数字信号处理

2017年秋冬学期

第五讲

2017年10月23日

第3章 离散傅里叶变换及其快速计算方法

3.3 DFT及其性质—DFT 的应用

■ 线性卷积求解-分段卷积-重叠相加法

- 重叠相加法—对输出$y(n)$ 进行后处理
- 基本思路
  - 假设一个系统的输入为长序列项 $x(n)$, 脉冲响应 $h(n)$ 为 M 点序列；
  - 把序列 $x(n)$ 分成多段 N点序列，并且$N>M$；
  - 将每段序列和 $h(n)$ 进行线性卷积（通过循环卷积实现），然后再将结果 $y(n)$ 重叠相加。
  - 问题：重叠部分如何处理？

3.3 DFT及其性质—DFT 的应用

■ 线性卷积求解-分段卷积-重叠相加法

步骤：

(1) 设 $h(n)$ 长为 M，$x(n)$ 为长序列，首先把 $x(n)$ 分段，每段长为 N，即把 $x(n)$ 分为 $x_0(n), x_1(n),\ldots,x_l(n),\ldots$ 表示为：

$$x_l(n)=\begin{cases}x(n) & lN\le n\le(l+1)N-1\\ 0 & \text{else}\end{cases}$$

则

$$x(n)=\sum_{k=-\infty}^{\infty}x_l(n)$$

3.3 DFT及其性质—DFT 的应用

■ 线性卷积求解-分段卷积-重叠相加法

(2) 线性卷积，

$$y(n)=x(n)*h(n)=\left[\sum_{l=-\infty}^{\infty}x_l(n)\right]*h(n)$$

$$=\sum_{l=-\infty}^{\infty}\underbrace{[x_l(n)*h(n)]}_{\text{第}l\text{段线性卷积}}=\sum_{l=-\infty}^{\infty}y_l(n)$$

$x_l(n)$ 长为 N，$h(n)$ 长为M

$y_l(n)$ 的长为：$L=N+M-1$

因此, $y_l(n)$ 比 $x_l(n)$ 长，多余的点数为

$L-N=N+M-1-N=M-1$

3.3 DFT及其性质—DFT 的应用

■ 线性卷积求解-分段卷积-重叠相加法

由于 $y_l(n)$ 的范围为：

$$lN\le n\le(l+1)N+M-2$$

而 $y_{l+1}(n)$ 的范围是：

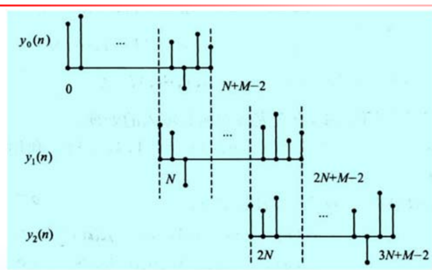
$$(l+1)N\le n\le(l+2)N+M-2$$

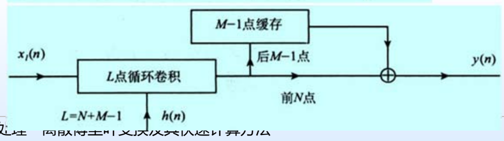
因此，线性卷积 $y_l(n)$ 最后的（M-1）项与 $y_{l+1}(n)$ 最开始的（M-1）项发生了重叠。

$$y(n)=y_k(n)+y_{k+1}(n)$$

这种方法把 $y_l(n)$ 最后的（M-1）项与 $y_{l+1}(n)$ 最开始的（M-1）项相加起来得到 $y((l+1)N),\ldots, y((l+1)N+M-2)$ 各项，所以称为重叠相加法。

3.3 DFT及其性质—DFT 的应用





3.3 DFT及其性质—DFT 的应用

■ 线性卷积求解-分段卷积-重叠相加法

例 3.21 设 $x(n)=(n+1), 0\le n\le 9, h(n)=\{1,0,-1\}$，按 $N=6$ 用重叠相加法计算

$$y(n)=x(n)*h(n)$$

解：因为 $N=6$，则把 $x(n)$ 分为两段：

$$x_1=[1,2,3,4,5,6]$$

$$x_2=[7,8,9,10,0,0]$$

因为 $x(n)$ 在 $n>9$ 时无值，可以填两个零，或者不填零。现在计算每一部分与 $h(n)$ 线性卷积或 $L=N+M-1$ 点循环卷积。

3.3 DFT及其性质—DFT 的应用

■ 线性卷积求解-分段卷积-重叠相加法

$$y_1(n)=x_1(n)*h(n)$$
$$=\{1\quad 2\quad 2\quad 2\quad 2\quad 2\quad -5\quad -6\}$$
$$y_2(n)=x_2(n)*h(n)$$
$$=\{7\quad 8\quad 2\quad 2\quad -9\quad -10\quad 0\quad 0\}$$

