



# 实验报告

课程名称 控制系统实验

实验内容 Matlab 仿真

开课学期 2025 春季学期

学生姓名 宋正非 学生学号 2151094

指导老师 赵霞

# 目录

2.1 建立一个 TF 对象 $G(z)$ .....	3
2.2 从 TF 转换到 ZPK 形式 .....	4
2.3 部分分式展开式和 $g(k)$ .....	6
2.4 由单极点引起的响应 .....	7
2.5 由单位圆上的复极点引起的响应 .....	10
2.6 重极点问题 .....	12
2.7 $G(z)$ 的阶跃响应 .....	13
2.8 任意输入的响应 .....	17
2.9 极点和系统的稳定性 .....	18
2.10 零点 .....	21
2.12 单输入双输出系统 .....	23
3.3 串/并联 .....	25
3.4 反馈联接 .....	29
3.5 双输入反馈系统 .....	32
4.1 采样定理 .....	35
4.2 零阶保持器 .....	37
4.3 采样及零阶保持 .....	38
5.1 .....	44
5.2 .....	54
5.3 .....	60
5.4 .....	63
6.1 .....	66
7.1 .....	81

## 2.1 建立一个 TF 对象 G(z)

创建一个三阶系统的 TF 对象，它的差分方程是：

$$y(k+3) - 2.7y(k+2) + 2.42y(k+1) - 0.72y(k) = 0.1u(k+2) + 0.03u(k+1) - 0.07u(k) \quad (2.4)$$

系统未指明采样周期。显示对象的属性 (get (G)) 并从 TF 对象中提取分子和分母多项式，同时提取零点、极点和增益。并使用 pzmap 在 z 平面上绘制该对象的零极点。

```
% 建立传递函数
clear
num = [0.1, 0.03, -0.07];
den = [1, -2.7, 2.42, -0.72];

G = tf(num, den, -1)

% 显示对象的属性
get(G)

% 提取分子和分母
[num_poly, den_poly] = tfdata(G, 'v')

% 提取零点、极点、增益
[z, p, k] = tf2zp(num, den)

% 绘制 z 平面上的零极点图
pzmap(G)
grid on

G =

```

$$\frac{0.1 z^2 + 0.03 z - 0.07}{z^3 - 2.7 z^2 + 2.42 z - 0.72}$$

```

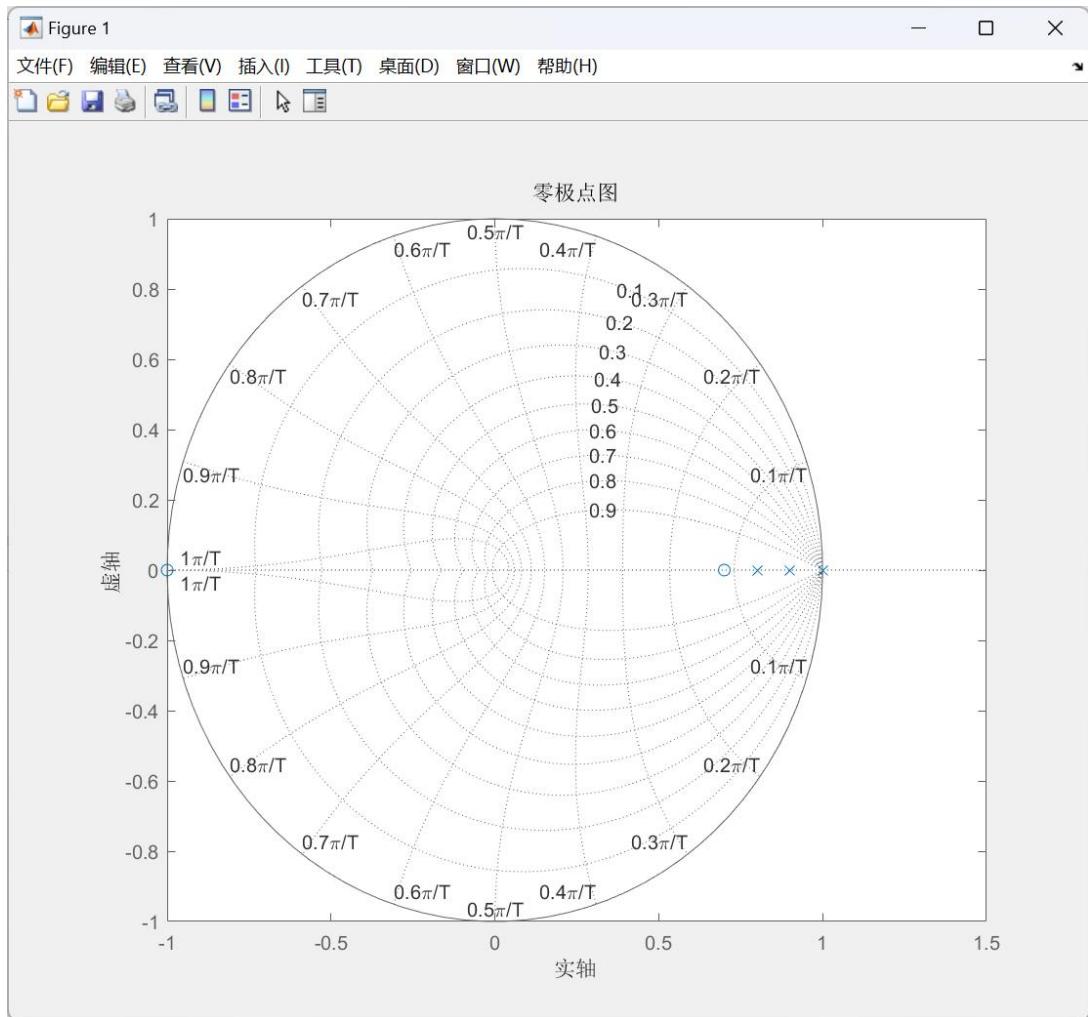
num_poly =
0     0.1000    0.0300   -0.0700

den_poly =
1.0000   -2.7000    2.4200   -0.7200

z =
-1.0000
0.7000

p =
1.0000
0.9000
0.8000

k =
0.1000
```



## 2.2 从 TF 转换到 ZPK 形式

用例 2.1 创建的 TF 对象直接转换为 ZPK 对象。然后提取它的零点、极点、增益和采样时间，说明它们同例 2.1 中创建的 TF 对象的对应值一致。将此系统描述成状态空间形式，并求其特征根、特征多项式。

```
% 定义传递函数 (TF) 对象
num = [0.1, 0.03, -0.07];
den = [1, -2.7, 2.42, -0.72];
G = tf(num, den, -1);

% 转换为 ZPK (零极点增益) 对象
Gzpk = zpk(G);

% 提取零点、极点、增益和采样时间
z = Gzpk.Z{:};
p = Gzpk.P{:};
k = Gzpk.K;
Ts = Gzpk.Ts;

% 显示结果
```

```

disp('零点:'); disp(z);
disp('极点:'); disp(p);
disp('增益:'); disp(k);
disp('采样时间 Ts:'); disp(Ts);

% 转换为状态空间模型
Gss = ss(G);

% 显示状态空间矩阵
disp('A ='); disp(Gss.A);
disp('B ='); disp(Gss.B);
disp('C ='); disp(Gss.C);
disp('D ='); disp(Gss.D);

% 特征根
eig_A = eig(Gss.A);
disp('特征根(极点) ='); disp(eig_A);

% 特征多项式
char_poly = poly(Gss.A);
disp('特征多项式系数 ='); disp(char_poly);

```

```

>> ex_2_2
零点:
-1.0000
 0.7000

极点:
 1.0000
 0.9000
 0.8000

增益:
 0.1000

采样时间 Ts:
 -1

A =
 2.7000   -1.2100    0.7200
 2.0000        0        0
 0        0.5000        0

B =
 0.5000
 0
 0

C =
 0.2000    0.0300   -0.1400

D =
 0

特征根(极点) =
 1.0000

```

```

0.9000
0.8000

特征多项式系数 =
1.0000   -2.7000    2.4200   -0.7200

```

## 2.3 部分分式展开式和 g(k)

线性时不变系统的传递函数

$$G(z) = \frac{2z^2 - 2.2z + 0.56}{z^3 - 0.6728z^2 + 0.0463z + 0.4860} \quad (2.5)$$

单位采样周期。实现  $G(z)/z$  的部分分式展开式（了解如何用得到的结果将  $G(z)$  写成项  $Az/(z-p)$  和的形式，每一项有单独的极点并在分子里面有  $z$ ）。然后用 TF 建立系统，并画出它的脉冲（单位 delta）响应。

提示： `[y,k]=impulse(G); stem(k,y,'filled')`

```

% 分子和分母系数
num = [2, -2.2, 0.56];
den = [1, -0.6728, 0.0463, 0.4860];

% G(z)/z
num_div_z = [0, 2, -2.2, 0.56];

% residuez 展开
[r, p, k] = residuez(num_div_z, den);

% 显示展开结果
disp('残差 (A_i) :'); disp(r);
disp('极点 (p_i) :'); disp(p);
disp('直接项 (k) :'); disp(k);

% 构建传递函数对象，单位采样周期
G = tf(num, den, 1);

% 绘制单位脉冲响应
[y, k] = impulse(G);
stem(k, y, 'filled');
title('单位脉冲响应 g(k)');
xlabel('k'); ylabel('g(k)');
grid on;

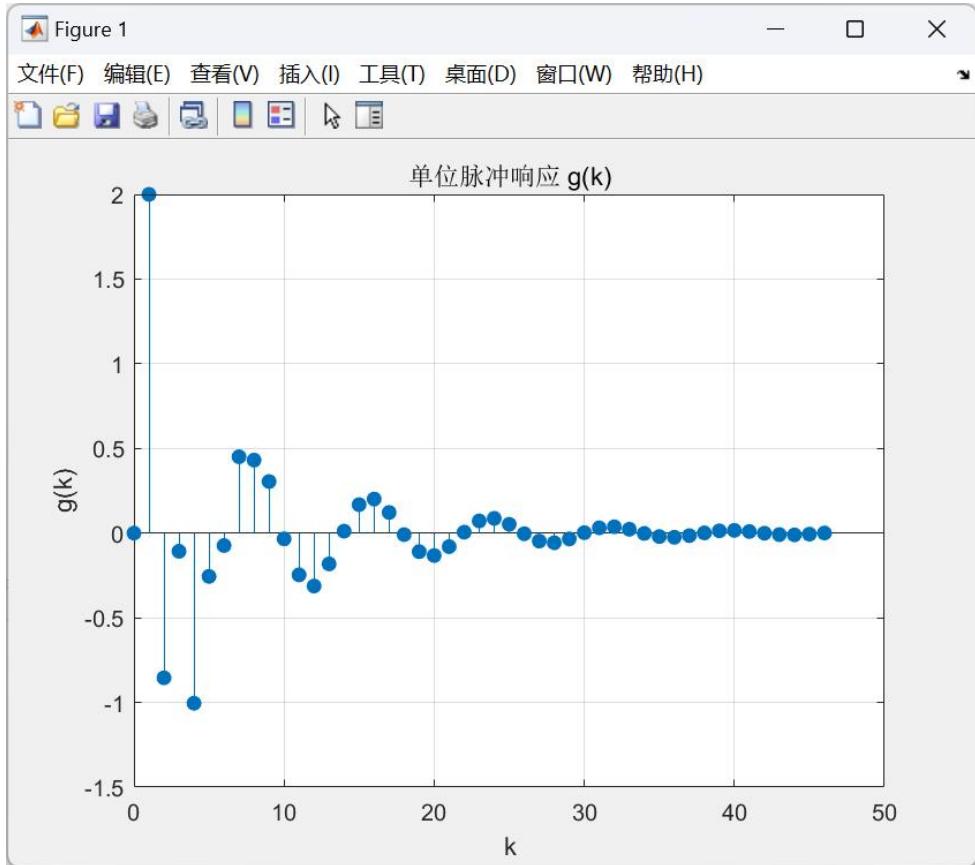
>> ex_2_3
残差 (A_i) :
0.5444 + 0.0294i
0.5444 - 0.0294i
-2.2410 + 0.0000i

极点 (p_i) :
0.6364 + 0.6364i
0.6364 - 0.6364i

```

```
-0.6000 + 0.0000i
```

直接项 (k) :  
1.1523



## 2.4 由单极点引起的响应

对于在例 2.3 里给出的  $G(z)$ , 分别计算传递函数  $G(z)/z$  的极点和留数, 即实极点  $z=0$  和  $z=-0.6$ 、复极点  $z=0.90e^{j0.7854}$  各自引起的脉冲响应  $g(k)$ ,  $dtime=[0:30]$ 。并验证所有极点对应的脉冲响应之和与例 2.3 系统的脉冲响应的一致性。(单独极点对应的响应可以自己编, 也可以采用下面提供的命令, 需要 copy 下列文件到 matlab 对应的文件夹里, 所有提供的非 matlab 原程序, 都在 tool 文件夹里)

复极点的脉冲响应函数 `cpole2k.m`     $y = \text{cpole2k}(cp,res,dtime)$

Inputs:  $cp$  - complex pole, having POSITIVE imaginary part

$res$  - residue at the complex pole

$dtime$  - discrete-time vector  $[0 1 2 \dots k_{\max}]$

Output:  $y$  - discrete-time response vector

实极点的脉冲响应函数 `rpole2k.m`

```
clc; clear;
```

```

% 原系统 G(z)
num = [2, -2.2, 0.56];
den = [1, -0.6728, 0.0463, 0.4860];

% 构造 G(z)/z
num_div_z = [0, 2, -2.2, 0.56]; % 在前面加 0 相当于除以 z
den_div_z = conv([1, 0], den); % 分母加一个 z=0 的极点

% 使用 residue 分解
[r, p, ~] = residue(num_div_z, den_div_z);

% 输出极点和留数
disp('极点和对应留数: ');
for i = 1:length(p)
    fprintf('p(%d) = %.4f%+.4fj,\t r(%d) = %.4f%+.4fj\n', ...
        i, real(p(i)), imag(p(i)), i, real(r(i)), imag(r(i)));
end

% 时间向量
k_time = 0:30;

% 初始化
g_real_all = zeros(size(k_time));
g_complex = zeros(size(k_time));

for i = 1:length(p)
    if imag(p(i)) == 0
        % 实极点响应 (z=0 或 -0.6)
        g_real_all = g_real_all + real(r(i)) * (p(i).^k_time));
    elseif imag(p(i)) > 0
        % 复极点对 (只算正虚部一个, 乘 2 取实部)
        g_complex = g_complex + cpole2k(p(i), r(i), k_time);
    end
end

% 总响应
g_total = g_real_all + g_complex;

% impulse 响应 (G(z))
G = tf(num, den, 1);
[y, k_imp] = impulse(G, length(k_time)-1);

% ----- 画图 -----
figure;

subplot(3,1,1);
stem(k_time, g_real_all, 'filled');
title('实极点 (z = 0 和 z = -0.6) 的脉冲响应');
ylabel('g_{real}(k)');
grid on;

subplot(3,1,2);
stem(k_time, g_complex, 'filled');
title('复极点对的脉冲响应');
ylabel('g_{complex}(k)');
grid on;

```

```

subplot(3,1,3);
stem(k_time, g_total, 'filled', 'DisplayName', 'g_{total}(k)'); hold on;
stem(k_imp, y, 'r--', 'DisplayName', 'impulse(G)');
title('总响应 vs impulse(G)');
xlabel('k');
ylabel('响应');
legend;
grid on;

%% 函数
function y = cpole2k(cp,res,dtime)
% Performs inverse z transform for two complex
% conjugate poles and generates a discrete-time
% function.
temp = zeros(size(dtime));
temp(1) = res;
for kk = 2:length(dtime)
    temp(kk) = temp(kk-1)*cp;
end
y = 2 * real(temp);
end

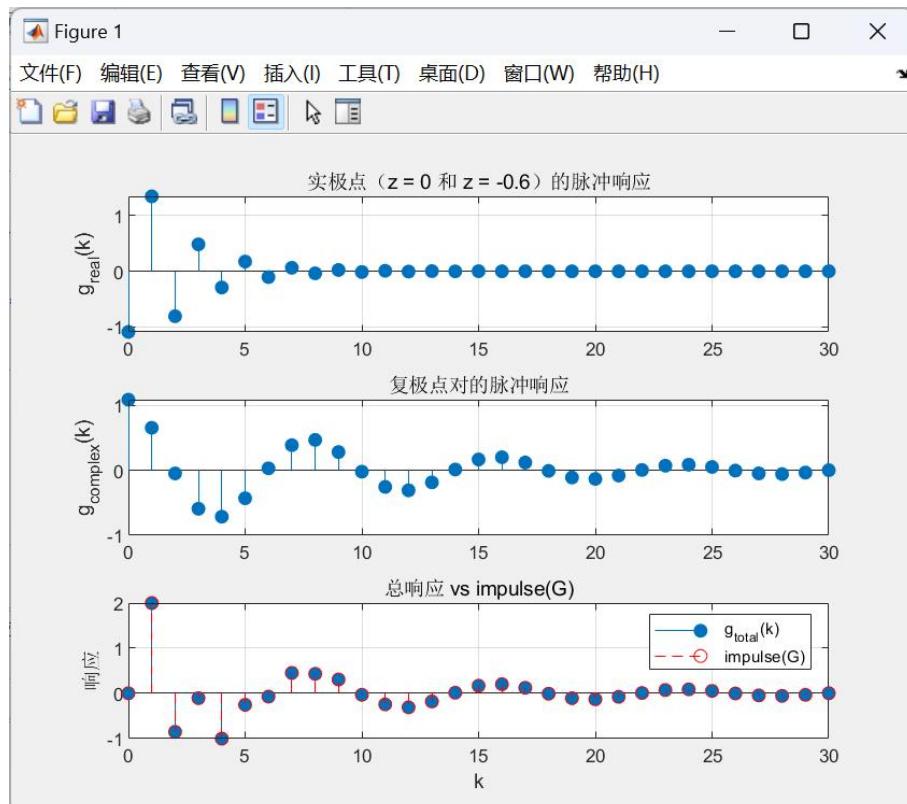
```

计算得  $G(z)/z$  的极点和留数并输出，如下所示。

极点和对应留数：

$p(1) = 0.6364+0.6364j$ ,	$r(1) = 0.5444+0.0294j$
$p(2) = 0.6364-0.6364j$ ,	$r(2) = 0.5444-0.0294j$
$p(3) = -0.6000+0.0000j$ ,	$r(3) = -2.2410+0.0000j$
$p(4) = 0.0000+0.0000j$ ,	$r(4) = 1.1523+0.0000j$

实际点与复极点的脉冲响应相加得到总响应，由下图中总响应 vs impulse(G)可看出，2.4 中所有极点对应的脉冲响应之和与 2.3 系统得脉冲响应一致，验证完成。



