目录

实验 1 卷积及相关分析

实验目的：1、掌握卷积的概念及计算方法；

2、掌握相关的概念及计算方法；

3、掌握卷积与相关分析的性质及二者之间的关系；

4、掌握 Matlab 环境下求解卷积和相关的方法。

实验内容：1、理解卷积积分和相关分析的物理意义；

2、画出原始信号$f_1(t)$和$f_2(t)$信号的波形图；

$$f_1(t) = u(t) - u(t-2)$$

$$f_2(t) = (t+1)[u(t+1) - u(t)] + (1-t)[u(t) - u(t-1)]$$

3、在 Matlab 环境下编写程序计算$f_1(t) * f_2(t)$，r_{xy}及r_{yx}，r_{xx}，r_{yy}并对结果进行对比分析；

4、手动计算$f_1(t) * f_2(t)$，r_{xy}及r_{yx}，画出图形，与 3 结果进行对比。（选做）

实验要求：1、在 matlab 环境下编写和调试程序

2、保存程序的运行结果，并结合算法进行分析；结果图件要求有横、纵坐标及图注；

3、手动计算$f_1(t) * f_2(t)$，r_{xy}及r_{yx}，r_{xx}，r_{yy}，要有详细的计算过程；

4、按要求编写实验报告（内容包括：实验目的、实验原理、实验内容步骤，实验结果及分析、实验总结）

解：

（1） 卷积的概念及计算方法

卷积积分的物理意义：卷积积分表示两个信号之间的相互影响。在时间域中，卷积积分是通过将一个信号反转并移动另一个信号来计算它们之间的积分。因此，卷积积分可以看作是一个信号在另一个信号上的滑动平均。

对于连续函数$f(x), g(x)$，计算方法为：

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau$$

对于离散序列有：

$$(f * g)(n) = \sum_{m=-\infty}^{+\infty} f[m]g[n-m]$$

（2） 相关的概念及计算方法

相关分析的物理意义：相关分析用于测量两个信号之间的相似性或相关性。在时间域中，相关分析是通过将一个信号与另一个信号的时间翻转后相乘，然后对乘积进行积分来计算它们之间的相关性。

相关分为自相关和互相关计算方法类似，只是将两个函数或序列都设为相同的函数或序列

连续情况下的互相关：

$$R_{fg}(t) = \int_{-\infty}^{+\infty} f(\tau)g(t+\tau)d\tau$$

离散情况下的互相关：

$$R_{fg}[n] = \sum_{m=-\infty}^{+\infty} f[m]g[n+m]$$

（3） 卷积和相关的区别和联系

首先是计算公式的不同，其次是目的和应用：卷积和相关的应用目的略有不同。卷积常用于信号处理和图像处理中的滤波、卷积神经网络等任务，相关常用于信号匹配、图像匹配、目标跟踪等任务，通过衡量两个信号之间的相似性或关联性来进行匹配和分析

（4） 卷积和相关的性质

1. 交换性：$f(\tau) * g(\tau) = g(\tau) * f(\tau)$
2. 结合性：$[f(\tau) * g(\tau)] * h(\tau) = f(\tau) * [g(\tau) * h(\tau)]$
3. 分配性：$[f(\tau) + g(\tau)] * h(\tau)) = f(\tau)h(\tau) + g(\tau)h(\tau)$
4. 位移特性：$\big(f(\tau)g(\tau - t_0)\big)(t_0) = \big(f(\tau) * g(\tau)\big)(t - t_0)$
5. 微分性质：$\Big(\big(f(\tau) * g(\tau)\big)(t)\Big)' = (f(\tau) * g'(\tau))(t)$

对实验内容求解有：

（1）画出原始信号$f_1(t)$和$f_2(t)$信号的波形图；

$$f_1(t) = u(t) - u(t-2)$$

$$f_2(t) = (t+1)[u(t+1) - u(t)] + (1-t)[u(t) - u(t-1)]$$

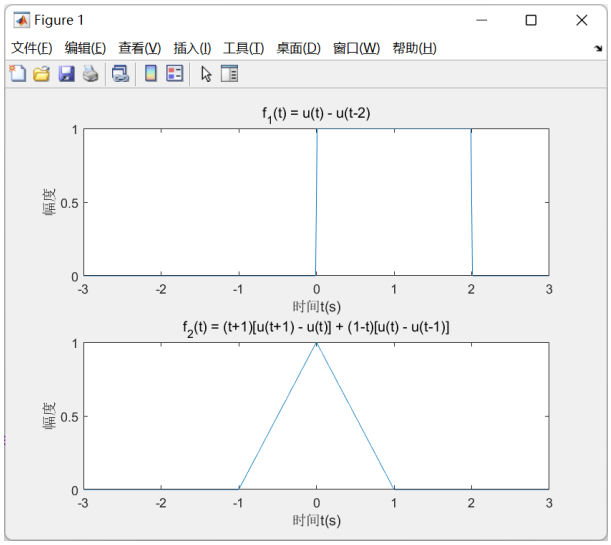

图 1.1 f_1, f_2信号波形图

代码如下：

```
% 定义时间轴 t
clear, clc, close all;
t = -3: 0.01: 3;
figure(1);
% f1
f1 = heaviside(t) - heaviside(t-2);
subplot(2, 1, 1);
plot(t,f1);
title('f_1(t) = u(t) - u(t-2)');
xlabel('时间 t(s)');
ylabel('幅度');

% f2
f2 = (t+1).* ( heaviside(t+1) - heaviside(t) ) + (1-t) .* (heaviside(t) -
heaviside(t-1) );
subplot(2, 1, 2);
plot(t,f2);
title('f_2(t) = (t+1)[u(t+1) - u(t)] + (1-t)[u(t) - u(t-1)]');
xlabel('时间 t(s)');
ylabel('幅度');
```

