第3章 信号的变换

信号与系统的分析方法中，除了时域分析方法外，还有变换域分析的方法。连续时间信号与系统的变换域分析方法主要是傅里叶变换和拉普拉斯变换。离散时间信号的 Z 变换是分析线性时不变离散时间系统问题的重要工具，在数字信号处理、计算机控制系统等领域有着广泛的应用。

学习目标：

(1) 了解、熟悉 Z 变换的概念与性质；
(2) 理解 Z 反变换的相关内容；
(3) 掌握离散系统中的 Z 域描述方法；
(4) 了解、熟悉傅里叶级数与变换；
(5) 理解离散傅里叶变换及其性质；
(6) 实现频率域采样和快速傅里叶变换；
(7) 熟悉实现离散余弦变换、Chirp Z 变换和 Gabor 函数。

3.1 Z 变换概述

连续系统一般使用微分方程、拉普拉斯变换的传递函数和频率特性等概念进行研究。一个连续信号 $f(t)$的拉普拉斯变换 $F(s)$是复变量 s 的有理分式函数，而微分方程通过拉普拉斯变换后也可以转换为 s 的代数方程，从而可以大大简化微分方程的求解，从传递函数可以很容易地得到系统的频率特征。

因此，拉普拉斯变换作为基本工具将连续系统研究中的各种方法联系在一起。计算机控制系统中的采样信号也可以进行拉普拉斯变换，从中找到简化运算的方法，引入了 Z 变换。

3.1.1 Z 变换的定义

序列 $x(n)$的 Z 变换(简称 ZT)定义为

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

上式称为双边 Z 变换。

如果 $x(n)$的非零值区间为$(-\infty,0]$或者$[0,+\infty)$,则上式可变为

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

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

此时,称为序列 $x(n)$的单边 Z 变换。

序列的 ZT 存在的条件为

$$|X(z)|=\left|\sum_{n=-\infty}^{+\infty}x(n)z^{-n}\right|\leqslant\sum_{n=-\infty}^{+\infty}|x(n)z^{-n}|=\sum_{n=-\infty}^{+\infty}|x(n)||z^{-n}|<+\infty$$

满足上式的 z 的取值范围称为 Z 变换的收敛域(Region of Convergence,ROC),它通常为 z 平面上的一个环状域,即

$$R_{x^-}<|z|<R_{x^+}$$

3.1.2 Z 变换的收敛域

序列 Z 变换的收敛域与序列的形态有关。反之,同一个 Z 变换的表达式,不同的收敛域,确定了不同序列形态。下面根据序列形态不同,分别讨论其收敛域。

对于任意给定的序列 $x(n)$,能使 $X(z)=\sum_{n=-\infty}^{\infty}x(n)z^{-n}$ 收敛的所有 z 值集合为收敛域。即满足

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

不同的 $x(n)$的 Z 变换,由于收敛域不同,可能对应于相同的 Z 变换,故在确定 Z 变换时,必须指明收敛域。

1. *有限长序列*

有限序列的描述函数是

$$x(n)=\begin{cases}x(n) & n_1\leqslant n\leqslant n_2\\ 0 & \text{其他}\end{cases}$$

其 Z 变换为

$$X(z)=\sum_{n=n_1}^{n_2}x(n)z^{-n}$$

因此 Z 变换式是有限项之和,故只要级数的每一项有界,则级数就收敛。收敛域为

$$0<|z|<\infty$$

2. *右边序列*

右边序列的描述函数是

$$x(n)=\begin{cases}x(n) & n\geqslant n_1\\ 0 & \text{其他}\end{cases}$$

其 Z 变换为

$$X(z)=\sum_{n=n_1}^{\infty}x(n)z^{-n}$$

因此 Z 变换样式是无限项之和，当 $n_1\geqslant 0$ 时，由根值判别法有

$$\lim_{n\to\infty}\sqrt[n]{|x(n)z^{-n}|}<1$$

所以此时收敛域为

$$|z|>\lim_{n\to\infty}\sqrt[n]{|x(n)|}=R_1$$

当 $n_1<0$ 时，此时级数全收敛，所以右边序列的收敛域为 $R_1<|z|<\infty$。

3. 左边序列

左边序列的描述函数为

$$x(n)=\begin{cases}x(n) & n\leqslant n_2\\ 0 & \text{其他}\end{cases}$$

其 Z 变换为

$$X(z)=\sum_{n=-\infty}^{n_2}x(n)z^{-n}=\sum_{n=-n_2}^{\infty}x(-n)z^{n}$$

当 $n_2<0$ 时，由根值判别法有

$$\lim_{n\to\infty}\sqrt[n]{|x(-n)z^{n}|}<1$$

由此求得的收敛域为

$$|z|<\lim_{n\to\infty}\sqrt[n]{|x(-n)|}=R_2$$

当 $n_2>0$ 时，此时相当于增加了一个 $n_2>0$ 的有限长序列，还应除去原点，左边序列的收敛域为

$$0<|z|<R_2$$

4. 双边序列

双边序列的描述函数为

$$x(n)=x(n)[u(-n-1)+u(n)]$$

其 Z 变换为

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

因为 $\sum_{n=0}^{\infty}x(n)z^{-n}$ 的收敛域为 $|z|>R_1$，$\sum_{n=-\infty}^{-1}x(n)z^{-n}$ 的收敛域为 $|z|<R_2$，所以双边序列的收敛域为

$$R_1<|z|<R_2$$

3.2 Z 变换的性质

3.2.1 线性性质

假设

$$Z[x_1(k)] = X_1(z) \quad (|z| > R_{x1})$$
$$Z[x_2(k)] = X_2(z) \quad (|z| > R_{x2})$$

则有

$$Z[ax_1(k) + bx_2(k)] = aX_1(z) + bX_2(z)$$

其中，a、b 为任意常数。

3.2.2 时域的移位

假设 $Z[f(t)]=F(z)$，那么有

$$Z[f(t+nT)] = z^n\left[F(z) - \sum_{k=0}^{n-1} f(kT)z^{-k}\right]$$

假设 $Z[f(t)]=F(z)$，那么有

$$Z[f(t-nT)] = z^{-n}F(z)$$

3.2.3 时域扩展性

若函数 $f(t)$有 Z 变换 $F(z)$，则

$$Z[e^{\mp at}f(t)] = F(ze^{\pm aT})$$

根据 Z 变换定义有

$$Z[e^{\mp at}f(t)] = \sum_{k=0}^{\infty} f(kT)e^{\mp akT}z^{-k}$$

令 $z_1 = ze^{\pm aT}$，则上式可写成

$$Z[e^{\mp at}f(t)] = \sum_{k=0}^{\infty} f(kT)z_1^{-k} = F(z_1)$$

代入 $z_1 = ze^{\pm aT}$，得

$$Z[e^{\mp at}f(t)] = F(ze^{\pm aT})$$

3.2.4 时域卷积性质

已知

$$x(k) \leftrightarrow X(z) \quad (\alpha_1 < |z| < \beta_1)$$
$$h(k) \leftrightarrow H(z) \quad (\alpha_2 < |z| < \beta_2)$$

则有

$$x(k) * h(k) \leftrightarrow X(z)H(z)$$

3.2.5 微分性

如果有

$$x(k) \leftrightarrow X(z) \quad \alpha < |z| < \beta$$

那么有

$$kx(k) \leftrightarrow -z\frac{\mathrm{d}X(z)}{\mathrm{d}z} \quad \alpha < |z| < \beta$$

3.2.6 积分性

已知

$$x(k) \leftrightarrow X(z) \quad \alpha < |z| < \beta$$

则有

$$\frac{x(k)}{k+m} \leftrightarrow z^m \int_z^{\infty} \frac{X(\eta)}{\eta^{m+1}}\mathrm{d}\eta \quad \alpha < |z| < \beta$$

3.2.7 时域求和

如果有

$$x(k) \leftrightarrow X(z) \quad \alpha < |z| < \beta$$

那么有

$$f(k) = \sum_{i=-\infty}^{k} x(i) \leftrightarrow \frac{z}{z-1}X(z) \quad \max(\alpha,1) < |z| < \beta$$

3.2.8 初值定理

如果函数 $f(t)$的 Z 变换为 $F(z)$,并存在极限$\lim\limits_{z\to\infty}F(z)$,则

$$\lim_{k\to 0} f(kT) = \lim_{z\to\infty} F(z)$$

3.2.9 终值定理

假定 $f(t)$的 Z 变换为 $F(z)$,并假定函数$(1-z^{-1})F(z)$在 z 平面的单位圆上或圆外没有极点,则

$$\lim_{k\to\infty} f(kT) = \lim_{z\to 1}(1-z^{-1})F(z)$$

3.3 Z反变换

定义 $X(z)$的Z反变换(IZT)为

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

式中,C为收敛域内一条环绕原点逆时针闭合围线。

求Z反变换的方法主要有两种,分别是留数法和部分分式展开法。

由留数定理可知:若函数在围线C上连续,在C以内有 K 个极点,而在C以外有 M 个极点,则有

$$\frac{1}{2\pi j}\int X(z)z^{n-1}dz=\sum_k \mathrm{Res}\left[X(z)z^{n-1}\right]_{z=z_k}$$

当极点为一阶时的留数为

$$\mathrm{Res}\left[X(z)z^{n-1}\right]_{Z=Z_r}=\left[(z-z_r)X(z)z^{n-1}\right]_{z=z_r}$$

当极点为多重极点时的留数为

$$\mathrm{Res}\left[X(z)z^{n-1}\right]_{z=z_r}=\frac{1}{(l-1)!}\frac{d^{l-1}}{dz^{l-1}}\left[(z-z_r)^l X(z)z^{n-1}\right]_{z=z_r}$$

部分分式法把 x 的一个实系数的真分式分解成几个分式的和,使各分式具有$\frac{a}{(x+A)^k}$或者$\frac{ax+b}{(x^2+Ax+B)^k}$的形式。

通常情况下传递函数可分解为

$$X(z)=\frac{B(z)}{A(z)}=\frac{\sum_{i=0}^{M}b_i z^{-i}}{1+\sum_{i=1}^{N}a_i z^{-i}}$$

MATLAB的符号数学工具箱提供了计算Z变换的函数ztrans()和Z反变换的函数iztrans(),其调用形式为

```
F = ztrans(f),
f = iztrans(F)
```

其中,右端的f和F分别为时域表示式和Z域表示式的符号表示,可应用函数sym来实现,其调用格式为

```
S = sym(A)
```

在MATLAB中,留数法求Z反变换可以使用函数residuez实现,调用格式为

```
[R P K] = residuez(B,A);
```

其中,B和A分别为X(z)的多项式中分子多项式和分母多项式的系数向量;返回值R为留数向量,P为极点向量,二者均为列向量;返回值K为直接项系数,仅在分子多项式最高次幂大于等于分母多项式最高次幂时存在,否则,返回值为空。

【例 3-1】 求 $f(n)=\sin(ak)u(k)$的 Z 变换和 $F(z)=\frac{z}{(z-3)^2}$的 Z 反变换。

程序如下：

```
f = sym('sin(a * k)');
F = ztrans(f)
F = sym('z/(z - 3)^2');
f = iztrans(F)
```

Z 变换运行结果如下：

```
F =
 (z * sin(a))/(z^2 - 2 * cos(a) * z + 1)
```

Z 逆变换运行结果如下：

```
f =
3^n/3 + (3^n * (n - 1))/3
```

【例 3-2】 求 $X(z)=\frac{(1+0.4z^{-1})^2}{(1+0.8z^{-1})^2(1-0.5z^{-1})^2(1+0.1z^{-1})}$，$|z|>0.8$ 的 Z 反变换。

程序如下：

```
clear all;close all;clc;
B = poly([ - 0.4  - 0.4]);
A = poly([ - 0.8  - 0.8 0.5 0.5  - 0.1]);
[R P K] = residuez(B,A);
R = R'
P = P'
K
```

运行结果如下：

```
R =
   0.2842 + 0.0000i   0.1082 - 0.0000i   0.2031 - 0.0000i   0.3994 + 0.0000i   0.0051
 + 0.0000i
P =
  - 0.8000 - 0.0000i  - 0.8000 + 0.0000i   0.5000 - 0.0000i   0.5000 + 0.0000i
 - 0.1000 + 0.0000i
K =
     []
```

3.4 离散系统中的 Z 域描述

线性时不变离散系统可用线性常系数差分方程描述，即

$$\sum_{i=0}^{N} a_i y(n-i) = \sum_{j=0}^{M} b_j x(n-j)$$

其中，$y(k)$为系统的输出序列，$x(k)$为输入序列。

将上式两边进行 Z 变换得到

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{j=0}^{M}b_jz^{-j}}{\sum_{i=0}^{N}a_iz^{-i}}=\frac{B(z)}{A(z)}$$

因式分解后有

$$H(z)=C\frac{\prod_{j=1}^{M}(z-q_j)}{\prod_{i=1}^{N}(z-p_i)}$$

其中，C 为常数，$q_j(j=1,2,\cdots,M)$为 $H(z)$的 M 个零点，$p_i(i=1,2,\cdots,N)$为 $H(z)$的 N 个极点。系统函数 $H(z)$的零极点分布完全决定了系统的特性，若某系统函数的零极点已知，则系统函数便可确定下来。