## 2.5 由单位圆上的复极点引起的响应

对于传递函数  $G(z)$ , 在单位圆上有复极点  $z=e^{\pm j\pi/3}$ , 一个零点  $z=-0.5$  和单位增益, 单位采样周期。画出脉冲响应的第一个 20s 的响应图。如果使复极点变为  $z=1.1e^{\pm j\pi/3}$ ,  $z=0.9e^{\pm j\pi/3}$ , 分别说出系统的脉冲响应发生了怎样的变化。

```
% 时间向量
k = 0:20;

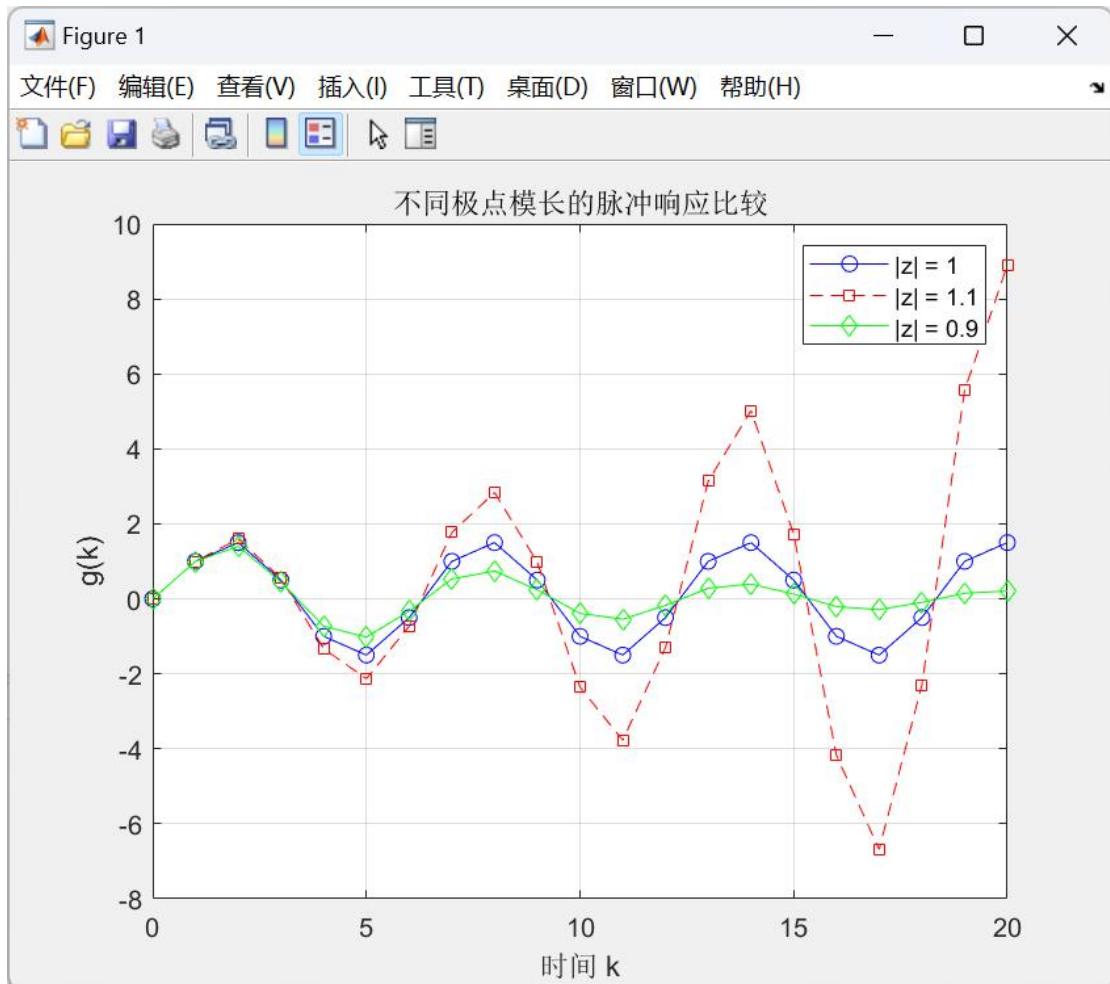
% 零点
z_zero = -0.5;

% 极点定义
p1 = exp(1j*pi/3); % |z| = 1
p2 = 1.1 * exp(1j*pi/3); % 模长 > 1
p3 = 0.9 * exp(1j*pi/3); % 模长 < 1

% 构造三个系统 (单位增益 G=1)
sys1 = zpk(z_zero, [p1, conj(p1)], 1, 1); % 原始系统
sys2 = zpk(z_zero, [p2, conj(p2)], 1, 1); % 放大系统
sys3 = zpk(z_zero, [p3, conj(p3)], 1, 1); % 衰减系统

% 脉冲响应
[y1, k1] = impulse(sys1, k);
[y2, ~] = impulse(sys2, k);
[y3, ~] = impulse(sys3, k);

% 绘图
figure;
plot(k1, y1, 'b-o', 'DisplayName', '|z| = 1');
hold on;
plot(k1, y2, 'r--s', 'DisplayName', '|z| = 1.1');
plot(k1, y3, 'g-d', 'DisplayName', '|z| = 0.9');
xlabel('时间 k');
ylabel('g(k)');
title('不同极点模长的脉冲响应比较');
legend;
grid on;
```



当模长为 1 时，见蓝色曲线，脉冲响应是一个不衰减的振荡信号，波动幅度保持恒定；属于边界稳定系统（临界稳定）。当复极点模长变为 1.1 时，见红色曲线，脉冲响应变成振动幅度逐渐增大的信号，响应越来越大；系统不稳定，小扰动会无限放大。当复极点模长变为 0.9 时，见绿色曲线，脉冲响应变成振荡幅度逐渐减小的信号，响应慢慢趋近于 0；系统稳定，响应最终消失，具有不错的收敛性。

## 2.6 重极点问题

系统的差分方程表示为：

$$y(k+2) - 1.8y(k+1) + 0.81y(k) = 3u(k+1) - 1.2u(k)$$

采样周期  $T_s=1s$ , 做  $G(z)/z$  的部分分式展开, 用得到极点 pole 和对应 residue 分别写出 (不是编程实现) 其单位 delta 响应 (写成三个离散时间变量 k 的函数和的形式)。并画出重极点对应的单位 delta 响应图。

提示:  $G(z)=Az/(z-p)^2+Bz/(z-p)+\text{其他项}$

$$g(k)=Ak^{k-1}+Bk^0+\text{其他项}$$

根据题目给出的差分方程  $y(k+2) - 1.8y(k+1) + 0.81y(k) = 3u(k+1) - 1.2u(k)$ , 对系统两边做 Z 变换, 根据延迟性质  $Z\{y(k+n)\} = z^n Y(z)$ , 可得

$$z^2 Y(z) - 1.8z Y(z) + 0.81 Y(z) = 3z U(z) - 1.2 U(z),$$

移项得到

$$Y(z)(z^2 - 1.8z + 0.81) = U(z)(3z - 1.2),$$

所以系统传递函数为

$$G(z) = \frac{Y(z)}{U(z)} = \frac{3z - 1.2}{z^2 - 1.8z + 0.81}$$

求解特征方程的根 (系统极点) ,

$$z^2 - 1.8z + 0.81 = 0$$

解得  $z = 0.9$  (重根)。

该系统是一个具有双重极点的系统, 极点位于  $z = 0.9$ 。通过部分分式展开,

$$G(z) = -\frac{40}{27} + \frac{40}{27} \cdot \frac{z}{z - 0.9} + \frac{5}{3} \cdot \frac{z}{(z - 0.9)^2}$$

通过如下代码画出单位冲激响应。**G1** 为常数项, 对应系统中不含  $z$  的项; **G2** 为一阶极点项, **G3** 为二阶极点项。**G = G2 + G3** 为系统传递函数, 不包括常数项, 因为常数项在单位脉冲响应中为  $k = 0$  时有瞬时响应, 但不是系统的因果部分, 所以通常省略只保留延迟项。**[y, t] = impulse(G, N);** 这个 **impulse** 函数就是在计算系统的单位脉冲响应 (**delta** 响应), 对应离散信号输入为  $\delta(k)$ 。

根据 Z 反变换公式,

$$\begin{aligned} Z^{-1}\left\{\frac{z}{z-a}\right\} &= a^k \cdot u(k), \\ Z^{-1}\left\{\frac{z}{(z-a)^2}\right\} &= (k+1) \cdot a^k \cdot u(k), \end{aligned}$$

将  $a = 0.9$  代入, 对于  $\frac{5}{3} \cdot \frac{z}{(z-0.9)^2}$  得到  $g_1(k) = \frac{5}{3}(k+1) \cdot 0.9^k$ , 对于  $\frac{40}{27} \cdot \frac{z}{z-0.9}$  得到  $g_2(k) = \frac{40}{27} \cdot 0.9^k$ 。将两部分相加得到

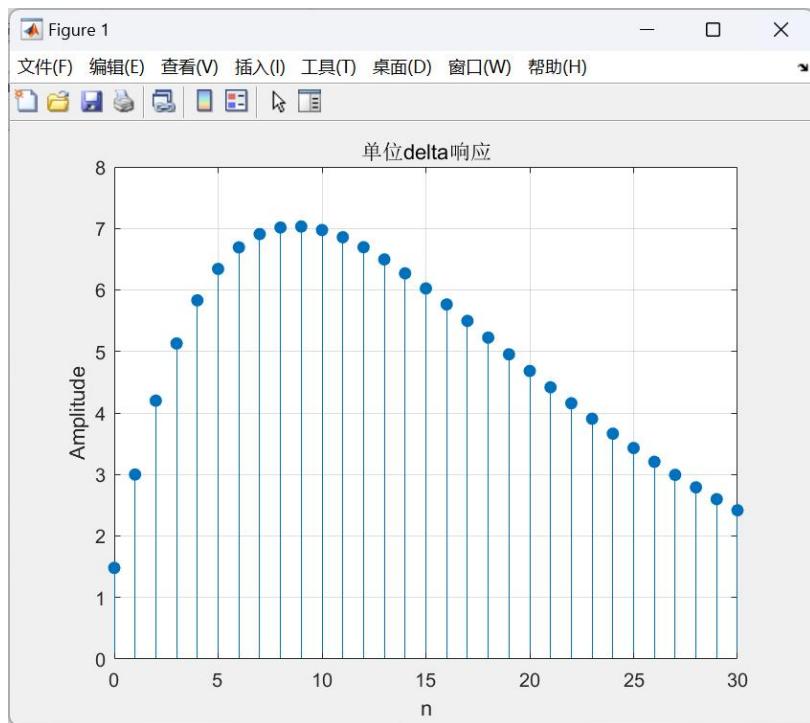
$$g(k) = \left[ \frac{5}{3}(k+1) + \frac{40}{27} \right] \cdot 0.9^k \cdot u(k)$$

```

Ts = 1;
z = tf('z', Ts);
G1 = -40/27;
G2 = (40/27) * z / (z - 0.9);
G3 = (5/3) * z / (z - 0.9)^2;
G = G2 + G3;
N = 30;
[y, t] = impulse(G, N);
stem(t, y, 'filled');
title('单位 delta 响应');
xlabel('n'); ylabel('Amplitude');
grid on;

```

得到结果如下图所示。该系统极点在 0.9，说明系统是稳定但响应较慢衰减，对应结果图中前几个点是主贡献项且后续逐渐衰减。



## 2.7 $G(z)$ 的阶跃响应

对于式(2.5)中给出的传递函数  $G(z)$ ，对  $G(z)$ 增加极点  $z=1$  和零点  $z=0$ ，并用 impulse 命令绘制响应图（等价于  $G(z)$ 的阶跃响应），同时直接用阶跃命令获得阶跃响应，两者得到的结果进行对比。采样周期  $Ts=1$  s。

$$G(z) = \frac{2z^2 - 2.2z + 0.56}{z^3 - 0.6728z^2 + 0.0463z + 0.4860} \quad (2.5)$$

设置采样周期  $Ts$  为 1 秒，并定义离散传递函数  $G(z)$ 。为了通过冲激响应间接获取系统的阶跃响应，添加极点  $z=1$  和零点  $z=0$  构造等效系统  $G_{eq}(z) = \frac{z}{z-1}G(z)$ ，本质上是将  $G(z)$ 与单位

阶跃输入的  $z$  变换  $\frac{1}{1-z^{-1}}$  相乘，从而实现阶跃输入对原始系统的响应。现补充单位阶跃输入的  $z$  变换公式推导如下，单位阶跃函数时域表达式为

$$f(nT) = 1; (n \geq 0)$$

相应数列为：{1, 1, 1, 1, ...}，因此，单位阶跃函数的  $Z$  变换为：

$$F(z) = 1 + z^{-1} + z^{-2} + z^{-3} + \dots$$

上式可以看作一个非零次的无限求和式，可得

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

分别利用 `impulse` 函数对等效系统求出其单位冲激响应，即模拟的阶跃响应；并使用 `step` 函数对原始系统直接求出其阶跃响应。随后将两种响应结果进行可视化展示，以便进行直观比较。除了定性分析，定量地计算两种响应结果之间的误差范数  $L_2$  范数以量化它们之间的数值差异。

```
clc; clear; close all;

% 设置采样时间
Ts = 1;
z = tf('z', Ts);

% 定义原始离散系统 G(z)
num1 = [2 -2.2 0.56];
den1 = [1 -2.2 0.56];
Gz = tf(num1, den1, Ts);

% 构造通过 impulse 模拟阶跃响应的等效系统: G(z)/1/(1 - z^-1)
num_eq = conv(num1, [1 0]); % 分子乘 z
den_eq = conv(den1, [1 -1]); % 分母乘 (1 - z^-1)
Gz_eq = tf(num_eq, den_eq, Ts);

% 设置仿真时间范围
N = 20;
t = 0:Ts:N;

% 分别计算系统的单位冲激响应和阶跃响应
[resp_impulse, t1] = impulse(Gz_eq, t);
[resp_step, t2] = step(Gz, t);

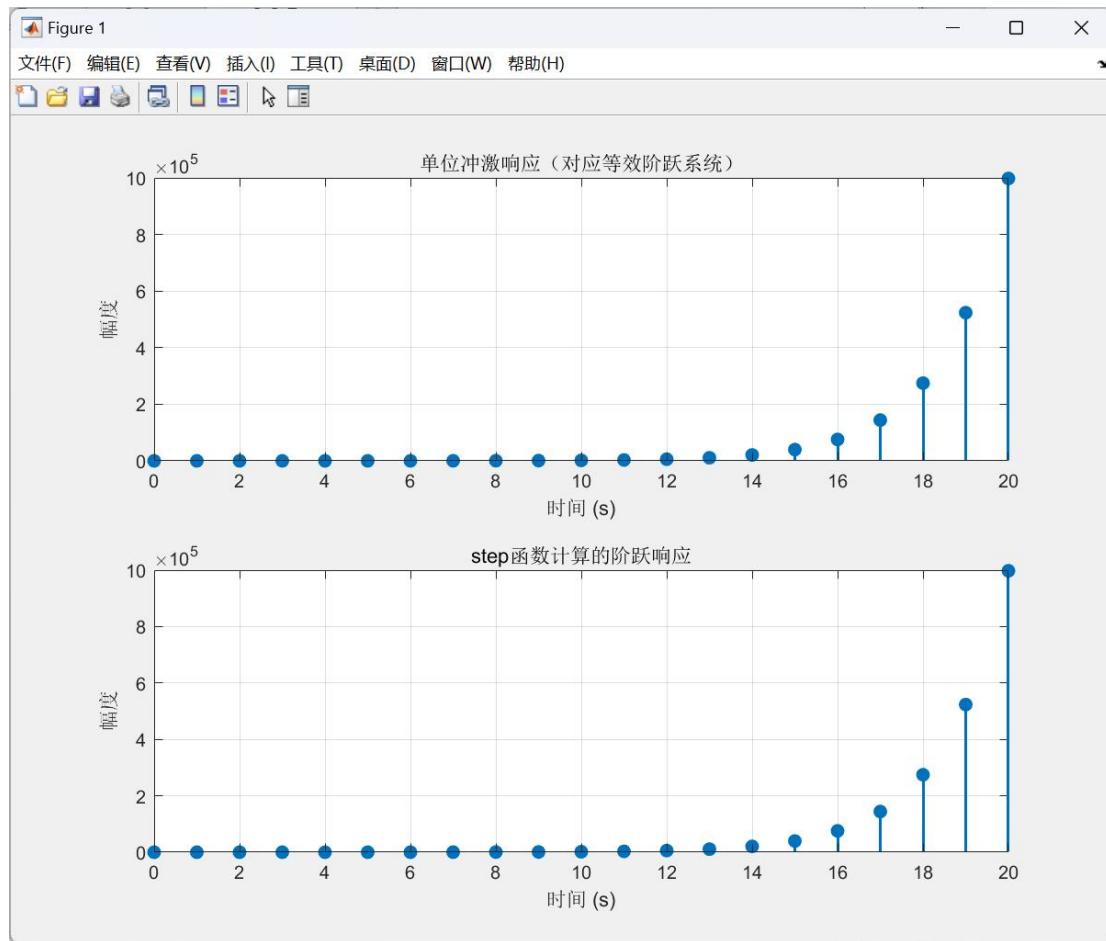
% 可视化结果
figure('Position', [100 100 800 600])

subplot(2,1,1);
stem(t1, resp_impulse, 'filled', 'LineWidth', 1.5);
title('单位冲激响应 (对应等效阶跃系统)');
xlabel('时间 (s)'); ylabel('幅度'); grid on;

subplot(2,1,2);
stem(t2, resp_step, 'filled', 'LineWidth', 1.5);
title('step 函数计算的阶跃响应');
xlabel('时间 (s)'); ylabel('幅度'); grid on;
```

```
% 输出系统信息与误差指标
disp('== 系统信息对比 ==');
disp('原始传递函数 G(z):'); disp(Gz);
disp('等效传递函数 G_step_equiv(z):'); disp(Gz_eq);

diff_norm = norm(resp_impulse - resp_step(1:length(resp_impulse)));
disp(['两种方式结果差异 (L2 范数) : ', num2str(diff_norm)]);
```



输出部分如下。

== 系统信息对比 ==

原始传递函数 G(z):

tf - 属性:

Numerator: {[2 -2.2000 0.5600]}

Denominator: {[1 -2.2000 0.5600]}

Variable: 'z'

IODelay: 0

InputDelay: 0

OutputDelay: 0

InputName: {}

```
InputUnit: {}
InputGroup: [1×1 struct]
OutputName: {}
OutputUnit: {}
OutputGroup: [1×1 struct]
Notes: [0×1 string]
UserData: []
Name: "
Ts: 1
TimeUnit: 'seconds'
SamplingGrid: [1×1 struct]
```

等效传递函数 G\_step\_equiv(z):

tf - 属性:

```
Numerator: {[2 -2.2000 0.5600 0]}
Denominator: {[1 -3.2000 2.7600 -0.5600]}
Variable: 'z'
IODelay: 0
InputDelay: 0
OutputDelay: 0
InputName: {}
InputUnit: {}
InputGroup: [1×1 struct]
OutputName: {}
OutputUnit: {}
OutputGroup: [1×1 struct]
Notes: [0×1 string]
UserData: []
Name: "
Ts: 1
TimeUnit: 'seconds'
SamplingGrid: [1×1 struct]
```

两种方式结果差异 (L2 范数) : 1.2889e-09

由于数学上这两种方法是等价的，因此误差应非常小，从而验证了冲激响应与阶跃响应之间的数学关系。本小题展示了离散时间系统中单位冲激响应与阶跃响应的关系，验证了冲激响应通过与单位阶跃信号的卷积可以重建出完整的阶跃响应，是离散系统分析中的一个重要技巧和验证手段。

## 2.8 任意输入的响应

用 lsim 命令仿真，式(2.5)中系统的零状态响应，输入为：

$$u(k) = \begin{cases} 0, & k < 0 \\ 2, & 0 \leq k < 10 \\ 0.5, & k \geq 10 \end{cases}$$