把 $y_1(n)$ 最后的 $(M-1)=2$ 项与 $y_2(n)$ 最开始的 $(M-1)=2$ 项相加起来得到相应的各项，最后输出为：

$$y(n)=\{1,2,2,2,2,2,2,2,2,2,2,-9,-10\}$$

3.3 DFT及其性质—DFT 的应用

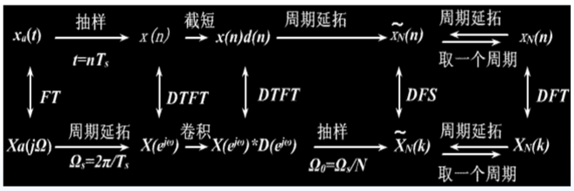
■ 线性卷积求解-分段卷积-小结

- 重叠相加法: 对输入序列分段后进行线性卷积（循环卷积），对分段输出进行重叠相加，形成完整输出（输出端处理）。
- 重叠保留法: 对输入序列进行重叠分段，构成短序列，分别进行循环卷积运算，对分段输出序列，先舍去再拼接，形成最终输出（输入端处理）。
- 共同特点: 以逐段的方式通过循环卷积来完成线性卷积的计算，具有相同的运算量和处理效率。

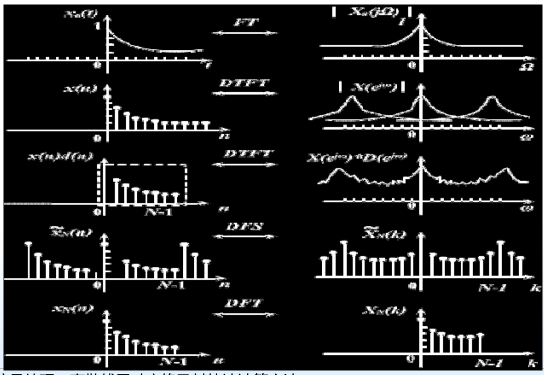
3.3 DFT及其性质—DFT 的应用

■ 频谱分析

信号的频谱分析：计算信号的傅里叶变换



3.3 DFT及其性质—DFT 的应用



3.3 DFT及其性质—DFT 的应用

■ 频谱分析

参数的选择

T – 时域取样间隔

f_s – 时域取样频率

T_0 – 信号记录长度（时域周期）

F_0 –（频率分辨率）频域取样间隔

N – 取样点数

f_h – 信号最高频率

$$f_s\ge 2f_h$$
$$f_s=1/T$$
$$T_0=NT$$
$$T_0=1/F_0$$
$$f_s=1/T=N/T_0=NF_0$$
$$N=\frac{T_0}{T}=\frac{f_s}{F_0}$$

在信号最高频率不变情况下，要提高频率分辨率，需增加采样点数 N，即采集更长的数据。为便于 FFT 计算，一般选择 $N=2^r$（r 为正整数）

3.3 DFT及其性质—DFT 的应用

- **频谱分析**

DFT 计算频谱

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

- 当 $k=0,1,\cdots,\frac{N}{2}$ 时，$X(k)$ 对应频率为 $\frac{f_s}{N}k$ 的频谱值。
- 频谱 $X(k)$ 是由实部 U 和虚部 V 组成：

$$X(k)=U(k)+jV(k),\quad k=0,\cdots,\cdots N$$

- 于是幅频响应$A(k)$、相频响应$\Phi(k)$、功率谱$G(k)$ 分别为：

$$A(k)=|X(k)|\qquad \Phi(k)=\arctan\frac{V(k)}{U(k)}\qquad G(k)=|X(k)|^2$$

3.3 DFT及其性质—DFT 的应用

- **频谱分析**

例 有一频谱分析用的 DFT/FFT 处理器，其取样点数为 2 的整数次幂，假设没有采用任何的数据处理措施，已给条件为：频率分辨率 ≤10Hz，信号最高频率≤4 kHz。

试确定以下参量：

1）最小记录长度 T_0；

2）取样点最大时间间隔 T（即最小取样频率）；

3）在一个记录中最少点数 N。

3.3 DFT及其性质—DFT 的应用

- **频谱分析**

解：1）最小记录长度：

$$T_0\ge\frac{1}{F_0}=\frac{1}{10Hz}=0.1s$$

2）最大取样间隔 $(f_s>2f_h\quad f_s=1/T)$

$$T<\frac{1}{2f_h}=\frac{1}{2\times4\times10^3Hz}=0.125ms$$

3）最小记录点数

$$N>\frac{f_s}{F_0}=\frac{2f_h}{F_0}=\frac{2\times4\times10^3}{10}=800$$

$$\text{取 } N=2^m=2^{10}=1024>800$$

3.3 DFT及其性质—DFT 的应用

- **频谱分析**

例 有一调幅信号

$$x_a(t)=[1+\cos(2\pi\times100t)]\cos(2\pi\times600t)$$

用 DFT 做频谱分析，要求能分辨 $x_a(t)$ 的所有频率分量，问

(1) 取样频率应为多少赫兹?

