



数字信号处理

Digital Signal Processing

第 3 章 离散傅里叶变换及其快速计算方法

提纲:

3.1 问题的提出

3.2 DFS及其性质（离散傅里叶级数）

3.3 DFT及其性质（有限离散傅里叶变换）

3.4 FFT（快速离散傅里叶变换）

3.5 FFT的应用

3.6 DFT相关变换

3.7 本章小结

3.1 问题的提出：连续信号的傅里叶变换

❖ 数字系统特征：

- 无法处理模拟量 \Rightarrow 采样
- 存储能力有限

❖ 连续信号 $x_a(t)$ ，其傅里叶变换为

$$X_a(\Omega) = \int_{-\infty}^{\infty} x_a(t) e^{-j\Omega t} dt$$

$$x_a(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_a(\Omega) e^{j\Omega t} d\Omega$$

$x_a(t)$ 为时域连续信号。

$X_a(\Omega)$ 为频域连续信号。

3.1 问题的提出：离散信号的变换

❖ 离散信号在两种变换域中的表示方法：

1) 离散时间傅里叶变换 DTFT —— 提供了绝对可加的离散时间序列在频域 (ω) 中的表示方法。

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

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{jn\omega} d\omega$$

2) Z 变换 —— 提供任意序列的 z 域表示。

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

$$x(n) = \frac{1}{2\pi j} \oint_c X(z) z^{n-1} dz$$

● 这两种变换有两个共同特征：

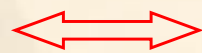
- 1) 变换适合于无限长序列；
- 2) 它们是连续变量 ω 或 z 的函数

3.1 问题的提出：可计算性

- ❖ 问题： $X(z)$, $X(e^{j\omega})$ 都是连续的，利用计算机处理有困难，例如使用 **Matlab**，因此
 - 提出了在频域内取样，使频谱离散化的问题；
 - 必须截断序列，得到有限个点的序列。
- ❖ 目标： 我们需要得到一个**可进行数值计算**的变换
- ❖ 方法：
 - 1) DTFT \rightarrow 频域中原始信号频谱的周期拓展(频域卷积得到)
 - 2) 对 DTFT 在频域中采样 \rightarrow DFS。
 - 3) 将 DFS 推广到有限持续时间序列 \rightarrow DFT。
(DFT 避免了前面提到的那两个问题，并且它是计算机可实现的变换方式。)
- ❖ DFT 已成为 DSP 算法中的核心变换，**原因**：
 - 1) 成为有限长序列傅里叶变换的重要方法。
 - 2) 有快速算法。 \Rightarrow FFT \Rightarrow DCT, DST

3.1 问题的提出：傅里叶变换的四种形式

时间函数

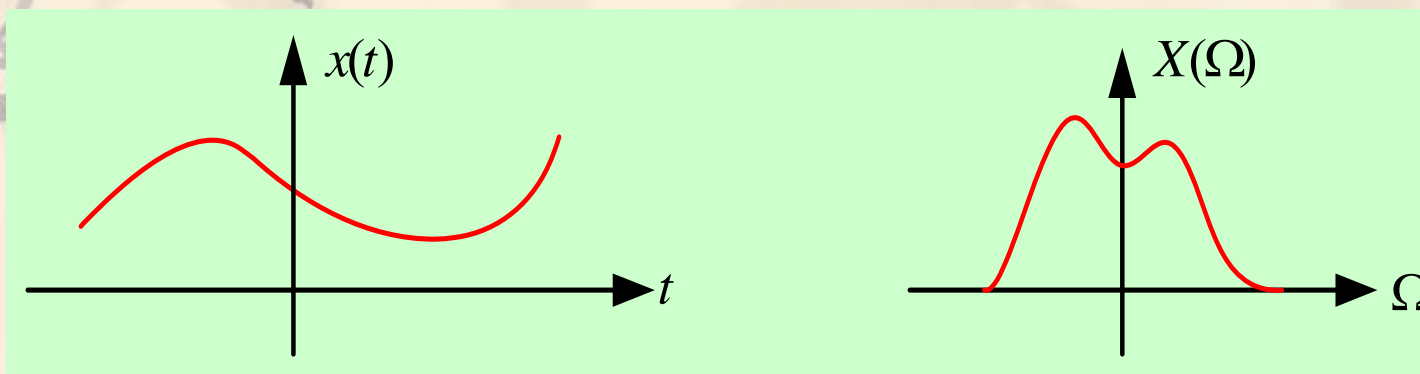


频率函数

- ❖ 非周期连续时间 — 傅里叶变换 (FT) — 连续频率
- ❖ 周期连续时间 — 傅里叶级数 (FS) — 离散频率
- ❖ 非周期离散时间 — 离散时间傅里叶变换 (DTFT) — 连续频率
- ❖ 周期离散时间 — 离散傅里叶级数 (DFS) — 离散频率

3.1 问题的提出： 傅里叶变换的四种形式

1. 非周期连续时间信号： 傅里叶变换 FT



$$x(t) \leftrightarrow X(\Omega), \quad t \leftrightarrow \Omega$$

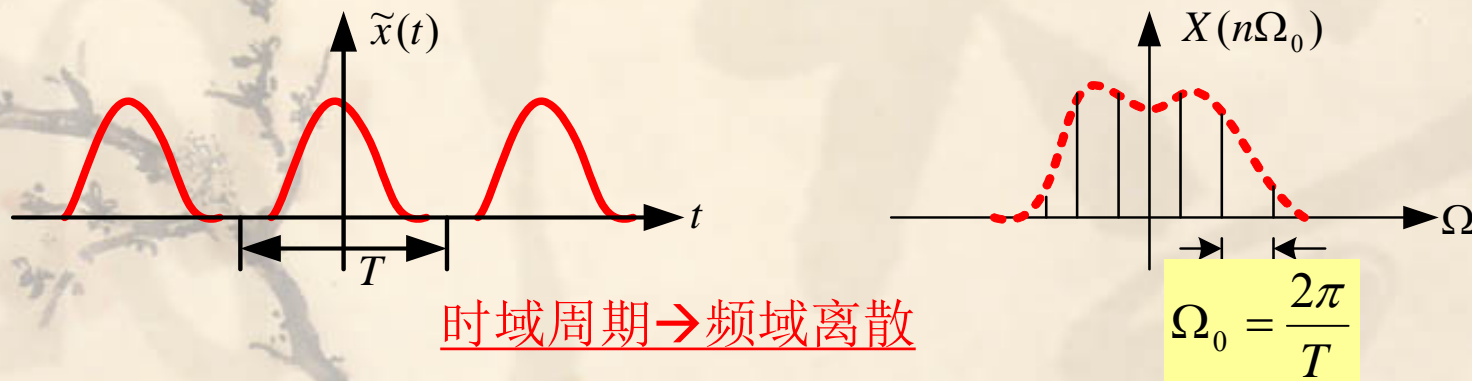
$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\Omega) e^{j\Omega t} d\Omega$$

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

- 时域连续函数造成频域是非周期的谱。
- 时域的非周期造成频域是连续的谱。

3.1 问题的提出：傅里叶变换的四种形式

2. 周期连续时间信号：傅里叶级数 FS



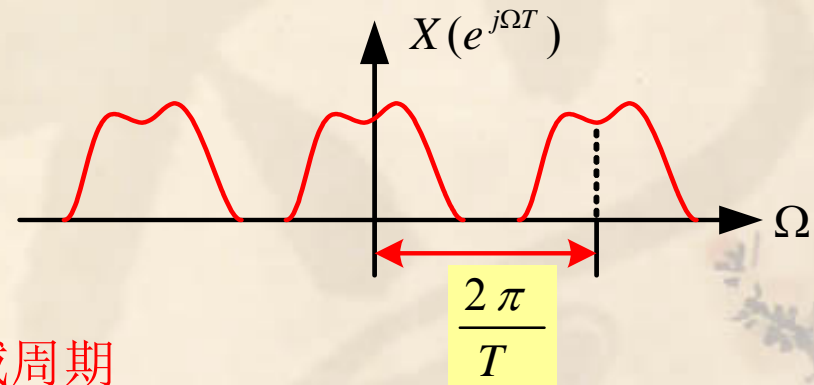
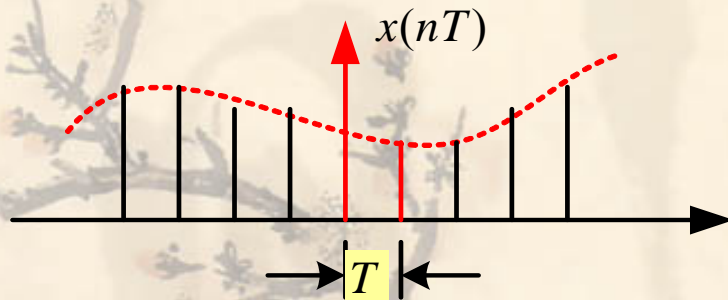
$$\tilde{x}(t) = \sum_{n=-\infty}^{\infty} X(n\Omega_0) e^{jn\Omega_0 t}$$

$$X(n\Omega_0) = \frac{1}{T} \int_{-T/2}^{T/2} \tilde{x}(t) e^{-jn\Omega_0 t} dt$$

时域连续函数造成频域是非周期的谱。
时域是周期函数对应频域的离散化。

3.1 问题的提出：傅里叶变换的四种形式

3. 非周期离散信号：离散时间傅里叶变换 DTFT



时域离散 \rightarrow 频域周期

$$x(nT) = \frac{T}{2\pi} \int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} X(e^{j\Omega T}) e^{jn\Omega T} d\Omega$$

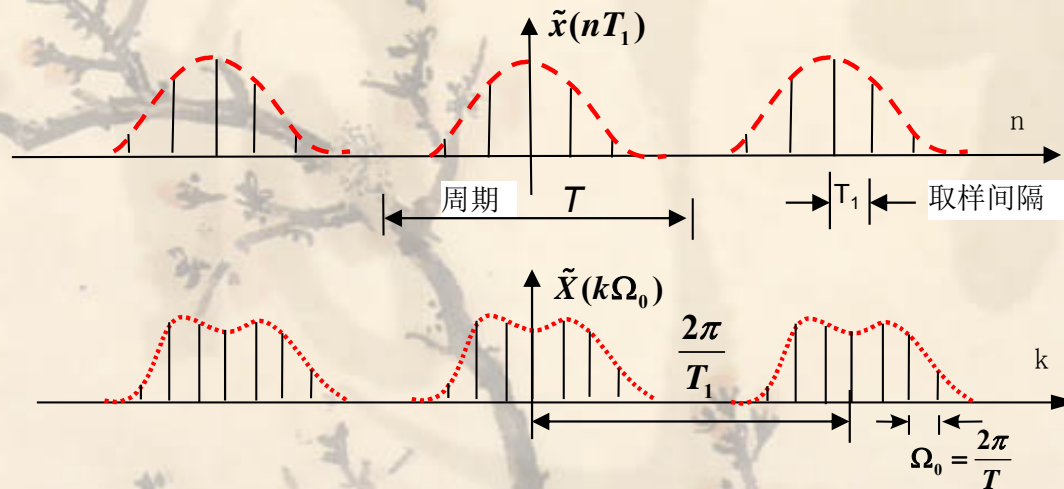
取样定理

$$\tilde{X}(e^{j\Omega T}) = \sum_{n=-\infty}^{\infty} x(nT) e^{-jn\Omega T} = \frac{1}{T} \sum_{n=-\infty}^{\infty} X(\Omega - \Omega_0)$$

时域的离散化造成频域的周期延拓
时域的非周期对应于频域的连续

3.1 问题的提出：傅里叶变换的四种形式

4. 周期离散时间信号：离散傅里叶级数 DFS



时域周期、离散 \rightarrow 频域周期、离散

$$N = \frac{T}{T_1} \Rightarrow T = NT_1$$

$$\Omega_0 \leq \frac{2\pi}{T} = \frac{2\pi}{NT_1}$$

$$N = \frac{2\pi}{\Omega_0 T_1}$$

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

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

一个域的离散造成另一个域的周期延拓

离散傅里叶级数的时域和频域都是离散的和周期的

3.1 问题的提出：傅里叶变换的四种形式

■ 四种傅里叶变换形式的归纳总结：

形式	时间函数	频率函数
傅里叶变换 FT	连续 非周期	非周期 连续
傅里叶级数 FS	连续 周期(T_0)	非周期 离散($\Omega_0=2\pi/T_0$)
离散时间傅里叶变换 DTFT	离散(T) 非周期	周期($\Omega_s=2\pi/T$) 连续
离散傅里叶级数 DFS	离散(T) 周期(T_0)	周期($\Omega_s=2\pi/T$) 离散($\Omega_0=2\pi/T_0$)

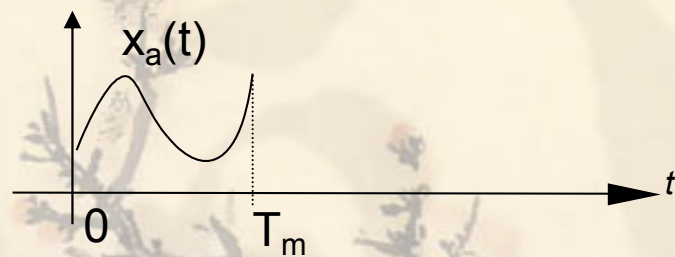
结论：

- ① 时域中函数取样(离散) → (映射) 频域中函数周期重复；
- ② 频域中函数取样(离散) → (映射) 时域中函数周期重复；
- ③ 取样间隔 → (映射) 周期 (2 π / 间隔)

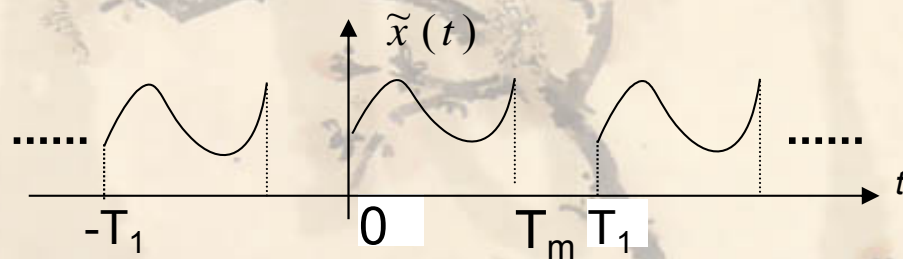
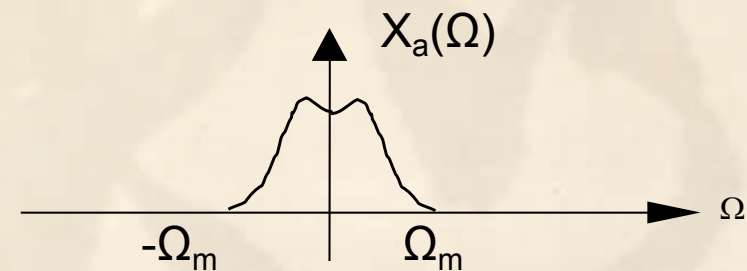
● 离散时间函数的取样间隔： T_1 ，取样频率： $f_s = \frac{\Omega_s}{2\pi} = \frac{1}{T_1}$

● 离散频率函数的取样间隔： F_0 ，时间周期： $T_0 = \frac{1}{F_0} = \frac{2\pi}{\Omega_0}$

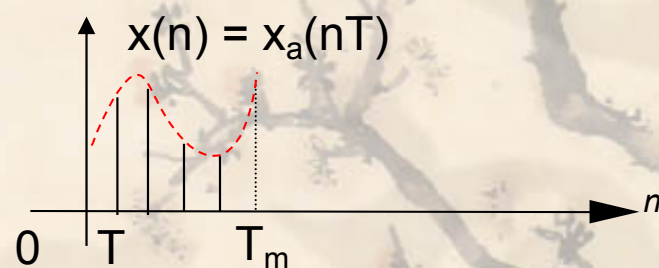
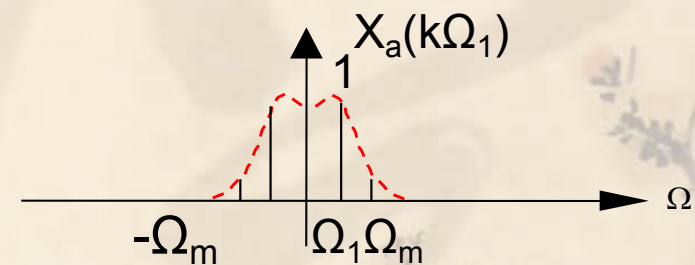
3.1 问题的提出：傅里叶变换的四种形式



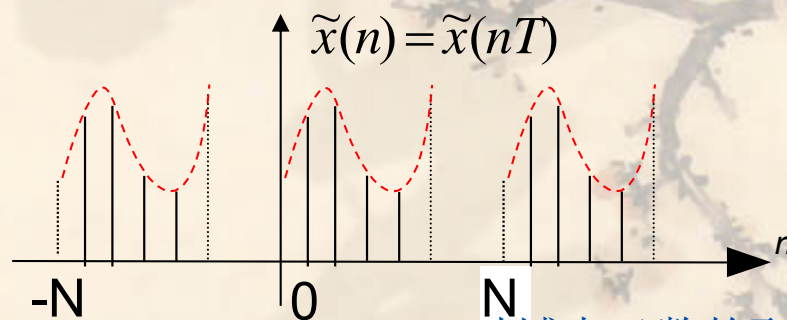
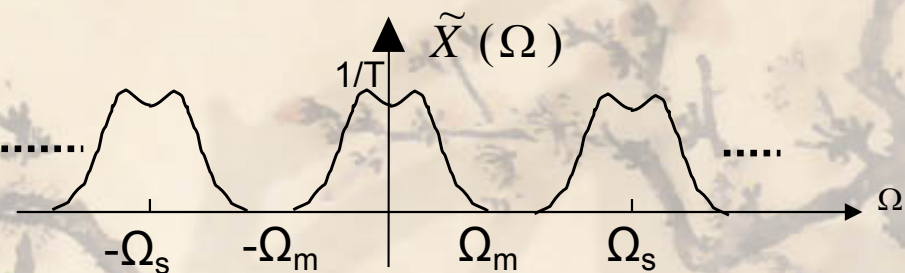
(a) FT



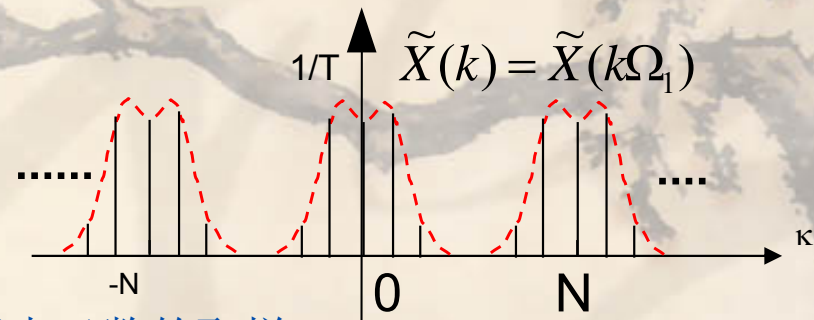
(c) FS



(b) DTFT



(d) DFS



提纲:

3.1 问题的提出

3.2 DFS (离散傅里叶级数)

3.3 DFT (有限离散傅里叶变换)

3.4 FFT (快速离散傅里叶变换)

3.5 CZT及其快速算法

3.6 其它变换

3.7 本章小结

3.2 DFS 及其性质

- 由以上讨论可以清楚地看到，时域取样将引起频域的周期延拓，频域取样也将引起时域的周期延拓。
- 因此可以设想，如果同时对频域和时域取样，其结果是时域和频域的波形都变成离散、周期性的波形，从而我们可以利用付氏级数这一工具，得到它们之间的离散付氏级数 DFS 关系。

3.2.1 DFS 定义：正变换

❖ 令 $\tilde{x}(n)$ 为一周期序列，

$$\tilde{x}(n) = \tilde{x}(n - rN), \quad r \text{ 为整数}$$

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

$$\sum_{n=-\infty}^{\infty} e^{-j\omega n} = 2\pi \sum_{k=-\infty}^{\infty} \delta(\omega - 2\pi k)$$

DFS

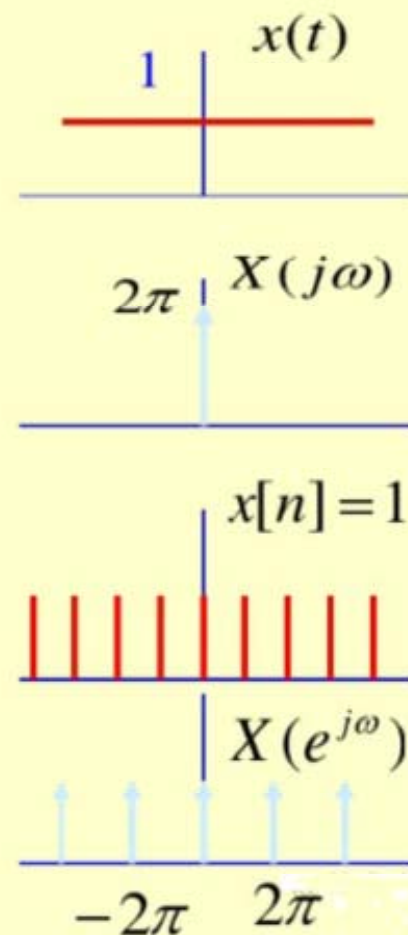
$$\begin{aligned} \text{DTFT}\left\{\sum_{m=-\infty}^{\infty} \delta(n - mN)\right\} &= \sum_{m=-\infty}^{\infty} e^{-j\omega mN} = 2\pi \sum_{k=-\infty}^{\infty} \delta(N\omega - 2\pi k) \\ &= \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi k}{N}\right) \end{aligned}$$

在连续时间傅里叶变换中，引进冲激函数关系式：

$$\int_{-\infty}^{+\infty} e^{-j\omega t} dt = 2\pi\delta(\omega)$$

在离散时间傅里叶变换中，引进冲激序列关系式为：

$$\sum_{n=-\infty}^{+\infty} e^{-j\omega n} = \sum_{l=-\infty}^{+\infty} 2\pi\delta(\omega - 2\pi l)$$



$$\tilde{x}(n) = \sum_{r=-\infty}^{\infty} x(n-rN) = x(n) * \sum_{r=-\infty}^{\infty} \delta(n-rN)$$

$$\begin{aligned} \text{DTFT}[\tilde{x}(n)] &= X(e^{j\omega}) \times \left[\frac{2\pi}{N} \sum_{k=-\infty}^{\infty} \delta(\omega - \frac{2\pi}{N}k) \right] \\ &= \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} X(e^{j\omega}) \delta(\omega - \frac{2\pi}{N}k) \\ &= \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} (X(e^{j\omega})|_{\omega=\frac{2\pi}{N}k}) \delta(\omega - \frac{2\pi}{N}k) \end{aligned}$$

$$X(e^{j\omega}) = \text{DTFT}\{x(n)\} \qquad \tilde{X}(k) = X(e^{j\omega})|_{\omega=\frac{2\pi}{N}k}$$

$$\tilde{x}(n) = \text{IDTFT}[X(e^{j\omega})]$$

DTFT

$$= \frac{1}{2\pi} \int_0^{2\pi} \left[\frac{2\pi}{N} \sum_{k=-\infty}^{\infty} \tilde{X}(k) \delta(\omega - \frac{2\pi}{N}k) \right] e^{j\omega n} d\omega$$

$$= \frac{1}{N} \int_0^{2\pi} \left[\sum_{k=0}^{N-1} \tilde{X}(k) \delta(\omega - \frac{2\pi}{N}k) \right] e^{j\omega n} d\omega$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) e^{jn \frac{2\pi}{N}k} \int_0^{2\pi} \delta(\omega - \frac{2\pi}{N}k) d\omega$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) e^{jn \frac{2\pi}{N}k}$$

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

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

$\{W_N^{-nk}, k = 0, 1, \dots, N-1\}$ 为一完备的离散正交函数系，即满足如下性质：

$$\frac{1}{N} \sum_{k=0}^{N-1} W_N^{-nk_1} W_N^{nk_2} = \begin{cases} 1 & k_1 = k_2 \\ 0 & k_1 \neq k_2 \end{cases}$$

$$\begin{aligned} \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{nk} &= \sum_{n=0}^{N-1} \left(\frac{1}{N} \sum_{l=0}^{N-1} \tilde{X}(l) W_N^{-nl} \right) W_N^{nk} \\ &= \sum_{l=0}^{N-1} \tilde{X}(l) \left(\frac{1}{N} \sum_{n=0}^{N-1} W_N^{-n(l-k)} \right) \\ &= \sum_{l=0}^{N-1} \tilde{X}(l) \delta(l-k) = \tilde{X}(k) \end{aligned}$$

DFS

$$\begin{cases} \tilde{X}(k) = \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{kn} \\ \tilde{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) W_N^{-kn} \end{cases}$$

3.2.1 DFS 定义：几点说明

$$X(z) \big|_{z=e^{j\omega}} = X(e^{j\omega}) = \sum_{n=0}^{N-1} \tilde{x}(n)e^{-jn\omega}$$

$$X(e^{j\omega}) \big|_{\omega=\frac{2\pi}{N}k} = X(e^{j\frac{2\pi}{N}k}) = \tilde{X}(k)$$

则

$$\begin{aligned}\tilde{X}(k) &= \sum_{n=0}^{N-1} \tilde{x}(n)e^{-jn(\frac{2\pi}{N}k)} = \sum_{n=0}^{N-1} \tilde{x}(n)e^{-j\frac{2\pi}{N}kn} \\ &= \sum_{n=0}^{N-1} \tilde{x}(n)W_N^{kn} \quad \text{其中} \quad W_N = e^{-j\frac{2\pi}{N}}\end{aligned}$$

3.2.1 DFS 定义：几点说明

❖ 在什么条件下不产生混迭失真？

$$\tilde{X}(k) = X(e^{j\omega}) \Big|_{\omega = \frac{2\pi}{N}k}$$

—频率取样

- 频率取样：若时间信号有限长，当满足下列条件时， $X(e^{j\omega})$ 的样本值 $X(k)$ 能不失真的恢复成原信号。
- 为了避免时间上的混迭：
 - (1) 必须是时间限制（有限时宽）

$$x(n) = \begin{cases} \tilde{x}(n), & 0 \leq n \leq N-1 \\ 0, & \text{其它} \end{cases}$$

(2) 取样频率间隔小 $\omega_0 \leq \frac{2\pi}{N}$ 或 $\Omega_0 \leq \frac{2\pi}{NT_1}$

3.2.1 DFS 定义：几点说明

■ 周期信号的频谱

❖ 由傅里叶系数 $\tilde{X}(k)$ 可得到 $\tilde{x}(n)$ 的幅度频谱 $|\tilde{X}(k)|$ 和相位频谱 $\arg(\tilde{X}(k))$ ，不难证明，如果 $\tilde{x}(n)$ 是实序列，那么**幅度频谱是周期性偶函数，相位频谱是周期性奇函数**。

❖ **周期**信号由离散傅里叶级数 **DFS** 得到的频谱，和非**周期**信号由离散时间傅里叶变换 **DTFT** 得到的频谱之间**有重要区别**。

① **DTFT** 产生**连续**频谱，这意味着频谱在所有的频率处都有值，因而非周期信号的幅度和相位频谱是光滑无间断的曲线。

② 与之相反，**DFS** 仅有 **N** 点的频谱，仅包含有限个频率，因而周期信号的幅度和相位频谱是**线谱**，即相等间隔的竖线，当频谱的横坐标变量用实际频率 **f** 代替 **k** 时，谱线间隔为 **f_s/N** 。并不是所有的周期信号都含有全部谐波，例如有些频谱只有奇次谐波，比如三角波，偶次谐波为**0**，而有些频谱仅在一些谐波处的值为**0**。

3.2.1 DFS 定义：Matlab 实现

- ❖ 由 **DFS** 的定义可以看出它是一种可数值计算表示式，它可由多种方式实现。

1) 利用循环语句 **for.....end** 实现

- 为了计算每个样本，可用 **for end** 语句实现求和。为了计算所有的 **DFS** 系数，需要另外一个 **for.....end** 循环，这将导致运行嵌套的两个 **forend** 循环。显然，这种方法的效率较低。

2) 利用矩阵——矢量乘法

- **DFS** 正变换的定义为

$$\tilde{X}(k) = \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{kn} \quad k, n = 0, 1, \dots, N-1$$

- 把具体的 n, k 值代入上述定义，展开得下列方程组：

3.2.1 DFS 定义: Matlab 实现

$$n \quad 0 \cdots \rightarrow (N-1)$$

k

0

\vdots

\downarrow

$(N-1)$

$$\tilde{X}(0) = \sum_{n=0}^{N-1} \tilde{x}(n)W_N^{0 \cdot n} = \tilde{x}(0)W_N^0 + \tilde{x}(1)W_N^0 + \cdots + \tilde{x}(N-1)W_N^0$$

$$\tilde{X}(1) = \sum_{n=0}^{N-1} \tilde{x}(n)W_N^{1 \cdot n} = \tilde{x}(0)W_N^0 + \tilde{x}(1)W_N^1 + \cdots + \tilde{x}(N-1)W_N^{(N-1)}$$

$$\tilde{X}(2) = \sum_{n=0}^{N-1} \tilde{x}(n)W_N^{2 \cdot n} = \tilde{x}(0)W_N^0 + \tilde{x}(1)W_N^2 + \cdots + \tilde{x}(N-1)W_N^{2(N-1)}$$

.....

$$\tilde{X}(N-1) = \sum_{n=0}^{N-1} \tilde{x}(n)W_N^{(N-1) \cdot n} = \tilde{x}(0)W_N^0 + \tilde{x}(1)W_N^{(N-1)} + \cdots + \tilde{x}(N-1)W_N^{(N-1)^2}$$

表示为
矩阵形式:

$$\begin{bmatrix} \tilde{X}(0) \\ \tilde{X}(1) \\ \tilde{X}(2) \\ \vdots \\ \tilde{X}(N-1) \end{bmatrix} = \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & W_N^2 & \cdots & W_N^{(N-1)} \\ W_N^0 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} \tilde{x}(0) \\ \tilde{x}(1) \\ \tilde{x}(2) \\ \vdots \\ \tilde{x}(N-1) \end{bmatrix}$$

3.2.1 DFS 定义: Matlab 实现

- 若令列向量

$$\tilde{x} = (\tilde{x}(0), \tilde{x}(1), \dots, \tilde{x}(N-1))^T$$

$$\tilde{X} = (\tilde{X}(0), \tilde{X}(1), \dots, \tilde{X}(N-1))^T$$

- 则可由矩阵运算的形式给出 **DFS** 的定义为:

$$\begin{aligned}\tilde{X} &= W_N \tilde{x} \\ \tilde{x} &= \frac{1}{N} W_N^* \tilde{X}\end{aligned}$$

$$W_N \stackrel{\Delta}{=} [W_N^{kn} \quad 0 \leq k, n \leq N-1]$$
$$= \begin{matrix} \xrightarrow{n} \\ \downarrow k \end{matrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & \cdots & W_N^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & \cdots & W_N^{(N-1)^2} \end{bmatrix}$$

- ✧ 矩阵 W_N 为方阵, 叫做 **DFS** 矩阵。
- ✧ 用下面的 **Matlab** 函数 **dfs** 和 **idfs** 实现 **DFS** 正反变换。

3.2.1 DFS 定义: Matlab 实现

```
function [Xk] = dfs(xn,N)
```

```
% Computes Discrete Fourier Series Coefficients
```

```
% -----
```

```
% [Xk] = dfs(xn,N)
```

```
% Xk = DFS coeff. array over  $0 \leq k \leq N-1$ 
```

```
% xn = One period of periodic signal over  $0 \leq n \leq N-1$ 
```

```
% N = Fundamental period of xn
```

```
%
```

```
n = [0:1:N-1];
```

```
% row vector for n
```

```
k = [0:1:N-1];
```

```
% row vector for k
```

```
WN = exp(-j*2*pi/N);
```

```
% Wn factor
```

```
nk = n'*k;
```

```
% creates a N by N matrix of nk values
```

```
WNnk = WN .^ nk;
```

```
% DFS matrix
```

```
Xk = xn * WNnk;
```

```
% row vector for DFS coefficients
```

$n = [0, 1, 2]$ $k = [0, 1, 2]$

$$n' = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

$$n'k = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} [0, 1, 2] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 2 & 4 \end{bmatrix}$$

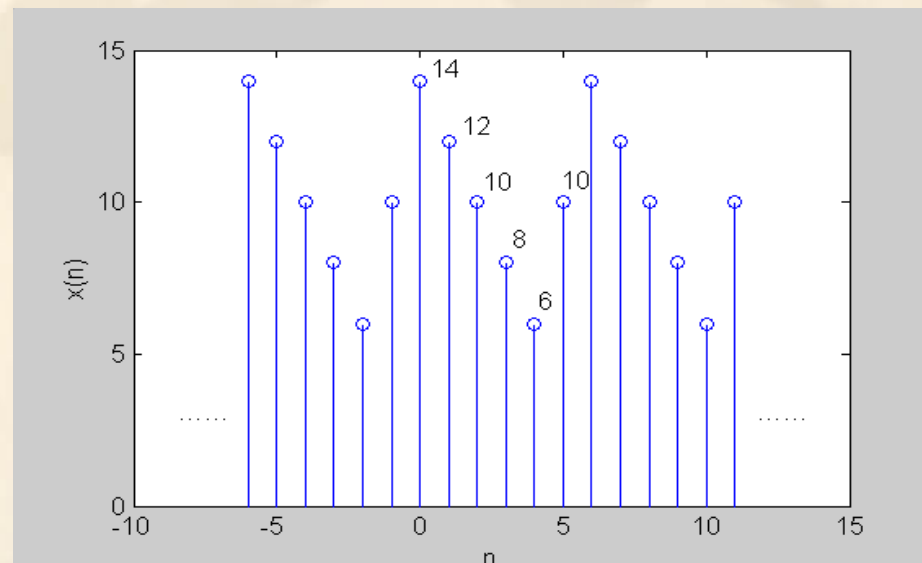
3.2.1 DFS 定义: Matlab 实现

```
function [xn] = idfs(Xk,N)
% Computes Inverse Discrete Fourier Series
% -----
% [xn] = idfs(Xk,N)
% xn = One period of periodic signal over 0 <= n <= N-1
% Xk = DFS coeff. array over 0 <= k <= N-1
% N = Fundamental period of Xk
%
n = [0:1:N-1];           % row vector for n
k = [0:1:N-1];           % row vector for k
WN = exp(-j*2*pi/N);      % Wn factor
nk = n'*k;                % creates a N by N matrix of nk values
WNnk = WN .^ (-nk);       % IDFS matrix
xn = (Xk * WNnk)/N;       % row vector for IDFS values
```

3.2.1 DFS 定义：计算举例

例3.1: 已知序列 $x(n)$ 是周期为 6 的周期序列，如图所示，试求其DFS 的系数

解：根据定义求解



$$\begin{aligned}\tilde{X}(k) &= \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{nk} = \sum_{n=0}^5 \tilde{x}(n) W_6^{nk} \\ &= 14 + 12e^{-j\frac{2\pi}{6}k} + 10e^{-j\frac{2\pi}{6}2k} + 8e^{-j\frac{2\pi}{6}3k} + 6e^{-j\frac{2\pi}{6}4k} + 10e^{-j\frac{2\pi}{6}5k}\end{aligned}$$

$$\begin{aligned}\tilde{X}(0) &= 60 & \tilde{X}(1) &= 9 - j3\sqrt{3} & \tilde{X}(2) &= 3 + j\sqrt{3} \\ \tilde{X}(3) &= 0 & \tilde{X}(4) &= 3 - j\sqrt{3} & \tilde{X}(5) &= 9 + j3\sqrt{3}\end{aligned}$$

3.2.1 DFS 定义：计算举例

例 3.2 求出下面周期序列的 DFS 表示式

$$\tilde{x}(n) = \{\dots, 4, 5, 6, 7, 4, 5, 6, 7, \dots\}$$

解：上述序列的基本周期为 $N=4$ ，因而 $W_4 = e^{-j2\pi/4} = -j$ ，

$$\tilde{X}(k) = \sum_{n=0}^3 \tilde{x}(n) W_4^{nk}$$

因而，

$$\tilde{X}(0) = \sum_{n=0}^3 \tilde{x}(n) W_4^{n \cdot 0} = \sum_{n=0}^3 \tilde{x}(n) = \tilde{x}(0) + \tilde{x}(1) + \tilde{x}(2) + \tilde{x}(3) = 22$$

$$\tilde{X}(1) = \sum_{n=0}^3 \tilde{x}(n) W_4^n = \sum_{n=0}^3 \tilde{x}(n) (-j)^n = 4 - 5j - 6 + 7j = (-2 + 2j)$$

$$\tilde{X}(2) = \sum_{n=0}^3 \tilde{x}(n) W_4^{2n} = \sum_{n=0}^3 \tilde{x}(n) (-j)^{2n} = -2$$

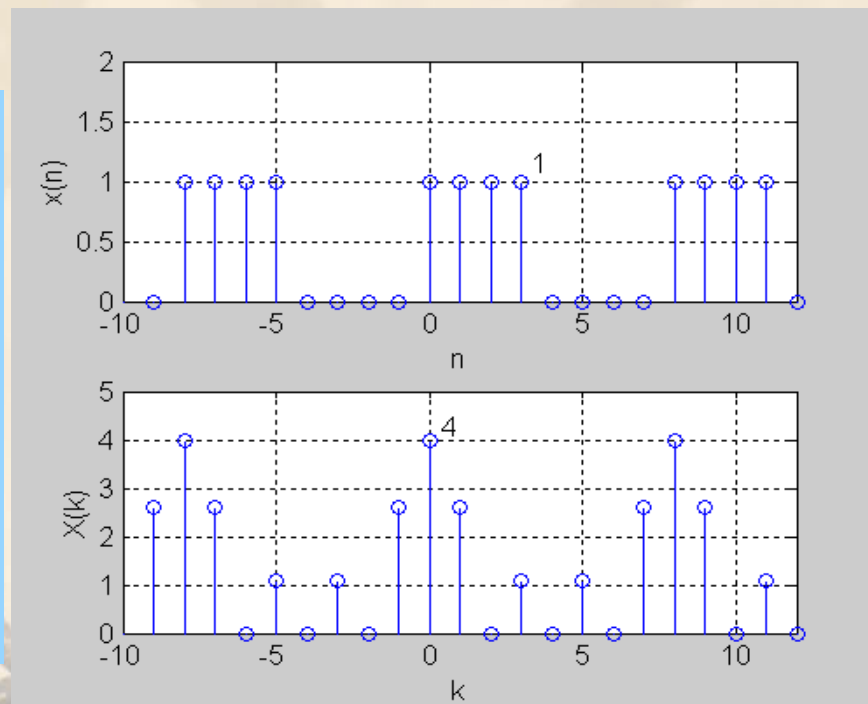
$$\tilde{X}(3) = \sum_{n=0}^3 \tilde{x}(n) W_4^{3n} = \sum_{n=0}^3 \tilde{x}(n) (-j)^{3n} = (-2 - 2j)$$

3.2.1 DFS 定义：计算举例

例3.3： 已知序列 $x(n) = R_4(n)$ ，将 $x(n)$ 以 $N = 8$ 为周期进行周期延拓成 $\tilde{x}(n)$ ，求 $\tilde{x}(n)$ 的 *DFS*。

解法一：数值解

$$\begin{aligned}\tilde{X}(k) &= \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{nk} = \sum_{n=0}^7 \tilde{x}(n) W_8^{nk} \\ &= \sum_{n=0}^3 W_8^{nk} \\ &= 1 + e^{-j\frac{2\pi}{8}k} + e^{-j\frac{2\pi}{8}2k} + e^{-j\frac{2\pi}{8}3k}\end{aligned}$$

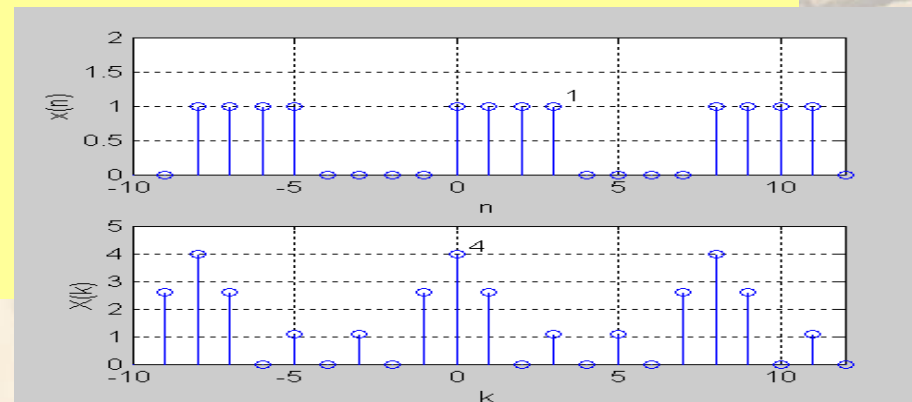


$$\begin{aligned}\tilde{X}(0) &= 4 & \tilde{X}(1) &= 1 - j(\sqrt{2} + 1) & \tilde{X}(2) &= 0 & \tilde{X}(3) &= 1 - j(\sqrt{2} - 1) \\ \tilde{X}(4) &= 0 & \tilde{X}(5) &= 1 + j(\sqrt{2} - 1) & \tilde{X}(6) &= 0 & \tilde{X}(7) &= 1 + j(\sqrt{2} + 1)\end{aligned}$$

3.2.1 DFS 定义：计算举例

解法二：公式解

$$\begin{aligned}\tilde{X}(k) &= DFS[\tilde{x}(n)] = \sum_{n=0}^{N-1} \tilde{x}(n) e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^7 \tilde{x}(n) e^{-j\frac{2\pi}{8}kn} \\&= \sum_{n=0}^3 e^{-j\frac{\pi}{4}kn} = \frac{1 - e^{-j\frac{\pi}{4}k \cdot 4}}{1 - e^{-j\frac{\pi}{4}k}} = \frac{e^{-j\frac{\pi}{2}k} \left(e^{j\frac{\pi}{2}k} - e^{-j\frac{\pi}{2}k} \right)}{e^{-j\frac{\pi}{8}k} \left(e^{j\frac{\pi}{8}k} - e^{-j\frac{\pi}{8}k} \right)} \\&= e^{-j\frac{3}{8}\pi k} \frac{\sin \frac{\pi}{2}k}{\sin \frac{\pi}{8}k}\end{aligned}$$



3.2.1 DFS 定义：计算举例

例 3.4 下面给出一周期“方波”序列：

$$\tilde{x}(n) = \begin{cases} 1, & mN \leq n \leq mN + L - 1 \\ 0, & mN + L \leq n \leq (m+1)N - 1 \end{cases}$$

其中， $m=0, \pm 1, \pm 2, \dots$ ， N 是基本周期， L/N 是占空比。

- a) 确定一种用 L 与 N 描述的 $|\tilde{x}(k)|$ 的表达式。
- b) 分别画出当 $L=5, N=20$; $L=5, N=40$; $L=5, N=60$; $L=7, N=60$ 时 $|\tilde{x}(k)|$ 表达式。
- c) 对所得结果进行讨论。

3.2.1 DFS 定义：计算举例

解：a) 由 DFS 定义可得

$$\begin{aligned}\tilde{X}(k) &= \sum_{n=0}^{N-1} \tilde{x}(n) e^{-j\frac{2\pi}{N}nk} = \sum_{n=0}^{L-1} e^{-j\frac{2\pi}{N}nk} = \sum_{n=0}^{L-1} \left(e^{-j\frac{2\pi}{N}k} \right)^n \\ &= \begin{cases} L, & k = 0, \pm N, \pm 2N, \dots \\ \frac{1 - e^{-j\frac{2\pi}{N}Lk}}{1 - e^{-j\frac{2\pi}{N}k}}, & \text{others} \end{cases}\end{aligned}$$

而：

$$\frac{1 - e^{-j\frac{2\pi L}{N}k}}{1 - e^{-j\frac{2\pi}{N}k}} = \frac{e^{-j\frac{\pi Lk}{N}} e^{j\frac{\pi Lk}{N}} - e^{-j\frac{\pi Lk}{N}}}{e^{-j\frac{\pi k}{N}} e^{j\frac{\pi k}{N}} - e^{-j\frac{\pi k}{N}}} = e^{-j\frac{\pi(L-1)k}{N}} \frac{\sin(\frac{\pi k L}{N})}{\sin(\frac{\pi k}{N})}$$

$\tilde{X}(k)$ 的幅值可表示为：

$$|\tilde{X}(k)| = \begin{cases} L, & k = 0, \pm N, \pm 2N, \dots \\ \left| \frac{\sin(\pi k L / N)}{\sin(\pi k / N)} \right|, & \text{others} \end{cases}$$

3.2.1 DFS 定义：计算举例

b. Matlab 程序如下：

```
% Chapter 3: Example 3.03
```

```
L = 5; N = 20; (改变参数)
```

```
x = [ones(1,L), zeros(1,N-L)];
```

```
xn = x' * ones(1,3);
```

```
xn = (xn(:))';
```

```
n = -N:1:2*N-1;
```

```
subplot(1,1,1); subplot(2,1,2);
```

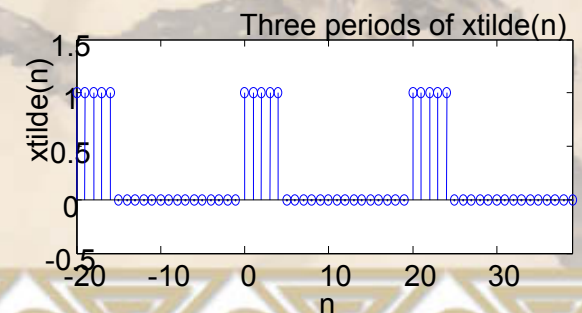
```
stem(n,xn); xlabel('n');
```

```
ylabel('xtilde(n)')
```

```
title('Three periods of xtilde(n)')
```

```
axis([-N,2*N-1,-0.5,1.5])
```

$$\begin{aligned} x &= [1, 2, 3] & x' &= \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \\ xn &= x' \cdot \text{ones}(1,3) = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} [1, 1, 1] = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix} \\ xn(:) &= \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 3 \end{bmatrix} \\ (xn(:))' &= [1 \ 2 \ 3 \ 1 \ 2 \ 3 \ 1 \ 2 \ 3] \end{aligned}$$

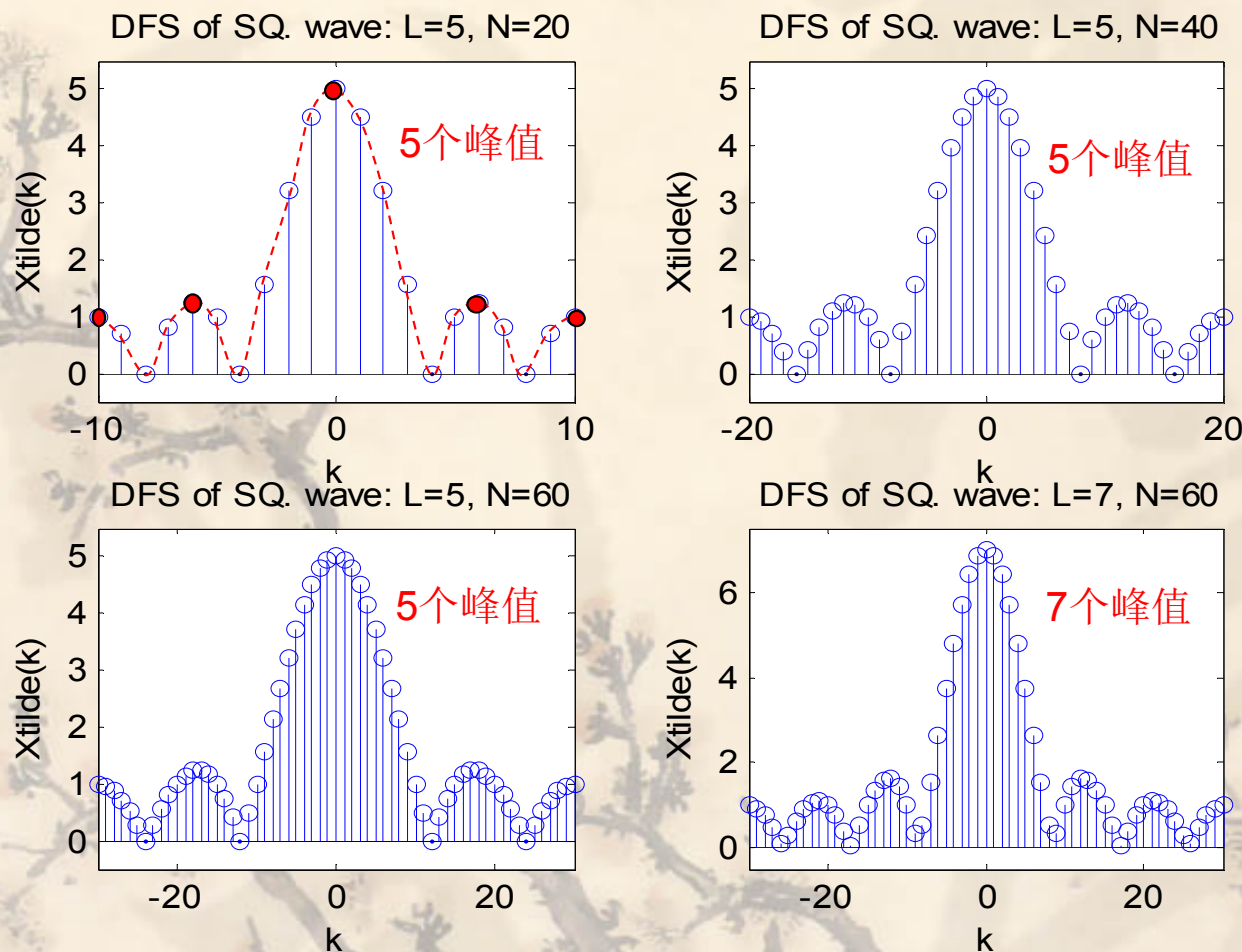


3.2.1 DFS 定义： 计算举例

```
% Part (b)
L = 5; N = 20; (改变参数)
xn = [ones(1,L), zeros(1,N-L)];
Xk = dfs(xn,N);
magXk = abs([Xk(N/2+1:N) Xk(1:N/2+1)]);
k = [-N/2:N/2];

subplot(2,2,1);
stem(k,magXk);
axis([-N/2,N/2,-0.5,5.5])
xlabel('k'); ylabel('Xtilde(k)')
title('DFS of SQ. wave: L=5, N=20')
```

3.2.1 DFS 定义：计算举例



- ❖ 注意: $\tilde{X}(k)$ 是周期信号, 图中只画出了从 $-N/2$ 到 $N/2$ 的部分。
- ❖ c. 从图中可以看到, 方波的 DFS 系数的包络像 “Sinc” 函数, ① $K=0$ 时的幅度等于 L ; ② 同时函数的零点位于 N/L (占空比的倒数) 的整数倍处; ③ $L=5$ 不变, N 变大 (即填0, 但有效信息没有增加), 则形状不变, 只是更平滑, 即获得了一个高密度谱; ④ $N=60$ 不变, L 变大 (即增加了原始数据长度), 则变换后得形状发生了变化, 获得了更多的信息, 即高分辨率谱。

3.2.2 DFS 的性质：线性

❖ 线性：

若：两个周期序列 $\tilde{x}_1(n)$ 和 $\tilde{x}_2(n)$ ，周期均为 N

且：

$$\tilde{X}_1(k) = DFS[\tilde{x}_1(n)]$$

$$\tilde{X}_2(k) = DFS[\tilde{x}_2(n)]$$

则

$$DFS[a\tilde{x}_1(n) + b\tilde{x}_2(n)] = a\tilde{X}_1(k) + b\tilde{X}_2(k)$$

—— a, b 为任意常数

3.2.2 DFS 的性质：序列的周期移位

■ 序列的周期移位（时域）

若 $\tilde{x}(n)$ 是周期序列，其周期为 N ，移位后仍为周期序列，且：

$$DFS[\tilde{x}(n+m)] = W_N^{-mk} \tilde{X}(k) = e^{j\frac{2\pi}{N}mk} \tilde{X}(k)$$

证明：

$$\begin{aligned} DFS[\tilde{x}(n+m)] &= \sum_{n=0}^{N-1} \tilde{x}(n+m) W_N^{nk} && \text{令 } i = n+m \\ &= \sum_{i=m}^{N-1+m} \tilde{x}(i) W_N^{k(i-m)} = W_N^{-mk} \sum_{i=m}^{N-1+m} \tilde{x}(i) W_N^{ki} \\ &= W_N^{-mk} \left(\sum_{i=m}^{N-1} \tilde{x}(i) W_N^{ki} + \sum_{i=N}^{N-1+m} \tilde{x}(i) W_N^{ki} \right) = W_N^{-mk} \left(\sum_{i=m}^{N-1} \tilde{x}(i) W_N^{ki} + \sum_{i=0}^{m-1} \tilde{x}(i) W_N^{ki} \right) \\ &= W_N^{-mk} \sum_{i=0}^{N-1} \tilde{x}(i) W_N^{ki} = W_N^{-mk} \tilde{X}(k) \end{aligned}$$

3.2.2 DFS 的性质：调制特性

■ 调制特性（频域周期移位）

$$DFS[W_N^{nl} \tilde{x}(n)] = \tilde{X}(k + l)$$

证明：

$$\begin{aligned} DFS[W_N^{ln} \tilde{x}(n)] &= \sum_{n=0}^{N-1} W_N^{ln} \tilde{x}(n) W_N^{nk} \\ &= \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{(l+k)n} \\ &= \tilde{X}(k + l) \end{aligned}$$

若： $\tilde{X}_2(k) = \tilde{X}(k + m)$ 则有： $\tilde{x}_2(n) = W_N^{nm} \tilde{x}(n)$

3.2.2 DFS的性质：共轭对称性

❖ 由任一周期性序列 $\tilde{x}(n)$ ，定义如下两个序列：

■ 共轭偶对称周期性序列 $\tilde{x}_e(n)$

$$\tilde{x}_e(n) = \frac{1}{2} [\tilde{x}(n) + \tilde{x}^*(-n)]$$

■ 共轭奇对称周期性序列 $\tilde{x}_o(n)$

$$\tilde{x}_o(n) = \frac{1}{2} [\tilde{x}(n) - \tilde{x}^*(-n)]$$

显然， $\tilde{x}(n)$ 、 $\tilde{x}_e(n)$ 和 $\tilde{x}_o(n)$ 具有相同的周期， $\tilde{x}(n)$ 则可表示为 $\tilde{x}_e(n)$ 与 $\tilde{x}_o(n)$ 之和

$$\tilde{x}(n) = \tilde{x}_e(n) + \tilde{x}_o(n)$$

且具有如下关系：

$$\begin{aligned}\tilde{x}_e(-n) &= \tilde{x}_e^*(n) \\ \tilde{x}_o(-n) &= -\tilde{x}_o^*(n)\end{aligned}$$

其它对称性质见教科书

3.2.2 DFS 的性质：周期卷积

■ 周期卷积（时域）

设：两个周期序列 $\tilde{x}_1(n)$ 和 $\tilde{x}_2(n)$ ，周期均为 N ，则周期卷积

$$\begin{aligned}\tilde{y}(n) &= \tilde{x}_1(n) * \tilde{x}_2(n) \\ &= \sum_{m=0}^{N-1} \tilde{x}_1(m) \tilde{x}_2(n-m) = \sum_{m=0}^{N-1} \tilde{x}_2(m) \tilde{x}_1(n-m)\end{aligned}$$

若 $DFS[\tilde{x}_1(n)] = \tilde{X}_1(k)$ $DFS[\tilde{x}_2(n)] = \tilde{X}_2(k)$

则

$$\tilde{y}(n) = \tilde{x}_1(n) * \tilde{x}_2(n) \xleftarrow{IDFS} \tilde{Y}(k) = \tilde{X}_1(k) \cdot \tilde{X}_2(k)$$