总的响应时间间隔是  $0 \leq k \leq 40$ ，采样周期  $T_s=0.2\text{s}$ 。（提示：ones, find, lsim 等命令）

```

Ts = 0.2; % 采样周期
t = 0:Ts:40; % 时间向量

% 设定输入信号 u(k)
u = zeros(size(t));
u(t >= 0 & t < 10) = 2;
u(t >= 10) = 0.5;

% 定义系统传递函数 G(z)
num = [2 -2.2 0.56];
den = [1 -0.6728 0.0463 0.4860];
G = tf(num, den, Ts); % 定义传递函数 G(z)

% 使用 lsim 计算零状态响应
[y, t_out] = lsim(G, u, t);

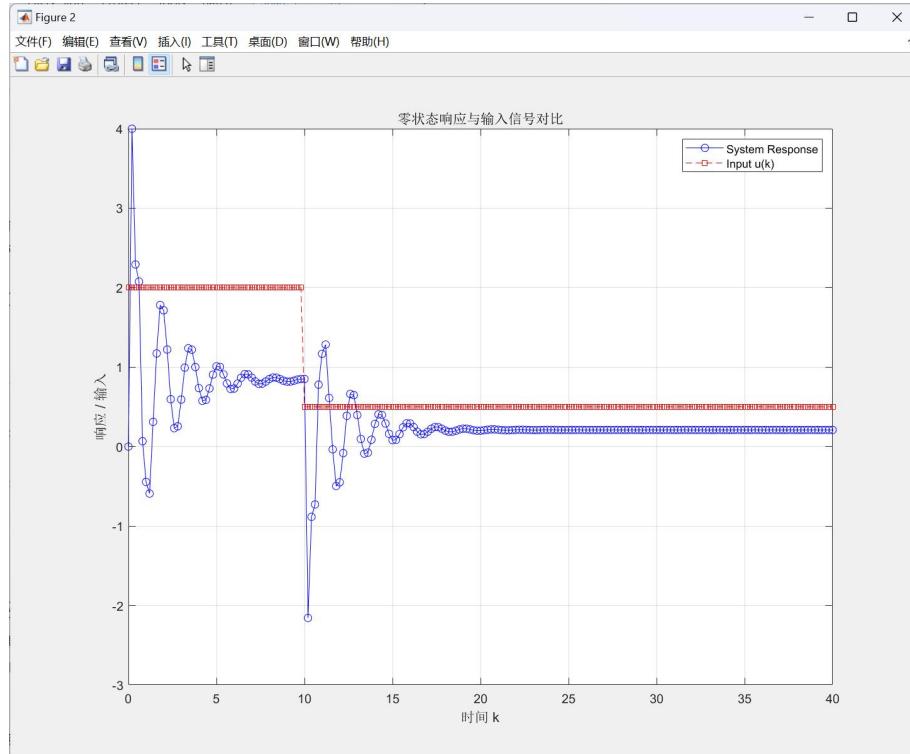
% 绘制响应
figure;
plot(t_out, y, 'b-o', 'DisplayName', 'System Response');
hold on;
plot(t, u, 'r--s', 'DisplayName', 'Input u(k)');
xlabel('时间 k');
ylabel('响应 / 输入');
title('零状态响应与输入信号对比');
legend;
grid on;

```

上述代码，先明确采样周期为  $0.2$  秒，时间间隔是  $0$  到  $40$ 。随后按照输入公式设置输入信号。`u = zeros(size(t));` 将信号初始化为  $0$ ；`u(t >= 0 & t < 10) = 2`；修改  $0 \leq k < 10$  对应的数值为  $2$ ，修改  $k \geq 10$  对应的数值为  $0.5$ 。然后，通过指定分子和分母多项式系数，定义了一个三阶的离散系统传递函数  $G(z)$ 。

系统响应的计算使用了 `lsim` 函数，该函数用于模拟连续或离散系统对任意输入信号的时域响应，名字来源是 `Linear System simulation`，其能够计算线性动态系统在给定输入信号和初始条件下的响应。`[y, t_out, x] = lsim(sys, u, t, x0)` 其中 `sys` 为系统模型对象；`u` 为输入信号向量或矩阵，每列代表一个输入通道；`t` 为时间向量，其长度必须与 `u` 的行数一致；`x0` 可选，其表示系统的初始状态，默认为零状态；`y` 为系统响应输出向量，每列对应一个输出通道；`t_out` 为与 `y` 对应的时间向量；`x` 可选，表示系统的状态变量随时间的演化。

最终，以图形方式展示了系统响应  $y(k)$  和输入信号  $u(k)$  随时间的变化情况，便于观察系统对该分段输入的动态响应行为。



## 2.9 极点和系统的稳定性

找出如下闭环传递函数：

$$G(z) = \frac{3z^2 + 1.8z + 1.08}{z^4 - 1.25z^3 + 0.495z^2 - 0.0035z - 0.1862}$$

的极点并确定系统是否稳定。对分母多项式应用 `roots` 命令或对系统应用 `pole` 命令的方法确定极点。在 Z 平面上画出  $G(z)$  的极点和零点。最后用系统的单位冲激响应的方法验证系统的稳定性。单位采样周期。

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

```
num = [3 1.8 1.08];
den = [1 -1.252 0.495 -0.0035 -0.1862];
Ts = 1;
```

```

G = tf(num, den, Ts);

poles = roots(den);
zeros_ = roots(num);

disp('极点 (poles) :');
disp(poles);
disp('零点 (zeros) :');
disp(zeros_);

is_stable = all(abs(poles) < 1);
if is_stable
    disp('系统稳定：所有极点都在单位圆内。');
else
    disp('系统不稳定：存在极点在单位圆外或在圆上。');
end

figure;
zplane=zeros_, poles);
title('G(z) 的极点与零点图');
grid on;

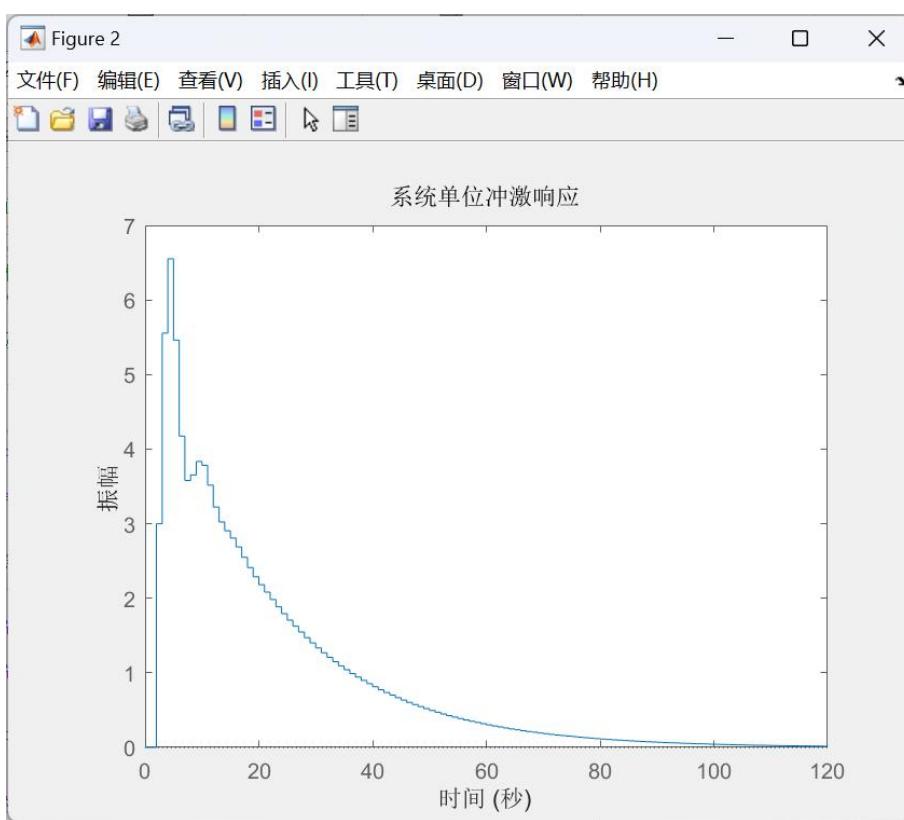
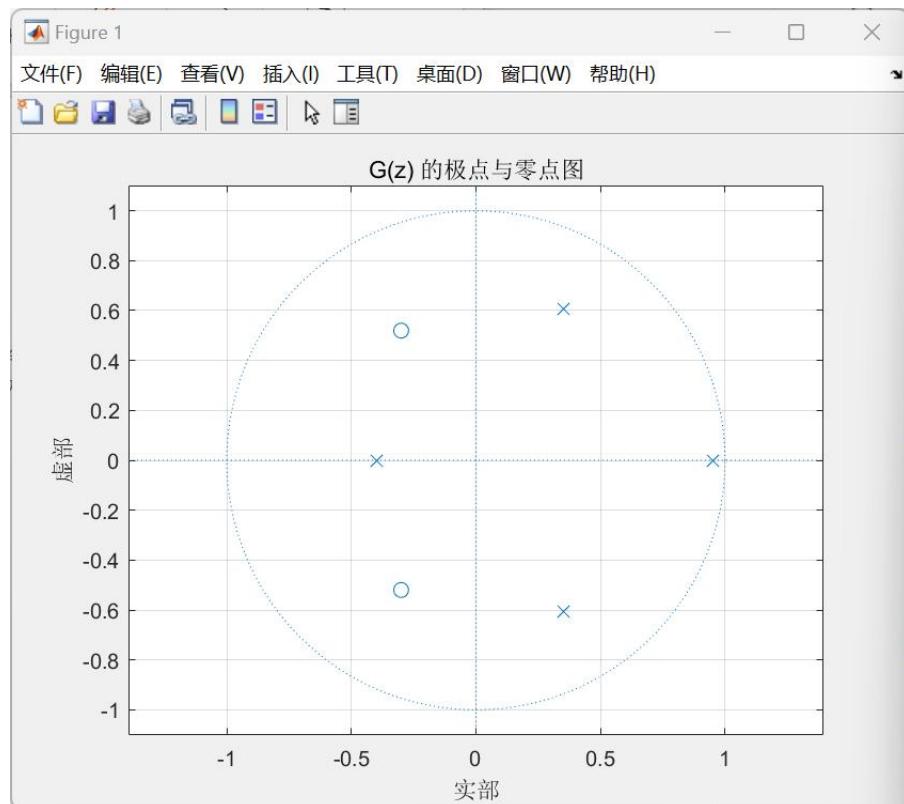
figure;
impulse(G);
title('系统单位冲激响应');

```

使用 `num = [3 1.8 1.08];` 定义分子系数，对应  $B(z) = 3z^2 + 1.8z + 1.08$ ；使用 `den = [1 -1.252 0.495 -0.0035 -0.1862];` 定义分母系数，对应  $A(z) = 1 - 1.252z^{-1} + 0.495z^{-2} - 0.0035z^{-3} - 0.1862z^{-4}$ ；采样周期设置为 1 秒；使用 `G = tf(num, den, Ts);` 构造离散时间传递函数  $G(z) = \frac{B(z)}{A(z)}$ 。

首先，使用 `roots()` 求解多项式根，极点为分母的根，零点为分子的根。随后，判断系统稳定性，即判断所有极点是否在单位圆内，`abs(poies) < 1` 检查每个极点模是否小于 1，`all(...)` 表示所有极点都满足条件系统才稳定。绘制 Z 平面的极点零点图，可直观判断系统是否稳定，若极点在单位圆内则系统稳定，若极点在圆上则边界稳定，若极点在圆外则系统不稳定。最后，用系统的单位冲激响应的方法验证系统的稳定性，使用 `impulse(G);` 绘制单位冲激响应，若响应随时间收敛至零，则系统行为稳定。

得到可视化结果如下。从 Figure1 可以看出每个极点都在单位圆内，则系统稳定。从 Figure2 可以看出响应随时间收敛至零，则系统行为稳定。



输出部分如下。

极点 (poles) :

$0.9517 + 0.0000i$   
 $0.3501 + 0.6055i$   
 $0.3501 - 0.6055i$   
 $-0.3999 + 0.0000i$

```
零点 (zeros) :  
-0.3000 + 0.5196i  
-0.3000 - 0.5196i
```

系统稳定：所有极点都在单位圆内。

## 2.10 零点

**例 2.10 零点（零点不影响系统的稳定性，但会影响模态函数的幅值，阻碍输入信号的传送）**

建立系统的 ZPK 对象，传递函数表示为：

$$G(z) = \frac{(z+0.3)(z-e^{j0.5})(z-e^{-j0.5})}{(z-0.9)(z-0.7)^2(z+0.5)}$$

然后提取分子多项式并用 roots 命令来计算零点，验证它们与 G(z) 中零点的一致性。最后，Ts=1s 的情况下，输入 u(k)=sin(0.5k)，0≤k≤50 时，画出系统的输出 y(k)。解释输入是正弦曲线，为什么输出不包含任何正弦曲线的成分（提示：系统输出信号 z 变换）。如果换成 u(k)=sin(0.2k)，结果会如何？

本题解答思路如下。使用零极点增益（ZPK）形式定义一个离散系统，将系统转换为分子-分母（TF）形式；输入两种不同频率的正弦信号，观察系统的时域响应；可视化输入输出关系，比较系统的频率响应特性。

```
clc; clear; close all;  
  
% 定义系统的零点和极点  
zeros = [-0.3, exp(1j*0.5), exp(-1j*0.5)]; % 包括一个实零点和一对共轭复数零点  
poles = [0.9, 0.7, 0.7, -0.5]; % 包括两个重复的极点和一个负极点  
gain = 1; % 系统增益  
  
% 构造 ZPK (零极点增益) 模型，采样时间为 1 (即离散系统)  
sys_zpk = zpk=zeros, poles, gain, 1);  
  
% 将 ZPK 模型转换为传递函数的分子和分母系数向量  
[num, den] = tfdata(sys_zpk, 'v');  
  
% 通过分子多项式求解系统的零点，用于验证  
calc_zeros = roots(num);  
  
% 显示零点信息  
disp('ZPK 对象指定的零点:');  
disp=zeros'); % 转置成列向量输出  
  
disp('通过分子多项式计算的零点:');  
disp(calc_zeros);
```

```

% 定义时间序列 k, 从 0 到 50
k = 0:50;

% 构造两个不同频率的输入信号
u1 = sin(0.5*k); % 输入信号 1: 频率较高
u2 = sin(0.2*k); % 输入信号 2: 频率较低

% 分别对两个输入信号进行离散系统的响应仿真
y1 = lsim(sys_zpk, u1, k); % 系统对 u1 的响应
y2 = lsim(sys_zpk, u2, k); % 系统对 u2 的响应

% 绘制系统响应图像
figure;

% 第一个子图: 输入 u1 和其系统响应
subplot(2,1,1);
stem(k, u1, 'b'); hold on; % 蓝色离散点表示输入信号
stem(k, y1, 'r', 'LineWidth', 1.5); % 红色离散点表示输出响应
title('输入 u(k)=sin(0.5k)的响应');
legend('输入信号', '输出响应');
xlabel('采样时刻 k'); ylabel('幅值');
grid on;

% 第二个子图: 输入 u2 和其系统响应
subplot(2,1,2);
stem(k, u2, 'b'); hold on; % 蓝色离散点表示输入信号
stem(k, y2, 'r', 'LineWidth', 1.5); % 红色离散点表示输出响应
title('输入 u(k)=sin(0.2k)的响应');
legend('输入信号', '输出响应');
xlabel('采样时刻 k'); ylabel('幅值');
grid on;

```

输出部分如下:

ZPK 对象指定的零点:

```

-0.3000 + 0.0000i
0.8776 - 0.4794i
0.8776 + 0.4794i

```

通过分子多项式计算的零点:

```

0.8776 + 0.4794i
0.8776 - 0.4794i
-0.3000 + 0.0000i

```

从上述输出部分, 可得提取分子多项式并用 roots 命令得到得零点与 G(z)中所得到的零点一致。

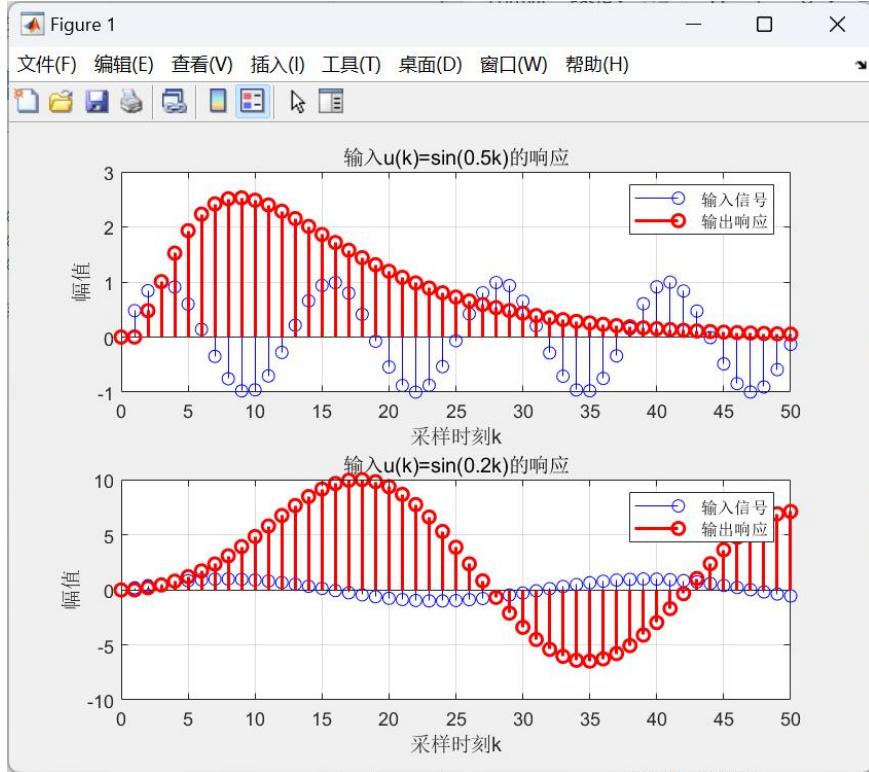
从下图的子图 1 可看出, 当输入为正弦信号  $u(k)=\sin(0.5k)$ 时, 输出不包含正弦曲线的成分, 这一现象可以通过系统的零点分布解释。具体而言, 当正弦信号的频率为  $w=0.5 \text{ rad/sample}$  时, 对应的单位圆上的复数表示为  $z = e^{\pm j0.5}$ 。如果系统的零点刚好位于这些位置, 即

$$G(z) = 0 \text{ (当 } z = e^{\pm j0.5})$$

那么, 即使输入在该频率处有能量, 也会因为乘以零而被抑制, 导致

$$Y(z) = G(z) \cdot U(z) = 0 \text{ (当 } z = e^{\pm j0.5})$$

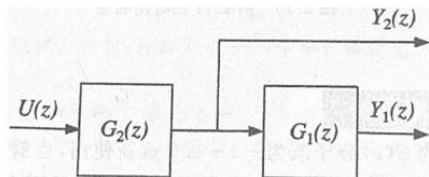
这意味着系统阻断了该频率信号的传输，输入信号的该频率成分无法传输至输出，导致输出不包含正弦成分。



当输入频率为 0.2 时，可见上图的子图 2，输出仍为同频率正弦波，因为该频率未被零点阻断；但受到系统频率响应的影响，幅值和相位发生了明显的变化。

## 2.12 单输入双输出系统

下图所示的离散时间系统有一个输入  $u(k)$  和两个输出  $y_1(k)$  与  $y_2(k)$ 。



$$\text{其中 } G_2(z) = \frac{Y_2(z)}{U(z)}, \quad G_1(z) = \frac{Y_1(z)}{Y_2(z)}, \quad G_1(z) = \frac{0.06059}{z - 0.9394}, \quad G_2(z) = \frac{0.2212}{z - 0.7788}$$

确定一个输入  $u(k)$  和两个输出  $y_1(k)$  与  $y_2(k)$  之间的两个传递函数。假设采样周期  $T_s=0.05$  s，画出单位阶跃响应作用下两个输出的响应。从复合系统的输出里提取各自的响应。