（2）在 Matlab 环境下编写程序计算$f_1(t) * f_2(t)$，r_{xy}及r_{yx}，r_{xx}，r_{yy}并对结果进行对比分析

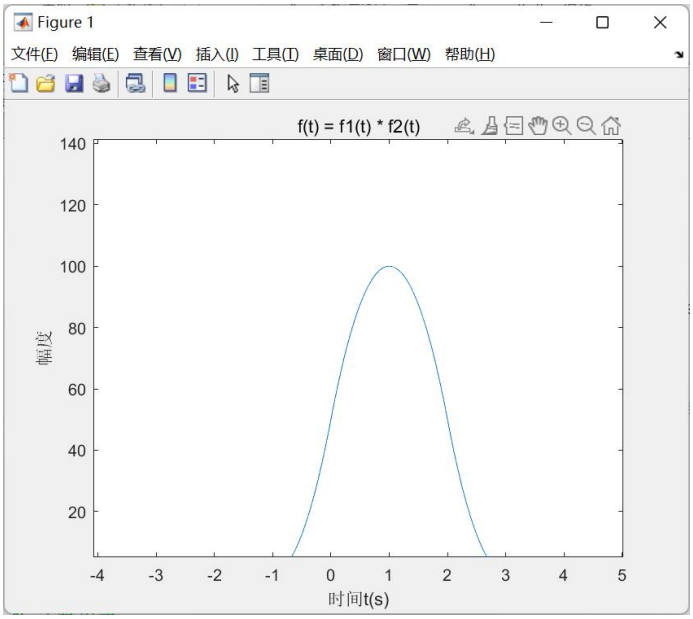

图 1.2 $f_1(t) * f_2(t)$图像

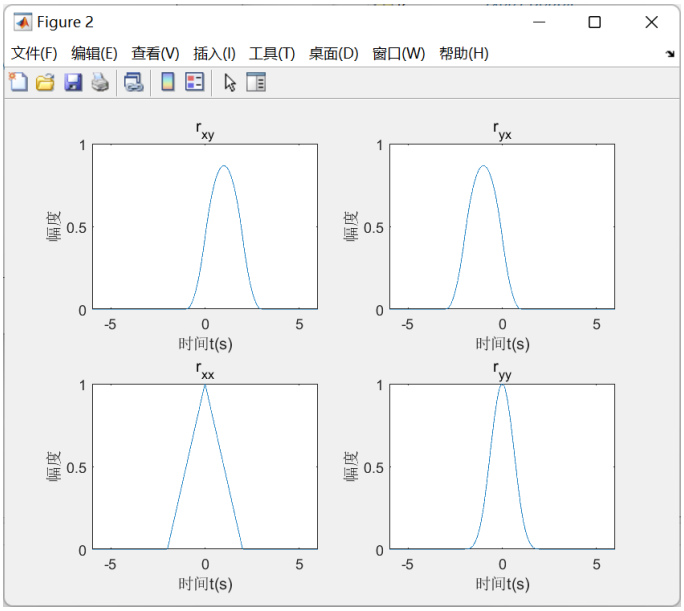

图 1.3 自相关互相关图像

代码如下：

```
% 定义时间轴 t
clear, clc, close all;
t = -3: 0.01: 3;
% f1
f1 = heaviside(t) - heaviside(t-2);
% f2
```

```
f2 = (t+1).* ( heaviside(t+1) - heaviside(t) ) + (1-t) .* (heaviside(t) -
heaviside(t-1) );

% f1(t) * f2(t);
f = conv(f1, f2, 'same'); % 使用 same 参数使卷积结果与输入信号同样长度
figure(1);
plot(t, f);
title('f(t) = f1(t) * f2(t)');
xlabel('时间 t(s)');
ylabel("幅度");
axis([-5, 5, 0, 150]);

% 计算相关
figure(2);
r_xy = xcorr(f1, f2, 'coeff'); % 计算 r_xy，使用 coeff 参数使结果进行归一化
r_yx = xcorr(f2, f1, 'coeff'); % 计算 r_yx，使用 coeff 参数使结果进行归一化
r_xx = xcorr(f1, f1, 'coeff'); % 计算 r_xx，使用 coeff 参数使结果进行归一化
r_yy = xcorr(f2, f2, 'coeff'); % 计算 r_yy，使用 coeff 参数使结果进行归一化

tt = -6 : 0.01 : 6;
subplot(2, 2, 1);
plot(tt, r_xy);
title('r_{xy}');
xlabel('时间 t(s)');
ylabel("幅度");

subplot(2, 2, 2);
plot(tt, r_yx);
title('r_{yx}');
xlabel('时间 t(s)');
ylabel("幅度");

subplot(2, 2, 3);
plot(tt, r_xx);
title('r_{xx}');
xlabel('时间 t(s)');
ylabel("幅度");
subplot(2, 2, 4);
plot(tt, r_yy);
title('r_{yy}');
xlabel('时间 t(s)');
ylabel("幅度");
```

数字信号处理基础 实验二

一、 实验目的

(1) 掌握离散傅里叶变换的定义及概念，在Matlab环境下实现序列的离散傅里叶变换（DFT）

(2) 掌握采样定理的原理和方法，Matlab实现Sa函数的抽样及重构

(3) Matlab求频率响应函数，根据零点分布系统的稳定性

(4) $X(z)$部分分式展开的MATLAB实现，$H(z)$零极点与系统特性的MATLAB计算

二、 实验内容

1） 计算余弦序列 $x(n)=\cos(\frac{\pi}{8}n)R_N(n)$ 的 DFT。分别对 N=10、16、22 时计算 DFT，绘出 X (k) 幅频特性曲线，分析是否有差别及产生差别的原因。

解：

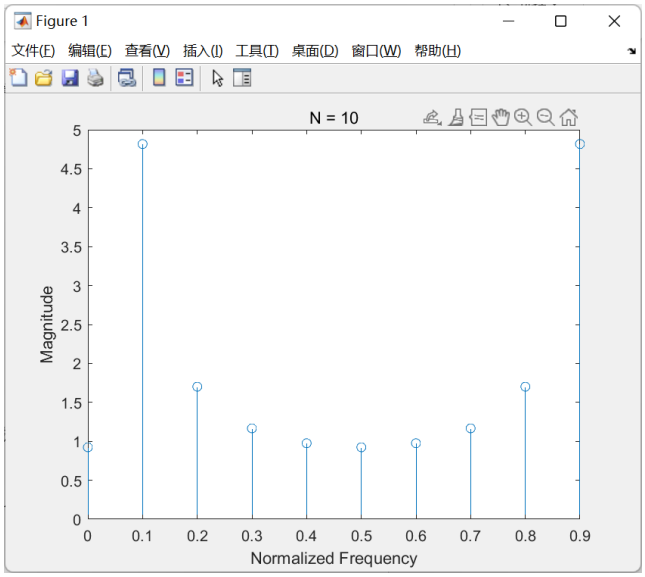

图 2.1 $N = 10$频幅特性曲线

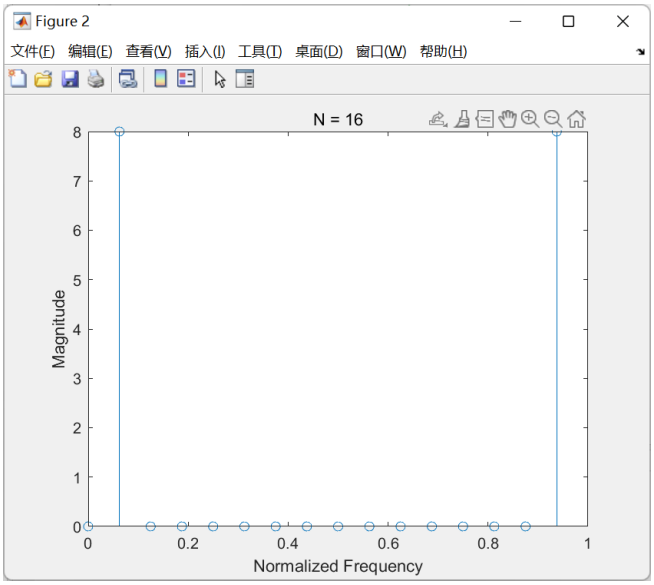

图 2.2 $N = 16$频幅特性曲线

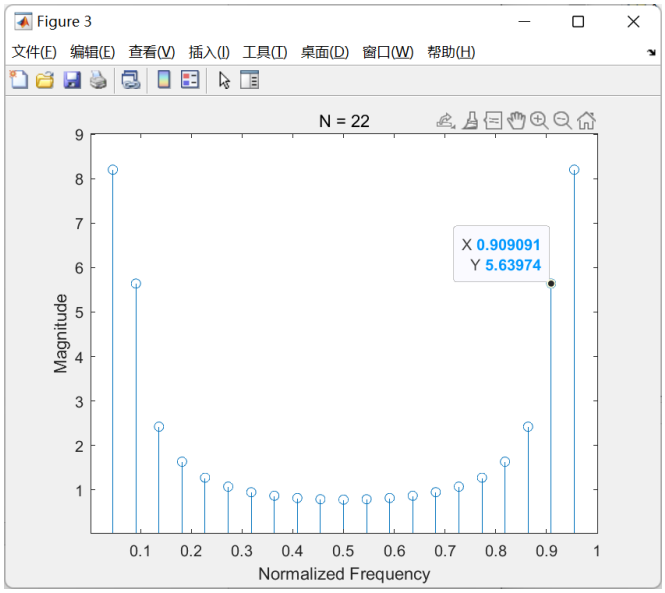

图 2.3 $N = 22$频幅特性曲线

分析差异和产生差异的原因：

可以观察到，随着N的增加，幅频特性曲线的细节变得更加明显。较小的 N 值（如$N = 10$）导致频率分辨率较低，而较大的N值（如$N = 22$）导致频率分辨

率更高。这意味着在较小的N值下，无法捕捉到较高频率分量的细节，而较大的N值允许更好地分辨出更高的频率分量。

此差异是由DFT算法中离散采样点的数量（即 N 值）引起的。较大的N值提供了更多的采样点，从而允许更准确地表示信号的频谱信息。因此，在选择DFT的N值时，需要根据应用需求和期望的频率分辨率来进行权衡。

代码如下：

```
clear, clc, close all;
N = [10, 16, 22];
for i = 1:length(N)
    n = 0:N(i)-1;  % 时间索引
    x = cos(pi/8 * n) .* rectwin(N(i))';  % 序列
    X = fft(x);
    % 计算幅频特性
    magnitude = abs(X);
    f = (0:N(i)-1) / N(i);  % 频率轴
    figure;
    stem(f, magnitude);
    xlabel('Normalized Frequency');
    ylabel('Magnitude');
    title(['N = ', num2str(N(i))]);
end
```