(2) 取样时间间隔应为多少秒?

(3) 若用 f_s= 3 kHz 频率取样，取样数据为 512 点，做频谱分析，求 512 点 $X(k)$，并画出 $X(k)$ 的幅频特性 $|X(k)|$，标出主要点的坐标值。

3.3 DFT及其性质—DFT 的应用

- **频谱分析**

解： $x_a(t)=[1+\cos(2\pi\times100t)]\cos(2\pi\times600t)$

$$=\cos(2\pi\times600t)$$
$$+\frac{1}{2}\cos(2\pi\times700t)+\frac{1}{2}\cos(2\pi\times500t)$$

（1）取样频率应为:

$$f_s>2\times700=1400\text{Hz}$$

（2）取样时间间隔应为:

$$T<\frac{1}{f_s}=\frac{1}{1400}=0.00072\text{ s}=0.72\text{ ms}$$

3.3 DFT及其性质—DFT 的应用

- **频谱分析**

③ $x(n)=x_a(t)|_{t=nT}$

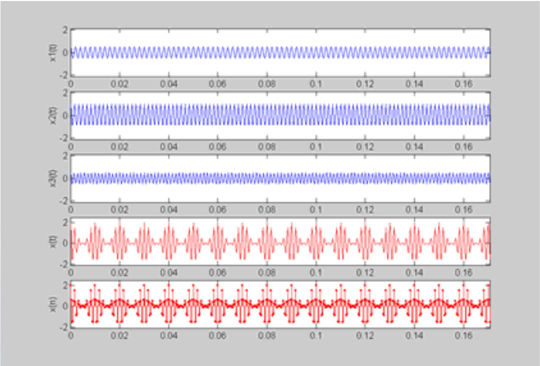
$$=\cos\left(2\pi\times\frac{6}{14}n\right)$$
$$+\frac{1}{2}\cos\left(2\pi\times\frac{7}{14}n\right)+\frac{1}{2}\cos\left(2\pi\times\frac{5}{14}n\right)$$

如果 $2\pi/\omega$ 是有理数（整数或分数），则仍为周期序列

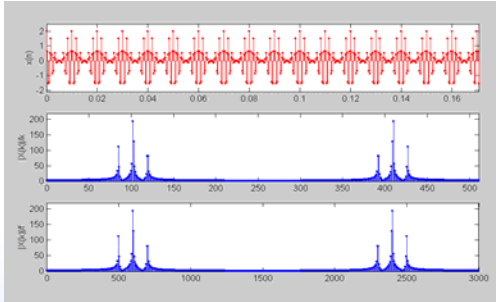
所以此例中 $x(n)$ 为周期序列，周期 N=14

因此，取样点数至少为 14 点，最小记录点数 N=14。

3.3 DFT及其性质—DFT 的应用



3.3 DFT及其性质—DFT 的应用



3 kHz 512点

3 kHz 3000点

$$\omega = \Omega T = 2\pi f / f_s \qquad f = \frac{f_s}{N}k$$

$$\omega = 2\pi k / N$$

3.3 DFT及其性质—DFT 的应用

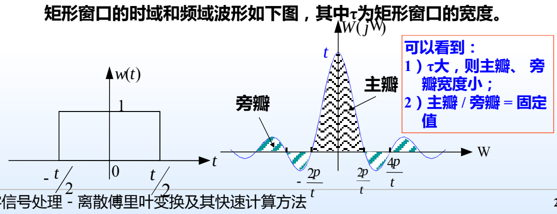
■ 频谱泄漏（窗效应）

对时域截短，使频谱变宽拖尾，称为泄漏

– 实际中的离散时间序列 $x(n)$ 可能是非时限的，处理这样序列时需要将它***截短***，即把该序列限定为 N 点，相当于将 $x(n)$ 乘以一个矩形窗口 $W(nT)$，根据频域卷积定理：时域相乘映射为频域卷

$$x(n)\cdot W(n) \to X(e^{j\omega}) * W(e^{j\omega})$$

矩形窗口的时域和频域波形如下图，其中τ为矩形窗口的宽度。

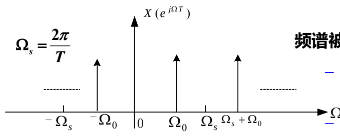


3.3 DFT及其性质—DFT 的应用

■ 频谱泄漏（窗效应）

例如： $x(n) = e^{j\Omega_0 nT}$ $\xleftrightarrow{DFT}$ $X(e^{j\Omega T}) = \frac{2\pi}{T}\sum_{n=-\infty}^{\infty}\delta(\Omega_0 + n\Omega_s)$

