=8为例，分解为2个4点的DFT，然后做8/2=4次蝶形运算即可求出所有8点$X(k)$的值。

Fourier变换离散算法

- N点DFT的运算量

  复数乘法次数：N^2

  复数加法次数：$N(N-1)$

- 分解一次后所需的运算量＝2个N/2的DFT＋N/2蝶形：

  复数乘法次数：$2*(N/2)^2+N/2=N^2/2+N/2$

  复数加法次数：$2*(N/2)(N/2-1)+2*N/2=N^2/2$

- 因此通过一次分解后，运算工作量减少了差不多一半。

由于$N=2^L$，因而$N/2$仍是偶数，可以进一步把每个$N/2$点子序列再按其奇偶部分分解为两个$N/4$点的子序列。

以$N/2$点序列$x_1(r)$为例 $\left.\begin{aligned} x_1(2l) &= x_3(l) \\ x_1(2l+1) &= x_4(l) \end{aligned}\right\}$ $l = 0,1,\ldots,\frac{N}{4}-1$

则有

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

$$= \sum_{l=0}^{N/4-1} x_3(l) W_{N/4}^{lk} + W_{N/2}^{k} \sum_{l=0}^{N/4-1} x_4(l) W_{N/4}^{lk}$$

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

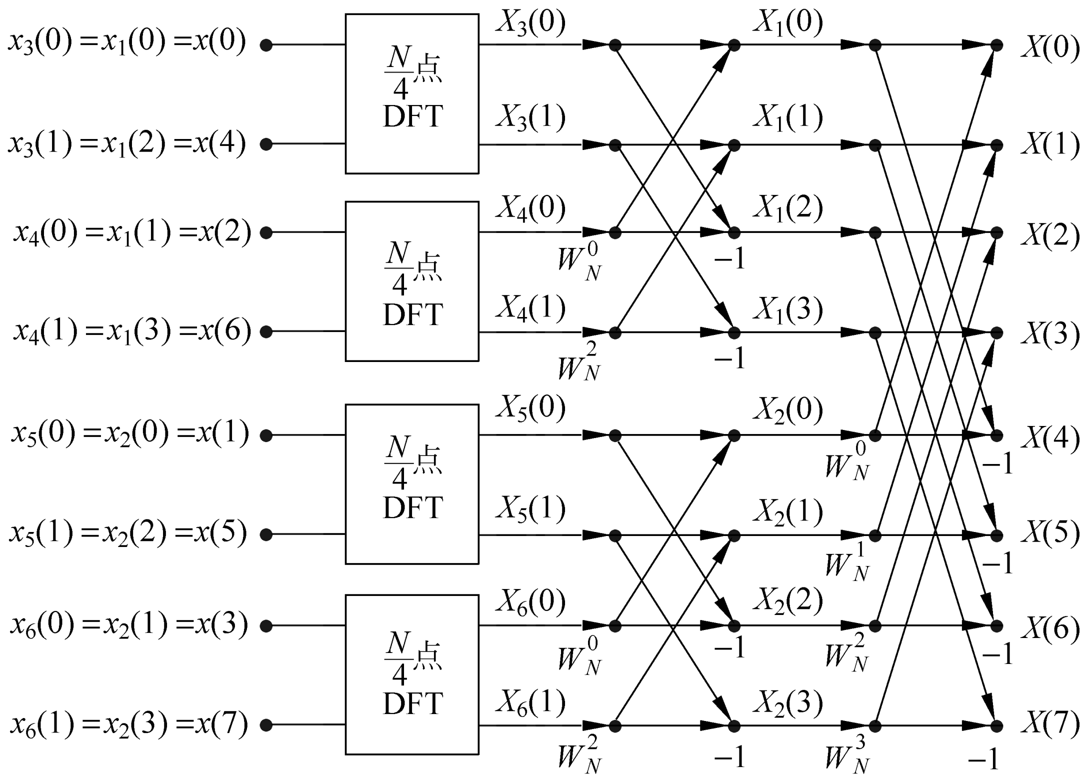
且

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

由此可见，一个$N/2$点DFT可分解成两个$N/4$点DFT。

同理，也可对$x_2(n)$进行同样的分解，求出$X_2(k)$。

Fourier变换离散算法



对此例N=8，最后剩下的是4个N/4= 2点的DFT，2点DFT也可以由蝶形运算来完成。以$X_3(k)$为例。

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

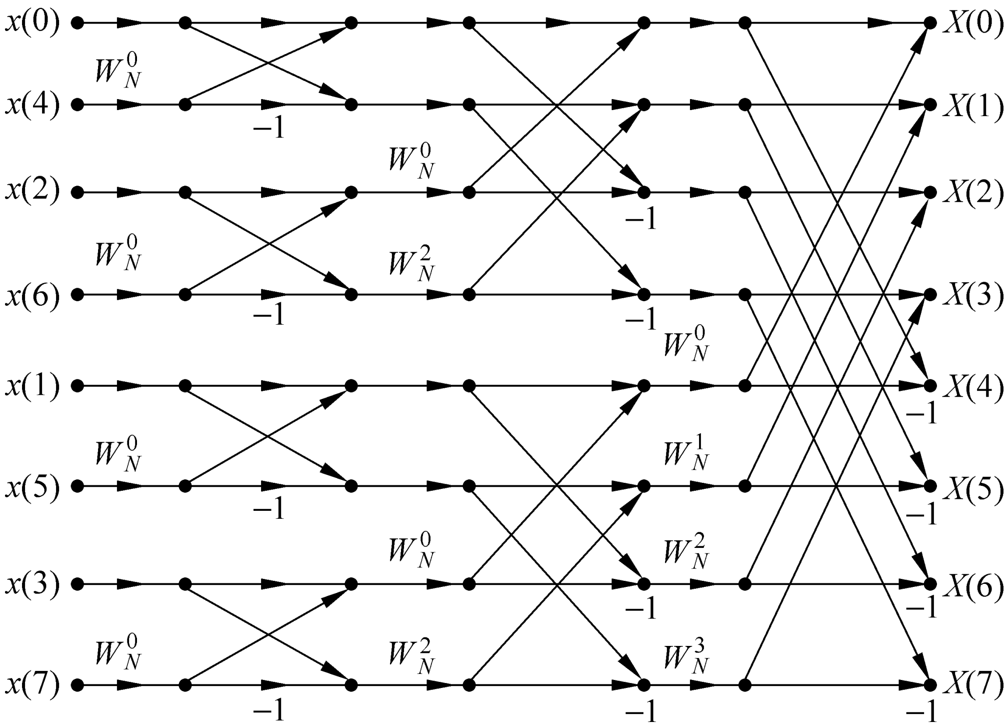
即

$$X_3(0)=x_3(0)+W_2^0x_3(1)=x(0)+W_2^0x(4)=x(0)+W_N^0x(4)$$

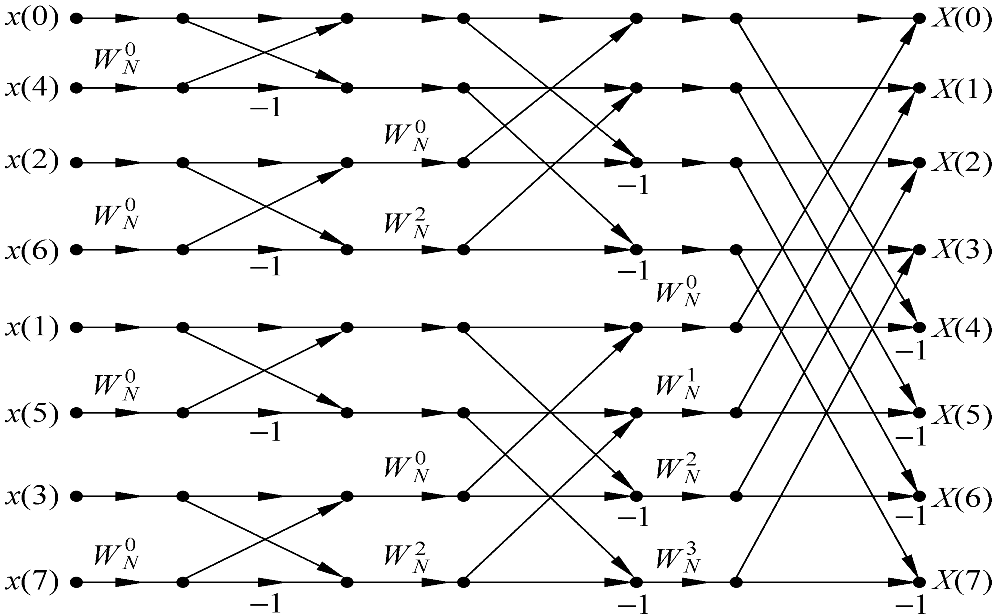
$$X_3(1)=x_3(0)+W_2^1x_3(1)=x(0)+W_2^1x(4)=x(0)-W_N^0x(4)$$