频域相乘 \rightarrow 时域卷积

周期卷积：两个周期序列在一个周期上的线性卷积，是一种特殊的卷积计算形式。

3.2.2 DFS的性质：周期卷积

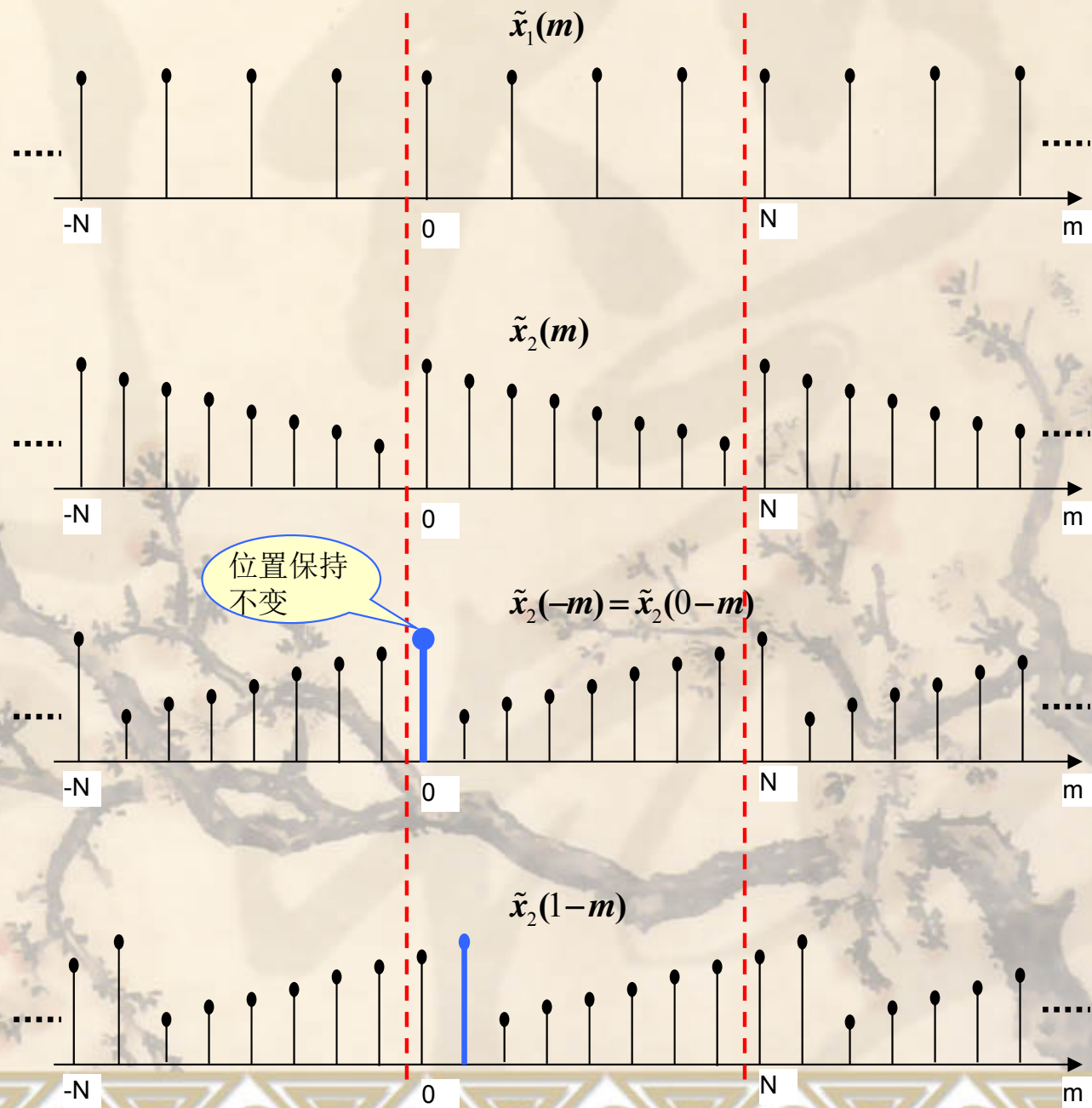
证明：

$$\begin{aligned}\tilde{y}(n) &= IDFS[\tilde{X}_1(k) \cdot \tilde{X}_2(k)] \\&= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}_1(k) \tilde{X}_2(k) W_N^{-kn} \\&= \frac{1}{N} \sum_{k=0}^{N-1} \left[\sum_{m=0}^{N-1} \tilde{x}_1(m) W_N^{mk} \right] \tilde{X}_2(k) W_N^{-kn} \\&= \sum_{m=0}^{N-1} \tilde{x}_1(m) \left[\frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}_2(k) W_N^{-(n-m)k} \right] \\&= \sum_{m=0}^{N-1} \tilde{x}_1(m) \tilde{x}_2(n-m)\end{aligned}$$

3.2.2 DFS 的性质：周期卷积

- (1) $x_1(n)$ 和 $x_2(n)$ 是周期的;
- (2) 求和范围为一个周期;
- (3) 周期序列周期卷积后, 序列的长度仍然是周期的。

$$\tilde{y}(n) = \sum_{m=0}^{N-1} \tilde{x}_1(m) \tilde{x}_2(n-m)$$



3.2.2 DFS 的性质：周期卷积

■ 序列的线性卷积与周期卷积的几点区别：

- 线性卷积的求和对参与卷积的两个序列无任何要求，而周期卷积要求两个序列是周期相同的周期序列。
- 线性卷积的求和范围由两个序列的长度和所在的区间决定，而周期卷积的求和范围是一个周期 N 。
- 线性卷积所得序列的长度 $(M+N-1)$ 由参与卷积的两个序列的长度确定，而周期卷积的结果仍是周期序列，且周期与原来的两个序列周期相同。
- 周期卷积等同于两个周期序列在一个周期上的线性卷积计算。

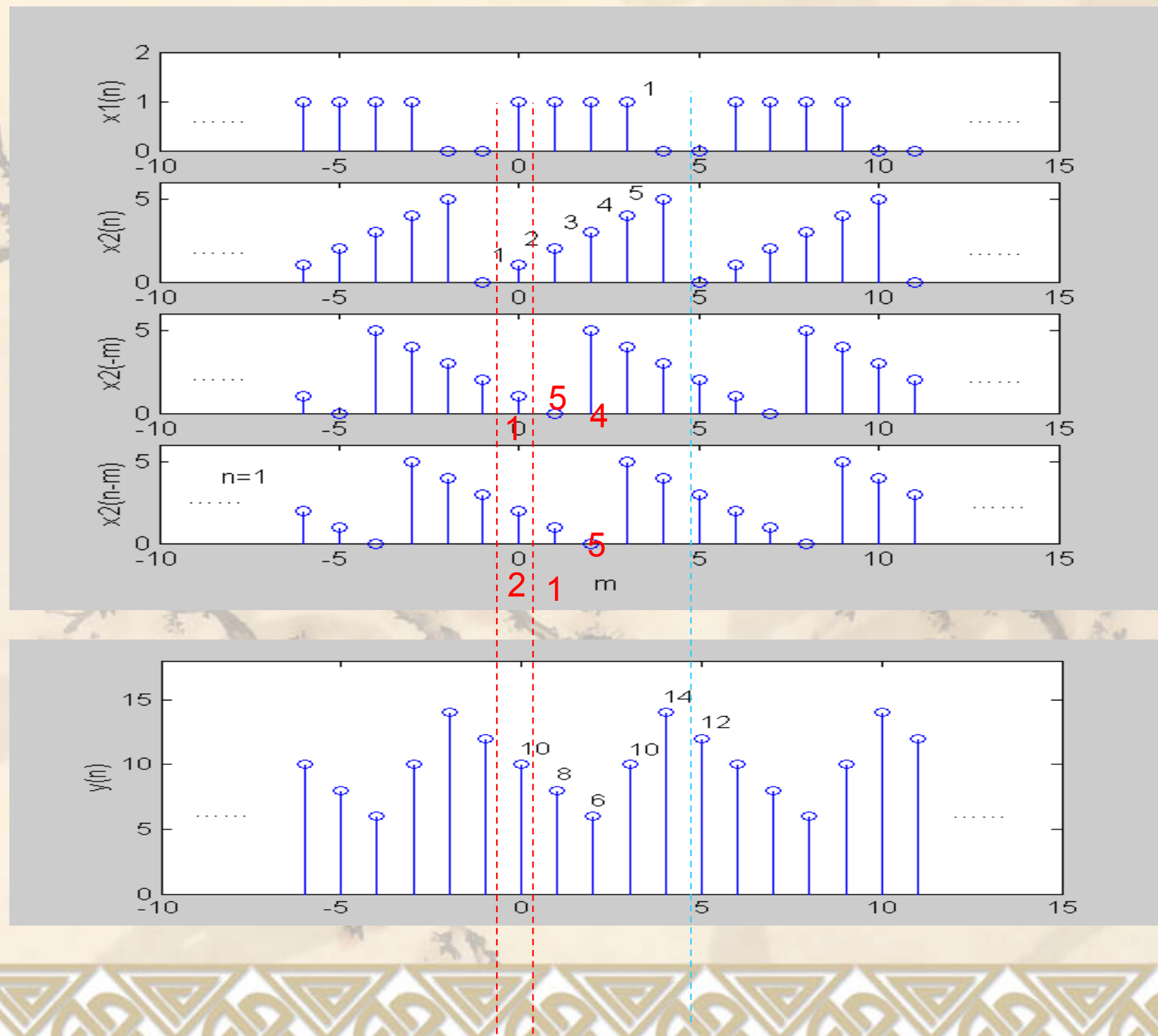
3.2.2 DFS 的性质：周期卷积

例3.6 已知序列 $x_1(n)=R_4(n)$, $x_2(n)=(n+1)R_5(n)$, 分别将序列以周期为 $N=6$ 拓展成周期序列, 求两个周期序列的周期卷积和。

解:

$$\tilde{y}(n) = \sum_{m=0}^{N-1} \tilde{x}_1(m) \tilde{x}_2(n-m) = \sum_{m=0}^5 \tilde{x}_1(m) \tilde{x}_2(n-m)$$

3.2.2 DFS 的性质：周期卷积



3.2.2 DFS 的性质：周期卷积

❖ 频域周期卷积

利用DFS的对偶性有：

若

$$\tilde{y}(n) = \tilde{x}_1(n) \tilde{x}_2(n)$$

则

$$\begin{aligned}\tilde{Y}(k) &= DFS[\tilde{y}(n)] = \sum_{n=0}^{N-1} \tilde{y}(n) W_N^{nk} \\ &= \frac{1}{N} \sum_{l=0}^{N-1} \tilde{X}_1(l) \tilde{X}_2(k-l) \\ &= \frac{1}{N} \sum_{l=0}^{N-1} \tilde{X}_2(l) \tilde{X}_1(k-l)\end{aligned}$$

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

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

注意频域卷积的求和号前面有 $1/N$ 。

时域相乘 → 频域卷积

3.2 DFS 定义和性质：小结

❖ 时域周期序列与频域周期序列的关系 DFS

$$\begin{cases} \tilde{X}(k) = \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{kn} \\ \tilde{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) W_N^{-kn} \end{cases}$$

❖ DFS 的性质

重点：周期移位

$$\text{DFS}[\tilde{x}(n+m)] = W_N^{-mk} \tilde{X}(k) = e^{j\frac{2\pi}{N}mk} \tilde{X}(k)$$

调制特性

$$\text{DFS}[W_N^{nl} \tilde{x}(n)] = \tilde{X}(k+l)$$

周期卷积

$$\tilde{y}(n) = \sum_{m=0}^{N-1} \tilde{x}_1(m) \tilde{x}_2(n-m)$$

提纲:

3.1 问题的提出

3.2 DFS (离散傅里叶级数)

3.3 DFT (有限离散傅里叶变换)

3.4 FFT (快速离散傅里叶变换)

3.5 CZT及其快速算法

3.6 其它变换

3.7 本章小结

3.3 有限离散傅里叶变换

❧ 采样定理:

- ① 时间取样: 取样频率大于信号最高频率两倍;
- ② 频率取样: 取样间隔足够小, 使时间函数的周期 (单位圆上等分 (取样) 的点数) 大于信号的时域长度。

❧ 结果: 频域和时域中均不出现混迭现象。

❧ 离散傅氏级数提供了一种对离散时间傅氏变换作数值计算的技巧, 它在时域和频域都是**周期**的, 但在实际中大多数信号不具有周期性, 它们**很可能具有有限持续时间**。

❧ 对这些信号, 怎样探讨一种可数值计算的傅氏表达式?

❧ 理论上, 可通过**构造一周期信号**, 其基本形状为有限持续时间信号, 然后计算此周期信号的 **DFS**。

❧ **实际上, 这也就是定义了一种新的变换, 称为离散傅氏变换 (DFT), 它是 DFS 的主周期。**

❧ **DFT** 是对任意有限持续时间序列可数值计算的傅氏变换。

3.3.1 DFT定义：表达式

❖ 周期序列的表示

长度为 N 的有限长序列 $x(n)$
周期为 N 的周期序列 $\tilde{x}(n)$

关系？

$x(n) = \tilde{x}(n)R_N(n) \longrightarrow \tilde{x}(n)$ 的主值序列（加窗处理）

$\tilde{x}(n) = \sum_{r=-\infty}^{\infty} x(n + rN) = x((n))_N \longrightarrow x(n)$ 的周期延拓

其中： $((n))_N = n \text{ 模 } N$

同样： $X(k)$ 也是一个 N 点的有限长序列

$$X(k) = \tilde{X}(k)R_N(k) \quad \tilde{X}(k) = X((k))_N$$

3.3.1 DFT定义：表达式

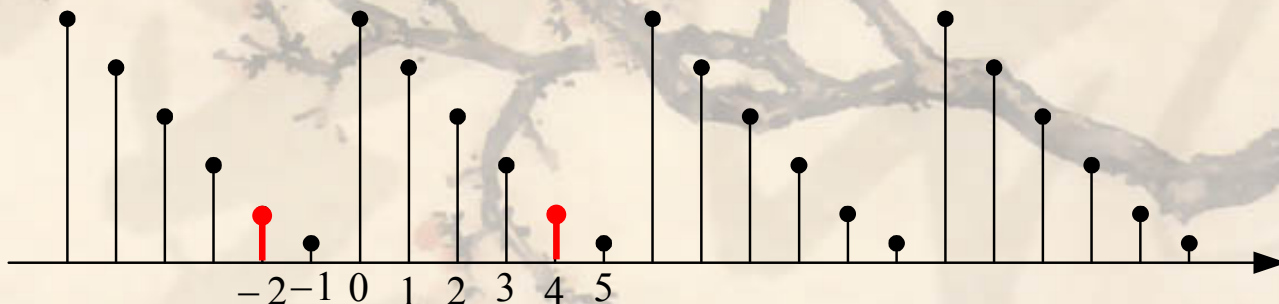
❖ 若 $n=n_1+n_2N$ 成立，且 n_1 满足 $0 \leq n_1 \leq N-1$ ，则把 n_1 称做 n 对 N 的模数，用符号 $((n))_N$ 表示，即： **n 模 $N=((n))_N=n_1$** ，也就是 n 对 N 取余数。

例如： $\tilde{x}(n)$ 是周期为 $N=6$ 的序列，求 $n=19$ 及 $n=-2$ 两数对 N 的余数。

解： $n=19=1+3 \times 6$ ， $((19))_6=1$

$n=-2=(-1) \times 6+4$ ， $((-2))_6=4$

即： $\tilde{x}(19) = x((19))_6 = x(1)$ $\tilde{x}(-2) = x((-2))_6 = x(4)$



3.3.1 DFT 定义： 表达式

❖ Matlab 求模值： 函数 `rem(n,N)` 确定 n 对 N 的模数。

⌘ 当 $n \geq 0$ 时，此函数可用来实现模 N 运算；

⌘ 但 $n < 0$ 时，为得到正确值，需要对结果进行修正，这由下面的 `m=mod(n,N)` 函数完成。在此函数中， n 可以是任意整数数组，数组 m 返回 n 的模 N 值。

Function `m=mod(n,N)`

% 计算 n 对 N 的模 `m=(n mod N)`

`m = rem(n,N);`

`m = m + N;`

`m = rem(m,N);`

3.3.1 DFT 定义：表达式

❖ 在无混迭的情况下，我们看如何把 DFS 变成 DFT ?

DFS:

$$\begin{cases} \tilde{X}(k) = \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{kn} \\ \tilde{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) W_N^{-kn} \end{cases}$$

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

- 因无混迭，则时域中一个周期长的主值序列对应于频域中一个周期长的主值序列。从DFS的时域和频域中各取出一个周期，即得到有限长度离散序列的时域和频域傅氏变换。

3.3.1 DFT 定义：表达式

■ 时域频域各取主值序列得有限长序列的 DFT 正反变换：

$$\begin{cases} X(k) = \tilde{X}(k)R_N(k) = \left[\sum_{n=0}^{N-1} \tilde{x}(n)W_N^{nk} \right] R_N(k) \\ x(n) = \tilde{x}(n)R_N(n) = \left[\frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k)W_N^{-nk} \right] R_N(n) \end{cases}$$

即
DFT

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

注意：从工程角度看，DFS 和 DFT 的表达式没有本质区别。

其中： $W_N = e^{-j\frac{2\pi}{N}}$

DFT

$$\left\{ \begin{array}{l} X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n)W_N^{nk} \quad 0 \leq k \leq N-1 \\ x(n) = IDFT[X(k)] = \frac{1}{N} \sum_{k=0}^{N-1} X(k)W_N^{-nk} \quad 0 \leq n \leq N-1 \end{array} \right.$$

DTFT

$$\left\{ \begin{array}{l} X(e^{j\omega}) = DTFT[x(n)] = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \\ x(n) = IDTFT[X(k)] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n} d\omega \end{array} \right.$$

$$X(k) = X(e^{j\omega}) \Big|_{\omega = \frac{2\pi}{N}k}$$

3.3.1 DFT 定义：表达式

■ DFT的矩阵表示形式

若令：

$$\mathbf{x} = (x(0), x(1), \dots, x(N-1))^T$$

$$\mathbf{X} = (X(0), X(1), \dots, X(N-1))^T$$

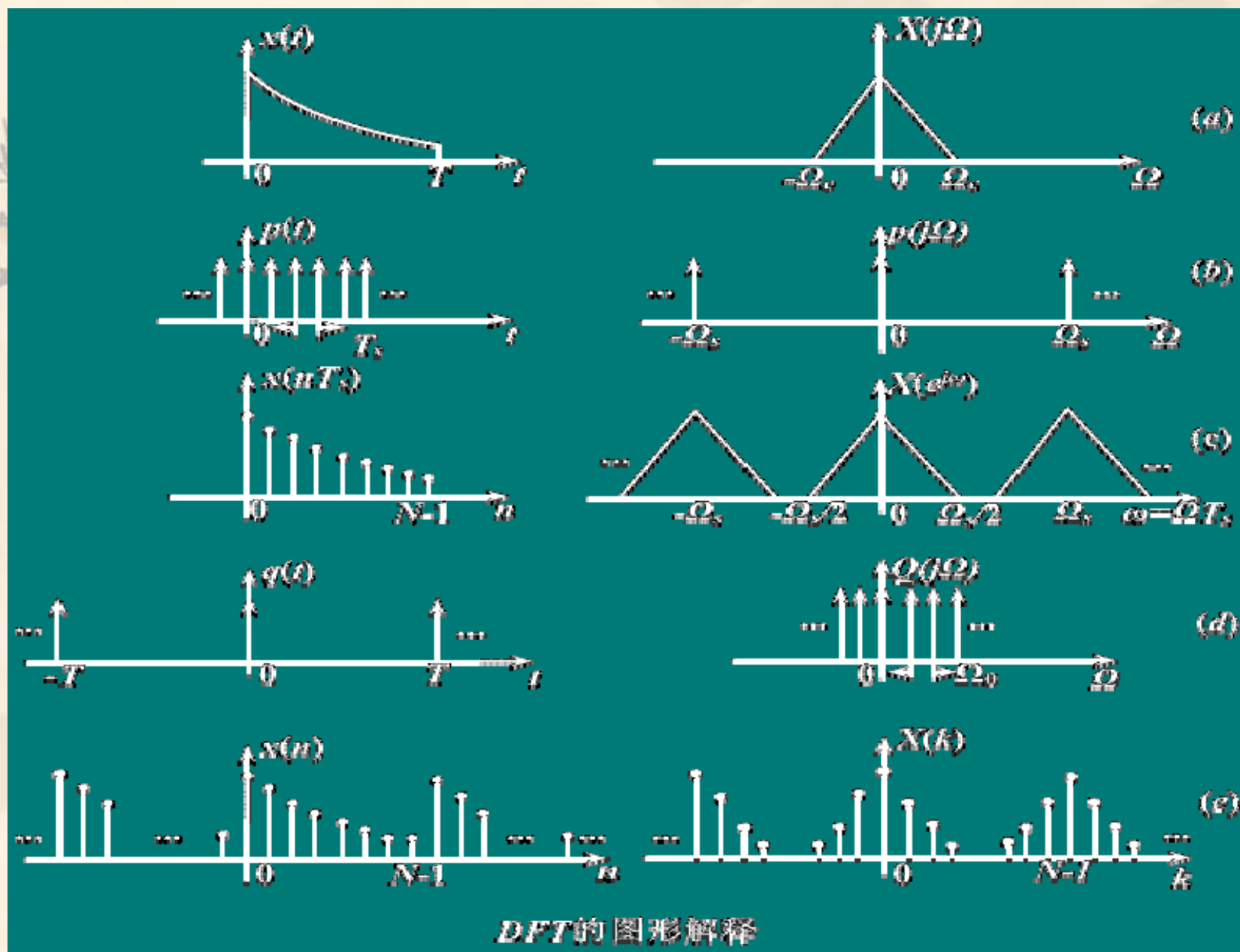
$$\mathbf{W} = \begin{bmatrix} W_N^{0 \times 0} & W_N^{1 \times 0} & \dots & W_N^{(N-1) \times 0} \\ W_N^{0 \times 1} & W_N^{1 \times 1} & \dots & W_N^{(N-1) \times 1} \\ \vdots & \vdots & \vdots & \vdots \\ W_N^{0 \times (N-1)} & W_N^{1 \times (N-1)} & \dots & W_N^{(N-1) \times (N-1)} \end{bmatrix}$$

则：

$$\mathbf{X} = \mathbf{W} \mathbf{x} = \mathbf{W}^T \mathbf{x}$$

$$\mathbf{x} = \frac{1}{N} \mathbf{W}^* \mathbf{X} = \frac{1}{N} \mathbf{W}^{-1} \mathbf{X}$$

3.3.1 DFT 定义：表达式



3.3.1 DFT 定义：表达式

■ DFT 意义

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

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

- ❖ $X(k)$ 不仅浓缩了 $\tilde{X}(k)$ 的全部内容，同时也浓缩了 $X(e^{j\omega})$ 的全部内容。
- ❖ $X(k)$ 能够如实、全面地表示 $x(n)$ 的频域特征，所以DFT具备明确的物理含义。

3.3.1 DFT 定义：举例

例 3.7 $x(n)$ 是一个 4 点序列：

$$x(n) = \begin{cases} 1, & 0 \leq n \leq 3 \\ 0, & \text{others} \end{cases}$$

- 1) 计算离散时间傅氏变换 $X(e^{j\omega})$ ，并画出它的幅度和相位。
- 2) 计算 $x(n)$ 的 4 点 DFT。

3.3.1 DFT 定义：举例

解：1) 离散时间傅氏变换为：

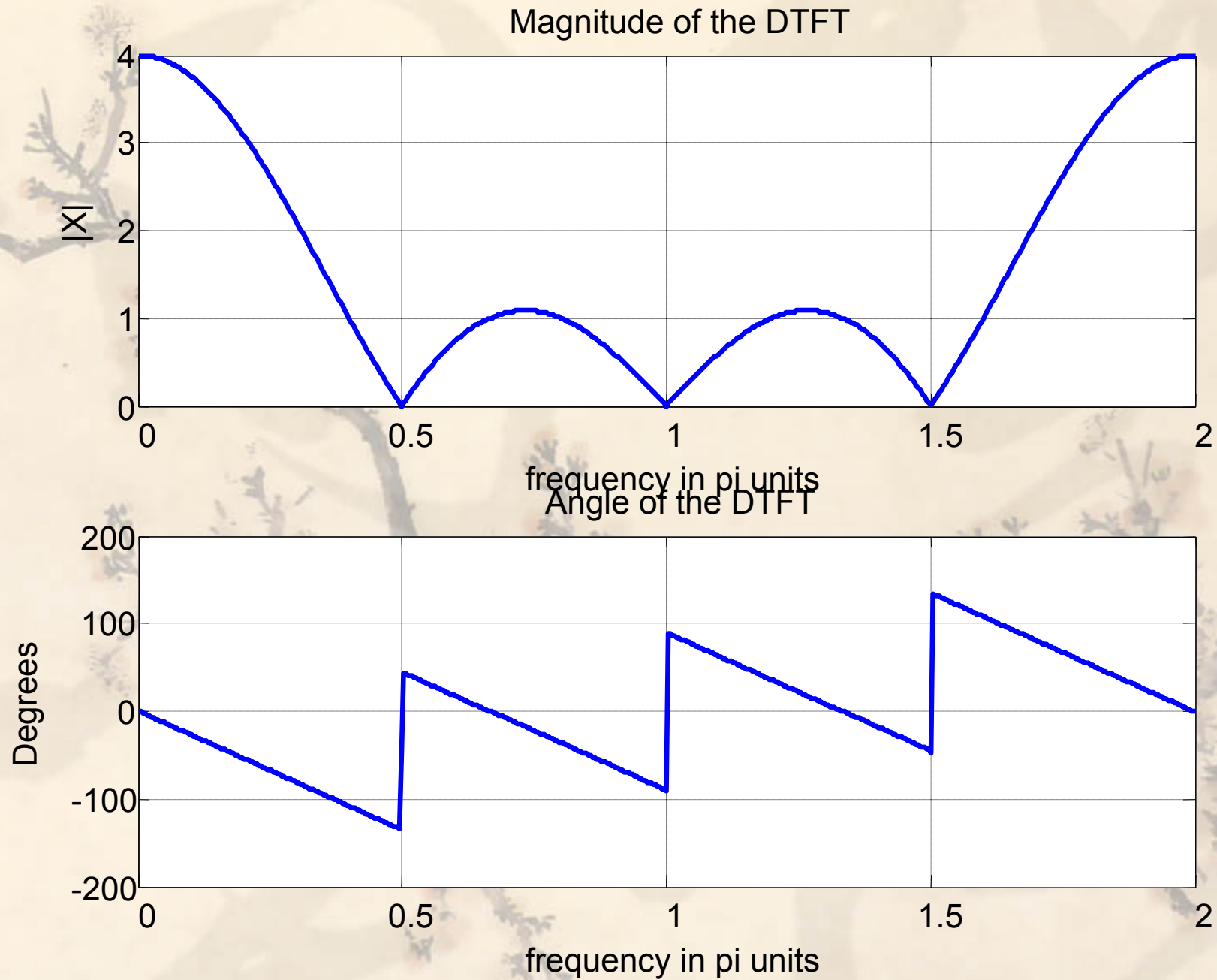
$$\begin{aligned} X(e^{j\omega}) &= \sum_0^3 x(n)e^{-j\omega n} = 1 + e^{-j\omega} + e^{-j2\omega} + e^{-j3\omega} \\ &= \frac{1 - e^{-j4\omega}}{1 - e^{-j\omega}} = e^{-j\frac{3}{2}\omega} \frac{\sin 2\omega}{\sin \omega / 2} \end{aligned}$$

因而

$$|X(e^{j\omega})| = \left| \frac{\sin 2\omega}{\sin \omega / 2} \right|$$

$$\angle X(e^{j\omega}) = \begin{cases} -\frac{3\omega}{2}, & \frac{\sin(2\omega)}{\sin(\omega/2)} > 0 \\ -\frac{3\omega}{2} \pm \pi, & \frac{\sin(2\omega)}{\sin(\omega/2)} < 0 \end{cases}$$

3.3.1 DFT 定义：举例



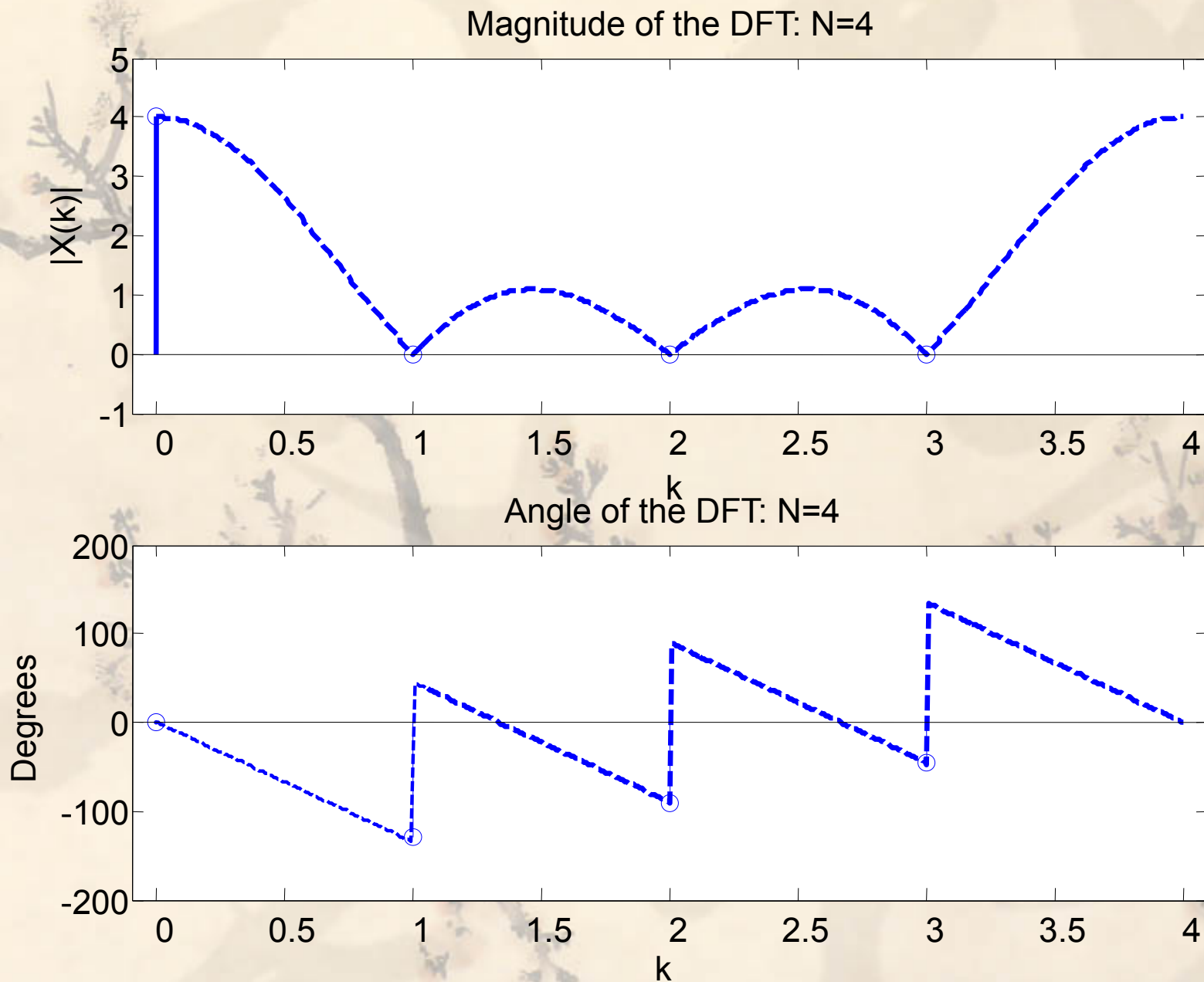
3.3.1 DFT 定义： 举例

2) 用 $X_4(k)$ 表示 4 点 DFT:

$$X_4(k) = \sum_{n=0}^3 x(n) W_4^{nk}; \quad k = 0, 1, 2, 3;$$

```
% b) 4-point DFT
N = 4; w1 = 2*pi/N; k = 0:N-1;
X = dft(x,N);
magX = abs(X), phaX = angle(X)*180/pi
subplot(2,1,1);
plot(w*N/(2*pi),magH,'--');
axis([-0.1,4.1,-1,5]); hold on
stem(k,magX);
```

3.3.1 DFT 定义：举例



3.3.1 DFT定义：举例

例 3.8 怎样得到 DTFT $X(e^{j\omega})$ 的其他样本？

解：显然，我们的采样频率应更小一些，也就是说，应增加 N 的长度。

❖ 有两种方法，

- 一种是取样时就采集更多的样本；
- 另一种是在序列后面添加一定长度的零，叫做填零运算，在实际中，为了得到一个较密的频谱，这种运算是常用的。

3.3.1 DFT 定义： 举例

(a) 给 $x(n)$ 后附加 4 个零得到一个 **8 点序列**。

$$x(n) = \{1, 1, 1, 1, 0, 0, 0, 0\}$$

设 $X_8(k)$ 为一 8 点 DFT, 则

$$X_8(k) = \sum_{n=0}^7 x(n) W_8^{nk}; \quad k = 0, 1, 2, 3 \cdots 7;$$

$$W_8 = e^{-j\pi/4}$$

在这种情况下, 频率分辨率为 $\omega_1 = 2\pi/8 = \pi/4$ 。

3.3.1 DFT 定义： 举例

```
% b) 8-point DFT
```

```
N = 8;
```

```
w1 = 2*pi/N; k = 0:N-1;
```

```
x = [x, zeros(1,4)];
```

```
X = dft(x,N);
```

```
magX = abs(X), phaX = angle(X)*180/pi
```

结果如下：

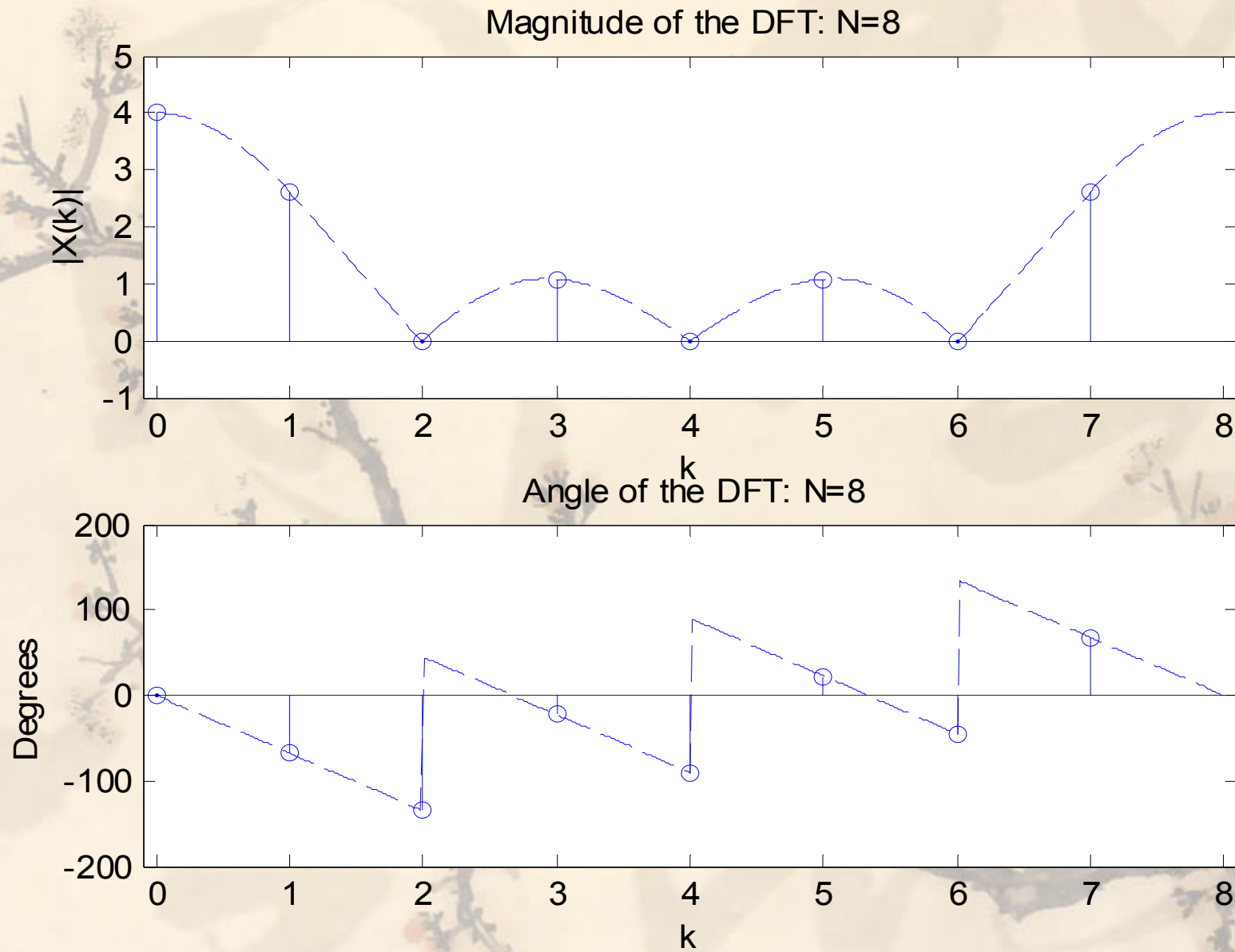
```
magX =
```

```
4.0000 2.6131 0.0000 1.0824 0.0000 1.0824 0.0000 2.6131
```

```
phaX =
```

```
0 -67.5000 -153.4349 -22.5000 -90.0000 22.5000 -53.1301 67.5000
```


3.3.1 DFT 定义： 举例



3.3.1 DFT 定义： 举例

(b) 更进一步，给 $x(n)$ 填充 12 个零，变成一个 16 点序列， 即

$$x(n) = \{1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$$

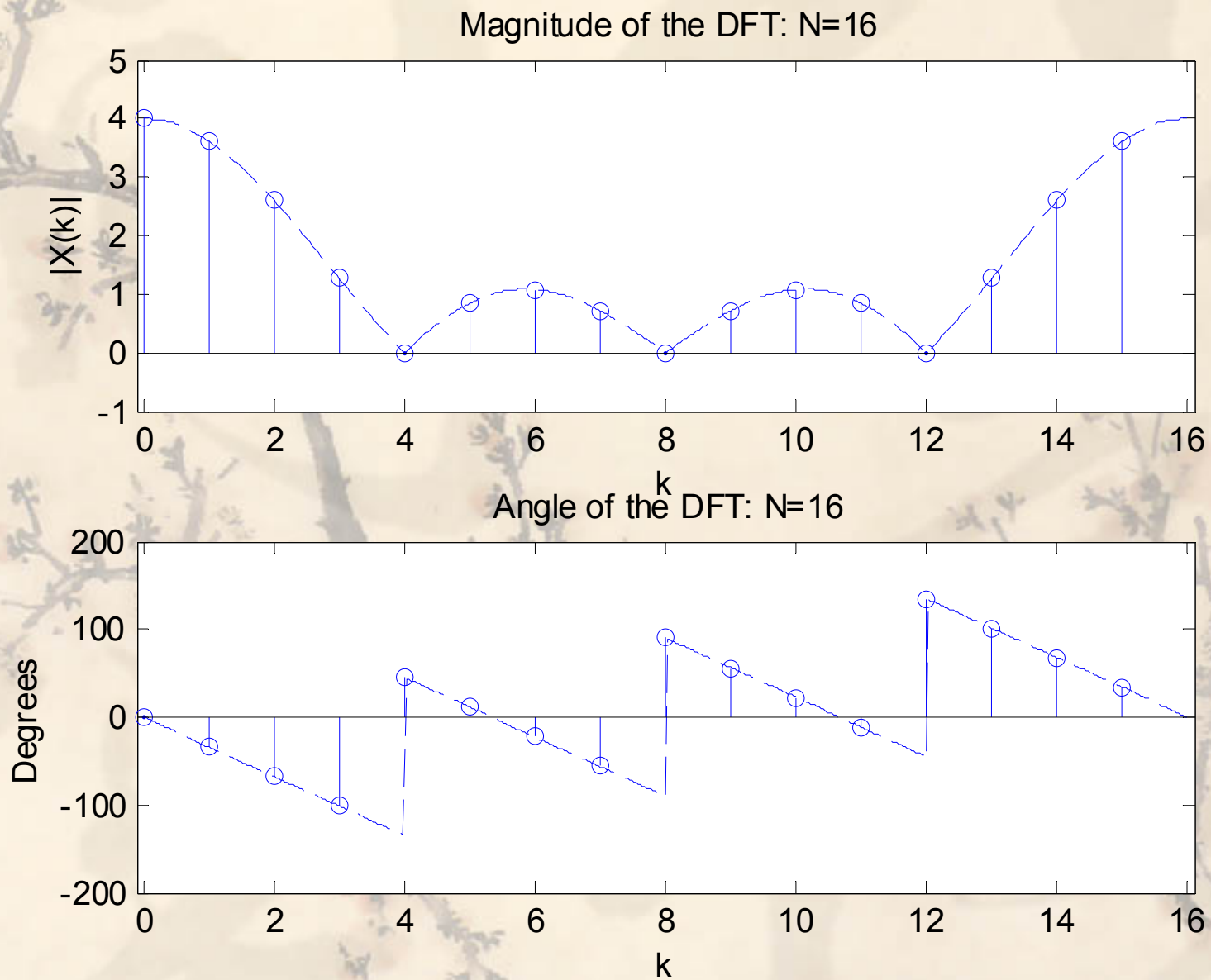
$$X_{16}(k) = \sum_{n=0}^{16} x(n) W_{16}^{nk}; \quad k = 0, 1, 2, 3 \cdots 15;$$
$$W_{16} = e^{-j\pi/8}$$

在这种情况下，频率分辨率为 $\omega_1 = 2\pi/16 = \pi/8$ 。

3.3.1 DFT 定义： 举例

```
% c) 16-point DFT
subplot(1,1,1)
N = 16; w1 = 2*pi/N; k = 0:N-1;
x = [x, zeros(1,8)];
X = dft(x,N);
magX = abs(X), phaX = angle(X)*180/pi
subplot(2,1,1);plot(w*N/(2*pi),magH,'--');
axis([-0.1,16.1,-1,5]); hold on
stem(k,magX);
xlabel('k');
ylabel('|X(k)|'); title('Magnitude of the DFT: N=16')
hold off
subplot(2,1,2);plot(w*N/(2*pi),phaH*180/pi,'--');
axis([-0.1,16.1,-200,200]); hold on
stem(k,phaX);
xlabel('k');
ylabel('Degrees'); title('Angle of the DFT: N=16')
```


3.3.1 DFT 定义：举例



3.3.1 DFT 定义： 举例

结论：基于以上两个例子，可以得到以下结论。

- ① 填零是给原始序列填零的运算。这导致较长的 **DFT**，它会给原始序列的离散时间傅氏变换提供间隔更密的样本。在 **Matlab** 中，用 **zeros** 函数实现填零运算。
- ② 为精确地画出离散时间傅氏变换 $X(e^{j\omega})$ ，只需要 4 点 **DFT** $X_4(k)$ 。这是因为 $x(n)$ 仅有 4 个非零样本，因此，可通过填零得到 $X_8(k)$ 、 $X_{16}(k)$ 等等，用它们来填充 $X(e^{j\omega})$ 的值。
- ③ 填零运算提供了一个较密的频谱和较好的图示形式，但因为在信号中只是附加了零，而**没有增加任何新的信息**，还是原始连续谱的 N 点取样，只是补零观察到了更多的频点，但这**并不意味着补零能够提高真正的频谱分辨率**。
- ④ 采集更多的数据，可以真正提高频谱分辨率。其他的先进方法则是利用边缘信息和非线性技术。

3.3.1 DFT 定义： 举例

例 3.9 为了说明补零（高密度频谱）和采集更多数据（高分辨率频谱）之间的区别，考察序列

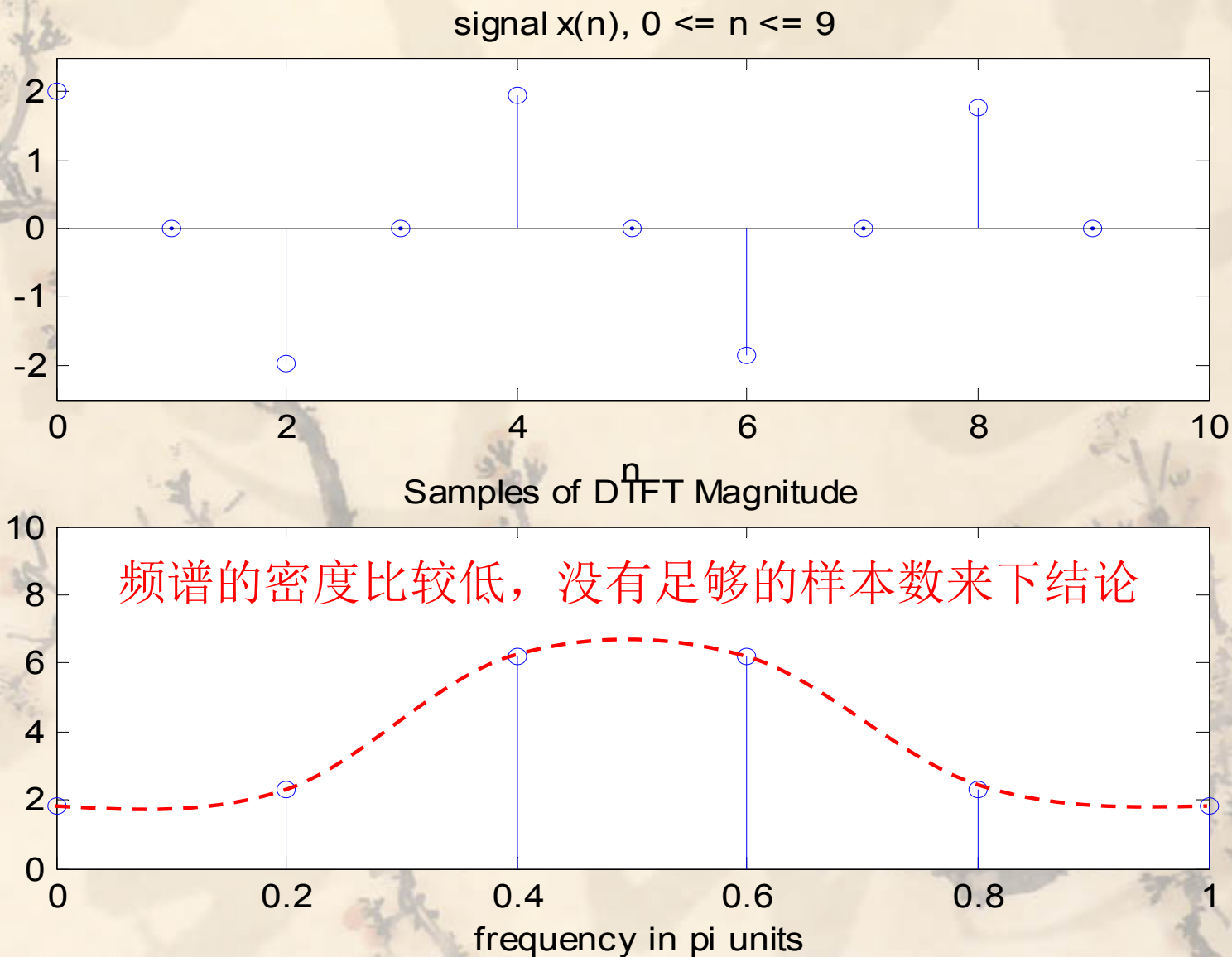
$$x(n) = \cos(0.48\pi n) + \cos(0.52\pi n)$$

求出它基于有限个样本的频谱。

- a) 当 $0 \leq n \leq 10$ 时，确定并画出 $x(n)$ 的离散傅氏变换。
- b) 当 $0 \leq n \leq 100$ 时，确定并画出 $x(n)$ 的离散傅氏变换。

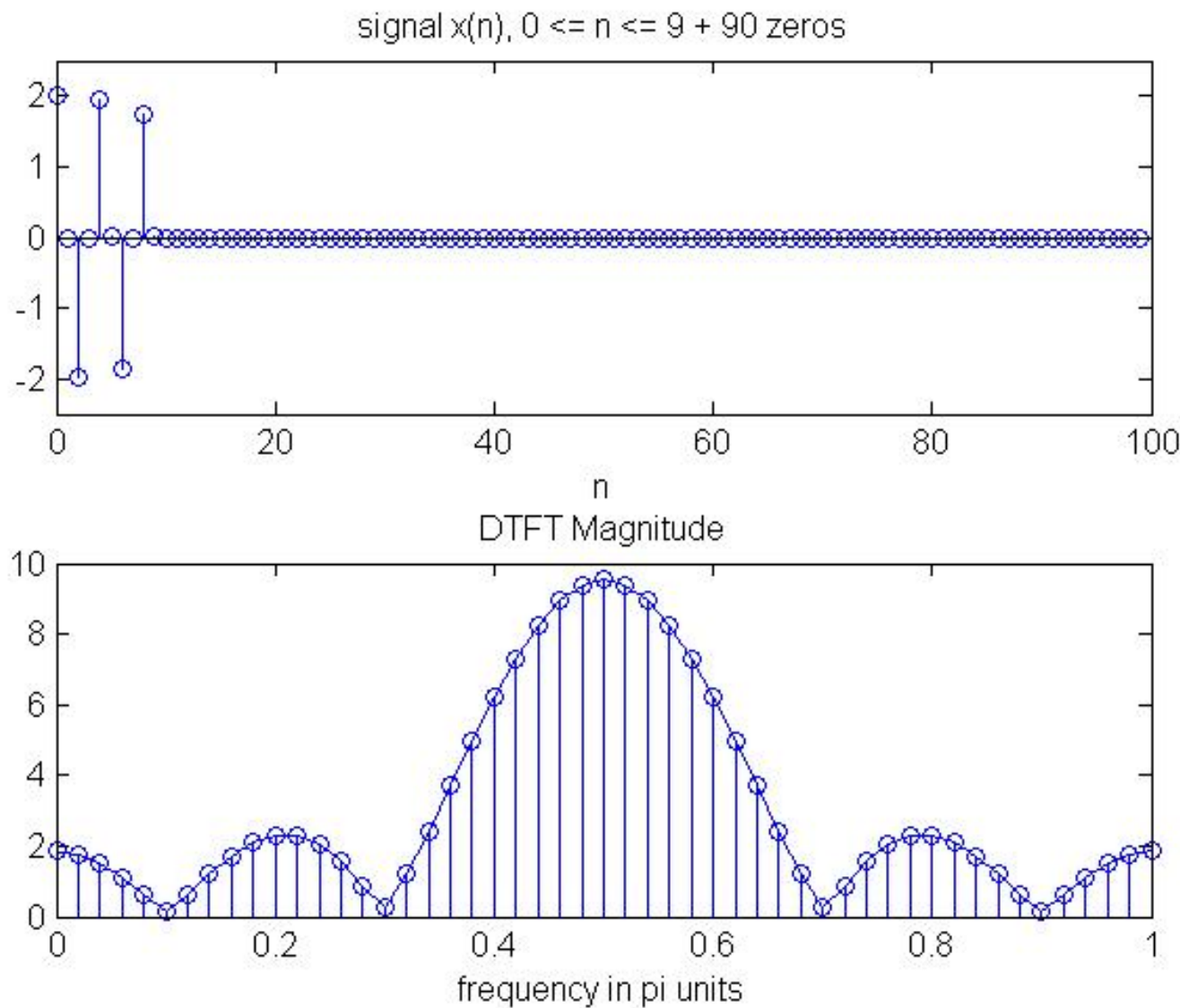
3.3.1 DFT 定义：举例

a) 首先确定 $x(n)$ 的 10 点 DFT，得到其离散时间傅氏变换的估计



3.3.1 DFT 定义：举例

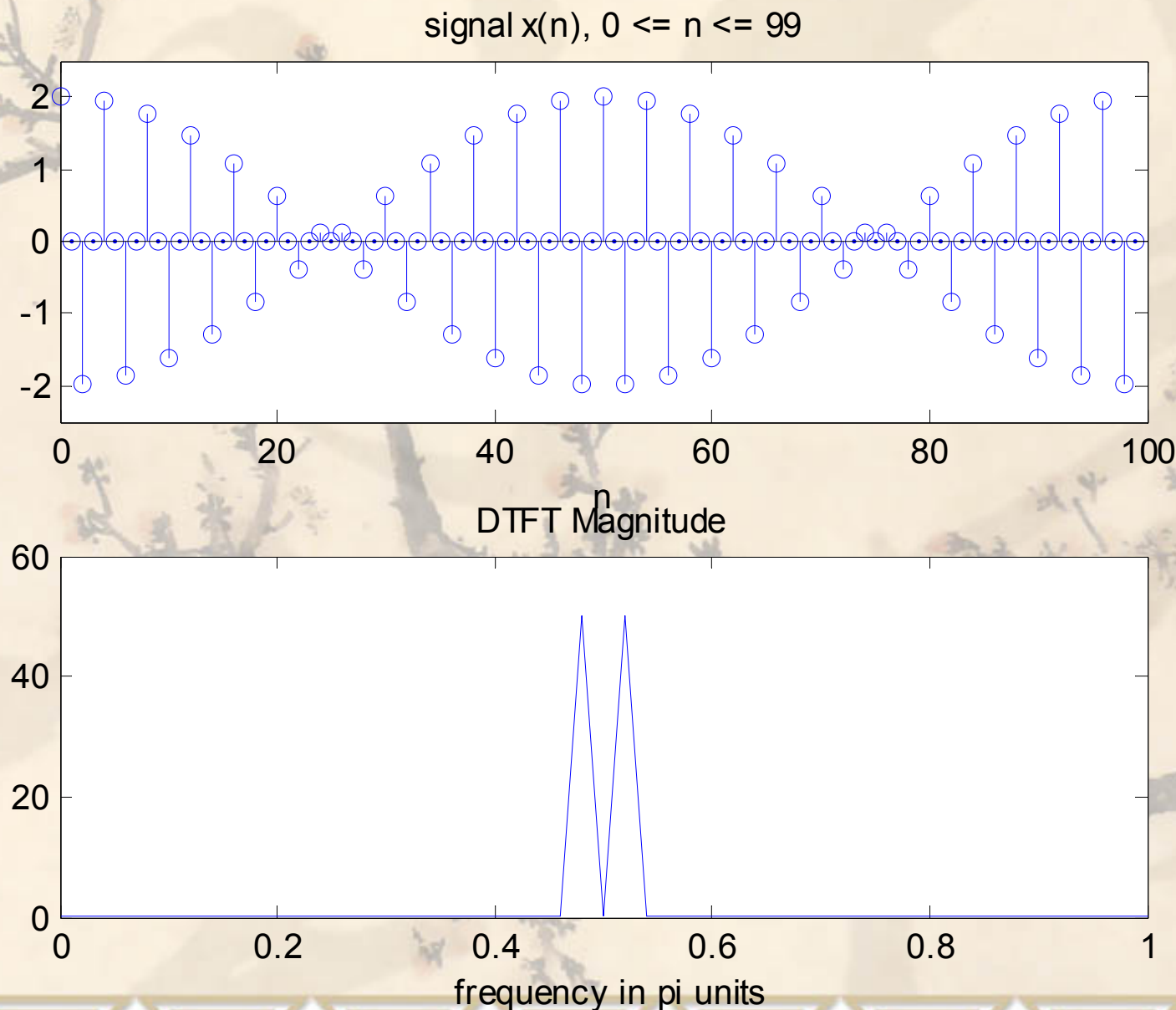
填充 90 个零以得到较密的频谱



从结果图中可以看到，此序列在 $\omega = 0.5\pi$ 处有一主频率，原始序列则没有说明这一点。填零运算提供了更加平滑，高密度的频谱曲线

3.3.1 DFT 定义：举例 (100 个采样点)

b) 为得到更多的频谱信息，采集更多的样本，用 $x(n)$ 的 100 个样本来确定它的 DFT。

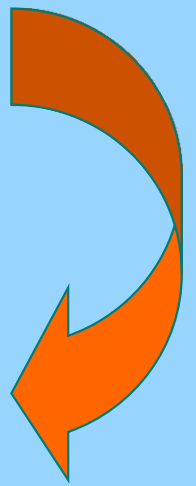


采样更多的
数据提供了
更多的信息

DFT 清楚地表
明了两个靠得
很紧的频率，
这是 $x(n)$ 的
高分辨率的频
谱

3.3.2 DFT 与 DTFT 和 Z 变换的关系

- ❖ 序列 $x(n)$ 的 Z 变换在单位圆上进行 N 等分，即 $\omega = 2\pi/N$ ，就是序列的 **DFT** 变换。**(取样)**
- ❖ Z 变换和 **DFT** 的关系是**取样和内插**的关系， 这在实际应用中很重要。


$$\begin{aligned} x(n) &\Rightarrow X(k) = \text{DFT}[x(n)] = \sum_{n=0}^{N-1} x(n) W_N^{nk} \\ &\downarrow \\ X(z) &= \sum_{n=0}^{N-1} x(n) z^{-n} \quad (\text{Z 变换}) \\ &\downarrow \\ X(e^{j\omega}) &= X(z) \Big|_{z=e^{j\omega}} \quad (\text{DTFT}) \end{aligned}$$