分析：

在以上代码中，使用循环对不同的N值进行计算和绘制。首先，根据给定的N值生成时间索引n和序列x。其中，序列x由余弦函数和矩形窗函数相乘得到。然后，使用fft函数计算DFT，并计算幅频特性（即DFT的幅度）存储在$magnitude$中。接下来，使用$stem$函数绘制幅频特性曲线，其中x轴表示归一化频率，y轴表示幅度。

2） Matlab 实现Sa信号的采样和恢复

信号$Sa(t)$作为被采样信号，信号带宽$B=1$，采样频率$\omega s=2B$，此频率下的采样为 Nyquist 采样，对采样及恢复过程用 Matlab 进行仿真。

MATLAB 中的抽样函数 $\sin c(\mathrm{t})=\frac{\sin(\pi \mathrm{t})}{\pi \mathrm{t}}$ 来表示$Sa（t）$，有$Sa（t）=sinc(\frac{t}{\pi})$，信号重构的内插公式可以表示为：

$$f(\mathrm{t})=\frac{\mathrm{Ts}\,\mathrm{Wc}}{\pi}\sum_{\mathrm{n}=-\infty}^{\infty}\mathrm{f}(\mathrm{nTs})\sin c\left[\frac{Wc}{\pi}(\mathrm{t}-\mathrm{nTs})\right]$$

解：

流程为：

1.定义时间范围：通过$t=-20:0.5:20$;定义了一个时间范围，从-20 到 20，间隔为0.5。这个时间范围用于绘制Sa函数的图像。

2.绘制 Sa 函数图像：使用$sinc(t/pi)$计算 Sa 函数，并使用 plot 函数绘制Sa函数的图像。

3.定义信号参数：$B=1$表示信号带宽，$wc=1$表示截止频率，$\mathrm{ws}=2$ 表示理想低通截止频率，$Ts=2*\frac{pi}{ws}$计算采样周期。

4.定义采样点：通过$n=-40:40$，定义了一个采样点的范围。$nTs=n*Ts$计算了对应的采样时间点。

5.计算抽样信号：使用$sinc(nTs/pi)$计算了抽样信号。这里使用了抽样信号$sinc(t/pi)$来表示 Sa 信号。

6.信号恢复通过内插公式对信号进行恢复。

7.采样信号图像：使用 stem 函数绘制采样信号的图像。

8.恢复信号图像：使用 plot 函数绘制恢复信号的图像。

代码中使用的抽样信号和恢复公式是基于 Sa 函数的特定表示形式和理想低通截止频率的情况。这些公式在给定的信号带宽和采样频率条件下，实现了对 Sa 信号的采样和恢复过程的仿真

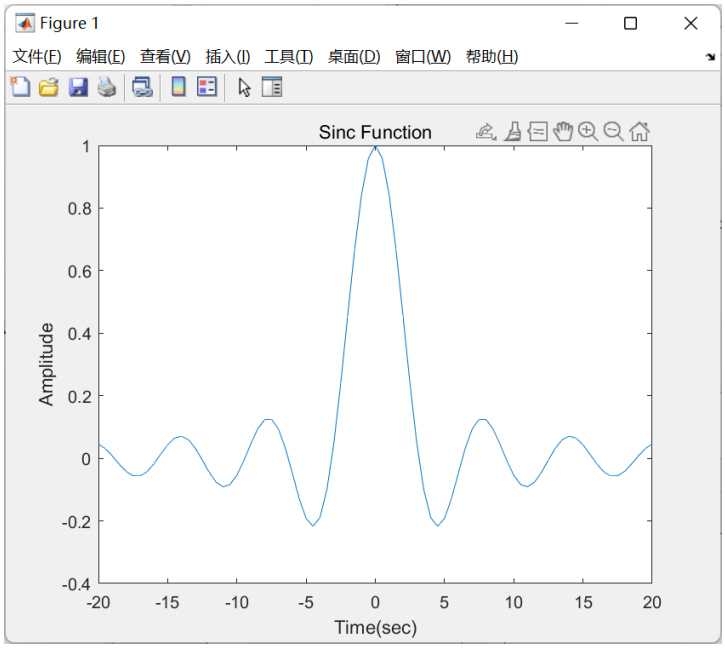

图 2.4 *Sa*函数图像

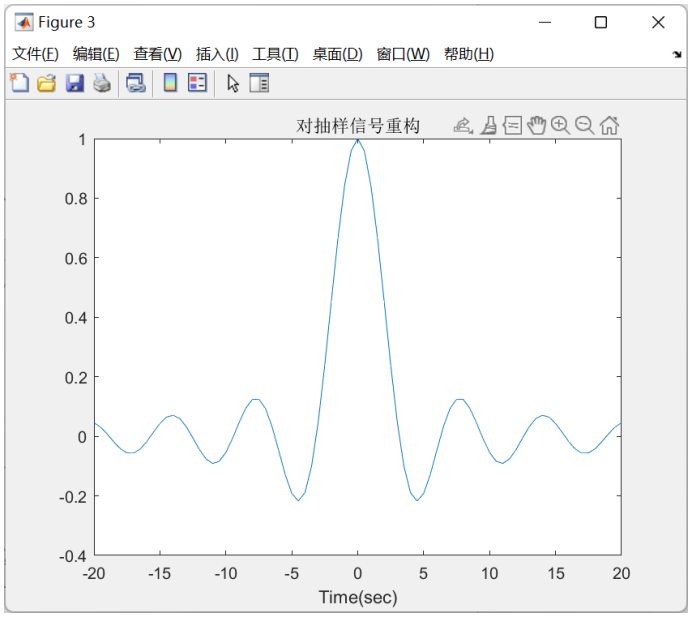

图 2.5 对抽样信号重构

代码如下：

Matlab 代码为：

```
clear, clc, close all;
t = -20: 0.5: 20;
y = sinc(t/pi); % Sa 函数
plot(t, y);
xlabel("Time(sec)"); ylabel("Amplitude"); title("Sinc Function")
B = 1; % 信号带宽
wc = 1; % 截止频率
ws = 2; % 理想低通截止频率
Ts = 2*pi/ws; % 采样周期

n = -40: 40; %
nTs = n*Ts; % 采样点
f = sinc(nTs / pi); % 抽样信号

% 信号冲激
fa = f*Ts*wc/pi*sinc((wc/pi)*(ones(length(nTs), 1)*t-
nTs'*ones(1,length(t))));
figure
stem(t, f);
title('抽样信号');
xlabel("Time(sec)")

figure;
plot(t, fa);
title("对抽样信号重构");
xlabel("Time(sec)")
```

3） 已知系统函数 $H(s)=\frac{1}{s^3+2s^2+3s+1}$，试用 MATLAB 画出其零极点分布，求系统的单位冲激响应 ，和频率响应，并判断系统是否稳定。

解：

*tf*函数是 MATLAB 中用于创建传递函数对象的函数。它接受分子和分母多项式的系数作为输入，并返回一个传递函数对象。传递函数对象可以直接与其他函数

进行操作，如绘制零极点分布、计算单位冲激响应、计算频率响应等。

通过使用传递函数对象，可以更方便地对系统进行分析和操作。例如，可以直接使用$pzmap$函数绘制零极点分布图，而无需手动提取分子和分母多项式的根。同样地，可以使用$impulse$函数和$freqs$函数直接对传递函数对象进行操作，而无需手动进行计算。