3.4.1 离散系统函数频域分析

离散系统的频率响应 $H(e^{j\omega})$对于某因果稳定离散系统，如果激励序列为正弦序列

$$x(n)=A\sin(\omega_0 n)u(n)$$

则系统的稳态响应为

$$y_{ss}(n)=A\mid H(e^{j\omega})\mid\sin[\omega n+\varphi(\omega)]u(n)$$

定义离散系统的频率响应为

$$H(e^{j\omega})=H(z)\mid_{z=e^{j\omega}}=\mid H(e^{j\omega})\mid e^{j\varphi(\omega)}$$

其中，$|H(e^{j\omega})|$为离散系统的幅频特性，$\varphi(\omega)$为离散系统的相频特性，$H(e^{j\omega})$是以 2π 为周期的周期函数，只要分析 $H(e^{j\omega})$在$|\omega|\leqslant\pi$ 范围内的情况，便可分析出系统的整个频率特性。

利用几何矢量求解离散系统的频率响应，设

$$e^{j\omega}-p_i=A_ie^{j\theta_i}$$

$$e^{j\omega}-q_j=B_je^{j\psi_j}$$

那么离散系统的频率响应为

$$H(e^{j\omega})=\frac{\prod_{j=1}^{M}B_je^{j(\psi_1+\psi_2+\cdots+\psi_M)}}{\prod_{i=1}^{N}A_ie^{j(\theta_1+\theta_2+\cdots+\theta_N)}}=\mid H(e^{j\omega})\mid e^{j\varphi(\omega)}$$

那么系统的幅频特性和相频特性为

$$\mid H(e^{j\omega})\mid=\frac{\prod_{j=1}^{M}B_j}{\prod_{i=1}^{N}A_i}$$

$$\varphi(\omega)=\sum_{j=1}^{M}\psi_j-\sum_{i=1}^{N}\theta_i$$

利用 MATLAB 来求解频率响应的过程如下：

(1) 根据系统函数 $H(z)$定义分子、分母多项式系数向量 B 和 A；

(2) 调用前述的 ljdt()函数求出 $H(z)$的零极点，并绘出零极点图；

(3) 定义 z 平面单位圆上的 k 个频率分点；

(4) 求出 $H(z)$所有的零点和极点到这些等分点的距离；

(5) 求出 $H(z)$所有的零点和极点到这些等分点矢量的相角；

(6) 求出系统的$|H(e^{j\omega})|$和 $\varphi(\omega)$；

(7) 绘制指定范围内系统的幅频曲线和相频曲线。

在 MATLAB 中，函数 freqz 用于求离散时间系统频响特性，该函数的调用方法如下：

```
[H,w] = freqz(B,A,N)
[H,w] = freqz(B,A,N,'whole')
```

其中，B 与 A 分别表示 H(z)的分子与分母多项式的系数向量；N 为正整数，默认值为 512；返回值 ω 包含[0，π]范围内的 N 个频率等分点；返回值 H 则是离散时间系统频率响应 $H(e^{j\omega})$在 0～π 范围内 N 的频率处对应的值。

【例 3-3】 绘制如下系统的频响曲线：

$$H(z)=\frac{z-1.3}{z}$$

程序如下：

```
B = [1 -1.3];
A = [1 0];
[H,w] = freqz(B,A,400,'whole');
Hf = abs(H);
Hx = angle(H);
clf
subplot(121)
plot(w,Hf)
title('离散系统幅频特性曲线')
xlabel('频率');ylabel('幅度')
grid on
subplot(122)
plot(w,Hx)
xlabel('频率');ylabel('幅度')
grid on
title('离散系统相频特性曲线')
```

运行结果如图 3-1 所示。

【例 3-4】 绘制离散系统的幅频响应和相频响应示例。

程序如下：

```
clear all;close all;clc;
w = (-4 * pi:0.001:4 * pi) + eps;
```

```
X = 1./(1 - 0.6 * exp( - j * w));
subplot(211),
plot(w/pi,abs(X),'LineWidth',2);
xlabel('\omega/\pi');
ylabel('|H(e^j^\omega)|');
title('幅频响应');
axis([ - 3.2 3.2 0.5 2.2]);
grid;
subplot(212),
plot(w/pi,angle(X),'LineWidth',2);
xlabel('\omega/\pi');
ylabel('\theta(\omega)');
title('相频响应');
axis([ - 3.2 3.2  - 0.6 0.6]);grid;
set(gcf,'color','w');
```

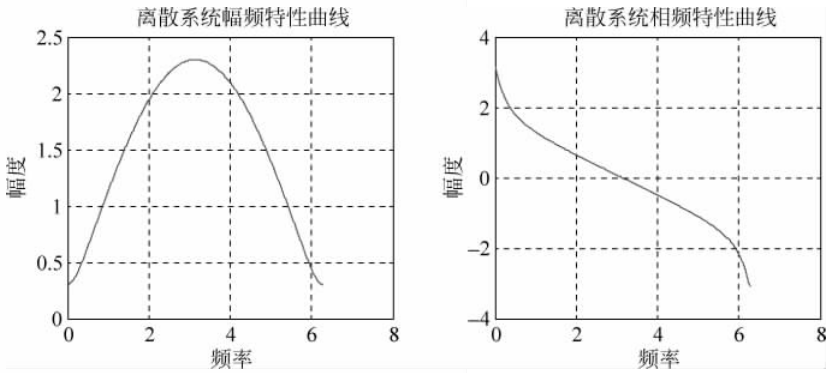

图 3-1　离散系统频响曲线

运行结果如图 3-2 所示。

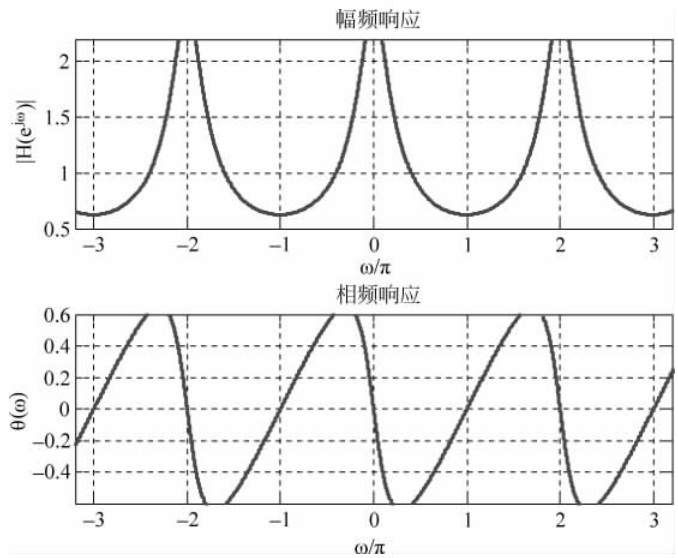

图 3-2　系统的幅频响应和相频响应

3.4.2 离散系统函数零点分析

离散时间系统的系统函数定义为

$$H(z)=\frac{Y(z)}{X(z)}$$

如果系统函数的有理函数表达式为

$$H(z)=\frac{b_1z^m+b_2z^{m-1}+\cdots+b_mz+b_{m+1}}{a_1z^m+a_2z^{n-1}+\cdots+a_nz+a_{n+1}}$$

在 MATLAB 中，系统函数的零极点就可以通过函数 roots 得到，也可以借助函数 tf2zp 得到，tf2zp 的语句格式为

```
[Z,P,K] = tf2zp(B,A)
```

其中，B 与 A 分别表示为 H(z)的分子与分母多项式的系数向量。上式的作用是将 $H(z)$ 的有理分式表示为转换为零极点增益形式，即

$$H(z)=k\frac{(z-z_1)(z-z_2)\cdots(z-z_m)}{(z-p_1)(z-p_x)\cdots(z-p_n)}$$

zplane 函数用于绘制 $H(z)$的零极点图，该函数的调用格式为

```
zplane(z,p)
```

绘制出列向量 z 中的零点(以符号“o”表示)和列向量 p 中的极点(以符号“×”表示)，以及参考单位圆，在多阶零点和极点的右上角标出其阶数。如果 z 和 p 为矩阵，则会以不同颜色绘出 z 和 p 各列中的零点和极点。

【例 3-5】 已知某离散系统的系统函数为

$$H(z)=\frac{2z+1}{3z^5-2z^4+1}$$

试用 MATLAB 求出该系统的零极点，并画出零极点分布图，判断系统是否稳定。用 roots()求得 $H(z)$的零极点后，就可以用 plot()函数绘制出系统的零极点图。

子程序如下：

```
function ljdt(A,B)
p = roots(A);                          % 求系统极点
q = roots(B);                          % 求系统零点
p = p';                                % 将极点列向量转置为行向量
q = q';                                % 将零点列向量转置为行向量
x = max(abs([p q 1]));                 % 确定纵坐标范围
x = x + 0.1;
y = x;                                 % 确定横坐标范围
clf
hold on
axis([ - x x  - y y])                  % 确定坐标轴显示范围
w = 0:pi/300:2 * pi;
t = exp(i * w);
plot(t)                                % 画单位圆
axis('square')
```

```
plot([ - x x],[0 0])                % 画横坐标轴
plot([0 0],[ - y y])                % 画纵坐标轴
text(0.1,x,'jIm[z]')
text(y,1/10,'Re[z]')
plot(real(p),imag(p),'x')           % 画极点
plot(real(q),imag(q),'o')
title('零极点图')                    % 标注标题
hold off
```

程序如下：

```
% 绘制零极点分布图的实现程序
a = [3 - 2 0 0 0 1];
b = [2 1];
ljdt(a,b)
p = roots(a)
q = roots(b)
pa = abs(p)
```

如图 3-3 所示，运行结果如下：

```
p =
   0.8212 + 0.4270i
   0.8212 - 0.4270i
  - 0.1367 + 0.7316i
  - 0.1367 - 0.7316i
  - 0.7024 + 0.0000i
q =
   - 0.5000
pa =
    0.9256
    0.9256
    0.7442
    0.7442
    0.7024
```

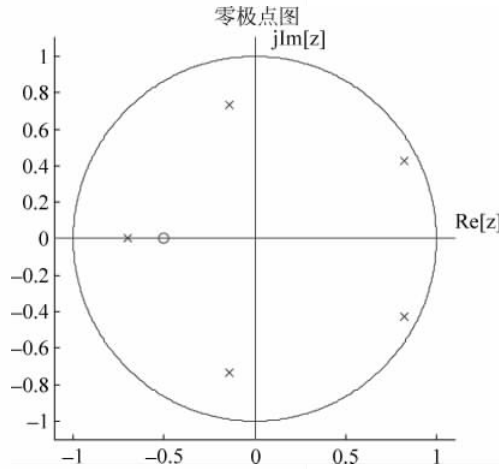

图 3-3　零极点图

【例 3-6】 各种系统零极点图的实现。

程序如下：

```
%绘制情况(a)系统零极点分布图及系统单位序列响应
z = 0;                              %定义系统零点位置
p = 0.25;                           %定义系统极点位置
k = 1;                              %定义系统增益
subplot(221)
zplane(z,p)
grid on;
%绘制系统零极点分布图
subplot(222);
[num,den] = zp2tf(z,p,k);           %零极点模型转换为传递函数模型
impz(num,den)
%绘制系统单位序列响应时域波形
title('h(n)')
grid on;
%定义标题
%绘制情况(b)系统零极点分布图及系统单位序列响应
p = 1;
subplot(223);
zplane(z,p)
grid on;
[num,den] = zp2tf(z,p,k);
subplot(224);
impz(num,den)
title('h(n)')
grid on;
```

运行结果如图 3-4 所示。

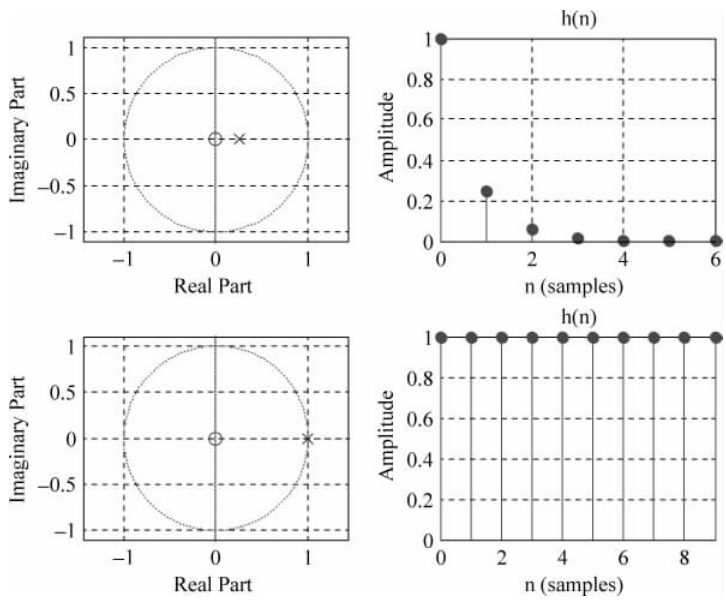

图 3-4 a 和 b 系统结果图

3.4.3 离散系统差分函数求解

连续函数 $f(t)$经过采样后，获得采样函数 $f(kT)$，那么一阶向前和向后差分形式分别为

$$\Delta f(k) = f(k+1) - f(k)$$
$$\nabla f(k) = f(k) - f(k-1)$$

二阶向前和向后的差分形式分别为

$$\Delta^2 f(k) = \Delta f(k+1) = f(k+2) - 2f(k+1) + f(k)$$
$$\nabla^2 f(k) = \nabla[\Delta f(k)] = f(k) - 2f(k-1) + f(k-2)$$