3.3.2 DFT 与 Z 变换的关系：取样 Z 变换

■ 取样 Z 变换

■ 设 $x(n)$ 为一个长度为 N 的有限长序列，则有：

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} = \sum_{n=0}^{N-1} x(n)z^{-n}$$

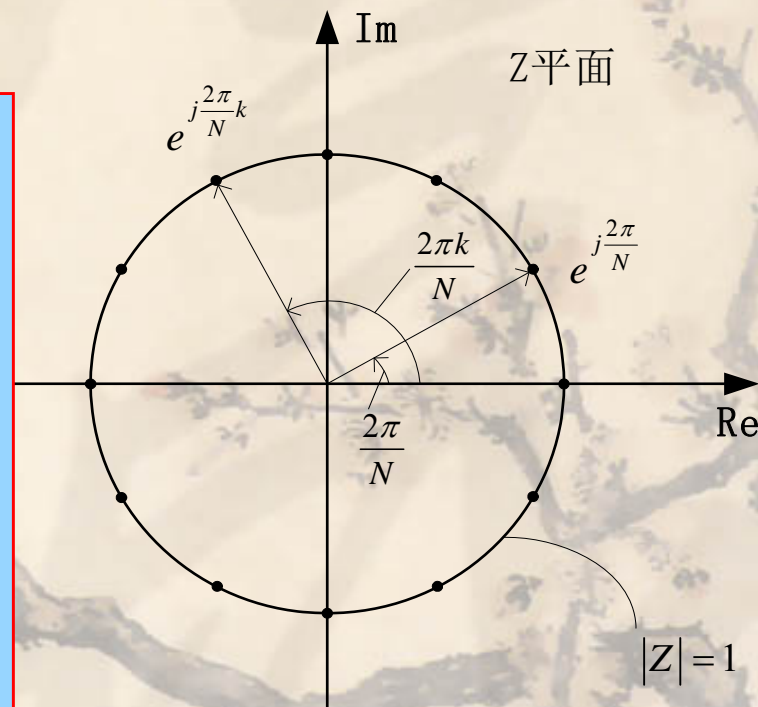
$$X(e^{j\omega}) = \sum_{n=0}^{N-1} x(n)e^{-jn\omega} = DTFT\{x(n)\}$$

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n)W_N^{nk} = X(z) \bigg|_{z=W_N^{-k}=e^{j\frac{2\pi}{N}k}} \\ &= X(e^{j\omega}) \bigg|_{\omega=\frac{2\pi}{N}k} \\ &= DFT\{x(n)\} \end{aligned} \quad k \in \{0, 1, \dots, (N-1)\}$$

3.3.2 DFT 与 Z 变换的关系：取样 Z 变换

- 从 DTFT 的角度看：有限长序列的 DFT 结果包含了 N 个离散频率点处的 DTFT 结果，这个离散频率点等间隔地分布在区间 $[0, 2\pi)$ 内；

- 从 Z 变换的角度看：DFT 结果包含了 z 平面上 N 个离散点处的 Z 变换结果，这 N 个离散点均匀地分布在单位圆上，由此也称 DFT 为单位圆上的取样 Z 变换。



3.3.2 DFT与Z变换的关系：z域内插

- **时域取样定理**：在满足奈奎斯特定理条件下，时域取样信号可以不失真地还原原连续信号。
- **频域取样**情况如何？取样条件？内插公式？

❖ 频域取样定理：若序列长度为 M ，则只有当频域采样点数 N 满足 $N \geq M$ 时，才有：

$$\tilde{x}_N(n)R_N(n) = IDFS[\tilde{X}(k)]R_N(n) = x(n)$$

- 即可由频域取样 $X(k)$ 不失真地恢复原信号 $x(n)$ ，否则产生时域混叠现象。
- 此时可由 N 个取样值 $X(k)$ **内插恢复**出 $X(z)$ 或 $X(e^{j\omega})$

。

3.3.2 DFT与Z变换的关系：z域内插

❖ **Z 域内插公式：** 由 DFT $X(k)$ 可以确定 z 平面上任一点处的, $X(z)$, 确定唯一的 $x(n)$

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

$$\Phi(z) = \frac{1}{N} \frac{1 - z^{-N}}{1 - z^{-1}}$$

内插函数

z 平面内插公式

3.3.2 DFT与Z变换的关系：z域内插

■ 内插函数的零极点分布

$$\Phi(z) = \frac{1}{N} \frac{1 - z^{-N}}{1 - z^{-1}} = \frac{1}{N} \frac{z^N - 1}{z^{N-1}(z - 1)}$$

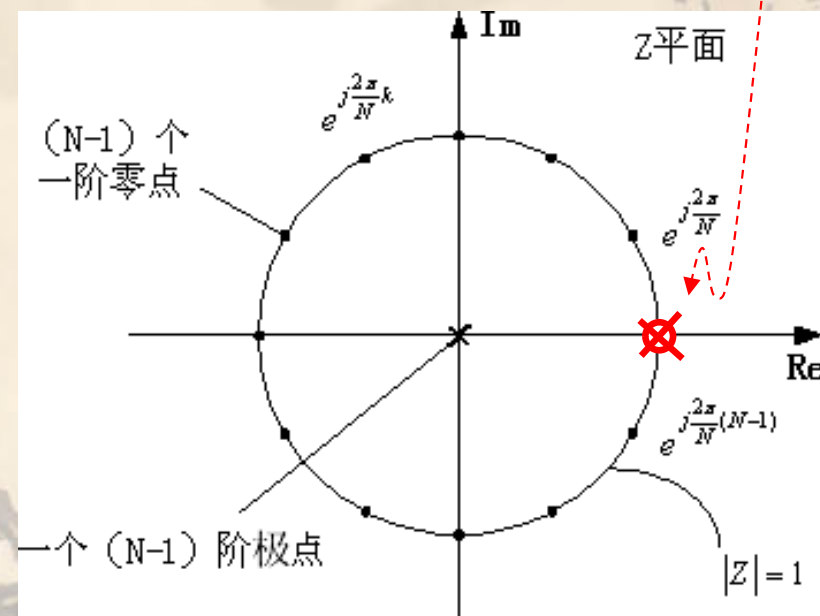
□ 极点：(N-1)阶极点 $z = 0$ ；
一阶极点 $z = 1$ ；

□ 零点：N个一阶零点：

$$z_l = e^{j\frac{2\pi l}{N}}, \quad l = 0, 1, 2, \dots, (N-1)$$

□ 抵消： $z = 1$ 处的一阶极点和一阶零点互相抵消，一阶零点数量变为(N-1)个。

零、极点
互相抵消



$\Phi(z)$ 的零、极点分布

3.3.2 DFT 与 Z 变换的关系：F 域内插

❖ F 域内插公式：由频域取样 DFT $X(k)$ 表示 DTFT $X(e^{j\omega})$

$$\begin{aligned} X(e^{j\omega}) &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \frac{1 - z^{-N}}{1 - W_N^{-k} z^{-1}} \Big|_{z=e^{j\omega}} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \frac{1 - e^{-j\omega N}}{1 - e^{j\frac{2\pi k}{N}} e^{-j\omega}} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \frac{e^{-j\frac{\omega N}{2}}}{e^{j\frac{\pi k}{N}} e^{-j\frac{\omega}{2}}} \frac{e^{j\frac{\omega N}{2}} - e^{-j\frac{\omega N}{2}}}{e^{-j\frac{\pi k}{N}} e^{j\frac{\omega}{2}} - e^{j\frac{\pi k}{N}} e^{-j\frac{\omega}{2}}} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \frac{(-1)^k e^{-j\frac{\omega N}{2}}}{e^{-j\frac{1}{2}(\omega - \frac{2\pi k}{N})}} \frac{(-1)^k \sin(\frac{\omega N}{2})}{\sin(\frac{1}{2}(\omega - \frac{2\pi k}{N}))} \end{aligned}$$

3.3.2 DFT 与 Z 变换的关系: F 域内插

$$\begin{aligned} &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \frac{e^{-j\frac{N}{2}(\omega - \frac{2\pi k}{N})} \sin(\frac{N}{2}(\omega - \frac{2\pi k}{N}))}{e^{-j\frac{1}{2}(\omega - \frac{2\pi k}{N})} \sin(\frac{1}{2}(\omega - \frac{2\pi k}{N}))} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{-j\frac{N-1}{2}(\omega - \frac{2\pi k}{N})} \frac{\sin(\frac{N}{2}(\omega - \frac{2\pi k}{N}))}{\sin(\frac{1}{2}(\omega - \frac{2\pi k}{N}))} \end{aligned}$$

即: $X(e^{j\omega}) = \sum_{k=0}^{N-1} X(k) \Phi_k(\omega) = \sum_{k=0}^{N-1} X(k) \Phi(\omega - \frac{2\pi k}{N})$

其中:

$$\Phi(\omega) = \frac{1}{N} e^{-j\frac{N-1}{2}\omega} \frac{\sin(\frac{N\omega}{2})}{\sin(\frac{\omega}{2})} = \Phi(z)|_{z=e^{j\omega}}$$

频域内插公式

频域内插函数

3.3.2 DFT 与 Z 变换的关系: F 域内插

■ F 内插函数的零极点分布

根据 $\Phi(z)$ 的零极点分布规律可知:
(零极点对系统频率响应的影响)

极点: $e^{j\omega}$ 到极点 $z=0$ 的距离恒为1, 对幅频特性没有影响

零点:

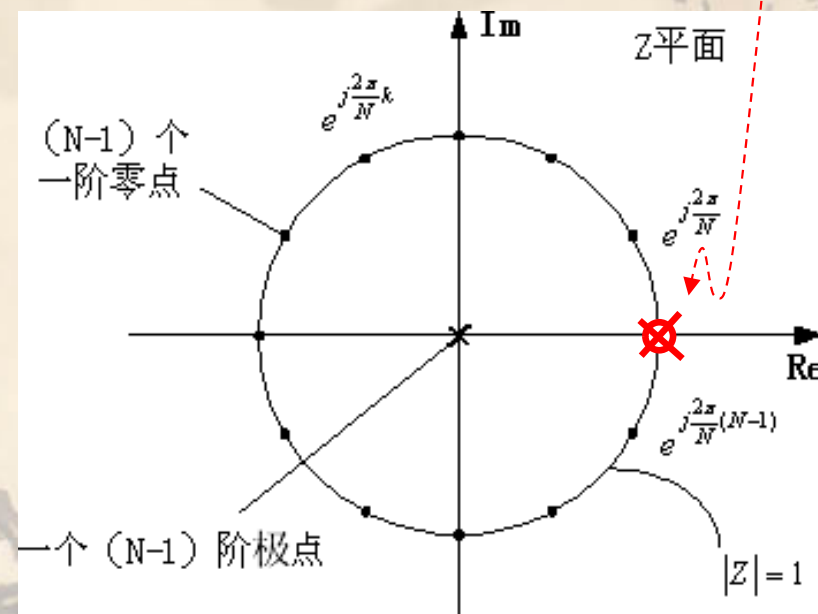
✧ 在区间 $[0, 2\pi]$ 内, $|\Phi(\omega)|$ 存在 $(N-1)$ 个零值点

$$\omega = \frac{2\pi k}{N}, k = 1, 2, \dots, (N-1)$$

✧ 存在 $(N-1)$ 个极值点, 分别为:

$$\omega = 0$$

零、极点
互相抵消



$\Phi(z)$ 的零、极点分布

3.3.2 DFT 与 Z 变换的关系：F 域内插

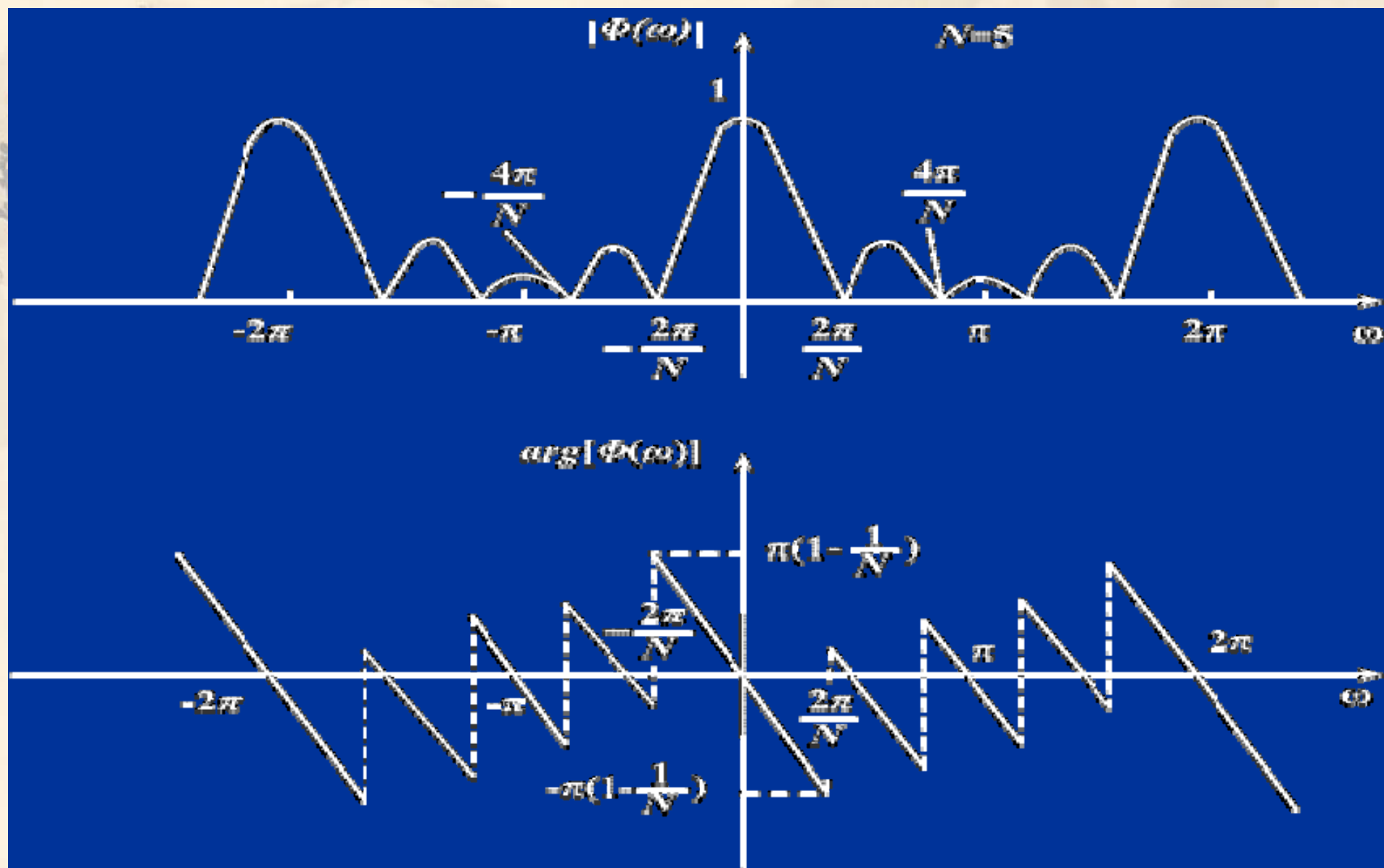
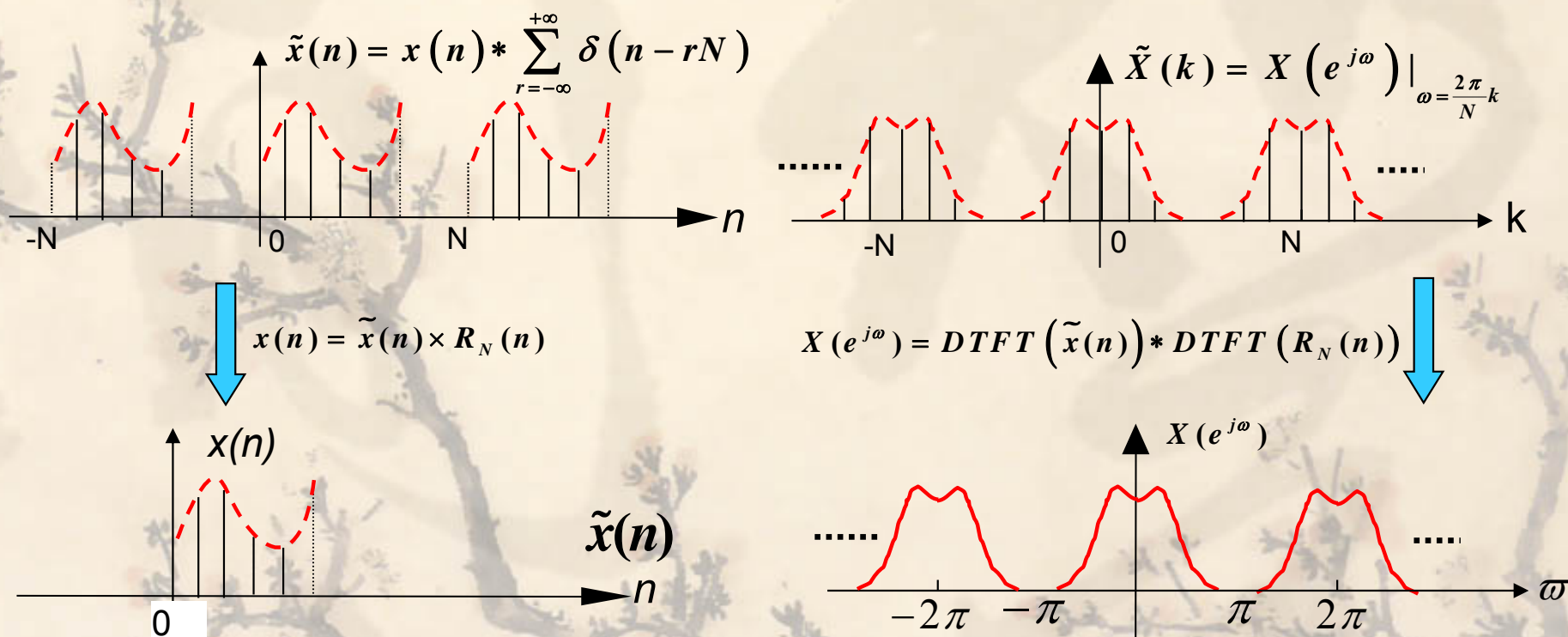


图3-13 插值函数 $\Phi(\omega)$ 的幅度特性与相位特性 ($N=5$)

3.3.2 DFT 与 Z 变换的关系：F 域内插

❖ 频域内插的物理含义

频域取样 时域周期延拓



■ **频域内插的物理意义：**频域对 $X(e^{j\omega})$ 取样得到 $X(k)$ 对应时域 $x(n)$ 序列的周期性延拓相加 $\tilde{x}(n)$ 。因此，如果能在时域上从 $\tilde{x}(n)$ 恢复 $x(n)$ ，那么频域就可以由 $X(k)$ 准确重建 $X(e^{j\omega})$ 。

$$\begin{aligned}
 X(e^{j\omega}) &= DTFT(x(n)) = DTFT(\tilde{x}(n)) * DTFT(R_N(n)) \\
 &= \left(\frac{2\pi}{N} \sum_{k=-\infty}^{+\infty} \tilde{X}(k) \times \delta(\omega - \frac{2\pi}{N}k) \right) * \frac{e^{-j\frac{(N-1)}{2}\omega} \sin(\frac{\omega N}{2})}{\sin(\frac{\omega}{2})} \\
 &= \frac{1}{2\pi} \int_0^{2\pi} \frac{2\pi}{N} \sum_{k=-\infty}^{+\infty} \tilde{X}(k) \times \delta(\theta - \frac{2\pi}{N}k) \times \frac{e^{-j\frac{(N-1)}{2}(\omega-\theta)} \sin(\frac{(\omega-\theta)N}{2})}{\sin(\frac{(\omega-\theta)}{2})} d\theta \\
 &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \frac{\sin(\frac{N}{2}(\omega - \frac{2\pi}{N}k))}{\sin(\frac{1}{2}(\omega - \frac{2\pi k}{N}))} e^{-j\frac{(N-1)}{2}(\omega - \frac{2\pi k}{N})} \\
 &= \sum_{k=0}^{N-1} X(k) \Phi_k(\omega)
 \end{aligned}$$

■ 频域取样时，如果取样间隔为 $2\pi/M$ ，则时域延拓周期为 M ，而只有 $M \geq N$ 时， $\tilde{x}(n)$ 中不会出现混迭现象，因此，能够从 $\tilde{x}(n)$ 中如实恢复 $x(n)$ ，即能够由 $X(k)$ 准确重建 $X(z)$ 和 $X(e^{j\omega})$ 。所以，对序列做DFT时要求DFT的点数不应低于序列长度。

■ $X(k)$ 浓缩了 $x(n)$ 在变换域中的全部特性。DFT具有明确的物理意义。

3.3.3 DFT 的性质：线性

若两序列 $x_1(n)$ 和 $x_2(n)$ 的长度均为 N ，且其 N 点 DFT 分别为：

$$X_1(k) = DFT[x_1(n)]$$

$$X_2(k) = DFT[x_2(n)]$$

则：

$$DFT[ax_1(n) + bx_2(n)] = aX_1(k) + bX_2(k)$$

a, b 为任意常数

- 这里，序列长度及 DFT 点数均为 N
- 若不等，分别为 N_1 、 N_2 ，则需补零使两序列长度相等，均为 N ，且 $N \geq \max[N_1, N_2]$

3.3.3 DFT 的性质：对称性

- ❖ 对于周期序列，有：

$$\tilde{x}(n) = \tilde{x}_e(n) + \tilde{x}_o(n)$$

其中： $\tilde{x}_e(n) = \frac{1}{2}[\tilde{x}(n) + \tilde{x}^*(-n)]$

— 周期共轭偶对称分量

$$\tilde{x}_o(n) = \frac{1}{2}[\tilde{x}(n) - \tilde{x}^*(-n)]$$

— 周期共轭奇对称分量

- ❖ 又定义：

$$x_e(n) = \tilde{x}_e(n)R_N(n)$$

— 共轭偶对称分量

$$x_o(n) = \tilde{x}_o(n)R_N(n)$$

— 共轭奇对称分量

- ❖ 又由于 $x(n) = \tilde{x}(n)R_N(n)$ 则： $x(n) = x_e(n) + x_o(n)$

- ❖ 即有限长序列由共轭偶对称和共轭奇对称两部分组成。

3.3.3 DFT 的性质：对称性

若 $x(n) \xleftrightarrow{DFT} X(k)$, 则

$$1) x^*(n) \leftrightarrow X^*((-k))_N R_N(n)$$

$$2) x^*((-n))_N R_N(n) \leftrightarrow X^*(k)$$

$$3) R_e[x(n)] \leftrightarrow X_e(k) \quad -X(k) \text{ 的共轭偶对称部分}$$

$$4) jI_m[x(n)] \leftrightarrow X_o(k) \quad -X(k) \text{ 的共轭奇对称部分}$$

$$5) x_e(n) \leftrightarrow R_e[X(k)]$$

$$6) x_o(n) \leftrightarrow jI_m[X(k)]$$

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

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

若 $x(n)$ 为实序列, $x(n) = x^*(n)$

$$7) X(k) = X^*((-k))_N R_N(k)$$

$$8) \operatorname{Re}[X(k)] = \operatorname{Re}[X((-k))_N]$$

$$9) \operatorname{Im}[X(k)] = -\operatorname{Im}[X((-k))_N]$$

$$10) \arg[X(k)] = -\arg[X((-k))_N]$$

例3.3: 已知序列 $x(n) = R_4(n)$, 求 $x(n)$ 的8点DFT。

$$\begin{aligned}
 X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} = \sum_{n=0}^7 x(n) W_8^{nk} \\
 &= \sum_{n=0}^3 W_8^{nk} \\
 &= 1 + e^{-j\frac{2\pi}{8}k} + e^{-j\frac{2\pi}{8}2k} + e^{-j\frac{2\pi}{8}3k}
 \end{aligned}$$

$$X(k) = X^*((-k))_N R_N(k)$$

$$\begin{aligned}
 X(0) &= 4 & X(1) &= 1 - j(\sqrt{2} + 1) & X(2) &= 0 & X(3) &= 1 - j(\sqrt{2} - 1) \\
 X(4) &= 0 & X(5) &= 1 + j(\sqrt{2} - 1) & X(6) &= 0 & X(7) &= 1 + j(\sqrt{2} + 1)
 \end{aligned}$$

3.3.3 DFT 的性质：帕斯瓦尔定理

- ❖ 若长度为 N 的有限长序列 $x(n)$ ，其 N 点 DFT 的结果为 $X(k)$ ，则有，

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

证明：

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

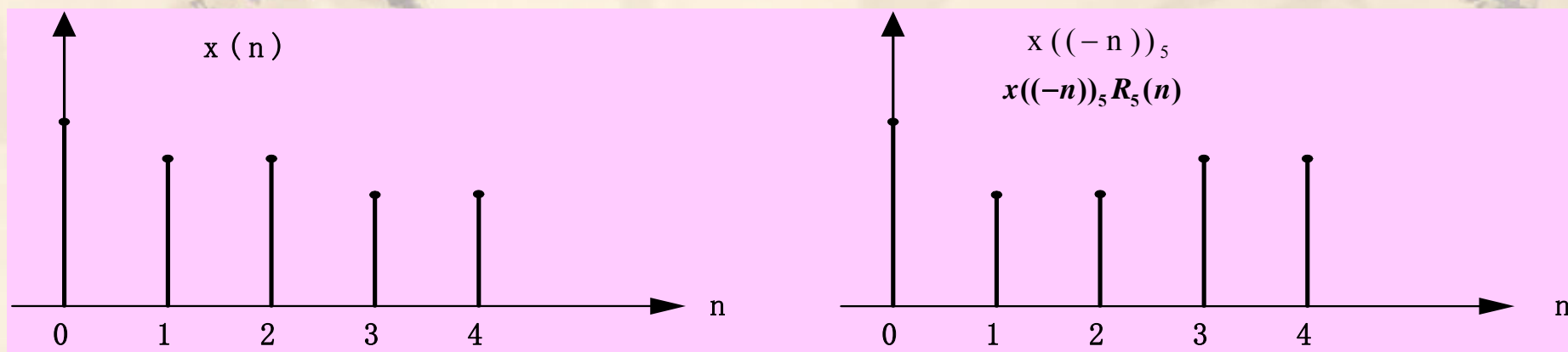
- **该定理表明**：可利用序列的 DFT 结果表示信号的能量，序列在时域计算的能量与在频域计算的能量相等，即变换前后的能量保持不变。
- 这进一步说明，虽然 DFT 有别于 DTFT，但其仍然具有明确的物理含义。

3.3.3 DFT 的性质：反转定理

❖ 循环反转的定义：

■ 如果 $x(n)$ 是长度为 N 的序列，则称 $x((-n))_N R_N(n)$ 为 $x(n)$ 的**循环反转运算**。

$$x((-n))_N R_N(n) = \begin{cases} x(0) & n = 0 \\ x(N - n) & n = 1, \dots, (N - 1) \end{cases}$$



循环反转运算是有限长序列所特有的一种运算，其结果仍然是集合 $\{0, 1, \dots, (N-1)\}$ 上的有限长序列，**特别注意 $n=0$ 时情况**。

计算过程： 1) 补零为 N ； 2) 周期延拓；
3) 纵轴镜像； 4) 取主值序列。

3.3.3 DFT 的性质：反转定理

❖ 循环反转的 DFT

$$\text{若 } x(n) \xrightarrow{DFT} X(k)$$

$$\text{则 } x((-n))_N R_N(n) \xrightarrow{DFT} X((-k))_N R_N(k)$$

证明：

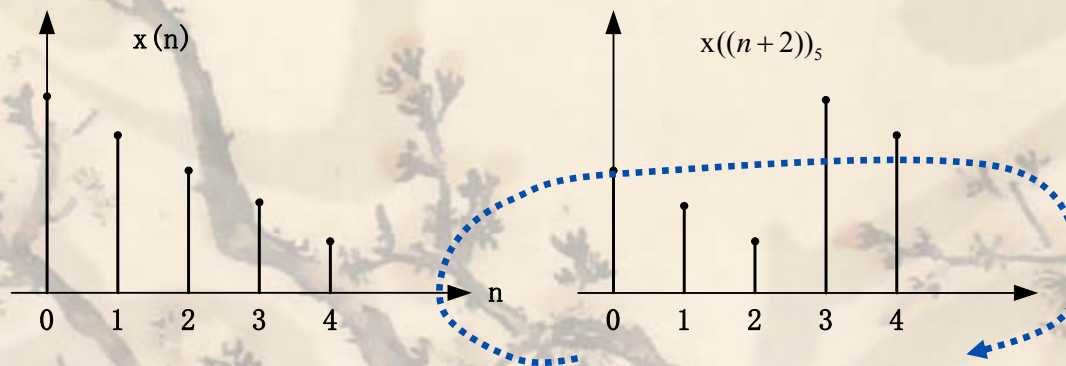
$$\begin{aligned} DFT[x((-n))_N R_N(n)] &= \sum_{n=0}^{N-1} x((-n))_N R_N(n) W_N^{nk} \\ &= \sum_{r=0}^{N-1} x(r) W_N^{-(lN+r)k} \quad (-n = lN + r, 0 \leq r \leq (N-1)) \\ &= \sum_{r=0}^{N-1} x(r) W_N^{-rk} = \sum_{r=0}^{N-1} x(r) W_N^{r(-k)} = \sum_{r=0}^{N-1} x(r) W_N^{r[-k-lN]R_N(k)} \\ &= \sum_{r=0}^{N-1} x(r) W_N^{r((-k))_N R_N(k)} = X((-k))_N R_N(k) \end{aligned}$$

3.3.3 DFT 的性质：序列的循环移位(圆周移位)

❖ 循环移位的定义：

$$x_m(n) = x((n+m))_N R_N(n)$$

- ❑ 称其为循环移位的原因在于，当序列从一端移出范围时，移出的部分又会从另一端移入该范围。
- ❑ 线性移位：若 N 点序列沿一方向线性移位，它将不再位于区间 $0 \leq n \leq N-1$ 上。



$$x(n) \xrightarrow{\text{周期延拓}} \tilde{x}(n) \xrightarrow{\text{移位}} \tilde{x}(n+m) \xrightarrow{\text{取主值序列}} x_m(n) = x((n+m))_N R_N(n)$$

3.3.3 DFT 的性质：序列的循环移位(圆周移位)



或称为**圆周移位**。把 $x(n)$ 看作排列在 N 等分的圆周上，循环移位就相当于序列 $x(n)$ 在圆周上移动，故称为圆周移位。实际上重复观察几周时，看到的就是周期序列。

3.3.3 DFT 的性质：序列的循环移位(圆周移位)

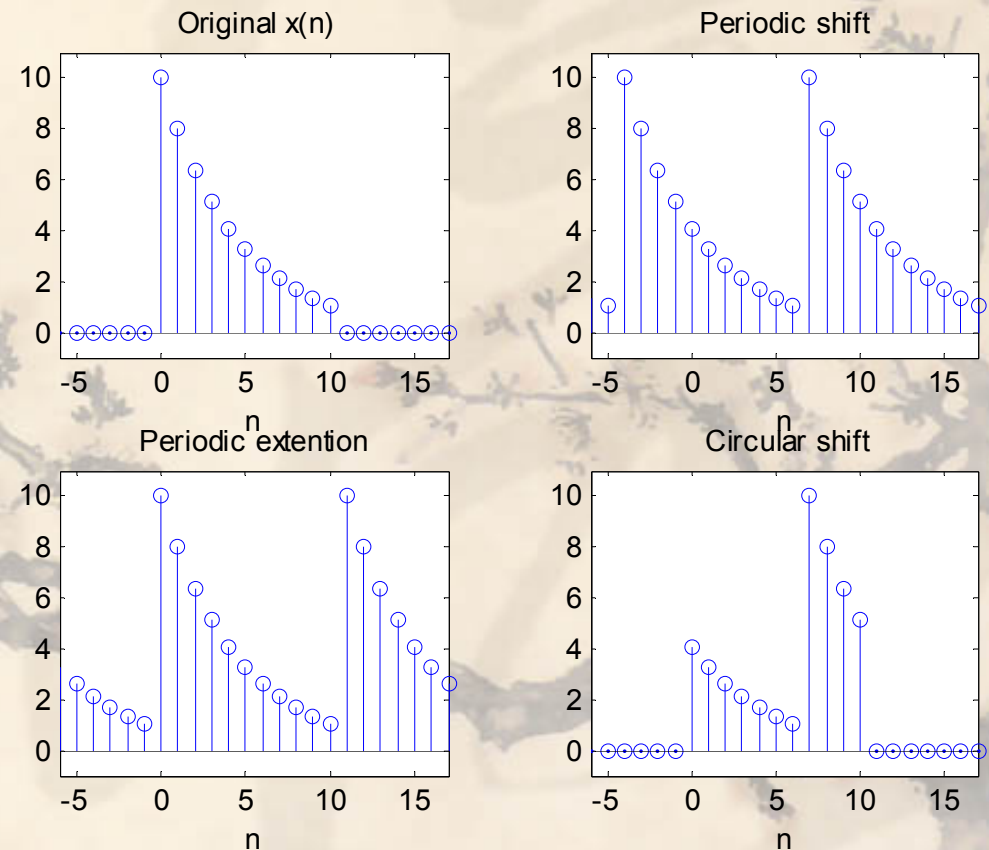
例3.12 设 $x(n) = 10(0.8)^n$, $0 \leq n \leq 10$ 为 11 点序列

a) 画出 $x((n+4))_{11}R_{11}(n)$, 也就是向左循环移位 4 个样本的序列;

b) 画出 $x((n-3))_{15}R_{15}(n)$, 也就是假定 $x(n)$ 为 15 点序列, 向右循环移位 3 个样本

解: **a)** 特别注意当样本从一个方向移出 $[0, N-1]$, 它们将从相反方向再现。这就是循环移位的含义, 它不同于线性移位。

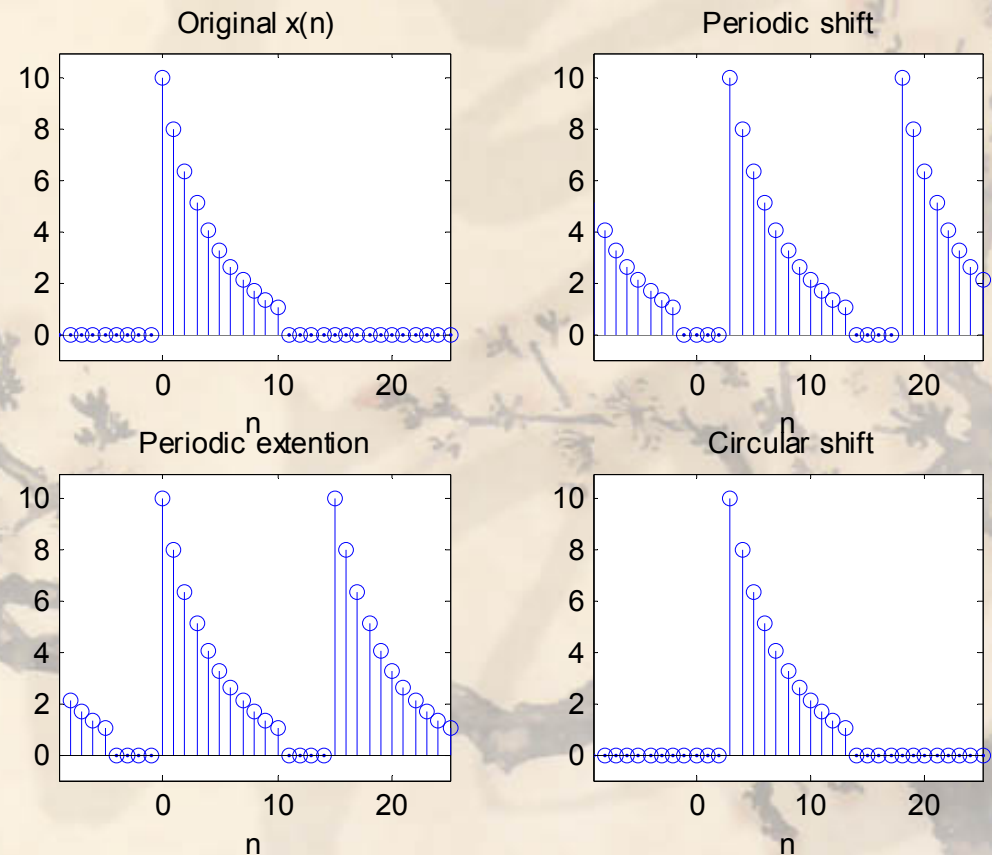
```
% Circular shift graphical display
subplot(1,1,1)
% a) plot x((n+4))11
n = 0:10; x = 10*(0.8).^ n;
n1 = -11:21; x1 = [zeros(1,11), x, zeros(1,11)];
subplot(2,2,1); stem(n1,x1); title('Original x(n)')
xlabel('n'); axis([-6,17,-1,11])
x2 = [x, x, x];
subplot(2,2,3); stem(n1,x2);
title('Periodic extention')
xlabel('n'); axis([-6,17,-1,11])
x3 = [x2(4+1:33), x(1:4)];
subplot(2,2,2); stem(n1,x3); title('Periodic shift')
xlabel('n'); axis([-6,17,-1,11])
x4 = x3 .* [zeros(1,11), ones(1,11), zeros(1,11)];
subplot(2,2,4); stem(n1,x4); title('Circular shift')
xlabel('n'); axis([-6,17,-1,11])
pause;
```



3.3.3 DFT 的性质：序列的循环移位(圆周移位)

b) 在这种情况下，给 $x(n)$ 后填充 4 个零，将其看作一个 15 点序列。此时的循环移位与 $N=11$ 时不同，看起来像线性移位 $x(n-3)$ 。因此，序列的周期长度是非常重要的一个参数。

```
% b) plot x((n-3))15
n = 0:10; x = [10*(0.8).^n zeros(1,4)];
n1 = -15:29;
x1 = [zeros(1,15), x, zeros(1,15)];
subplot(2,2,1);
stem(n1,x1);
title('Original x(n)')
xlabel('n'); axis([-9,25,-1,11])
x2 = [x, x, x];
subplot(2,2,3); stem(n1,x2);
title('Periodic extention')
xlabel('n'); axis([-9,25,-1,11])
x3 = [x2(43:45),x2(1:42)];
subplot(2,2,2); stem(n1,x3);
title('Periodic shift')
xlabel('n'); axis([-9,25,-1,11])
x4 = x3 .* [zeros(1,15), ones(1,15), zeros(1,15)];
subplot(2,2,4); stem(n1,x4);
title('Circular shift')
xlabel('n'); axis([-9,25,-1,11])
pause;
```



3.3.3 DFT 的性质：序列的循环移位(圆周移位)

❖ Matlab 循环移位函数

∞ 为了实现循环移位，可以在时域中对变量 $(n-m)$ 实行取模 N 运算。

function y = cirshfft(x,m,N)

% Circular shift of m samples wrt size N in sequence x: (time domain)

% -----

% [y] = cirshfft(x,m,N)

% y = output sequence containing the circular shift

% x = input sequence of length $\leq N$

% m = sample shift

% N = size of circular buffer

% Method: $y(n) = x((n-m) \bmod N)$

% Check for length of x

if length(x) > N

 error('N must be \geq the length of x')

end

x = [x zeros(1,N-length(x))]; %在序列 $x(n)$ 后面补零

n = [0:1:N-1];

n = mod(n-m,N); %对变量 $(n-M)$ 进行模 N 运算

y = x(n+1);

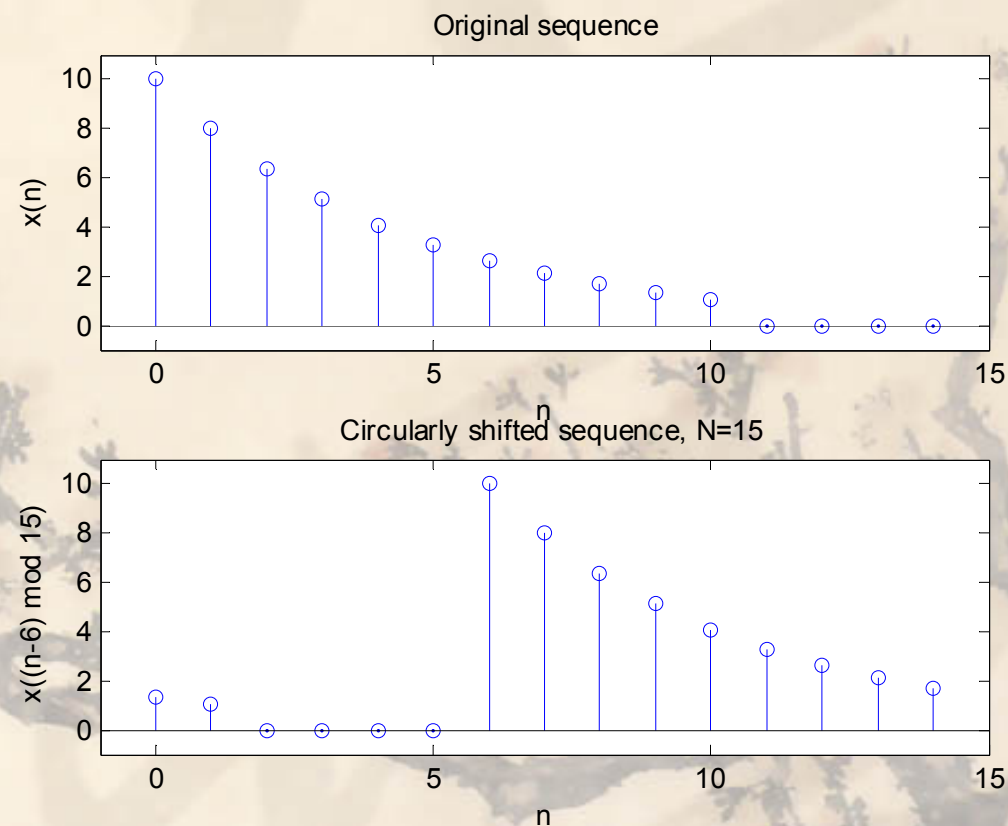
3.3.3 DFT 的性质：序列的循环移位(圆周移位)

例3.13 给定 11 点序列 $x(n) = 10(0.8)^n$, $0 \leq n \leq 10$, 求出并画出 $x((n-6))_{15}R_{15}(n)$ 。

```
% Circular shift example
subplot(1,1,1)
% a) plot x((n-6))15
n = 0:10; x = 10*(0.8).^ n;

y = cirshftt(x,6,15);

n = 0:14; x = [x, zeros(1,4)];
subplot(2,1,1); stem(n,x);
title('Original sequence')
xlabel('n'); ylabel('x(n)'); axis([-1,15,-1,11])
subplot(2,1,2); stem(n,y);
title('Circularly shifted sequence, N=15')
xlabel('n'); ylabel('x((n-6) mod 15)');
axis([-1,15,-1,11])
```



3.3.3 DFT 的性质：序列循环移位后的 DFT

若 $x(n) \xrightarrow{DFT} X(k)$ ，则

$$x_m(n) = x((n+m))_N R_N(n) \xrightarrow{DFT} W_N^{-mk} X(k)$$

证明：

$$\begin{aligned} DFT[x((n+m))_N R_N(n)] &= DFT[\tilde{x}(n+m) R_N(n)] \\ &= DFS[\tilde{x}(n+m)] R_N(k) \\ &= W_N^{-mk} \tilde{X}(k) R_N(k) \\ &= W_N^{-mk} X(k) \end{aligned}$$

结论：

有限长序列的循环移位，在离散频域中只引入了一个和频率成正比的线性相移 $W_N^{-mk} = e^{j\frac{2\pi}{N}mk}$ ，对幅频特性没有影响。

3.3.3 DFT 的性质：频域循环移位后的 IDFT

■ 频域循环移位后的 IDFT（调制特性）：

✧ 由 DFT 所具有的对偶特性不难看出，在频域内循环移位时，将有类似的结果，即：

$$x(n)e^{-j\frac{2\pi}{N}nl} = W_N^{nl} x(n) \xleftarrow{\text{IDFT}} X((k+l))_N R_N(k)$$

证明：

$$\begin{aligned} \text{IDFT}[X((k+l))_N R_N(k)] &= \text{IDFT}[\tilde{X}(k+l)R_N(k)] \\ &= \text{IDFS}[\tilde{X}(k+l)]R_N(n) \\ &= W_N^{nl} \tilde{x}(n)R_N(n) \\ &= W_N^{nl} x(n) \end{aligned}$$

时域序列的调制等效于频域的循环移位

3.3.3 DFT 的性质：频域循环移位后的 IDFT

$$DFT \left[x(n) \cos \left(\frac{2\pi nl}{N} \right) \right] = \frac{1}{2} [X((k-l))_N + X((k+l))_N] R_N(k)$$

$$DFT \left[x(n) \sin \left(\frac{2\pi nl}{N} \right) \right] = \frac{1}{2j} [X((k-l))_N - X((k+l))_N] R_N(k)$$

证明：

$$\begin{aligned} & IDFT \left\{ \frac{1}{2j} [X((k-l))_N - X((k+l))_N] R_N(k) \right\} \\ &= \frac{1}{2j} [W_N^{-nl} x(n) - W_N^{nl} x(n)] = \frac{e^{j\frac{2\pi}{N}nl} - e^{-j\frac{2\pi}{N}nl}}{2j} x(n) \\ &= x(n) \sin \frac{2\pi nl}{N} \end{aligned}$$

3.3.3 DFT 的性质：循环卷积(圆周卷积)

❖ 循环卷积定义：

∞ 设 $x(n)$ 和 $h(n)$ 都是长度为 N 的有限长序列，把它们分别拓展为周期序列 $\tilde{x}(n)$ 和 $\tilde{h}(n)$ ，定义循环卷积为：

$$y_c(n) = x(n) \otimes h(n) \\ = \left[\sum_{m=0}^{N-1} \tilde{x}(m) \tilde{h}(n-m) \right] R_N(n)$$

——周期序列卷积后取主值

3.3.3 DFT 的性质：循环卷积(圆周卷积)

- 因为上式的求和范围是 m 由 0 到 $N-1$ ，因此第一个序列 $x(m)$ 可以不作周期拓展，即

$$y_c(n) = \left[\sum_{m=0}^{N-1} \underbrace{x(m)}_{\text{有限长序列}} \underbrace{h((n-m))_N}_{\text{周期序列}} \right] R_N(n)$$

窗函数限定了循环卷积的范围

- 注意两个 N 点序列的线性卷积将导致一个更长的序列。而循环卷积将区间限制在 $0 \leq n \leq N-1$ ，结果仍为 N 点序列，它与线性卷积的结构类似。不同点在于求和范围和 N 点循环移位。它与 N 有关，也叫做 N 点循环卷积。

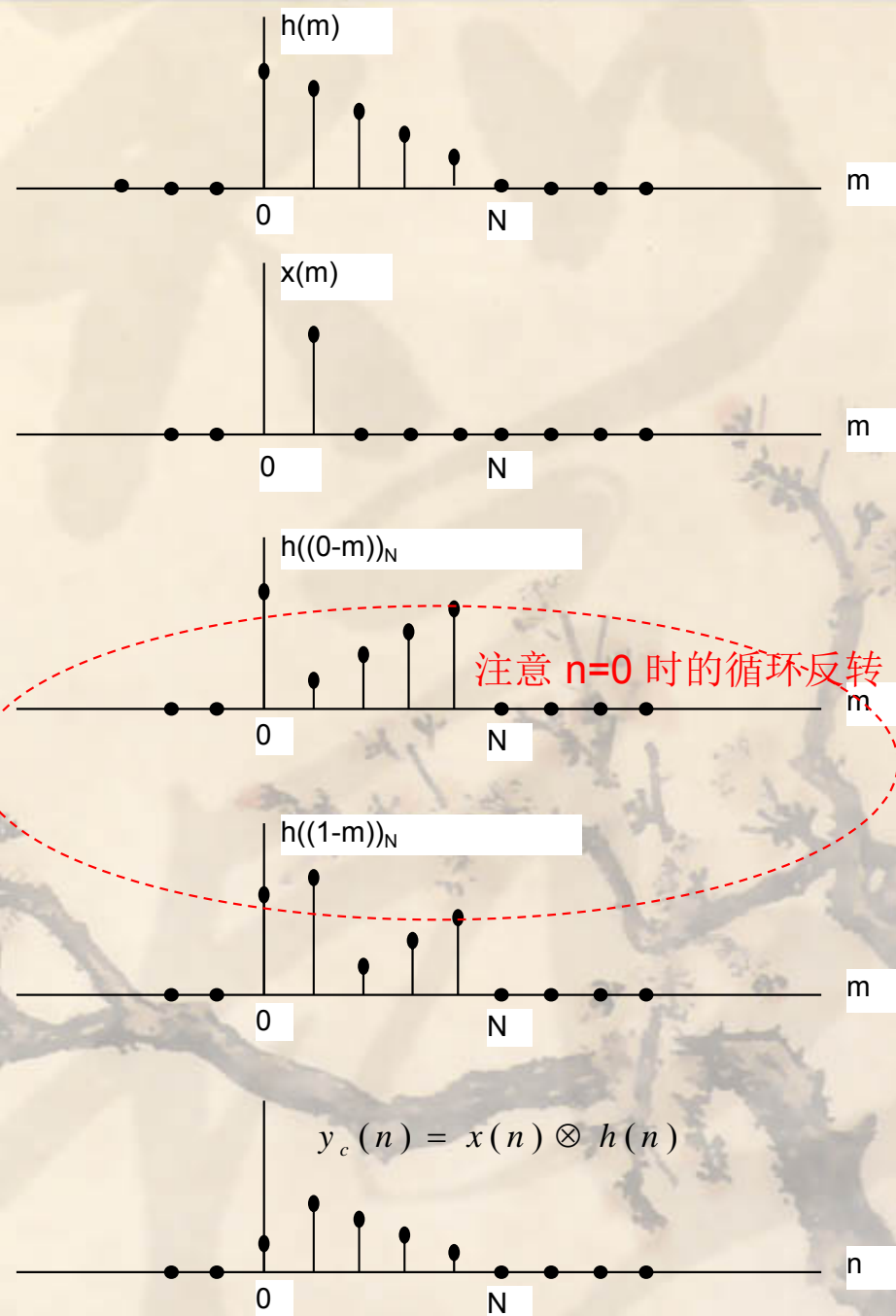
用 \odot_N 或 \otimes 表示循环卷积

3.3.3 DFT 的性质：循环卷积(圆周卷积)

$$y_c(n) = \left[\sum_{m=0}^{N-1} \underbrace{x(m)}_{\text{有限长序列}} \underbrace{h((n-m))_N}_{\text{周期序列}} \right] R_N(n)$$

❖ 循环卷积过程：

- 1) 补零
- 2) 周期延拓
- 3) 反转，取主值序列
(循环反转)
- 4) 对应位相乘，然后求和，
得到 $n=0$ 时的卷积结果。
- 5) 向右**循环移位**（圆周移位）
- 6) 重复， $n=1 \sim (N-1)$



例：设 $x_1(n) = \{1, 2, 3, 4, 5\}$, $x_2(n) = \{6, 7, 8, 9\}$, 计算5点循环卷积 $x_1(n) \otimes x_2(n)$

$$x_1(n) \otimes x_2(n) = \sum_{m=0}^4 x_1(m) x_2((n-m))_5$$

当 $n = 0$

$$\sum_{m=0}^4 x_1(m) x_2((0-m))_5 = \sum_{m=0}^4 [\{1, 2, 3, 4, 5\} * \{6, 0, 9, 8, 7\}]$$

$$= \sum_{m=0}^4 \{6, 0, 27, 32, 35\} = 100$$

$n = 1$

$$\sum_{m=0}^4 x_1(m) x_2((1-m))_5 = \sum_{m=0}^4 [\{1, 2, 3, 4, 5\} * \{7, 6, 0, 9, 8\}]$$

$$= \sum_{m=0}^4 \{7, 12, 0, 36, 40\} = 95$$

$$\begin{aligned}
 n = 2 \quad \sum_{m=0}^4 x_1(m) x_2((2-m))_5 &= \sum_{m=0}^4 [\{1,2,3,4,5\} * \{8,7,6,0,9\}] \\
 &= \sum_{m=0}^4 \{8,14,18,0,45\} = 85
 \end{aligned}$$

$$\begin{aligned}
 n = 3 \quad \sum_{m=0}^4 x_1(m) x_2((3-m))_5 &= \sum_{m=0}^4 [\{1,2,3,4,5\} * \{9,8,7,6,0\}] \\
 &= \sum_{m=0}^4 \{9,16,21,24,0\} = 70
 \end{aligned}$$

$$\begin{aligned}
 n = 4 \quad \sum_{m=0}^4 x_1(m) x_2((4-m))_5 &= \sum_{m=0}^4 [\{1,2,3,4,5\} * \{0,9,8,7,6\}] \\
 &= \sum_{m=0}^4 \{0,18,24,28,30\} = 100
 \end{aligned}$$

3.3.3 DFT 的性质：循环卷积(圆周卷积)

❖ 循环卷积的时频映射关系

■ 由 **DTFT** 的性质可知，两个序列时域上的线性卷积运算在频域上表现为两个序列 **DTFT** 结果的乘积。

■ 同样的，若

$$x_1(n) \xleftrightarrow{DFT} X_1(k), \quad x_2(n) \xleftrightarrow{DFT} X_2(k)$$

■ 则

$$x_1(n) \otimes x_2(n) \xleftrightarrow{DFT} X_1(k)X_2(k)$$

即当在频域中进行两个 N 点 DFT 相乘时，在时域中映射为循环卷积（而不是通常的线性卷积！！！！）。

3.3.3 DFT 的性质：循环卷积(圆周卷积)

证明：

将 $X_1(k)$ 和 $X_2(k)$ 作周期延拓，分别得到 $X_1((k))_N$ 和 $X_2((k))_N$ 。

令 $Y(k) = X_1(k) \cdot X_2(k)$ 则 $Y((k))_N = X_1((k))_N \cdot X_2((k))_N$

再设 $y((n))_N \xleftrightarrow{DFS} Y((k))_N$

于是

$$\begin{aligned} y((n))_N &= \frac{1}{N} \sum_{k=0}^{N-1} Y((k))_N W^{-kn} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \left[\sum_{m=0}^{N-1} x_1((m))_N W^{km} \right] \left[\sum_{r=0}^{N-1} x_2((r))_N W^{kr} \right] W^{-kn} \\ &= \sum_{m=0}^{N-1} x_1((m))_N \sum_{r=0}^{N-1} x_2((r))_N \left[\frac{1}{N} \sum_{k=0}^{N-1} W^{-k(n-m-r)} \right] \end{aligned}$$

3.3.3 DFT 的性质：循环卷积(圆周卷积)

因为

$$\frac{1}{N} \sum_{k=0}^{N-1} W^{-k(n-m-r)} = \begin{cases} 1 & r = n - m \\ 0 & \text{其它} \end{cases}$$