**提示：**  $G=[G1;1]*G2;$  %单输入两输出

$Y=step(G,dtime);$

$Plot(dtime,y(:,1),'o',dtime,y(:,2),'x');$

$Ts = 0.05;$  %采样周期

```

% 定义 G1(z) 和 G2(z)
G1 = tf(0.06059, [1 -0.9394], Ts);
G2 = tf(0.2212, [1 -0.7788], Ts);

% 构建复合系统 G = [G1*G2; G2]， 单输入双输出
G = [G1 * G2; G2];

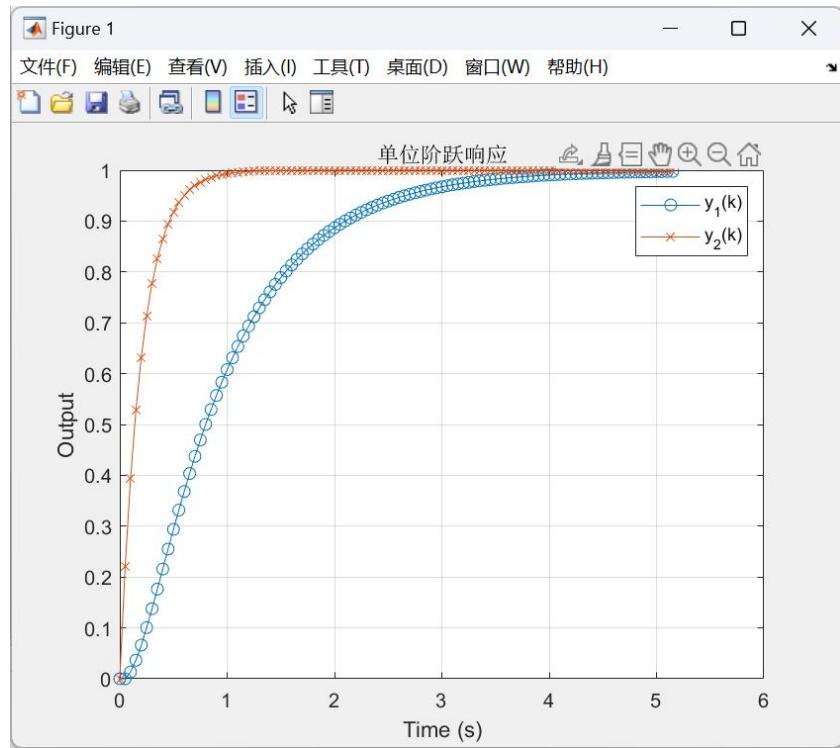
% 计算单位阶跃响应
[Y, T] = step(G);

% 绘图
plot(T, Y(:,1), 'o-', T, Y(:,2), 'x-');
xlabel('Time (s)');
ylabel('Output');
legend('y_1(k)', 'y_2(k)');
title('单位阶跃响应');
grid on;

```

通过 `tf(num, den, Ts)` 定义离散时间传递函数  $G1(z)$  和  $G2(z)$ 。 $G1(z)$  是一个一阶递归滤波器，极点位于  $z = 0.9394$ ，在单位圆内，系统稳定； $G2(z)$  也是一阶滤波器，极点  $0.7788$ ，同样稳定。构造单输入双输出系统， $G1 * G2$  表示串联连接，也即  $G1(z) \times G2(z)$ ； $G = [G1 * G2; G2]$  构成一个多输出系统，第一输出通道为  $y_1(k) = G1(z)G2(z) * u(k)$ ，第二输出通道为  $y_2(k) = G2(z) * u(k)$ 。输入  $u(k)$  是单位阶跃，系统同时输出  $y_1(k)$  和  $y_2(k)$ ，构成一个“共享输入、不同路径”的并联系统。`step(G)` 会对系统施加一个单位阶跃输入，返回两个变量， $Y$  每一行是对应时间点的系统输出，列表示不同输出通道； $T$  为时间向量，与  $Ts$  一致的离散时间轴。

绘图部分，`plot(...)` 绘制两个输出信号随时间的响应曲线； $Y(:,1)$  是第一个输出通道  $y_1(k)$ ， $Y(:,2)$  是第二个输出通道  $y_2(k)$ ；'o-'，'x-' 分别为实线带圆点、带叉号的样式，便于区分；添加标题、坐标轴标签、图例、网格，以增强可读性。



$y_1(k)$  经过了两个串联系统  $G_1$  和  $G_2$ , 响应更“迟缓”, 波形上升更慢;  $y_2(k)$  只经过  $G_2$ , 响应更快。因此, 上图结果中,  $y_2(k)$  早于  $y_1(k)$  收敛到稳态。

### 3.3 串/并联

图 3.4 中四个子系统的采样周期都为  $T_s=1s$ 。其中:

$$G_1(z) = \frac{0.02(2z+1.4)}{z^2 - 1.7z + 0.72}, \quad G_2(z) = \frac{z+0.2}{(z+0.8)(z-0.9e^{\frac{\pi}{5}j})(z-0.9e^{-\frac{\pi}{5}j})}$$

$$G_3(z) = \frac{z+0.5}{z+0.75}, \quad G_4(z) = \frac{0.15z}{(z-0.9)(z-0.8)(z+0.8)}$$

用\*和+算子表示整体传递函数, 并验证整体传递函数的极点是单个传递函数极点的合并。并验证整个传递函数的零点不包含  $G_1(z)$ 、 $G_2(z)$  或  $G_3(z)$  的零点, 但包含了  $G_4(z)$  的零点。画出组合后系统的阶跃响应。

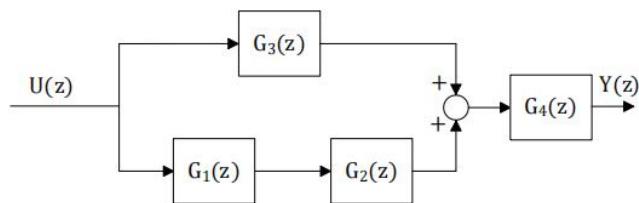


图 3.4 四个子系统的串/并联

```
clc; clear; close all;

Ts = 1; % 采样周期

% 构建子系统 H1(z)
```

```

num_G1 = 0.02 * [2 1.4];
den_G1 = [1 -1.7 0.72];
G1 = tf(num_G1, den_G1, Ts, 'Variable', 'z');

% 构建子系统 H2(z), 包含复共轭极点
angle_rad = pi / 5;
p_G2 = [-0.8; 0.9 * exp(1j * angle_rad); 0.9 * exp(-1j * angle_rad)];
num_G2 = [1 0.2];
den_G2 = poly(p_G2);
G2 = tf(num_G2, den_G2, Ts, 'Variable', 'z');

% 构建子系统 H3(z), 简单一阶系统
G3 = tf([1 0.5], [1 0.75], Ts, 'Variable', 'z');

% 构建子系统 H4(z), 含 z 项, 三阶多项式极点
p_G4 = [0.9; 0.8; -0.8];
G4 = tf(0.15 * [1 0], poly(p_G4), Ts, 'Variable', 'z');

% 系统连接: 串联 + 并联结构
sys_series = series(G1, G2); % H1 串联 H2
sys_parallel = parallel(sys_series, G3); % 与 H3 并联
Sys_all = series(sys_parallel, G4); % 并联后再串联 H4

% 输出极点信息
disp('==== 各子系统极点 =====');
disp('G1 的极点:'); disp(pole(G1));
disp('G2 的极点:'); disp(pole(G2));
disp('G3 的极点:'); disp(pole(G3));
disp('G4 的极点:'); disp(pole(G4));
disp('整体系统极点:'); disp(pole(Sys_all));

% 输出零点信息
disp('==== 各子系统零点 =====');
disp('G1 的零点:'); disp(zero(G1));
disp('G2 的零点:'); disp(zero(G2));
disp('G3 的零点:'); disp(zero(G3));
disp('G4 的零点:'); disp(zero(G4));
disp('整体系统零点:'); disp(zero(Sys_all));

% 绘制阶跃响应图
figure;
step(Sys_all, 50);
title('整体系统阶跃响应');
xlabel('时间步 k');
ylabel('系统输出');
grid on;

% 绘制零极点图
figure;
pzmap(Sys_all);
title('整体系统的零极点图');
grid on;

```

首先设置采样周期为 1，意味着系统每秒采样一次。整个分析基于离散时间域。

随后，构建各个子系统。`G1 = tf(num_G1, den_G1, Ts, 'Variable', 'z');`构造系统 G1，其为二阶离散滤波器，具有两个极点和一个零点。

$$G_1(z) = \frac{0.04z + 0.028}{z^2 - 1.7z + 0.72}$$

```
p_G2 = [-0.8; 0.9 * exp(1j * angle_rad); 0.9 * exp(-1j * angle_rad)];
num_G2 = [1 0.2];
den_G2 = poly(p_G2);
G2 = tf(num_G2, den_G2, Ts, 'Variable', 'z');
```

构造系统 G2，其为含有共轭对的三阶系统。

$$G_2(z) = \frac{z + 0.2}{(z + 0.8)(z - 0.9e^{\frac{\pi}{5}j})(z - 0.9e^{-\frac{\pi}{5}j})}$$

```
G3 = tf([1 0.5], [1 0.75], Ts, 'Variable', 'z');
```

构造系统 G3，其有一个零点和一个稳定极点。

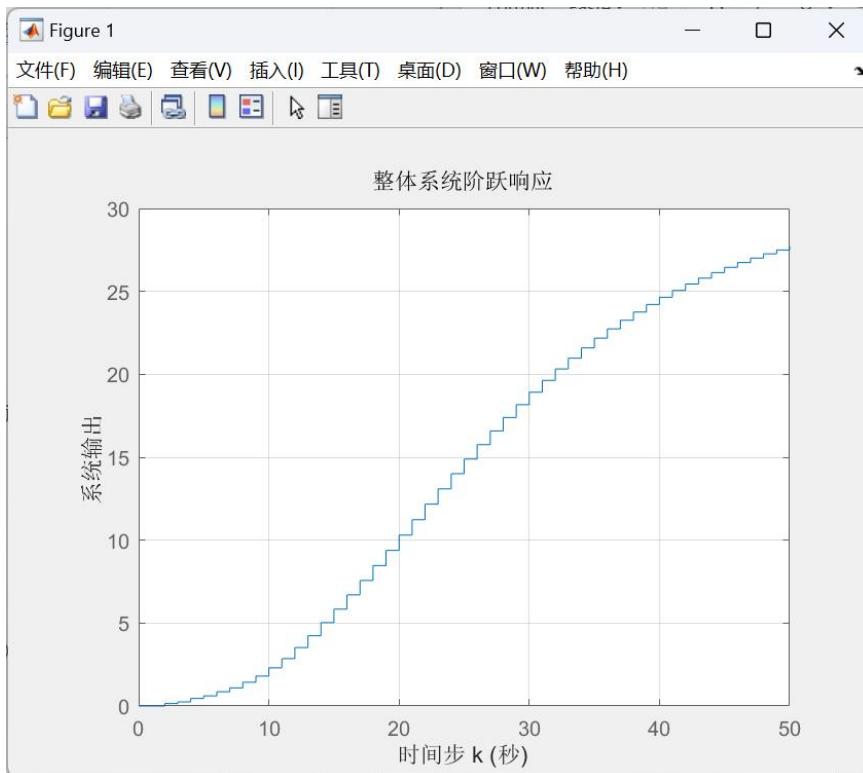
$$G_3(z) = \frac{z + 0.5}{z + 0.75}$$

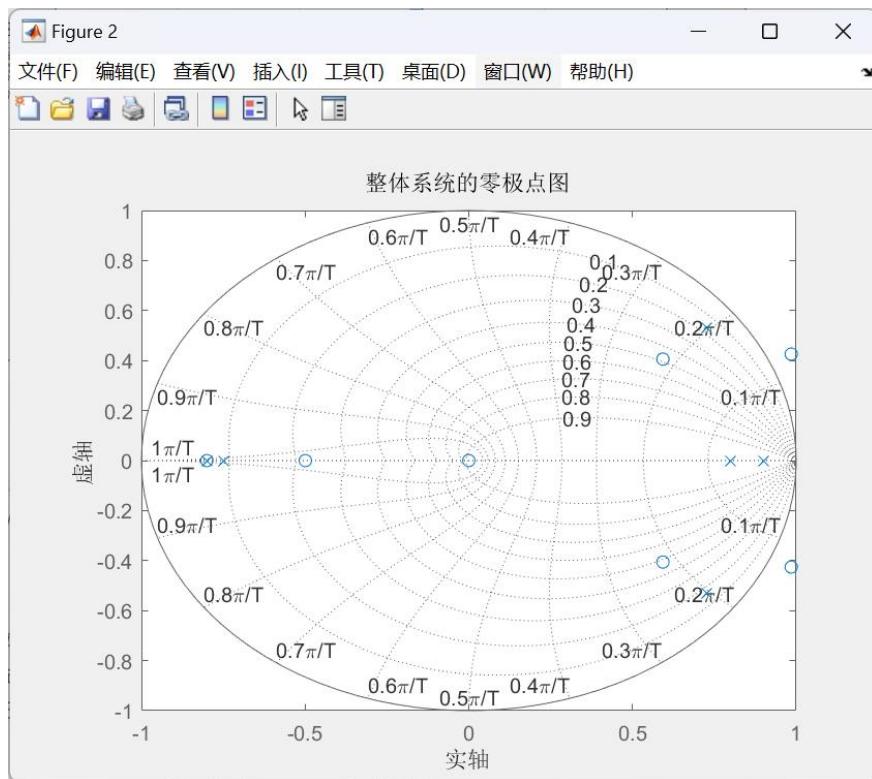
```
G4 = tf(0.15 * [1 0], poly(p_G4), Ts, 'Variable', 'z');
```

构造系统 G4，具有三个极点的三阶系统。

$$G_4(z) = \frac{0.15z}{(z - 0.9)(z - 0.8)(z + 0.8)}$$

随后，将系统结构组合。把 G1 和 G2 串联，再与 G3 并联，之后再串联 G4 得到总系统。使用 `pole()` 与 `zero()` 分析每个子系统的极点零点，判断稳定性。仿真输出响应和极点图，如下所示。





输出部分如下：

===== 各子系统极点 =====

G1 的极点：

0.9000  
0.8000

G2 的极点：

-0.8000 + 0.0000i  
0.7281 + 0.5290i  
0.7281 - 0.5290i

G3 的极点：

-0.7500

G4 的极点：

-0.8000  
0.9000  
0.8000

整体系统极点：

-0.8000 + 0.0000i  
-0.8000 + 0.0000i  
-0.7500 + 0.0000i  
0.7281 + 0.5290i  
0.7281 - 0.5290i  
0.9000 + 0.0000i  
0.9000 + 0.0000i  
0.8000 + 0.0000i  
0.8000 + 0.0000i

==== 各子系统零点 ====

G1 的零点:

-0.7000

G2 的零点:

-0.2000

G3 的零点:

-0.5000

G4 的零点:

0

整体系统零点:

0.0000 + 0.0000i

0.9849 + 0.4259i

0.9849 - 0.4259i

0.5930 + 0.4058i

0.5930 - 0.4058i

-0.8001 + 0.0000i

-0.4994 + 0.0000i

## 3.4 反馈联接

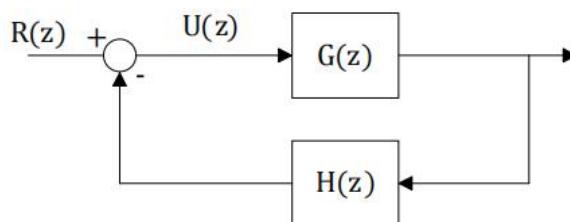
用 MATLAB 命令创建如图 3.6 所示的负反馈系统，其中

$$G(z) = \frac{0.2(z-0.85)}{(z-0.9e^{j\pi/4})(z-0.9e^{-j\pi/4})}, \quad H(z) = \frac{z-0.3}{(z-0.8)(z-0.1)}$$

单位采样周期。求出用多项式比形式表示的闭环传递函数并且确定它的零点、极点和

增益。将闭环传递函数  $T(z)$  的极点同  $G(z)$  和  $H(z)$  的极点进行比较。确定  $T(z)$  的稳定性。

验证  $T(z)$  的零点是由  $G(z)$  的零点和  $H(z)$  的极点组成的。



```
clear; clc; close all;
Ts = 1;
theta = pi/4; % π/4 相位角
pole_G = 0.9*[exp(1j*theta); exp(-1j*theta)]; % 复数极点
num_G = 0.2*[1 -0.85]; % 分子多项式
den_G = poly(pole_G); % 分母多项式
G = tf(num_G, den_G, Ts, 'Variable', 'z');
pole_H = [0.8; 0.1];
num_H = [1 -0.3];
```

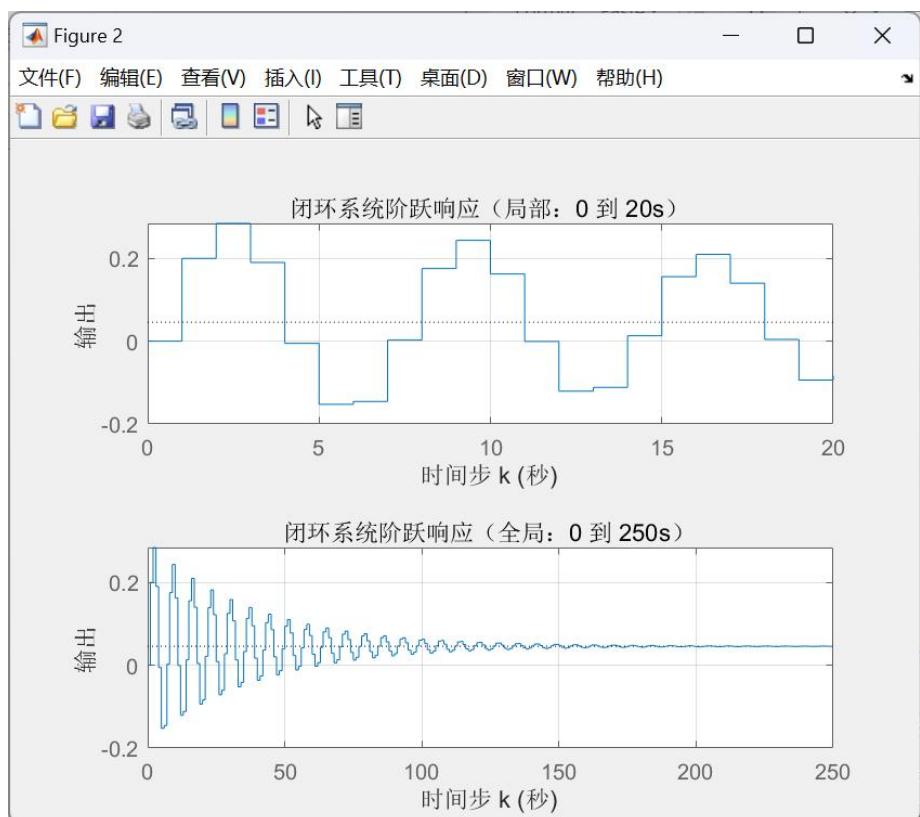
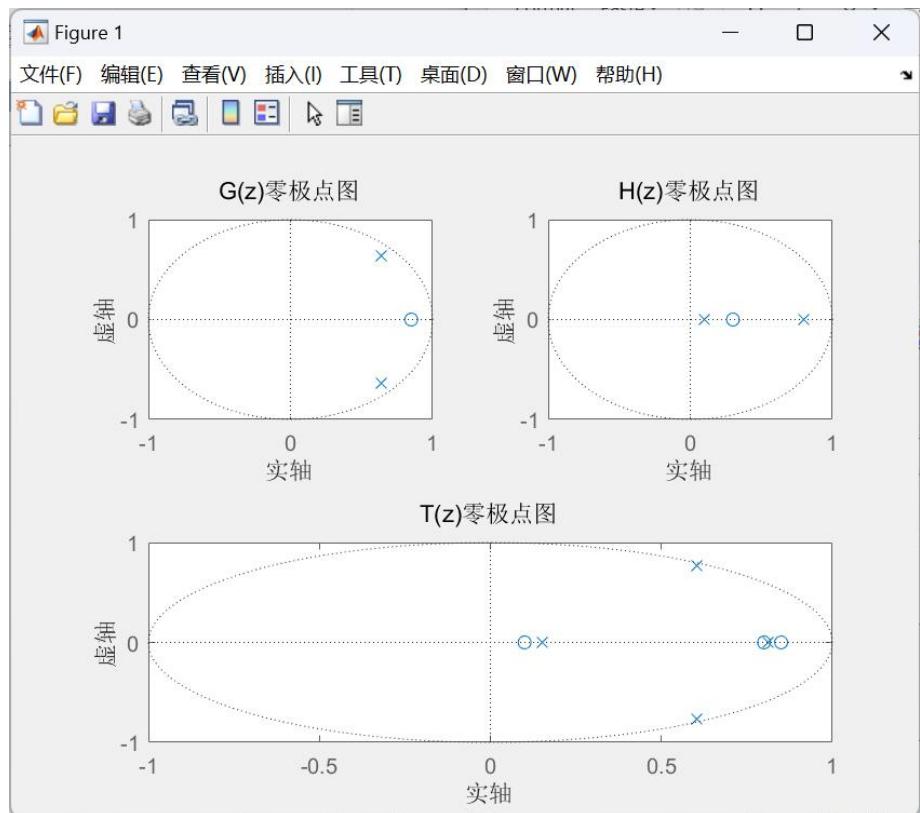
```

den_H = poly(pole_H);
H = tf(num_H, den_H, Ts, 'Variable', 'z');
T = feedback(G, H);
disp('==== 闭环传递函数 T(z) ====');
disp('多项式形式:');
disp(['分子: ', poly2str(T.num{1}, 'z')]);
disp(['分母: ', poly2str(T.den{1}, 'z')]);
disp('==== 系统特性 ====');
disp(['系统增益: ', num2str(dcgain(T))]);
disp('G(z)零点:'); disp(zero(G));
disp('T(z)零点:'); disp(zero(T));
disp('G(z)极点:'); disp(pole(G));
disp('H(z)极点:'); disp(pole(H));
disp('T(z)极点:'); disp(pole(T));
figure;
subplot(2,2,1);
pzmap(G);
title('G(z)零极点图');
subplot(2,2,2);
pzmap(H);
title('H(z)零极点图');
subplot(2,2,[3,4]);
pzmap(T);
title('T(z)零极点图');

figure;
%子图 1: 0-20s
subplot(2,1,1);
step(T, 0:1:20);
title('闭环系统阶跃响应（局部: 0 到 20s）');
xlabel('时间步 k');
ylabel('输出');
grid on;
%子图 2: 0-250s
subplot(2,1,2);
step(T, 0:1:250);
title('闭环系统阶跃响应（全局: 0 到 250s）');
xlabel('时间步 k');
ylabel('输出');
grid on;