根据上式可以推导向前和向后的 n 阶差分为

$$\Delta^n f(k) = \Delta^{n-1} f(k+1) - \Delta^{n-1} f(k)$$
$$\nabla^n f(k) = \nabla^{n-1} f(k) - \nabla^{n-1} f(k-1)$$

连续系统的时间序列方程为

$$\mathrm{d}^2 c(t)/\mathrm{d}t^2 + a\mathrm{d}c(t)/\mathrm{d}t + bc(t) = kr(t)$$

上式中的微分用差分替代，则有

$$\mathrm{d}^2 c(t)/\mathrm{d}t^2 = \Delta^2 c(t) = c(k+2) - 2c(k+1) + c(k)$$
$$\mathrm{d}c(t)/\mathrm{d}t = c(k+1) - c(k)$$

推导到离散时间系统，$c(k)$代替 $c(t)$，$r(k)$代替 $r(t)$，则有

$$[c(k+2) - 2c(k+1) + c(k)] + a[c(k+1) - c(k)] + bc(k) = kr(k)$$

整理得

$$c(k+2) + (a-2)c(k+1) + (1-a+b)c(k) = kr(k)$$

由此可以推出一般离散系统的差分方程为

$$c(k+n) + a_1 c(k+n-1) + a_2 c(k+n-2) + \cdots a_n c(k)$$
$$= b_0 r(k+m) + b_1 r(k+m-1) + \cdots + b_m r(k)$$

差分方程的解也分为通解与特解，通解是与方程初始状态有关的解，特解与外部输入有关，它描述系统在外部输入作用下的强迫运动。

【例 3-7】 求解如下差分方程：

$y(n) - 0.5y(n-1) - 0.45y(n-2) = 0.55x(n) + 0.5x(n-1) - x(n-2)$，

其中，$x(n) = 0.7^n \varepsilon(n)$，初始状态 $y(-1)=1$，$y(-2)=2$，$x(-1)=2$，$x(-2)=3$。

程序如下：

```
num = [0.55 0.5 -1];
den = [1 -0.5 -0.45];
x0 = [2 3];y0 = [1 2];
N = 50;
n = [0:N-1]';
x = 0.7.^n;
```

```
Zi = filtic(num,den,y0,x0);
[y,Zf] = filter(num,den,x,Zi);
plot(n,x,'r - ',n,y,'b -- ');
title('响应');
xlabel('n');ylabel('x(n) - y(n)');
legend('输入 x','输出 y',1);
grid;
```

运行结果如图 3-5 所示。

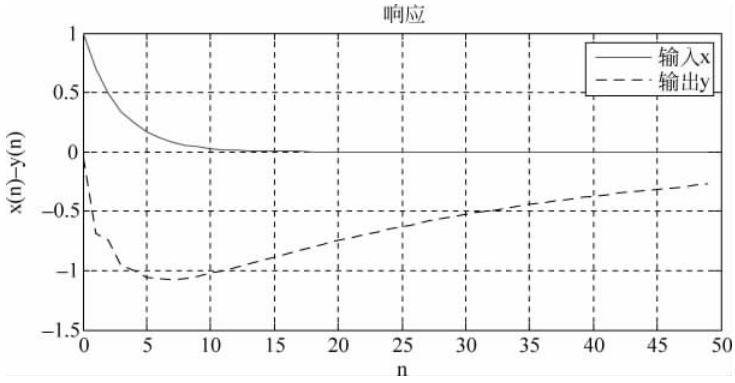

图 3-5　离散系统差分方程解

【例 3-8】 编制程序求解下列两个系统差分方程的单位脉冲响应，并绘出其图形。

$$y[n]+0.75y[n-1]+0.125y[n-2]=x[n]-x[n-1]$$

程序如下：

```
clc;
N = 32;
x_delta = zeros(1,N);
x_delta(1) = 1;
p = [1, - 1,0]
d = [1,0.75,0.125];
h1_delta = filter(p,d,x_delta);
subplot(211);
stem(0:N - 1,h1_delta,'r');hold off;
xlabel('方程 1 的单位脉冲响应');
x_unit = ones(1,N);
h1_unit = filter(p,d,x_unit);
subplot(212);
stem(0:N - 1,h1_unit,'r');hold off;
xlabel('方程 1 的阶跃响应');
```

运行结果如图 3-6 所示。

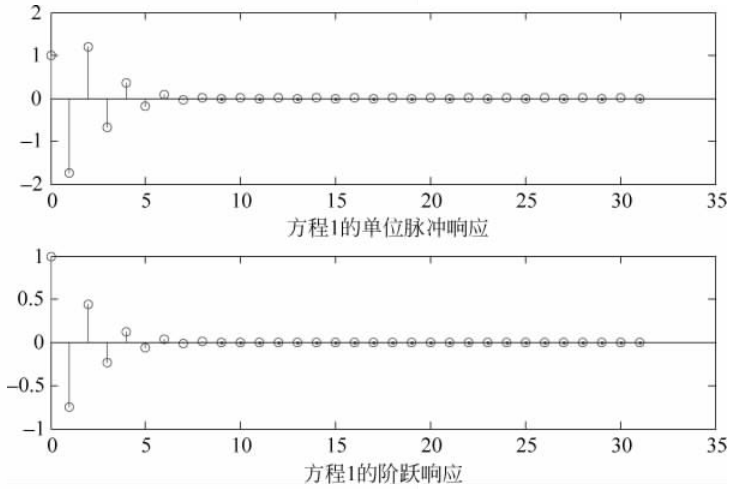

图 3-6　方程的脉冲响应

3.5　傅里叶级数和傅里叶变换

描述周期现象最简单的周期函数是物理学上所说的谐波函数，它是由正弦或者余弦函数来表示

$$y(t) = A\cos(\omega t + \varphi)$$

利用三角公式，上式可以写成

$$y(t) = A\cos\varphi\cos\omega t - A\sin\varphi\sin t$$

由于 φ 是常数，令 $a = A\cos\varphi$，$b = -A\sin\varphi$，那么可以得到

$$y(t) = a\cos\omega t + b\sin\omega t$$

式中

$$A = \sqrt{a^2 + b^2}, \quad \varphi = \arctan\left(-\frac{b}{a}\right)$$

从这里可以看出：一个带初相位的余弦函数可以看成是一个不带初相位的正弦函数与一个不带初相位的余弦函数的合成。

谐波函数是周期函数中最简单的函数，它描述的也是最简单的周期现象，在实际中所碰到的周期现象往往要比它复杂很多，但这些复杂的函数都可以近似分解成不同频率的正弦函数和余弦函数。下面就介绍一种复杂的函数分解为一系列不同频率的正弦函数和余弦函数的方法。

在高等代数中有这样一个问题，怎么将一个周期为 2^l 函数分解成傅里叶级数，给出的解答式为

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi x}{l} + b_n\sin\frac{n\pi x}{l}\right)$$

其中，

$$a_0 = \frac{1}{l}\int_{-l}^{l} f(x)\mathrm{d}x, \quad a_n = \frac{1}{l}\int_{-l}^{l} f(x)\cos\frac{n\pi x}{l}\mathrm{d}x, \quad b_n = \frac{1}{l}\int_{-l}^{l} f(x)\sin\frac{n\pi x}{l}\mathrm{d}x$$

如果 $f(x)$是奇函数，积分上下限相互对称，则此时 $f(x)\cos\frac{n\pi x}{l}$项成为奇函数，可以知道 a_n 均为零，得到的傅里叶正弦级数为

$$f(x)=\sum_{n=1}^{\infty}b_n\sin\frac{n\pi x}{l}$$

式中，b_n 的积分可以简写为

$$b_n=\frac{2}{l}\int_0^l f(x)\sin\frac{n\pi x}{l}\mathrm{d}x$$

如果 $f(x)$是偶函数，同样因为积分上下限相互对称，这时 $f(x)\sin\frac{n\pi x}{l}$为奇函数，故 b_n 均为零，得到的傅里叶级数是余弦级数，即

$$f(x)=\frac{a_0}{2}+\sum_{n=1}^{\infty}a_n\cos\frac{n\pi x}{l}$$

式中，a_n 可以简写为

$$a_n=\frac{2}{l}\int_0^l f(x)\cos\frac{n\pi x}{l}\mathrm{d}x$$

3.6 周期序列的离散傅里叶级数

对于周期信号，通常都可以用傅里叶级数来描述，如连续时间周期信号

$$f(t)=f(t+mT)$$

用指数形式的傅里叶级数表示为

$$f(t)=\sum_{n=-\infty}^{\infty}F_n\mathrm{e}^{\mathrm{j}n\Omega t}$$

可以看成信号被分解成不同次谐波的叠加，每个谐波都有一个幅值，表示该谐波分量所占的比重。其中 $\mathrm{e}^{\mathrm{j}\Omega t}$为基波，基频为 $\Omega=2\pi/T$(T 为周期)。设$\tilde{x}(n)$是周期为 N 的一个周期序列，即$\tilde{x}(n)=\tilde{x}(n+rN)$，$r$ 为任意整数，用指数形式的傅里叶级数表示应该为

$$\tilde{x}(n)=\sum_{k=-\infty}^{\infty}\widetilde{X}_k\mathrm{e}^{\mathrm{j}kw_0n}$$

其中，$\omega_0=2\pi/N$ 是基频，基频序列为 $\mathrm{e}^{\mathrm{j}w_0n}$。

下面来分析一下第$(K+rN)$次谐波 $\mathrm{e}^{\mathrm{j}(k+rN)w_0n}$和第 k 次谐波 $\mathrm{e}^{\mathrm{j}kw_0n}$之间的关系。因为 $\omega_0=2\pi/N$，代入表达式中，得到 $\mathrm{e}^{\mathrm{j}(k+rN)w_0n}=\mathrm{e}^{\mathrm{j}kw_0n}$，$r$ 为任意整数，这说明$(K+rN)$次谐波能够被第 k 次谐波代表，也就是说，在所有的谐波成分中，只有 N 个是独立的，用 N 个谐波就可完全地表示出$\tilde{x}(n)$，K 的取值从 0 到 $N-1$。这样，$\tilde{x}(n)=\frac{1}{N}\sum_{k=0}^{N-1}\widetilde{X}_k\mathrm{e}^{\mathrm{j}kw_0n}$，$\frac{1}{N}$是为了计算的方便而加入的。

下面来看看$\widetilde{X}_k$ 如何根据$\tilde{x}(n)$来求解。先来证明复指数的正交性：

$$\sum_{n=0}^{N-1}\mathrm{e}^{\mathrm{j}\left(\frac{2\pi}{N}\right)(k-r)n}=\begin{cases}1 & k-r=mN,m\text{ 为整数}\\ 0 & \text{其他}\end{cases}$$

其中，该表达式是对 n 求和，而表达式的结果取决于$(k-r)$的值。

在 $\tilde{x}(n)=\frac{1}{N}\sum_{k=0}^{N-1}\widetilde{X}_k e^{jkw_0 n}$ 两边都乘以 $e^{-j(2\pi/N)rn}$，于是有

$$\sum_{n=0}^{N-1}\tilde{x}(n)e^{-j(2\pi/N)rn}=\sum_{n=0}^{N-1}\frac{1}{N}\sum_{k=0}^{N-1}\widetilde{X}_k e^{j(2\pi/N)(k-r)n}$$

交换求和顺序，再根据前面证明的正交性结论可以得出

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

从$\widetilde{X}(k)$的表达式可以看出$\widetilde{X}(k)$也是周期为 N 的周期序列，即$\widetilde{X}(k)=\widetilde{X}(k+N)$。上式即为周期序列的傅里叶级数。

3.7 离散的傅里叶变换

现在将上述公式应用于离散的傅里叶级数中。

在信号处理中，遇到的常常不是一个函数，而是一个离散的数列，举一个例子，等间隔时间取样的时间序列$\{x_0,x_1,x_2,\cdots,x_{N-1}\}$，在这里数据的个数是 N，一般取 N 为偶数，如果取 2 的对数对应的偶数能够加快计算速度。

下面对取值范围进行改造，首先，得到的数字信号只能在正的时间段取值，在负的时间段不能取值，但由于取的是无限长的周期序列，周期为 $2l$，因此，把取值范围$(-l,l)$修改为$(0,2l)$，这样就可以避免在负的时间段取值。