所以

$$y((n)_N) = \sum_{m=0}^{N-1} x_1((m)_N) \sum_{r=0}^{N-1} x_2((n-m)_N)$$

此时上式最后一个求和号中，已不对 r 求和，故此求和号可以去掉，因此，

$$y((n)_N) = \sum_{m=0}^{N-1} x_1((m)_N) x_2((n-m)_N)$$

因而，

$$y(n) = y((n)_N) R_N(n) = x_1(n) \otimes x_2(n)$$

循环卷积即为周期序列卷积后取主值

即

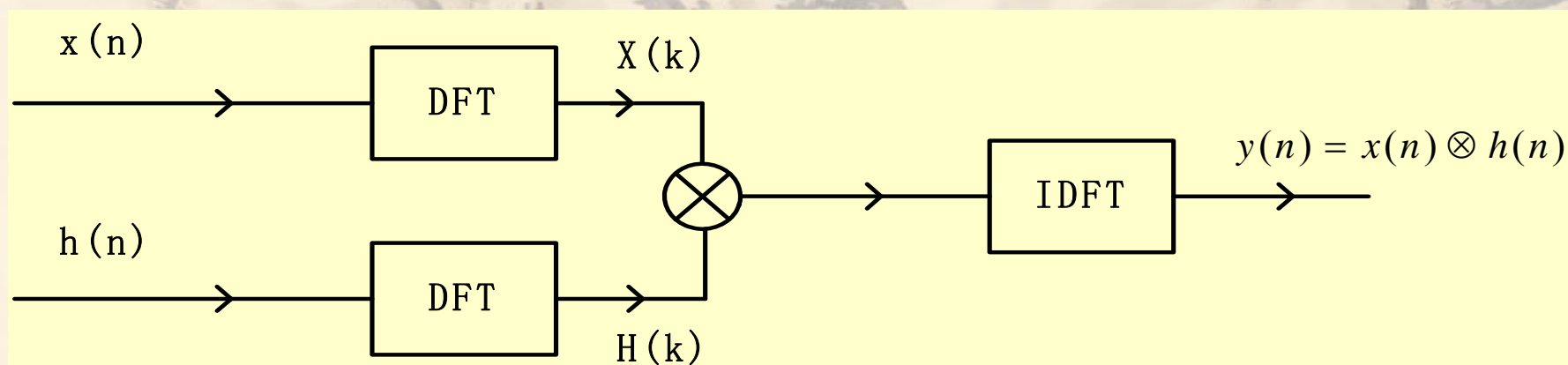
$$x_1(n) \otimes x_2(n) \xleftarrow{IDFT} X_1(k) X_2(k)$$

3.3.3 DFT 的性质：循环卷积(圆周卷积)

❖ 利用时域、频域的对偶性可得频域循环卷积：

若 $y(n) = x_1(n) \cdot x_2(n)$
则
$$Y(k) = DFT[y(n)] = \sum_{n=0}^{N-1} y(n) W_N^{nk}$$
$$= \frac{1}{N} \left[\sum_{l=0}^{N-1} X_1(l) X_2((k-l))_N \right] R_N(k)$$
$$= \frac{1}{N} \left[\sum_{l=0}^{N-1} X_2(l) X_1((k-l))_N \right] R_N(k)$$

■ 时域循环卷积可在频域中完成：



$$\begin{aligned}
 y_c(n) &= \left[\sum_{m=0}^{N-1} \underbrace{x(m)}_{\text{有限长序列}} \underbrace{h((n-m))_N}_{\text{周期序列}} \right] R_N(n) \\
 &= \left[\sum_{m=0}^{N-1} x(m) \tilde{h}(n-m) R_N(n) \right] \quad \tilde{h}(n) = h((n))_N \\
 &= \left[\sum_{m=0}^{N-1} \tilde{x}(m) \tilde{h}(n-m) \right] R_N(n) \\
 &= \tilde{y}_c(n) R_N(n)
 \end{aligned}$$

$y_c(n)$ 是 $\tilde{x}(n)$ 和 $\tilde{h}(n)$ 周期卷积的主值序列

3.3.4 DFT 的应用

❖ 利用 DFT 求离散线性卷积

- 条件: $N \geq N_1 + N_2 - 1$

- 方法:

 - 重叠相加法

 - 重叠保留法

- 特点: 以逐段的方式通过循环卷积计算线性卷积, 运算量小。

❖ 用 DFT 进行信号频谱分析

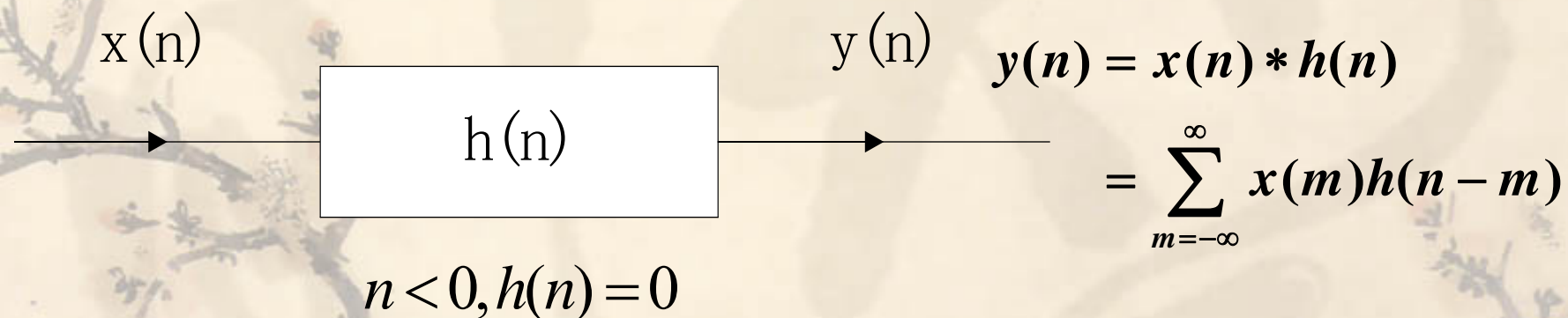
- 参数选择: 频谱混叠

- 频谱泄漏

- 栅栏效应

3.3.4 DFT 的应用：线性卷积求解

■ 线性移不变离散系统



当长度 N 增大时，线性卷积的运算量较大，需寻找实用的线性卷积的快速计算方法！

■ 循环卷积的时频映射

$$x(n) \otimes h(n) \xleftarrow{IDFT} X(k)H(k)$$

快速算法
FFT

频域中进行两个 N 点 DFT 相乘时，在时域中映射为循环卷积（而不是通常的线性卷积 !!!）。

3.3.4 DFT 的应用：线性卷积求解

■ 循环卷积和线性卷积的关系

$$\begin{aligned} \text{设: } x(n) \quad 0 \leq n \leq N-1 \\ h(n) \quad 0 \leq n \leq M-1 \end{aligned} \quad \text{令 } N \geq \max[N, M]$$

N点循环卷积:

$$\begin{aligned} y_c(n) &= x(n) \otimes h(n) = \left[\sum_{m=0}^{N-1} x(m) h((n-m))_N \right] R_N(n) \\ &= \left[\sum_{m=0}^{N-1} h(m) x((n-m))_N \right] R_N(n) = h(n) \otimes x(n) \end{aligned}$$

线性卷积:

$$\begin{aligned} y_L(n) &= x(n) * h(n) = \sum_{m=-\infty}^{\infty} x(m) h(n-m) \\ &= \sum_{m=-\infty}^{\infty} h(m) x(n-m) = h(n) * x(n) \end{aligned}$$

3.3.4 DFT 的应用：线性卷积求解

✧ 讨论线性卷积 $y(n)=x(n)*h(n)$ 的长度

从 $x(m)$ 看，非零值区为： $0 \leq m \leq N-1$

从 $h(n-m)$ 看，非零值区为： $0 \leq n-m \leq M-1$

将二不等式相加，得到 $y(n)$ 的非零区：

$$0 \leq n \leq M+N-2$$

在此区间之外，不是 $x(m)=0$ ，就是 $h(n-m)=0$ ，即 $y(n)=0$ ，
因此， $y(n)$ 的长度为

$$\underline{M+N-1}$$

3.3.4 DFT 的应用：线性卷积求解

✧ 讨论循环卷积的长度

循环卷积是周期卷积的主值序列。为了研究长度不等的两个序列的周期卷积，**构造周期序列**，使它们的长度相等，且均为 L 。

$$\tilde{x}(n) \quad \overbrace{x(0), \cdots \cdots, x(N-1), 0, 0, \cdots, 0}^L$$

$$\tilde{h}(n) \quad \overbrace{h(0), \cdots \cdots, h(M-1), 0, 0, \cdots, 0}^L$$

表示为

$$\tilde{x}(n) = \sum_{q=-\infty}^{\infty} x(n+qL), \quad \tilde{x}(n) = x(n), \quad 0 \leq n \leq N-1$$

$$\tilde{h}(n) = \sum_{r=-\infty}^{\infty} h(n+rL), \quad \tilde{h}(n) = h(n), \quad 0 \leq n \leq M-1$$

3.3.4 DFT 的应用：线性卷积求解

✧ 周期卷积如下：

$$\begin{aligned}\tilde{y}(n) &= \sum_{m=0}^{L-1} \tilde{x}(m) \tilde{h}(n-m) = \sum_{m=0}^{L-1} \overbrace{\sum_{q=-\infty}^{\infty} x(m+qL)}^{\text{只取一个周期}} \sum_{r=-\infty}^{\infty} h(n-m+rL) \\ &= \sum_{r=-\infty}^{\infty} \underbrace{\sum_{m=0}^{L-1} x(m) h(n+rL-m)}_{\text{线性卷积的表达式}} \\ &= \sum_{r=-\infty}^{\infty} y(n+rL)\end{aligned}$$

因为，即 $0 \leq m \leq L-1$ 范围内 $x(m+qL)$ 只有一个周期。

- 周期卷积是线性卷积的周期延拓，周期为 L 。
- 当周期为 $L \geq N+M-1$ 时，不会发生混迭，线性卷积正好是周期卷积的一个周期。
- 循环卷积又是周期卷积的主值序列。

3.3.4 DFT 的应用：线性卷积求解

由于循环卷积是周期卷积的主值序列。因此，此时循环卷积与线性卷积完全相同。即：

$$\begin{aligned} y_c(n) &= x(n) \otimes h(n) = \tilde{y}(n) R_N(n) \\ &= y(n) = \sum_{m=0}^{L-1} x(m) h(n-m) \quad 0 \leq n \leq L-1 \end{aligned}$$

■ 因此，循环卷积等于线性卷积的条件是：

$$L \geq M + N - 1$$

即对于 N 长度的 $x(n)$ ， M 长度的 $h(n)$ ，在它们后面补零，使其长度均变为 $L \geq M+N-1$ 。

$$L = N$$

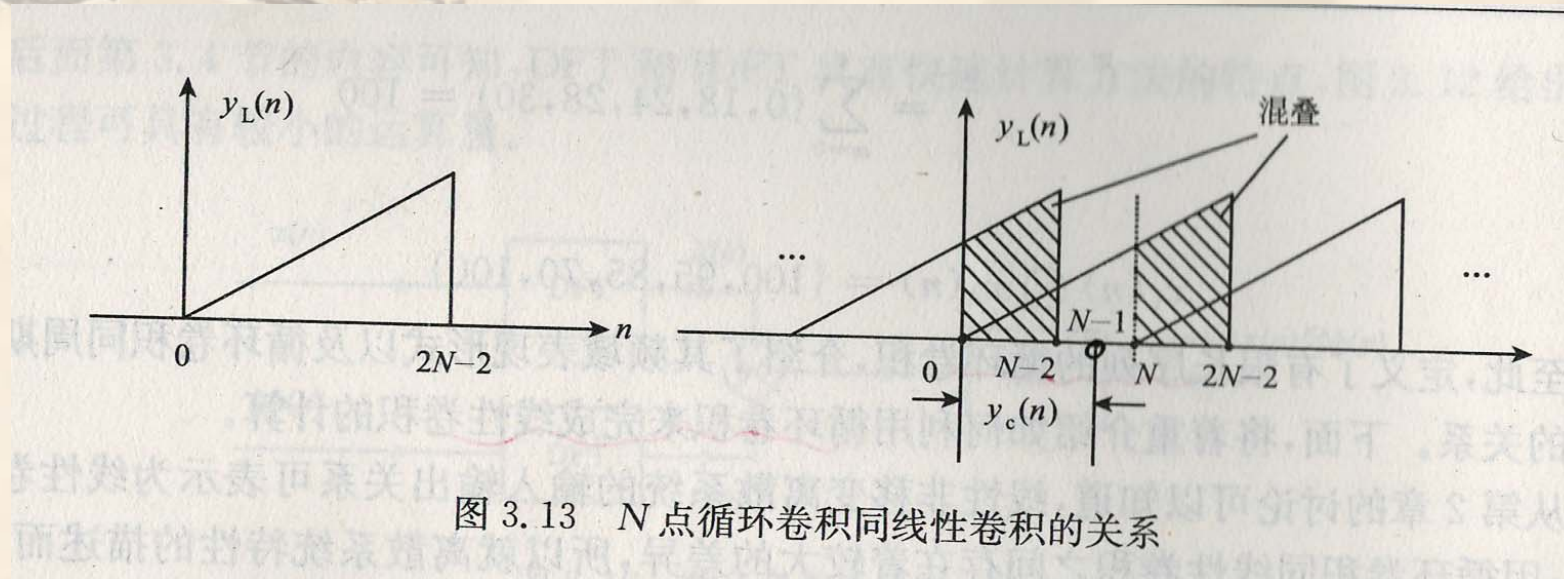


图 3.13 N 点循环卷积同线性卷积的关系

$$y_c(N-1) = y_L(N-1)$$

$$N_1 + N_2 - 1 \quad N_1 \quad N_2 \quad N_1 > N_2$$

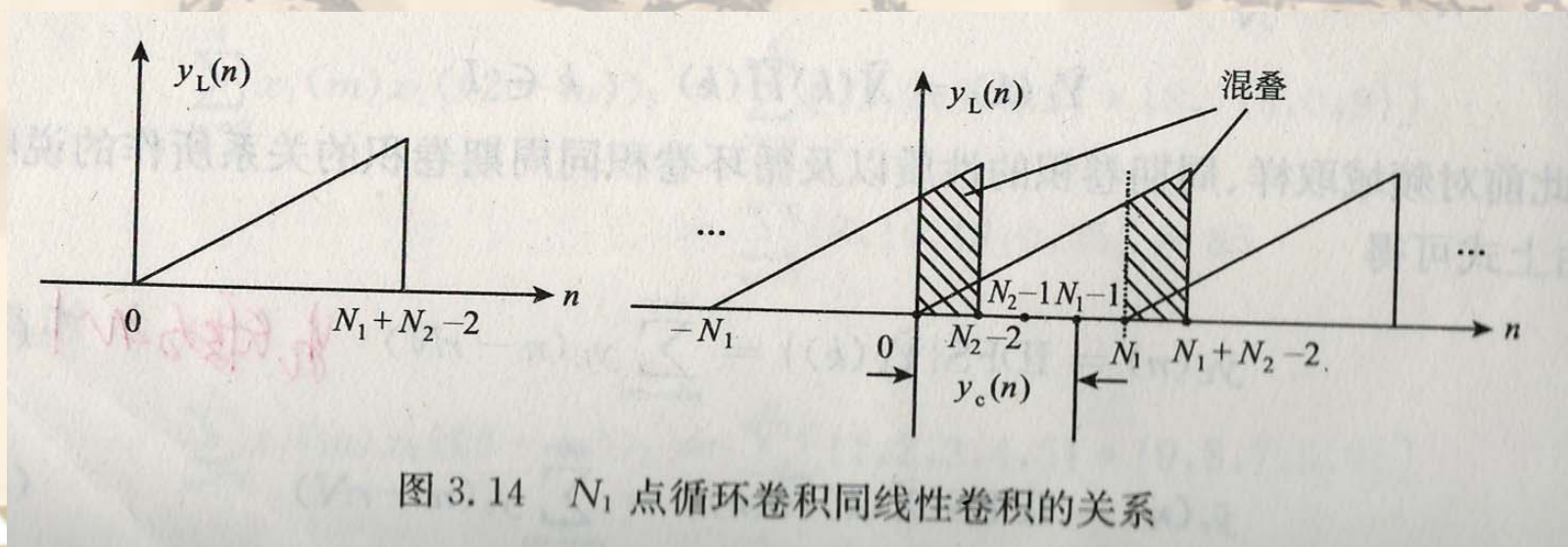
$$y_c(n) = \sum_{m=0}^{N_1-1} x(m)h_1((n-m))_{N_1}, n = 0, 1, \dots, N_1 - 1$$

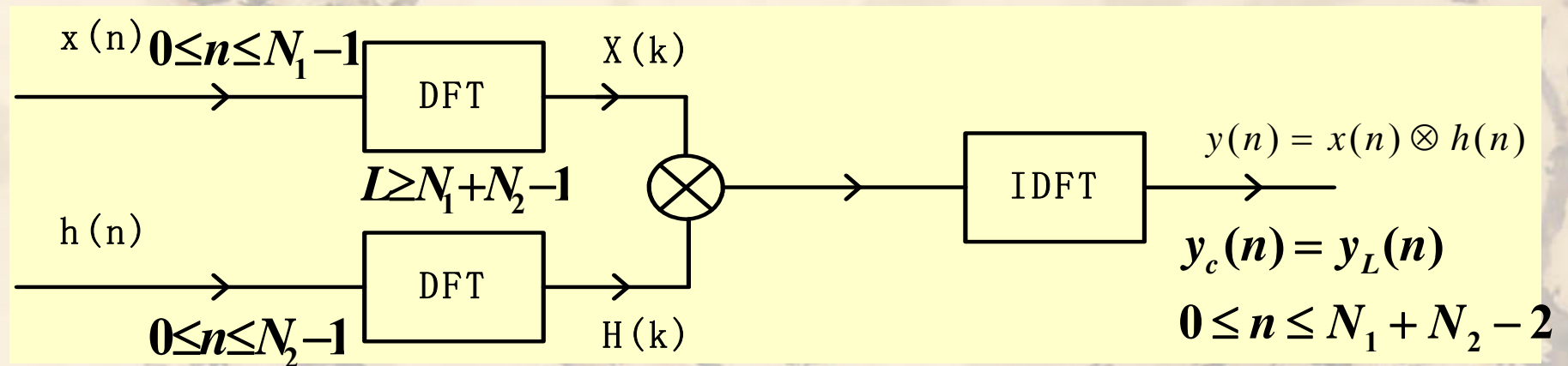
$$h_1(n) = \begin{cases} h(n) & n = 0, 1, \dots, N_2 - 1 \\ 0 & n = N_2, N_2 + 1, \dots, N_1 - 1 \end{cases}$$

$\tilde{y}_L(n)$ 主值序列前 $N_2 - 1$ 点存在混叠

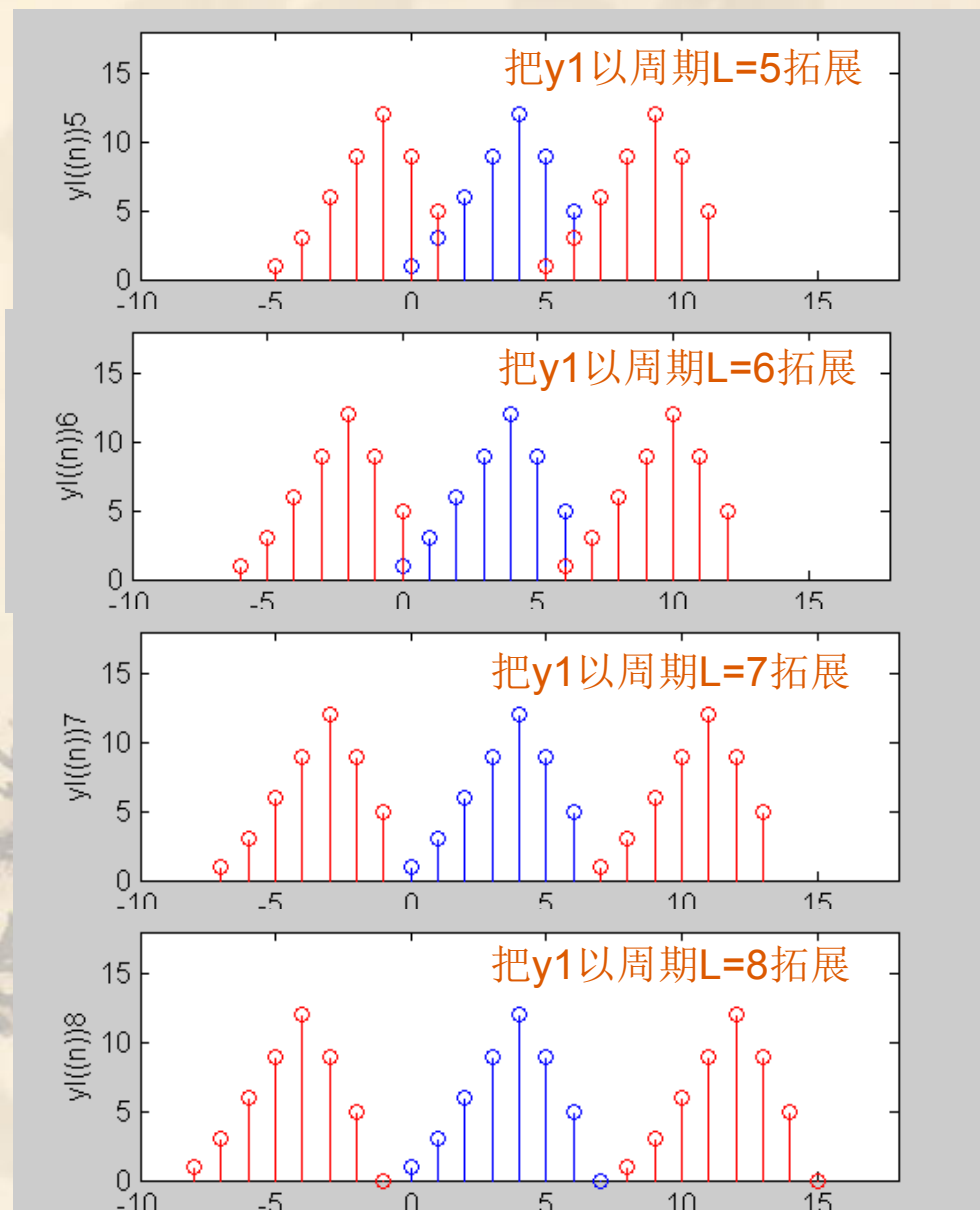
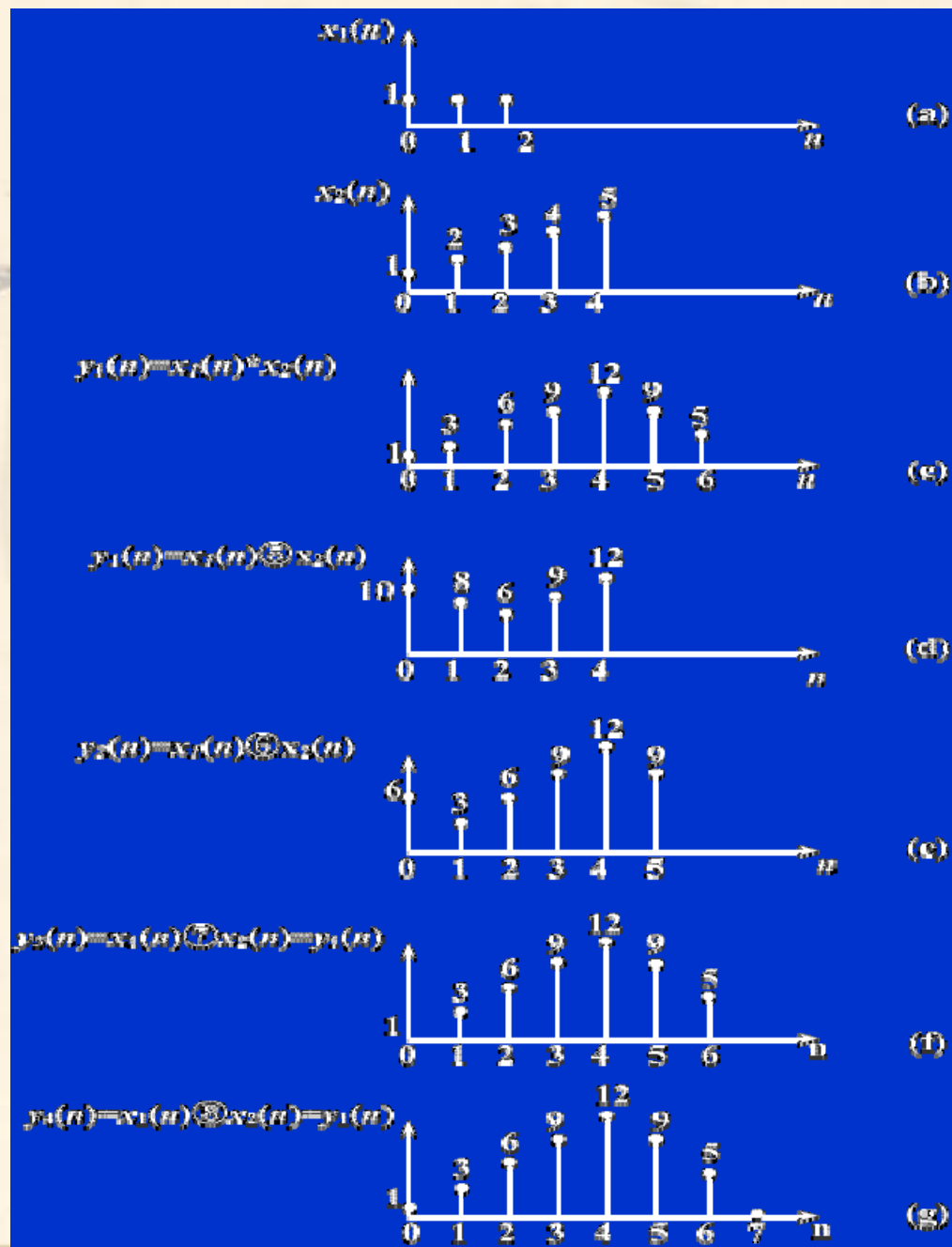
$$y_c(l) = y_L(l), l = N_2 - 1, N_2, \dots, N_1 - 1$$

即 $y_c(n)$ 中最后 $N_1 - N_2 + 1$ 点与 $y_L(n)$ 结果相同





3.3.4 DFT 的应用：线性卷积求解



三种卷积运算对比

	线性卷积	周期卷积	循环卷积
要求	无	两个周期相同的周期序列	有限长序列
相关变换	DTFT (z变换)	DFS	DFT
求和范围	由两个序列的长度和所在区间决定 N+M-1	周期为 N 的周期序列	一个主值周期 N
计算方法	$x(n) * h(n) = \sum_{m=-\infty}^{\infty} x(m)h(n-m)$	$\tilde{y}(n) = \sum_{m=0}^{N-1} \tilde{x}_1(m)\tilde{x}_2(n-m)$	$x(n) \otimes h(n) = \tilde{y}(n)R_N(n)$ $= \sum_{m=0}^{L-1} x(m)h(n-m) \quad 0 \leq n \leq L-1$

联系:

- 周期卷积是线性卷积的周期延拓, 周期为 L 。
- 当周期为 $L \geq N+M-1$ 时, 不会发生混迭, 线性卷积正好是周期卷积的一个周期。
- 循环卷积又是周期卷积的主值序列

三种卷积方法的联系



3.3.4 DFT 的应用：线性卷积求解

■ 用 DFT 求解线性卷积

设 $x(n) \xleftrightarrow{DFT} X(k), h(n) \xleftrightarrow{DFT} H(k)$

若 $y(n) = x(n) \otimes h(n) \xleftrightarrow{DFT} Y(k) = X(k)H(k)$

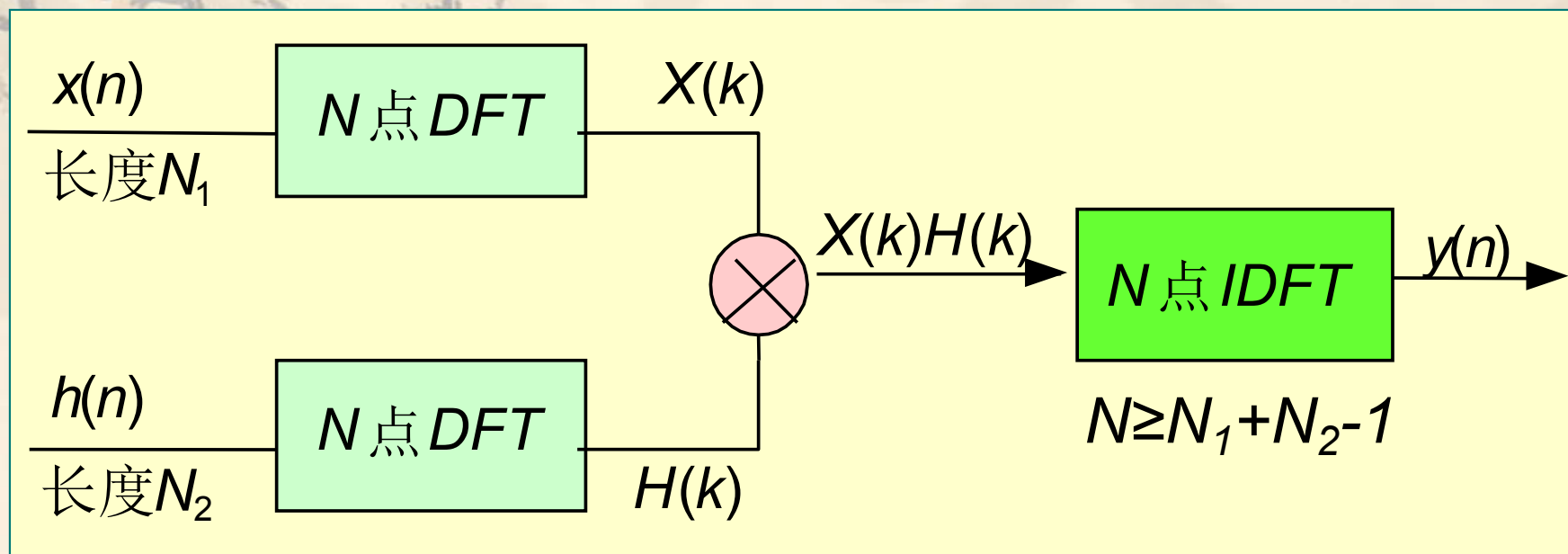
即时域循环卷积 $\xrightarrow{DFT \text{ 映射}}$ 频域相乘

则

$$\begin{aligned} y(n) &= IDFT [Y(k)] \\ &= IDFT [X(k)H(k)] \end{aligned}$$

3.3.4 DFT 的应用：线性卷积求解

- 如果循环卷积的长度 N 满足 $N \geq N_1 + N_2 - 1$ ，则此循环卷积 $Y(k)$ 就等于 $x(n)$ 和 $h(n)$ 的线性卷积。用流程图表示法求 $y(n)=x(n)*h(n)$ 的过程如下：



- 因为 DFT、IDFT 都有快速算法，因此，线性卷积可以实现快速算法。

3.3.4 DFT 的应用：线性卷积求解

❖ 乘法次数

- 循环卷积：在上述 DFT 求解线性卷积过程中，需要 3 次 DFT（FFT）运算，后面我们会讲到 N 点 FFT 所需要的乘法次数为 $\frac{N}{2} \log_2 N$ ，因此，用 DFT 求线性卷积所需要的总的乘法次数：

$$M_c = N + 3\left[\frac{N}{2} \log_2 N\right] = N\left(1 + \frac{3}{2} \log_2 N\right) = N\left(\frac{3}{2} \log_2 N - \frac{1}{2}\right)$$

- 线性卷积：每一个输入值 $x(n)$ 都必须和全部的 $h(n)$ 值乘一次，因此，总共需要 $N_1 N_2$ 次乘法运算， $M_L = N_1 N_2$ 。

$$C_m = \frac{M_L}{M_c} = \frac{N_1 N_2}{(N_1 + N_2 - 1)(1.5 \log_2 (N_1 + N_2 - 1) - 0.5)}$$

若 $N_1 = N_2$ 时，则

$$C_m \approx \frac{N_1^2}{2N_1(1.5 \log_2 2N_1 - 0.5)} = \frac{N_1}{2(2.5 + \log_2 N_1)}$$

3.3.4 DFT 的应用：线性卷积求解

$$N_1 \gg N_2, \quad N_1 + N_2 - 1 \approx N_1$$

$$C_m = \frac{N_1 N_2}{(N_1 + N_2 - 1)(1.5 \log_2 (N_1 + N_2 - 1) - 0.5)}$$
$$\approx \frac{N_1 N_2}{N_1 \times 1.5 \times \log_2 N_1} = \frac{N_2}{1.5 \log_2 N_1}$$

✧ **结论：** $N_1=N_2$ 越长，循环卷积的优越性越大。但当其中一个序列很长时， C_m 下降，循环卷积快速算法的优点不能发挥出来。

■ 克服的办法：分段卷积

- ✧ 将长序列分段，每一段分别与短序列进行循环卷积（即用 FFT 运算）
- ✧ 分为重叠相加法，重叠保留法。

3.3.4 DFT 的应用：线性卷积求解

❖ 线性卷积求解小结：

■ 时域直接求解

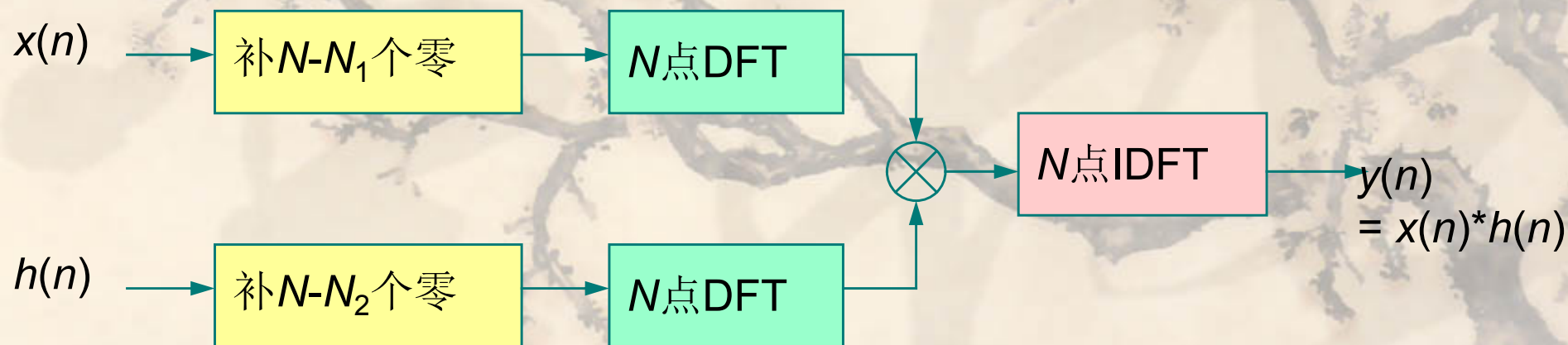
$$y(n) = x(n) * h(n) = \sum_{m=-\infty}^{\infty} x(m)h(n-m)$$

■ z 变换法

$$X(z) = Z[x(n)] \quad H(z) = Z[h(n)]$$

$$y(n) = Z^{-1}[Y(z)] = Z^{-1}[X(z) \cdot H(z)]$$

■ DFT 法



3.3.4 DFT 的应用：分段卷积

- 上面介绍的是两个有限长序列 $x_1(n)$ 、 $x_2(n)$ 的线性卷积。但有时其中某个序列会很长或无限长，若等长序列存储或输入完后再做卷积运算，将产生问题：

- (1) 存储量过大，运算量也太大；
- (2) 等待输入的时间很长，引起不能忍受的延迟；
- (3) 采用 DFT/FFT 快速算法的效率降低

- 解决此问题的思路：

把长序列分段，每一分段分别与短序列进行卷积——分段卷积。

具体方法：重叠保留法、重叠相加法

3.3.4 DFT 的应用：重叠相加法

❖ 重叠相加法—对输出 $y(n)$ 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_1 点序列, 并且 $M < N_1$;
- 将每段序列和 $h(n)$ 进行线性卷积, 然后再将结果 $y(n)$ 重叠相加。
- 问题: 重叠部分如何处理?

3.3.4 DFT 的应用：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_1 ，即把 $x(n)$ 分为 $x_0(n), x_1(n), \dots, x_k(n), \dots$ 表示为：

$$x_k(n) = \begin{cases} x(n) & KN_1 \leq n \leq (k+1)N_1 - 1 \\ 0 & \text{else} \end{cases}$$

\therefore 反过来

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

3.3.4 DFT 的应用：重叠相加法

(2) 因此，

$$\begin{aligned} y(n) &= x(n) * h(n) = \left[\sum_{k=-\infty}^{\infty} x_k(n) \right] * h(n) \\ &= \sum_{k=-\infty}^{\infty} \underbrace{[x_k(n) * h(n)]}_{\text{第}k\text{段线性卷积}} = \sum_{k=-\infty}^{\infty} y_k(n) \end{aligned}$$

$\because x_k(n)$ 长为 N_1 , $h(n)$ 长为 M

$\therefore y_k(n)$ 的长为: $N = N_1 + M - 1$

因此, $y_k(n)$ 比 $x_k(n)$ 长, 多余的点数为

$$N - N_1 = N_1 + M - 1 - N_1 = M - 1$$

3.3.4 DFT 的应用：重叠相加法

由于 $y_k(n)$ 的范围为：

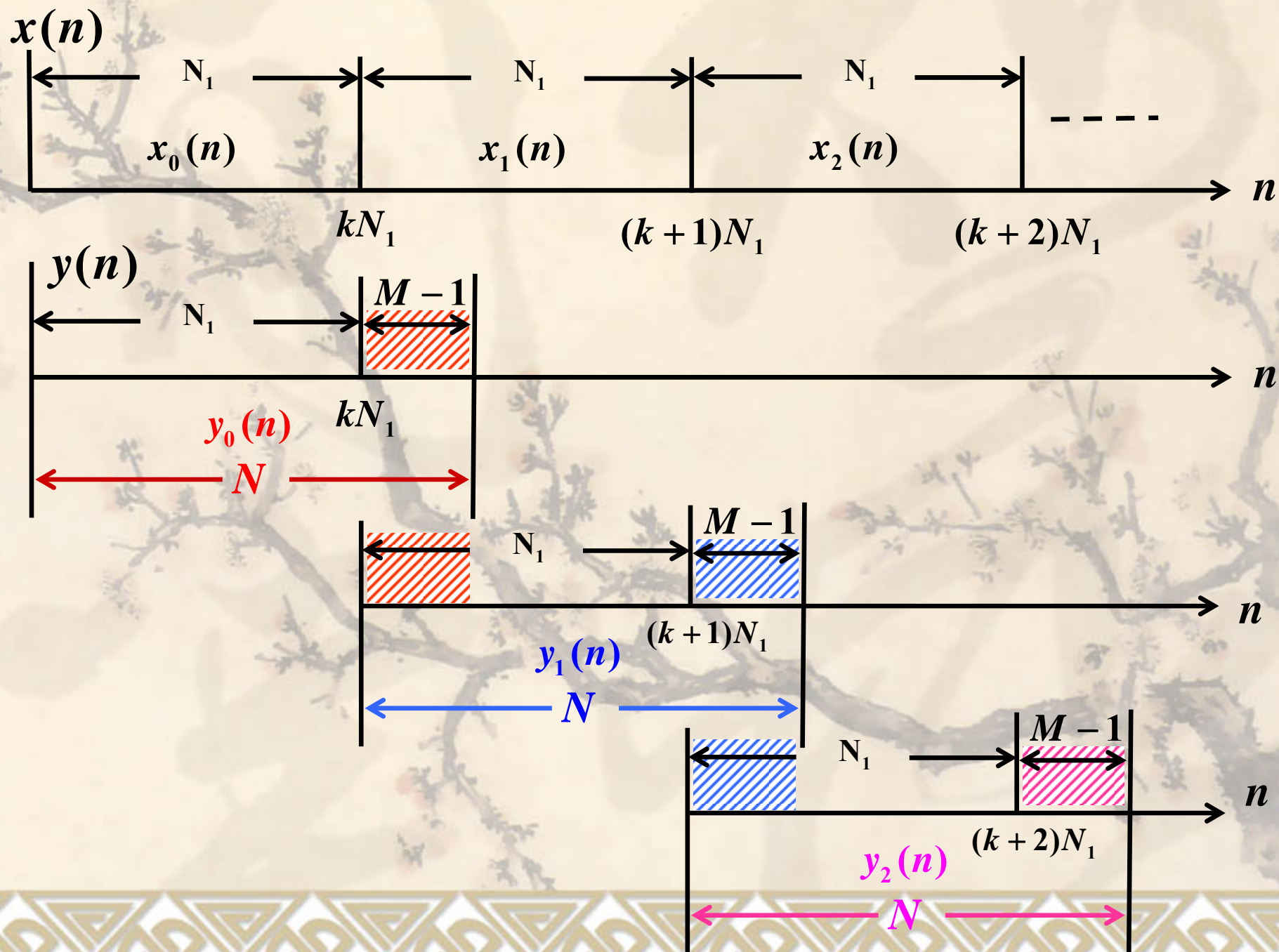
$$kN_1 \leq n \leq (k+1)N_1 + M - 2$$

而 $y_{k+1}(n)$ 的范围是：

$$(k+1)N_1 \leq n \leq (k+2)N_1 + M - 2$$

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

3.3.4 DFT 的应用：重叠相加法

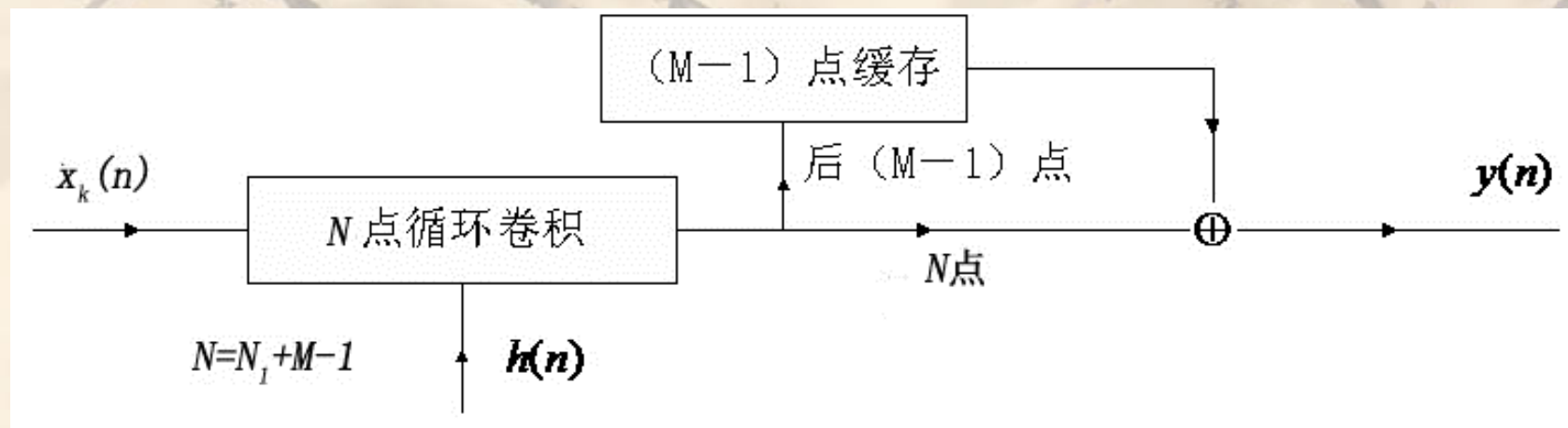


3.3.4 DFT 的应用：重叠相加法

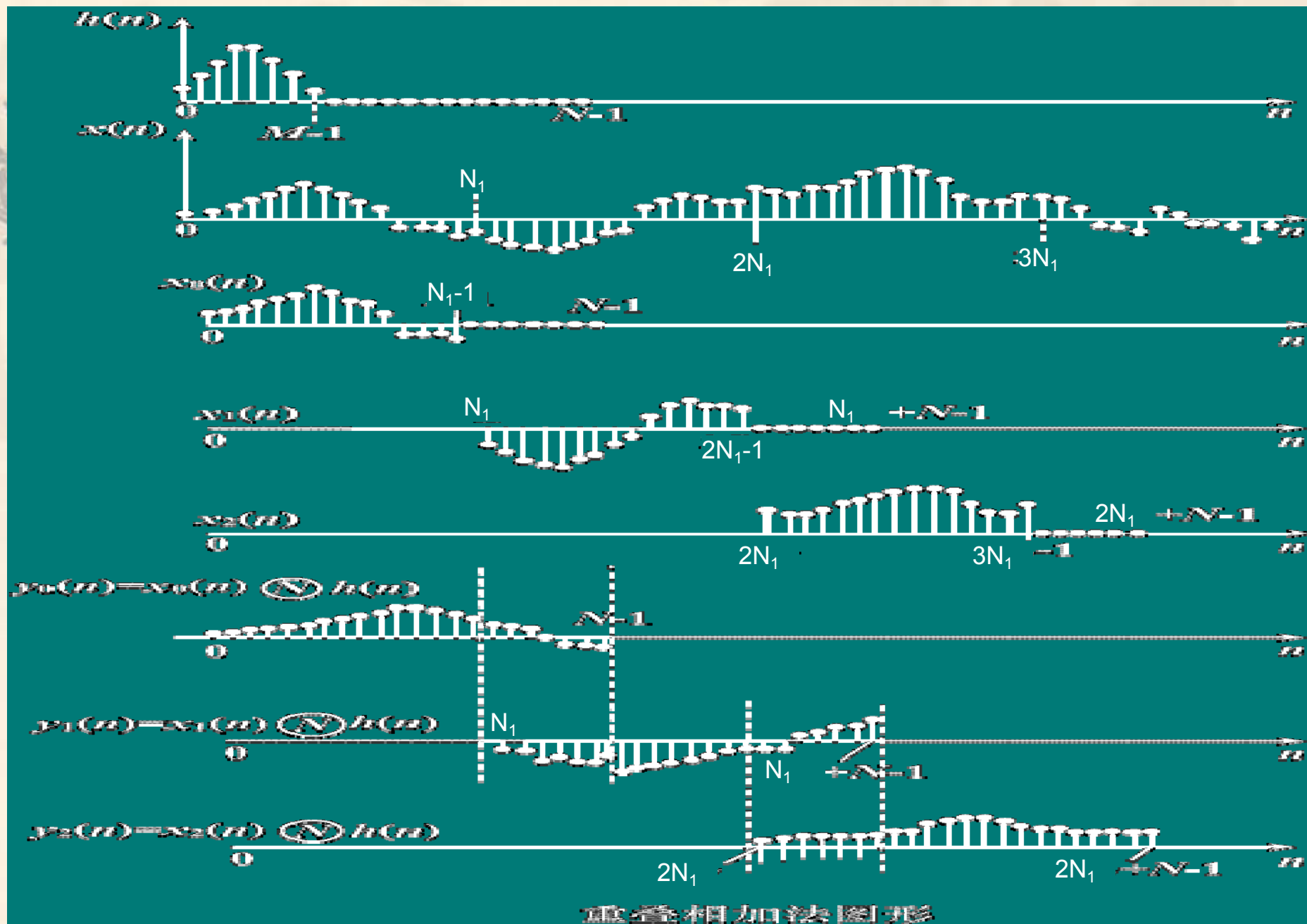
(3) 对于在 $(k+1)N_1$ 到 $(k+1)N_1+M-2$ 的范围内（即重叠部分）的每一个 n 值，原序列 $x(n)$ 与 $h(n)$ 的线性卷积之值 $y(n)$ 应为：

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

- ❖ 这种方法把 $y_k(n)$ 最后的 $(M-1)$ 项与 $y_{k+1}(n)$ 最开始的 $(M-1)$ 项相加起来得到 $y((k+1)N_1), \dots, y((k+1)N_1+M-2)$ 各项，所以称为重叠相加法。



3.3.4 DFT 的应用：重叠相加法



3.3.4 DFT 的应用：重叠相加法

例 3.21 设 $x(n) = (n+1)$, $0 \leq n \leq 9$, $h(n) = \{1, 0, -1\}$, 按 $N_1 = 6$ 用重叠相加法计算

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

解：因为 $N_1 = 6$, 则把 $x(n)$ 分为两段：

$$x_1(n) = \{1, 2, 3, 4, 5, 6\}$$

$$x_2(n) = \{7, 8, 9, 10, 0, 0\}$$

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 $N = N_1 + M - 1$ 点循环卷积。

3.3.4 DFT 的应用：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：

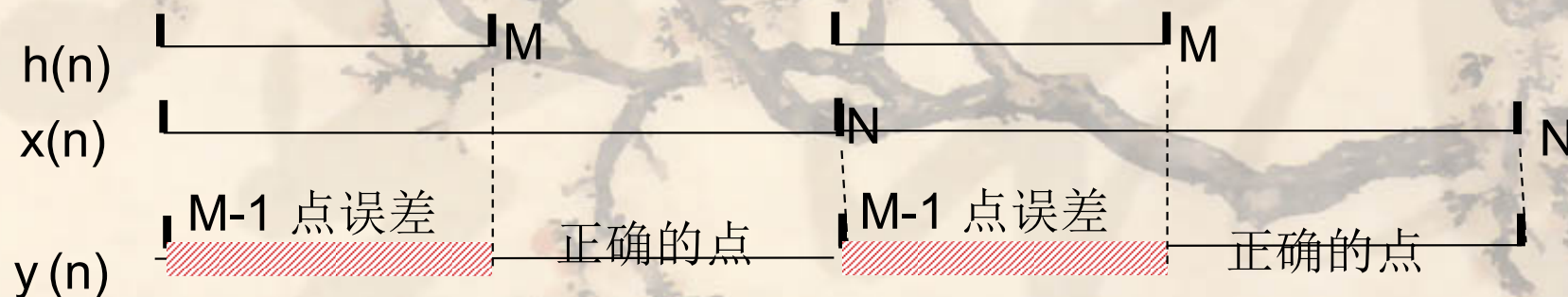
$$y(n) = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, -9, -10\}$$

3.3.4 DFT 的应用：重叠保留法

❖ 重叠保留法 —— 对输入 $x(n)$ 预处理

❖ 基本思路

- ❧ 假设把序列 $x(n)$ 分成多段 N 点序列，一个系统的脉冲响应为 M 点序列， $M < N$ 。
- ❧ 由上面的结论可知，输入分段序列和脉冲响应之间的 N 点循环卷积产生该段的输出序列，其中前 $M-1$ 个样本不是正确的输出值。
- ❧ 若将 $x(n)$ 简单地分成互不重叠的各段，则所得的输出序列会有不正确样本区间存在。
- ❧ 输入数据如何分段？结果如何处理？



3.3.4 DFT 的应用：重叠保留法

- 为了解决这个问题，将 $x(n)$ 分为长度 N 的分段，然后每段再往前多取 $(M-1)$ 个样本，即第 k 段的最后 $M-1$ 点保留下来作为第 $(k+1)$ 段的前 $M-1$ 点，各段长变为：

$$N + M - 1$$

其中最前面的一段 $x_0(n)$ ，在其前面补上 $M-1$ 个零，使其长度也为 $N+M-1$ 。

$$x_0(n) = \begin{cases} 0 & -M+1 \leq n \leq -1 \\ x(n) & 0 \leq n \leq N-1 \\ 0 & \text{其他} \end{cases}$$

$x(n) \rightarrow N + M - 1$ ，相邻两段之间存在 $M-1$ 点重叠

$$y(n) \text{ 分段，长度为 } N, y_l(n) = \begin{cases} y(n) & lN \leq n \leq (l+1)N - 1 \\ 0 & \text{其他} \end{cases}$$

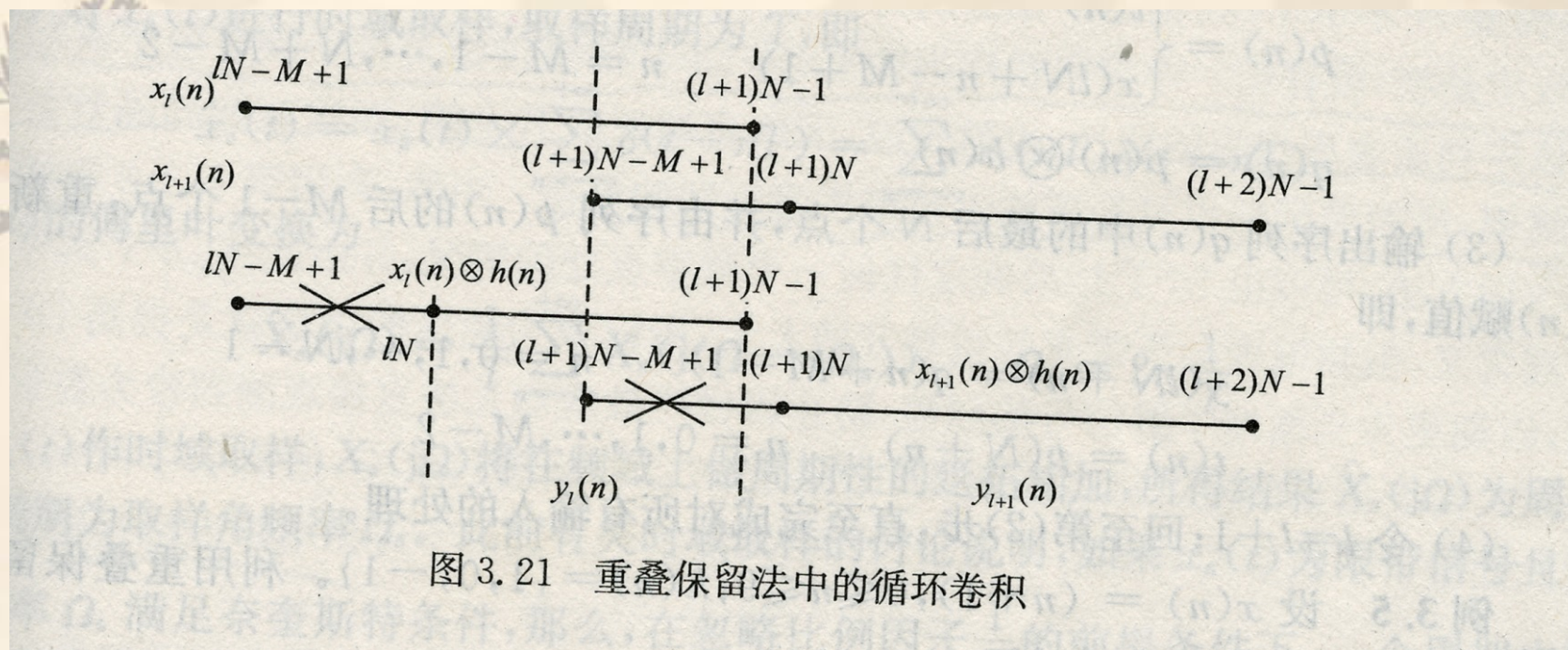


图 3.21 重叠保留法中的循环卷积

$$y_l(n) = \sum_{m=-lN-M+1}^{(l+1)N-1} x_l(m)h(n-m), \quad lN \leq n \leq (l+1)N-1$$