$\Omega_s = \frac{2\pi}{T}$

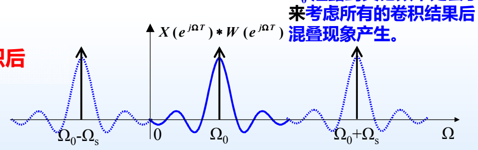


频谱被展宽，同时也造成混叠失真

– 原来在 Ω_0 处的一根谱线，变成了以 Ω_0 为中心的连续谱。

– 也就是说 $X(e^{j\Omega T})$ 的频率成分从 Ω_0 泄露到其他频率处去了。原来考虑所有的卷积结果后，则有混叠现象产生。

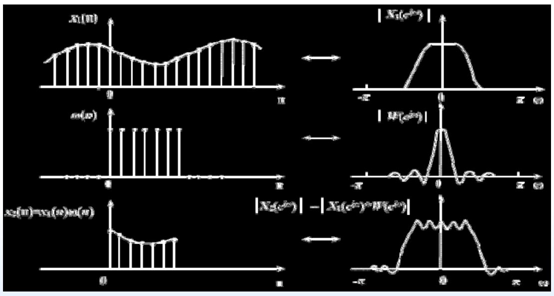
卷积后



3.3 DFT及其性质—DFT 的应用

■ 频谱泄漏（窗效应）

对时域截短，使频谱变宽拖尾，称为泄漏



改善方法：1）增加 $x(n)$ 长度 N；2）缓慢截短(窗函数)

3.3 DFT及其性质—DFT 的应用

■ 频谱泄漏（窗效应）

怎样减少泄漏效应？

– 选择适当长度的窗函数使主瓣变窄，提高分析的精度。
  - 当 $\tau\to\infty$ 时，$W(e^{j\Omega T})$ 变为在 $\Omega=0$ 处的冲激函数，而冲激函数与任何 $X(e^{j\Omega T})$ 相卷积都不会使 $X(e^{j\Omega T})$ 有所改变。

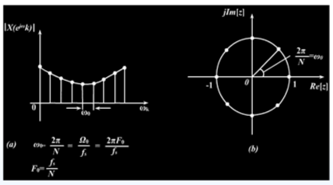
– 选择适当的窗函数
  - 为了减少泄露，尽量寻找频谱接近冲激函数的窗函数（即旁瓣小，主瓣窄，但这是矛盾的）。
  - 常用的窗函数有汉明窗、汉宁窗、凯塞窗等

3.3 DFT及其性质—DFT 的应用

■ 栅栏效应

栅栏效应：DFT 只计算离散点（基频 F_0 的整数倍处）的频谱，而不是连续函数。

$$x(n)\xrightarrow{DFT}\text{离散频谱}\rightarrow\text{连续频谱上的若干点}$$

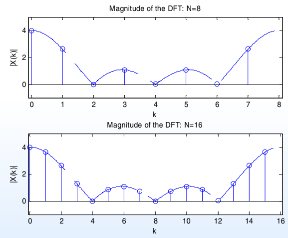


这好象是在栅栏的一边通过缝隙观看另一边的景色一样，所以称之为栅栏效应。被“栅栏”挡住的景色是看不到的，所以有可能漏掉大的频谱分量。

3.3 DFT及其性质—DFT 的应用

■ 栅栏效应

- 改善方法：增加频域取样点数 N（时域补零），使谱线更密
- 补0不会影响信号的 DFT 特性，因为它没有增加信号的信息，它只是可以减少“栅栏”，看到了原来就存在的信息。



3.4 FFT

- FFT: Fast Fourier Transform
- 1965 年，James W. Cooley 和 John W. Tukey 在《Mathematics of Computation》上发表了 ”一种用机器计算复序列傅立叶级数的算法（An algorithm for the machine calculation of complex Fourier series）” 论文。
- 自此之后，新的算法不断涌现。一种是对N等于 2 的整数次幂的算法，如基 2 算法，基 4 算法。另一种是N不等于 2 的整数次幂的算法，例如 混合基算法。

James Cooley(1926-)美国数学家，哥伦比亚大学的数学博士，以他所创造的快速傅立叶变换(FFT)而著名，FFT的数学意义不光在于使大家明白了傅立叶(Fourier)变换计算起来是多么容易，而且使得数字信号处理技术取得了突破性的进展，对于现在的网络通信，图形图像处理等领域的发展与前进奠定了基础。