由于处理的是离散的数据序列，因此不能再用积分，而应用积分的离散形式，用求和来表示，即

$$\int_0^{2l}\rightarrow\sum_{k=1}^{N}x_k$$

在$(0,2l)$里等间隔取 N 个取值点，取样时间间隔为 $dx\rightarrow\Delta t$，其中，$l=\frac{N\Delta t}{2}$。

有了上述的改正，可以得到

$$f(x)\rightarrow\{x_0,x_1,x_2,\cdots,x_{N-1}\}$$

$$\frac{n\pi x}{l}\rightarrow\frac{k\pi i\Delta t}{\frac{N\Delta t}{2}}=\frac{2\pi ki}{N}$$

离散形式为

$$x_i=\frac{a_0}{2}+\sum_{k=1}^{m}\left(a_k\cos\frac{2\pi ki}{N}+b_k\sin\frac{2\pi ki}{N}\right)$$

式中

$$a_0=\frac{1}{\frac{N\Delta t}{2}}\sum_{i=0}^{N-1}x_i=\frac{2}{N}\sum_{i=0}^{N-1}x_i$$

$$a_k=\frac{1}{\frac{N\Delta t}{2}}\sum_{i=0}^{N-1}x_i\cos\frac{2\pi ki}{N}=\frac{2}{N}\sum_{i=0}^{N-1}x_i\cos\frac{2\pi ki}{N}$$

$$b_k = \frac{1}{\frac{N\Delta t}{2}}\sum_{i=0}^{N-1} x_i \sin\frac{2\pi ki}{N} = \frac{2}{N}\sum_{i=0}^{N-1} x_i \sin\frac{2\pi ki}{N}, \quad k = 1,2,3,\cdots,m$$

在实际数据处理中，k 一般取 $N/2$，此时波的周期最小，获得的频率范围最大，所以想要获得高频率的信号，就需要缩短取样间隔。

【例 3-9】 计算序列 $x(n)$的 DFT。

程序如下：

```
clear all;
t = linspace(1e - 3,100e - 3,10);
xn = sin(100 * 2 * pi * t);                    % 产生有限序列 x(n)
N = length(xn);                                % 获得序列的长度
WNnk = dftmtx(N);
Xk = xn * WNnk;                                % 计算 x(n)的 DFT
subplot(1,2,1);stem(1:N,xn);
subplot(1,2,2);stem(1:N,abs(Xk));
```

运行结果如图 3-7 所示。

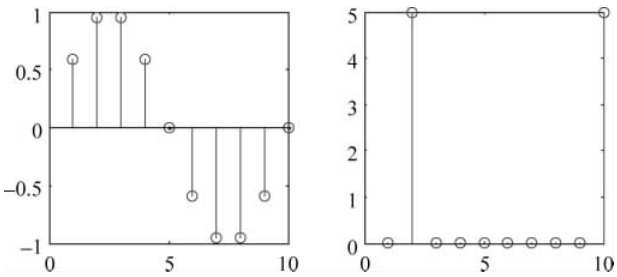

图 3-7 时域离散序列 $x(n)$和 $x(n)$的 DFT 变换结果

【例 3-10】 已知复正弦序列 $x_1(n)=e^{j\frac{\pi}{8}n}R_N(n)$，余弦序列 $x_2(n)=\cos\left(\frac{\pi}{8}n\right)R_N(n)$，分别对序列求当 $N=16$ 和 $N=8$ 时的 DFT，并绘出幅频特性曲线，并分析两种 N 值下 DFT 是否有差别及差别产生的原因。

程序如下：

```
N = 16;N1 = 8;
n = 0:N - 1;k = 0:N1 - 1;
x1n = exp(j * pi * n/8);                       % 产生 x1(n)
X1 = fft(x1n,N);                               % 计算 N 点 DFT[x1(n)]
X2 = fft(x1n,N1);                              % 计算 N1 点 DFT[x1(n)]
x2n = cos(pi * n/8);                           % 产生 x2(n)
X3 = fft(x2n,N);                               % 计算 N 点 DFT[x2(n)]
X4 = fft(x2n,N1);                              % 计算 N1 点 DFT[x2(n)]
subplot(2,2,1);
stem(n,abs(X1),'.');
axis([0,20,0,20]);
ylabel('|X1(k)|')
title('16 点的 DFT[x1(n)]')
```

```
subplot(2,2,2);
stem(n,abs(X3),'.');
axis([0,20,0,20]);
ylabel('|X2(k)|')
title('16 点的 DFT[x2(n)]')
subplot(2,2,3);
stem(k,abs(X2),'.');
axis([0,20,0,20]);
ylabel('|X1(k)|')
title('8 点的 DFT[x1(n)]')
subplot(2,2,4);
stem(k,abs(X4),'.');
axis([0,20,0,20]);
ylabel('|X2(k)|')
title('8 点的 DFT[x2(n)]')
```

运行结果如图 3-8 所示。

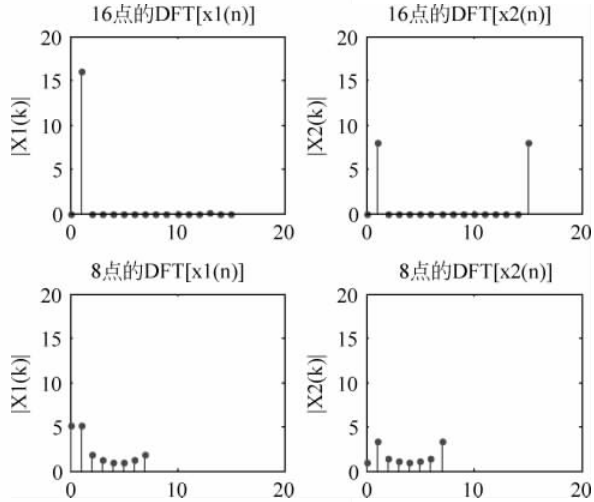

图 3-8　离散傅里叶变换

3.8　离散傅里叶变换的性质

3.8.1　线性

注意特殊情况下如何定线性组合后序列的长度，以长度大的为周期，对任意常数 a_m $(1\leqslant m\leqslant M)$，有

$$\mathrm{DFT}\left[\sum_{m=1}^{M}a_m x_m(n)\right]\Leftrightarrow\sum_{m=1}^{M}a_m\mathrm{DFT}[x_m(n)]$$

3.8.2 循环移位

循环移位定义为

$$y(n) = x((n-m))_N R_N(n)$$

循环移位定理：

若 DFT$[x(n)]=X(k)$，$y(n)=x((n-m))_N R_N(n)$，则 DFT$[y(n)]=W_N^{mk}X(k)$。

循环移位形式与 DFS 的周期移位相同，表明序列圆周移位后的 DFT 为 X(k)乘上相移因子 W_N^{mk}，即时域中圆周移 m 位，仅使频域信号产生 W_N^{mk} 的相移，而幅度频谱不发生改变，即

$$\left| |W_N^{mk}X(k)| = X(k) \right|$$

3.8.3 循环卷积定理

$x_1(n)$和 $x_2(n)$的长度都为 N，如果 $Y(k)=X_1(k)X_2(k)$，则

$$\begin{aligned} y(n) &= \left[\sum_{m=0}^{N-1} x_1(m)x_2((n-m))_N\right]R_N(n) \\ &= \left[\sum_{m=0}^{N-1} x_2(m)x_1((n-m))_N\right]R_N(n) = x_1(n) \otimes x_2(n) \end{aligned}$$

根据定理可以求出圆周卷积，当然求圆周卷积可以借助 DFT 来计算，即 IDFT$[Y(k)]=y(n)$。

可见圆周卷积与周期卷积在主值区的结果相同，所以求圆周卷积可以把序列延拓成周期序列，进行周期卷积，然后取主值。

3.8.4 共轭对称性

如果对于给定的整数 M，复序列 $x(n)$满足下式：

$$x(n) = \pm x^*(M-n) \quad (-\infty < n < +\infty)$$

则称 $x(n)$关于 $M/2$ 共轭对称(式中取“+”)或共轭反对称(式中取“−”)。

给定整数 M，任何序列 $x(n)$都可以分解成关于 $M/2$ 共轭对称的序列 $x_e(n)$和共轭反对称的序列 $x_o(n)$之和，即

$$x(n) = x_e(n) + x_o(n)$$

其中，$x_e(n)=\frac{1}{2}[x(n)+x^*(M-n)]$，$x_o(n)=\frac{1}{2}[x(n)-x^*(M-n)]$

如果 $x(n)=x_R(n)+jx_I(n)$，$x(n)$的 DTFT 为 $X(e^{j\omega})=X_e(e^{j\omega})+X_e(e^{j\omega})$，$X_e(e^{j\omega})$，$X_o(e^{j\omega})$为 $X(e^{j\omega})$的实部和虚部，则

$$\begin{cases} X_e(e^{j\omega}) = \text{DTFT}[x_R(n)] \\ X_o(e^{j\omega}) = \text{DTFT}[jx_I(n)] \end{cases}$$

如果 $x(n)=x_e(n)+x_o(n)$，$x(n)$的 DTFT 为 $X(e^{j\omega})=X_R(e^{j\omega})+jX_I(e^{j\omega})$，则

$$\begin{cases} X_R(e^{j\omega}) = \mathrm{DTFT}[x_e(n)] \\ jX_I(e^{j\omega}) = \mathrm{DTFT}[x_o(n)] \end{cases}$$

3.9 频率域采样

前面讨论过周期序列的离散傅里叶级数的系数$\widetilde{X}(k)$的值和$\tilde{x}(n)$的一个周期的 Z 变换在单位圆(即序列的傅里叶变换)的 N 个均匀点上的抽样值相等，这其实就是频域的抽样。因此我们得到一个结论：可以用 N 个点的 $X(k)$来代表序列的傅里叶变换。

注意，不是所有的序列都可以这样。已经证明过 $\tilde{x}(n)=\sum_{r=-\infty}^{\infty}x(n+rN)$，即周期序列可以看作是非周期序列以某个 N 为周期进行延拓而成，只有在 N 大于非周期序列 $x(n)$的长度时，延拓后才不会发生重叠。所以要求 $x(n)$为有限长序列，且长度小于等于 N，这样就可以用$\widetilde{X}(k)$来代表 $X(e^{j\omega})$。

其实，$\widetilde{X}(k)$的一个周期就可以代表 $X(e^{j\omega})$，所以只看一个周期，即 $X(k)$，接下来分析如何用 $X(k)$来表示 $X(e^{j\omega})$。

有限长序列 $x(n)(0\leqslant n\leqslant N-1)$的 Z 变换为

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

$$=\frac{1}{N}\sum_{k=0}^{N-1}X(k)\,\frac{1-W_N^{-Nk}z^{-N}}{1-W_N^{-k}z^{-1}}=\frac{1-z^{-N}}{N}\sum_{k=0}^{N-1}\frac{X(k)}{1-W_N^{-k}z^{-1}}$$

这就是用 N 个频率抽样值来恢复 $X(z)$的插值公式。上式中把 z 换成 $e^{j\omega}$就变成用 N 个频率抽样值来恢复 $X(e^{j\omega})$的插值公式。

3.9.1 频率响应的混叠失真

抽样定理要求 $f_s>2f_h$，一般取 $f_s=(2.5\sim3.0)f_h$，如不满足该条件，则会产生频域响应的周期延拓分量重叠现象，即频率响应的混叠失真。根据 $f_0=f_s/N$，若增加 f_s，而 N 固定时，则 f_0 要增加，导致分辨率下降。反之，要提高分辨率，即 f_0 减小，当 N 给定时，则导致 f_s 的减小。若想不发生混叠，则 f_h 要减小。这样要想兼顾 f_h 和 f_0，只有增加 N。得到 $N=f_s/f_0>(2f_h)/f_0$，这是实现 DFT 算法必须满足的最低条件。

3.9.2 频谱泄漏

实际情况下，我们取的信号都是有限长的，即对原始序列做加窗处理使其成为有限长，时域的乘积对应频域的卷积，造成频谱的泄漏。

减小泄漏的方法：可以取更长的数据(与原始数据就越相近)，缺点运算量加大；可以选择窗的形状，从而使窗谱的旁瓣能量更小。

3.9.3 栅栏效应

DFT 上看到的谱线都是离散的，而从序列的傅里叶变换知道谱线是连续的，所以相当于看到谱的一些离散点，而不是全部，感觉像是透过栅栏看到的情景，称为栅栏效应。

3.9.4 频率分辨率

增加分辨率只有通过加大取样点 N，但不是补零的方式来增加 N，因为补零不是原始信号的有效信号。

【例 3-11】 已知模拟信号 $X(k)$，分别取采样频率为 5000Hz 和 1000Hz 时，绘出其傅里叶变换图。

程序如下：

```
Dt = 0.00005;t =- 0.005:Dt:0.005;                    % 模拟信号
xa = exp( - 2000 * abs(t));
Ts = 0.0002;n =- 25:1:25;                            % 离散时间信号
x = exp( - 1000 * abs(n * Ts));
K = 500;k = 0:1:K;w = pi * k/K;
% 离散时间傅里叶变换
X = x * exp( - j * n' * w);X = real(X);
w = [ - fliplr(w),w(2:501)];
X = [fliplr(X),X(2:501)];
figure
subplot(2,2,1);
plot(t * 1000,xa,'.');
ylabel('x1(t)'); xlabel('t');
title ('离散信号');
hold on
stem(n * Ts * 1000,x);hold off
subplot(2,2,2);
plot(w/pi,X,'.');
ylabel('X1(jw)'); xlabel('f');
title('离散时间傅里叶变换');
Ts = 0.001;n =- 25:1:25;
% 离散时间信号
x = exp( - 1000 * abs(n * Ts));
K = 500;k = 0:1:K;w = pi * k/K;
% 离散时间傅里叶变换
X = x * exp( - j * n' * w);X = real(X);
w = [ - fliplr(w),w(2:501)];
X = [fliplr(X),X(2:501)];
subplot(2,2,3);
plot(t * 1000,xa,'.');
ylabel('x2(t)'); xlabel('t');
title ('离散信号');
```

```
hold on
stem(n * Ts * 1000,x);hold off
subplot(2,2,4);
plot(w/pi,X,'.');
ylabel('X2(jw)'); xlabel('f');
title('离散时间傅里叶变换');
```