这说明，N=2^M的DFT可全部由蝶形运算来完成。

Fourier变换离散算法



N=8按时间抽取法FFT信号流图



由按时间抽取法FFT的信号流图可知，当$N=2^L$时，共有__L__级蝶形运算；每级都由__$N/2$__个蝶形运算组成，而每个蝶形有__1__次复乘、__2__次复加，因此每级运算都需__$N/2$__次复乘和__N__次复加。

这样 L 级运算总共需要：

复数乘法：$\frac{N}{2} \cdot L = \frac{N}{2} \log_2 N$

复数加法：$N \cdot L = N \log_2 N$

直接DFT算法运算量

复数乘法：N^2

复数加法：$N(N-1)$

直接计算DFT与FFT算法的计算量之比为M

$$M = \frac{N^2}{\frac{N}{2} \log_2 N} = \frac{2N}{\log_2 N}$$

FFT算法与直接DFT算法运算量的比较

N	N^2	$\frac{N}{2}\log_2 N$	计算量之比M	N	N^2	$\frac{N}{2}\log_2 N$	计算量之比M
2	4	1	4.0	128	16 384	448	36.6
4	16	4	4.0	256	65 536	1 024	64.0
8	64	12	5.4	512	262 144	2 304	113.8
16	256	32	8.0	1024	1 048 576	5 120	204.8
32	1028	80	12.8	2048	4 194 304	11 264	372.4
64	4049	192	21.4				

由于这种方法每一步分解都是按输入时间序列是属于偶数还是奇数来抽取的，所以称为“按时间抽取法”或“时间抽取法”。

3、按时间抽取的FFT（N点DFT运算的分解）

```
function Xk = FFT(xn,N)
    % N为2的幂次方
    if N==1
        Xk = xn
    end
    w = e^(2*pi*j/N);
    x1 = [xn_0,xn_2,...,xn_N-2];
    x2 = [xn_1,xn_3,...,xn_N-1]; % 按照奇偶分成两半
    G = FFT(x1, N/2);
    H = FFT(x2, N/2);  % 递归
    Xk = zeros(1,N);
    for k=1:N/2
        Xk(k) = G(k)+w^k*H(k);
        Xk(k+N/2) = G(k)-w^k*H(k);
    end
```

FFT运算的mworks代码

```
ffunction fft(x::Vector{Complex{Float64}})
    N = length(x)
    if N == 1
        return x
    else
        X_even = fft(x[1:2:end-1])
        X_odd = fft(x[2:2:end])
        factor = exp(-2im * pi * (0:N/2-1) ./ N)
        return [X_even + factor .* X_odd; X_even - factor .*
X_odd]
    end
end

function to_complex(x::Vector{Float64})
    return x + 0im
end
```

4、按频率抽取的FFT（N点DFT运算的分解）

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

$$x(n):\ \{x_0, x_1, x_2, x_3, \cdots, x_{N-2}, x_{N-1}\}$$

$$\{x_0, x_1, x_2, \cdots, x_{N/2-1}\} \qquad \{x_{N/2}, x_{N/2+1}, \cdots, x_{N-1}\}$$