```

构造了两个传递函数， $G(z)$ ，一个具有共轭复极点的二阶系统； $H(z)$ ，一个简单的实极点反馈路径。`T = feedback(G, H);`生成闭环系统  $T(z) = G / (1 + G*H)$ 。使用 `pole()` 和 `zero()` 分析系统的极点和零点。使用 `dcgain()` 获取静态增益（单位阶跃响应的稳态值）。使用 `pzmap` 绘制系统的极点零点分布图，验证系统稳定性。`step` 分别绘制局部响应（前 20 个采样时刻）与全局响应（长达 250 个采样时刻）。



输出部分如下：

```
==== 闭环传递函数 T(z) ====
多项式形式:
分子: 0.2 z^3 - 0.35 z^2 + 0.169 z - 0.0136
```

分母:  $z^4 - 2.1728 z^3 + 2.2355 z^2 - 1.0608 z + 0.1158$

==== 系统特性 ====

系统增益: 0.04588

G(z)零点:

0.8500

T(z)零点:

0.8500

0.8000

0.1000

G(z)极点:

0.6364 + 0.6364i

0.6364 - 0.6364i

H(z)极点:

0.8000

0.1000

T(z)极点:

0.6051 + 0.7631i

0.6051 - 0.7631i

0.8124 + 0.0000i

0.1503 + 0.0000i

从上述结果可以看出, T(z)是稳定的且 T(z)的零点是由 G(z)的零点和 H(z)的极点组成的。

### 3.5 双输入反馈系统

考虑带有参考输入和扰动输入的反馈系统, 如图 3.7(a)所示, 其中对象和传感器传递函数为

$$G(z) = \frac{0.0031z + 0.003}{z^2 - 1.9z + 0.905}, \quad H(z) = \frac{0.55}{z - 0.45}$$

采样周期为  $T_s=0.04s$ 。控制器是一个带有比例增益  $K_p=10$  和积分增益  $K_i=1/20s^{-1}$  积分控制器  $G_i(z) = K_p K_i \frac{T_s z}{z - 1}$ 。开发两个 MATLAB 模型, 一个模型的输入为参考输入  $R(z)$ , 另一个的输入为扰动输入  $D(z)$ , 然后确定各自的闭环零点和极点, 并在一张图中画出它们的单位阶跃响应。

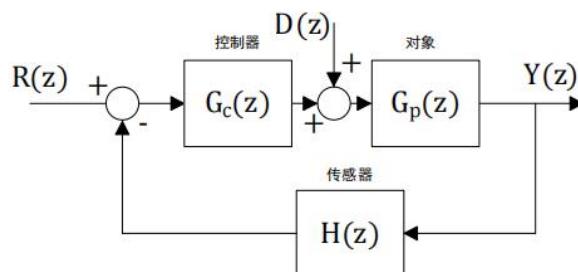


图 3.7 带有参考输入和扰动输入的反馈系统模块图

```

Ts = 0.04;

% 定义控制器传递函数 Gi(z)
Kp = 10;
Ki = 1/20;
num_Gi = Kp * Ki * Ts * [1 0]; % 0.02z/(z-1)
den_Gi = [1 -1];
Gi = tf(num_Gi, den_Gi, Ts, 'Variable', 'z');

% 定义对象传递函数 G(z)
num_G = [0.0031 0.003]; % 0.0031z + 0.003
den_G = [1 -1.9 0.905];
G = tf(num_G, den_G, Ts, 'Variable', 'z');

% 定义传感器传递函数 H(z)
num_H = 0.55;
den_H = [1 -0.45];
H = tf(num_H, den_H, Ts, 'Variable', 'z');

% 闭环传递函数: 参考输入 R(z) 到输出 Y(z)
YR = feedback(Gi * G, H);

% 闭环传递函数: 扰动输入 D(z) 到输出 Y(z)
YD = feedback(G, Gi * H);

% 显示零极点
disp('YR 的零点: ');
disp(zero(YR));
disp('YR 的极点: ');
disp(pole(YR));
disp('YD 的零点: ');
disp(zero(YD));
disp('YD 的极点: ');
disp(pole(YD));

% 绘制单位阶跃响应
figure;
step(YR, 'b', YD, 'r');
legend('参考输入 R(z) 的响应', '扰动输入 D(z) 的响应');
title('参考输入和扰动输入的单位阶跃响应');
grid on;

```

首先，设置采样时间  $T_s = 0.04$ 。定义控制器传递函数  $Gi(z)$ ，传递函数为

$$Gi(z) = K_p * K_i * T_s * \frac{z}{z - 1} = \frac{0.02z}{z - 1}$$

离散 PI 控制器，包含比例与积分控制。定义被控对象传递函数  $G(z)$ ，

$$G(z) = \frac{0.0031z + 0.003}{z^2 - 1.9z + 0.905}$$

定义传感器传递函数  $H(z)$ ，

$$H(z) = \frac{0.55}{z - 0.45}$$

$YR = feedback(Gi * G, H);$

计算参考输入到输出的闭环传递函数。 $YD = feedback(G, Gi * H);$  计算扰动输入到输出的闭环传递函数。随后，显示系统零极点，绘制单位阶跃响应曲线。

输出部分如下：

$YR$  的零点：

0  
-0.9677  
0.4500

YR 的极点:

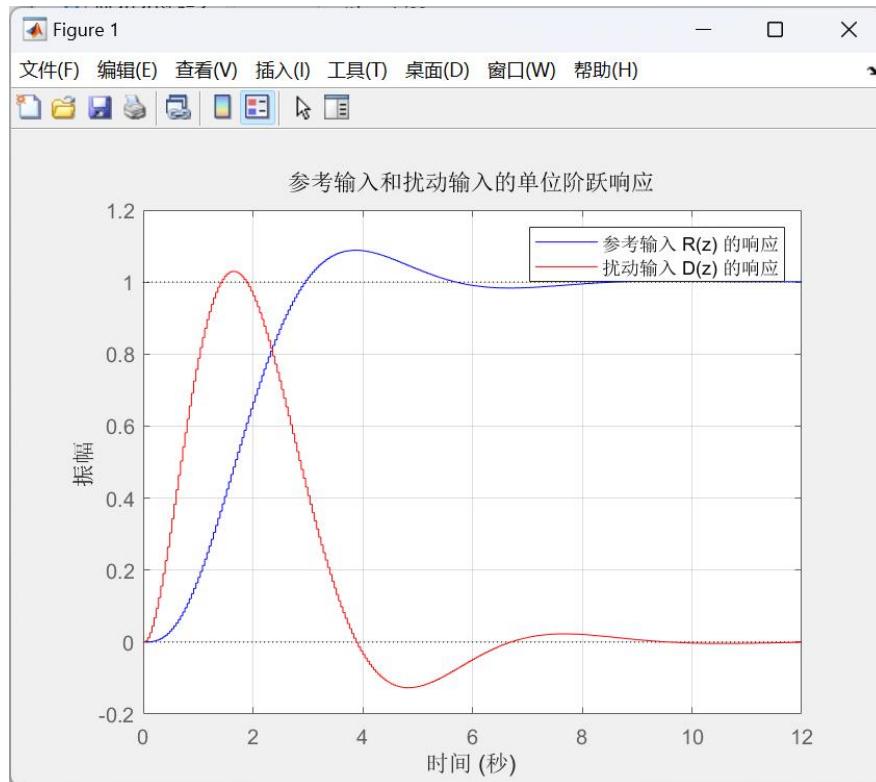
0.9748 + 0.0427i  
0.9748 - 0.0427i  
0.9503 + 0.0000i  
0.4502 + 0.0000i

YD 的零点:

-0.9677  
1.0000  
0.4500

YD 的极点:

0.9748 + 0.0427i  
0.9748 - 0.0427i  
0.9503 + 0.0000i  
0.4502 + 0.0000i



## 4.1 采样定理

$$e_1(t) = \sin(2\pi t), e_2(t) = \sin(18\pi t + 180^\circ), e_3(t) = \sin(22\pi t)$$

三种连续时间信号，用采样频率 10Hz 进行周期采样，试用 MATLAB 画出 t=0 到 1 秒的采样信号。分析发生了什么现象及原因？

画出所有连续时间信号。

如果采样周期变为 20Hz、30Hz 时，重复完成上述要求，并从频域解释产生上述现象的原因，可定性分析，也可通过画出信号的频谱进行分析。

```
clear; clc;

%% 连续时间信号
t_cont = 0:0.0001:1;
e1 = sin(2*pi*t_cont);
e2 = sin(18*pi*t_cont + pi);
e3 = sin(22*pi*t_cont);

%% 设置采样频率
Fs_list = [10, 20, 30];
colors = {'r', 'g', 'b'};

%% 创建图像窗口
figure('Position', [100, 100, 1200, 900]);

%% 连续信号（前三行）
subplot(6,3,1); plot(t_cont, e1, 'k'); title('e1 连续');
ylabel('e1'); grid on;
subplot(6,3,2); plot(t_cont, e2, 'k'); title('e2 连续');
ylabel('e2'); grid on;
subplot(6,3,3); plot(t_cont, e3, 'k'); title('e3 连续');
ylabel('e3'); grid on;

%% 每种采样频率下的采样信号（第 4~6 行）
for i = 1:length(Fs_list)
    Fs = Fs_list(i); Ts = 1/Fs;
    t_sample = 0:Ts:1;
    e1_s = sin(2*pi*t_sample);
    e2_s = sin(18*pi*t_sample + pi);
    e3_s = sin(22*pi*t_sample);
    row = i + 1;
    subplot(4,3,row*3 - 2);
    stem(t_sample, e1_s, colors{i}, 'filled');
    title(['e1 采样 Fs=' , num2str(Fs), 'Hz']); grid on;

    subplot(4,3,row*3 - 1);
    stem(t_sample, e2_s, colors{i}, 'filled');
    title(['e2 采样 Fs=' , num2str(Fs), 'Hz']); grid on;

    subplot(4,3,row*3);
    stem(t_sample, e3_s, colors{i}, 'filled');
    title(['e3 采样 Fs=' , num2str(Fs), 'Hz']); grid on;
end

sgtitle('连续信号与不同采样频率下的采样信号');
```

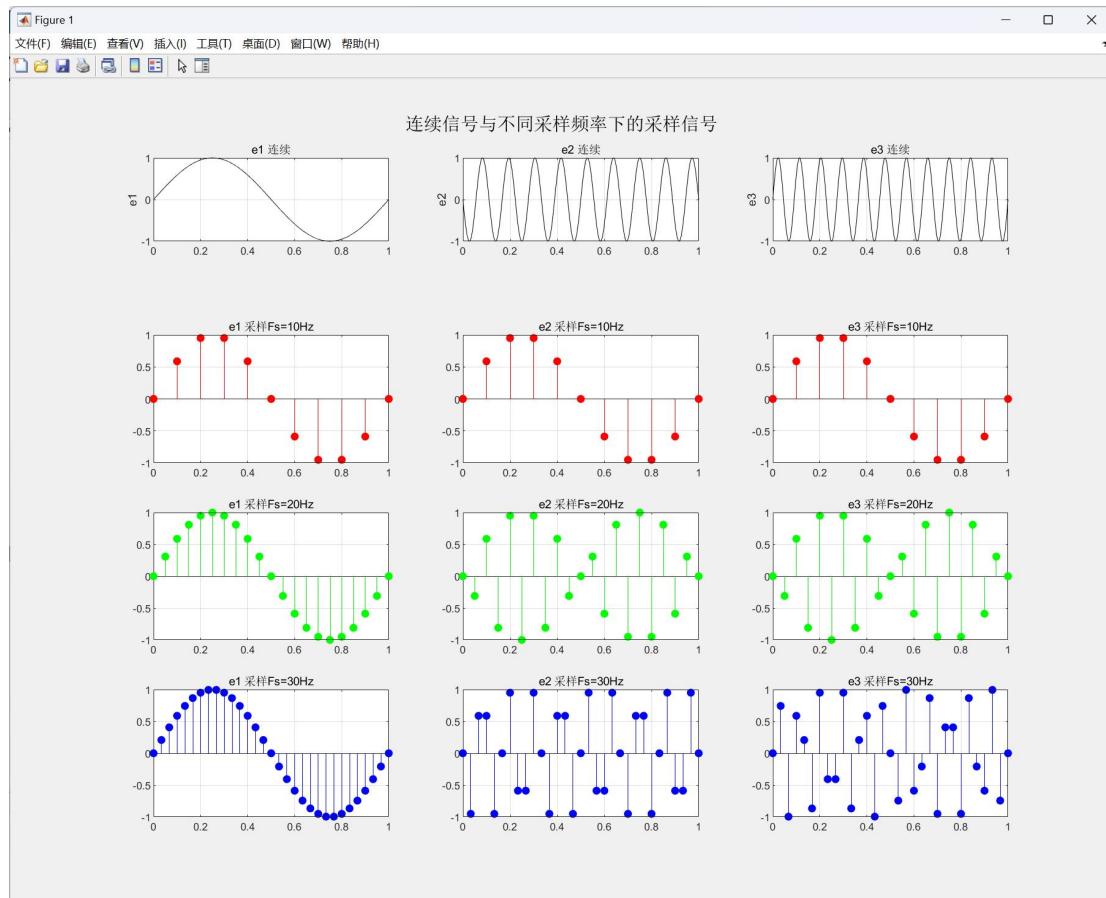
```

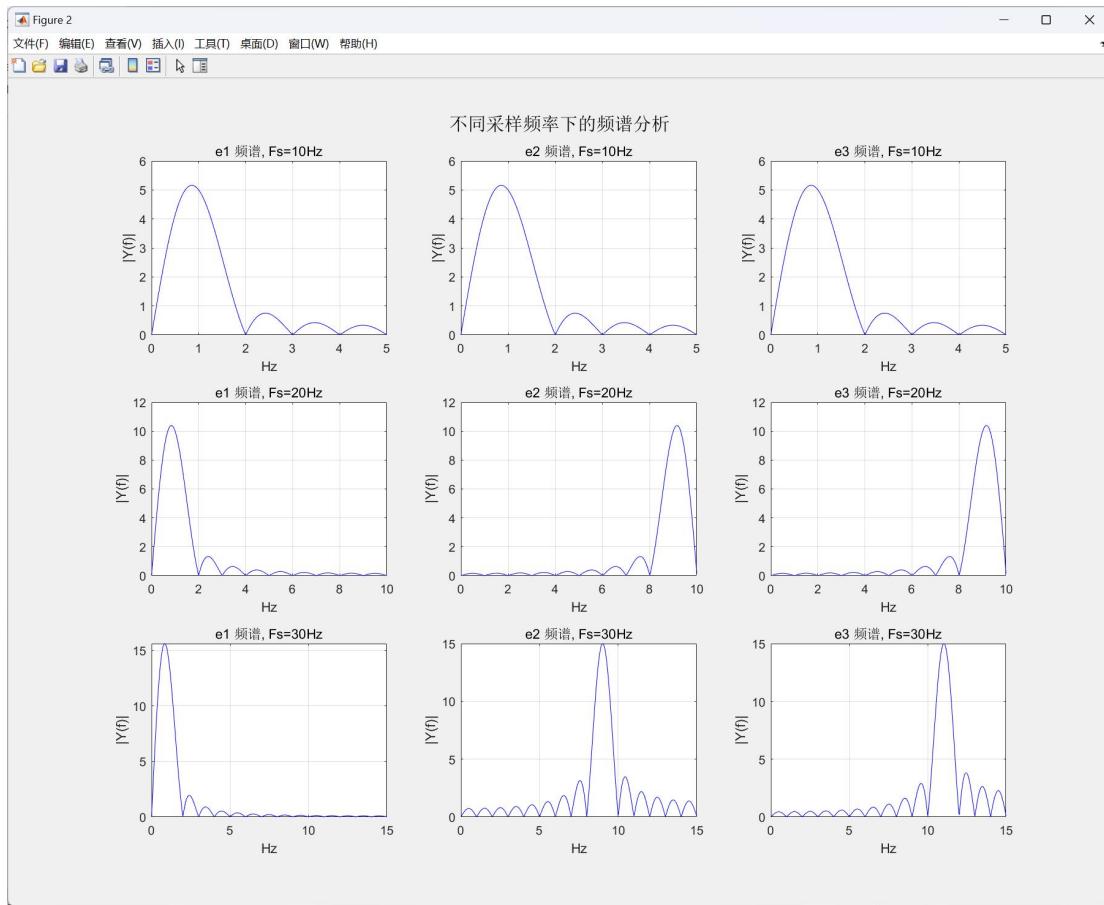
%% 新图绘制频谱 (频域分析)
figure('Position', [100, 100, 1200, 900]);

for i = 1:length(Fs_list)
Fs = Fs_list(i);
t_sample = 0:1/Fs:1;
plot_fft(sin(2*pi*t_sample), Fs, (i-1)*3 + 1, 'e1 频谱');
plot_fft(sin(18*pi*t_sample + pi), Fs, (i-1)*3 + 2, 'e2 频谱');
plot_fft(sin(22*pi*t_sample), Fs, (i-1)*3 + 3, 'e3 频谱');
end
sgtitle('不同采样频率下的频谱分析');

% 嵌套函数用于 FFT 绘图
function plot_fft(sig, Fs, subplot_idx, label)
N = 1024;
Y = abs(fft(sig, N));
f = linspace(0, Fs, N);
subplot(3,3,subplot_idx);
plot(f, Y, 'b');
title([label, ', Fs=', num2str(Fs), 'Hz']);
xlim([0, Fs/2]);
xlabel('Hz'); ylabel('|Y(f)|'); grid on;
end