运行程序结果如图 3-9 所示。

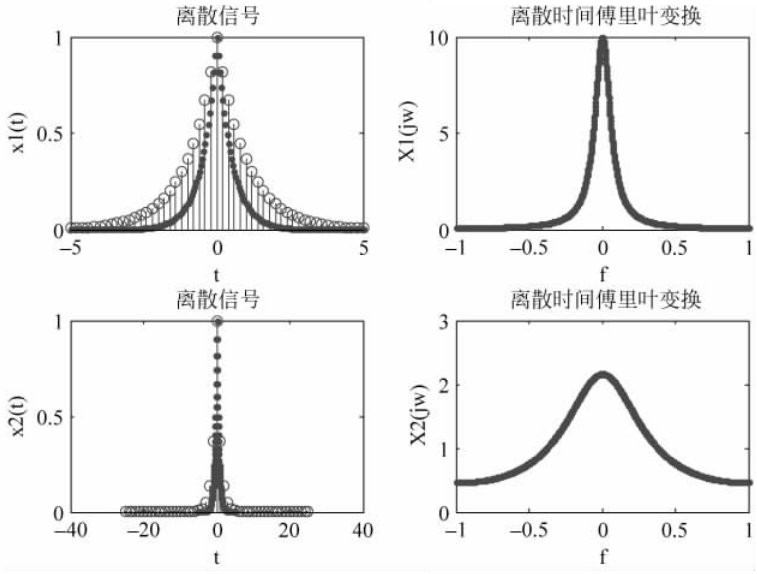

图 3-9　取样频率为 5000Hz 和 1000Hz 时傅里叶变换

【例 3-12】 对上例中产生的离散序列 $x_1(n)$和 $x_2(n)$,采用 sinc 函数进行内插重构。程序如下：

```
Ts1 = 0.0002;Fs1 = 1/Ts1;n1 =- 25:1:25;nTs1 = n1 * Ts1;                    % 离散时间信号
x1 = exp( - 1000 * abs(nTs1));
Ts2 = 0.001;
Fs2 = 1/Ts2;
n2 =- 5:1:5;
nTs2 = n2 * Ts2;
x2 = exp( - 2000 * abs(nTs2));
Dt = 0.00005;t =- 0.005:Dt:0.005;                                          % 模拟信号重构
xa1 = x1 * sinc(Fs1 * (ones(length(nTs1),1) * t - nTs1' * ones(1,length(t))));
xa2 = x2 * sinc(Fs2 * (ones(length(nTs2),1) * t - nTs2' * ones(1,length(t))));
subplot(2,1,1);
plot(t * 1000,xa1,'.');
ylabel('x1(t)'); xlabel('t');
title('从 x1(n)重构模拟信号 x1(t)');
hold on
stem(n1 * Ts1 * 1000,x1);
hold off
```

```
subplot(2,1,2);
plot(t * 1000,xa2,'.');
ylabel('x2(t)'); xlabel('t');
title('从 x2(n)重构模拟信号 x2(t)');
hold on
stem(n2 * Ts2 * 1000,x2);
hold off
```

运行程序结果如图 3-10 所示。

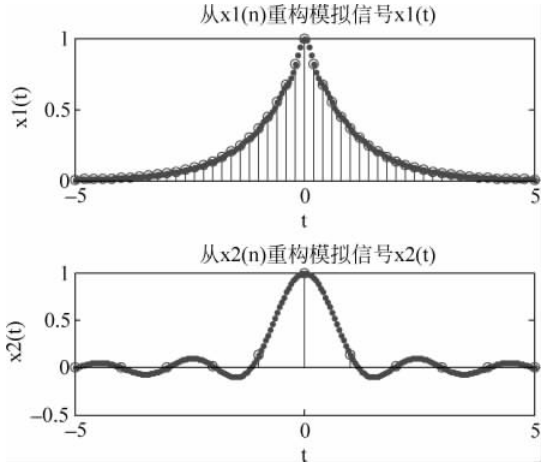

图 3-10　重构模拟信号效果图

3.10　快速傅里叶变换

快速傅里叶变换是傅里叶变换的一种快速算法，简称 FFT，采用这种算法能大大减少计算离散傅里叶变换所需要的乘法次数，特别是被变换的抽样点数 N 越多，FFT 算法计算量的节省就越显著。

3.10.1　直接计算 DFT 的问题及改进途径

设 $x(n)$为一个 N 点复序列，其 N 点 DFT 序列 $X(k)$通常也是一个复序列。将 $x(n)$和周期复指数序列 W_N^{nk} 表示成实部和虚部的组合形式为

$$x(n) = \mathrm{Re}[x(n)] + \mathrm{jIm}[x(n)]$$

$$W_N^{nk} = \mathrm{Re}[W_N^{nk}] + \mathrm{jIm}[W_N^{nk}] = \cos\left(\frac{2\pi}{N}nk\right) - \mathrm{jsin}\left(\frac{2\pi}{N}nk\right)$$

则 DFT 的定义式可表示为

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

$$= \sum_{n=0}^{N-1}\left\{\mathrm{Re}[x(n)]\cos\left(\frac{2\pi}{N}nk\right)+\mathrm{Im}[x(n)]\sin\left(\frac{2\pi}{N}nk\right)\right\}$$

$$+\mathrm{j}\left\{\mathrm{Im}[x(n)]\cos\left(\frac{2\pi}{N}nk\right)-\mathrm{Re}[x(n)]\sin\left(\frac{2\pi}{N}nk\right)\right\} \quad (0\leqslant k\leqslant N-1)$$

易知，一次复数乘法相当于 4 次实数乘法和 2 次实数加法，而一次复数加法相当于两次实数加法。因此，每计算一个 $X(k)$需要 $4N$ 次实数乘法和 $2N+2(N-1)=2(2N-1)$次实数加法。DFT 计算共需要计算 N 个 $X(k)$，因此，完成整个 DFT 计算共需要 $4N$ 次实数乘法和 $2N(2N-1)=4N2-2N$ 次实数加法。可见，直接计算 N 点 DFT 所需的计算量是和 N 成正比的，当 N 非常大时，计算量将显著增加。

考察 DFT 与 IDFT 的运算发现，利用以下两个特性可减少运算量：

(1) 系数 $w_N^{nk}=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}nk}$是一个周期函数，它的周期性和对称性可用来改进运算，提高计算效率。

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

$$w_N^{N/2}=-1,\quad w_N^{(k+N/2)}=-w_N^k$$

利用这些周期性和对称性，使 DFT 运算中有些项可合并。

(2) 利用 w_N^{nk} 的周期性和对称性，把长度为 N 点的大点数的 DFT 运算依次分解为若干个小点数的 DFT。因为 DFT 的计算量正比于 N^2，N 越小，计算量也就越小。FFT 算法正是基于这样的基本思想发展起来的，它有多种形式，但基本上可分为两类：时间抽取法和频率抽取法。

3.10.2 基 2 时分的 FFT 算法

假定 N 是 2 的整数次方，首先将序列 $x(n)$分解为两组，一组为偶数项，一组为奇数项，即

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

将 DFT 运算也相应分为两组，即

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

$$=\sum_{\text{偶数}n=0}^{N-2}x(n)w_N^{nk}+\sum_{\text{奇数}n=1}^{N-1}x(n)w_N^{nk}$$

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

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

根据对称性可知

$$w_N^{2n}=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}2n}=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N/2}n}=w_{N/2}^n$$

因此有

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

其中，$G(k)=\sum_{r=0}^{N/2-1}x(2r)W_{\frac{N}{2}}^{rk}$，$H(k)=\sum_{r=0}^{N/2-1}x(2r+1)W_{\frac{N}{2}}^{rk}$。注意，$H(k)$，$G(k)$有 $N/2$ 个点，即 $k=0,1,\cdots,N/2-1$，还必须应用系数 w_N^{nk} 的周期性和对称性。由对称性知

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

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

那么

$$X\left(k+\frac{N}{2}\right)=G(k)-W_N^kH(k),\quad k=0,1,\cdots,\frac{N}{2}-1$$

可见，一个 N 点的 DFT 被分解为两个 $N/2$ 点的 DFT，这两个 $N/2$ 点的 DFT 再合成为一个 N 点 DFT，即

$$X\left(k+\frac{N}{2}\right)=G(k)-W_N^kH(k),\quad k=0,1,\cdots,\frac{N}{2}-1$$

$$X\left(k+\frac{N}{2}\right)=G(k)-W_N^kH(k),\quad k=0,1,\cdots,\frac{N}{2}-1$$

以此类推，可以继续分下去，这种按时间抽取算法是在输入序列分成越来越小的子序列上执行 DFT 运算，最后再合成为 N 点的 DFT。

对于 $N=2^M$，总是可以通过 M 次分解最后成为 2 点的 DFT 运算，这样构成从 $x(n)$ 到 $X(k)$的 M 级运算过程。从上面的流图可看到，每一级运算都由 $N/2$ 个蝶形运算构成。

因此每一级运算都需要 $N/2$ 次复乘和 N 次复加，这样，经过时间抽取后，M 级运算总共需要的运算为$\frac{N}{2}\log_2N$ 次复数乘法和 $N\log_2N$ 复数加法。

3.10.3 基 2 频分的 FFT 算法

对于 $N=2^M$情况下的另外一种普遍使用的 FFT 结构是频率抽取法。

对于频率抽取法，输入序列不是按奇偶数分解，而是按前后对半分开，这样便将 N 点 DFT 写成前后两部分，即

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

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

进一步分解为偶数组和奇数组，即

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

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

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

令

$$b(n)=x(n)+x(n+N/2)$$

$$b(n)=[x(n)-x(n+N/2)]W_N^n$$

于是有

$$X(2r)=\sum_{n=0}^{N/2-1}a(n)W_{N/2}^{nr}$$

$$X(2r+1)=\sum_{n=0}^{N/2-1}b_2(n)W_{N/2}^{nr}$$

这正是两个 $N/2$ 点的 DFT 运算，即将一个 N 点的 DFT 分解为两个 $N/2$ 点的 DFT。与时间抽取法一样，由于 $N=2^M$，$N/2$ 仍是一个偶数，这样，一个 $N=2^M$ 点的 DFT 通过 M 次分解后，最后只剩下全部是 2 点的 DFT，2 点 DFT 实际上只有加减运算。

3.10.4 快速傅里叶变换的 MATLAB 实现

在 MATLAB 中，函数 fft 用于快速计算 DFT，其调用格式为

```
y = fft(x)
```

在此格式中，x 是取样的样本，可以是一个向量，也可以是一个矩阵，y 是 x 的快速傅里叶变换。在实际操作中，会对 x 进行补零操作，使 x 的长度等于 2 的整数次幂，这样能提高程序的计算速度。

```
y = fft(x,n)
```

通过改变 n 值来直接对样本进行补零或者截断的操作。

ifft 函数是用来计算序列的逆傅里叶变换，MATLAB 信号处理工具箱中提供的快速傅里叶反变换的调用格式为

```
y = ifft(X), y = ifft(X,n)
```

在此格式中，x 为需要进行逆变换的信号，多数情况下是复数，y 为快速傅里叶反变换的输出。

【例 3-13】 已知信号 $x=0.5\sin(2\pi\times20\times t)+2\sin(2\pi\times60\times t)$，其中 $f_1=20$Hz，$f_2=60$Hz，采样频率为 100Hz，绘制 $y(t)$经过快速傅里叶变换后的频谱图。