因此，使用tf函数创建传递函数对象H可以简化系统分析和操作的过程，并提供更方便的函数接口来处理传递函数。

运行结果：

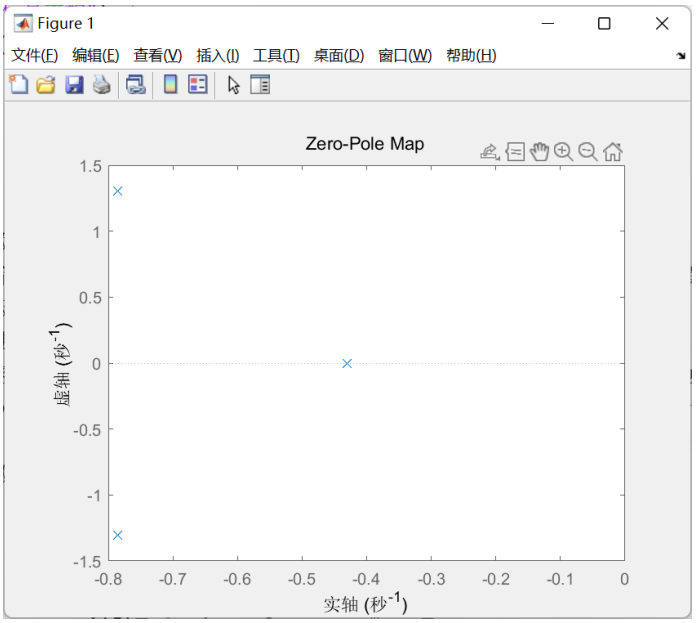

图 2.6 零极点分布

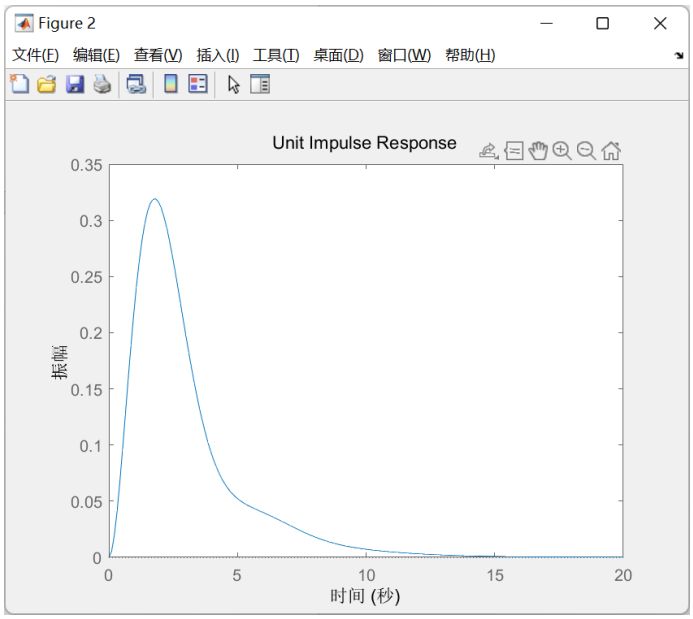

图 2.7 系统的单位冲激响应

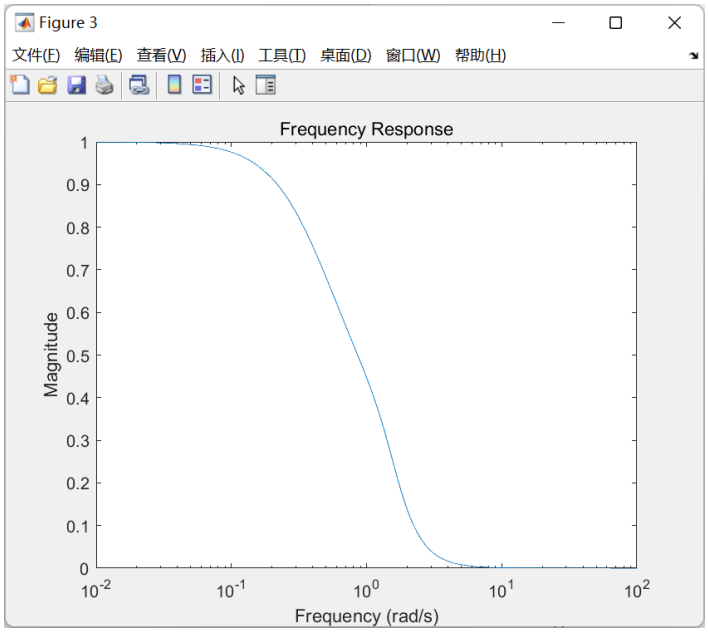

图 2.8 频率响应

代码如下：

```
% 系统函数
num = 1;
den = [1, 2, 3, 1];
H = tf(num, den);
```

```
% 绘制零极点分布
figure;
pzmap(H);
title('Zero-Pole Map');

% 计算并绘制单位冲激响应
figure;
impulse(H);
title('Unit Impulse Response');

% 计算频率响应
w = logspace(-2, 2, 1000);
H_freq = freqs(num, den, w);

% 绘制频率响应
figure;
semilogx(w, abs(H_freq));
title('Frequency Response');
xlabel('Frequency (rad/s)');
ylabel('Magnitude');
% 判断系统稳定性
if all(real(pole(H)) < 0)
    disp('系统是稳定的');
else
    disp('系统是不稳定的');
end
```

判断一个系统的稳定性，可以通过分析系统的极点位置来进行判断。

极点位置法：对于一个线性时不变系统，如果系统的所有极点的实部都小于零，则系统是稳定的。也就是说，所有极点位于左半平面（包括虚轴上的极点）。

运行结果表示：该系统是稳定的

4）

[1] 已知，离散信号的 z 变换 X(z)通常可用有理分式表示：

$$X(z)=\frac{b_0+b_1z^{-1}+b_2z^{-2}+\cdots+b_mz^{-m}}{a_0+a_1z^{-1}+a_2z^{-2}+\cdots+a_nz^{-n}}=\frac{\text{num}(z)}{\text{den}(z)}$$
$$=k+\frac{r_1}{1-p_1z^{-1}}+\frac{r_2}{1-p_2z^{-1}}+\cdots+\frac{r_n}{1-p_nz^{-1}}$$

可以调用 residuez 函数对极点，零点以及部分分式的系数进行求取，

[r,p,k]=residuez(num,den)

其中，num 为 X(z)分子多项式的系数向量；

den 为 X(z)分母多项式的系数向量；

r = r1, r2, ⋯, rn 为部分分式的系数，p = p1, p2, ⋯, pn 为极点。

使用 Matlab 实现 $X(z)=\frac{18}{18+3z^{-1}-4z^{-2}-z^{-3}}$ 部分分式展开式

解:

离散信号的分子多项式系数为18，分母多项式系数为[18, 3, −4, −1]。通过调用*residuez*函数，将得到部分分式展开的结果。

*residuez*函数的结果中，部分分式的系数r是一个向量，包含了每个部分分式的系数。极点p也是一个向量，包含了每个极点的值。常数项k是一个常数表示没有极点的常数项。

运行结果为:

部分分式的系数:

0.3600
0.2400
0.4000

极点:

0.5000

-0.3333
-0.3333

常数项:空

代码如下:

```
clear, clc;
% 离散信号的分子和分母多项式系数
num = [18];
den = [18, 3, -4, -1];
% 调用 residuez 函数计算部分分式展开
[r, p, k] = residuez(num, den);

% 显示部分分式的系数
disp('部分分式的系数:');
disp(r);
% 显示极点
disp('极点:');
disp(p);
% 显示常数项
disp('常数项:');
disp(k);
```

[2] 已知，离散系统的系统函数 H(z)可由有理分式表示:

$$H(z)=\frac{b_0+b_1z^{-1}+b_2z^{-2}+\cdots+b_mz^{-m}}{a_0+a_1z^{-1}+a_2z^{-2}+\cdots+a_nz^{-n}}=\frac{\text{num}(z)}{\text{den}(z)}$$

利用 impz 函数计算 H(z)的单位脉冲响应 h[k]，调用形式为

h = impz(num, den, N)

num 和 den 分别为 H(z)分子多项式和分母多项式的系数向量。N 表示单位脉冲响应输出序列个数，返回值 h 是单位脉冲响应。

H(z)零极点分布图可用 zplane 函数画出，调用形式为：zplane(num, den)

- **试画出离散系统 $H(z)=\frac{z^{-1}+2z^{-2}+z^{-3}}{1-0.5z^{-1}-0.005z^{-2}+0.3z^{-3}}$，零极点分布图，求其单位脉冲响应 h[k]和频率响应 $H(e^{j\Omega})$ ；**

解:

离散系统的分子多项式系数为$[0,1,2,1]$，分母多项式系数为$[1,-0.5,-0.005,0.3]$。通过调用$zplane$函数，可以画出零极点分布图。使用$impz$函数计算单位脉冲响应，并使用$stem$函数绘制单位脉冲响应图。使用$freqz$函数计算频率响应。