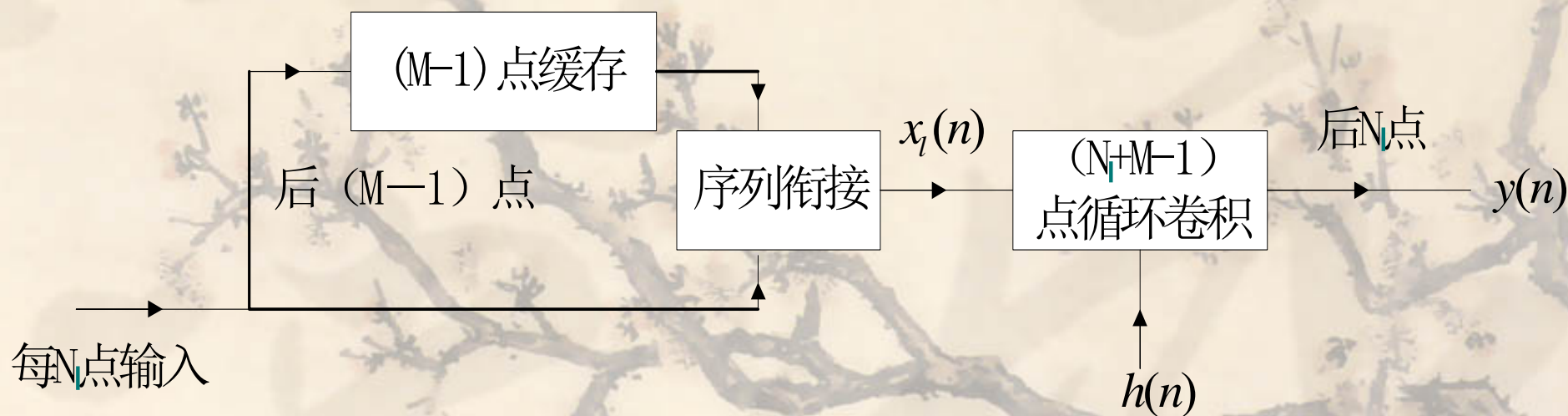
3.3.4 DFT 的应用：重叠保留法

- ❖ 因此，循环卷积 $y_c(n)$ 的前 $M-1$ 个值是线性卷积 $y_l(n)$ 前面的 $M-1$ 个值与 $y_l(n)$ 后面 $M-1$ 个值的混迭，是错误的样本。
- ❖ 循环卷积 $y_c(n)$ 的后 N 个值才与 $y_l(n)$ 中间的 N 个值相同。

将每一段所得到的循环卷积的前 $M-1$ 个值去掉，保留后面的 N 个值，再将各段保留的 N 个值前后拼接起来，就得到所要求的线性卷积 $y(n)$ 。

3.3.4 DFT 的应用：重叠保留法

这种通过将 $y_l(n)$ 与 $y_{l-1}(n)$ 相重叠的部分舍去，即舍去循环卷积中前 $M-1$ 项，保留后面的 N 个数值，来得到所求结果的方法称为重叠保留法，有的书中又称为重叠舍去法。



3.3.4 DFT 的应用：重叠保留法

例 3.20: 设 $x(n) = (n+1)$, $0 \leq n \leq 9$, $h(n) = \{1, 0, -1\}$, 按 $N = 6$ 用重叠舍去法计算

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

解：由于 $M = 3$, 必须使每一段与前一段重叠两个样本, $x(n)$ 为 10 点序列, 需要在开头加 $(M-1) = 2$ 个零。因为 $N = 6$, 则可划分为三部分:

$$x_1(n) = \{0, 0, 1, 2, 3, 4\}$$

$$x_2(n) = \{3, 4, 5, 6, 7, 8\}$$

$$x_3(n) = \{7, 8, 9, 10, 0, 0\}$$

因为 $x(n)$ 在 $n > 9$ 时无值, 因此在 $x_3(n)$ 中必须填两个零。

3.3.4 DFT 的应用：重叠保留法

现在计算每一部分与 $h(n)$ 循环卷积 (不是线性卷积)

$$y_1(n) = x_1(n) \otimes h(n) = \{-3, -4, 1, 2, 2, 2\}$$

$$y_2(n) = x_2(n) \otimes h(n) = \{-4, -4, 2, 2, 2, 2\}$$

$$y_3(n) = x_3(n) \otimes h(n) = \{7, 8, 2, 2, -9, -10\}$$

注意舍去前两个样本，装配输出 $y(n)$ 为

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

也可以直接计算线性卷积，结果为：

$$x(n) * h(n) = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, -9, -10\}$$

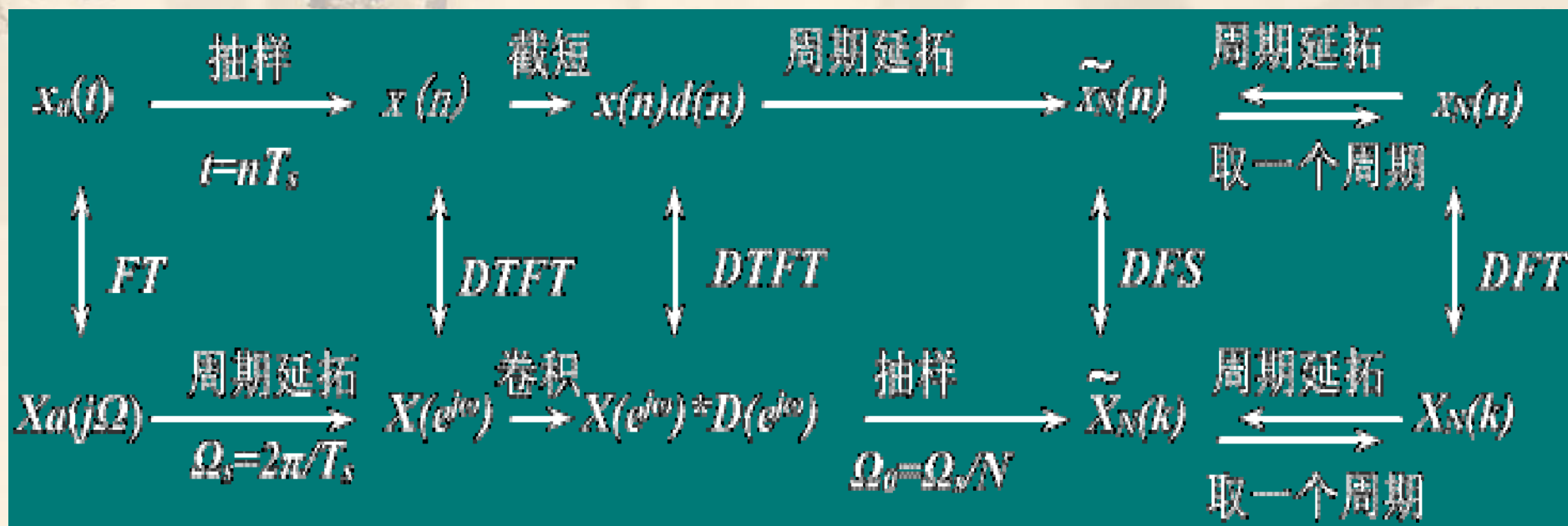
与重叠保留法结果相同。

3.3.4 DFT 的应用：分段卷积小结

- ❖ 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。
- ❖ 重叠相加法：对输入序列分段后进行线性卷积，对分段输出进行重叠相加，形成完整输出。
- ❖ 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 and 处理效率。

3.3.4 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



3.3.4 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

$$f_s \geq 2 f_h$$

$$f_s = 1 / T$$

$$T_0 = NT$$

$$T_0 = 1 / F_0$$



$$f_s = 1 / T = N / T_0 = NF_0$$



$$N = \frac{T_0}{T} = \frac{f_s}{F_0}$$



为便于 FFT 计算，一般选择 $N = 2^r$
(r 为正整数)

3.3.4 DFT 的应用：频谱分析

■ 信号最高频率与频率分辨率之间的矛盾

$$N = \frac{T_0}{T} = \frac{f_s}{F_0}$$

① 信号最高频率 $f_h \uparrow$ 提高，则需要提高 $f_s \uparrow$

当 N 给定，必增大频域取样间隔 $F_0 \uparrow$ ，即降低了频率分辨率 \downarrow

② 要提高频率分辨率 $F_0 \downarrow$ ，则增大了 $T_0 = \frac{1}{F_0} \uparrow$

给 N 定，则必提高 $T \uparrow$ ，降低 $f_s \downarrow$ ，为不产生混叠，必降低 $f_h \downarrow$

要同时提高信号最高频率和频率分辨率，需增加采样点数 N ，即采集更长的数据。

3.3.4 DFT 的应用：频谱分析

■ DFT 计算频谱

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

✧ 当 $k = 0, 1, \dots, \frac{N}{2}$ 时, $X(k)$ 对应频谱值。

✧ 频谱 $X(k)$ 是由实部 U 和虚部 V 组成:

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

✧ 于是幅频特性 $A(k)$ 、相位谱 $\Phi(k)$ 、功率谱 $G(k)$ 分别为:

$$A(k) = |X(k)|$$

$$\Phi(k) = \arctan \frac{V(k)}{U(k)}$$

$$G(k) = |X(k)|^2$$

3.3.4 DFT 的应用：频谱分析

频谱混叠：

■ 原因：

- 信号不具备限带的特点
- 取样角频率不满足奈奎斯特条件，频谱会产生混叠，造成DFT谱 $X(k)$ 和 $X_a(j\frac{2\pi k}{NT})$

■ 常见方法：

- 在时域取样前对信号进行滤波
 - 选取恰当的取样频率以降低混叠程度。
- 注：频谱混叠并非DFT分析信号频谱时才会遇到的问题，进行DTFT也要控制因频谱混叠所导致的误差。

3.3.4 DFT 的应用：频谱分析

例3.22 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 $\leq 10\text{Hz}$ ，信号最高频率 $\leq 4\text{kHz}$ 。

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 。

3.3.4 DFT 的应用：频谱分析

解：1) 最小记录长度：

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

2) 最大取样间隔 ($f_s > 2f_h$ $f_s = 1/T$)

$$T < \frac{1}{2f_h} = \frac{1}{2 \times 4 \times 10^3 \text{Hz}} = 0.125\text{ms}$$

3) 最小记录点数

$$N > \frac{f_s}{F_0} = \frac{2f_h}{F_0} = \frac{2 \times 4 \times 10^3}{10} = 800$$

$$\text{取 } N = 2^m = 2^{10} = 1024 > 800$$

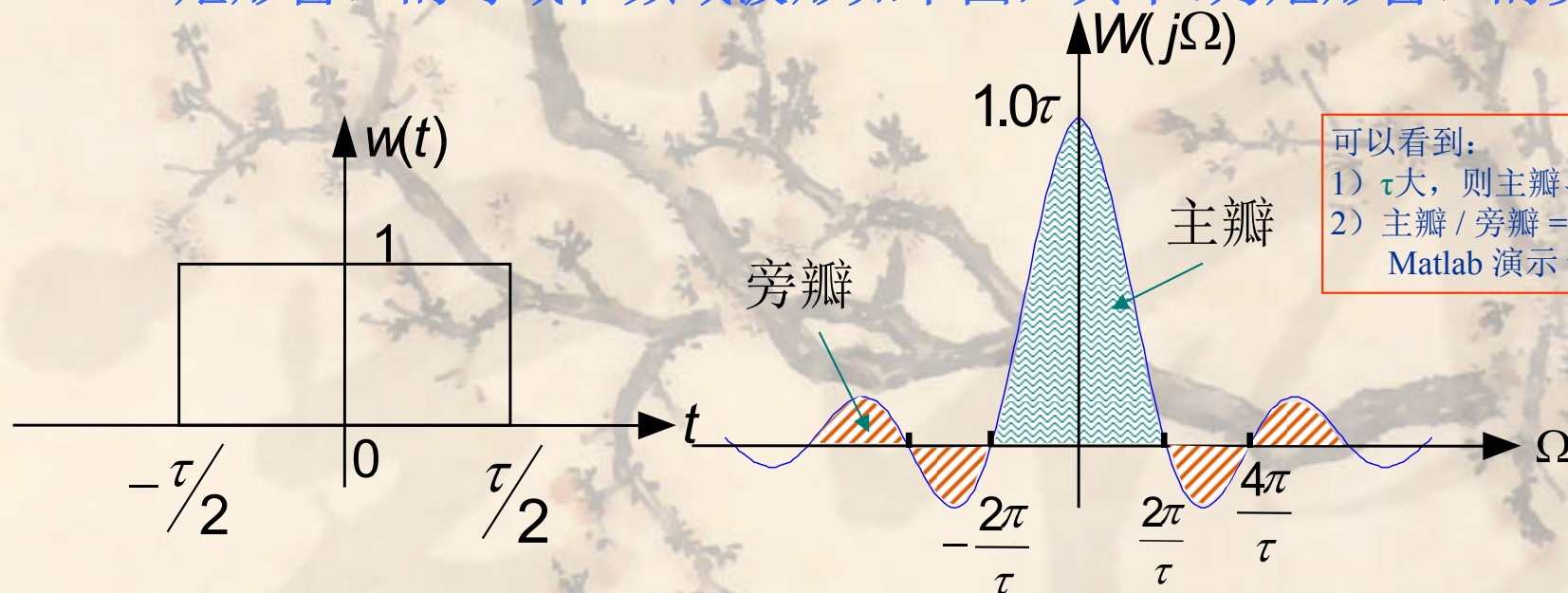
3.3.4 DFT 的应用：频谱泄漏(窗效应)

❖ 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 $W(nT)$ 或 $W(t)$ ，即： $x(nT)W(t)$
- 根据频域卷积定理：时域相乘映射为频域卷

$$x(nT) \cdot W(t) \rightarrow X(e^{j\Omega T}) * W(j\Omega)$$

✧ 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。



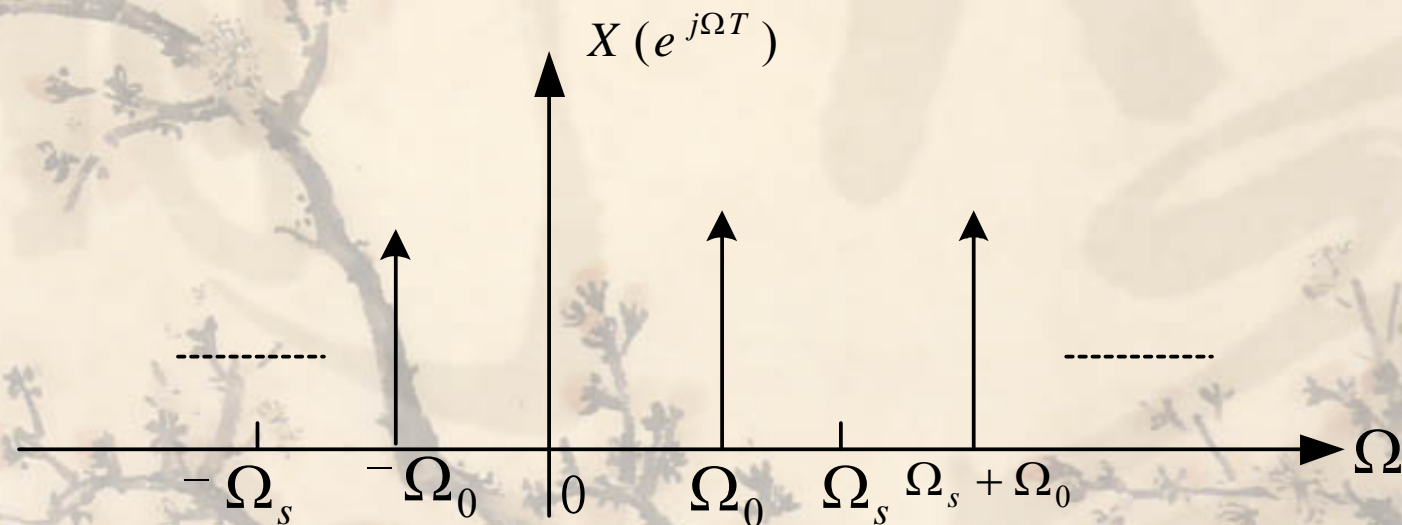
可以看到：

- 1) τ 大，则主瓣、旁瓣宽度小；
- 2) 主瓣 / 旁瓣 = 固定值 (13dB, Matlab 演示 wintool)

3.3.4 DFT 的应用：频谱泄漏(窗效应)

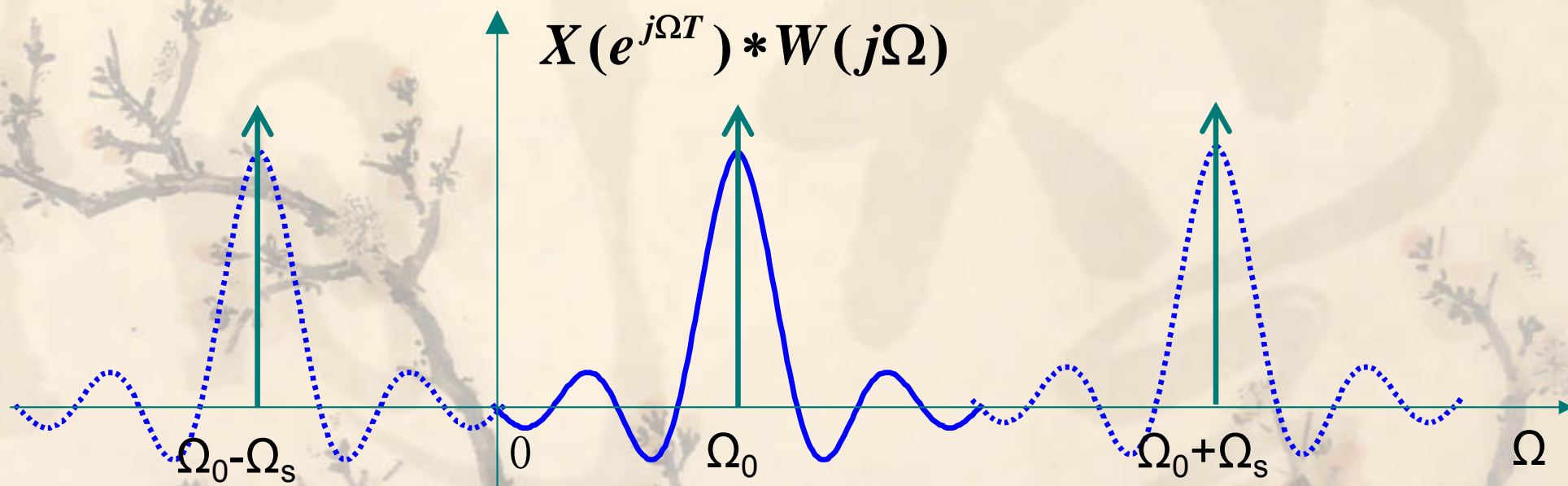
例如：

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



在频域中，其谱线大小为 $2\pi/T$ ，位于 $\Omega_0 + n\Omega_s$ 处。但将其截短，即加窗后，频域中要与 $\text{sinc}(\frac{\Omega T}{2})$ 相卷积，其中在 Ω_0 处卷积后的频谱变成如下图：

3.3.4 DFT 的应用：频谱泄漏(窗效应)



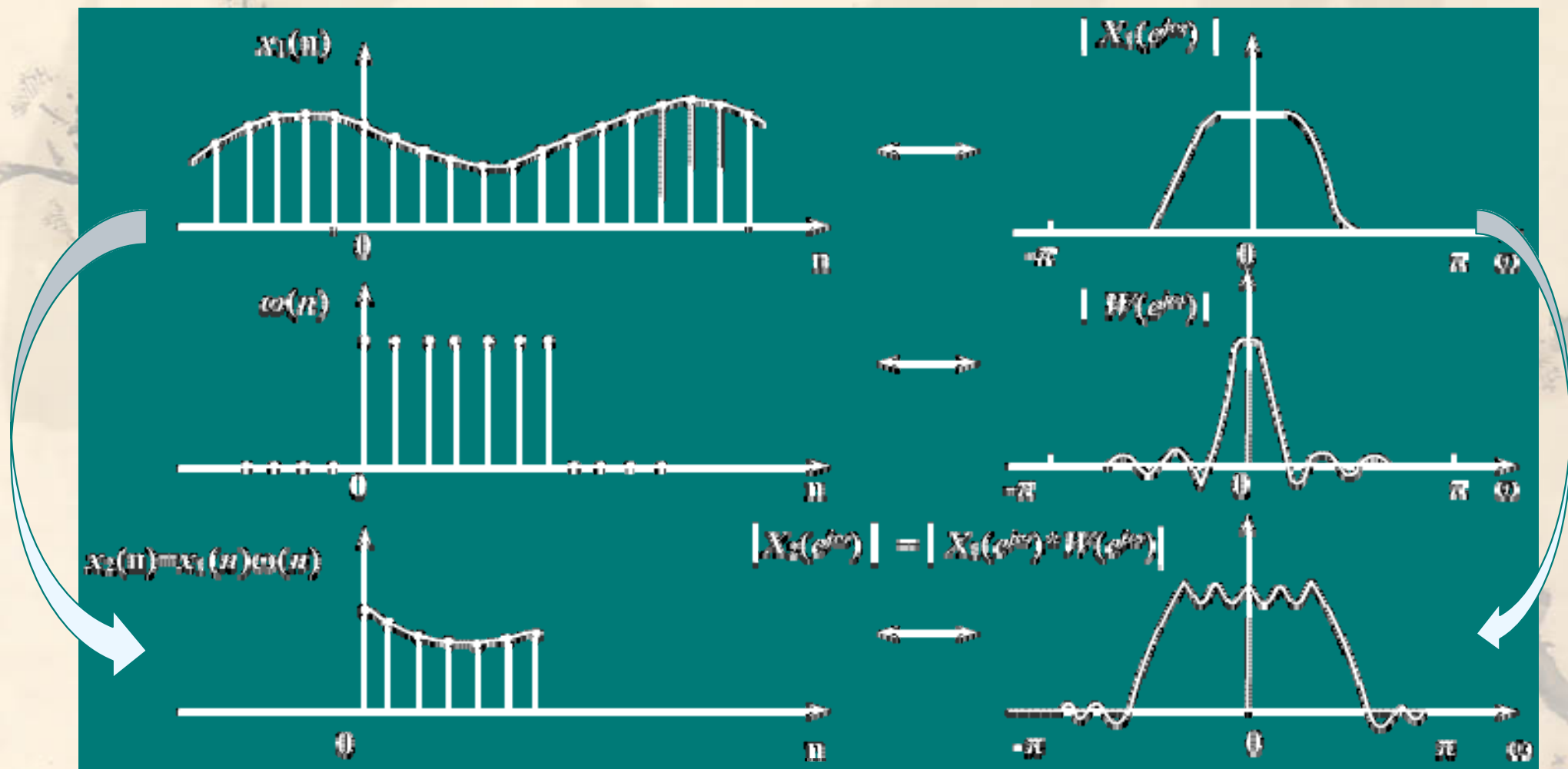
❖ 主要问题就是频谱被展宽，同时也会造成混叠失真。

∞ 可以看到原来在 Ω_0 处的一根谱线，变成了以 Ω_0 为中心的连续谱线。

∞ 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。原来在一个周期 Ω_s 内只有一个频率上有非零值，而现在一个周期内几乎所有频率上都有非零值。在 $\Omega_0 + n\Omega_s$ 处还有相同的卷积结果，考虑所有的卷积结果后，则还有混迭现象产生。

3.3.4 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



注：频谱泄漏的存在必然会导致频谱混叠。

3.3.4 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 $\tau \rightarrow \infty$ 时， $W(j\Omega)$ 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，即：

$$\delta(\Omega) * X(e^{j\Omega T}) = X(e^{j\Omega T})$$

⊕ 因此加大窗宽度可使泄露减少，但也不能无限加宽，因为当 $\tau \rightarrow \infty$ 时，等于没有截短。而且实际的信号的窗口长度未必总能随意加长，例如，如果信号是时变的，长窗可能会使计算结果发生混淆。选择窗长度需要同时考虑时域和频域。

□ 选择适当的窗函数

⊕ 泄漏现象是由于矩形窗函数的频谱具有旁瓣，而且主瓣又有一定宽度所造成。为了减少泄漏，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、布莱克曼窗和凯塞窗等

■ 注意：

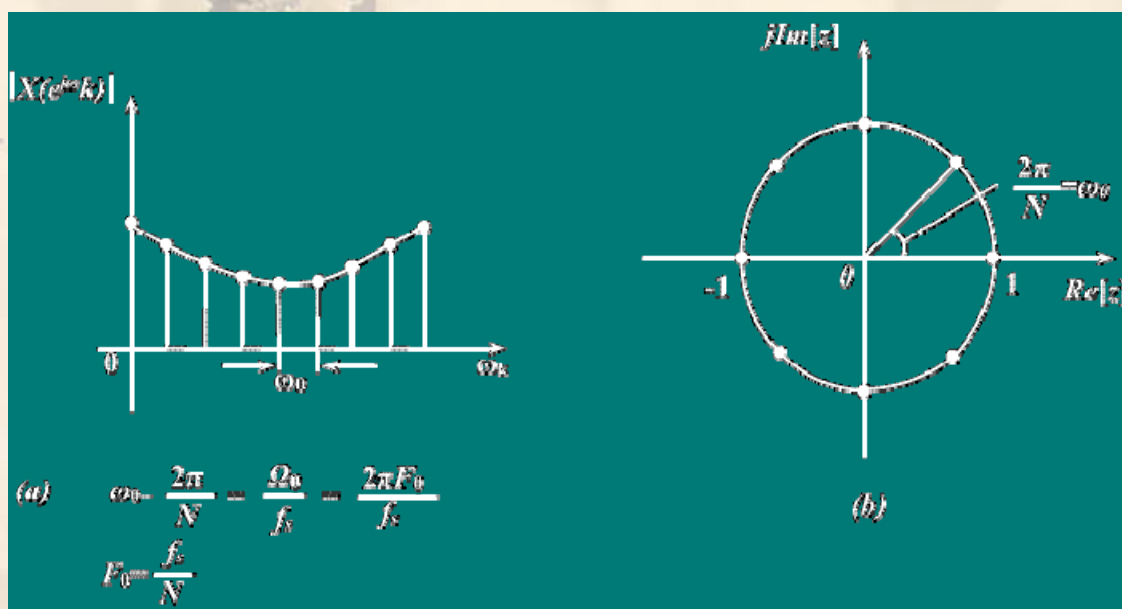
① 窗口宽度应与所需数据长度 N 相同；

② 若计算过程中需要补零，应先乘窗口变为 N 点后，再补零，即补零后的点不乘窗函数。

DFT 应用问题：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，所以DFT仅反映有限个离散频率点处所表现出来的特性，不能直接反映全貌。

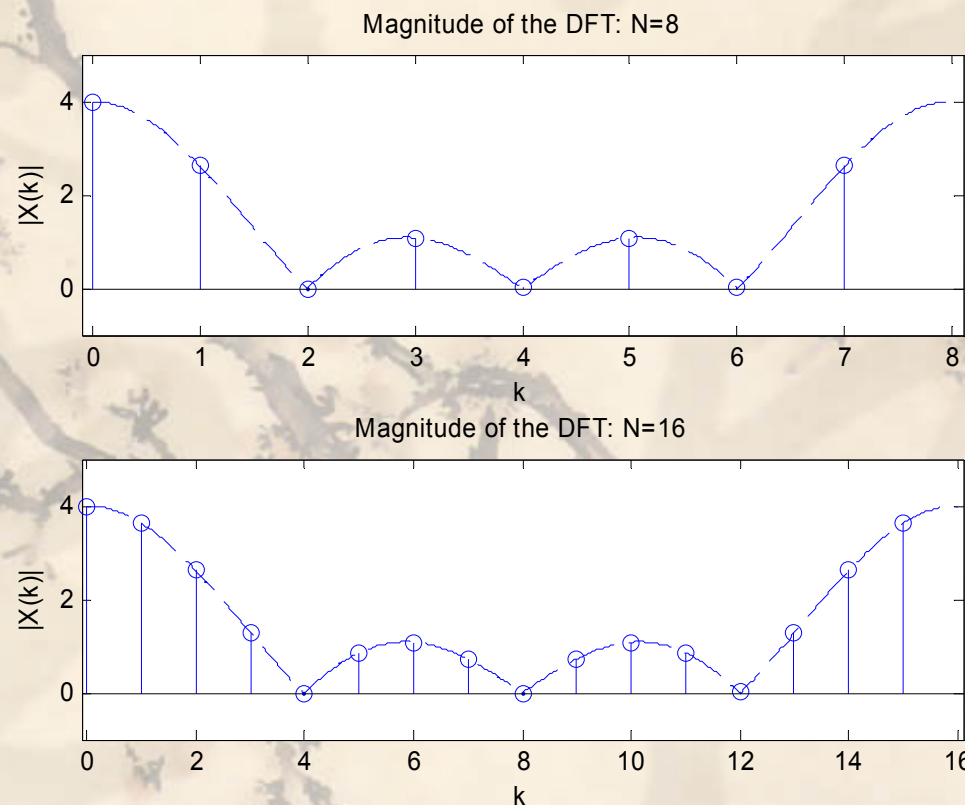
$x(n) \xrightarrow{DFT}$ 离散频谱 \rightarrow 连续频谱上的若干点



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。

3.3.4 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
- ❖ 在实际问题中，被“栅栏”挡住大的频谱的情况很少，所以栅栏效应不是很严重的问题。
- ❖ 另外，补0不会影响信号的 **DFT** 特性，因为它没有增加信号的信息，也不能提高 **DFT** 频谱的准确性，它只是可以减少频率间隔，移动了“栅栏”，看到了原来就存在的信息。



3.4 FFT (快速离散傅里叶变换)

- ❖ **FFT: Fast Fourier Transform**
- ❖ 1965 年, James W. Cooley 和 John W. Tukey 在《计算数学》(《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 Winograd 算法,素因子算法。

An-Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

An efficient method for the calculation of the interactions of a 2^n factorial experiment was introduced by Yates and is widely known by his name. The generalization to 3^n was given by Box et al. [1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one must multiply an N -vector by an $N \times N$ matrix which can be factored into m sparse matrices, where m is proportional to $\log N$. This results in a procedure requiring a number of operations proportional to $N \log N$ rather than N^2 . These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of N . It is also shown how special advantage can be obtained in the use of a binary computer with $N = 2^n$ and how the entire calculation can be performed within the array of N data storage locations used for the given Fourier coefficients.

Consider the problem of calculating the complex Fourier series

$$(1) \quad X(j) = \sum_{k=0}^{N-1} A(k) \cdot W^{jk}, \quad j = 0, 1, \dots, N-1,$$

where the given Fourier coefficients $A(k)$ are complex and W is the principal N th root of unity,

$$(2) \quad W = e^{2\pi i/N}$$

A straightforward calculation using (1) would require N^2 operations where "operation" means, as it will throughout this note, a complex multiplication followed by a complex addition.

The algorithm described here iterates on the array of given complex Fourier amplitudes and yields the result in less than $2N \log_2 N$ operations without requiring more data storage than is required for the given array A . To derive the algorithm, suppose N is a composite, i.e., $N = r_1 \cdot r_2$. Then let the indices in (1) be expressed

$$(3) \quad \begin{aligned} j &= j_1 r_2 + j_0, & j_0 &= 0, 1, \dots, r_2 - 1, & j_1 &= 0, 1, \dots, r_1 - 1, \\ k &= k_1 r_2 + k_0, & k_0 &= 0, 1, \dots, r_2 - 1, & k_1 &= 0, 1, \dots, r_1 - 1. \end{aligned}$$

Then, one can write

$$(4) \quad X(j_1, j_0) = \sum_{k_1=0}^{r_1-1} \sum_{k_0=0}^{r_2-1} A(k_1, k_0) \cdot W^{j_1 r_2 k_1 + j_0 k_0}$$

Received August 17, 1964. Research in part at Princeton University under the sponsorship of the Army Research Office (Durham). The authors wish to thank Richard Garwin for his essential role in communication and encouragement.



Dr. James W. Cooley

Worked at IBM Watson Research Center in Yorktown Heights, N.Y.. After his retirement from IBM in 1991, he joined the Electrical Engineering Department at the University of Rhode Island.

John Tukey



John Wilder Tukey

Born	June 16, 1915 New Bedford, Massachusetts, USA
Died	July 26, 2000 (aged 85) New Brunswick, New Jersey
Residence	United States
Nationality	American
Fields	Mathematician
Institutions	Bell Labs Princeton University
Alma mater	Brown University Princeton University
Doctoral advisor	Solomon Lefschetz
Doctoral students	Frederick Mosteller
Known for	FFT algorithm Box plot Coining the term 'bit'

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：

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

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

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 个 $X(k)$ (N 点 DFT)	$4N^2$	$(N - 1)(2N + 2) = 2(2N - 1)$

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

$$N^2 = 512^2 = 2^{18} = 262144 \text{ (26万次)}$$

$$N^2 = 1024^2 = 2^{20} = 1048576 \text{ (105万次)}$$

$$N^2 = 8192^2 = 2^{26} = 67108864 \text{ (6千7百万次)}$$

❧ 对于大 N ，在实际中是不能接受的，无法“**实时**”应用 DFT。

❖ 一般地，在计算机中，一次加法比一次乘法所需时间要短；

❖ 在 **DSP** 中，由于乘法用特殊的硬件电路专门完成，因此乘法和加法所需机器周期相同。

❧ **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^{kn} 的性质

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

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

$$W_N^{Nk} \cdot W_N^{-nk} \quad W_N^{nN} \cdot W_N^{-nk}$$

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

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

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

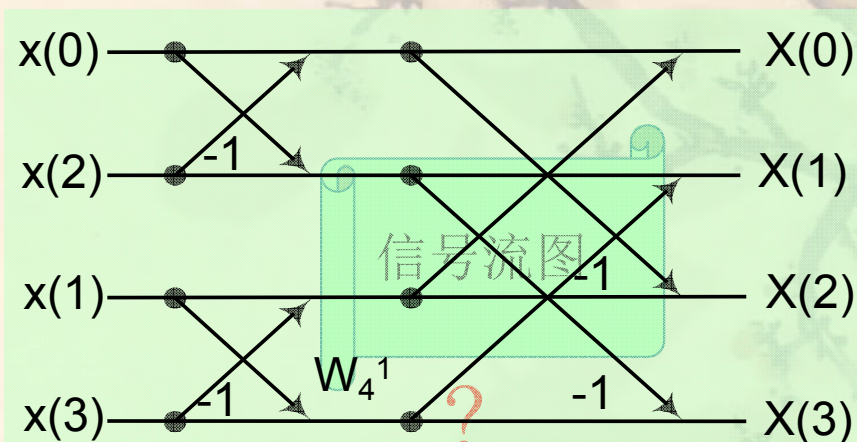
特殊点: $W_N^0 = 1$ $W_N^{N/2} = -1$ $W_N^{(k+N/2)} = -W_N^k$

3.4 FFT: W_N^{kn} 的性质

❖ 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:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W_4^{0 \times 0} & W_4^{0 \times 1} & W_4^{0 \times 2} & W_4^{0 \times 3} \\ W_4^{1 \times 0} & \color{red}{W_4^{1 \times 1}} & W_4^{1 \times 2} & \color{red}{W_4^{1 \times 3}} \\ W_4^{2 \times 0} & W_4^{2 \times 1} & W_4^{2 \times 2} & W_4^{2 \times 3} \\ W_4^{3 \times 0} & \color{red}{W_4^{3 \times 1}} & W_4^{3 \times 2} & \color{red}{W_4^{3 \times 3}} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4^1 & -1 & -W_4^1 \\ 1 & -1 & 1 & -1 \\ 1 & -W_4^1 & -1 & W_4^1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$



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

$$X(1) = [x(0) - x(2)] + \color{red}{[x(1) - x(3)] W_4^1}$$

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

$$X(3) = [x(0) - x(2)] - \color{red}{[x(1) - x(3)] W_4^1}$$

只需要 1
次复数乘

3.4 FFT: 基本思想

■ 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**: 将傅氏变换 DFT 分解成相继小的 DFT 计算, 即 N 变小, 而计算量与 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

3.4.1 FFT: 基2时间抽选法-算法原理

❖ 算法原理

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: $N = 2M$

$$\left. \begin{array}{l} \text{偶序列} \quad x^{(e)}(m) = x(2m) \\ \text{奇序列} \quad x^{(o)}(m) = x(2m+1) \end{array} \right\} m = 0, 1, \dots, \frac{N}{2} - 1$$

即一组由偶数序号组成, 另一组由奇数序号组成。

注意其长度为 $N/2$

3.4.1 FFT: 基2时间抽选法-算法原理

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

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

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

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

$$= X^e((k))_{N/2} + W_N^k X^o((k))_{N/2}$$

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

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

3.4.1 FFT：基2时间抽选法-算法原理

偶数取样点DFT为：

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

$$\begin{aligned} r &= 0, 1, \dots, N/2-1 \\ k &= 0, 1, \dots, N-1 \end{aligned}$$

奇数取样点DFT为：

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

- ① k 的整个范围为 $0 \sim (N-1)$ ，而 $X_1(k)$ 、 $X_2(k)$ 是由 $N/2$ 个样点形成的 DFT， $x(2r)$ 和 $x(2r+1)$ 的长度为 $N/2$ ；
- ② 由这两个偶数和奇数 $N/2$ 个时域样值可以计算出前 $N/2$ 个 DFT 系数，也可以计算出后 $N/2$ 个 DFT 系数。
- ③ 问题：这前后 $N/2$ 个 DFT 有无关系？ k 取 $N/2 \sim (N-1)$ 时， $X_1(k)$ 、 $X_2(k)$ 、 W_N 情况如何？

3.4.1 FFT: 基2时间抽选法-算法原理

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

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

观察

分为三部分

$$X(k) = \underbrace{X^e(k)}_{\text{第一}} + \underbrace{W_N^k}_{\text{第三}} \underbrace{X^o(k)}_{\text{第二}}$$

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

则

$$\begin{aligned} X^e(k) &= X^e\left(\frac{N}{2} + l\right) = \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_{N/2}^{r(\frac{N}{2}+l)} = \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_{N/2}^{r \frac{N}{2}} \cdot W_{N/2}^{rl} \\ &= \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_{N/2}^{rl} = X^e(l) \end{aligned}$$

$$W_{N/2}^{r \frac{N}{2}} = e^{-j \frac{2\pi}{N} \cdot r \cdot \frac{N}{2}} = e^{-j2\pi r} = 1$$

3.4.1 FFT: 基2时间抽选法-算法原理

同理

$$X^o(k) = X^o\left(\frac{N}{2} + l\right) = X^o(l)$$

而

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

因此

$$\begin{cases} X(k) = X^e(k) + W_N^k X^o(k) \\ X\left(k + \frac{N}{2}\right) = X^e(k) - W_N^k X^o(k) \end{cases}$$

$$k = 0, 1, \dots, \frac{N}{2} - 1 = M$$

- ▶ 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。
- ▶ 构成 M 次蝶形，故需完成 M 次复乘和 $2M$ 次复加，此外两个 M 点 DFT 的计算包含 $2M^2$ 次复乘和 $2M(M-1)$ 次复加，总共要 $M(2M+1)$ 次复乘和 $2M^2$ 次复加。

3.4.1 FFT: 基2时间抽选法-算法原理

- 上式表示为信号流程图:

1个蝶形运算需要1次复乘和2次复加



- 此信号流程图也称为**蝶形**流程图

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

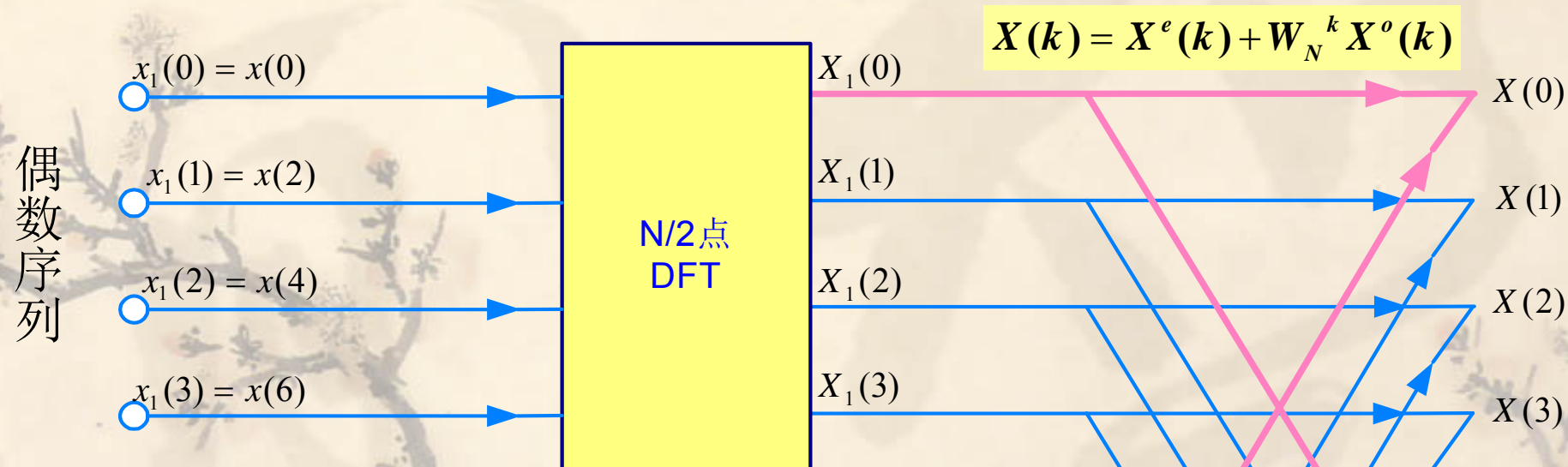
$$= X(k + \frac{N}{2})$$

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

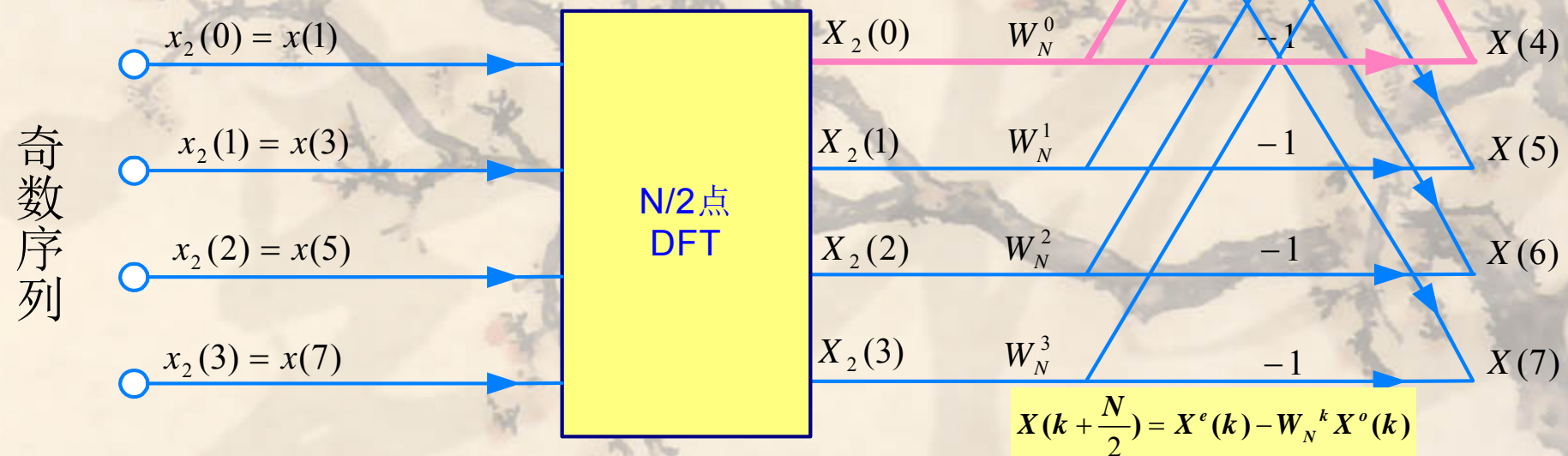
因此: 整个 $X(k)$ 的计算, 可以分解为前、后半部分的运算。而只要求出前半, 就可以由上式求出整个序列。

构成 M 次蝶形, 故需完成 M 次复乘和 $2M$ 次复加, 此外两个 M 点 DFT 的计算包含 $2M^2$ 次复乘和 $2M(M-1)$ 次复加, 总共要 $M(2M+1)$ 次复乘和 $2M^2$ 次复加。

3.4.1 FFT: 基2时间抽选法-算法原理



► 以N=8为例，其信号流程图：



3.4.1 FFT: 基2时间抽选法-算法原理

$N/2$ 仍为偶数, 进一步分解: $N/2 \rightarrow N/4$

$$\left. \begin{array}{l} x(n) \end{array} \right\} \left\{ \begin{array}{l} x(2r) = x^e(r) \left\{ \begin{array}{l} x(4r) = x^e(2l) = x^{ee}(l) \left\{ \dots \right. \\ x(4r+2) = x^e(2l+1) = x^{eo}(l) \left\{ \dots \right. \end{array} \right. \\ x(2r+1) = x^o(r) \left\{ \begin{array}{l} x(4r+1) = x^o(2l) = x^{oe}(l) \left\{ \dots \right. \\ x(4r+3) = x^o(2l+1) = x^{oo}(l) \left\{ \dots \right. \end{array} \right. \end{array} \right.$$

3.4.1 FFT: 基2时间抽选法-算法原理

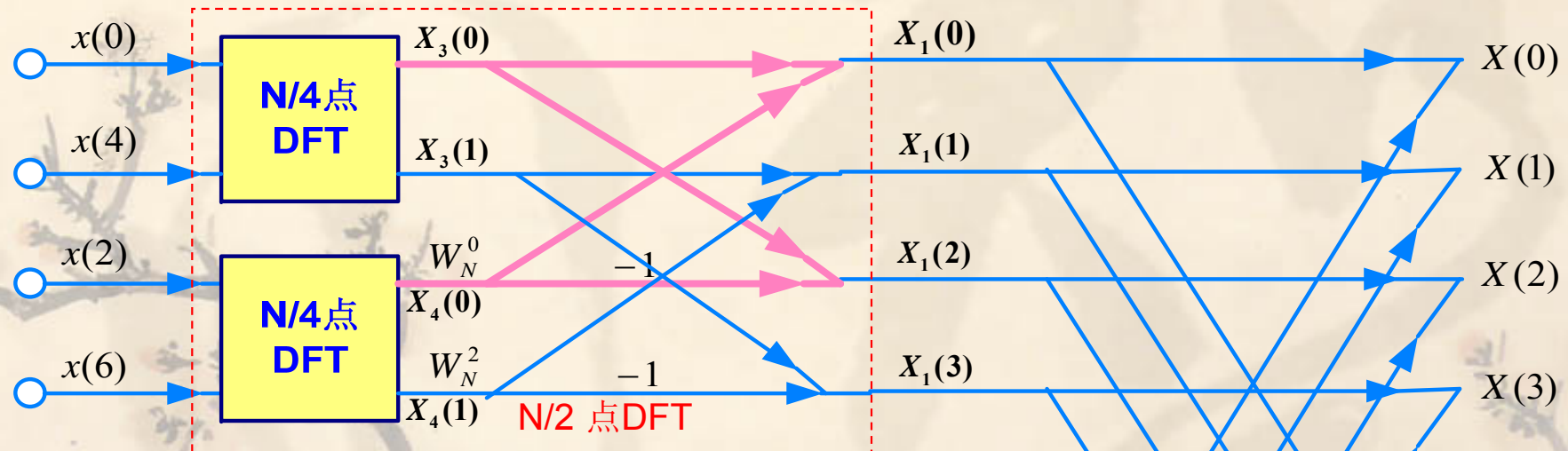
$$\begin{cases} x^e(2l) = x^{ee}(l) \\ x^e(2l+1) = x^{eo}(l) \end{cases} \quad \begin{cases} x^o(2l) = x^{oe}(l) \\ x^o(2l+1) = x^{oo}(l) \end{cases} \quad l = 0, 1, \dots, \frac{N}{4} - 1$$

$$\begin{cases} X^e(k) = X^{ee}(k) + W_{N/2}^k X^{eo}(k) \\ X^e(k + \frac{N}{4}) = X^{ee}(k) - W_{N/2}^k X^{eo}(k) \end{cases} \quad k = 0, 1, \dots, \frac{N}{4} - 1$$

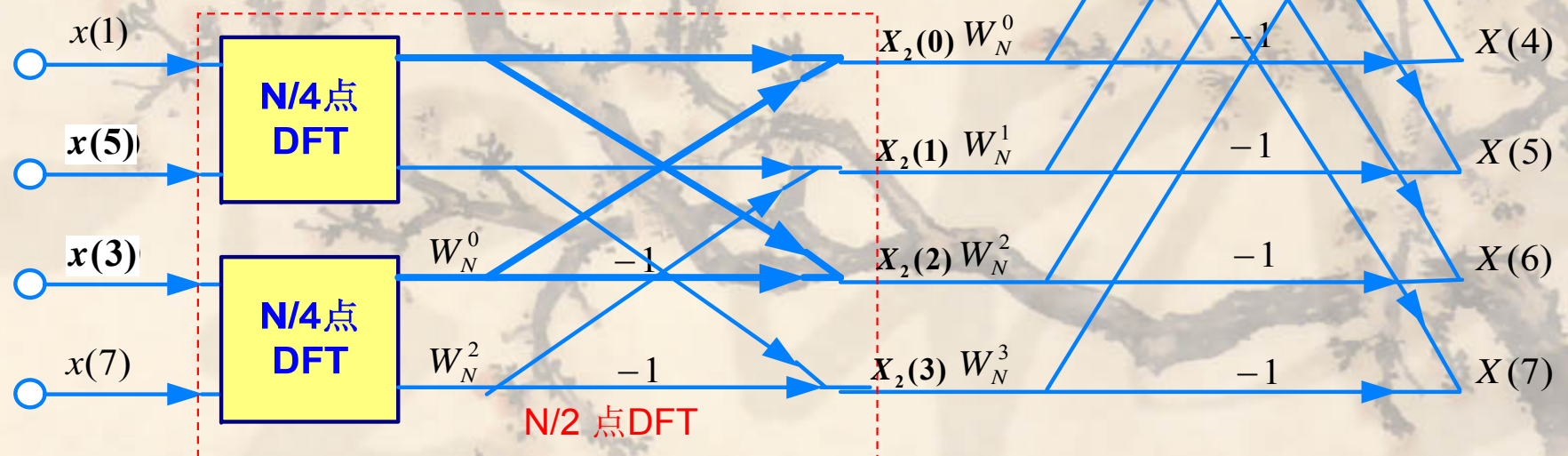
同理

$$\begin{cases} X^o(k) = X^{oe}(k) + W_{N/2}^k X^{oo}(k) \\ X^o(k + \frac{N}{4}) = X^{oe}(k) - W_{N/2}^k X^{oo}(k) \end{cases} \quad k = 0, 1, \dots, \frac{N}{4} - 1$$

3.4.1 FFT: 基2时间抽选法-算法原理

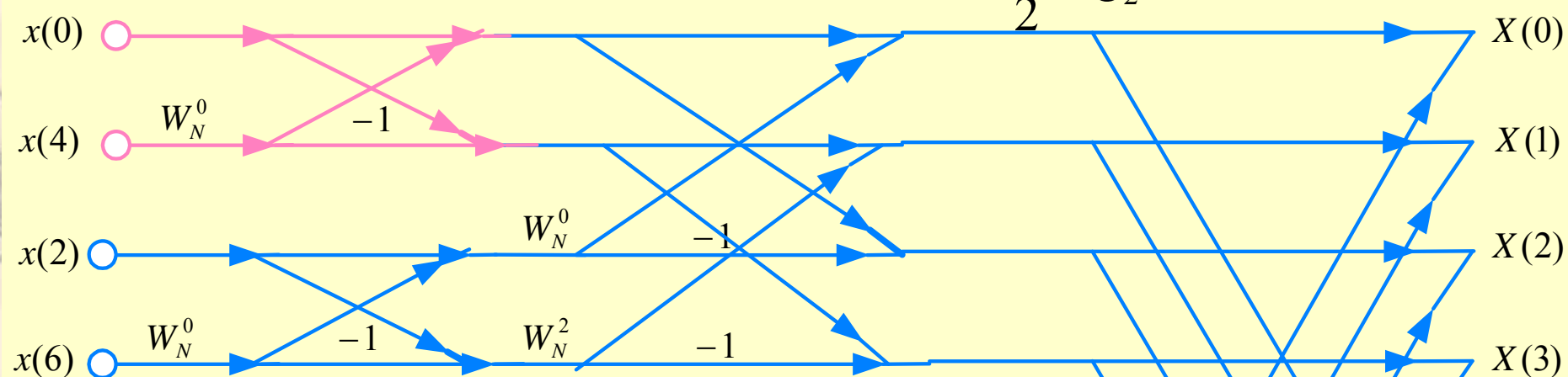


以 $N=8$ 为例，其信号流程图为



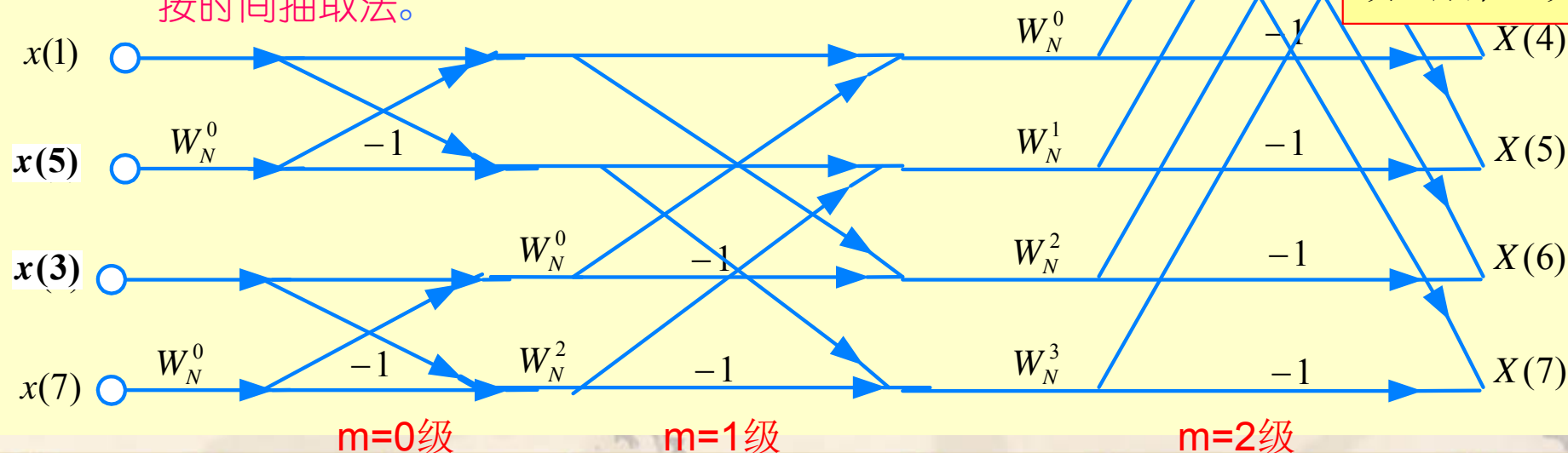
3.4.1 FFT：基2时间抽选法-算法原理

✱ $N=2^L$ 点FFT算法，流图分为L级，每级 $N/2$ 个蝶形，总计 $\frac{N}{2} \log_2 N$ 蝶形



► 仍以 $N=8$ 为例：

✱ 每一次分解都是按时间域输入序列的奇偶性次序，分解为两个半长的子序列，所以称为按时间抽取法。



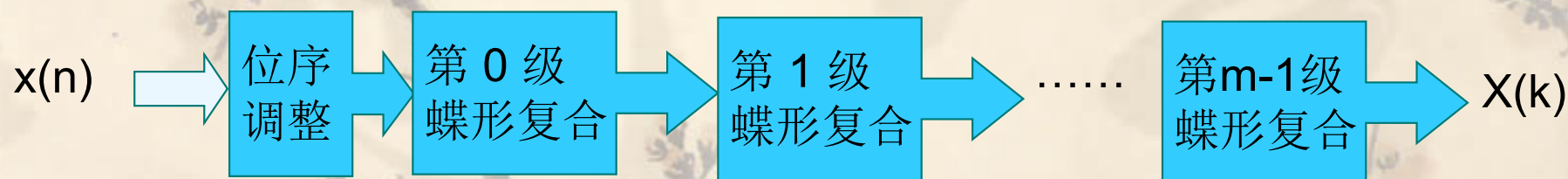
注：复数乘法5次，原来64次；
复数加法为24次，原来56次。

3.4.1 FFT: 基2时间抽选法-算法原理

上式中

$$\begin{bmatrix} W_4^0 \\ W_4^1 \end{bmatrix} = \begin{bmatrix} e^{-j\frac{2\pi}{4} \cdot 0} \\ e^{-j\frac{2\pi}{4} \cdot 1} \end{bmatrix} = \begin{bmatrix} 1 \\ -j \end{bmatrix}$$

$$\begin{bmatrix} W_8^0 \\ W_8^1 \\ W_8^2 \\ W_8^3 \end{bmatrix} = \begin{bmatrix} 1 \\ (1-j)/\sqrt{2} \\ -j \\ -(1+j)/\sqrt{2} \end{bmatrix}$$