程序如下：

```
clear all;
fs = 100;                                                    % 采样频率
Ndata = 32;                                                  % 数据长度
N = 32;                                                      % FFT 的数据长度
n = 0:Ndata - 1;
t = n/fs;                                                    % 数据对应的时间序列
x = 0.5 * sin(2 * pi * 20 * t) + 2 * sin(2 * pi * 60 * t);   % 时间域信号
y = fft(x,N);                                                % 信号的傅里叶变换
mag = abs(y);                                                % 求取振幅
f = (0:N - 1) * fs/N;                                        % 真实频率
subplot(2,2,1);plot(f(1:N/2),mag(1:N/2) * 2/N);
xlabel('频率/Hz');ylabel('振幅');
title('Ndata = 32; FFT 所以采点个数 = 32');
grid on;

Ndata = 32;                                                  % 数据长度
N = 128;                                                     % FFT 采用的数据长度
n = 0:Ndata - 1;
t = n/fs;                                                    % 时间序列
x = 0.5 * sin(2 * pi * 20 * t) + 2 * sin(2 * pi * 60 * t);   % 时间域信号
y = fft(x,N);
mag = abs(y);
f = (0:N - 1) * fs/N;                                        % 真实频率
subplot(2,2,2);plot(f(1:N/2),mag(1:N/2) * 2/N);
xlabel('频率/Hz');ylabel('振幅');
title('Ndata = 32; FFT 所以采点个数 = 128');
grid on;

Ndata = 136;                                                 % 数据长度
N = 128;                                                     % FFT 采用的数据长度
n = 0:Ndata - 1;
t = n/fs;                                                    % 时间序列
x = 0.5 * sin(2 * pi * 20 * t) + 2 * sin(2 * pi * 60 * t);   % 时间域信号
y = fft(x,N);
mag = abs(y);
f = (0:N - 1) * fs/N;                                        % 真实频率
subplot(2,2,3);plot(f(1:N/2),mag(1:N/2) * 2/N);
xlabel('频率/Hz');ylabel('振幅');
title('Ndata = 136; FFT 所以采点个数 = 128');
grid on;

Ndata = 136;                                                 % 数据长度
N = 512;                                                     % FFT 采用的数据长度
n = 0:Ndata - 1;
t = n/fs;                                                    % 时间序列
x = 0.5 * sin(2 * pi * 20 * t) + 2 * sin(2 * pi * 60 * t);   % 时间域信号
y = fft(x,N);
```

```
mag = abs(y);
f = (0:N - 1) * fs/N;                                    % 真实频率
subplot(2,2,4);plot(f(1:N/2),mag(1:N/2) * 2/N);
xlabel('频率/Hz');ylabel('振幅');
title('Ndata = 136; FFT 所以采点个数 = 512');
grid on;
```

运行结果如图 3-11 所示。

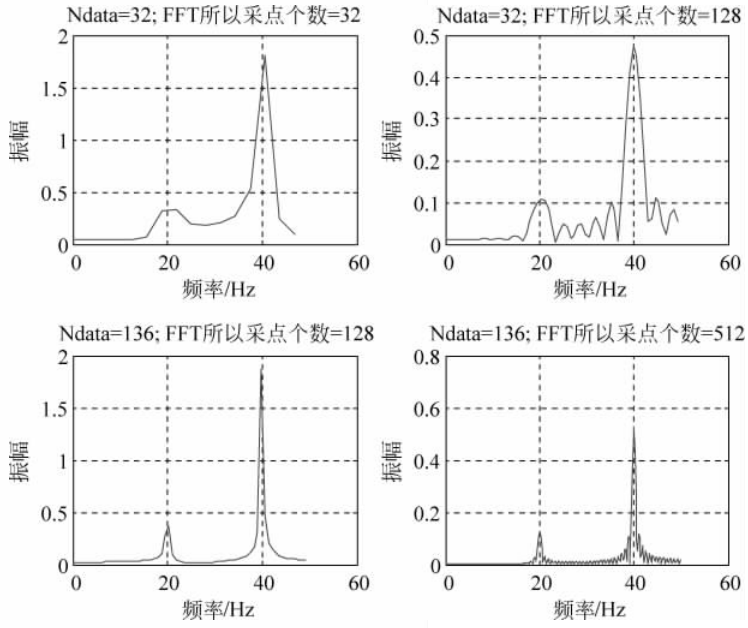

图 3-11　改变后对傅里叶谱的影响效果图

【例 3-14】 设 $x(n)$为一个正弦信号、一个余弦信号及白噪声的叠加信号，试用 FFT 文件对其作频谱分析。

程序如下：

```
N = 1024;
f1 = .1;f2 = .2;fs = 1;
a1 = 5;a2 = 3;
w = 2 * pi/fs;
x = a1 * sin(w * f1 * (0:N - 1)) + a2 * cos(w * f2 * (0:N - 1)) + randn(1,N);
 % 应用 FFT 求频谱;
subplot(2,1,1);
plot(x(1:N/4));
title('原始信号');
f = - 0.5:1/N:0.5 - 1/N;
X = fft(x);
subplot(2,1,2);
plot(f,fftshift(abs(X)));
title('频域信号');
```

运行结果如图 3-12 所示。

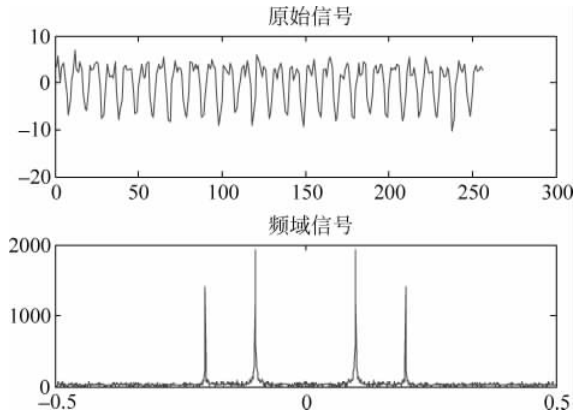

图 3-12　频谱分析图

【例 3-15】 逆 FFT 示例。

程序如下：

```
clear all;
fs = 200;                                                    % 采样频率
N = 128;                                                     % 数据个数
n = 0:N - 1;
t = n/fs;                                                    % 数据对应的时间序列
x = 0.5 * sin(2 * pi * 20 * t) + 2 * sin(2 * pi * 60 * t);   % 时间域信号
subplot(2,2,1);plot(t,x);
xlabel('时间/s');ylabel('x');
title('原始信号');
grid on;

y = fft(x,N);                                                % 傅里叶变换
mag = abs(y);                                                % 得到振幅谱
f = n * fs/N;                                                % 频率序列
subplot(2,2,2);plot(f(1:N/2),mag(1:N/2) * 2/N);
xlabel('频率/Hz');ylabel('振幅');
title('原始信号的 FFT');
grid on;

xifft = ifft(y);                                             % 进行傅里叶逆变换
realx = real(xifft);                                         % 求取傅里叶逆变换的实部
ti = [0:length(xifft) - 1]/fs;                               % 傅里叶逆变换的时间序列
subplot(2,2,3);plot(ti,realx);
xlabel('时间/s');ylabel('x');
title('运用傅里叶逆变换得到的信号');
grid on;

yif = fft(xifft,N);                          % 将傅里叶逆变换得到的时间域信号进行傅里叶变换
mag = abs(yif);
f = [0:length(y) - 1]' * fs/length(y);                       % 频率序列
subplot(2,2,4);plot(f(1:N/2),mag(1:N/2) * 2/N);
```

```
xlabel('频率/Hz');ylabel('振幅');
title('运用 IFFT 得到信号的快速傅里叶变换');
grid on;
```

运行结果如图 3-13 所示。

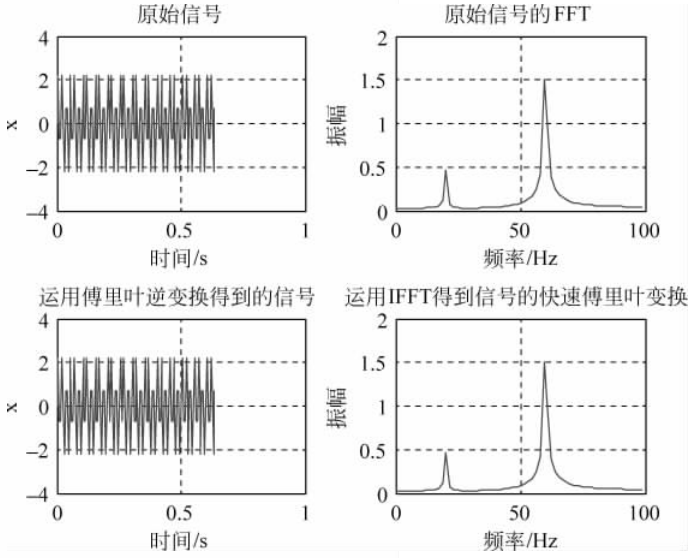

图 3-13　FFT 与 IFFT 对比效果图

【例 3-16】 利用 FFT 进行滤波示例。利用 FFT 对信号 x＝0.5＊cos(2＊pi＊f1＊t)＋sin(2＊pi＊f2＊t)＋randn(1,N)进行滤波,将频率为 4～8Hz 的波滤除掉。

程序如下:

```
clear all;
dt = 0.05;N = 1024;
n = 0:N - 1; t = n * dt;                                    %时间序列
f = n/(N * dt);                                             %频率序列
f1 = 3; f2 = 10;                                            %信号的频率成分
x = 0.5 * cos(2 * pi * f1 * t) + sin(2 * pi * f2 * t) + randn(1,N);
subplot(2,2,1);plot(t,x);                                   %绘制原始的信号
title('原始信号的时间域');xlabel('时间/s');
y = fft(x);                                                 %对原信号作 FFT 变换
xlim([0 12]);ylim([ - 1.5 1.5]);
subplot(2,2,2);plot(f,abs(y) * 2/N);                        %绘制原始信号的振幅谱
xlabel('频率/Hz');ylabel('振幅');
xlim([0 50]);title('原始振幅谱');
ylim([0 0.8]);
f1 = 4;f2 = 8;                                              %要滤去频率的上限和下限
yy = zeros(1,length(y));                                    %设置与 y 相同的元素数组
for m = 0:N - 1                          %将频率落在该频率范围及其大于 Nyquist 频率的波滤去
    % 小于 Nyquist 频率的滤波范围
    if (m/(N * dt)> f1 & m/(N * dt)< f2) | (m/(N * dt)>(1/dt - f2) & m/(N/dt)<(1/dt - f1));
    % 大于 Nyquist 频率的滤波范围
```

```
    % 1/dt 为一个频率周期
    yy(m+1)=0;                          %置在此频率范围内的振动振幅为零
    else
        yy(m+1)=y(m+1);                 %其余频率范围的振动振幅不变
    end
end
subplot(2,2,4);plot(f,abs(yy)*2/N)      %绘制滤波后的振幅谱
xlim([0 50]);ylim([0 0.5]);
xlabel('频率/Hz');ylabel('振幅');
gstext=sprintf('自%4.1f-%4.1fHz的频率被滤除',f1,f2);
%将滤波范围显示作为标题
title(gstext);
subplot(2,2,3);plot(t,real(ifft(yy)));
%绘制滤波后的数据运用 ifft 变换回时间域并绘图
title('通过 IFFT 回到时间域');
xlabel('时间/s');
ylim([-0.6 0.6]);xlim([0 12]);
```

运行结果如图 3-14 所示。

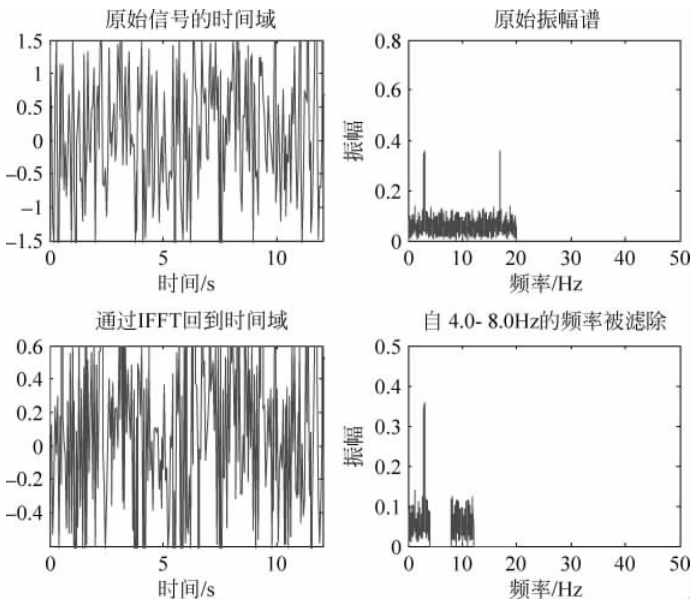

图 3-14　利用 FFT 进行滤波示例效果

3.11　离散余弦变换

离散余弦变换(Discrete Cosine Transform,DCT)是一种与傅里叶变换紧密相关的数学运算。在傅里叶级数展开式中,如果被展开的函数是实偶函数,那么其傅里叶级数中只包含余弦项,再将其离散化可导出余弦变换,因此称之为离散余弦变换。

3.11.1 一维离散余弦变换

$f(x)$为一维离散函数，$x=0,1,\cdots,N-1$，进行离散变换后，有

$$F(u)=\sqrt{\frac{2}{N}}\sum_{x=0}^{N-1}f(x)\cos\left[\frac{\pi}{2N}(2x+1)u\right]\qquad u=1,2,\cdots,N-1$$

$$F(0)=\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}f(x)\qquad u=0$$

其反变换为

$$f(x)=\frac{1}{\sqrt{N}}F(0)+\sqrt{\frac{2}{N}}\sum_{u=1}^{N-1}F(u)\cos\left[\frac{\pi}{2N}(2x+1)u\right]\qquad x=0,1,\cdots,N-1$$