John W. Tukey(1915-2000) 普林斯顿大学和贝尔实验室统计学家 快速傅立叶变换的发展者之一

3.4 FFT

直接计算 DFT 的运算量分析

N 点有限长序列 $x(n)$ 的 DFT 变换对的定义为：

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

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

其中 $W_N=e^{-j\frac{2\pi}{N}}$

假设 $x(n)$ 是复序列， 同时 $X(k)$ 一般也是复数。

3.4 FFT

直接计算 DFT 的运算量分析

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

	复数乘法	复数加法
每一个$X(k)$	N	$N-1$
N 个$X(k)$ (N 点 DFT)	N^2	$N(N-1)$

$$(e+jf)+(g+jh)=(e+g)+j(f+h)$$

$$(a+jb)(c+jd)=(ac-bd)+j(ad+cb)$$

	实数乘法	实数加法
一次复乘	4	2
一次复加		2
每一个 $X(k)$	$4N$	$2N+2(N-1)=2(2N-1)$
N个$X(k)$ (N点DFT)	$4N^2$	$2N(2N-1)$

复加的加法次数

复乘的加法次数

3.4 FFT

直接计算 DFT 的运算量分析

如 N=512、1024 和 8192 时，DFT 的乘法运算

$N^2=512^2=2^{18}=262144$（26万次）

$N^2=1024^2=2^{20}=1048576$（105万次）

$N^2=8192^2=2^{26}=67108864$（6千7百万次）

– 对于大 N，在实际中是不能接受的，无法“实时”应用 DFT。

– Cooley 与 Turkey 提出的 FFT 算法，大大减少了计算次数。如 N=512 时，FFT 的乘法次数约为 2000 次，提高了约 128 倍，而且简化随 N 的增加而巨增，因而，用数值方法计算频谱得到实际应用。

3.4 FFT

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

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

可约性 $W_N^{nk}=W_{mN}^{mnk}$ $\quad W_N^{nk}=W_{N/m}^{nk/m}$

$$\downarrow$$

$$e^{-j\frac{2\pi}{mN}mnk} \qquad e^{-j\frac{2\pi}{N}\cdot\frac{N}{2}}=e^{-j\pi}=-1$$

$$\uparrow$$

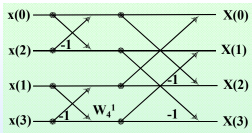
特殊点： $W_N^0=1$ $\quad W_N^{N/2}=-1$ $\quad W_N^{(k+N/2)}=-W_N^k$

3.4 FFT

以 4 点 DFT 为例：

直接计算需要：$4^2=16$ 次复数乘。而按周期性及对称性，可以将 DFT 表示为：

$$\begin{bmatrix}X(0)\\X(1)\\X(2)\\X(3)\end{bmatrix}=\begin{bmatrix}W_4^{0\times0}&W_4^{0\times1}&W_4^{0\times2}&W_4^{0\times3}\\W_4^{1\times0}&W_4^{1\times1}&W_4^{1\times2}&W_4^{1\times3}\\W_4^{2\times0}&W_4^{2\times1}&W_4^{2\times2}&W_4^{2\times3}\\W_4^{3\times0}&W_4^{3\times1}&W_4^{3\times2}&W_4^{3\times3}\end{bmatrix}\begin{bmatrix}x(0)\\x(1)\\x(2)\\x(3)\end{bmatrix}=\begin{bmatrix}1&1&1&1\\1&W_4^1&-1&-W_4^1\\1&-1&1&-1\\1&-W_4^1&-1&W_4^1\end{bmatrix}\begin{bmatrix}x(0)\\x(1)\\x(2)\\x(3)\end{bmatrix}$$



$$X(0)=[x(0)+x(2)]+[x(1)+x(3)]$$
$$X(1)=[x(0)-x(2)]+[x(1)-x(3)]W_4^1$$
$$X(2)=[x(0)+x(2)]-[x(1)+x(3)]$$
$$X(3)=[x(0)-x(2)]-[x(1)-x(3)]W_4^1$$

只需要 1 次复数乘

3.4 FFT

基本思路：

- 虽然存在不同的 FFT 方法，但其核心思想大致相同，即通过迭代，反复利用低点数的 DFT 完成高点数的 DFT 计算，以此达到降低运算量的目的。
- 迭代：利用 W_N^{kn} 的周期性、特殊点和对称性，合并 DFT 计算中很多重复的计算，达到降低运算量的目的。
- 低点数：将傅里叶变换 DFT 分解成相继小的 DFT 计算，即 N 变小（计算量与 N^2 成正比）。

- *FFT* 算法分类:
  - 时间抽选法 DIT: Decimation-In-Time
  - 频率抽选法 DIF: Decimation-In-Frequency