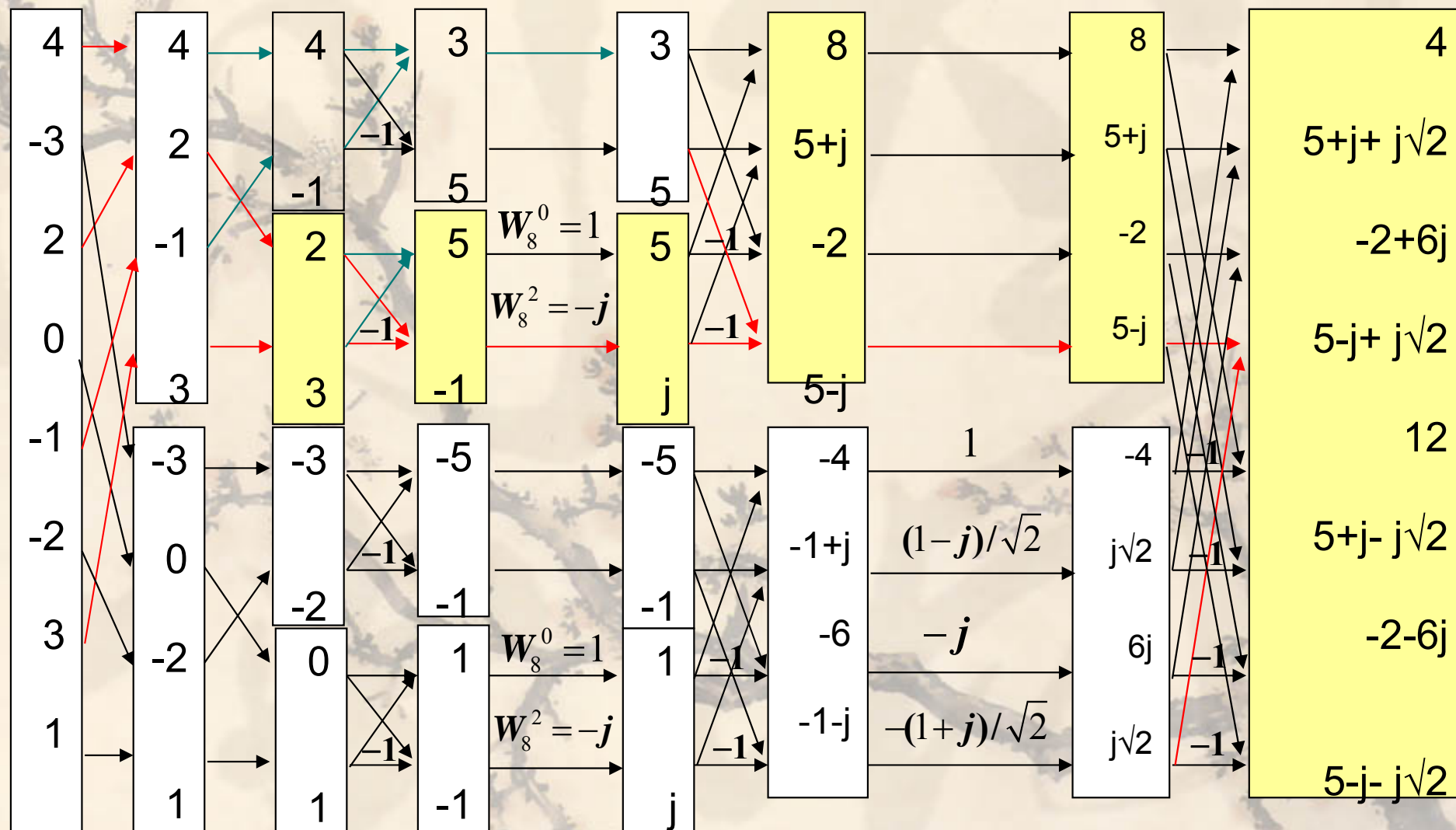
2^m 点 DFT 的基 2 时间抽选计算过程

■ 可见:

- ✧ 为什么引入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。

3.4.1 FFT: 基2时间抽选法-算法原理

例3.25 使用基 2 时间抽选法 FFT 流图, 计算下列数据的 8点 DFT:
 $x=[4,-3,2,0,-1,-2,3,1]$



3.4.1 FFT: 基2时间抽选法-蝶形运算

❖ 蝶形运算

在 FFT 信号流图中每一级里节点都是成对存在的，计算时总是下节点的值乘以 W_N^r ，然后与上节点的值相加减，形成下一列两个节点值。

这种计算的基本关系是：

$$x_{m+1}(p) = x_m(p) + W_N^r x_m(q)$$

$$x_{m+1}(q) = x_m(p) - W_N^r x_m(q)$$

式中 p, q 是上下节点的序号。（从 0 开始， $p, q = 0, 1, \dots, N/2$ ）



每一级中每对节点称为对偶节点，对偶节点的间距为：

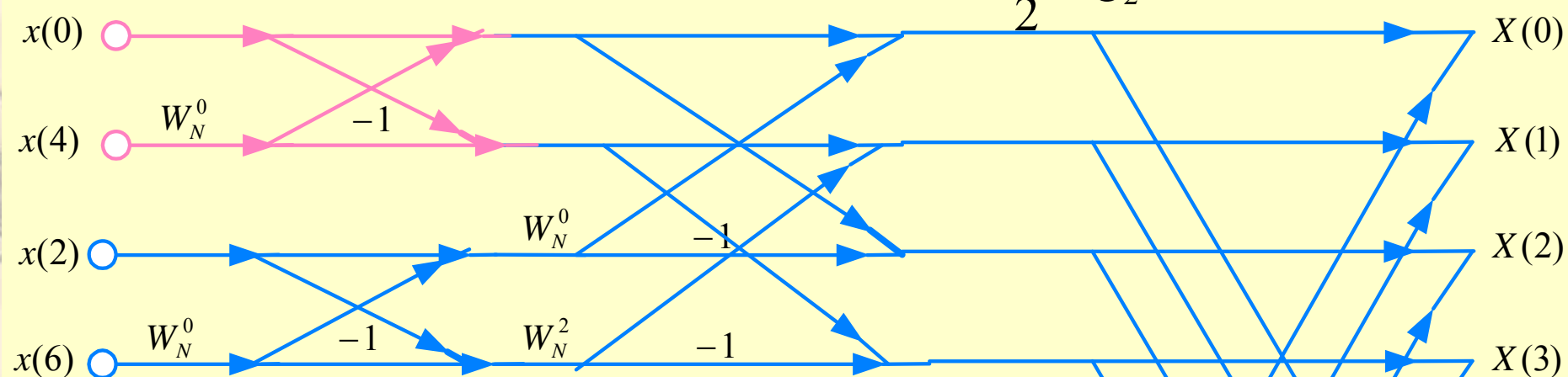
$$(p - q) = 2^m$$

$$r = \frac{N}{2^{m+1}} \cdot p = \frac{N}{2} \left[\frac{q}{2^m} - 1 \right]$$

注意：只有下节点乘以加权因子 W_N^r

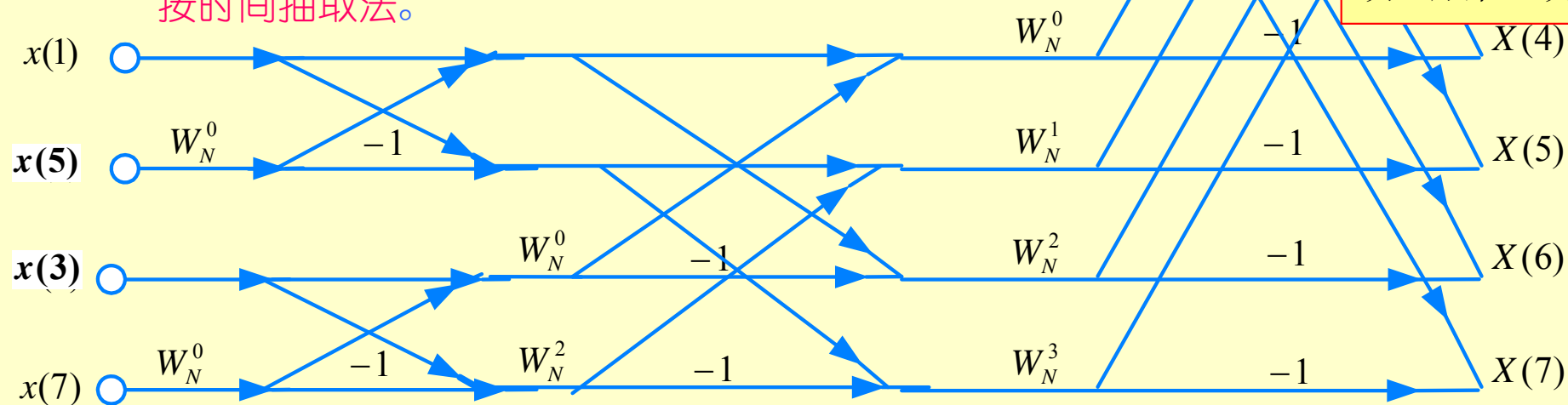
3.4.1 FFT：基2时间抽选法-算法原理

✱ $N=2^L$ 点FFT算法，流图分为L级，每级 $N/2$ 个蝶形，总计 $\frac{N}{2} \log_2 N$ 蝶形



► 仍以 $N=8$ 为例：

✱ 每一次分解都是按时间域输入序列的奇偶性次序，分解为两个半长的子序列，所以称为按时间抽取法。



注：复数乘法5次，原来64次；
复数加法为24次，原来56次。

m=0级

m=1级

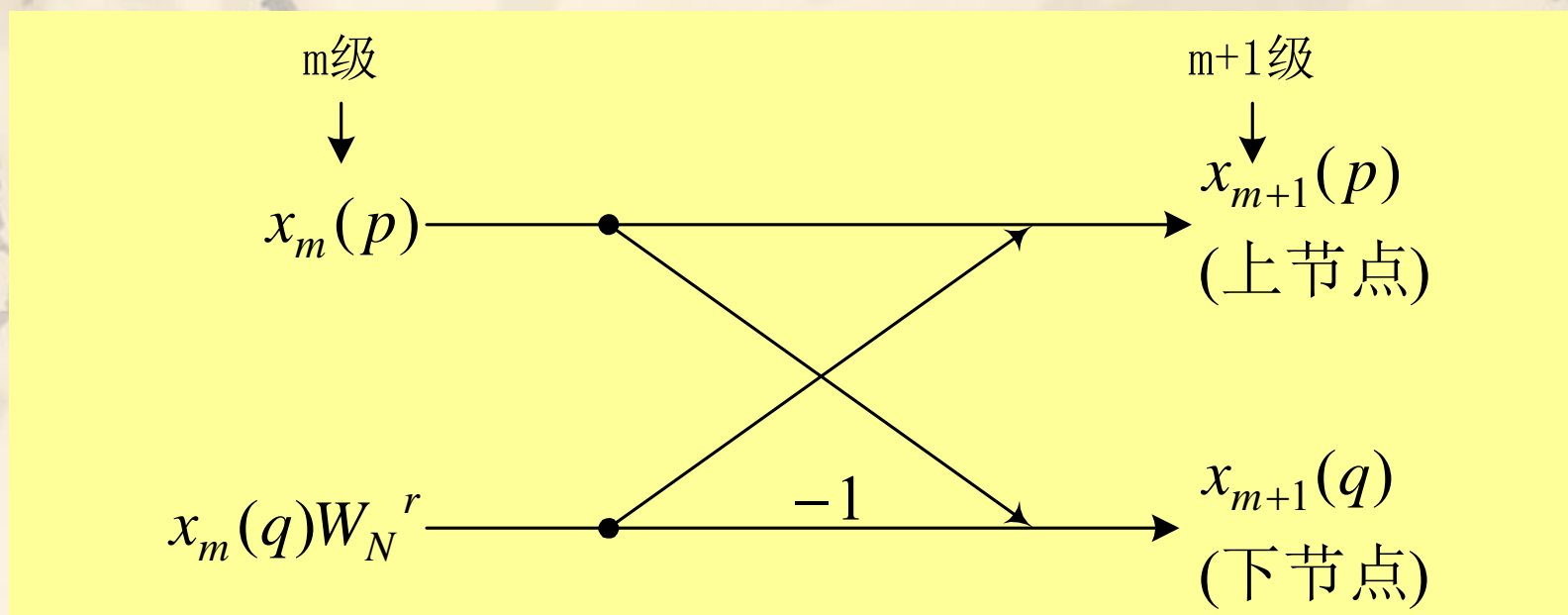
m=2级

3.4.1 FFT: 基2时间抽选法-同址计算

❖ 同址计算（同位特性）

1) 不同级数的节点序号不变

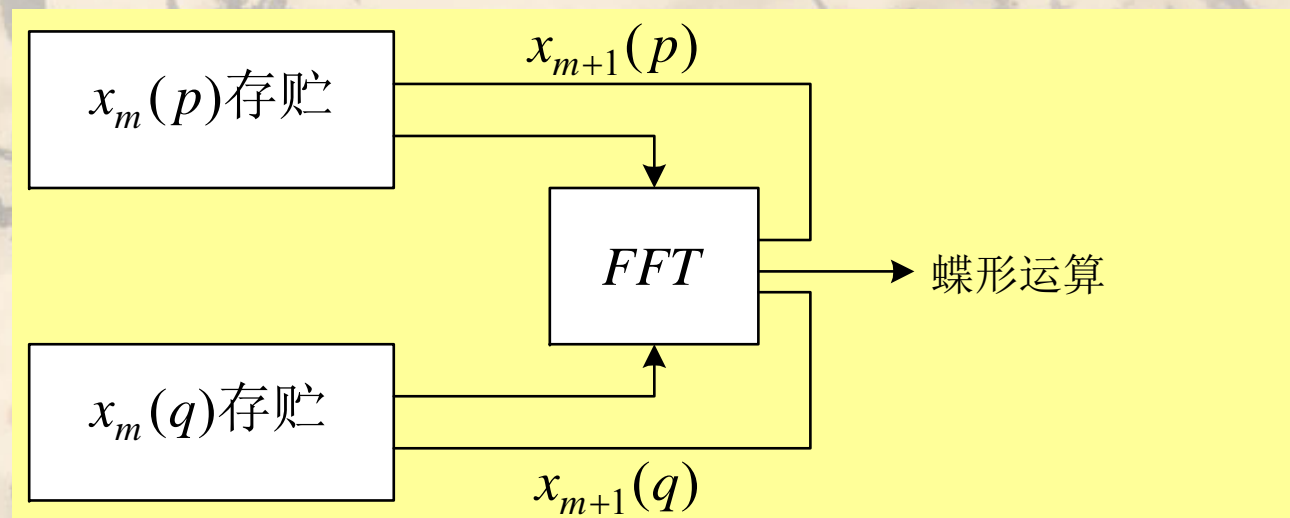
从蝶形运算可以看出第 m 列的 $x_m(p)$, $x_m(q)$ 经蝶形运算后, 在第 $(m+1)$ 列中的节点序号不变, 即 $x_{m+1}(p)$ 中的 p 与 $x_{m+1}(q)$ 中的 q 仍分别是 $x_m(p)$, $x_m(q)$ 中的 p 和 q 值。



3.4.1 FFT: 基2时间抽选法-同址计算

2) 蝶形运算是自成独立单元

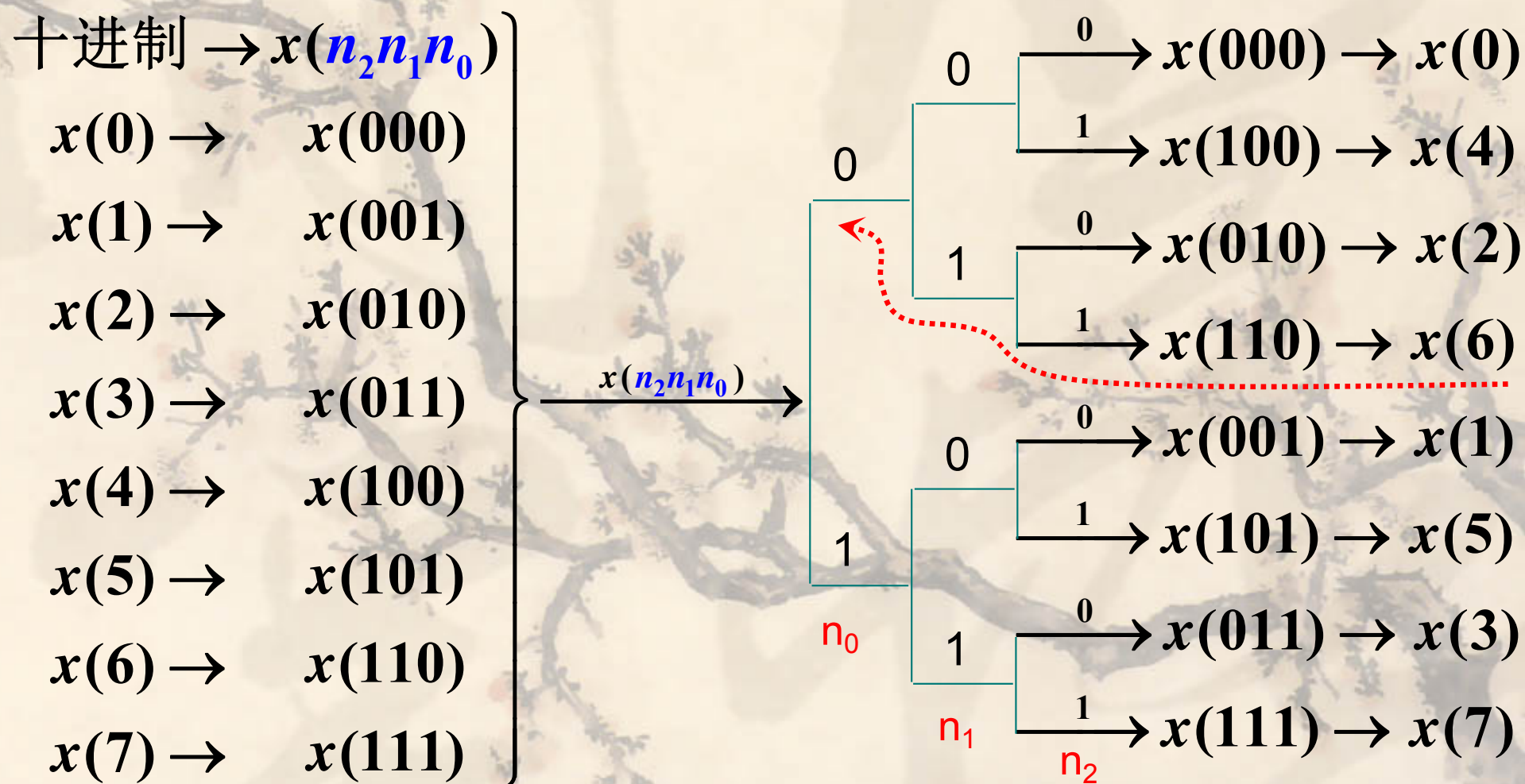
- 蝶形运算自成独立单元，即 $x_{m+1}(p)$ 、 $x_{m+1}(q)$ 只与 $x_m(p)$ 、 $x_m(q)$ 有关，而与其他节点的值无关，同时 $x_m(p)$ 、 $x_m(q)$ 也不参与另外的蝶形运算，后面的运算也不再用到前面的数据。因此，就可把计算结果 $x_{m+1}(p)$ 、 $x_{m+1}(q)$ 放入计算前 $x_m(p)$ 、 $x_m(q)$ 的存储单元中，而不用再建新的存储单元——同址计算。
- 同址计算所需存储量仅等于给定数据所需的存储量，这可大大节省存储单元。



3.4.1 FFT: 基2时间抽选法-位序重排

❖ 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。即 FFT 输出自然序列 → 输入序列 $x(n)$ 位序重排。



3.4.1 FFT: 基2时间抽选法-位序重排

■ 说明:

- ❖ 首先把 $x(n)$ 中的 n 表示为二进制。
- ❖ 假定 $N=8$, 则 $x(n) \rightarrow x(n_2n_1n_0)$ $n_i = 0, 1$
- ❖ **FFT** 的时间抽选法按 n 的奇偶分离而形成位置重排的原理如上图所示。
- ❖ 由此可以看出, 时间抽选法 **FFT** 的输入位序重排, 输出分成上下两部分, 输出结果为自然顺序。

■ 实际操作

- ❖ 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:

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

3.4.1 FFT: 基2时间抽选法-运算量

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:

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

DFT:

复数乘法: N^2

复数加法: $N(N-1)$

比较 DFT/FFT

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

N值越大, FFT方法优越性越明显

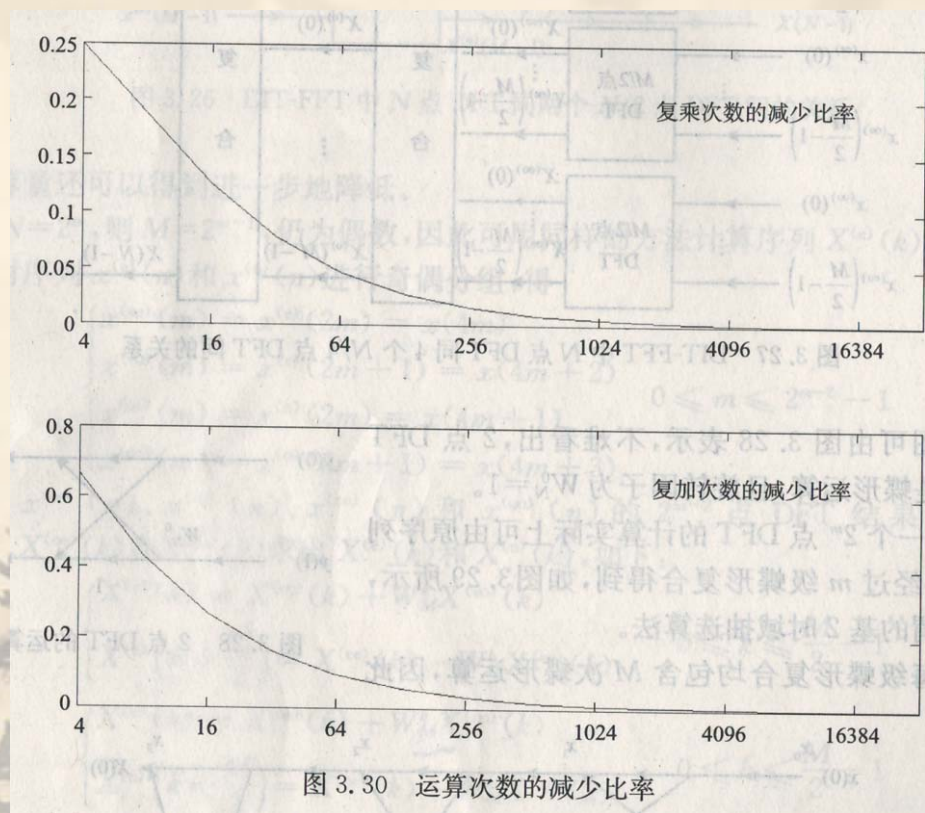


图 3.30 运算次数的减少比率

3.4.2 FFT: 基2时间抽选法-矩阵表示

❖ DFT 定义表示为矩阵形式为: $X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk}$

$$\text{DFT: } \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & W_N^2 & \cdots & W_N^{(N-1)} \\ W_N^0 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix}$$

$$\text{IDFT: } \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} = \frac{1}{N} \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^{-1} & W_N^{-2} & \cdots & W_N^{-(N-1)} \\ W_N^0 & W_N^{-2} & W_N^{-4} & \cdots & W_N^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{-(N-1)} & W_N^{-2(N-1)} & \cdots & W_N^{-(N-1)^2} \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix}$$

3.4.2 FFT: 基2时间抽选法-矩阵表示

❖ 令:

$$\mathbf{x} = \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix}$$

$$\mathbf{W}_N = \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & W_N^2 & \cdots & W_N^{(N-1)} \\ W_N^0 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix}$$

■ 矩阵 \mathbf{W}_N 为:

$$\{\mathbf{W}_N\}_{ij} = W_N^{ij}, \quad \text{for } 0 \leq i, j \leq N-1$$

特性

1) $\mathbf{W}_N^T = \mathbf{W}_N$ (对称性)

2) $\mathbf{W}_N^{-1} = \mathbf{W}_N^*$

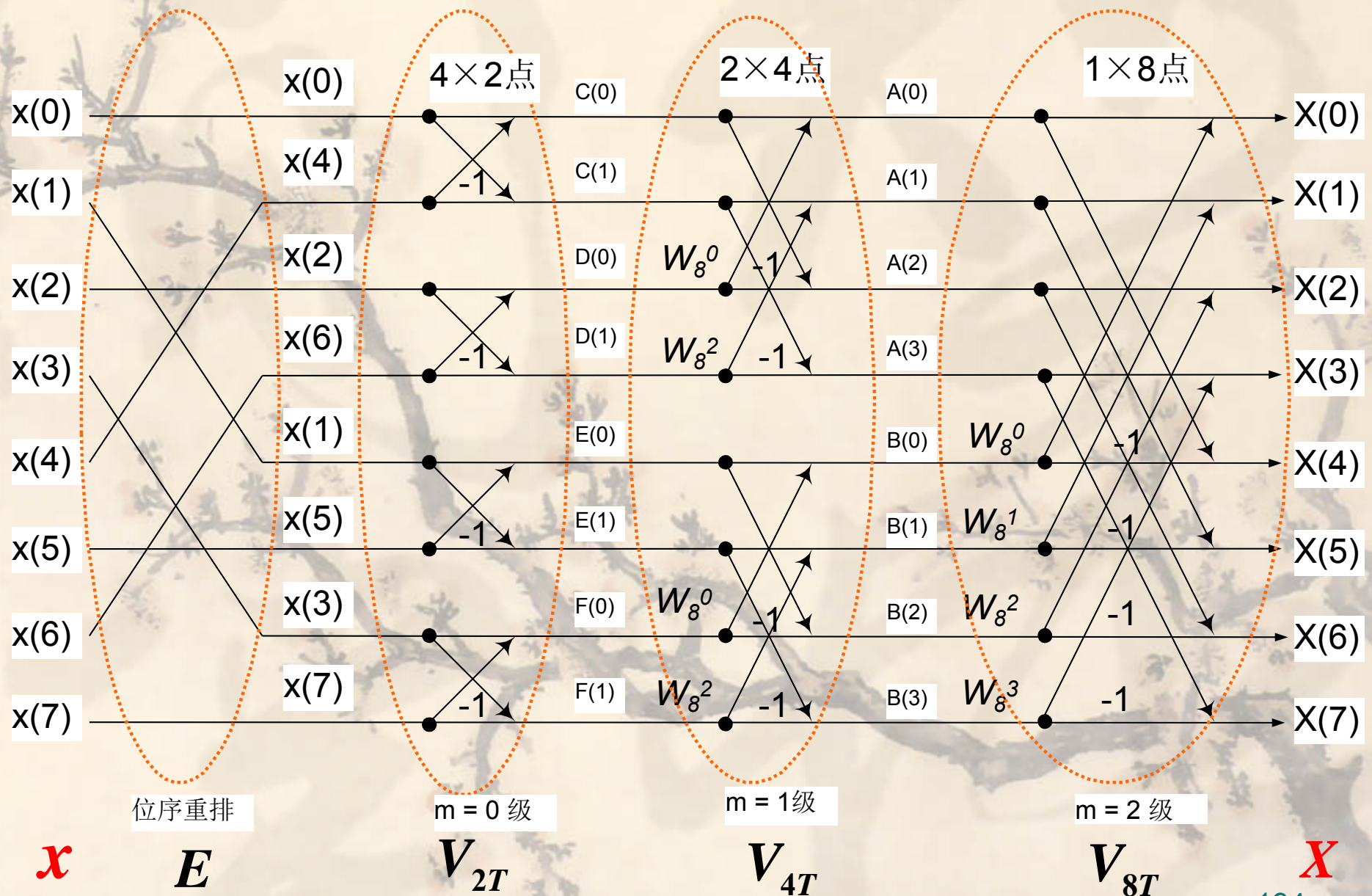
■ DFT 矩阵简单地表示为:

$$\mathbf{X} = \mathbf{W}_N \mathbf{x}$$

$$\mathbf{x} = \frac{1}{N} \mathbf{W}_N^* \mathbf{X}$$

3.4.2 FFT: 基2时间抽选法-矩阵表示

❖ FFT 基 2 时间抽选法信号流图 (N=8)



3.4.2 FFT: 基2时间抽选法-矩阵表示

❖ FFT 矩阵形式表示为:

$$X = V_{8T} V_{4T} V_{2T} E x$$

1) 输入矢量 x 与 E 相乘, 则只是输入的位序重排, 没有乘法操作。

$$Ex = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_4 \\ x_2 \\ x_6 \\ x_1 \\ x_5 \\ x_3 \\ x_7 \end{bmatrix}$$

$$E = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$$

3.4.2 FFT: 基2时间抽选法-矩阵表示

2) 第 0 级运算: 2 点 DFT

$$V_{2T}Ex = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_4 \\ x_2 \\ x_6 \\ x_1 \\ x_5 \\ x_3 \\ x_7 \end{bmatrix} = \begin{bmatrix} x_0 + x_4 \\ x_0 - x_4 \\ x_2 + x_6 \\ x_2 - x_6 \\ x_1 + x_5 \\ x_1 - x_5 \\ x_3 + x_7 \\ x_3 - x_7 \end{bmatrix} = \begin{bmatrix} C_0 \\ C_1 \\ D_0 \\ D_1 \\ E_0 \\ E_1 \\ F_0 \\ F_1 \end{bmatrix}$$

$$V_{2T} = \begin{bmatrix} 1 & W_2^0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & W_2^1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W_2^0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W_2^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W_2^0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W_2^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W_2^0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W_2^1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

3.4.2 FFT: 基2时间抽选法-矩阵表示

3) 第 1 级运算: 4 点 DFT

$$V_{4T}V_{2T}Ex = \begin{bmatrix} 1 & 0 & W_4^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_4^1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -W_4^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -W_4^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & W_4^0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_4^1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -W_4^0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -W_4^1 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ D_0 \\ D_1 \\ E_0 \\ E_1 \\ F_0 \\ F_1 \end{bmatrix} = \begin{bmatrix} C_0 + W_4^0 D_0 \\ C_1 + W_4^1 D_1 \\ C_0 - W_4^0 D_0 \\ C_1 - W_4^1 D_1 \\ E_0 + W_4^0 F_0 \\ E_1 + W_4^1 F_1 \\ E_0 - W_4^0 F_0 \\ E_1 - W_4^1 F_1 \end{bmatrix} = \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

$$V_{4T} = \begin{bmatrix} 1 & 0 & W_4^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_4^1 & 0 & 0 & 0 & 0 \\ 1 & 0 & W_4^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_4^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & W_4^0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_4^1 \\ 0 & 0 & 0 & 0 & 1 & 0 & W_4^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_4^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & W_4^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_4^1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -W_4^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -W_4^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & W_4^0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_4^1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -W_4^0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -W_4^1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_4^1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_4^1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -W_4^1 \end{bmatrix}$$

3.4.2 FFT: 基2时间抽选法-矩阵表示

4) 第 2 级运算: 8 点 DFT

$$V_{8T}V_{4T}V_{2T}Ex = \begin{bmatrix} 1 & 0 & 0 & 0 & W_8^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8^1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^3 \\ 1 & 0 & 0 & 0 & -W_8^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -W_8^1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -W_8^3 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} = \begin{bmatrix} A_0 + W_8^0 B_0 \\ A_1 + W_8^1 B_1 \\ A_2 + W_8^2 B_2 \\ A_3 + W_8^3 B_3 \\ A_0 - W_8^0 B_0 \\ A_1 - W_8^1 B_1 \\ A_2 - W_8^2 B_2 \\ A_3 - W_8^3 B_3 \end{bmatrix} = \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} = X$$

$$V_{8T} = \begin{bmatrix} 1 & 0 & 0 & 0 & W_8^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8^1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^3 \\ 1 & 0 & 0 & 0 & W_8^4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8^5 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^6 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & W_8^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8^1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^3 \\ 1 & 0 & 0 & 0 & -W_8^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -W_8^1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -W_8^3 \end{bmatrix}$$

3.4.2 FFT: 基2时间抽选法-矩阵表示

❖ FFT算法看成是 DFT 矩阵 W_8 的分解: $W_8 = V_{8T} V_{4T} V_{2T} E$

$$\begin{bmatrix} W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\ W_8^0 & W_8^1 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\ W_8^0 & W_8^2 & W_8^4 & W_8^6 & W_8^8 & W_8^{10} & W_8^{12} & W_8^{14} \\ W_8^0 & W_8^3 & W_8^6 & W_8^9 & W_8^{12} & W_8^{15} & W_8^{18} & W_8^{21} \\ W_8^0 & W_8^4 & W_8^8 & W_8^{12} & W_8^{16} & W_8^{20} & W_8^{24} & W_8^{28} \\ W_8^0 & W_8^5 & W_8^{10} & W_8^{15} & W_8^{20} & W_8^{25} & W_8^{30} & W_8^{35} \\ W_8^0 & W_8^6 & W_8^{12} & W_8^{18} & W_8^{24} & W_8^{30} & W_8^{36} & W_8^{42} \\ W_8^0 & W_8^7 & W_8^{14} & W_8^{21} & W_8^{28} & W_8^{35} & W_8^{42} & W_8^{49} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & W_8^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8^1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^3 \\ 1 & 0 & 0 & 0 & W_8^4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8^5 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^6 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^7 \end{bmatrix} \begin{bmatrix} 1 & 0 & W_4^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_4^1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -W_4^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -W_4^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & W_4^0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_4^1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -W_4^0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -W_4^1 \end{bmatrix} \begin{bmatrix} 1 & W_2^0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & W_2^1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W_2^0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W_2^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W_2^0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W_2^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W_2^0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W_2^1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

✧ 这个因式分解对应于快速算法, 因为矩阵 V_{8T} , V_{4T} , V_{2T} 和 E 的大部分元素为 0, 是稀疏矩阵。因此上述每个矩阵的乘法运算最多只需要 8 次复乘运算, 而 E 只是置换操作, 没有乘法操作。

■ 不同的 FFT 算法对应于将 DFT 矩阵 W_N 分解成不同的稀疏矩阵。

3.4.3 FFT: 基2频率抽选法 (DIF-FFT)

❖ **算法原理:** 根据时间-频率的对偶性

- **时间抽选法:** 是把输入 $x(n)$ 按奇偶分解成两个子序列, 即 N 点 $x(n)$ 序列 $\rightarrow N/2$ 点子序列, 而输出 $X(k)$ 是按自然顺序排列的。
- **频率抽选法:** 是把输入 $x(n)$ 按照前后对半分开, 而不是奇偶数分开, 而输出 $X(k)$ 逐项分解成偶数点子序列和奇数点子序列。

❖ **DFT 变换表达式为:**

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

如果将输入 $x(n)$ 按前后等分, 即将求和分成两部分, 范围分别为:

$$n = 0 \sim \left(\frac{N}{2} - 1\right)$$

$$n = \frac{N}{2} \sim (N - 1)$$

3.4.3 FFT: 基2频率抽选法 – 算法原理

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

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

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

$$W_N^{N/2} = -1$$

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

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

3.4.3 FFT: 基2频率抽选法 – 算法原理

按 k 的奇偶将 $X(k)$ 分成两部分:
$$\begin{cases} k = 2r \\ k = 2r + 1 \end{cases} \quad r = 0, 1, \dots, \frac{N}{2} - 1$$

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

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

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

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

3.4.3 FFT: 基2频率抽选法—算法原理

令:
$$\begin{cases} k \text{ 为偶数:} & x_1(n) = x(n) + x\left(n + \frac{N}{2}\right) \\ k \text{ 为奇数:} & x_2(n) = \left[x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n \end{cases}$$

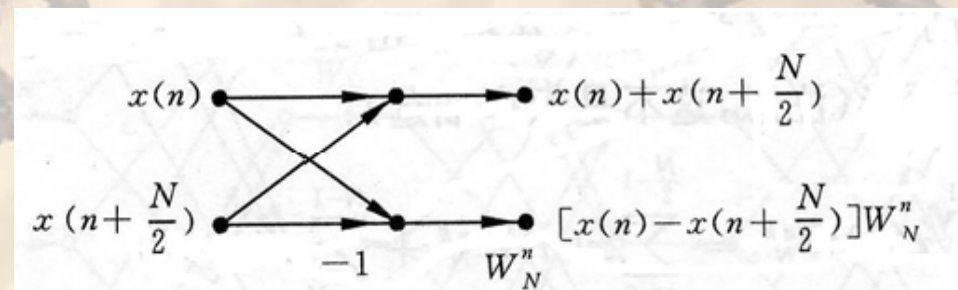
$n = 0, 1, \dots, \frac{N}{2} - 1$

则 $X(2r)$ 和 $X(2r+1)$ 分别是 $x_1(n)$ 和 $x_2(n)$ 的 $N/2$ 点 DFT, 记为 $X_1(k)$ 和 $X_2(k)$ 。

k 为偶数: $X_1(k) = X(2r) = \sum_{n=0}^{\frac{N}{2}-1} x_1(n) W_{N/2}^{nr}$

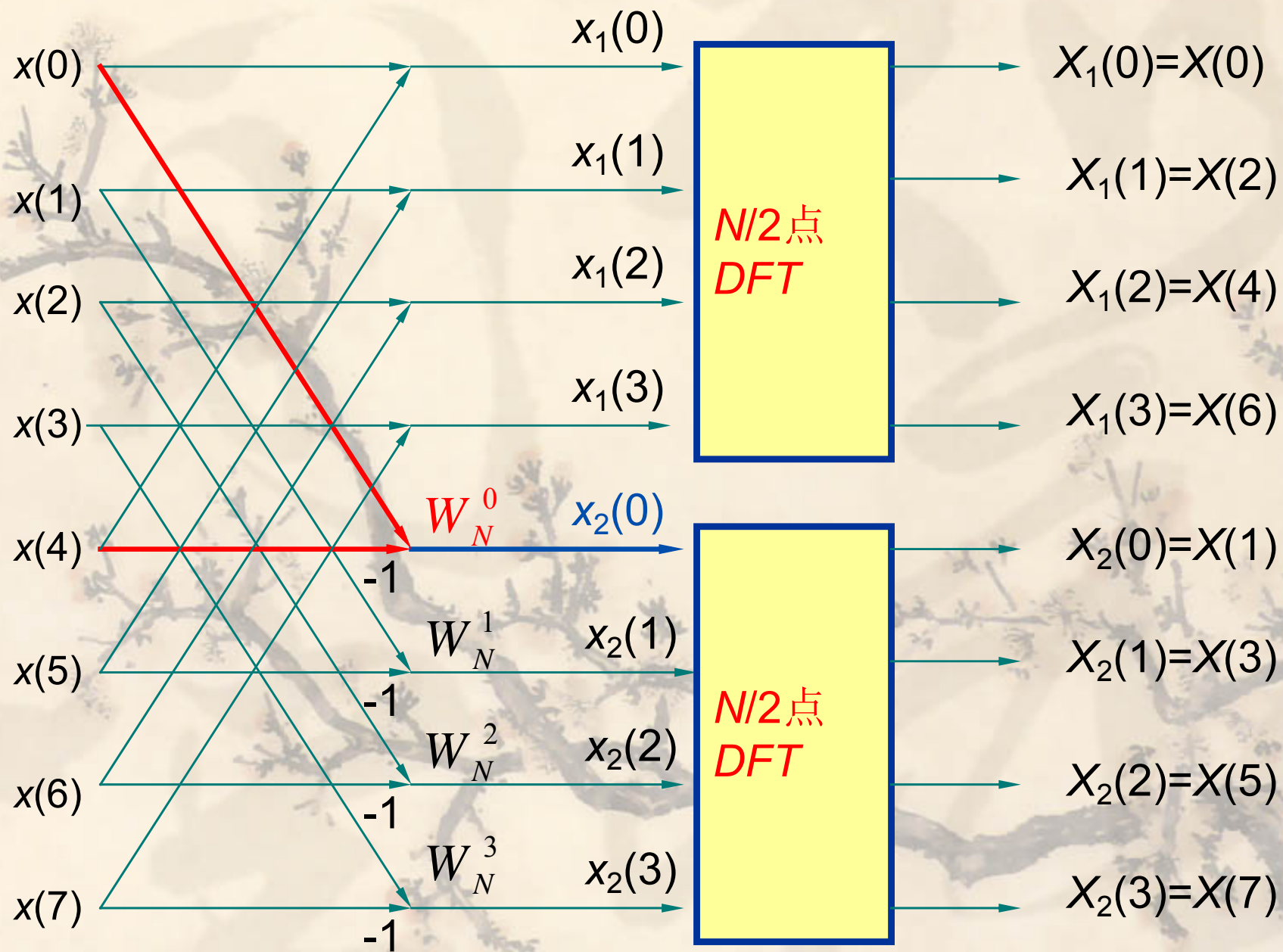
k 为奇数: $X_2(k) = X(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} x_2(n) W_{N/2}^{nr}$

蝶形运算



1次复乘和2次复加

3.4.3 FFT: 基2频率抽选法 – 算法原理

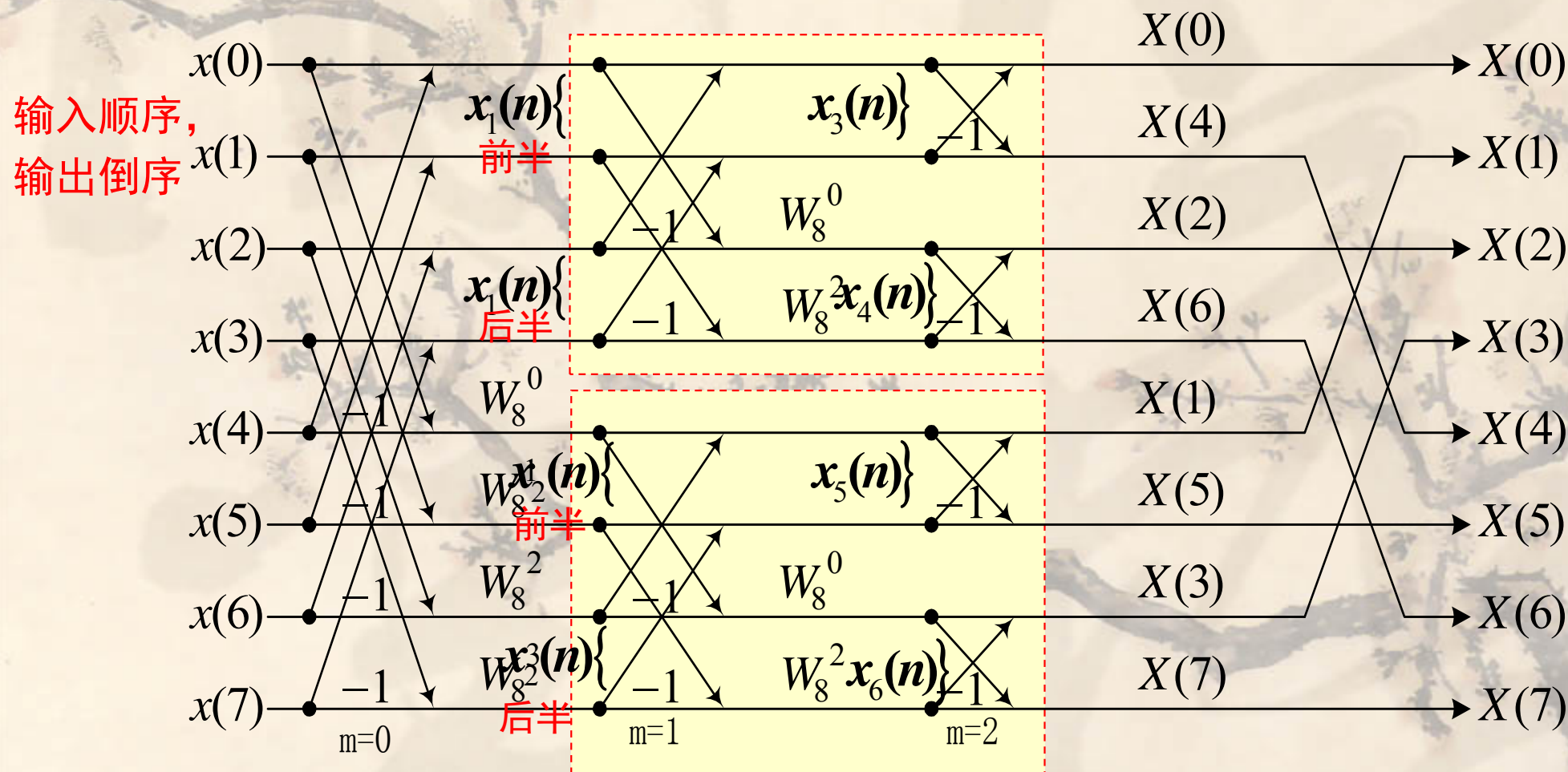


3.4.3 FFT: 基2频率抽选法—算法原理

第二次分解，每个N/2点DFT分解为2个N/4点DFT

$$\begin{cases} x_5^3(n) = x_2^1(n) + x_2^1\left(n + \frac{N}{4}\right) \\ x_6^4(n) = (x_2^1(n) - x_2^1\left(n + \frac{N}{4}\right))W_{\frac{N}{2}}^n \end{cases}$$

逐级分解，直到 2 点 DFT



经 L 级分解，只有蝶算，没有DFT。

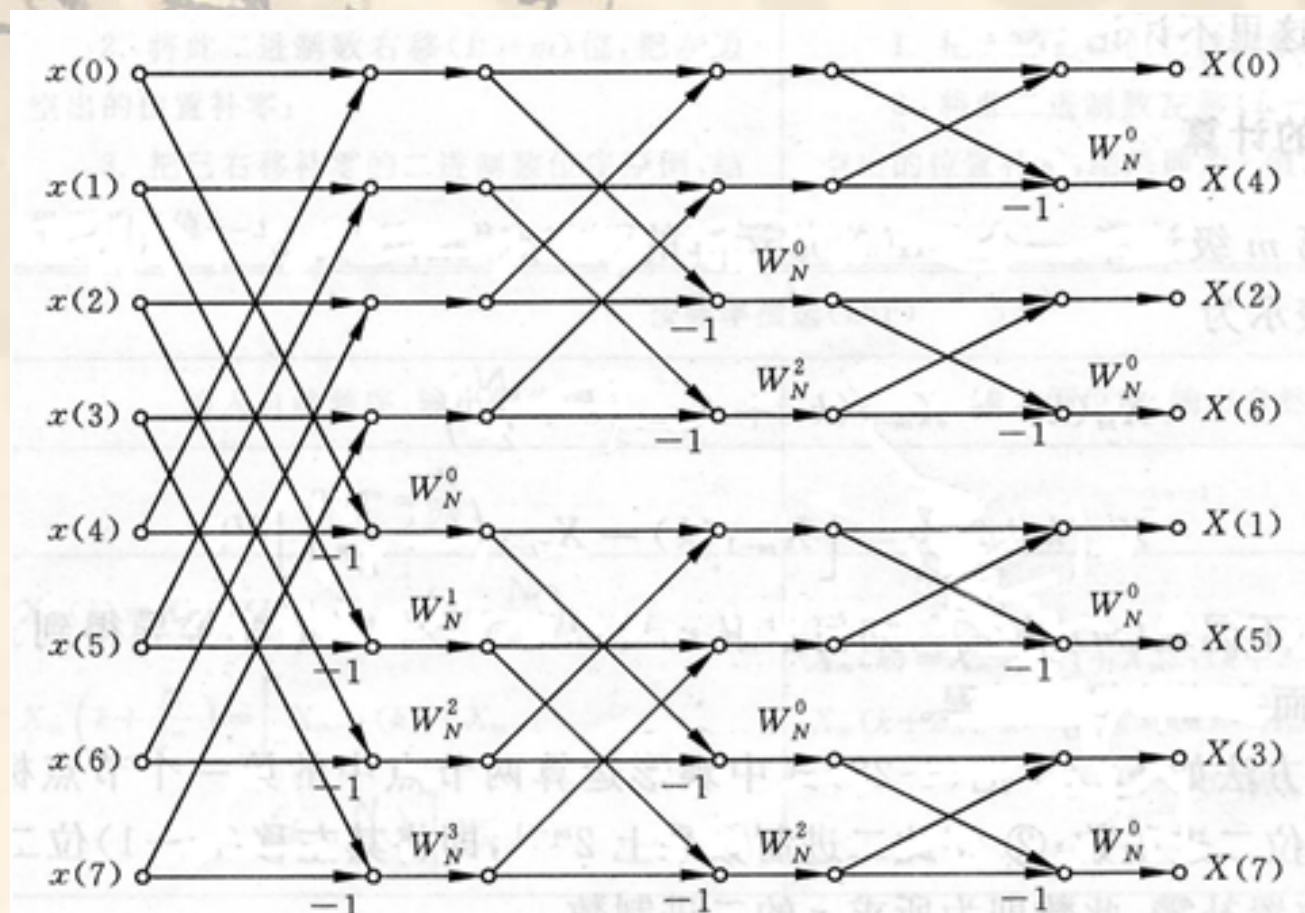
算法特点：

L 级：每级 $\frac{N}{2}$ 个蝶形

运算量：复乘 $\frac{N}{2} \log_2 N$

复加 $N \log_2 N$

同DIT：输入顺序，输出倒序

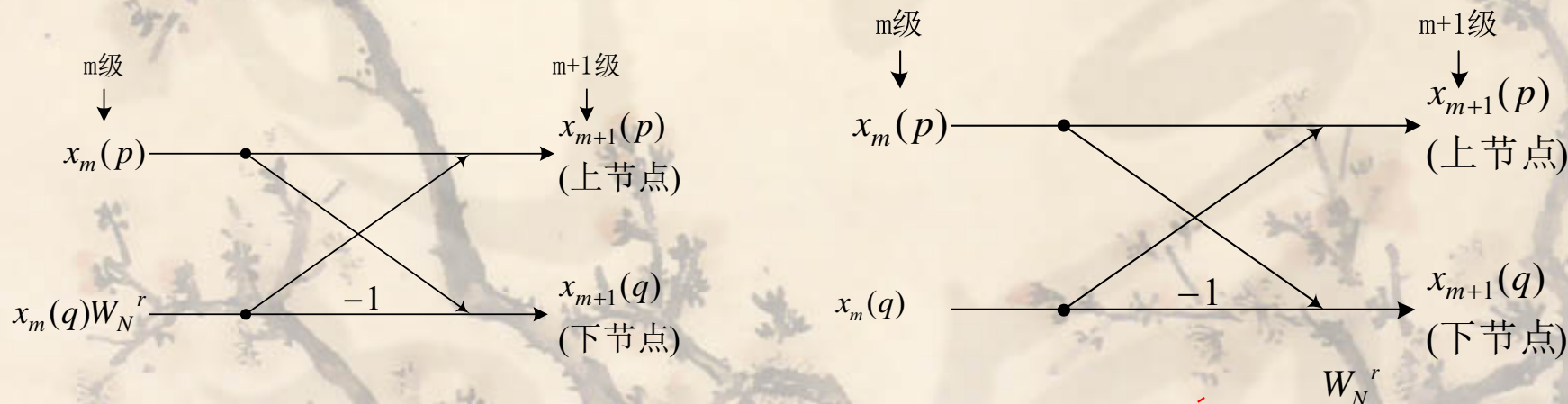


3.4.3 FFT: 基2时间和频率抽选法的异同

❖ 基本蝶形运算不同

∞ **DIT**: 先复乘后加减, **W** 因子在上下节点都有体现

$$\begin{cases} x_{m+1}(p) = x_m(p) + W_N^r x_m(q) \\ x_{m+1}(q) = x_m(p) - W_N^r x_m(q) \end{cases}$$



❖ **DIF**: 先减后复乘, **W** 因子仅体现在下节点

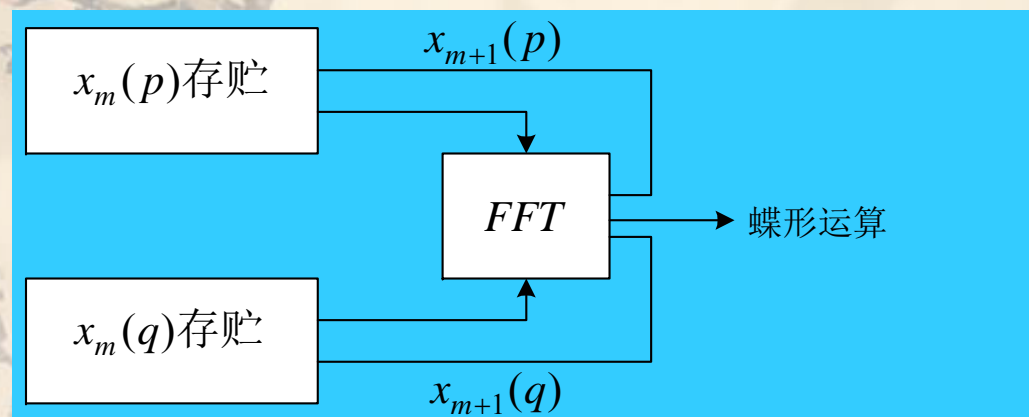
$$\begin{cases} x_{m+1}(p) = x_m(p) + x_m(q) \\ x_{m+1}(q) = [x_m(p) - x_m(q)] W_N^r \end{cases}$$

3.4.3 FFT: 基2时间和频率抽选法的异同

■ 运算量相同

$$\text{乘法: } m_F = \frac{N}{2} \log_2 N \quad \text{加法: } a_F = N \log_2 N$$

■ 同址计算



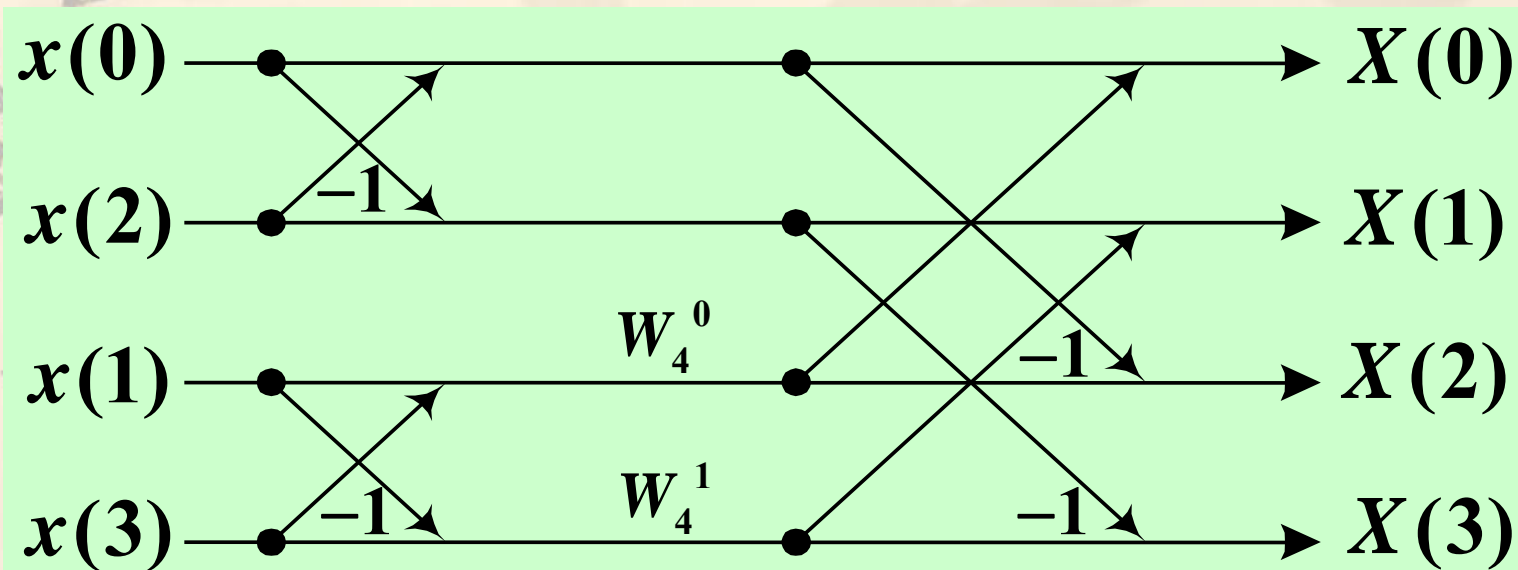
■ DIF 输出位序重排, DIT 输入位序重排

$$X(n_{M-1} \cdots n_2 n_1 n_0) \xrightarrow{\text{倒置}} X(n_0 n_1 \cdots n_{M-1})$$