其中，$g(x,0)=\frac{1}{\sqrt{N}}$，$g(x,u)=\sqrt{\frac{2}{N}}\cos\frac{(2x+1)u\pi}{2N}$

3.11.2 二维离散余弦变换

$f(m,n)$为二维离散函数 $m,n=0,1,2,\cdots,N-1$，进行离散变换后，有

$$F(u,v)=a(u)a(v)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}f(m,n)\cos\frac{(2m+1)u\pi}{2N}\cos\frac{(2n+1)v\pi}{2N}$$

其反变换为

$$f(m,n)=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}a(u)a(v)F(u,v)\cos\frac{(2m+1)u\pi}{2N}\cos\frac{(2n+1)v\pi}{2N}$$

其中，$m,n=0,1,2,\cdots,N-1$，$a(u)=a(v)=\begin{cases}\sqrt{\frac{1}{N}} & u=0\text{ 或 }v=0\\ \sqrt{\frac{2}{N}} & u,v=1,2,\cdots,N-1\end{cases}$

3.11.3 离散余弦函数

在 MATLAB 中，dct 函数用于进行 DCT 变换，该函数的调用方法如下：

y=dct(x)：返回序列 x 的 DCT 结果。

dict2 函数用于 DCT 反变换，该函数调用方法为

B=idct2(A)：计算 A 的 DCT 反变换 B，A 与 B 的大小相同。

【例 3-17】 对信号进行离散余弦变换示例。

程序如下：

```
clear all;
clc;
close all;
```

```
n = 1:100;
x = 10 * sin(2 * pi * n/20) + 20 * cos(2 * pi * n/30);
y = dct(x);
subplot(1,2,1),plot(x),title('原始信号');
subplot(1,2,2),plot(y),title('DCT 效果');
```

运行结果如图 3-15 所示。

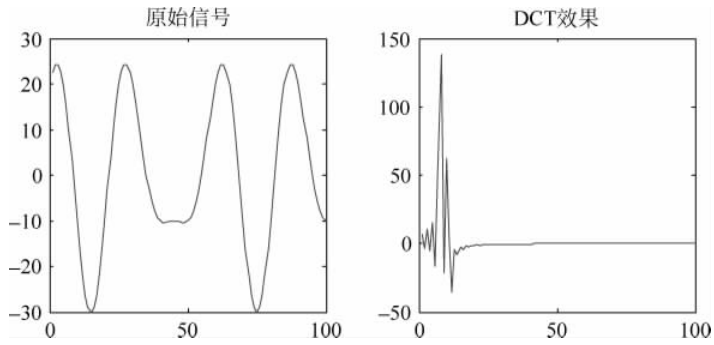

图 3-15　对信号进行离散余弦变换效果图

【例 3-18】 DCT 反变换示例。

程序如下：

```
clear all;
n = 0:200 - 1;
f = 200;   fs = 3000;
x = cos(2 * pi * n * f/fs);
y = dct(x);                                  % 计算 DCT 变换
m = find(abs(y < 5));                        % 利用阈值对变换系数截取
y(m) = zeros(size(m));
z = idct(y);                                 % 对门限处理后的系数 DCT 反变换
subplot(1,2,1);plot(n,x);
xlabel('n');title('序列 x(n)');
subplot(1,2,2);plot(n,z);
xlabel('n');title('序列 z(n)');
```

运行结果如图 3-16 所示。

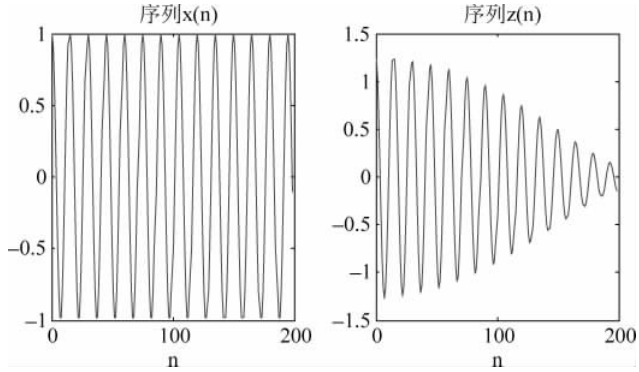

图 3-16　原始与重建后的对比图

3.12 Chirp Z 变换

序列的 Z 变换公式为

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

将 $z_k = AW^{-k}$ 代入可以得到

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

定义 $g(n) = x(n)A^{-n}W^{\frac{n^2}{2}}$，$h(n) = W^{-\frac{n^2}{2}}$，那么有

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

以上运算转换为卷积形式，从而可采用 FFT 进行，这样可大大提高计算速度。系统的单位冲激响应 $h(n) = W^{-\frac{n^2}{2}}$ 与频率随时间成线性增加的线性调频信号相似，因此称为 Chirp-Z 变换。

在 MATLAB 中，czt 函数用于实现 Chirp-Z 变换，该函数的调用方法如下：

```
Y = czt(x,m,w,a)
```

此函数计算由 z=a * w.^(-(0:m-1))定义的 z 平面螺旋线上各点的 Z 变换，a 规定了起点，w 规定了相邻点的比例，m 规定了变换的长度，后 3 个变量默认值为 a=1，w=exp(j * 2 * pi/m)及 m=length(x)，因此 y=czt(x)就等于 y=fft(x)。

【例 3-19】 MATLAB 的 czt 函数实现频率细化。

```
fs = 256;                                          % 采样频率
N = 512;                                           % 采样点数
nfft = 512;
n = 0:1:N - 1;                                     % 时间序列号
% n/fs:采样频率下对应的时间序列值
n1 = fs * (0:nfft/2 - 1)/nfft;                     % F F T 对应的频率序列
x = 3 * sin(2 * pi * 100 * n/fs) + 3 * cos(2 * pi * 101.45 * n/fs) + 2 * sin(2 * pi * 102.3 * n/fs)
 + 4 * cos(2 * pi * 103.8 * n/fs) + 5 * sin(2 * pi * 104.5 * n/fs);
figure;
subplot(231);
plot(n,x);
xlabel('时间 t');
ylabel('value');
title('信号的时域波形');

XK = fft(x,nfft);                                  % 单边幅值谱
subplot(232);stem(n1,abs(XK(1:(nfft/2))));  % 用杆状来画 FFT 的图
axis([95,110,0,1500]);
title('直接利用 FFT 变换后的频谱');
subplot(233);plot(n1,abs(XK(1:(N/2))));
```

```
axis([95,110,0,1500]);
title('直接利用 FFT 变换后的频谱');

f1 = 100;                                    % 细化频率段起点
f2 = 110;                                    % 细化频率段终点
M = 256;                                     % 细化频段的频点数(这里其实就是细化精度)
w = exp( - j * 2 * pi * (f2 - f1)/(fs * M)); % 细化频段的跨度(步长)
a = exp(j * 2 * pi * f1/fs);          % 细化频段的起始点,这里需要运算一下才能代入 czt 函数
xk = czt(x,M,w,a);
h = 0:1:M - 1;                               % 细化频点序列
f0 = (f2 - f1)/M * h + 100;                  % 细化的频率值
subplot(234);stem(f0,abs(xk));
xlabel('f');
ylabel('value');
title('利用 CZT 变换后的细化频谱');
subplot(235);plot(f0,abs(xk));
xlabel('f');
ylabel('value');
title('利用 CZT 变换后的细化频谱');
```

运行结果如图 3-17 所示。

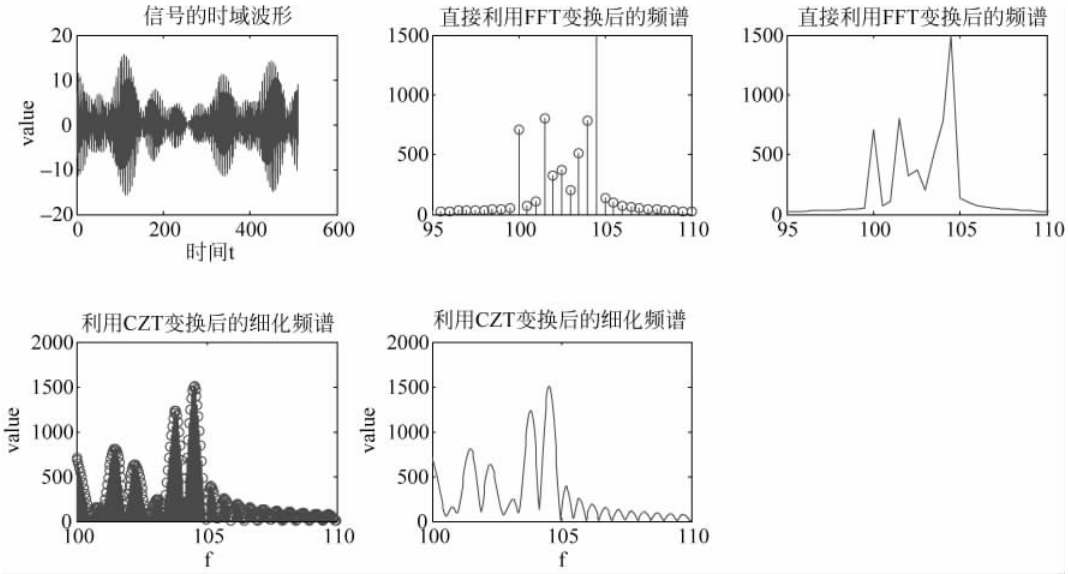

图 3-17　MATLAB 的 czt 函数实现频率细化效果图

【例 3-20】 利用 Chirp-Z 变换计算滤波器在 100～200Hz 的频率特性，并比较 czt 和 fft 函数。

程序如下：

```
h = fir1(30,125/500,boxcar(31));
Fs = 1000;
f1 = 100;
f2 = 200;
```

```
m = 1024;
w = exp( - j * 2 * pi * (f2 - 1)/(m * Fs));
a = exp(j * 2 * pi * f1/Fs);
y = fft(h,m);
z = czt(h,m,w,a);
fy = (0:length(y) - 1)' * Fs/length(y);
fz = (0:length(z) - 1)' * (f2 - f1)/length(z) + f1;
subplot(2,1,1)
plot(fy(1:500),abs(y(1:500)));
title('fft');
subplot(2,1,2)
plot(fz,abs(z));
title('czt');
```

运行结果如图 3-18 所示。

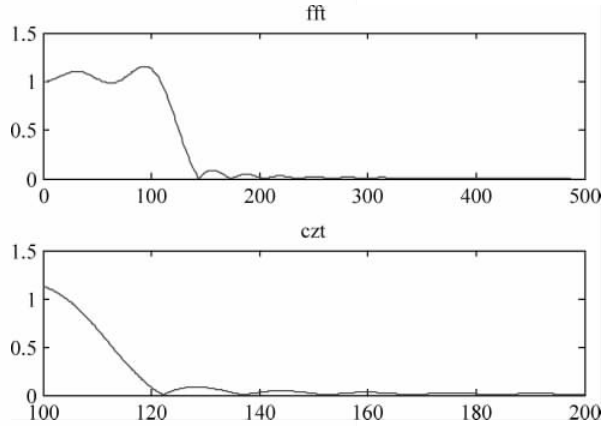

图 3-18　利用 Chirp-Z 变换计算在 100～200Hz 的频率特性效果图

3.13　Gabor 函数

Gabor 变换属于加窗傅里叶变换，Gabor 函数可以在频域不同尺度、不同方向上提取相关的特征。另外 Gabor 函数与人眼的生物作用相仿，所以经常用作纹理识别上，并取得了较好的效果。

3.13.1　Gabor 函数定义

Gabor 变换是 D. Gabor 1946 年提出的。由于经典 Fourier 变换只能反映信号的整体特性(时域、频域)。另外，要求信号满足平稳条件。如果用傅里叶变换研究频域信息，必须知道信号时域上的信息，另外，信号在某时刻的一个小的邻域内发生变化，那么信号的整个频谱都要受到影响，而频谱的变化从根本上来说无法标定发生变化的时间位置和发生变化的剧烈程度。也就是说，Fourier 变换对信号的齐性不敏感，不能给出在各个局

部时间范围内部频谱上的谱信息描述。

3.13.2 Gabor 函数的一般求法与解析理论

可根据实际需要选取适当的核函数，如高斯窗函数

$$g(t) = \left(\frac{\sqrt{2}}{T}\right)^2 \mathrm{e}^{-\pi\left(\frac{t}{T}\right)^2}$$

则其对偶函数 $\gamma(t)$ 为

$$\gamma(t) = \left(\frac{1}{\sqrt{2}T}\right)^{\frac{1}{2}} \left(\frac{K_0}{\pi}\right)^{\frac{-3}{2}} \mathrm{e}^{\pi\left(\frac{t}{T}\right)^2} \sum_{n+\frac{1}{2}>\frac{1}{T}} (-1)^n \mathrm{e}^{-\pi\left(n+\frac{1}{2}\right)^2}$$