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

4、按频率抽取的FFT（N点DFT运算的分解）

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

4、按频率抽取的FFT（N点DFT运算的分解）

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

然后按照偶数k=2r和奇数k=2r+1，上式变为:

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

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

根据折半引理： $W_N^{2nk}=W_{N/2}^{nk}$

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

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

N=2³=8 的例子: $X(k)=\sum_{n=0}^{7}x(n)W_8^{nk}$

$$X(2r)=\sum_{n=0}^{3}\left[x(n)+x(n+\frac{N}{2})\right]W_4^{nr},\ r=0,1,2,3$$

$$X(2r+1)=\sum_{n=0}^{3}\left[x(n)-x(n+\frac{N}{2})\right]W_8^{n}W_4^{nr},\ r=0,1,2,3$$

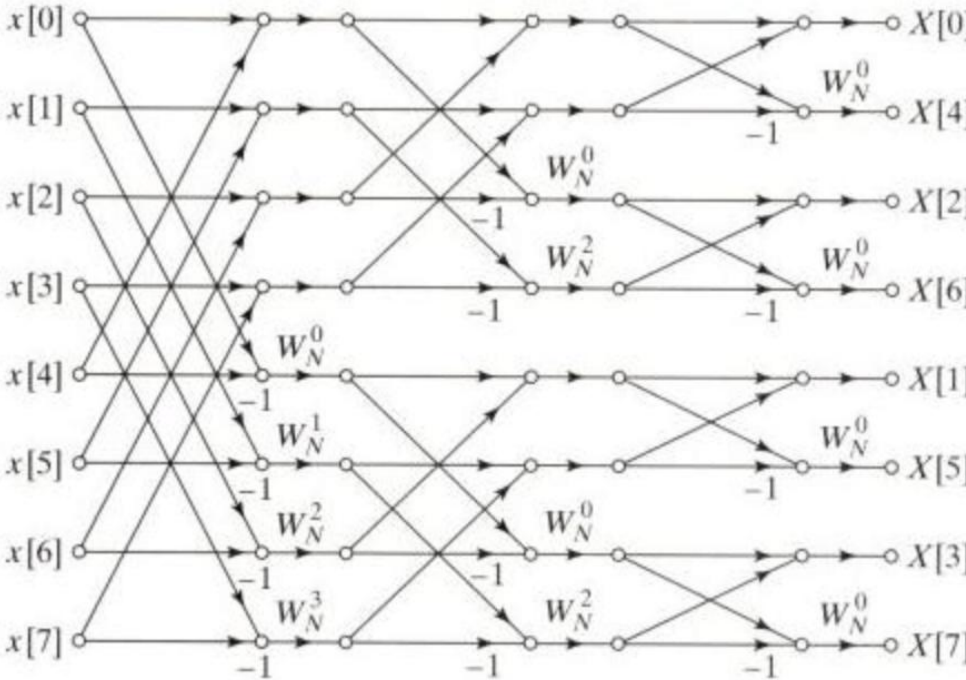
X(2r) 是以下序列的4点DFT:

$$[x(0)+x(4),\ x(1)+x(5),\ x(2)+x(6),\ x(3)+x(7)]$$

X(2r+1) 是以下序列的4点DFT:

$$\left[x(0)-x(4),\ [x(1)+x(5)]W_8^1,\ [x(2)-x(6)]W_8^2,\ [x(3)-x(7)]W_8^3\right]$$

DIF FFT



x[0]
x[1]
x[2]
x[3]
x[4]
x[5]
x[6]
x[7]
W_N^0
W_N^1
W_N^2
W_N^3
W_N^0
W_N^2
W_N^0
W_N^2
W_N^0
W_N^0
W_N^0
W_N^0
−1
X[0]
X[4]
X[2]
X[6]
X[1]
X[5]
X[3]
X[7]

作业

- 1、自己编写DFT与FFT程序，请自己录一段语音，并对此语音进行DFT和FFT运算，比较两种方法的异同。
- 2、理解FFT算法的核心思想，画出基-2时间抽取8点信号的蝶形运算图。

- **本周提交最终日期为3月29日。**

谢　谢