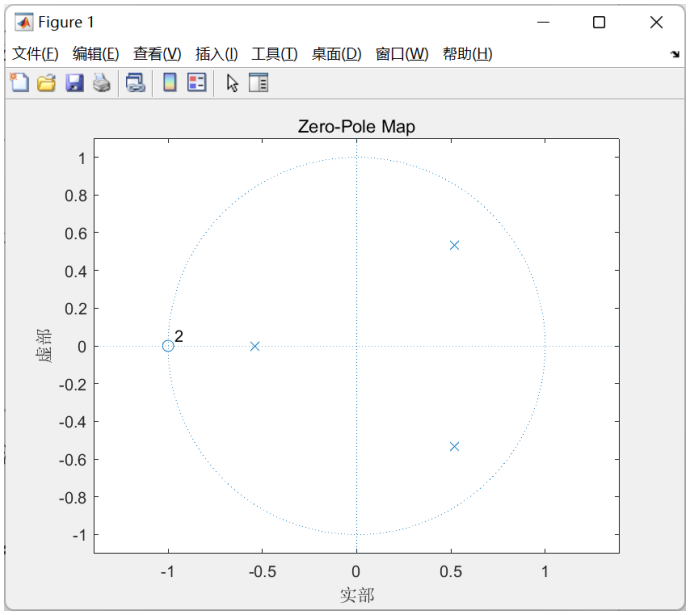

图 2.9 零极点分布图

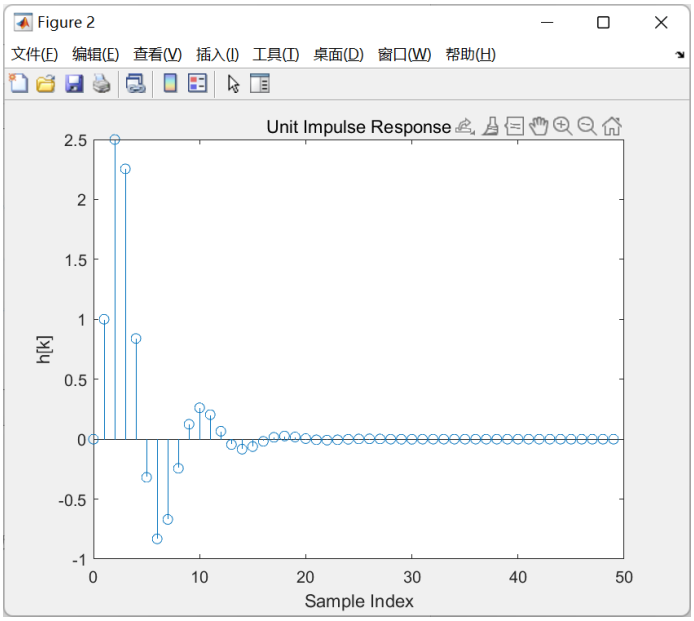

图 2.10 单位脉冲响应图

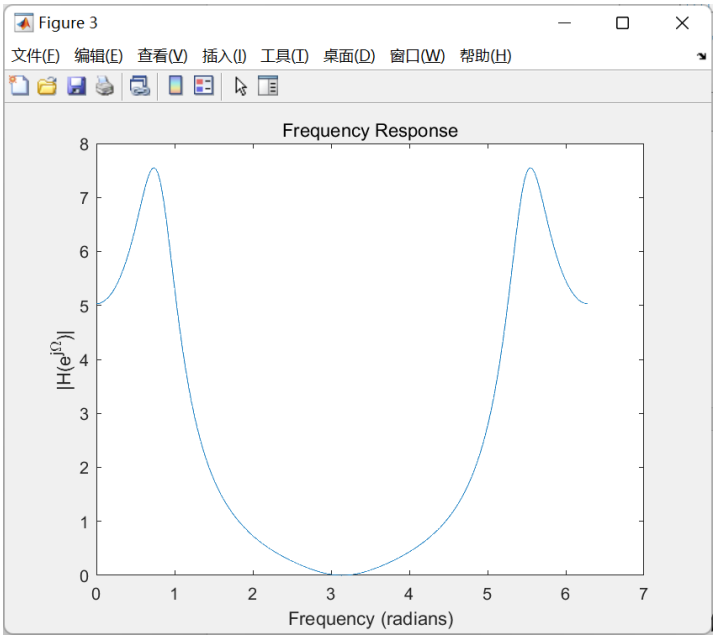

图 2.11 频率响应图

需要注意的是$freqz$函数使用角频率w作为输入，而不是频率Ω。因此，需要在绘制频率响应图时使用角频率

代码如下：

```
% 离散系统的分子和分母多项式系数
num = [0, 1, 2, 1];
den = [1, -0.5, -0.005, 0.3];
% 画零极点分布图
figure;
zplane(num, den);
title('Zero-Pole Map');
% 计算单位脉冲响应
N = 50; % 单位脉冲响应序列个数
h = impz(num, den, N);
% 画单位脉冲响应图
figure;
stem(0:N-1, h);
title('Unit Impulse Response');
xlabel('Sample Index');
ylabel('h[k]');
% 计算频率响应
w = linspace(0, 2*pi, 1000); % 角频率范围：0 到 2π
```

```
H_freq = freqz(num, den, w);

% 画频率响应图
figure;
plot(w, abs(H_freq));
title('Frequency Response');
xlabel('Frequency (radians)');
ylabel('|H(e^{j\Omega})|');
```

- **并且通过以下表达式作图讨论极点在实轴变化对应的系统冲击响应的特点。**

$H(z)=\frac{1}{1-az^{-1}}$， a 分别为 1.5，1，0.5，-0.5，-1，-1.5。

解：

该问题中每个图像对应不同的极点实部值。在每个图像中，将显示对应于特定a值的系统的单位脉冲响应。通过绘制单位脉冲响应图，可以观察极点实部值对系统响应的影响。

在这个例子中，a_values是一个包含不同a值的向量。对于每个a值，将构建相应的分子和分母多项式系数，并使用$impz$函数计算单位脉冲响应。通过使用$stem$函数绘制单位脉冲响应图。

值得注意的是，根据a的不同取值，单位脉冲响应图可能会显示出不同的特征。正值的a将导致振荡，负值的a将导致衰减。通过观察单位脉冲响应图，可以更好地理解极点实轴变化对系统响应的影响。