```





## 4.2 零阶保持器

试画出零阶保持器的幅值谱和相位谱，和已知的学习结果比较，并进行分析（采样频率 10Hz，请计算出 0 频率处的幅值谱）。

$$G_{\text{ho}}(j\omega) = T \frac{\sin \frac{\omega T}{2}}{\frac{\omega T}{2}} e^{-j\frac{\omega T}{2}}$$

$$|G_{\text{ho}}(j\omega)| = T \left| \frac{\sin \frac{\omega T}{2}}{\frac{\omega T}{2}} \right|$$

$$\angle G_{\text{ho}}(j\omega) = \angle \sin \frac{\omega T}{2} - \frac{\omega T}{2}$$

```
clear; close all; clc;
fs = 10; % 采样频率
f = 0:0.1:50; % 频率轴
omega = 2*pi*f/fs;
H = sin(omega/2) ./ (fs* omega/2) .* exp(-1i*omega/2); % ZOH 频率响应

mag = abs(H);
phase = angle(H);

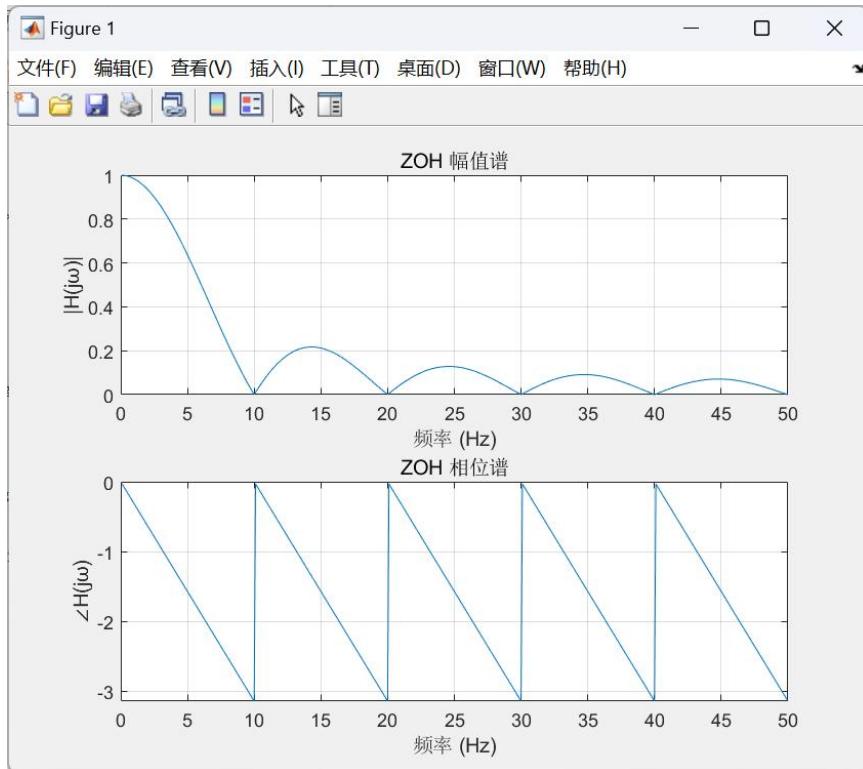
subplot(2,1,1);
plot(f, mag);
```

```

title('ZOH 幅值谱');
xlabel('频率 (Hz)'); ylabel('|H(jω)|'); grid on;

subplot(2,1,2);
plot(f, phase);
title('ZOH 相位谱');
xlabel('频率 (Hz)'); ylabel('∠H(jω)'); grid on;

```



### 4.3 采样及零阶保持

一周期信号  $e_1(t) = \sin(2\pi t)$  用采样频率 10Hz 进行周期采样，后通过零阶保持器，试画出

a 周期信号  $e_1(t) = \sin(2\pi t)$  的幅值谱；

b 周期信号  $e_1(t) = \sin(2\pi t)$  被周期采样后，离散信号的幅值谱；（注意离散信号频谱的幅值）

c 通过 ZOH 后输出信号的幅值谱和相位谱。（信号经过传递函数；预估一下幅值谱什么样子？）

d 步骤 c 输出信号频谱中前 5 个非零频率分量对应的时间信号，并画出 5 个时间域信号求和后的信号。

在同一张图上画出：采样信号  $e_1(kT) = \sin(2\pi kT)$ 、 $T = 0.1s$ 、 $k \geq 0$  经过零阶保持器的输出信号（stair 命令），并与前述求和信号进行对比。

提示：

首先获得  $e_1(t) = \sin(2\pi t)$  的频谱（对  $t=0:T:0.9$  秒，即一个周期内的信号进行 fft 变换，取  $N=10$ ，周期延拓成 0:1:29Hz 上的频谱）；

求零阶保持器的频谱（0:1:29Hz 上的频谱）；

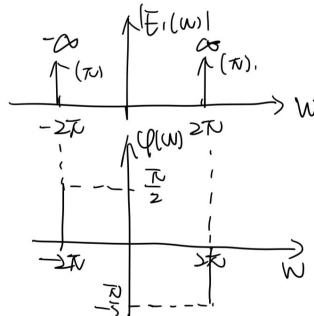
二者频谱（信号的和 ZOH 的）相乘获得输出信号的频谱；

对应于  $\pm f$  Hz 处的非零频率分量  $\alpha_i$ 、 $\alpha_r$  将产生正弦信号

$$2|\alpha_i| \cos(2\pi ft + \arg(\alpha_i))$$

(a) 理论分析如下

a.  $e_1(t) = \sin(2\pi t) = \frac{1}{2j}(e^{j2\pi t} - e^{-j2\pi t})$   
若  $f(t) \xrightarrow{\text{F}} F(w)$ , 由  $f(t) e^{j\omega_0 t} \xrightarrow{\text{F}} F(w-\omega_0)$   
 $E_1(w) = \frac{1}{2j} \sum_{k=-\infty}^{\infty} (\delta(w-2k\pi) - \delta(w+2k\pi)) = -j\pi \delta(w-2\pi) + j\pi \delta(w+2\pi)$



% (a) 高采样频率近似连续信号的频谱分析

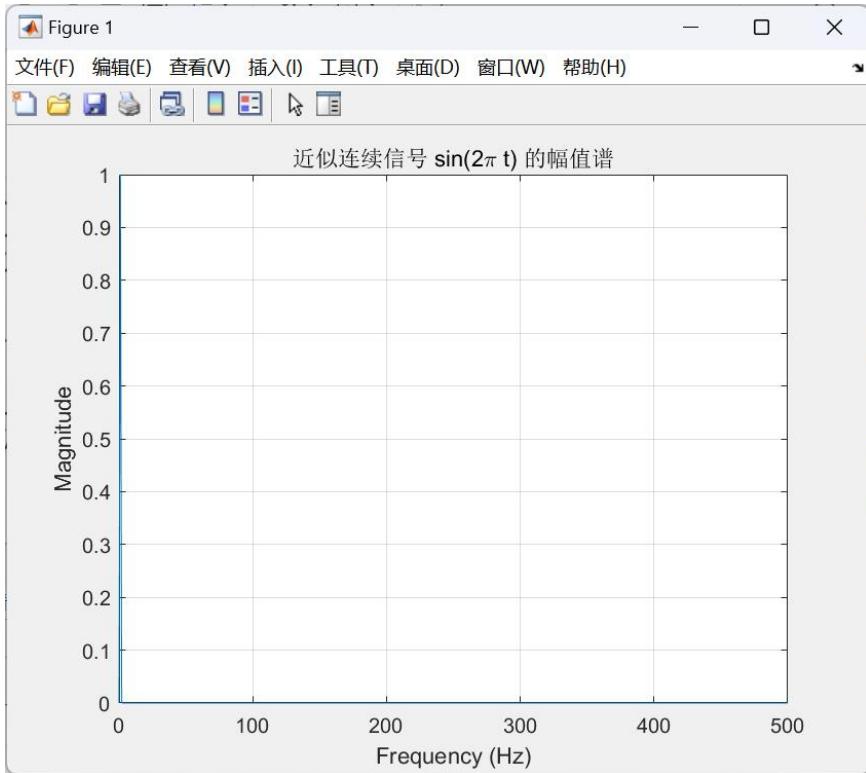
```
Fs1 = 1000; % 高采样频率
T = 1; % 持续 1 秒
t1 = 0:1/Fs1:T-1/Fs1; % 时间轴
x1 = sin(2*pi*t1); % 连续信号近似

N1 = length(x1);
X1 = fft(x1); % FFT
magX1 = abs(X1)/N1; % 幅值谱归一化
magX1 = magX1(1:N1/2+1);
magX1(2:end-1) = 2*magX1(2:end-1);
f1 = Fs1*(0:N1/2)/N1;

figure;
plot(f1, magX1);
title('近似连续信号 sin(2\pi t) 的幅值谱');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

本小题需要求解连续函数的幅值谱，因此需要设置高采样频率来模拟近似连续信号。在傅里叶变换后，`abs(X1)` 提取复数频谱的模值即各频率成分的幅值，除以 `N1` 对 FFT 幅值做归一化处理使幅值匹配原信号幅度。傅里叶变换结果对实数信号是共轭对称的，因此 `magX1 = magX1(1:N1/2+1);` 只保留前一半频率（0 到  $F_s/2$ ）；考虑到能量是分布在正负频率的，除直流（0 Hz）和 Nyquist ( $F_s/2$ ) 频率外，其他部分幅值要乘 2，对应代码 `magX1(2:end-1) =`

`2*magX1(2:end-1);`。然后构建频率坐标轴，绘制幅值谱，得到结果如下，与理论分析结果相对应。



(b) 首先获得 $e_1(t) = \sin(2\pi t)$ 的频谱，对 $t=0:T:0.9s$ 即一个周期内的信号进行 fft 变换，取 N=10，周期延拓成 0:1:29Hz 上的频谱。

实现代码如下。

```
%% (b) 周期信号频谱 (10Hz 采样)
Fs = 10;
t = 0:0.1:0.9;
x = sin(2*pi*t);

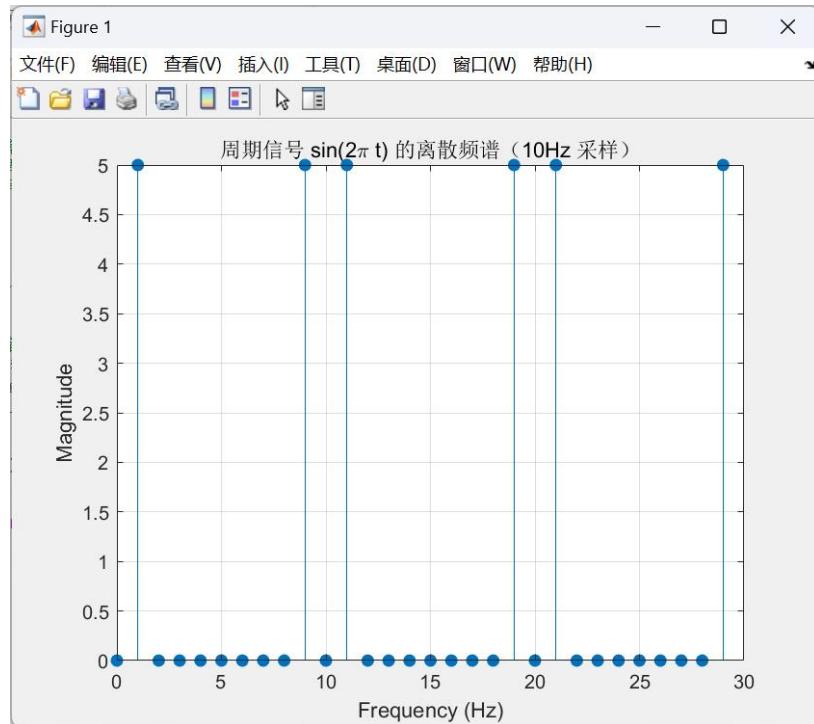
N = 10;
X = fft(x, N);
magX = abs(X);

f_base = 0:N-1;
f_extended = 0:29;
magX_extended = repmat(magX, 1, ceil(length(f_extended)/N));
magX_extended = magX_extended(1:length(f_extended));

figure;
stem(f_extended, magX_extended, 'filled');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('周期信号 sin(2\pi t) 的离散频谱 (10Hz 采样)');
grid on;
```

其中，`magX_extended = repmat(magX, 1, ceil(length(f_extended)/N));`实现了将傅里叶变换幅值谱 `magX` 周期性得重复扩展，以便构造一个长度更长的周期性频谱。函数函

数 `repmat(A, m, n)` 表示将矩阵 A 按照行方向重复 m 次、列方向重复 n 次。这里我们用它来对向量 `magX` 做列方向的复制。原始的频谱长度 `N = 10`, 预期拓展后的频谱长度为 `length(f_extended) = 30`, 用 `ceil` 对 `length(f_extended)/N` 向上取整得到了需要按列方向进行复制操作的次数。最后保留前 30 个点, 构成长度为 30 的周期性频谱图, 得到结果如下。



(c) 求零阶保持器的频谱, 即 0:1:29Hz 上的频谱; 并将信号的频谱与零阶保持器的频谱相乘获得输出信号的频谱。

实现代码如下。

```
%% (c) ZOH 系统频响作用下的频谱变化
Fs = 10;
t = 0:0.1:0.9;
x = sin(2*pi*t);

N = 10;
X = fft(x, N);
X_mag = abs(X);
X_phase = angle(X);

f_base = 0:N-1;
f_ext = 0:29;
X_mag_ext = repmat(X_mag, 1, ceil(length(f_ext)/N));
X_mag_ext = X_mag_ext(1:length(f_ext));
X_phase_ext = repmat(X_phase, 1, ceil(length(f_ext)/N));
X_phase_ext = X_phase_ext(1:length(f_ext));

T = 1/Fs;
f = f_ext;

H_zoh = sinc(f * T) .* exp(-1j * pi * f * T);
```

```

H_mag = abs(H_zoh);
H_phase = angle(H_zoh);

Y_mag = X_mag_ext .* H_mag;
Y_phase = X_phase_ext + H_phase;

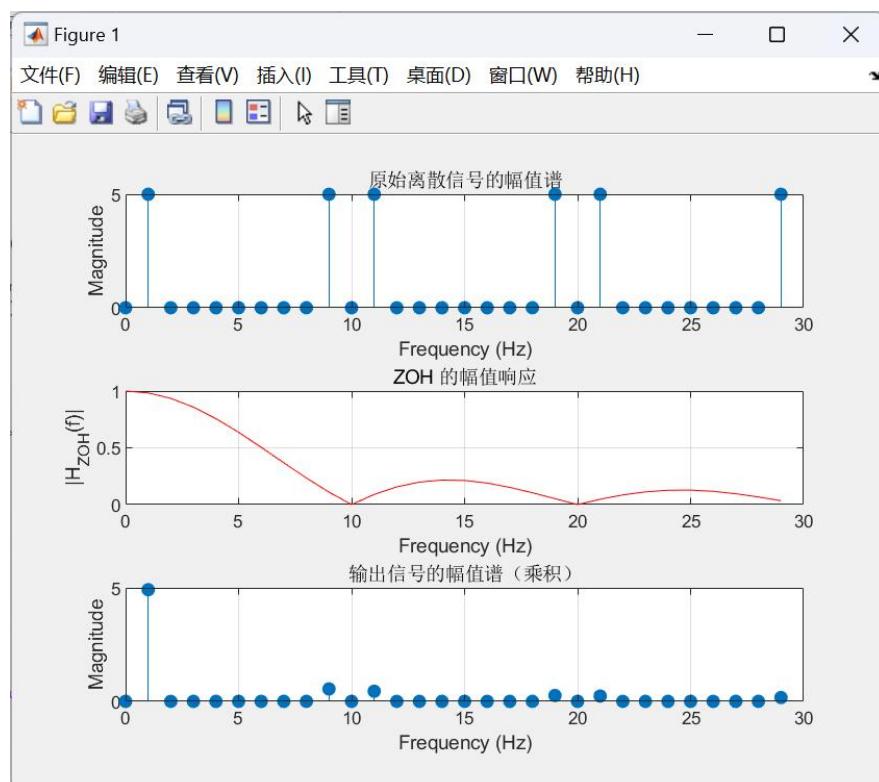
figure;
subplot(3,1,1);
stem(f, X_mag_ext, 'filled');
title('原始离散信号的幅值谱');
xlabel('Frequency (Hz)'); ylabel('Magnitude'); grid on;

subplot(3,1,2);
plot(f, H_mag, 'r');
title('ZOH 的幅值响应');
xlabel('Frequency (Hz)'); ylabel('|H_{ZOH}(f)|'); grid on;

subplot(3,1,3);
stem(f, Y_mag, 'filled');
title('输出信号的幅值谱 (乘积)');
xlabel('Frequency (Hz)'); ylabel('Magnitude'); grid on;

```

本小题主要部分在于零阶保持器的频率响应。`H_zoh` 是 ZOH 系统的频域响应，包括复数形式；`H_mag` 是幅值响应，随频率升高而衰减；`H_phase` 是相位响应。在求得零阶保持器的频谱后，将信号与零阶保持器的幅值谱相乘，`Y_mag = X_mag_ext .* H_mag`；其中 `.*` 是 Matlab 中的点乘，其表示逐元素相乘而非线性代数中的矩阵乘法。将信号与零阶保持器的相位相加，`Y_phase = X_phase_ext + H_phase`。随后绘图，我将原始离散信号的频谱、零阶保持器的幅值响应与滤波后的频谱绘制在了一起便于观察，得到如下结果。对于原始离散信号中幅值非零的部分，零阶保持器对其起到了一定的削弱作用。



(d) 根据提示, 对于 $\pm f$ Hz 处的非零频率分量 $\alpha_i$ 、 $\alpha_{-i}$ 将产生正弦信号  $2|\alpha_i| \cos(2\pi ft + arg(\alpha_i))$ 。因此, 使用 `x_components(:, i) = 2*X_mag(idx(i)) * cos(2*pi*frequencies(i)*t_zoh + phases(i));`构造 5 个频率分量的时域信号。

```
%% (d) ZOH 插值与频域重建信号比较
Fs = 10;
T = 1/Fs;
t = 0:T:1;
f = 1;
x = sin(2*pi*f*t);

t_zoh = 0:T/100:1;
x_zoh = interp1(t, x, t_zoh, 'previous');

L = length(t_zoh);
X_fft = fft(x_zoh);
f_axis = (0:L-1)*(Fs*100/L);