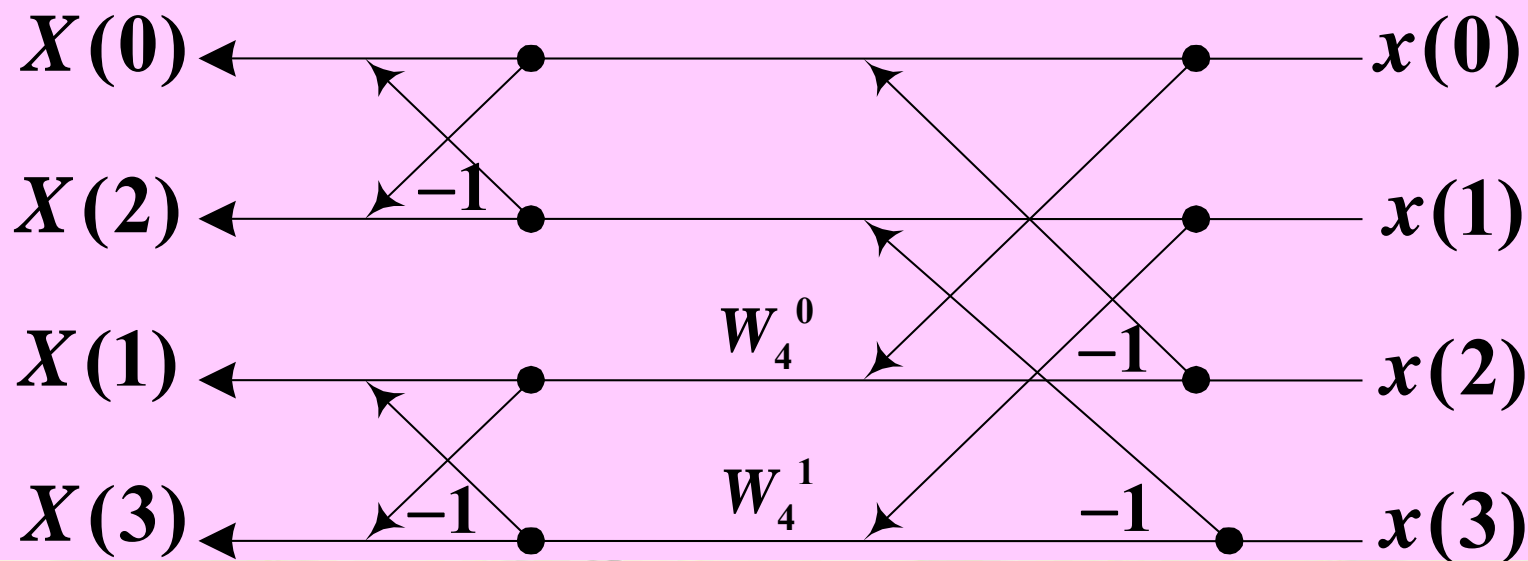
3.4.3 FFT: 基2时间和频率抽选法的异同

■ *DIT* 和 *DIF* 的基本蝶形互为信号流图转置

DIT

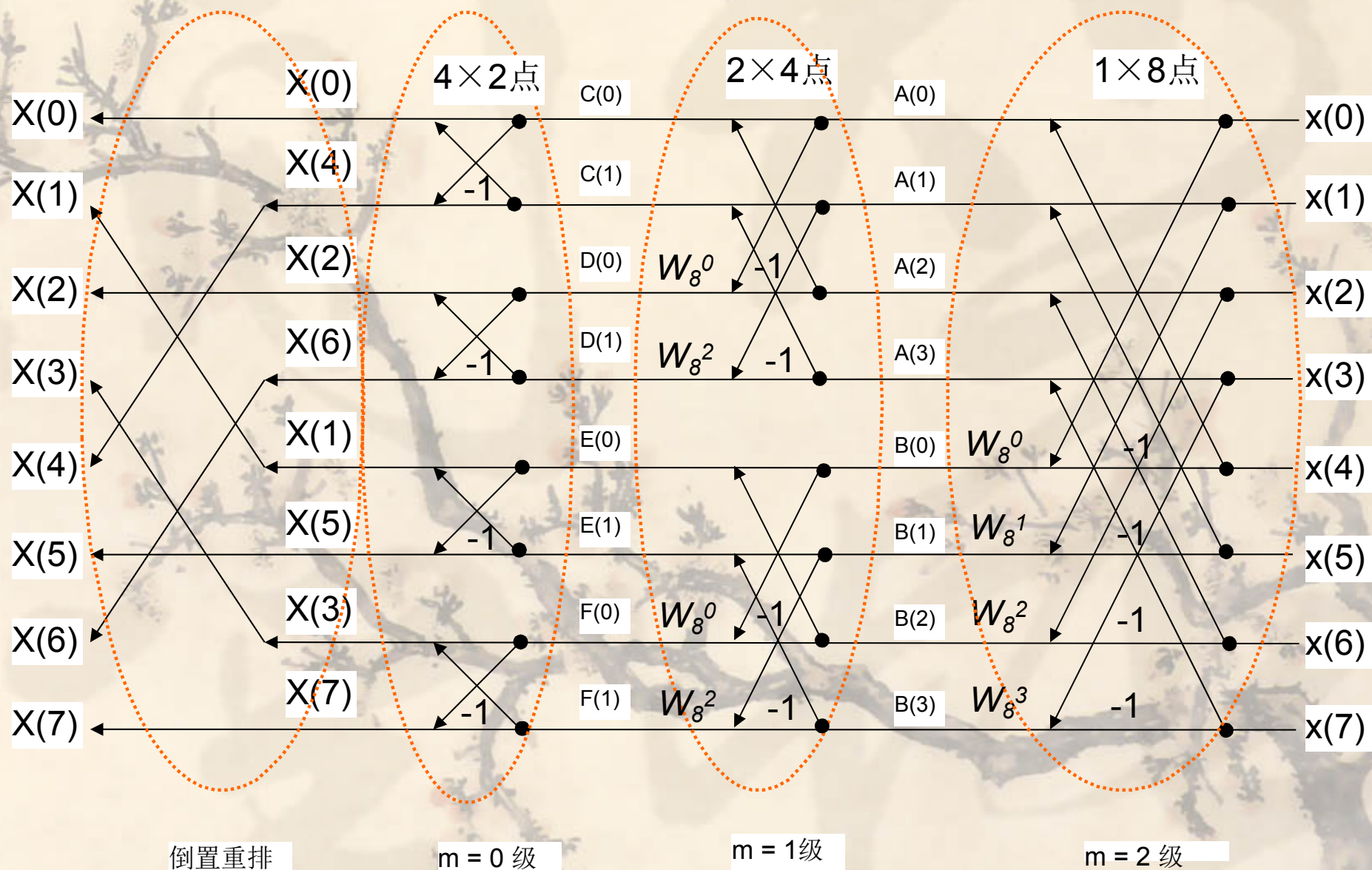


DIF



3.4.3 FFT: 基2时间和频率抽选法的异同

■ DIT-DIF FFT 算法互为转置（转置定理）



3.4.3FFT：基2抽选法-其它形式的流图

- ❖ 输入倒位序，输出自然序（**DIT**）
- ❖ 输入自然序，输出倒位序（**DIF**）
- ❖ 输入输出均自然序
- ❖ 各级具有相同几何形状
 - 输入倒位序，输出自然序
 - 输入自然序，输出倒位序

思考题：

是不是任何情况下，**FFT**都可以有效地降低运算量？

3.4.4 FFT: 基 4 时间抽选法

- ❖ 前面讲的都是基 2 的 FFT 算法，除此之外，还有基 4 的，基 8 的快速算法。原理和基 2 的类似，分解为 4 个交错的集合。相比基 2 的，可以进一步节约复数乘的次数，但是基 8 的和基 4 的差别不远。
- ❖ 算法原理省略，自学。
- ❖ 注：有些点数长度不是 2 的整数倍时，如采用基 2 的时域抽取，尾部加零较多，限制了 DFT 的计算，实际中许多采用多基多进制表示为基础。

3.4.4 FFT: 基4时间抽选法

❖ 考虑 $x(n)$ 的 N 点DFT结果为 $X(k)$ 且 $N=4^m$ ，按照模4对序列 $x(n)$ 分组

$$x^{(0)}(n) = x(4n); x^{(1)}(n) = x(4n+1)$$

$$x^{(2)}(n) = x(4n+2); x^{(3)}(n) = x(4n+3), 0 \leq n \leq \frac{N}{4} - 1$$

$$\begin{aligned} X(k) &= \sum_{l=0}^{4^{m-1}-1} x(4l)W_N^{4lk} + \sum_{l=0}^{4^{m-1}-1} x(4l+1)W_N^{(4l+1)k} \\ &\quad + \sum_{l=0}^{4^{m-1}-1} x(4l+2)W_N^{(4l+2)k} + \sum_{l=0}^{4^{m-1}-1} x(4l+3)W_N^{(4l+3)k} \\ &= \sum_{l=0}^{4^{m-1}-1} x^{(0)}(l)W_{4^{m-1}}^{lk} + W_N^k \sum_{l=0}^{4^{m-1}-1} x^{(1)}(l)W_{4^{m-1}}^{lk} + \\ &\quad W_N^{2k} \sum_{l=0}^{4^{m-1}-1} x^{(2)}(l)W_{4^{m-1}}^{lk} + W_N^{3k} \sum_{l=0}^{4^{m-1}-1} x^{(3)}(l)W_{4^{m-1}}^{lk} \\ &= X^{(0)}((k))_{4^{m-1}} + W_N^k X^{(1)}((k))_{4^{m-1}} + \\ &\quad W_N^{2k} X^{(2)}((k))_{4^{m-1}} + W_N^{3k} X^{(3)}((k))_{4^{m-1}} \quad 0 \leq k \leq 4^{m-1} - 1 \end{aligned}$$

3.4.4 FFT: 基4 时间抽选法

其中

$$X^{(0)}(k) = \text{DFT}_{4^{m-1}}\{x^{(0)}(n)\}; X^{(1)}(k) = \text{DFT}_{4^{m-1}}\{x^{(1)}(n)\}$$

$$X^{(2)}(k) = \text{DFT}_{4^{m-1}}\{x^{(2)}(n)\}; X^{(3)}(k) = \text{DFT}_{4^{m-1}}\{x^{(3)}(n)\}$$

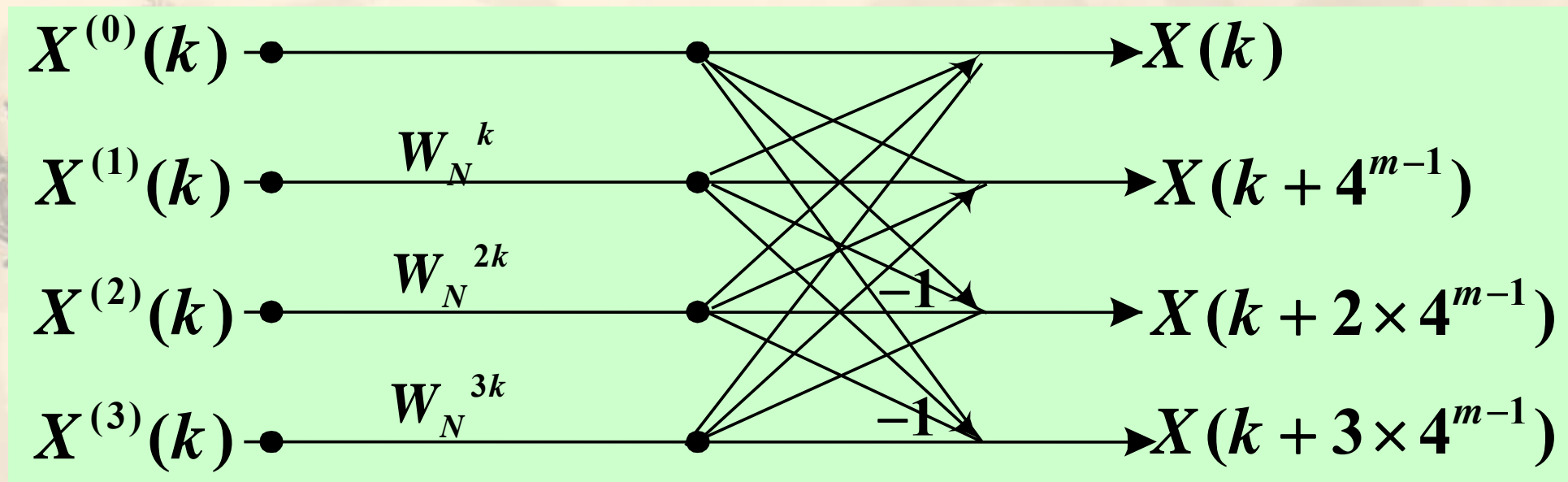
$$\begin{cases} X(k) = X^{(0)}(k) + W_N^k X^{(1)}(k) + W_N^{2k} X^{(2)}(k) + W_N^{3k} X^{(3)}(k) \\ X(k + 4^{m-1}) = X^{(0)}(k) - jW_N^k X^{(1)}(k) - W_N^{2k} X^{(2)}(k) + jW_N^{3k} X^{(3)}(k) \\ X(k + 2 \times 4^{m-1}) = X^{(0)}(k) - W_N^k X^{(1)}(k) + W_N^{2k} X^{(2)}(k) - W_N^{3k} X^{(3)}(k) \\ X(k + 3 \times 4^{m-1}) = X^{(0)}(k) + jW_N^k X^{(1)}(k) - W_N^{2k} X^{(2)}(k) - jW_N^{3k} X^{(3)}(k) \end{cases}$$

旋转因子为 W_N^k W_N^{2k} 和 W_N^{3k}

j 在同一复数相乘时, 只要调换该复数实部和虚部的位罝并相应的改变符号便可以得出结果。

$$j(a + jb) = -b + ja$$

3.4.4 FFT: 基 4 时间抽选法



3.4.4 FFT: 基4时间抽选法

一个N点DFT计算转换为4个 $\frac{N}{4}$ 的DFT计算和一级蝶形复合直接计算运算量:

- 一个4点DFT蝶形只需3次乘旋转因子, 和12个复加运算, 则共需 $\frac{3N}{4}$ 次乘旋转因子, 和 $3N$ 复加运算(每级 $N/4$ 个蝶形)。
- 4个 $\frac{N}{4}$ 的DFT共需 $\frac{N^2}{4}$ 复乘和 $N(\frac{N}{4}-1)$ 复加。
- 总运算量 $\frac{N(N+3)}{4}$ 复乘和 $N(\frac{N}{4}+2)$ 复加

每级有 $N/4$ 个4点FFT, 共 m 级 ($m-1$ 级要乘旋转因子)

一个4点DFT蝶形只需3次乘旋转因子, 和12个复加运算,
蝶形运算共需 $\frac{3N \log_4(N)}{4}$ 次复乘和 $3N \log_4(N)$ 复加运算

而基-2FFT $\frac{N}{2} \log_2 N$ 次复乘和 $N \log_2 N$ 复加运算

注: 复乘次数减少, 复加次数有所增加。

3.4.5 FFT: IDFT 快速算法 (IFFT)

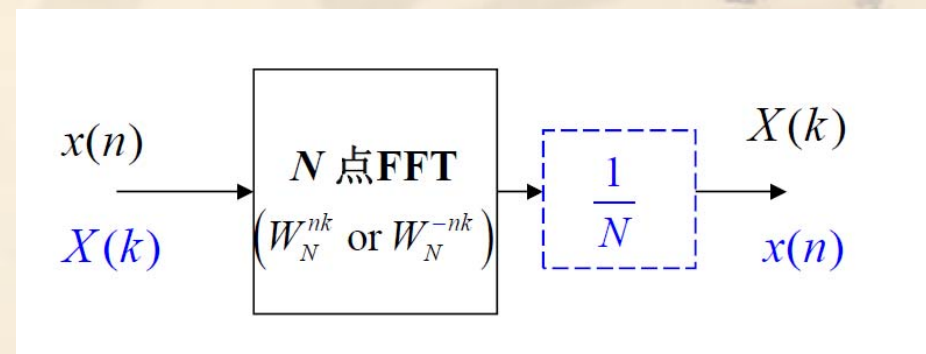
■ DFT 和 IDFT 的定义:

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

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

■ DFT 和 IDFT 的区别:

- ① 因子 W 的指数相差一个负号;
- ② 相差一个因子 $1/N$ 。



$$FFT : W_N^{nk} \rightarrow W_N^{-nk} , \quad (\times \frac{1}{N}) \rightarrow IFFT$$

FFT 算法中的分组方式、排序方式以及蝶形运算结构都可用于 IFFT 算法的设计，而这就是可依据现有的 FFT 算法直接得出 IFFT 算法的原因。

3.4.5 FFT: IDFT 快速算法 (IFFT)

❖ FFT 和 IFFT 基本蝶形运算之间的关系

∞ 设有序列 $x(n)$, 其 DFT 为 $X(k)$, 则 IDFT 为:

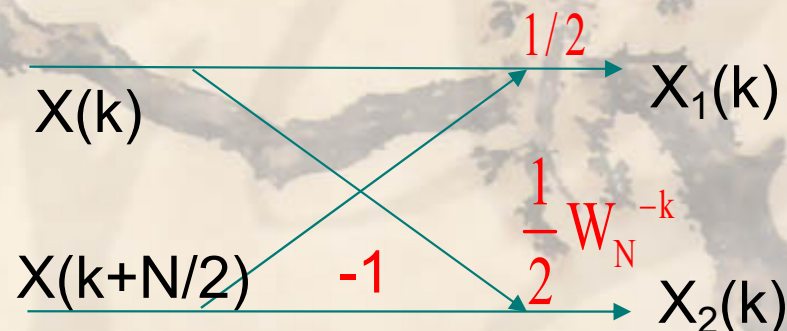
$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk}$$

✧ 在 FFT 的时间抽选法中:

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

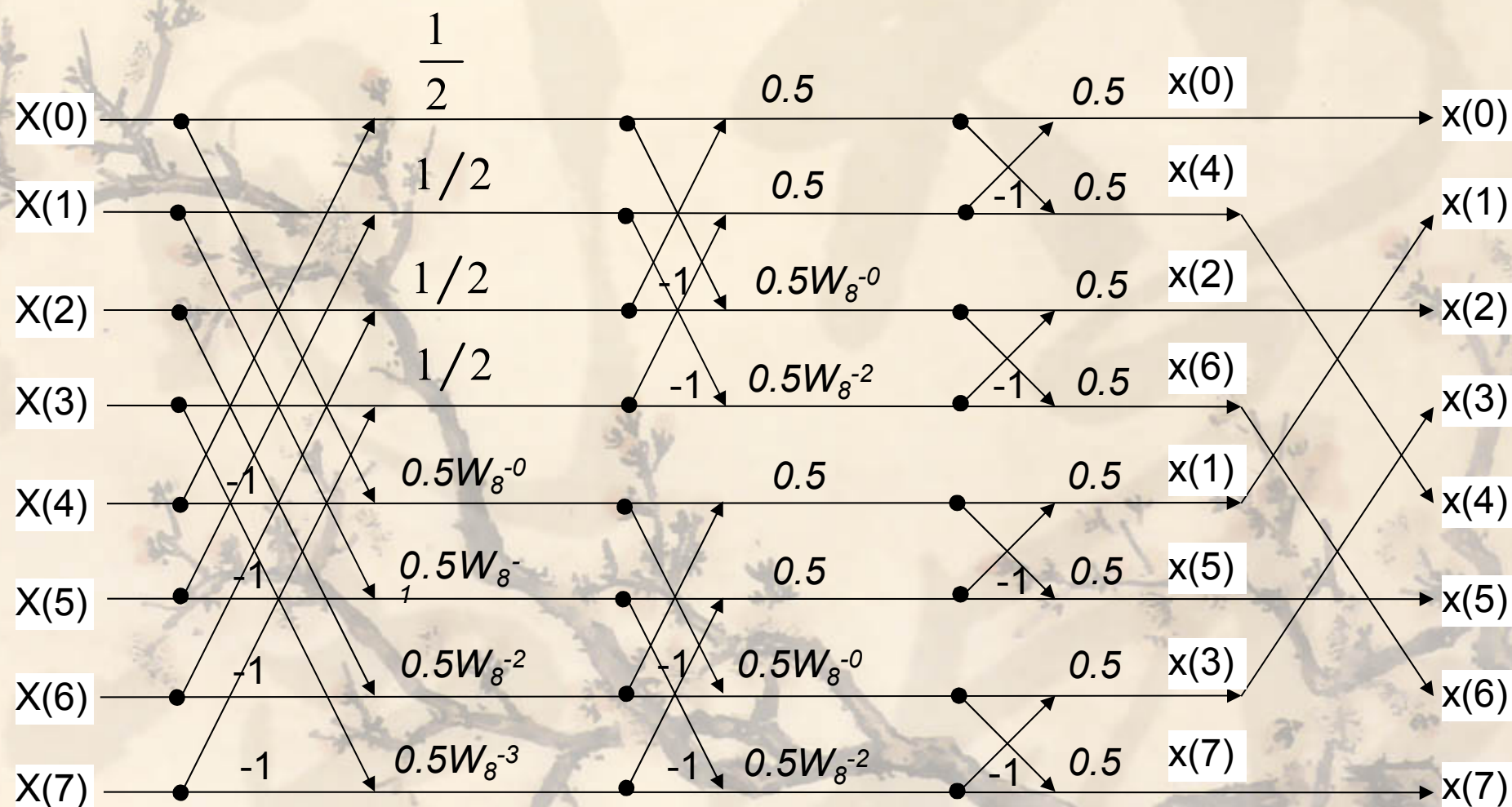
✧ 对于 IFFT 算法, 输入是 $X(k)$ 和 $X(k+N/2)$, 输出是 $X_1(k)$ 和 $X_2(k)$ 。
解上式可以得到 $X_1(k)$ 、 $X_2(k)$:

$$\begin{cases} X_1(k) = \frac{1}{2} [X(k) + X(k + \frac{N}{2})] \\ X_2(k) = \frac{1}{2} W_N^{-k} [X(k) - X(k + \frac{N}{2})] \end{cases}$$



3.4.5 FFT: IDFT 快速算法 (IFFT)

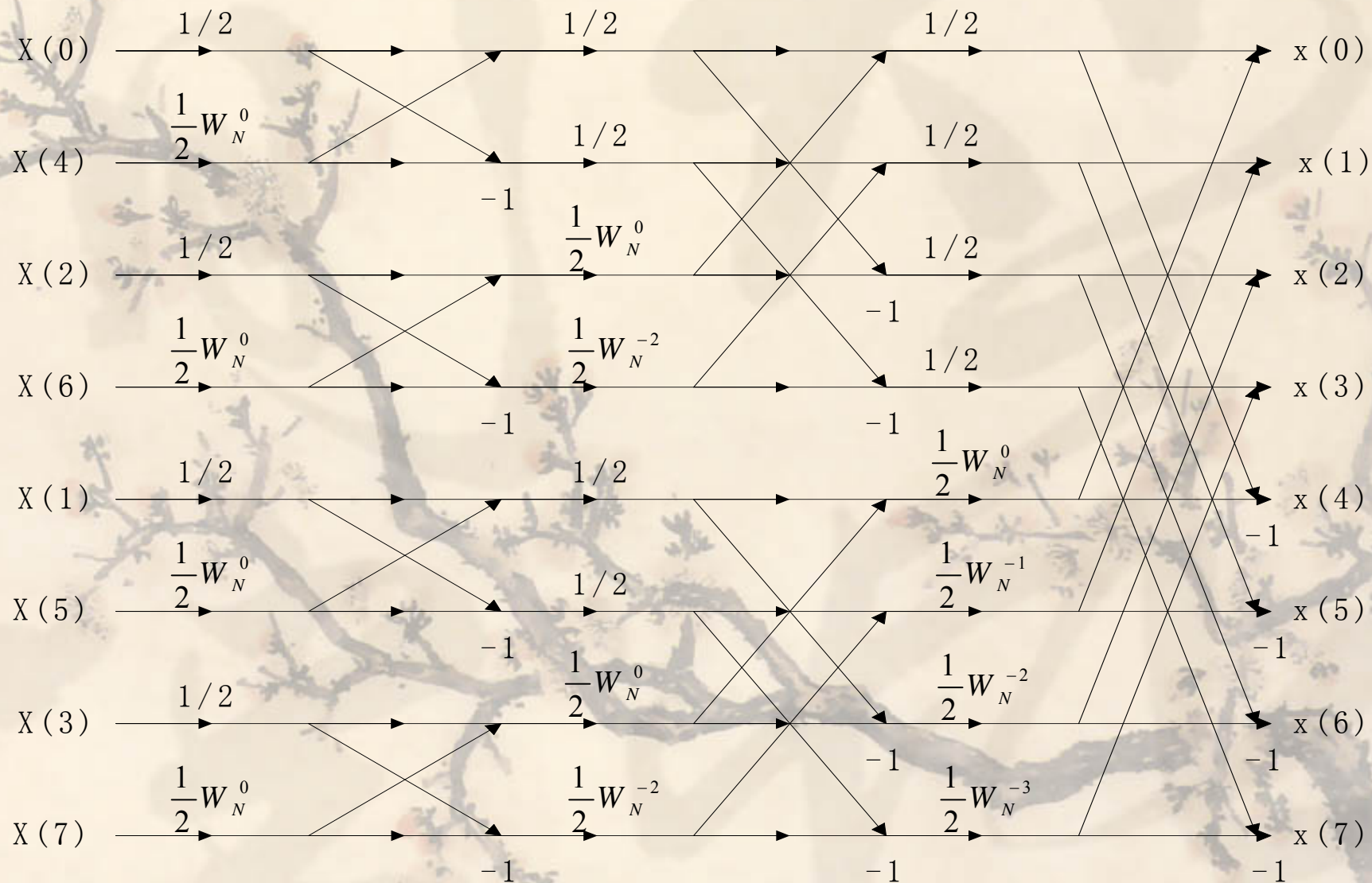
8 点 DIT-IFFT 算法



说明：1) 分组过程是按时间序列 $x(n)$ 的奇偶性在时域上展开的，故称此法为时间抽选算法 DIT-IFFT；2) $1/N$ 的分解， $N=2^m$ 。

3.4.5 FFT: IDFT 快速算法 (IFFT)

❖ 8 点 DIF-IFFT 算法



3.4.5 FFT: IDFT 快速算法 (IFFT)

❖ 用 FFT 程序求 IFFT 的方法

✧ 直接法：利用 DIT、DIF 的 FFT 程序，改变参量

- ① 把 $X(k)$ 作为输入序列，而输出序列就是 $x(n)$;
- ② 把因子 W_N^{kn} 改为 W_N^{-kn} ;
- ③ 输入序列的每一个元素除以 N 。

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

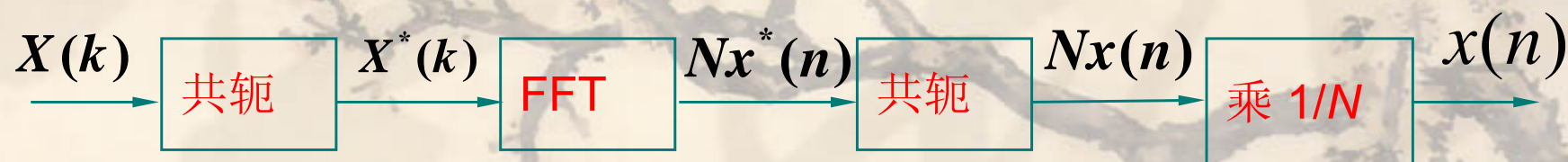
$$DFT : X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

3.4.5 FFT: IDFT 快速算法 (IFFT)

✧ 共轭法

$$\begin{aligned}x^*(n) &= \frac{1}{N} \left[\sum_{k=0}^{N-1} X(k) W_N^{-nk} \right]^* \\&= \frac{1}{N} \left[\sum_{k=0}^{N-1} X^*(k) W_N^{nk} \right] \\&= \frac{1}{N} [\text{FFT / DFT}[X^*(k)]], \quad n = 0, 1, \dots, N-1\end{aligned}$$

流程如下:



提纲:

3.1 问题的提出

3.2 DFS（离散傅里叶级数）

3.3 DFT（有限离散傅里叶变换）

3.4 FFT（快速离散傅里叶变换）

3.5 FFT的应用

- 线性卷积的快速计算

- CZT及其快速计算

3.6 其它变换

3.7 本章小结

3.5.1 线性卷积的快速计算

■ 有限长序列的线性卷积

$$x(n), \quad 0 \leq n \leq N-1 \quad h(n), \quad 0 \leq n \leq M-1$$

$$y(n) = x_N(n) * h_M(n) = \sum_{m=0}^{N-1} x_N(m) h_M(n-m) = \sum_{m=0}^{N-1} h_M(m) x_N(n-m), \quad 0 \leq n \leq N+M-2$$

用定义式直接计算线性卷积（直接卷积），共 MM 次复乘，以及 $(N-1)(M-1)$ 复数加法。

用FFT实现线性卷积（快速卷积），做3个 L 点FFT，频域 L 点复数乘。

$$x_N(n) * h_M(n) \xrightarrow{L \leq N+M-1} x_N(n) \otimes h_M(n)$$

(1) 补零 $x_N(n) \rightarrow x_L(n), h_M(n) \rightarrow h_L(n)$

(2)

(3) $Y(k) = X(k) \cdot H(k)$

(4) $Y(k) \rightarrow y(n)$

计算量：复乘次数 $L + \frac{3L}{2} \log_2 L$ ， $3L \log_2 L$ 复加次数

3.5.1 线性卷积的快速计算

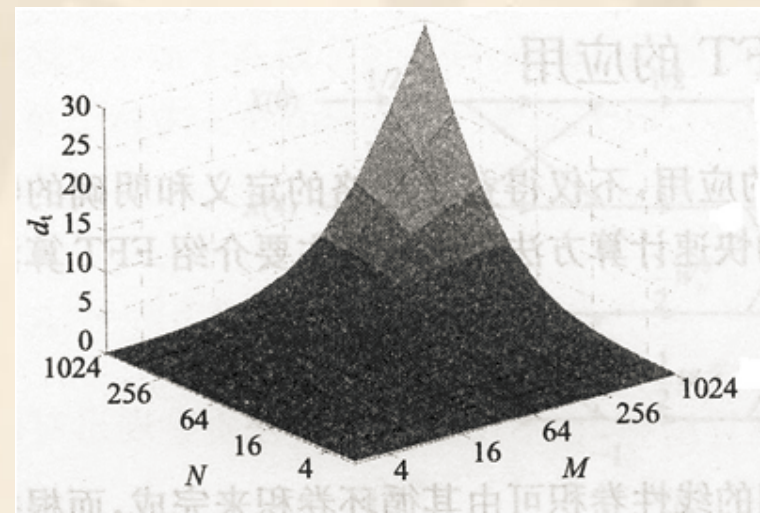
- ❖ **N**与**M**接近，两者均较大时，快速卷积的运算量低于直接卷积

直接卷积

- ❖ 复数乘次数之比 $d_i = \frac{NM}{\frac{3L}{2}\log_2 L + L}$

快速卷积

- ❖ 为了做基**2FFT**，取 $L=2^m \geq N+M-1$



(2) 逐段卷积

$h(n) \rightarrow M$ $x(n), N \gg M$ 或者无限长 $N \rightarrow \infty$

$x(n)$ 分段，使得段长与**M**接近；每一段做快速卷积；组合计算

3.5.2 线性调频Z变换 (CZT) 及其快速计算

DFT 不适用于:

- DFT 是均匀分布在 z 平面单位圆上 N 点处的频谱, 如果取
样点不均匀时, 则很麻烦。
- 只研究信号的某一频段, 要求对该频段取样密集, 提高分辨率(如窄带信号的频谱分析);
- 研究非单位圆上的取样值 (如频谱峰值探测, 极点需密集);
- 需要准确计算 N 点 DFT, 且 N 为大的素数;
- 当 $x(n)$ 是短时间序列时, 则得到的频率分辨率 $2\pi/N$ 是很低的。提高频谱密度的办法: 用补零的方法增加点数, 但 DFT 的点数又大大增加, 使计算工作量增大。

CZT 算法: 对 z 变换采用螺旋线取样, chirp- z 变换(线性调频 z 变换, **Chirp z -transform**)

3.5.1 CZT: 定义

❖ CZT 的定义

设 $x(n)$ 为已知时间序列，其 Z 变换的形式为：

$$Z[x(n)] = X(z) = \sum_{n=0}^{N-1} x(n)z^{-n}$$

式中复变量 z 为：

$$z = e^{sT} = e^{\sigma T} e^{j\Omega T} = Ae^{j\omega}$$

这里 s 为拉普拉斯变量， $s = \sigma + j\Omega$

$$A = e^{\sigma T} \quad \text{是个实数}$$

$$\omega = \Omega T$$

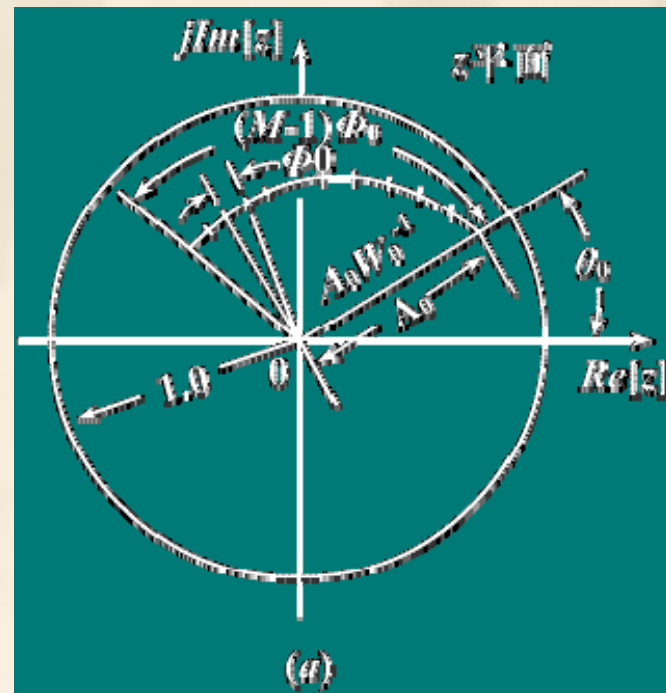
3.5.1 CZT: 定义

按照上式计算 $Z[x(n)]$ 必然是从 z 的实轴开始, 为得到任意起始点和以螺旋线规律变化的 z 值, 做如下假设:

$$\left\{ \begin{aligned} A &= A_0 e^{j\theta_0} \\ W &= W_0 e^{-j\phi_0} \\ z_k &= A W^{-k} \\ &= A_0 e^{j\theta_0} W_0^{-k} e^{jk\phi_0} \\ &= A_0 W_0^{-k} e^{j(\theta_0 + k\phi_0)} \end{aligned} \right.$$

$$k = 0, 1, 2, \dots, M-1$$

其中, A 、 W 为任意复数, θ_0 为 A 的起始角 (第1个取样点 ($k=0$)) , A_0 为 A 起始半径, ϕ_0 为在 Z 平面中相邻的 z_k (即 z_k 和 z_{k+1}) 之间的夹角, W_0 为任意正数值。 M 为要分析的点数, 不一定等于 N 。CZT 不受取样轨迹、取样点数以及取样点位置的约束。

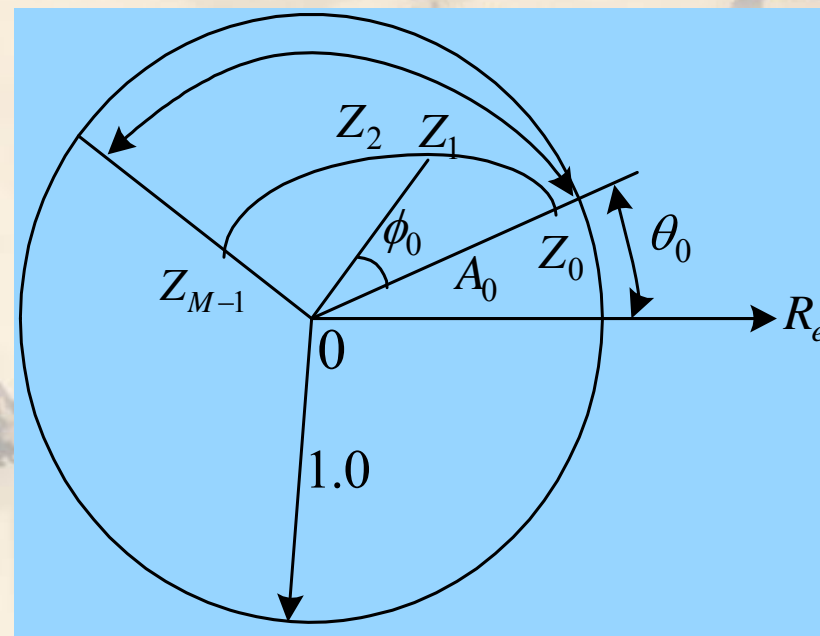


3.5.1 CZT: 定义

❖ CZT 表达式

$$\begin{aligned} CZT[x(n)] &= X(z_k) = \sum_{n=0}^{N-1} x(n) z_k^{-n} \\ &= \sum_{n=0}^{N-1} x(n) A^{-n} W^{nk}, \quad k = 0, \dots, M-1 \end{aligned}$$

- 1) 取样点沿螺旋线按角度间隔 ϕ_0 分布， $\phi_0 > 0$ 时，取样轨迹逆时针旋转； $\phi_0 < 0$ 时，取样轨迹顺时针旋转。
- 2) z_k 周线是一条螺旋线： W_0 表示螺旋线的伸展率； $W_0 > 1$ ，随着 k 的增加，向内盘旋，朝向原点； $W_0 < 1$ ，随着 k 的增加，向外盘旋；
- 3) 若 $W_0 = 1$ ，一段圆弧，若同时 $A_0 = 1$ ，则为单位圆一部分，此时 $CZT[x(n)] = DFT[x(n)]$ ($M=N$)。



3.5.1 CZT: 定义

例3.27

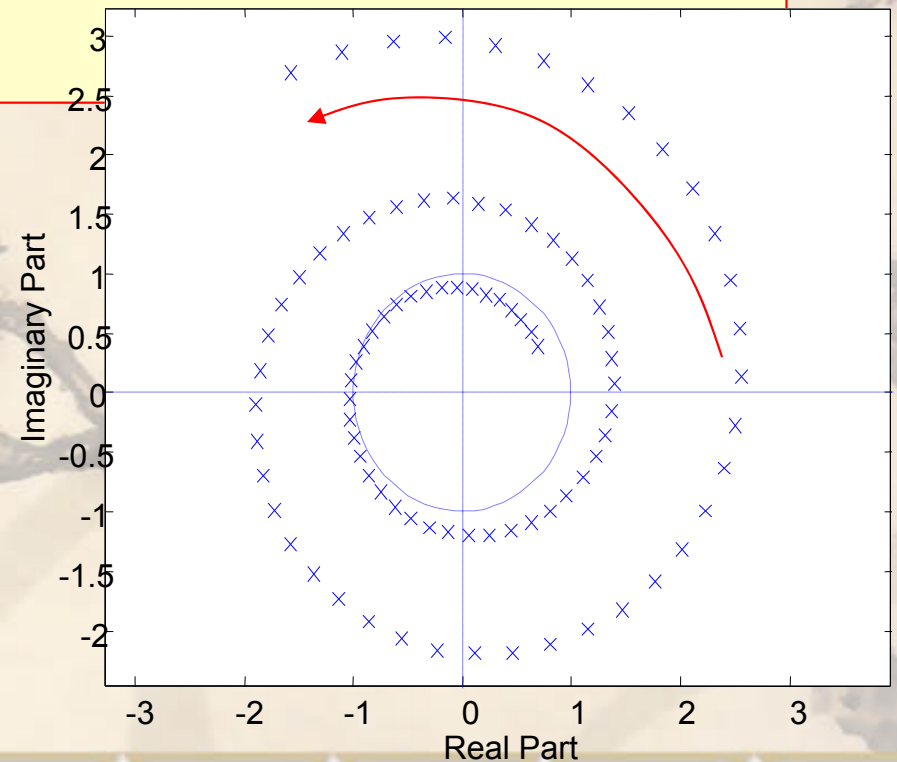
$A = 0.8 \exp(j\pi/6)$; %起始半径 0.8, 起始角 $\pi/6$

$W = 0.985 \exp(-j\pi \cdot 0.05)$; % $W_0 < 1$, 外旋; 取样间隔 0.05π

$M = 91$; %计算点数

$z1 = A \cdot (W.^{-(0:M-1)})$;

$Z_{\text{plane}}([], z1.)$



3.5.1 CZT: 定义

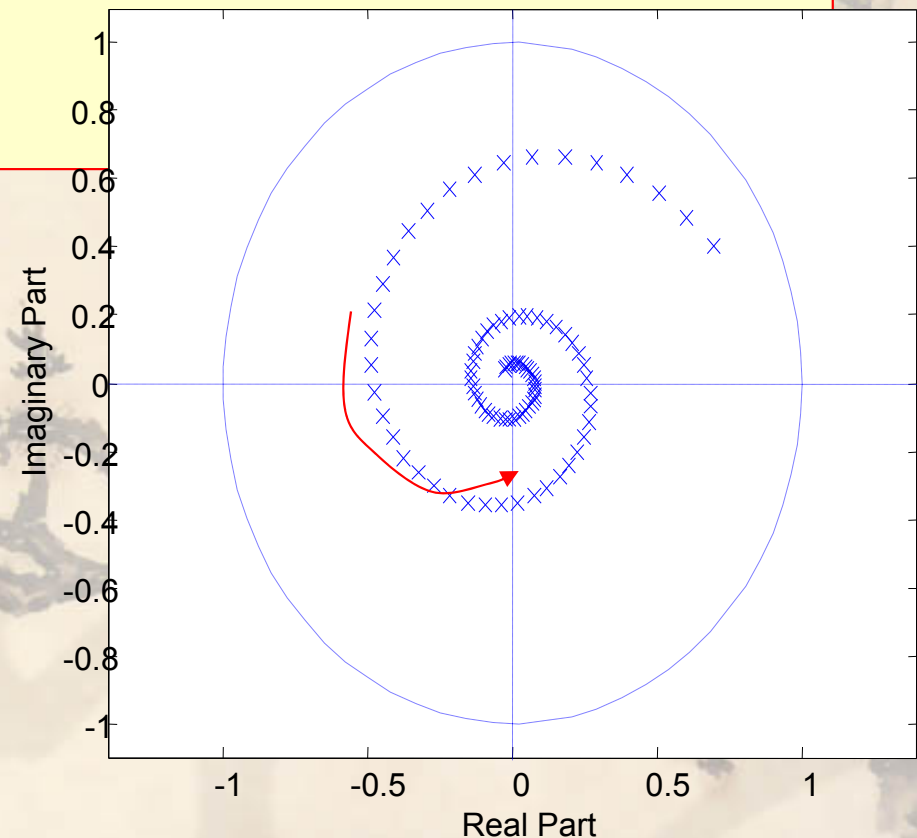
```
A = 0.8*exp(j*pi/6); %起始半径 0.8, 起始角 pi/6
```

```
W = 1.031*exp(-j*pi*0.05); %W0>1, 内旋; 取样间隔 0.05pi
```

```
M = 91; %计算点数
```

```
z2 = A*(W.^(-(0:M-1)));
```

```
Zplane([ ],z2.')
```



3.5.2 CZT：快速算法

■ **计算量：**直接计算 M 点的 CZT，需要 MN 次复数乘法， $M(N-1)$ 次复数加法，需要快速算法。

■ **CZT 表示为线性卷积形式：**

✧ 利用布鲁斯坦 (Bluestein) 提出的等式

把在 CZT[$x(n)$] 中的 W^{nk} 因子展开，得： $nk = \frac{1}{\gamma}[n^2 + k^2 - (n-k)^2]$

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

把上述运算转换为线性卷积形式，从而可以采用 FFT 算法，提高运算速度。

3.5.2 CZT: 快速算法

令

$$\begin{cases} g(n) = x(n) A^{-n} W^{\frac{n^2}{2}}, & 0 \leq n \leq N-1 \\ h(n) = W^{-\frac{n^2}{2}}, & -(N-1) \leq n \leq M-1 \end{cases}$$

则

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

式中

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

3.5.2 CZT: 快速算法

$g(n)$ 和 $h(n)$ 的取值范围:

$\because A = A_0 e^{j\theta_0}$, $W = W_0 e^{-j\phi_0}$ $\therefore A^{-n} W^{n^2/2}$ 是无穷长序列

$\because g(n) = x(n) A^{-n} W^{n^2/2}$, $x(n)$ 是 N 点序列

$\therefore g(n)$ 也是 N 点序列, 即 $n=0,1,\dots,N-1$

同样, $h(n) = W^{-n^2/2}$ 也应是无穷长序列, 且以 $n=0$ 为偶对称

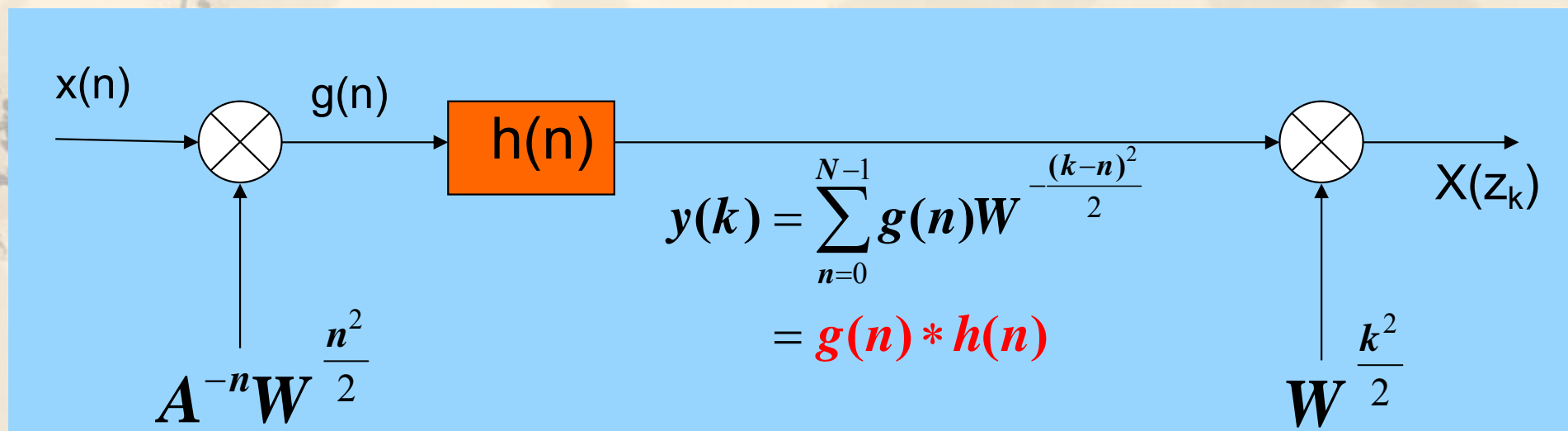
考虑 $n = 0, 1, \dots, N-1$, $k = 0, 1, \dots, M-1$, 上述线性卷积中的 $h(k-n)$

当 $n = 0$ 时, 此时 $h(n)$ 定义域最大取值为 $M-1$;

当 $k = 0$ 时, 此时 $h(n)$ 定义域最小取值范围为 $-(N-1)$ 。

因此, 考虑了 n, k 的取值范围后, $h(n)$ 定义域的取值范围为 $-(N-1) \sim M-1$ 。

3.5.2 CZT: 快速算法



序列: $h(n) = W^{-\frac{n^2}{2}}$

可以想像为频率随时间 (n) 线性增长的复指数序列, 在雷达中这类信号, 称为线性调频信号 (chirp Signal), 因此, 这里 z 变换称为线性调频 z 变换。

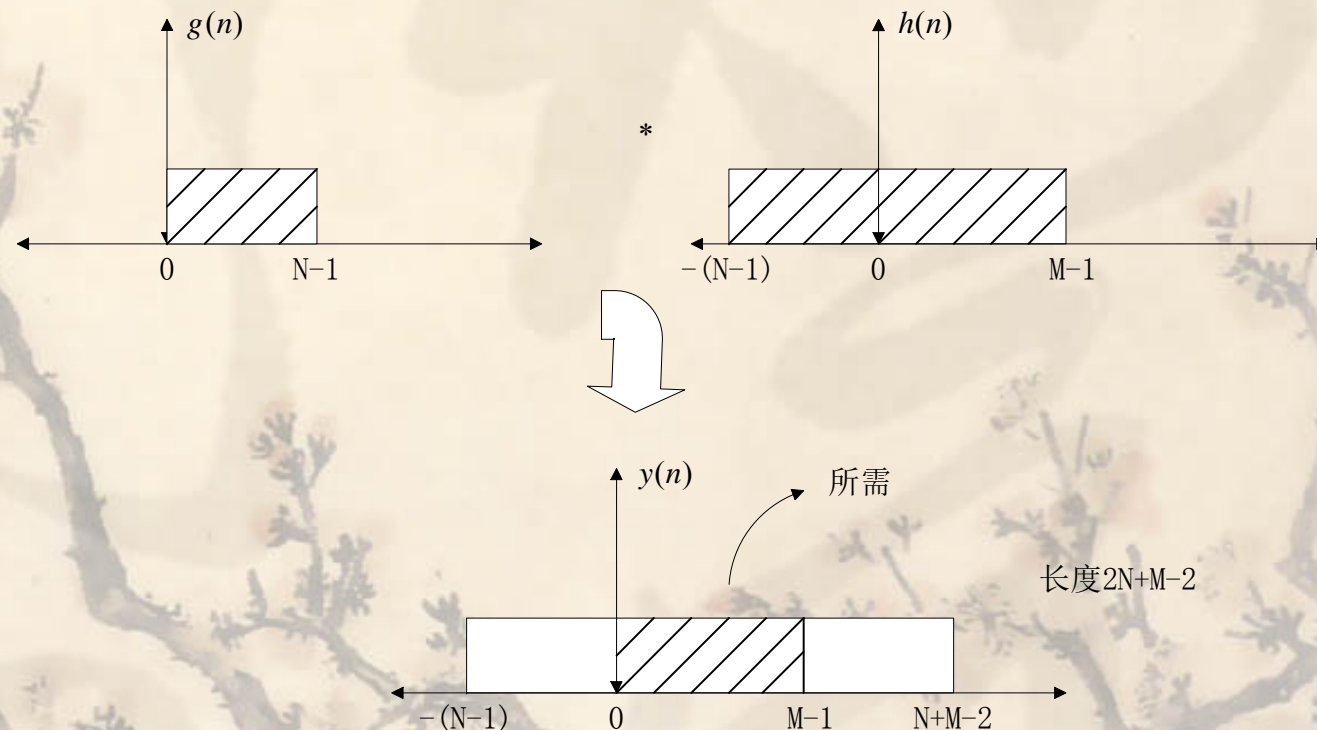
类比:

$$x(t) = e^{j\Omega t^2}$$

该信号的瞬时频率 $\Omega_i = 2\Omega t$ 正比于时间 t , 因此, 该信号的频率随时间线性增长。

3.5.2 CZT: 快速算法

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

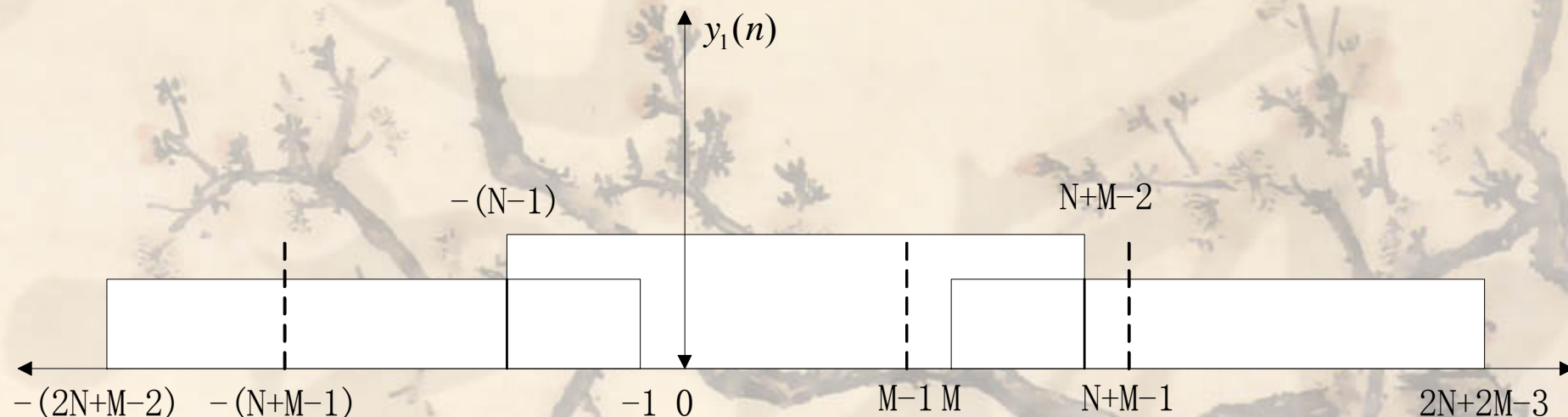


❖ 取值范围

- ❧ $n: 0 \sim N-1, k: 0 \sim M-1$
- ❧ 因果序列 $g(n)$ 的长度为 N , 而序列 $h(n)$ 的长度为 $N+M-1$, 并位于区间内 $-(N-1) \sim (M-1)$, 所以卷积后的序列 $y(n)$ 的长度为 $2N+M-2$, 且位于区间内 $-(N-1) \sim (N+M-2)$ 。
- ❧ 实际上只要得到区间 $0 \sim (M-1)$ 内的线性卷积结果就能够完成 CZT 的计算。

3.5.2 CZT: 快速算法

✧ 用循环卷积代替线性卷积且不产生混迭失真的条件是：循环卷积的点数（周期）应大于或等于 $2N+M-2$ ，但 CZT 只需要前 $0 \sim (M-1)$ 值，对以后的其它值是否混迭并不关心。因此，可将循环卷积的点数缩减到最小为 $N+M-1$ 。为了基 2 FFT 运算，取 $L \geq N+M-1$ ， $L=2^m$ 。