3.4 FFT——基2时域抽选法

设序列点数 $N=2^M$，M 为整数。若不满足，则补零。
N 为 2 的整数幂的 FFT 算法称基-2 FFT算法。

将序列 $x(n)$ 按 n 的奇偶分成两组：

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

一组由偶数序号组成，另一组由奇数序号组成。

3.4 FFT——基2时域抽选法

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

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

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

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

$$=X_1(k)+W_N^kX_2(k)$$

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

3.4 FFT——基2时域抽选法

偶数取样点DFT为：

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

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

奇数取样点DFT为：

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

3.4 FFT——基2时域抽选法

当 $k=\frac{N}{2},\frac{N}{2}+1,\cdots,N-1$ 时

令 $k=\frac{N}{2}+l,\quad l=0,\cdots,\frac{N}{2}-1$

观察

$$X(k)=\overbrace{\underbrace{X_1(k)}_{第一}+\underbrace{W_N^{k}}_{第三}\underbrace{X_2(k)}_{第二}}^{分为三部分}$$

则

$$X_1(k)=X_1(\frac{N}{2}+l)=\sum_{r=0}^{\frac{N}{2}-1}x(2r)W_{\frac{N}{2}}^{r(\frac{N}{2}+l)}=\sum_{r=0}^{\frac{N}{2}-1}x(2r)W_{\frac{N}{2}}^{r\frac{N}{2}}\cdot W_{\frac{N}{2}}^{rl}$$

$$=\sum_{r=0}^{\frac{N}{2}-1}x(2r)W_{\frac{N}{2}}^{rl}=X_1(l)$$

3.4 FFT——基2时域抽选法

同理 $X_2(k)=X_2(\frac{N}{2}+l)=X_2(l)$

而 $W_N^{k}=W_N^{(\frac{N}{2}+l)}=-W_N^{l}\qquad(\because W_N^{\frac{N}{2}}=e^{-j\frac{2\pi}{N}\frac{N}{2}}=e^{-j\pi}=-1)$

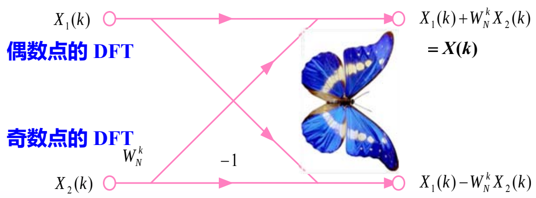
因此

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

因此：整个 $X(k)$ 的计算，可以分解为前、后半部分的运算。而只要求出前一半，就可以由上式求出整个序列。

3.4 FFT——基2时域抽选法

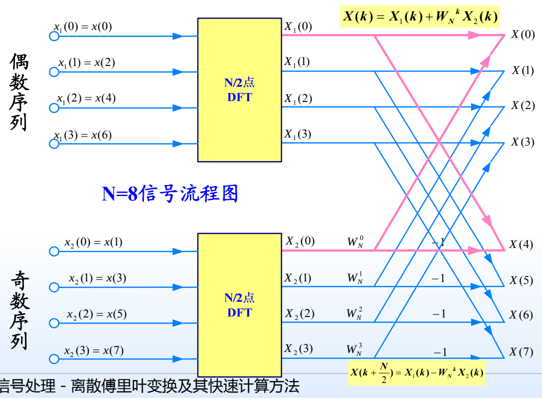
上式表示为信号流程图：



此信号流程图也称为蝶形流程图

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

3.4 FFT——基2时域抽选法



3.4 FFT——基2时域抽选法

$N/2$ 仍为偶数，进一步分解：$N/2\rightarrow N/4$

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

$$x(n)\begin{cases}x(2r)=x_1(r)\begin{cases}x_1(2l)=x_3(l)\{\ \cdots\\ x_1(2l+1)=x_4(l)\{\ \cdots\end{cases}\\ x(2r+1)=x_2(r)\begin{cases}x_2(2l)=x_5(l)\{\ \cdots\\ x_2(2l+1)=x_6(l)\{\ \cdots\end{cases}\end{cases}$$

3.4 FFT——基2时域抽选法

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

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

同理

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

3.4 FFT——基2时域抽选法

由于 $N=2^M$，这样逐级分解，直到 2 点 DFT

当 $N=8$时，即分解到 $X_3(k)$，$X_4(k)$，$X_5(k)$，$X_6(k)$，$k=0,1$

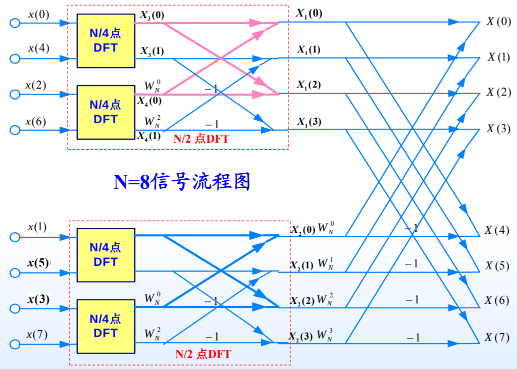
$$X_3(k)=\sum_{l=0}^{N/4-1}x_3(l)W_{N/4}^{lk}=\sum_{l=0}^{1}x_3(l)W_{N/4}^{lk}\qquad k=0,1$$

$$\begin{cases}X_3(0)=x_3(0)W_2^0+W_2^0x_3(1)=x(0)+W_N^0x(4)\\ X_3(1)=x_3(0)W_2^0+W_2^1x_3(1)=x(0)-W_N^0x(4)\end{cases}$$