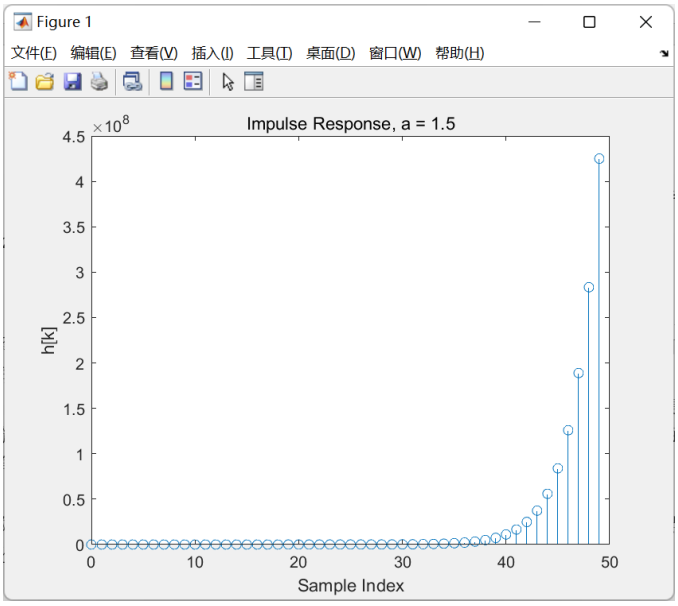

图 2.12 $a = 1.5$时的单位脉冲响应图

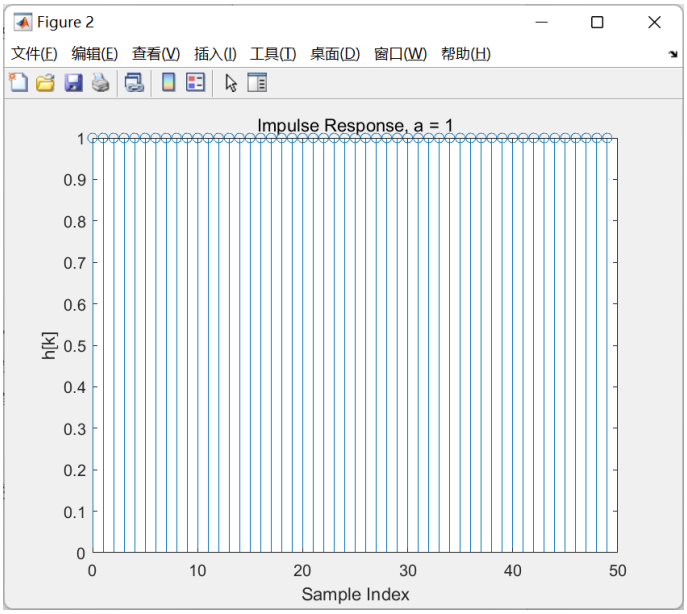

图 2.13 $a = 1$时的单位脉冲响应图

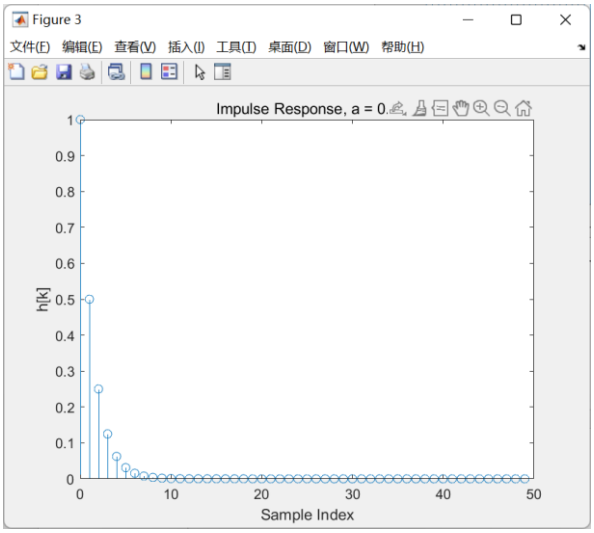

图 2.14 $a = 0.5$时的单位脉冲响应图

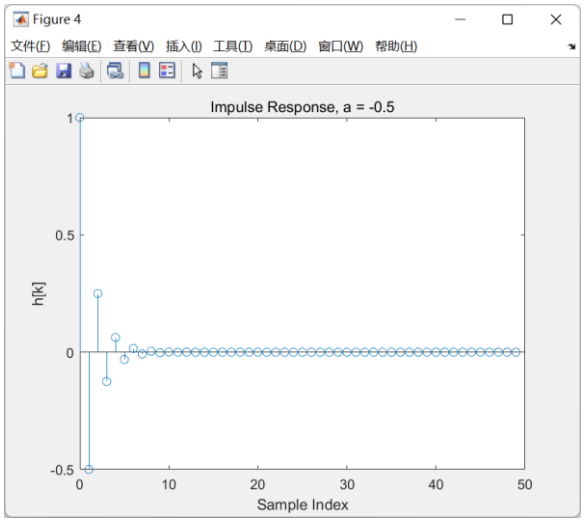

图 2.15 $a = -0.5$时的单位脉冲响应图

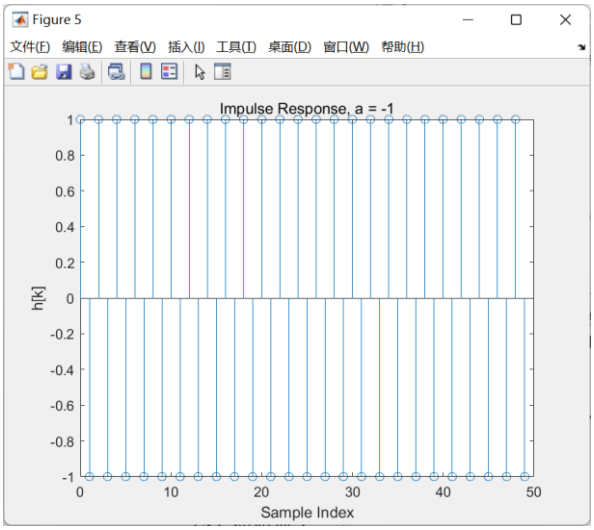

图 2.16 $a = -1$时的单位脉冲响应图

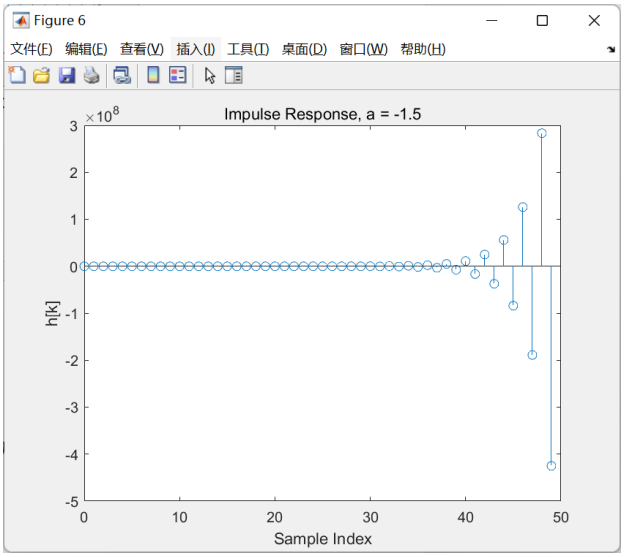

图 2.17 $a=-1.5$时的单位脉冲响应图

代码如下：

```
a_values = [1.5, 1, 0.5, -0.5, -1, -1.5];
N = 50; % 单位脉冲响应序列个数

for i = 1:length(a_values)
    a = a_values(i);
    num = 1;
    den = [1, -a];

    % 计算单位脉冲响应
    h = impz(num, den, N);

    % 画单位脉冲响应图
    figure;
    stem(0:N-1, h);
    title(['Impulse Response, a = ' num2str(a)]);
    xlabel('Sample Index');
    ylabel('h[k]');
end
```

- **通过以下表达式作图讨论极点在虚轴变化对应的系统冲击响应的特点。**

$H(z)=\frac{1}{1+a^2z^{-2}}$，a 分别为 0.8，1，1.5。

解：

每个图像对应不同的极点虚部值。在每个图像中，将显示对应于特定a值的系统的单位脉冲响应。通过绘制单位脉冲响应图，可以观察极点虚部值对系统响应的影响。

在这个例子中，a_values是一个包含不同a值的向量。对于每个a值，将构建相应的分子和分母多项式系数，并使用$impz$函数计算单位脉冲响应。通过使用$stem$函数绘制单位脉冲响应图。

值得注意的是，根据a的不同取值，单位脉冲响应图可能会显示出不同的特征。较大的a值将导致单位脉冲响应图在时间轴上出现更多的振荡。通过观察单位脉冲响应图，可以更好地理解极点虚轴变化对系统响应的影响。