离散 Gabor 变换的表达式为

$$G_{mn} = \int_{-\infty}^{\infty} \phi(t) g^*(t-mT)\mathrm{e}^{-\mathrm{j}n\omega t}\mathrm{d}t = \int_{-\infty}^{\infty} \phi(t) g_{mn}^*(t)\mathrm{d}t$$

$$\phi(t) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} G_{mn}\gamma(t-mT)\mathrm{e}^{\mathrm{j}n\omega t} = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} G_{mn}\gamma_{mn}(t)$$

其中，$g_{mn}(t)=g(t-mT)\mathrm{e}^{\mathrm{j}n\omega t}$，$\gamma(t)$ 是 $g(t)$ 的对偶函数，二者之间有如下双正交关系：

$$\int_{-\infty}^{\infty} \gamma(t) g^*(t-mT)\mathrm{e}^{-\mathrm{j}n\omega t}\mathrm{d}t = \delta_m\delta_n$$

Gabor 变换的解析理论就是由 $g(t)$ 求对偶函数 $\gamma(t)$ 的方法。

定义 $g(t)$ 的 Zak 变换为

$$\mathrm{Zak}[g(t)] = \hat{g}(t,\omega) = \sum_{k=-\infty}^{\infty} g(t-k)\mathrm{e}^{-\mathrm{j}2\pi k\omega}$$

可以证明对偶函数可由下式求出：

$$\gamma(t) = \int_0^1 \frac{\mathrm{d}\omega}{g^*(t,\omega)}$$

有了对偶函数可以使计算更为简洁方便。

【例 3-21】 用 Gabor 函数分析 δ 双时间信号。

程序如下：

```
clear all
a = 1/10;
m = 2;                    %设定窗口尺度和超高斯函数阶数
t1 = 5;t2 = 6;
%设定双信号的位置
%绘制双信号的三维网格立体图
[t,W] = meshgrid([2:0.2:7],[0:pi/6:3*pi]);
%设置时-频相平面网格点
Gs1 = (1/(sqrt(2*pi)*a))*exp(-0.5*abs((t1-t)/a).^m).*exp(-i*W*t1);
Gs2 = (1/(sqrt(2*pi)*a))*exp(-0.5*abs((t2-t)/a).^m).*exp(-i*W*t2);
Gs = Gs1 + Gs2;
subplot(2,3,1);
%绘制实部三维网格立体图
```

```
mesh(t,W/pi,real(Gs));
axis([2 7 0 3 -1/(sqrt(2*pi)*a) 1/(sqrt(2*pi)*a)]);
title('实部')
xlabel('t(s)'); ylabel('real(Gs)');
subplot(2,3,2);
%绘制虚部三维网格立体图
mesh(t,W/pi,imag(Gs));
axis([2 7 0 3 -1/(sqrt(2*pi)*a) 1/(sqrt(2*pi)*a)]);
title('虚部')
xlabel('t(s)'); ylabel('imag(Gs)');
subplot(2,3,3);
%绘制绝对值三维网格立体图
mesh(t,W/pi,abs(Gs));
axis([2 7 0 3 -1/(sqrt(2*pi)*a) 1/(sqrt(2*pi)*a)]);
title('绝对值')
xlabel('t(s)'); ylabel('abs(Gs)');
%绘制双信号的二维灰度图
[t,W]=meshgrid([2:0.2:7],[0:pi/20:3*pi]);
%设置时频相平面网格点
Gs1=(1/(sqrt(2*pi)*a))*exp(-0.5*abs((t1-t)/a).^m).*exp(-i*W*t1);
Gs2=(1/(sqrt(2*pi)*a))*exp(-0.5*abs((t2-t)/a).^m).*exp(-i*W*t2);
Gs=Gs1+Gs2;
subplot(2,3,4);
ss=real(Gs);ma=max(max(ss));
%计算最大值
pcolor(t,W/pi,ma-ss);
title('实部最大值')
xlabel('t(s)'); ylabel('maxreal(Gs)');
colormap(gray(50));shading interp;
subplot(2,3,5);
ss=imag(Gs);ma=max(max(ss));
%计算最大值
pcolor(t,W/pi,ma-ss);
title('虚部最大值')
xlabel('t(s)'); ylabel('maximag(Gs)');
colormap(gray(50));shading interp;
subplot(2,3,6);
ss=abs(Gs);ma=max(max(ss));
%计算绝对值的最大值
pcolor(t,W/pi,ma-ss);
title('绝对值最大值')
xlabel('t(s)'); ylabel('maxabs(Gs)');
colormap(gray(50));
shading interp;
```

运行结果如图 3-19 所示。

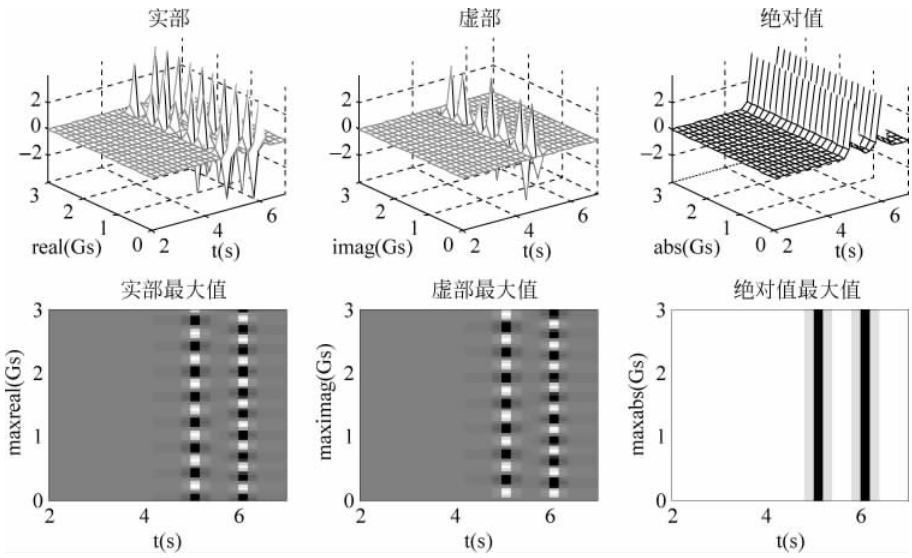

图 3-19　Gabor 变换

3.13.3　Gabor 展开

用过采样的 Gabor 展开来检测瞬时信号，效果要比传统的方法好。Gabor 展开固有的局部化特性，使它特别适合于描述瞬时信号，可选择单边指数窗作为 Gabor 展开的窗函数，与瞬时信号的非对称性及突变性相适应。

利用 Gabor 展开，得到观测信号展开后的系数，就可以用其系数来检测瞬时信号的存在。

【例 3-22】 利用 Gabor 展开检测信号的频谱。

程序如下：

```
clear;
t = 40;
fs = 10000;
f1 = 2000;
f2 = 4000;
le = fs * t;
a = 1/16;b = 16;
N = a * fs;M = fs/b;
s = fix(le/fs);
% 帧间重叠 1/2,算出所需循环次数
hn = boxcar(M)';
% 得到 STFT 分析窗,汉宁窗,帧长 256 点
T = 1:fs * t;
d = sin(2 * pi * f1 * T/fs) + sin(2 * pi * f2 * T/fs);
for n = 1:1:s
    d1(1:M) = d((n - 1) * N + 1:(n - 1) * N + M). * hn;
```

```
%时域加窗
    Xd(n,(1:M)) = fft(d1,M);   %FFT
end
[n,m] = size(Xd);
x = 1:n;y = 1:m;
mesh(y/m,x,abs(Xd));
axis([0,0.5,0,20,0,100])
xlabel('t');
ylabel('f ')
zlabel('幅度');
```

运行结果如图 3-20 所示。

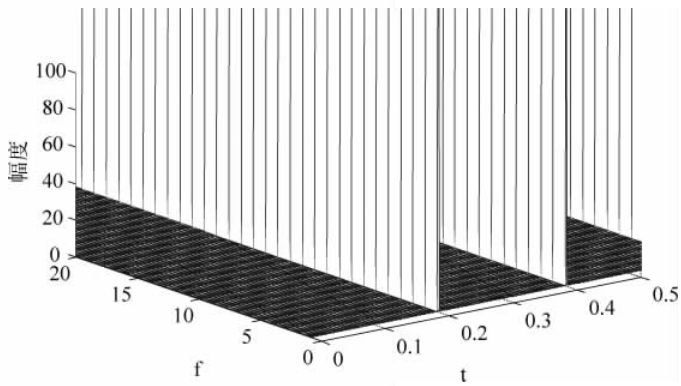

图 3-20　Gabor 分频图

【例 3-23】 Gabor 的 MATLAB 实现。

程序如下：

```
clf;
clear all;
close all;
fs = 100;
%采样率
Ts = 1/fs;
t = 0:Ts:10;
gass = 2^(1/4) * exp( - pi * (t).^2). * cos(5 * pi * t);
%生成一个高斯函数
subplot(231);
plot(t,gass);
title('高斯函数');
xlabel('t');
ylabel('幅度');
grid on;
T = 0:Ts:10;
ft = sin(T.^2 + 2 * T) + sin(T.^2);
%生成要变换的信号函数
subplot(232);
```

```
plot(t,ft);
title('信号函数');
grid on;
xlabel('t');
ylabel('幅度');
y = fft(ft);
% 信号做 FFT 变换
amp = abs(y);
grid on;
subplot(233);
plot(amp);
title('信号的 FFT 变换');
xlabel('f');
ylabel('幅度');
grid on;
subplot(234);
plot(t,imag(hilbert(ft)));
title('信号的 HHT 变换');
grid on;
shl = 100;
% 高斯窗每次平移点数
shn = (length(t) - 1)/shl;
% 求高斯窗平移总次数
y2 = zeros(shn,2001);
for k = 0:shn - 1;
gassc = 2 ^(1/4) * exp( - pi * (t - k * shl * Ts).^2). * cos(5 * pi * t);
% 平移后的高斯函数
gassc2 = gassc/sum(gassc.^2)
% 归一化
yl = conv(hilbert(ft),gassc2);
% 短时傅里叶变换,即对信号与 Gauss 函数做卷积
    y2(k + 1,:) = yl;
end
[F,T] = size(y2);
[F,T] = meshgrid(1:T,1:F);
subplot(235);
mesh(F,T,abs(y2))
title('信号尺度分布图');
xlabel('t');
ylabel('f ')
zlabel('幅度');
subplot(236);
contour(F,T,abs(y2))
% 等高线图
title('信号时频图');

xlabel('F(Hz)');
ylabel('尺度')
```

运行结果如图 3-21 所示。

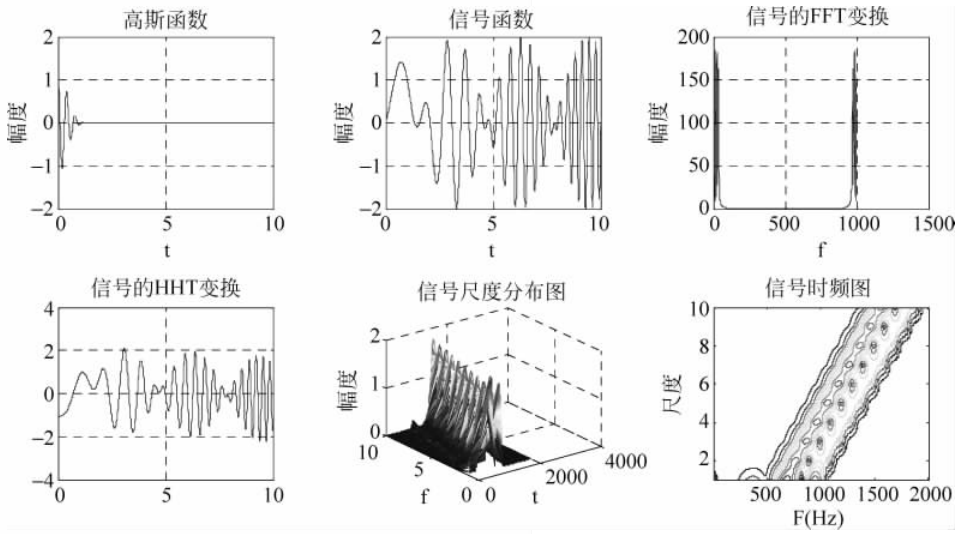

图 3-21　尺度分布图和时频图

本章小结

Z 变换可以说是针对离散信号与系统的拉普拉斯变换，在离散时间信号与系统的理论研究中，Z 变换是一种重要的数学工具，Z 变换的性质反映了信号的 n 域特性与 Z 域特性的关系。利用 Z 变换的一些基本性质可方便地求出一些常用离散信号的变换式，它可把离散系统的数学模型差分方程转化成简单的代数方程而使其求解过程简化。因此，Z 变换的基本性质对于变换分析法是非常重要的。

傅里叶的创造性工作为偏微分方程的边值问题提供了基本的求解方法——傅里叶级数法，从而极大地推动了微分方程理论的发展，特别是数学物理等应用数学的发展；其次，傅里叶级数拓广了函数概念，从而极大地推动了函数论的研究，其影响还涉及纯粹数学的其他领域。读者在学习过程中可以参考每个小节中的函数功能描述及相应的示例。