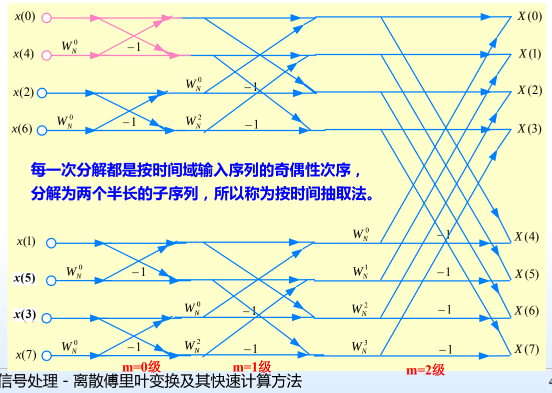
$$X_4(k)=\sum_{l=0}^{N/4-1}x_4(l)W_{N/4}^{lk}=\sum_{l=0}^{1}x_4(l)W_{N/4}^{lk}\qquad k=0,1$$

$$\begin{cases}X_4(0)=x_4(0)W_2^0+W_2^0x_4(1)=x(2)+W_N^0x(6)\\ X_4(1)=x_4(0)W_2^0+W_2^1x_4(1)=x(2)-W_N^0x(6)\end{cases}$$

3.4 FFT——基2时域抽选法



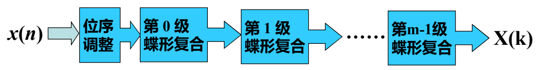
3.4 FFT——基2时域抽选法



3.4 FFT——基2时域抽选法

上式中

$$\begin{bmatrix}W_4^0\\W_4^1\end{bmatrix}=\begin{bmatrix}e^{-j\frac{2\pi}{4}0}\\e^{-j\frac{2\pi}{4}1}\end{bmatrix}=\begin{bmatrix}1\\-j\end{bmatrix}\qquad \begin{bmatrix}W_8^0\\W_8^1\\W_8^2\\W_8^3\end{bmatrix}=\begin{bmatrix}1\\(1-j)/\sqrt{2}\\-j\\-(1+j)/\sqrt{2}\end{bmatrix}$$

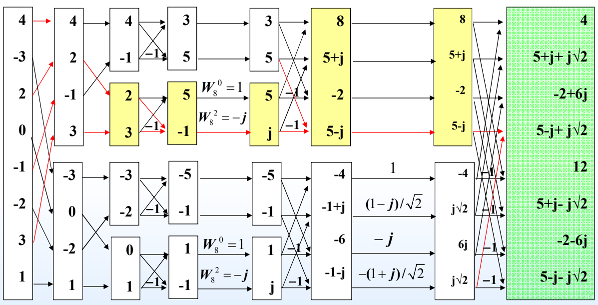


2^m 点 DFT 的基 2 时间抽选计算过程

- 引入FFT出发点：考虑 W 因子的特点和性质，简化算法。
- DIT-FFT算法：时间域输入信号逐级分解为奇偶序列。

3.4 FFT——基2时域抽选法

例 使用基 2 时间抽选法 FFT 流图，
计算下列数据的 8点 DFT：x=[4, -3, 2, 0, -1, -2, 3, 1]

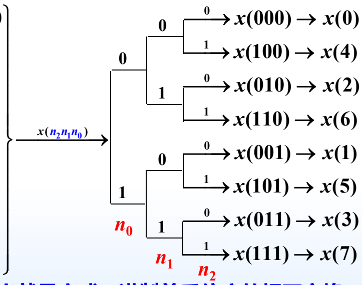


3.4 FFT——基2时域抽选法

输入位序重排：N 点 DFT 分解为两个 $N/2$ 点 DFT→输入序列按奇偶分组→再分解→再奇偶重排→直到 2 点DFT。

十进制 → $x(n_2n_1n_0)$

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



输入序列重排实际上就是完成二进制前后位序的相互交换

$$x(n_2n_1n_0)\leftrightarrow x(n_0n_1n_2)$$

3.4 FFT——基2时域抽选法

当 $N = 2^M$ 时，共有 $M=\log_2 N$ 级蝶形；每级 $N/2$ 个蝶形；每个蝶形有 1 次复数乘法，2 次复数加法。

复数乘法：

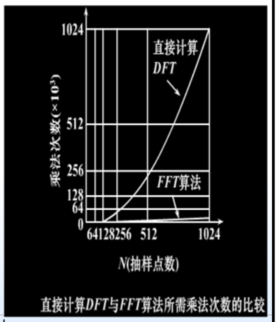
$$m_F = 1\times\frac{N}{2}\times M = \frac{N}{2}\log_2 N$$