3.5.2 CZT: 快速算法

■ $g(n)$ 、 $h(n)$ 序列的构造

✧ 为了进行循环卷积，把 $h(n)$ 以周期 L 进行周期拓展，再取主值序列，从而得到圆周卷积的一个序列。

⊕ 从 $n=M$ 开始补 $L - (N+M-1)$ 个零或任意值，补到 $n=L-N$ 处。

⊕ 从 $n=L-N+1$ 到 $L-1$ ，取 $h(n)$ 的周期拓展序列 $h(L-n)$ 。

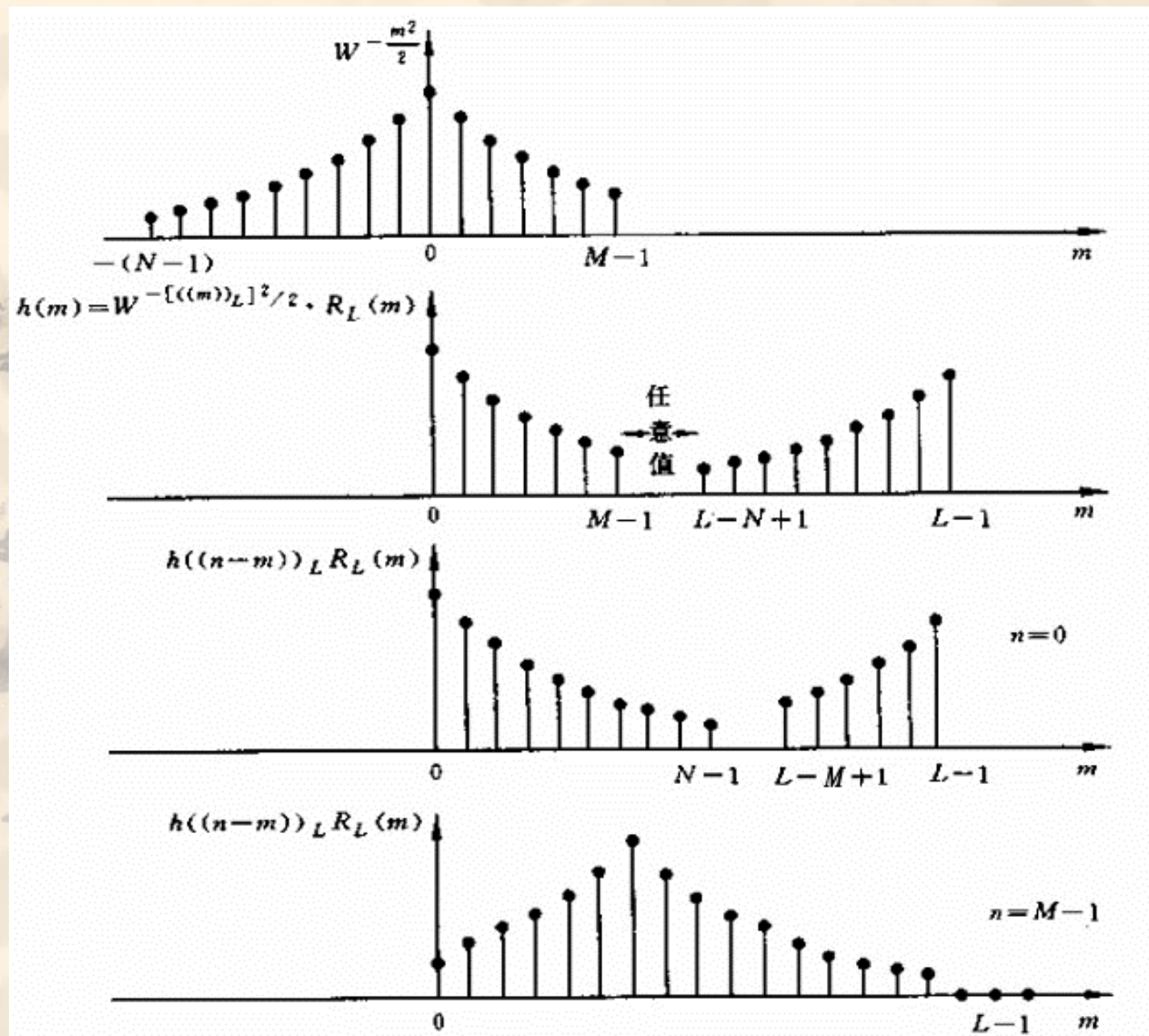
$$h(n) = \begin{cases} h(n) = W^{-\frac{n^2}{2}}, & 0 \leq n \leq M-1 \\ 0 \text{ 或任意值}, & M \leq n \leq L-N \\ h(L-N) = W^{-\frac{(L-n)^2}{2}}, & L-N+1 \leq n \leq L-1 \end{cases}$$

✧ 进行循环卷积的另一个序列 $g(n)$ 只需要补零到 L 即可。

$$y(n) = g(n) \otimes h(n) = \left[\sum_{m=0}^{N-1} \underbrace{g(m)}_{\text{有限长序列}} \underbrace{h((n-m))_N}_{\text{周期序列}} \right] R_N(n)$$

3.5.2 CZT: 快速算法

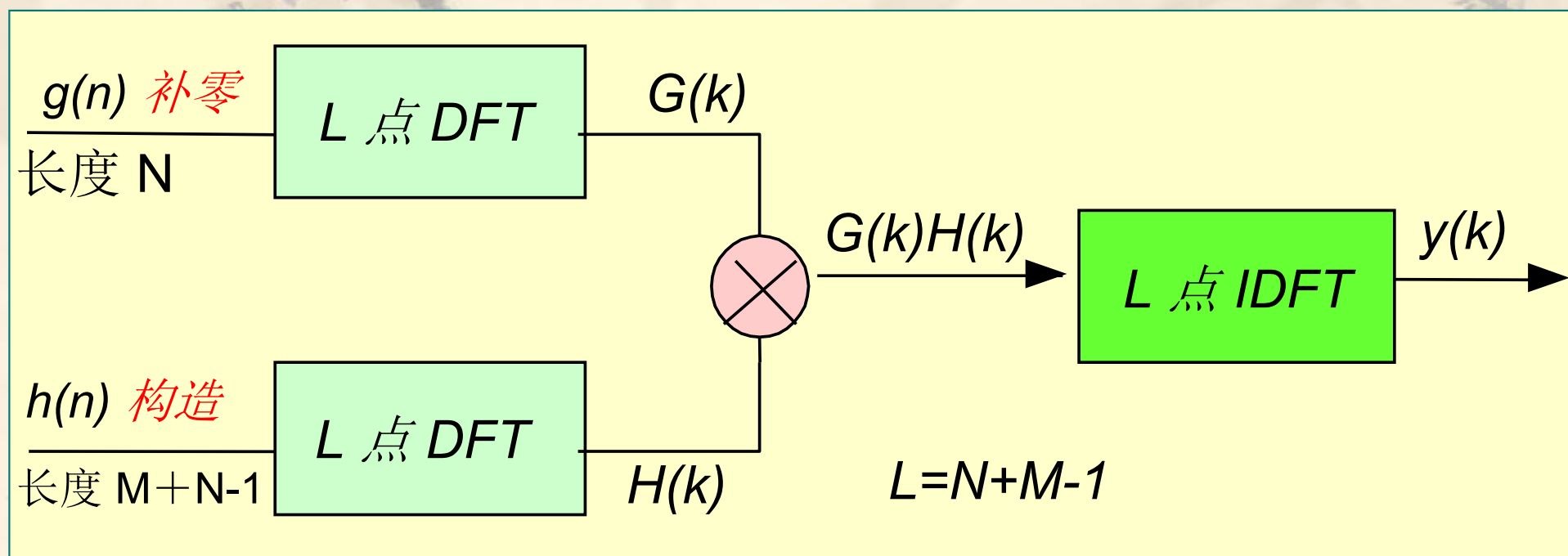
$h(n)$ 的构造:



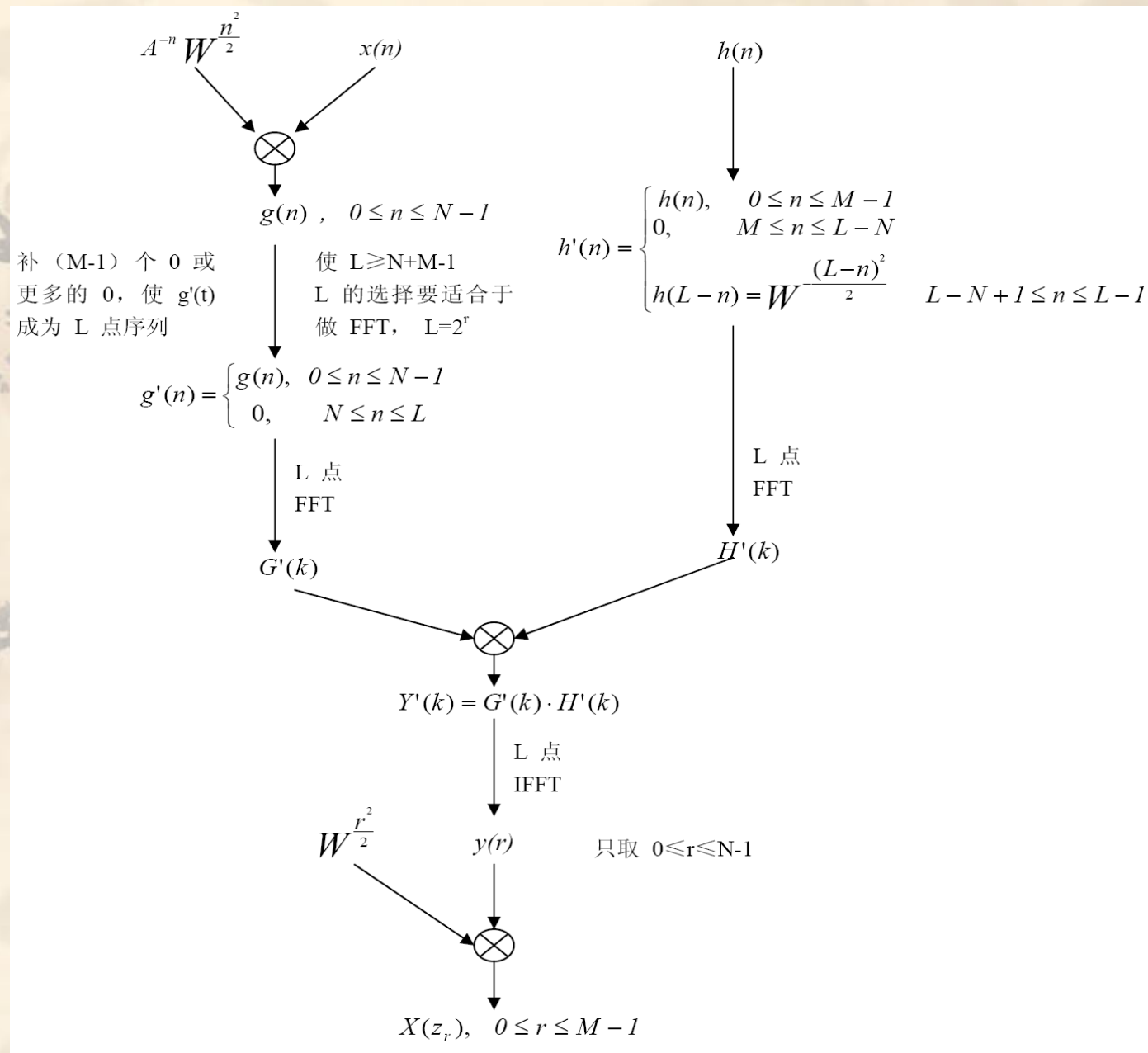
3.5.2 CZT: 快速算法

❖ CZT 快速算法

下面说明上述计算方案的具体实现，既然 $X(z_k)$ 可用线性卷积表示，我们就可以用 **DFT** 来计算线性卷积部分，只不过 **DFT** 换成 **FFT**，从而得到 **CZT** 的快速算法。



3.5.2 CZT: 快速算法



3.5.3 CZT: Matlab 实现

- ❖ CZT (Chirp z-transform) 变换的 Matlab 句法为:

$$y = \text{czt}(x, m, w, a)$$

$$y = \text{czt}(x)$$

- ∞ CZT 是 x 信号沿着 w 和 a 定义的螺旋线进行的 z 变换。 m 说明了变换的长度, w 是 z 平面上感兴趣的那部分螺旋线上取样点之间的比值, a 是螺旋线上的复数起始点。 z 平面上的螺旋线, 或“线性调频脉冲”定义为:

$$z = a * (w.^{-(0:m-1)})$$

- ✧ 如果 x 是矩阵, $\text{czt}(x, m, w, a)$ 是 x 的列变换。

- ∞ $y = \text{czt}(x)$ 使用下列缺省值:

$$m = \text{length}(x)$$

$$w = \exp(j * 2 * \pi / m)$$

$$a = 1$$

- ∞ 对于这些缺省值, czt 返回 x 信号在单位园上 m 份等间隔的 z 变换, 也就是 x 的离散傅立叶变换, 或者 $\text{fft}(x)$ 。空矩阵 $[]$ 说明了这些参数的缺省值。

3.5.3 CZT: Matlab 实现

例3.28 用 `czt` 放大一个滤波器频率响应的窄带部分（100~150Hz）。

❖ 首先设计一个滤波器：

```
% 30th order LPF filter, cut-off frequency Wn=125/500,use 'boxcar' window  
h = fir1(30,125/500,boxcar(31));
```

❖ 建立频率和 CZT 初始化参数：

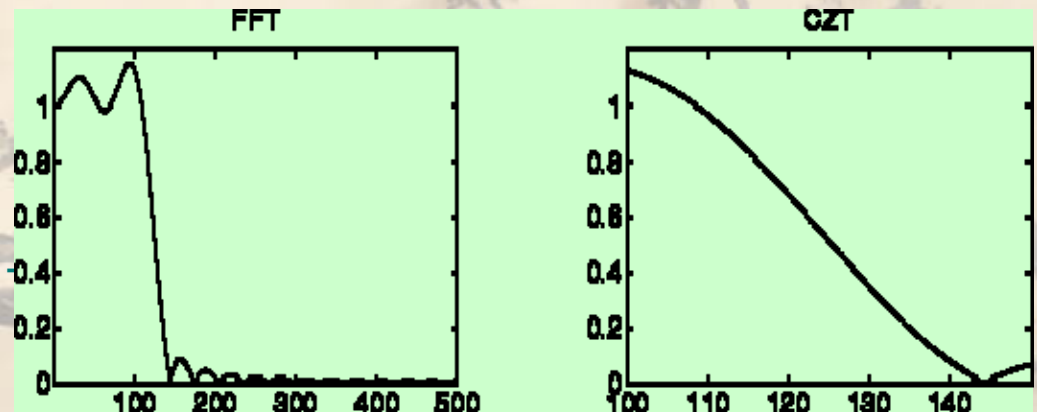
```
Fs = 1000; f1 = 100; f2 = 150; % in Hertz  
m = 1024; % 取样点数  
w = exp(-j*2*pi*(f2-f1)/(m*Fs)); % 取样点间隔  
a = exp(j*2*pi*f1/Fs); % 取样点起始位置
```

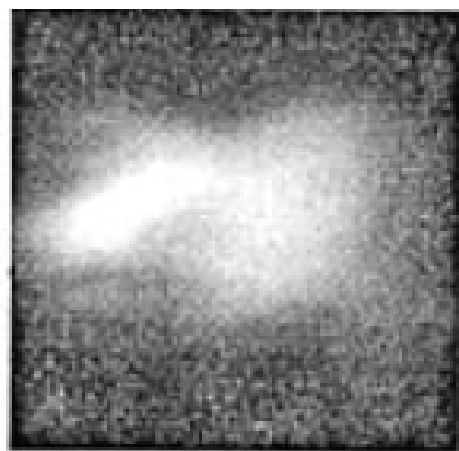
❖ 计算滤波器的 DFT 和 CZT 变换：

```
y = fft(h,1000);  
z = czt(h,m,w,a);
```

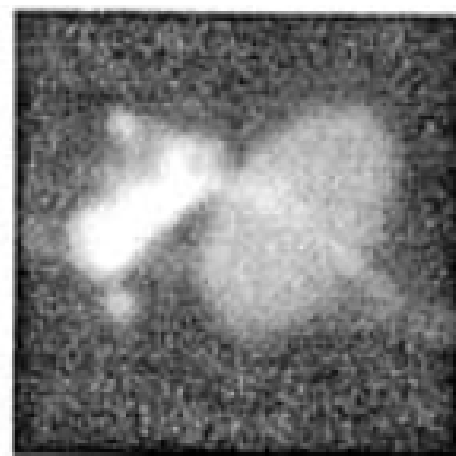
❖ 得到频率响应和比较结构：

```
fy = (0:length(y)-1)*1000/length(y);  
fz = ((0:length(z)-1)*(f2-f1)/length(z)) + f1;  
plot(fy(1:500),abs(y(1:500)));  
axis([1 500 0 1.2]);  
plot(fz,abs(z)); axis([f1 f2 0 1.2])
```





远程高空卫星照片
(校正前)



拟远程高空卫星照片
(校正后)

提纲:

3.1 问题的提出

3.2 DFS（离散傅里叶级数）

3.3 DFT（有限离散傅里叶变换）

3.4 FFT（快速离散傅里叶变换）

3.5 CZT及其快速算法

3.6 其它变换

3.7 本章小结

3.6.1 其它离散变换：离散余弦变换(DCT)

■ 1D-DCT 的基为：

$$c(k,n) = \begin{cases} \frac{1}{\sqrt{N}} & k=0, 0 \leq n \leq N-1 \\ \sqrt{\frac{2}{N}} \cos \frac{\pi(2n+1)k}{2N} & 1 \leq k \leq N-1, 0 \leq n \leq N-1 \end{cases}$$

特点：

- (1) 无虚数部分；
- (2) 正变换核与反变换核一样

则正变换可表示为：

$$X_c(k) = \sqrt{\frac{2}{N}} \beta(k) \sum_{n=0}^{N-1} x(n) \cos \left[\frac{\pi(2n+1)k}{2N} \right], \quad 0 \leq k \leq N-1$$

$$\text{where } \beta(0) = \sqrt{\frac{1}{2}}, \quad \beta(k) = 1, \quad 1 \leq k \leq N-1$$

反变换可表示为：

$$x(n) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \beta(k) X_c(k) \cos \left[\frac{\pi(2n+1)k}{2N} \right], \quad 0 \leq n \leq N-1$$

N 个余弦序列彼此正交，构成正交基

$$\sum_{n=0}^{N-1} \sqrt{\frac{2}{N}} \beta(i) \cos \left[\frac{\pi(2n+1)i}{2N} \right] \sqrt{\frac{2}{N}} \beta(j) \cos \left[\frac{\pi(2n+1)j}{2N} \right]$$

$$= \begin{cases} 1 & i = j \neq 0 \\ 1 & i = j = 0 \\ 0 & i \neq j \end{cases}$$

$$c_{ij} = \sqrt{\frac{2}{N}} \beta(i) \cos \left[\frac{\pi i(2j+1)}{2N} \right], j = 0, 1, 2, \dots, N-1$$

构造向量序列 $c_i, i = 0, 1, \dots, N-1 \rightarrow W_{\text{DCT}} \in R^{N \times N}$

$$W_{\text{DCT}} \times W_{\text{DCT}}^T = W_{\text{DCT}}^T \times W_{\text{DCT}} = I$$

$$W_{\text{DCT}} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{bmatrix}$$

$$\begin{cases} X_c = W_{\text{DCT}} x \\ x = W_{\text{DCT}}^{-1} X_c = W_{\text{DCT}}^T X_c \end{cases},$$

$$x = [x(0), x(1), \dots, x(N-1)]^T,$$

$$X_c = [X_c(0), X_c(1), \dots, X_c(N-1)]^T$$

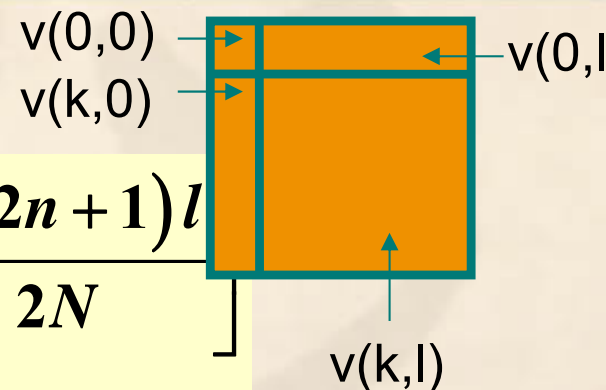
$$\|x\| = X_c^T X_c = \|X_c\|$$

酉变换

3.6.1 其它离散变换：离散余弦变换(DCT)

■ 2D-DCT的基为：

$$c_{k,l}(m,n) = \sqrt{\frac{2}{N}} \cos \left[\frac{\pi(2m+1)k}{2N} \right] \sqrt{\frac{2}{N}} \cos \left[\frac{\pi(2n+1)l}{2N} \right]$$
$$0 \leq k, l \leq N-1 \quad 0 \leq m, n \leq N-1$$



则正变换可表示为：

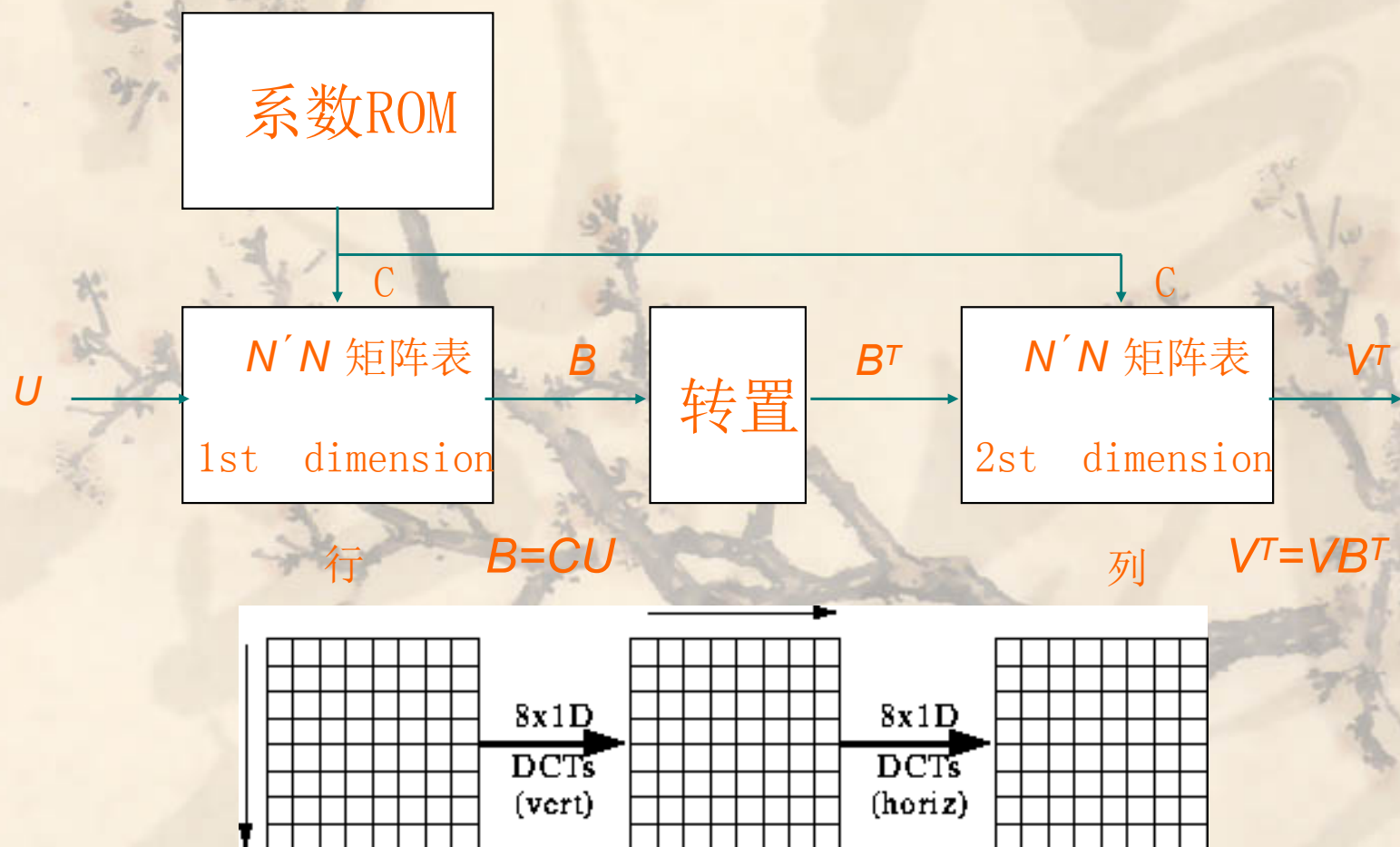
$$v(k,l) = d(k,l) \frac{2}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m,n) \cos \frac{\pi(2m+1)k}{2N} \cos \frac{\pi(2n+1)l}{2N}$$
$$d(0,0) = \frac{1}{2}, \quad d(k,l) = 1 \text{ 对其它}$$

反变换可表示为：

$$u(m,n) = d(k,l) \frac{2}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} v(k,l) \cos \frac{\pi(2m+1)k}{2N} \cos \frac{\pi(2n+1)l}{2N}$$

3.6.1 其它离散变换：DCT 快速算法

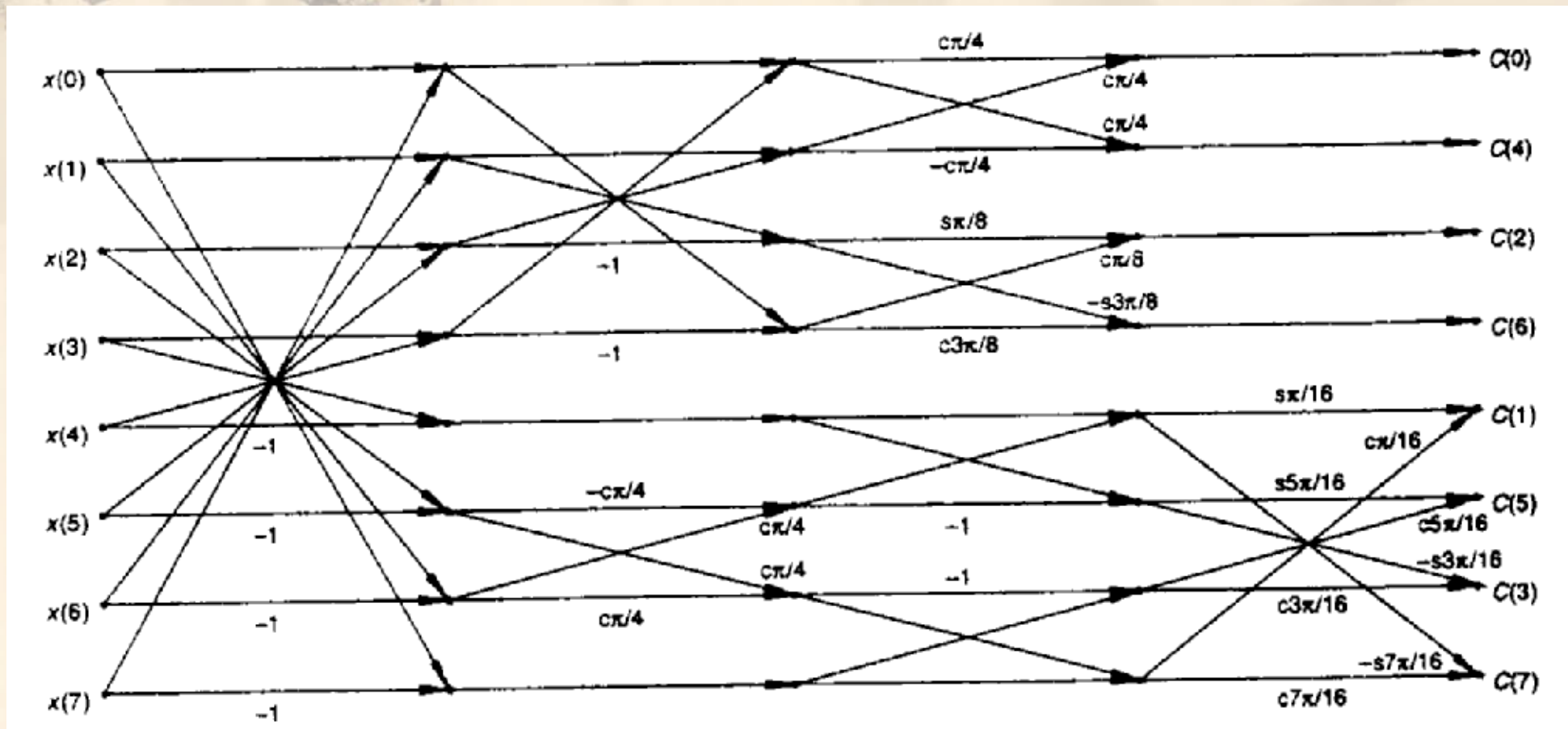
- 直接进行 DCT 计算：每个系数需要 64 次乘法、63 次加法， 8×8 DCT 将需要 1024 次乘法，896 次加法。
- DCT 有快速算法：2D-DCT 的行列运算，每一次运算变为 1D-DCT 运算，1D-DCT 采用 fast DCT。一般的 1D-DCT 的快速算法使运算量降低为 $N \log_2 N$ 数量级的实数乘法。



3.6.1 其它离散变换：DCT 快速算法

■ DCT 快速算法是中国人陈文雄 (1977) 提出的

- ✧ Chen W.H., et., "A fast computational algorithm for the discrete cosine transform", IEEE transactions on Communications, COM-25, 1004-1009.
- ✧ $N=8$ 时的算法如图
- ✧ c_x 对应于 $\cos x$, s_x 对应于 $\sin x$ 。



3.6.1 其它离散变换：DCT 快速算法

■ DCT矩阵因式分解为稀疏矩阵（Chen et.al.,1977）（对应上述流图）

$$C_8 = P_8 A_4 A_3 A_2 A_1$$

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\cos\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & \cos\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

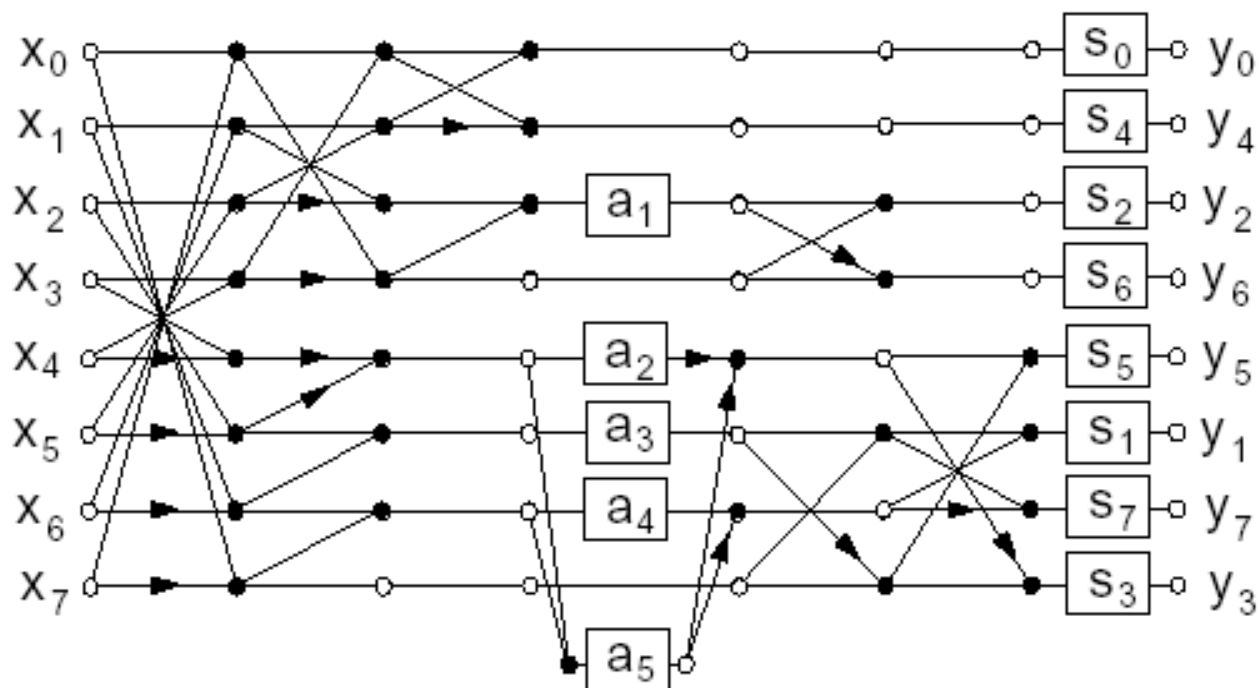
$$A_3 = \begin{bmatrix} \cos\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 & 1 & 0 & 0 & 0 & 0 \\ \cos\frac{\pi}{4} & -\cos\frac{\pi}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sin\frac{\pi}{8} & \cos\frac{\pi}{8} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{4} \\ 0 & 0 & 0 & 0 & 0 & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ 0 & 0 & 0 & 0 & -\sin\frac{7\pi}{16} & 0 & 0 & \cos\frac{\pi}{4} \end{bmatrix}$$

将码位倒置序列转
变成顺序序列的输
出向量

3.6.1 其它离散变换：DCT 快速算法

- Arai, Agui and Nakajima 算法的快速 8-DCT 信号流图
(此算法需要 13 次乘法, 29 次加法)



scaling
only 5 + 8
multiplications
(direct matrix
multiplication:
64 multiplications)

Multiplication:

$$u \circ \boxed{m} \circ m \cdot u$$

Addition:

$$u \circ v \circ u+v$$

$$u \circ v \circ u-v$$

$$a_1 = C_4$$

$$a_2 = C_2 - C_6$$

$$a_3 = C_4$$

$$a_4 = C_6 + C_2$$

$$a_5 = C_6$$

$$s_0 = \frac{1}{2\sqrt{2}}$$

$$s_k = \frac{1}{4 C_k} ; k = 1 \dots 7$$

$$C_k = \cos(k\pi/16)$$

3.6.1 其它离散变换：DCT 快速算法

- **AAN 快速 DCT 算法** (Arai, Agui and Nakajima, 1988)：矩阵分解为稀疏矩阵

$$v = C u = S P M_1 M_2 M_3 M_4 M_5 M_6 u$$

$$\begin{aligned} \underline{S} &= \begin{bmatrix} s_0 & & & & & & & \\ & s_1 & & & & & & \\ & & s_2 & & & & & \\ & & & s_3 & & & & \\ & & & & s_4 & & & \\ & & & & & s_5 & & \\ & & & & & & s_6 & \\ & & & & & & & s_7 \end{bmatrix} & \underline{P} &= \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \end{bmatrix} & \underline{M}_1 &= \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & 0 & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix} & \underline{M}_2 &= \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & 0 & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix} \\ \underline{M}_3 &= \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & 0 & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix} & \underline{M}_4 &= \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & 0 & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix} & \underline{M}_5 &= \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \end{bmatrix} & \underline{M}_6 &= \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & 0 & & & 1 & & & \\ & & 1 & & & 1 & & \\ & & & 1 & & & 1 & \\ & & & & 1 & & & 1 \end{bmatrix} \end{aligned}$$

DCT 和 DFT 比较

- DCT 是实的正交变换，但不对称。

$$C = C^* \Rightarrow C^{-1} = C^T$$

- DCT 不是 DFT 的实部

$$\text{Re}[F(U)] \neq \text{DCT}(U)$$

- 因为 $\cos(x) = (e^{jx} + e^{-jx})/2$ ，因此，对于 DCT 可用 $2N$ 点 DFT 计算。

$$\begin{aligned} \cos\left[\frac{(2n+1)k\pi}{2N}\right] &= \frac{1}{2} \left[e^{j\left(\frac{2\pi}{2N}kn + \frac{\pi}{2N}k\right)} + e^{-j\left(\frac{2\pi}{2N}kn + \frac{\pi}{2N}k\right)} \right] \\ &= \frac{1}{2} \left[W_{2N}^{\frac{k}{2}} W_{2N}^{kn} + W_{2N}^{-\frac{k}{2}} W_{2N}^{-kn} \right] \end{aligned}$$

DCT 和 DFT 比较

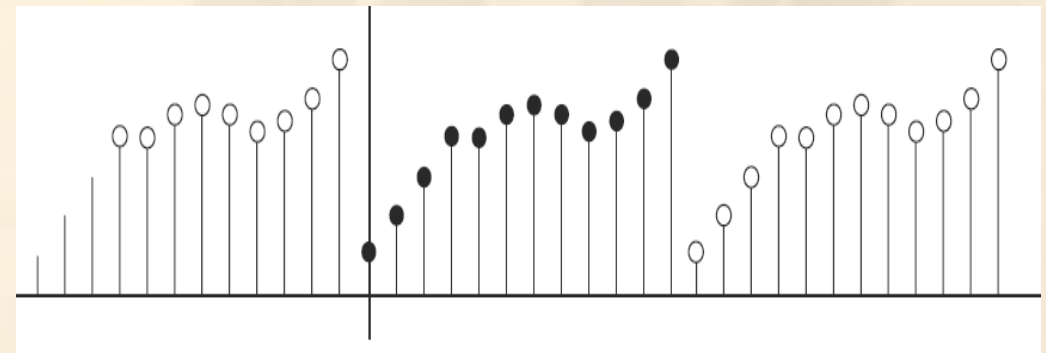
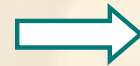
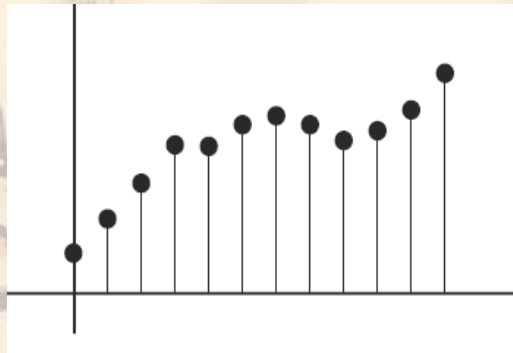
$$\begin{aligned} C(k) &= \alpha(k) \sum_{n=0}^{N-1} x(n) \cos \left[\frac{\left(n + \frac{1}{2} \right) k \pi}{N} \right] \\ &= \frac{1}{2} \left(\alpha(k) W_{2N}^{\frac{k}{2}} \sum_{n=0}^{N-1} x(n) W_{2N}^{kn} + \alpha(k) W_{2N}^{-\frac{k}{2}} \sum_{n=0}^{N-1} x(n) W_{2N}^{-kn} \right) \\ &= \frac{1}{2} \alpha(k) \left(W_{2N}^{\frac{k}{2}} DFT_{2N} \{ \hat{x}(n) \}(k) + W_{2N}^{-\frac{k}{2}} DFT_{2N} \{ \hat{x}(n) \}(-k) \right) \end{aligned}$$

对 $0 \leq k \leq N-1$ ，其中 \hat{x} 等价于将 $x(n)$ 补零成 $2N$ 点序列。上式第二项等价于第一项作下标为 $(-k)$ 的运算。因此，实际上只需要计算 $2N$ 点 DFT。

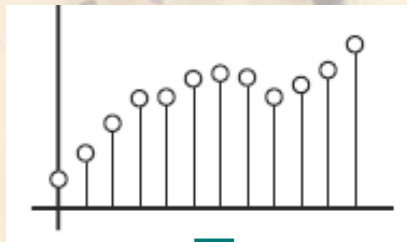
DCT 和 DFT 比较

DFT在边界不连续

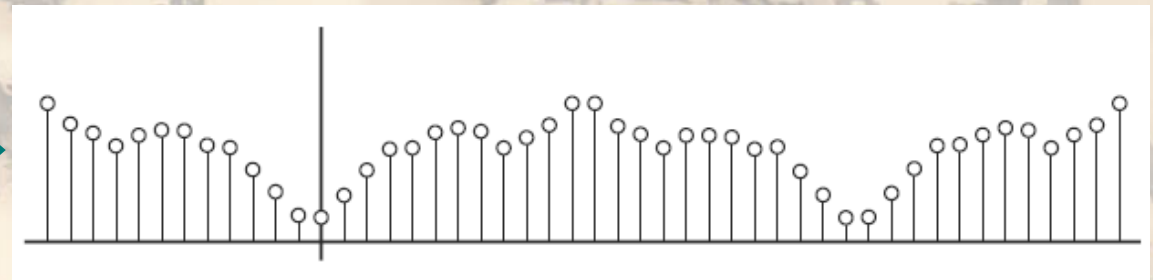
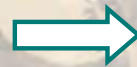
DFT:



DCT:



DCT更连续

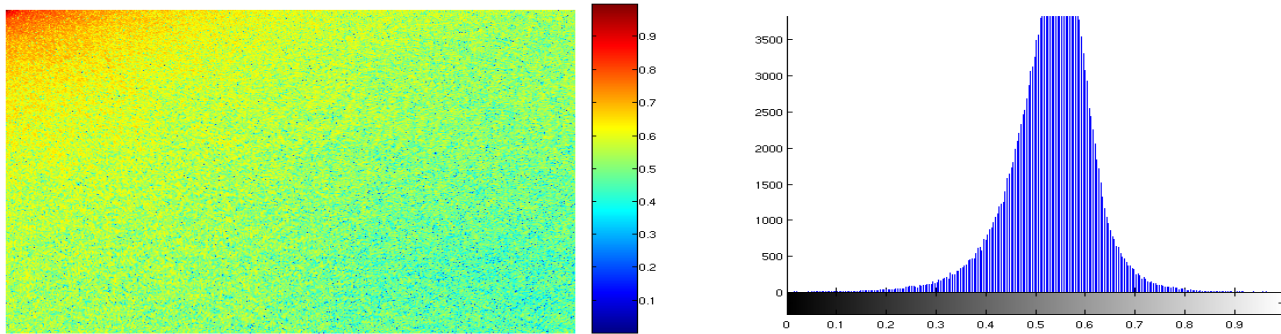


函数的不连续影响Fourier级数的收敛，从而需要更多基函数，影响压缩

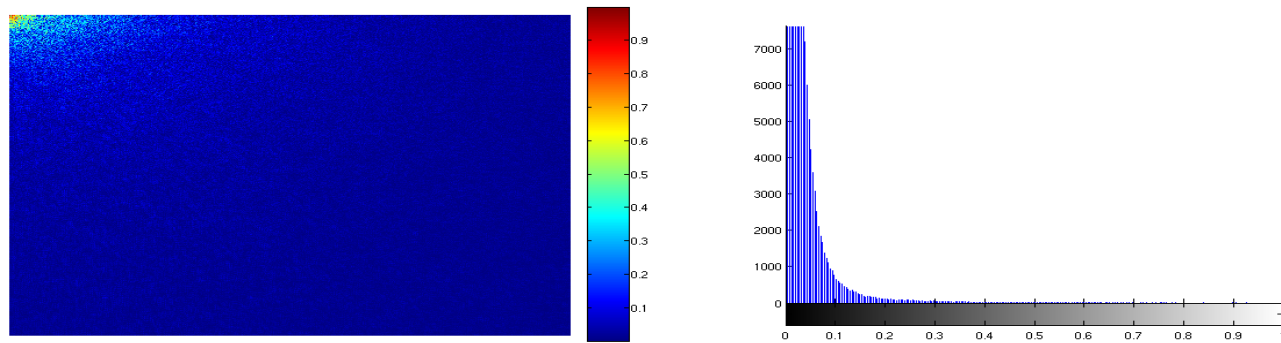
DCT 和 DFT 比较



DFT



DCT



DCT 变换后的能量更集中
更适合压缩

3.6.2 其它离散变换：离散正弦变换(DST)

■ 1D-DST 的基为：

$$c(k,n) = \begin{cases} \frac{1}{\sqrt{N}} & k=0, 0 \leq n \leq N-1 \\ \sqrt{\frac{2}{N}} \sin \frac{\pi(2n+1)(k+1)}{2N} & 1 \leq k \leq N-1, 0 \leq n \leq N-1 \end{cases}$$

特点：

- (1) 无虚数部分；
- (2) 正变换核与反变换核一样

则正变换可表示为：

$$X_s(k) = \sqrt{\frac{2}{N}} \alpha(k) \sum_{n=0}^{N-1} x(n) \sin \left[\frac{\pi(2n+1)(k+1)}{2N} \right], \quad 0 \leq k \leq N-1$$

$$\blacklozenge\blacklozenge\blacklozenge \alpha(N-1) = \sqrt{\frac{1}{2}}, \quad \alpha(k) = 1 \quad \blacklozenge\blacklozenge 0 \leq k \leq N-2$$

反变换可表示为：

$$x(n) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \alpha(k) X_s(k) \sin \left[\frac{\pi(2n+1)(k+1)}{2N} \right], \quad 0 \leq n \leq N-1$$

N 个正弦序列彼此正交，构成正交基

$$\sum_{n=0}^{N-1} \sqrt{\frac{2}{N}} \alpha(i) \sin \left[\frac{\pi(2n+1)(i+1)}{2N} \right] \sqrt{\frac{2}{N}} \alpha(j) \sin \left[\frac{\pi(2n+1)(j+1)}{2N} \right]$$

$$= \begin{cases} 1 & i = j \neq 0 \\ 1 & i = j = 0 \\ 0 & i \neq j \end{cases}$$

$$s_{ij} = \sqrt{\frac{2}{N}} \alpha(i) \cos \left[\frac{\pi(i+1)(2j+1)}{2N} \right], j = 0, 1, 2, \dots, N-1$$

构造向量序列 $s_i, i = 0, 1, \dots, N-1 \rightarrow W_{\text{DST}} \in R^{N \times N}$

$$W_{\text{DST}} \times W_{\text{DST}}^T = W_{\text{DST}}^T \times W_{\text{DST}} = I$$

$$W_{\text{DST}} = \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{N-1} \end{bmatrix}$$

$$\begin{cases} X_s = W_{\text{DST}} x \\ x = W_{\text{DST}}^{-1} X_s = W_{\text{DST}}^T X_s \end{cases},$$

$$x = [x(0), x(1), \dots, x(N-1)]^T,$$

$$X_s = [X_s(0), X_s(1), \dots, X_s(N-1)]^T$$

$$\|x\| = X_s^T X_s = \|X_s\|$$

酉变换

3.6.3 DFT局限性: Heisenberg测不准原理

❖ 不定原理 (Heisenberg 测不准原理, Heisenberg-Gabor 不定原理) :

给定信号 $x(t)$, 若 $\lim_{x \rightarrow \infty} \sqrt{t} x(t) = 0$, 则:

$$\Delta_t \Delta_\Omega \geq \frac{1}{2}$$

当且仅当 $x(t)$ 为高斯信号, 即 $x(t) = A e^{-\alpha t^2}$ 时等号成立, 式中:

$$\Delta_t^2 = \frac{1}{E} \int_{-\infty}^{\infty} (t - t_0)^2 |x(t)|^2 dt = \frac{1}{E} \int_{-\infty}^{\infty} t^2 |x(t)|^2 dt - t_0^2$$

$$\Delta_\Omega^2 = \frac{1}{2\pi E} \int_{-\infty}^{\infty} (\Omega - \Omega_0)^2 |X(\Omega)|^2 d\Omega = \frac{1}{2\pi E} \int_{-\infty}^{\infty} \Omega^2 |X(\Omega)|^2 d\Omega - \Omega_0^2$$

显然, 这是方差的标准定义。通常定义 $2\Delta_t$ 、 $2\Delta_\Omega$ 分别是信号的时宽和带宽。定义 $\Delta_t \Delta_\Omega$ 为信号的时宽—带宽积。

例1 $g(t) = (\alpha/\pi)^{1/4} \exp(-\frac{\alpha}{2}t^2)$

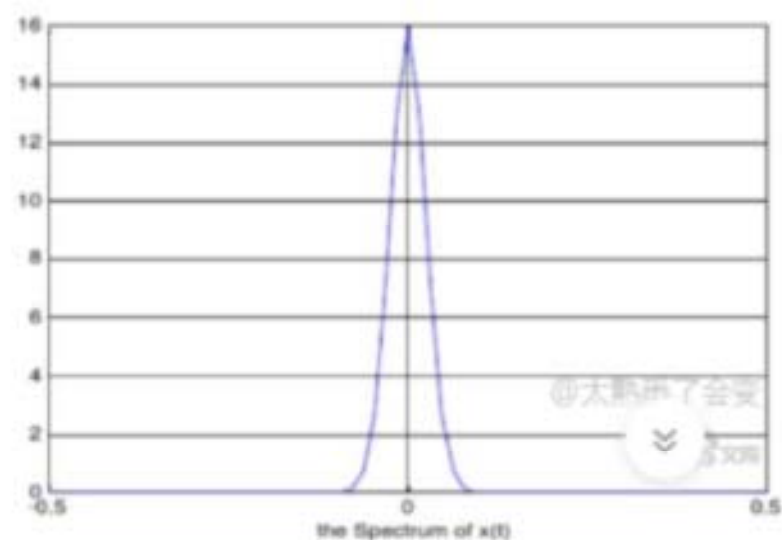
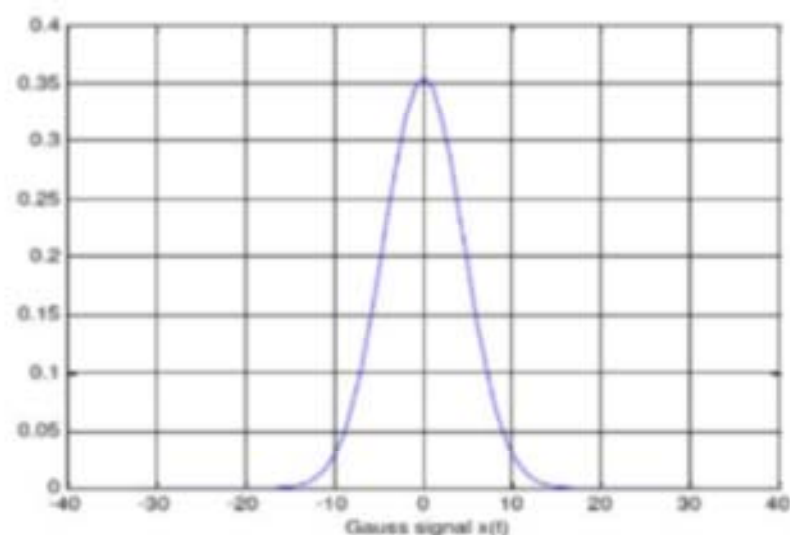
定义及推导

该信号的能量 $E=1$,称之为归一化高斯信号

$$\Delta_{\Omega}^2 = \alpha^2 \sqrt{\alpha/\pi} \int t^2 \exp(-\alpha t^2) dt = \alpha/2$$

$$\Delta_t^2 = 1/2\alpha$$

$$\Delta_t \Delta_{\Omega} = 1/2 \text{ 及 } TB = 2$$



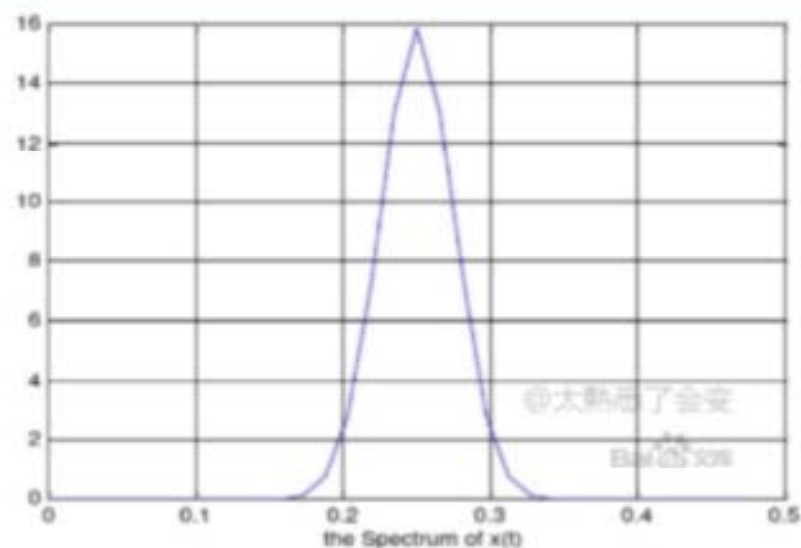
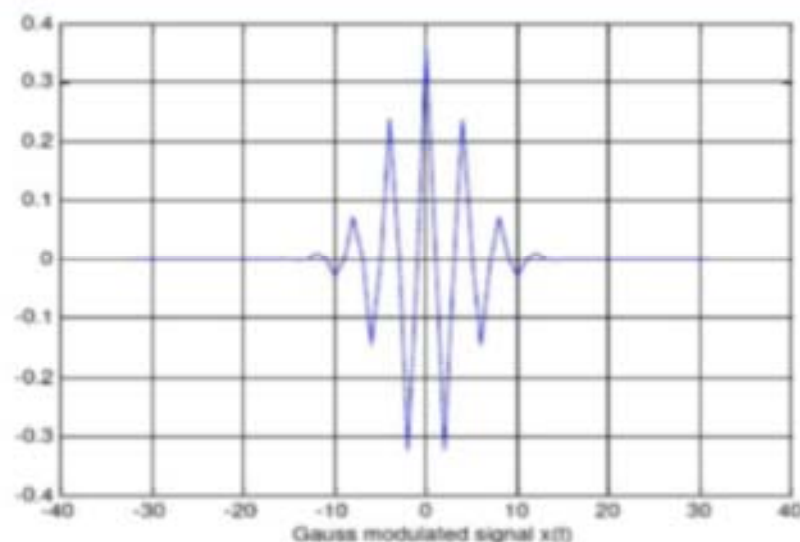
例2 $x(t) = g(t)e^{j\Omega_m t}$

定义及推导

$$\Omega_0 = \frac{1}{E} \int \Omega_m |x(t)|^2 dt = \Omega_m$$

$$\Delta_\Omega^2 = \alpha^2 \sqrt{\frac{\alpha}{\pi}} \int t^2 \exp(-\alpha t^2) dt = \frac{\alpha}{2}$$

$$\Delta_t^2 = 1/2\alpha$$



3.6.3 DFT局限性：Heisenberg测不准原理

- ❖ 测不准原理是信号处理中的一个重要的基本定理，不可能违背；
- ❖ 测不准原理给出了信号时宽-带宽之间的制约关系：对于给定的信号，其时带与带宽的乘积为一个常数。
 - 当信号的时宽减少时，其带宽将相应增加，当时宽减到无穷小时，带宽将变成无穷大，如时域的 δ 函数；
 - 反之亦然，如频域 Ω 处的 δ 函数，时域为 $e^{j\Omega t} = \cos\Omega t + j\sin\Omega t$ ，其在时域的持续时间是 $-\infty \sim +\infty$ 。
 - 这就是说，信号的时宽与带宽不可能同时趋于无限小，这一基本关系即是我们将要讨论的时间分辨率和频率分辨率的制约关系。
 - 在这一基本关系的制约下，人们在竭力探索既能得到好的时间分辨率（或窄的时宽）又能得到好的频率分辨率（或窄的带宽）的信号分析方法。
 - 若信号 $x(t)$ 的持续时间是有限的，我们称其为是“紧支撑”（**Compact Support**）的，其时间的持续区间（如 $t=t_1 \sim t_2$ ），称为“支撑范围”；对频率信号，也使用类似的称呼。

3.6.3 DFT 局限性：非平稳信号

■ 非平稳的定义：

✧ 两类说法，出发点不同，无大碍。

✧ 非平稳随机信号

⊕ 在平稳和非平稳随机信号定义中，前者是指随机信号的一阶和二阶统计特性（均值、方差）不随时间变换，自相关函数与观察的时间起点无关。

⊕ 若信号是平稳的，则满足维纳-辛钦关系，即功率谱密度与自相关函数互为傅立叶变换：

$$P(\Omega) \xleftrightarrow{FT} R(\tau)$$

若信号是非平稳的，则不满足上述关系。

✧ 非平稳信号

⊕ 频率随时间变化的信号，称为时变信号，或非平稳信号

⊕ 频率不随时间变化的时不变信号，称为平稳信号

⊕ 注意与随机信号中平稳/非平稳定义的区别，说一个信号是平稳信号，要指明是频率不随时间变化，还是平稳随机信号。

3.6.3 DFT 局限性：克服方法

❖ **DFT 的局限性成为寻找新的信号分析和处理的源动力。**

- 1932年，Wigner 提出了时间-频率联合分布的概念，并将其用于量子物理；
- 1946年，Gabor 提出了短时傅立叶变换和 Gaber 变换概念，从而开始了非平稳信号时频联合分析的研究；
- 20世纪80年代，提出了滤波器组理论，为信号的子带分解提供了有力工具；
- 20世纪80年代后期，发展起来了小波变换，它不仅扩展了信号时频联合分析的概念，而且在信号的分辨率方面具有对信号特点的适应性。
- 为了更一般地讨论信号的分解问题，人们提出了框架（Frame）理论
-

本章小结

- 理解傅里叶变换的几种形式
- 掌握 DFS及性质，掌握周期卷积过程
- 掌握 DFT及性质，掌握循环卷积、线性卷积及两者之间的关系
- 掌握线性卷积的 DFT 算法及分段卷积方法
- 掌握按时间抽选的基-2 FFT算法的算法原理、运算流图和算法特点及IFFT 算法
- 掌握按频率抽选的基-2 FFT算法的算法原理、运算流图和算法特点及IFFT 算法
- 理解频谱分析过程
- 了解 CZT 算法