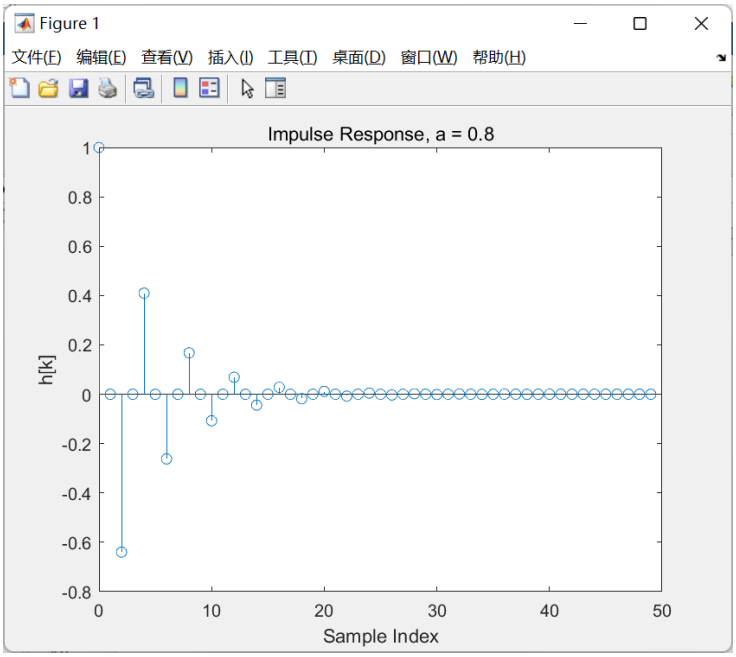

图 2.18 $a = 0.8$的单位脉冲响应图

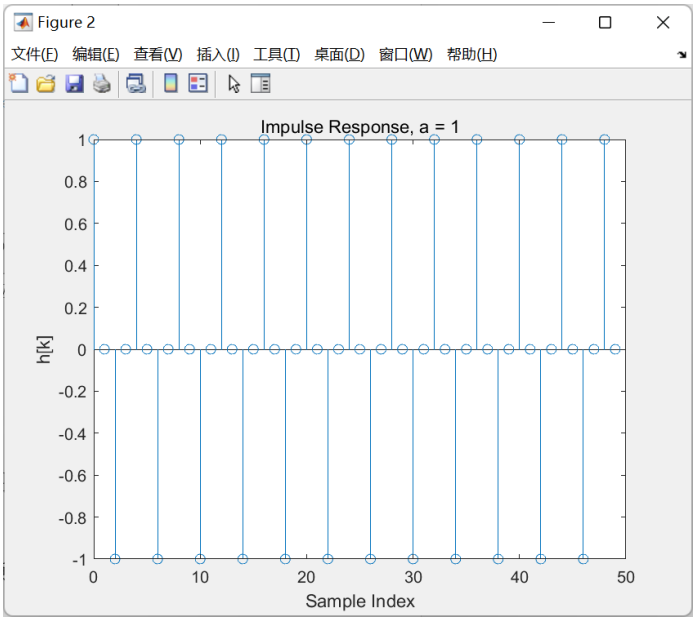

图 2.19 $a = 1$的单位脉冲响应图

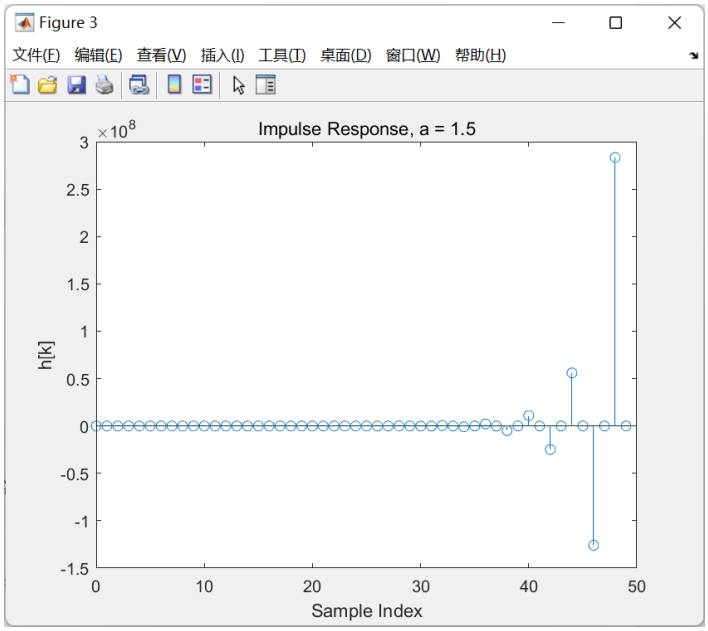

图 2.20 $a=1.5$的单位脉冲响应图

代码如下：

```
a_values = [0.8, 1, 1.5];
N = 50; % 单位脉冲响应序列个数
for i = 1:length(a_values)
    a = a_values(i);
    num = 1;
    den = [1, 0, a^2];
    % 计算单位脉冲响应
    h = impz(num, den, N);
    % 画单位脉冲响应图
    figure;
    stem(0:N-1, h);
    title(['Impulse Response, a = ' num2str(a)]);
    xlabel('Sample Index');
    ylabel('h[k]');
end
```

（3）实验要求：

1、在 matlab 环境下编写和调试程序

2、保存程序的运行结果，并结合算法进行分析；结果图件要求有横、纵坐标及图注；

数字信号处理基础 实验三

一、实验目的

按实验任务要求编制和运行有限长脉冲响应（FIR）滤波的程序。设计数字低通和高通滤波器，实现对离散信号的低通和高通滤波处理。加强对数字滤波器设计和实现方法的认识，提高实现数字信号处理的编程能力。

二、实验内容

任务-I： 编写频率域中零相位FIR滤波的程序

任务-II： 设计低通滤波器

1. 根据给定信号的频谱特征，在频率域设计低通滤波器；
2. 运行程序实现对给定的信号的低通滤波处理；
3. 绘图显示低通滤波的计算结果。

任务-III： 设计高通滤波器

1. 根据给定信号的频谱特征，在频率域设计高通滤波器；
2. 运行程序实现对给定的信号的高通滤波处理；
3. 绘图显示高通滤波的计算结果。

三、实验报告

本实验结束后要求写出实验报告，主要包括以下内容:

1．实验目的及内容

2．基本原理阐述

3．实验结果图件以及结果分析

解：

（1） 编写频率域中零相位 FIR 滤波的程序

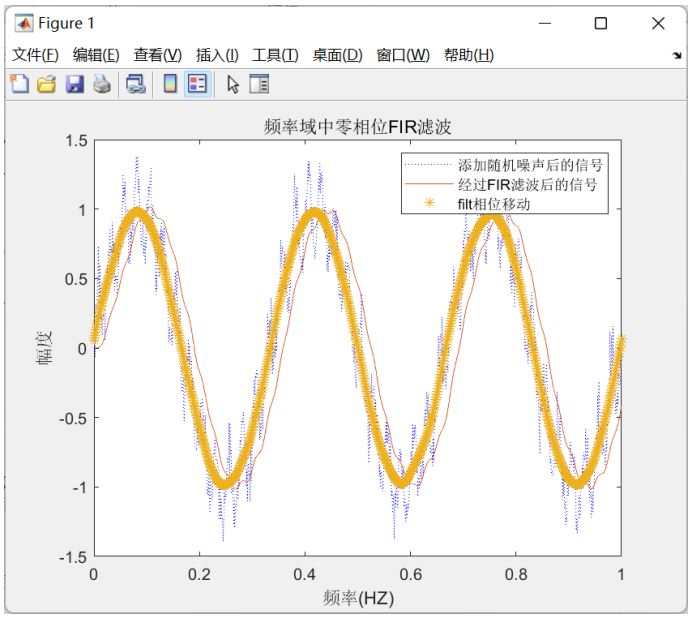

图 3.1 FIR 滤波

代码如下：

```
clear, clc, close all;
% 原始信号
t = 0:0.001:1; % 时间范围
x = sin(6*pi*t) + 0.25*sin(80*pi*t); % 原始信号

% 添加随机噪声
rng(0); % 设置随机种子，以确保结果可重复
noise = 0.1*randn(size(x)); % 产生服从正态分布的随机噪声
x_noisy = x + noise; % 添加随机噪声后的信号

% FIR 滤波器设计
order = 50; % 滤波器阶数
cutoff_freq = 20; % 截止频率
Fs = 1000; % 采样频率
nyquist_freq = Fs/2; % 奈奎斯特频率
normalized_cutoff_freq = cutoff_freq/nyquist_freq; % 归一化的截止频率
filter_coeffs = fir1(order, normalized_cutoff_freq); % FIR 滤波器系数
```

```
% FIR 滤波
filtered_signal = filter(filter_coeffs, 1, x_noisy);

filt_signal = filtfilt(filter_coeffs, 1, x_noisy);

% 绘制结果
axis([0, 0.5, -2, 2]);
plot(t, x_noisy, 'b:');
% title('添加随机噪声后的信号');
hold on;
plot(t, filtered_signal);
% title('经过 FIR 滤波后的信号');
hold on;
plot(t, filt_signal, '*');
legend("添加随机噪声后的信号", "经过 FIR 滤波后的信号", "filt 相位移动");
xlabel("频率(HZ)");
ylabel("幅度");
title("频率域中零相位 FIR 滤波");
```