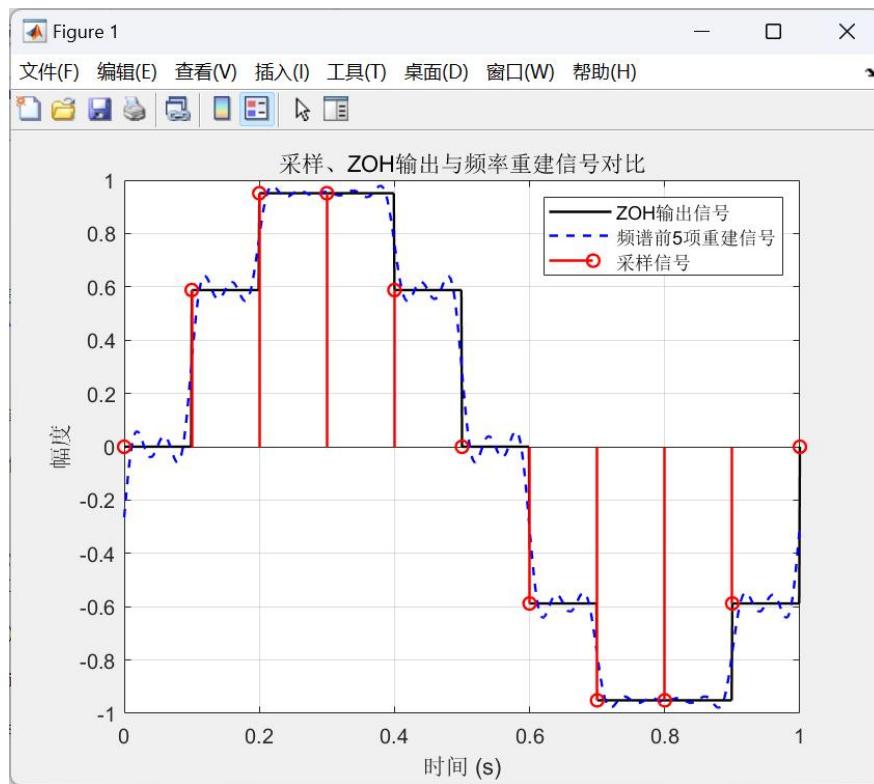
X_mag = abs(X_fft/L);
X_mag_half = X_mag(2:floor(L/2));
[~, idx] = maxk(X_mag_half, 5);
idx = idx + 1;
frequencies = f_axis(idx);
phases = angle(X_fft(idx));

x_components = zeros(length(t_zoh), 5);
for i = 1:5
    x_components(:, i) = 2*X_mag(idx(i)) * cos(2*pi*frequencies(i)*t_zoh +
    phases(i));
end
x_reconstructed = sum(x_components, 2);

figure;
plot(t_zoh, x_zoh, 'k-', 'LineWidth', 1.2); hold on;
plot(t_zoh, x_reconstructed, 'b--', 'LineWidth', 1.2);
stem(t, x, 'r|', 'LineWidth', 1.2);
legend('ZOH 输出信号', '频谱前 5 项重建信号', '采样信号');
title('采样、ZOH 输出与频率重建信号对比');
xlabel('时间 (s)');
ylabel('幅度');
grid on;
```

得到结果如下。图中红色表示采样信号, 蓝色虚线为频谱前 5 项重建信号, 黑色实线为 ZOH 输出信号。从图中可以看出, 在重构结果中出现了明显的振荡现象, 即吉布斯现象。

其是指在使用有限项傅里叶级数逼近具有不连续点的信号时, 逼近结果在不连续点附近会出现振荡。吉布斯现象的本质原因在于傅里叶级数使用的是连续且周期性的正弦和余弦函数来逼近任意信号, 而对于具有跳变的不连续信号, 高频分量需要逐渐增多才能更好地拟合突变位置。当我们仅使用前 5 个频率分量重构信号时, 相当于进行了一个带宽受限的低通滤波, 无法充分表达高频部分, 从而产生振铃效应。



## 5.1

例 5.1  $D(s) = \frac{s^2 + s + 64}{s^2 + 10s + 64}$ ,  $T=0.2$

- 1) 应用阶跃响应不变法离散化  $D(s)$ , 仿真验证离散化前后系统阶跃响应在采样点处的值相等。
- 2) 应用后向差分法离散化  $D(s)$ 。
- 3) 应用双线性变换法离散化  $D(s)$ 。
- 4) 应用预畸变双线性变换法离散化  $D(s)$ ,  $\omega_l=8$  处进行预畸变。
- 5) 比较 3、4 的差别。
- 6) 总结输入响应不变法、数值积分法、匹配零极点方法的异同。
- 7) 如果采样周期  $T=1$  秒, 应用双线性变换法离散化  $D(s)$ 会出现什么现象。
  
- 1) 应用阶跃响应不变法离散化, 其核心思想是使离散控制器在采样时刻产生的阶跃响应序列与连续控制器阶跃响应的采样值完全一致。对应代码 `D_z = c2d(D_s, T, 'zoh');` 函数 `c2d()` 用于将连续时间系统模型转换为离散时间系统模型, `D_s` 为连续系统的传递函数, `T` 为采样周期, `'zoh'` 指定离散方法采用零阶保持器, 是控制系统离散化最常用的方法之一。

```
% 连续系统
s = tf('s');
D_s = (s^2 + s + 64) / (s^2 + 10*s + 64);
T = 0.2;
```

```

% 阶跃响应不变法离散化
D_z = c2d(D_s, T, 'zoh');

% 验证频率响应
w = logspace(-1, 2, 500); % 频率范围
[mag_cont, phase_cont] = bode(D_s, w);
[mag_disc, phase_disc] = bode(D_z, w);

% 绘图: 幅频响应
figure;
subplot(2,1,1);
semilogx(w, 20*log10(squeeze(mag_cont)), 'b--', 'LineWidth', 1.5); hold on;
semilogx(w, 20*log10(squeeze(mag_disc)), 'r');
title('幅频响应');
legend('连续系统', '离散系统');
grid on;

% 相频响应
subplot(2,1,2);
semilogx(w, squeeze(phase_cont), 'b--', 'LineWidth', 1.5); hold on;
semilogx(w, squeeze(phase_disc), 'r');
title('相频响应');
xlabel('频率 (rad/s)');
grid on;

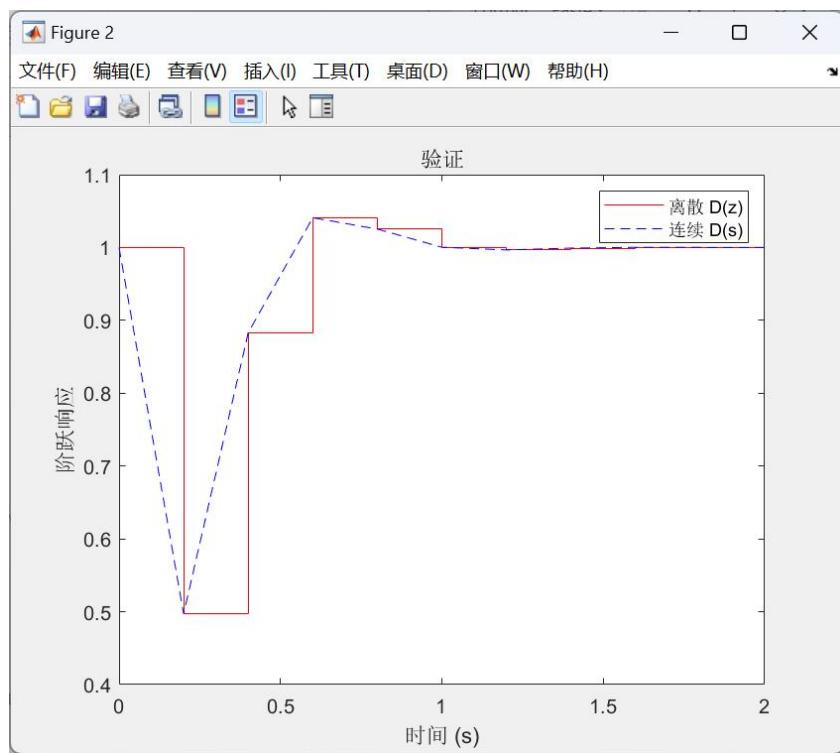
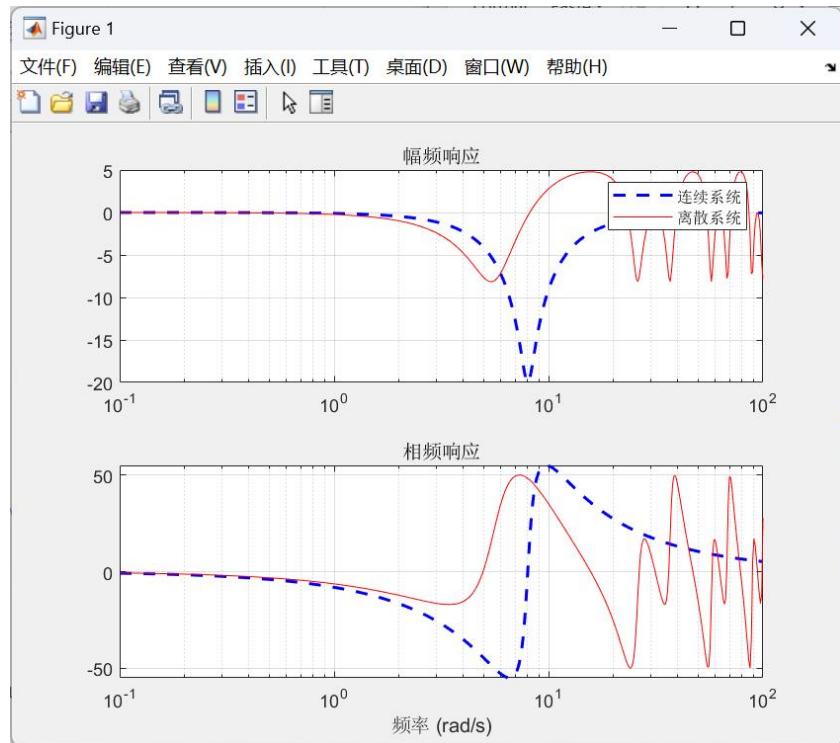
% 仿真阶跃响应
t_continuous = 0:T:2;
y_continuous = step(D_s, t_continuous);

t_discrete = 0:T:2;
y_discrete = step(D_z, t_discrete);

% 绘图比较
figure;
stairs(t_discrete, y_discrete, 'r'); hold on;
plot(t_continuous, y_continuous, 'b--');
legend('离散 D(z)', '连续 D(s)');
xlabel('时间 (s)'); ylabel('阶跃响应');
title('验证');

```

得到离散化结果如 Figure1 所示；并且绘图比较，比较结果如 Figure2 所示，离散化前后系统阶跃响应在采样点处的值相等，验证成功。



2) 使用后向差分法离散化。后向差分法是一种将连续系统离散化的数值积分方法，通过用后向差分近似代替微分运算实现离散化。其核心思想是用当前时刻和上一时刻的差值来近似微分：

$$\frac{dx(t)}{dt} \approx \frac{x[k] - x[k-1]}{T}$$

其中， $T$  为采样周期。在复频域中，这种近似对应以下变量替换关系，

$$s = \frac{z - 1}{T \cdot z}$$

对应代码中的这几句：

```
% 定义离散变量 z
z = tf('z', T);

% 后向差分替换: s = (1 - z^-1)/T = (z - 1)/(T * z)
s_backward = (z - 1) / (T * z);

% 将 s 替换到 D(s) 中
D_z = (s_backward^2 + s_backward + 64) / (s_backward^2 + 10*s_backward + 64);
```

完整实现代码如下。

```
% 连续系统
s = tf('s');
D_s = (s^2 + s + 64) / (s^2 + 10*s + 64);
T = 0.2;

% 定义离散变量 z
z = tf('z', T);

% 后向差分替换: s = (1 - z^-1)/T = (z - 1)/(T * z)
s_backward = (z - 1) / (T * z);

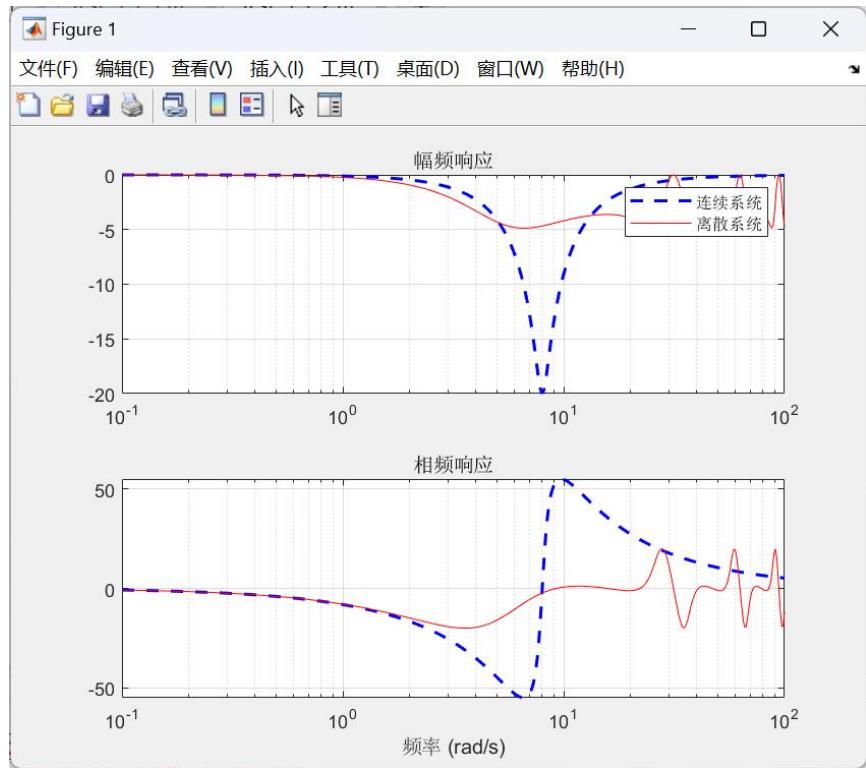
% 将 s 替换到 D(s) 中
D_z = (s_backward^2 + s_backward + 64) / (s_backward^2 + 10*s_backward + 64);

% 验证频率响应
w = logspace(-1, 2, 500); % 频率范围
[mag_cont, phase_cont] = bode(D_s, w);
[mag_disc, phase_disc] = bode(D_z, w);

% 绘图: 幅频响应
figure;
subplot(2,1,1);
semilogx(w, 20*log10(squeeze(mag_cont)), 'b--', 'LineWidth', 1.5); hold on;
semilogx(w, 20*log10(squeeze(mag_disc)), 'r');
title('幅频响应');
legend('连续系统', '离散系统');
grid on;

% 相频响应
subplot(2,1,2);
semilogx(w, squeeze(phase_cont), 'b--', 'LineWidth', 1.5); hold on;
semilogx(w, squeeze(phase_disc), 'r');
title('相频响应');
xlabel('频率 (rad/s)');
grid on;
```

得到离散化结果如下图所示，



3) 双线性变换法通过非线性变换将连续时间的 s 域映射到离散时间的 z 域。具体映射公式为:

$$s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}$$

通过代码 `D_z = c2d(D_s, T, 'tustin');` 实现, '`tustin`' 表示选择双线性变换法。

```
% 连续系统
s = tf('s');
D_s = (s^2 + s + 64) / (s^2 + 10*s + 64);
T = 0.2;

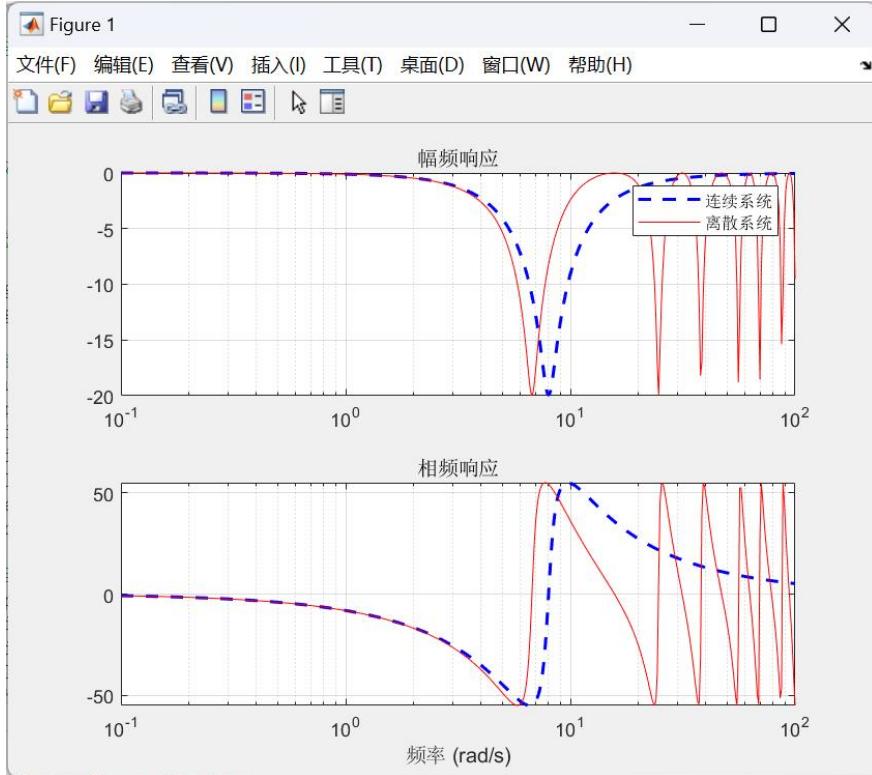
% 双线性变换法离散化
D_z = c2d(D_s, T, 'tustin');

% 验证频率响应
w = logspace(-1, 2, 500); % 频率范围
[mag_cont, phase_cont] = bode(D_s, w);
[mag_disc, phase_disc] = bode(D_z, w);

% 绘图: 幅频响应
figure;
subplot(2,1,1);
semilogx(w, 20*log10(squeeze(mag_cont)), 'b--', 'LineWidth', 1.5); hold on;
semilogx(w, 20*log10(squeeze(mag_disc)), 'r'); hold on;
title('幅频响应');
legend('连续系统', '离散系统');
grid on;

% 相频响应
subplot(2,1,2);
semilogx(w, squeeze(phase_cont), 'b--', 'LineWidth', 1.5); hold on;
semilogx(w, squeeze(phase_disc), 'r'); hold on;
title('相频响应');
```

```
xlabel('频率 (rad/s)');
grid on;
```



4) 预畸变双线性变换法。为了减小双线性变换中的频率失真问题，在离散化之前对连续系统的频率特性进行预补偿（预畸变）。这种方法通过在设计阶段调整频率参数，使得离散化后的系统在目标频率范围内更加接近原连续系统的特性。通过代码 `D_z = c2d(D_s, T, 'prewarp', omega0);` 实现，其中 '`prewarp`' 表示要使用预畸变双线性变换法；`omega0` 表示预扭曲频率。

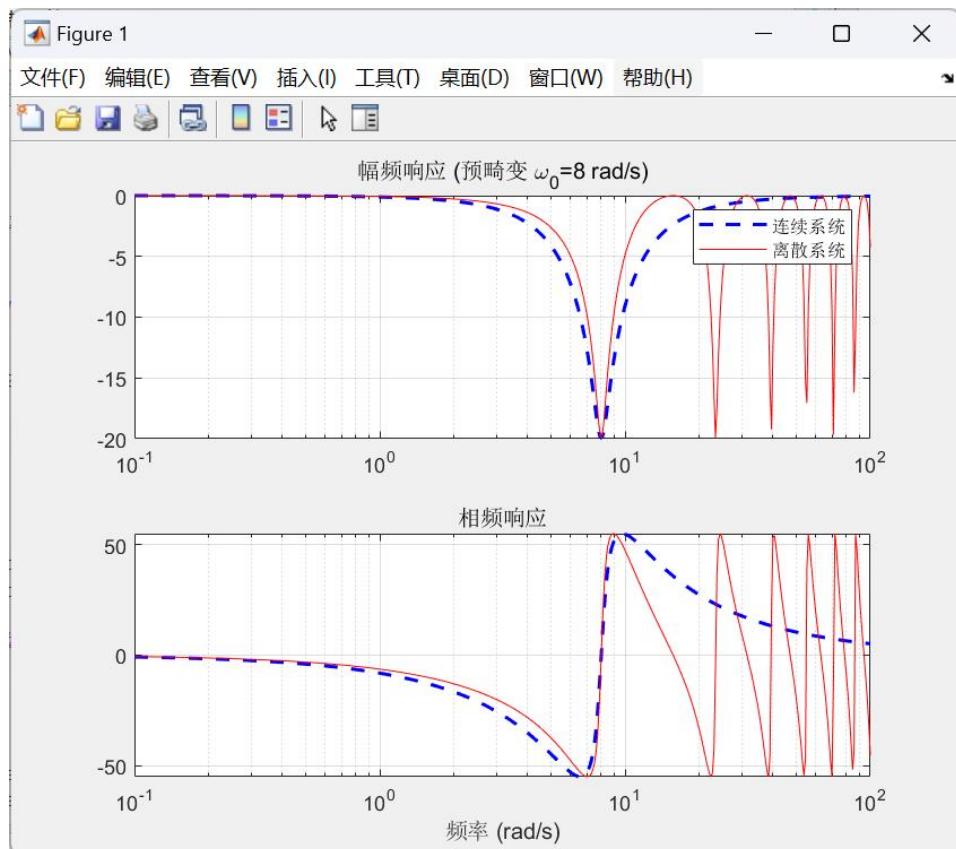
```
% 连续系统
s = tf('s');
D_s = (s^2 + s + 64) / (s^2 + 10*s + 64);
T = 0.2;
omega0 = 8; % 预畸变频率 (rad/s)