复数加法：

$$a_F = 2\times\frac{N}{2}\times M = N\log_2 N$$

比较 DFT/FFT

$$\frac{m_F(DFT)}{m_F(FFT)} = \frac{N^2}{\frac{N}{2}\log_2 N} = \frac{2N}{\log_2 N}$$



3.4 FFT——基2时域抽选法

- FFT 算法
  - 上面讨论的 FFT 算法假定 N 为 2 的整数次幂，即 $N = 2^M$，是基2 FFT 算法。
  - 当 $N=R^v$ 时，这样的算法叫做基R FFT算法，而当 $N=R_1^{v1}R_2^{v2}R_3^{v3}$ 时，叫做混合基算法。
  - 当 N 是一个高度合数时，可得到最有效的算法。最受欢迎也最易编程的算法是基2 FFT 算法。
  - 对于不能分解的质数，或者当 N 不是 2 的整数次幂时，可以在信号的末尾补0，使其成为高度合数或 2 的整数次幂。
- Matlab FFT 实现
  - Matlab 提供了内建的 X = fft(x, *N*) 函数来计算矢量 x 的 DFT。
  - fft 函数是机器码写成的，而不是以 Matlab 指令写成的，即不存在 .m 文件。因此，它的执行速度很快。
  - 采用混合算法。
    - 若 N 为 2 的幂，则得到高速的基 2 FFT 算法；
    - 若 N 不是 2 的幂，则将 N分解成质数，得到较慢的混合基 FFT；
    - 若 N 为质数，则 fft 函数采用原始 DFT 算法。

3.4 FFT——基2时域抽选法

- 如果输入序列 x 的长度小于 N，则填零使其成为 N 点序列，如果省略变量 N，则 DFT 的长度即为 x 的长度。如果 x 为一个矩阵，则计算 x 中每列的 N 点 DFT。
- IDFT 由 ifft 函数完成，它的特征与 fft 函数相同。

例 研究当 $1\le n\le 2048$ 时，fft 函数的执行时间，这将展示不同的 N的情形下，划分 - 组合的策略。

Matlab 提供了两个函数来确定执行时间。clock函数读取瞬时时钟，etime(t1,t2) 函数计算时刻 t1、t2 之间所经历的时间。为了确定执行时间，产生长度为 1 至 2048 的随机矢量，计算它们的 FFT，将计算时间存在一个数组里。最后画出执行时间相对于 N 的图。

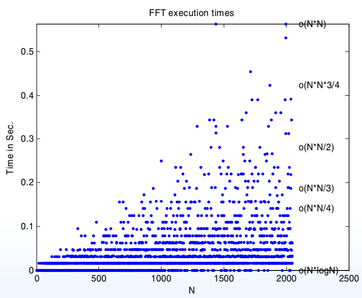
3.4 FFT——基2时域抽选法

```
% Computational Complexity of FFT using MATLAB
N = 2048;
fft_time=zeros(1,N);

for n=1:N
  x=rand(1,n);%Uniformly distributed random numbers.
  t=clock;
  fft(x);
  fft_time(n)=etime(clock,t); % 计算时间
end

n=[1:1:Nmax];top = max(fft_time);
plot(n,fft_time,'.');axis([0,2500,0,top]);
xlabel('N');ylabel('Time in Sec.');title('FFT execution times')
text(2100,top,'o(N*N)')
text(2100,top*3/4,'o(N*N*3/4')
text(2100,top/2,'o(N*N/2)')
text(2100,top/3,'o(N*N/3)')
text(2100,top/4,'o(N*N/4)')
text(2100,min(fft_time),'o(N*logN)')
```


3.4 FFT——基2时域抽选法



- 从图中可以看到，图中曲线分成几组，表示了 N 的不同组合情况。
- 当 N 高度可合时，划分-组合策略是很有效的。例如 N = 2048 时，执行时间约为 0 秒，N = 2047 时为 0.05 秒，N = 2029 时为 0.22 秒。
- 在实际中一般选择是 $N = 2^M$。当信号的取样点数不是 2 的整数次幂时，可以在信号的末尾补 0，外加的 0 不会影响信号的 DTFT 或 DFT 特性。

习题

3.24

3.26（用8点基2 DIT。要求：公式推导，画出流图，算出每列每点处的数值）

3.28 （提示：用两个实序列构成一个复序列，再利用DFT的性质之一）

习题下周交

实验三：基4-FFT算法编程

实验要求请到DSP公邮下载zju_dsp@163.com

密码：dsp_zju

可交纸质版或PDF电子版发送到：3130103370@zju.edu.cn

实验11月6日交