在上述代码中，首先定义了原始信号，使用sin函数生成了一个由两个正弦波组成的信号。然后，添加了服从正态分布的随机噪声，使用$randn$函数生成与原始信号大小相同的随机噪声，并将其加到原始信号上得到带噪声的信号。

接下来，使用$fir1$函数设计了一个FIR滤波器。$fir1$函数的第一个参数是滤波器的阶数，这里设置为50；第二个参数是归一化的截止频率，通过将截止频率除以奈奎斯特频率来进行归一化；输出的$filter_coeffs$是FIR滤波器的系数。

最后，使用$filter$函数对带噪声的信号进行滤波，$filter_coeffs$是FIR滤波器的系数，1 表示滤波器的分母系数为 1，即没有反馈。

（2） 设计低通滤波器

1.根据给定信号的频谱特征，在频率域设计低通滤波器；

2.运行程序实现对给定的信号的低通滤波处理；

3.绘图显示低通滤波的计算结果。

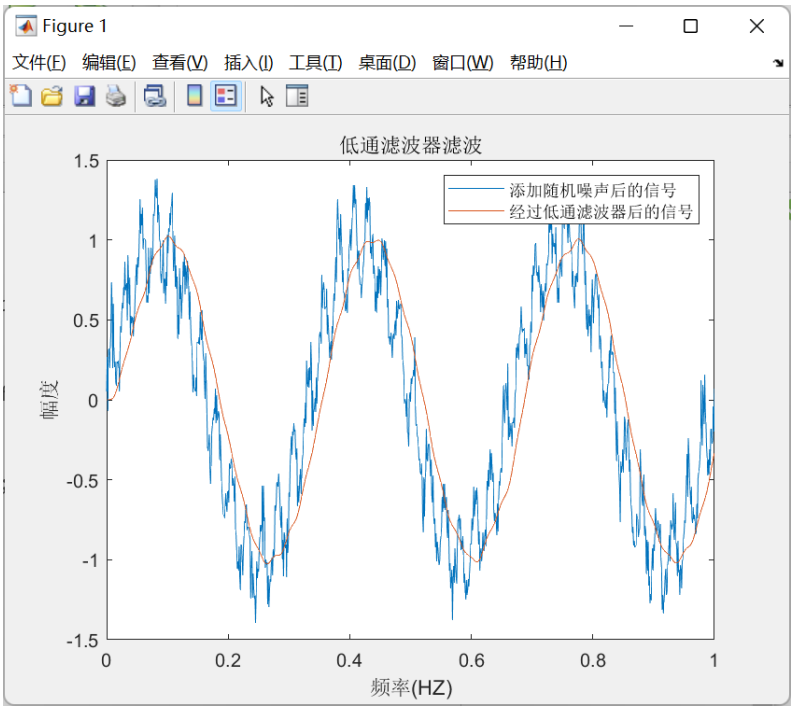

图 3.2 低通滤波

代码如下：

```
clear, clc, close all;
% 原始信号
t = 0:0.001:1; % 时间范围
x = sin(6*pi*t) + 0.25*sin(80*pi*t); % 原始信号

% 添加随机噪声
rng(0); % 设置随机种子，以确保结果可重复
noise = 0.1*randn(size(x)); % 产生服从正态分布的随机噪声
x_noisy = x + noise; % 添加随机噪声后的信号

% 低通滤波器设计
cutoff_freq = 20; % 截止频率
order = 4; % 滤波器阶数
```

```
% 归一化的截止频率
Fs = 1000; % 采样频率
nyquist_freq = Fs/2; % 奈奎斯特频率
normalized_cutoff_freq = cutoff_freq/nyquist_freq; % 归一化的截止频率

% 设计 Butterworth 低通滤波器
[b, a] = butter(order, normalized_cutoff_freq);

% 应用低通滤波器
filtered_signal = filter(b, a, x_noisy);

% 绘制结果
figure;
plot(t, x_noisy);
% title('添加随机噪声后的信号');
hold on;
plot(t, filtered_signal);
% title('经过低通滤波器处理后的信号');
legend('添加随机噪声后的信号', "经过低通滤波器后的信号")
xlabel("频率(HZ)");
ylabel("幅度");
title("低通滤波器滤波")
```

在上述代码中，首先定义了原始信号，使用sin函数生成了一个由两个正弦波组成的信号。然后，添加了服从正态分布的随机噪声，使用$randn$函数生成与原始信号大小相同的随机噪声，并将其加到原始信号上得到带噪声的信号。

接下来，我们使用$butter$函数设计了一个$Butterworth$低通滤波器。$butter$函数的第一个参数是滤波器的阶数，这里设置为 4；第二个参数是归一化的截止频率，通过将截止频率除以奈奎斯特频率来进行归一化。函数的输出变量b和a是滤波器的分子和分母系数。

最后，我们使用$filter$函数对带噪声的信号应用低通滤波器，将滤波器的分子和分母系数传递给该函数。

（3） 设计高通滤波器

1.根据给定信号的频谱特征，在频率域设计高通滤波器；

2.运行程序实现对给定的信号的高通滤波处理；

3.绘图显示高通滤波的计算结果。

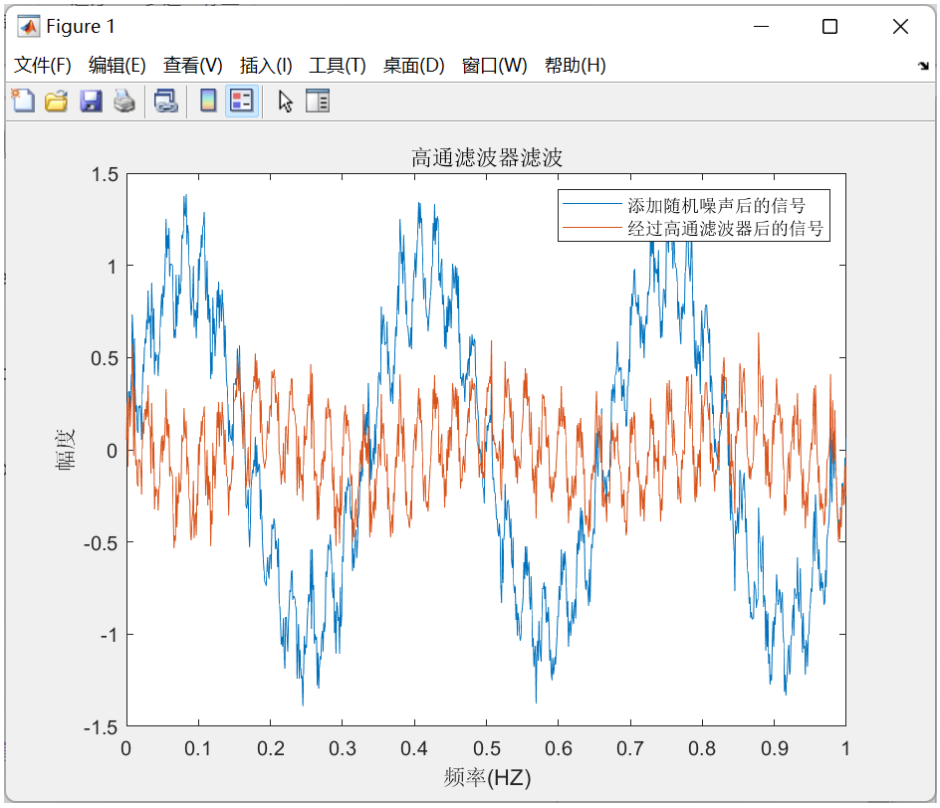

图3.3 高通滤波

在上述代码中基础上添加了服从正态分布的随机噪声，使用*randn*函数生成与原始信号大小相同的随机噪声，并将其加到原始信号上得到带噪声的信号。接下来，我们使用butter函数设计了一个*Butterworth*高通滤波器。*butter*函数的第一个参数是滤波器的阶数，这里设置为4；第二个参数是归一化的截止频率，通过将截止频率除以奈奎斯特频率来进行归一化；'high'表示高通

滤波器。函数的输出变量b和a是滤波器的分子和分母系数。

最后，使用$filter$函数对带噪声的信号应用高通滤波器，将滤波器的分子和分母系数传递给该函数。