% 预畸变双线性变换离散化
D_z = c2d(D_s, T, 'prewarp', omega0);

% 验证频率响应
w = logspace(-1, 2, 500); % 频率范围
[mag_cont, phase_cont] = bode(D_s, w);
[mag_disc, phase_disc] = bode(D_z, w);

% 绘图: 幅频响应
figure;
subplot(2,1,1);
semilogx(w, 20*log10(squeeze(mag_cont)), 'b--', 'LineWidth', 1.5); hold on;
semilogx(w, 20*log10(squeeze(mag_disc)), 'r');
title('幅频响应 (预畸变 \omega_0=8 rad/s)');
legend('连续系统', '离散系统');
grid on;
```

```
% 相频响应
subplot(2,1,2);
semilogx(w, squeeze(phase_cont), 'b--', 'LineWidth', 1.5); hold on;
semilogx(w, squeeze(phase_disc), 'r');
title('相频响应');
xlabel('频率 (rad/s)');
grid on;
```



5)

```
% 连续系统定义
s = tf('s');
D_s = (s^2 + s + 64) / (s^2 + 10*s + 64);

%% 双线性变换法离散化（普通 Tustin）
T1 = 0.2;
D_z1 = c2d(D_s, T1, 'tustin');

%% 预畸变双线性变换法离散化（prewarp）
T2 = 0.2; % 较小采样周期
omega0 = 8; % 预畸变频率 rad/s
D_z2 = c2d(D_s, T2, 'prewarp', omega0);

%% 幅频&相频响应对比
w = logspace(-1, 2, 500); % 频率范围
[mag_cont, phase_cont] = bode(D_s, w);
[mag_z1, phase_z1] = bode(D_z1, w);
[mag_z2, phase_z2] = bode(D_z2, w);

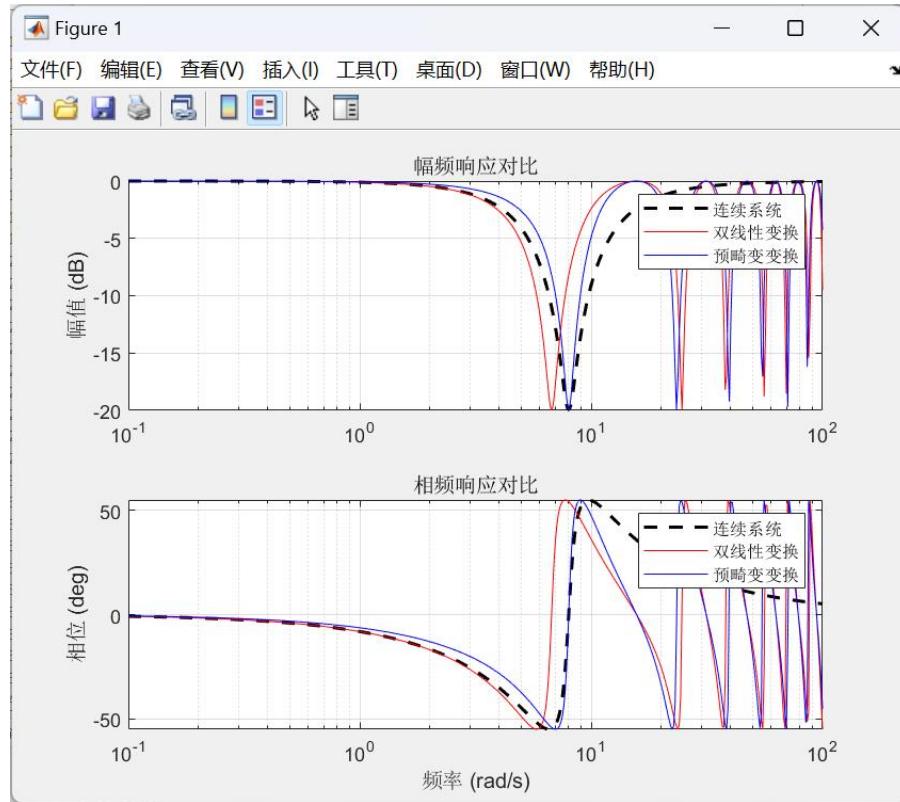
figure;
```

```

% 幅频响应
subplot(2,1,1);
semilogx(w, 20*log10(squeeze(mag_cont)), 'k--', 'LineWidth', 1.5); hold on;
semilogx(w, 20*log10(squeeze(mag_z1)), 'r-');
semilogx(w, 20*log10(squeeze(mag_z2)), 'b-');
legend('连续系统', '双线性变换', '预畸变变换');
title('幅频响应对比');
ylabel('幅值 (dB)');
grid on;

% 相频响应
subplot(2,1,2);
semilogx(w, squeeze(phase_cont), 'k--', 'LineWidth', 1.5); hold on;
semilogx(w, squeeze(phase_z1), 'r-');
semilogx(w, squeeze(phase_z2), 'b-');
legend('连续系统', '双线性变换', '预畸变变换');
xlabel('频率 (rad/s)');
ylabel('相位 (deg)');
title('相频响应对比');
grid on;

```



普通双线性变换法会引入频率压缩：高频部分映射到低频，导致频率失真，尤其当采样频率不够高时。预畸变双线性变换法在变换前对频率进行补偿，使特定频率（如  $w_1 = 8$ ）处的特性得到保留。

6) 三种方法的相同点包括：共同目标都是将连续时间系统转换为离散时间系统；三种方法都希望在一定程度上保持连续系统的动态行为和频率响应特性；并且这些方法广泛用于数字控制器设计、滤波器设计和信号处理等领域。

不同点如下。

方法名称	输入响应不变法	数值积分法	匹配零极点法
基本思想	保持时间域单位脉冲响应不变	基于数值积分近似	零极点直接映射
时域特性	保持时间域单位脉冲响应	时间域特性依赖积分方法	不直接保持时间域特性
频域特性	容易产生高频失真	梯形法能较好保持频率特性，但有失真	较好保持频率特性，但可能有增益误差
稳定性	连续系统稳定则离散系统稳定	梯形法较稳定，向前欧拉法可能不稳定	连续系统稳定则离散系统稳定
优点	时间响应精确，简单	梯形法避免混叠，适用于高精度场合	动态特性保持好，频率响应接近
缺点	高频失真明显	稳定性有限	不保持时间域单位脉冲响应
使用场景	时间域响应分析，关注相应精度	控制器设计，数字积分控制	动态特性分析，频率响应设计

7) T 改为 1s 后，代码如下。

```
% 连续系统
s = tf('s');
D_s = (s^2 + s + 64) / (s^2 + 10*s + 64);
T = 1;

% 双线性变换法离散化
D_z = c2d(D_s, T, 'tustin');

% 验证频率响应
w = logspace(-1, 2, 500); % 频率范围
[mag_cont, phase_cont] = bode(D_s, w);
[mag_disc, phase_disc] = bode(D_z, w);

% 绘图: 幅频响应
figure;
subplot(2,1,1);
semilogx(w, 20*log10(squeeze(mag_cont)), 'b--', 'LineWidth', 1.5); hold on;
semilogx(w, 20*log10(squeeze(mag_disc)), 'r');
title('幅频响应');
legend('连续系统', '离散系统');
ylabel('幅值 (dB)');
grid on;

% 相频响应
subplot(2,1,2);
semilogx(w, squeeze(phase_cont), 'b--', 'LineWidth', 1.5); hold on;
semilogx(w, squeeze(phase_disc), 'r');
title('相频响应');
xlabel('频率 (rad/s)');
ylabel('相位 (deg)');
grid on;

% 阶跃响应对比
t_cont = 0:0.01:20;
t_disc = 0:T:20;
y_cont = step(D_s, t_cont);
```

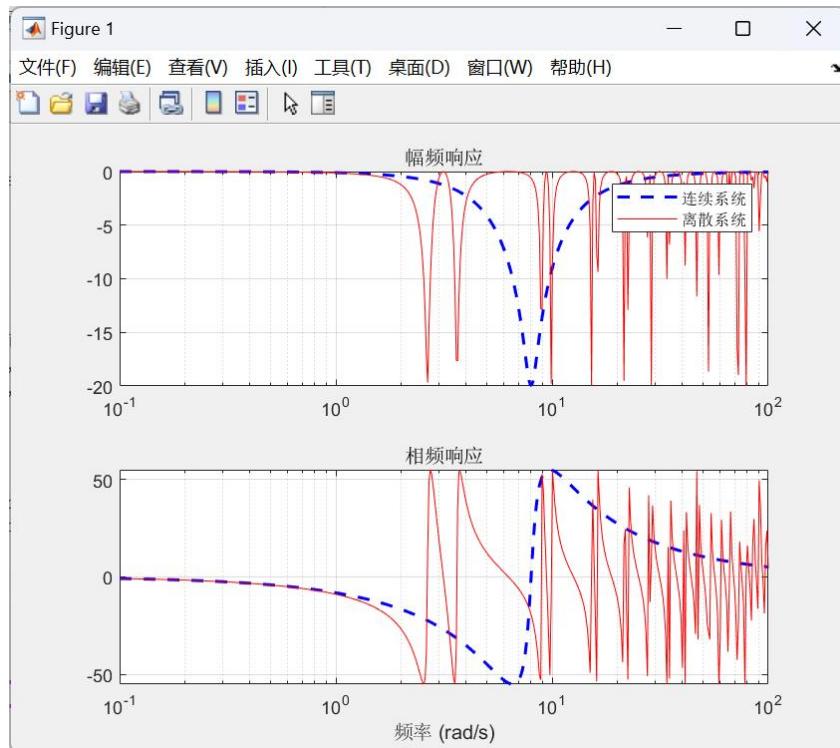
```

y_disc = step(D_z, t_disc);

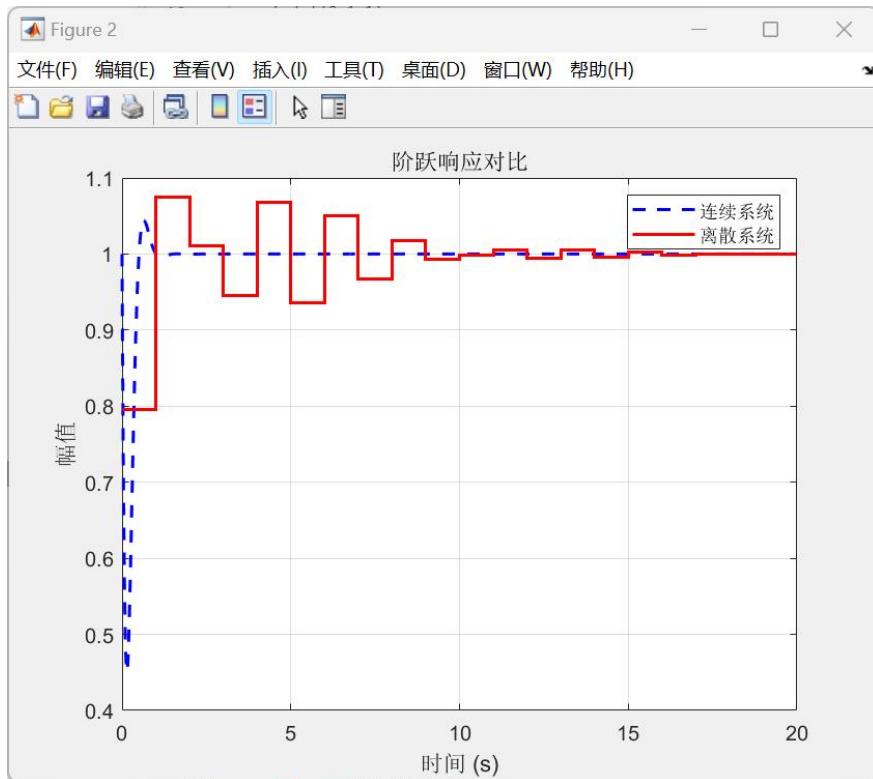
figure;
plot(t_cont, y_cont, 'b--', 'LineWidth', 1.5); hold on;
stairs(t_disc, y_disc, 'r', 'LineWidth', 1.5);
legend('连续系统', '离散系统');
xlabel('时间 (s)');
ylabel('幅值');
title('阶跃响应对比');
grid on;

```

得到结果如下。



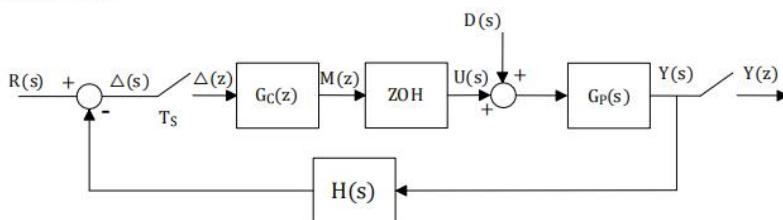
采样周期  $T=1s$ , 即采样频率为  $1Hz$ , 采样率过低, 会导致频率压缩加剧。由于采样频率过低, 离散化后的系统无法准确反映原系统中的高频成分, 导致频率失真严重, 进而使系统的动态特性发生较大变化。这种情况会直接影响离散化系统的性能表现, 例如系统响应可能变得迟缓, 振荡加剧, 甚至原有的稳定性边界信息会被破坏。当连续时间系统的带宽较高时, 这种问题尤为明显, 因为采样率低不仅会导致频率压缩, 还会引发频谱混叠, 使离散系统完全无法准确描述原系统的行为。



## 5.2

图中  $T_s=0.1s$ ,  $G_p(s)=\frac{4}{(2s+1)(0.5s+1)}$   $H(s)=\frac{4}{0.05s+1}$  按照以下的步骤设计一个

比例控制器  $G_c(z)=K_p$



- (a) Z 平面上画出随  $K_p$  变化的根轨迹图，并用 rlocfind 命令找出使系统临界稳定的  $K_p$  值和极点；用命令 dzline 和 rlocfind 确定使系统具有 0.8 阻尼比的  $K_p$  值和极点。
- (b) 令  $K_p = 0.5, 2.5, 5, 7$  时，画出系统的阶跃响应（参考输入）。用命令 kstats 命令，找出系统阶跃响应的超调量、上升时间、峰值时间、稳态值，并记录在表格里。
- (c) 令  $K_p=1.032$ ，画出单位阶跃参考输入和扰动输入响应。分析：比例控制的性能是否满意？

```

clc; clear; close all;

%% 系统建模
Ts = 0.1; % 采样周期
Gp = tf(4, conv([2 1], [0.5 1])); % Gp(s) = 4/[(2s+1)(0.5s+1)]
H = tf(1, [0.05 1]); % H(s) = 4/(0.05s+1)

% 离散化

```

```

Gpz = c2d(Gp, Ts, 'zoh'); % 零阶保持离散化
Hz = c2d(H, Ts, 'zoh'); % 零阶保持离散化
GHz = c2d(Gp*H, Ts, 'zoh');

%% (a) 根轨迹分析与交互选点
OpenLoop = GHz;

% 绘制根轨迹
figure;
rlocus(OpenLoop);
title('Z 平面内系统根轨迹');
grid on;

% 用户交互选取临界极点
disp('请在根轨迹图中点击你认为系统临界稳定的极点位置... ');
[Kcrit, PolesCrit] = rlocfind(OpenLoop);
fprintf('\n 你选择的临界极点对应 Kp = %.4f, 对应极点为: \n', Kcrit);
disp(PolesCrit);

% 添加阻尼比线  $\zeta = 0.8$  并交互选点
figure;
rlocus(OpenLoop);
hold on;
theta = acos(0.8); % 阻尼比线对应角度
r = linspace(0, 1.5, 100);
x = r .* cos(theta);
y = r .* sin(theta);
plot(x, y, 'r--'); % 上半部
plot(x, -y, 'r--'); % 下半部
title('根轨迹 + 阻尼比线  $\zeta = 0.8$ ');
grid on;
disp('请点击根轨迹上满足  $\zeta \approx 0.8$  的极点... ');
[Kzeta, PolesZeta] = rlocfind(OpenLoop);
fprintf('当  $\zeta \approx 0.8$  时的 Kp = %.4f, 对应极点为: \n', Kzeta);
disp(PolesZeta);

%% (b) Kp = 0.5, 2.5, 5, 7 的闭环响应分析
Kp_list = [0.5, 2.5, 5, 7];
figure;
sgrid('不同 Kp 下的单位阶跃响应');

fprintf('\n%-8s%-12s%-12s%-12s%-12s\n', 'Kp', 'Overshoot(%)', 'RiseTime(s)', 'PeakTime(s)', 'SteadyState');

for i = 1:length(Kp_list)
    Kp = Kp_list(i);
    Gc = Kp;

    CL = (Gc * Gpz) / (1 + Gc * GHz);

    % 生成阶跃响应数据
    [y, k] = step(CL, 5); % y: 响应, k: 时间向量
    k = k(:); y = y(:); % 确保是列向量

    % 计算性能指标
    [Mo, kp_, kr, ks, ess] = kstats(k, y, 1);

```

```

% 绘图
subplot(2,2,i);
plot(k, y, 'b', 'LineWidth', 1.5);
grid on;
title(['Kp = ', num2str(Kp)]);
xlabel('Time (s)'); ylabel('Output');

% 显示结果
fprintf('%-8.2f%-12.2f%-12.2f%-12.2f%-12.2f\n', ...
    Kp, Mo, kr, kp_, 1 - ess/100); % 将稳态值=1-稳态误差（百分比）
end

%% (c) Kp = 1.032 时的参考输入响应与扰动输入响应

Kp = 1.032;
Gc = Kp; % 离散控制器

% 参考输入响应 (R → Y)
T_r = (Gc * Gpz) / (1 + Gc * GHz);

% 扰动输入响应 (D → Y)
T_d = Gpz / (1 + Gc * GHz);

% 绘图
figure;
subplot(2,1,1);
step(T_r, 5);
title('Kp = 1.032: 单位阶跃参考输入响应 R → Y');
grid on;

subplot(2,1,2);
step(T_d, 5);
title('Kp = 1.032: 单位阶跃扰动输入响应 D → Y');
grid on;

% 输出性能指标
info_r = stepinfo(T_r);
info_d = stepinfo(T_d);

fprintf('\nKp = 1.032 时系统性能指标: \n');
fprintf('参考输入响应: 超调 %.2f%%, 上升时间 %.2f s, 稳态值 %.2f\n', ...
    info_r.Overshoot, info_r.RiseTime, info_r.SettlingMax);
fprintf('扰动输入响应: 超调 %.2f%%, 上升时间 %.2f s, 稳态值 %.2f\n', ...
    info_d.Overshoot, info_d.RiseTime, info_d.SettlingMax);

```

本题需要使用 tools 中的函数 kstats，如下。

```

%% 函数
function [Mo, tp, tr, ts, ess] = kstats(k, y, ref)
% KSTATS 计算离散时间系统的性能指标
% [Mo, tp, tr, ts, ess] = kstats(k, y, ref)
%
% 输入参数:
% k - 离散时间向量
% y - 对应的阶跃响应
% ref - 参考输入的稳态值（默认值为 1）
%
% 输出参数:

```

```

% Mo - 超调量 (以百分比表示)
% tp - 峰值时间 (单位: 秒)
% tr - 上升时间 (10%~90%)
% ts - 调节时间 (2% 区间)
% ess - 稳态误差 (以百分比表示)

% 默认参考值为 1
if nargin < 3
    ref = 1;
    disp('参考值未指定, 默认设为 1');
end

% 超调量和峰值时间
[y_max, idx_peak] = max(y);
tp = k(idx_peak);
Mo = max(100 * (y_max - ref) / ref, 0); % 百分比超调量

% 上升时间 (从 10% 到 90%)
idx_10 = find(y >= 0.1 * ref, 1, 'first');
idx_90 = find(y >= 0.9 * ref, 1, 'first');

if ~isempty(idx_10) && ~isempty(idx_90) && idx_10 > 1 && idx_90 > 1
    Ts = k(2) - k(1); % 采样周期

    % 线性插值精确估计 10% 与 90% 时刻
    t10 = k(idx_10) - Ts * (y(idx_10) - 0.1 * ref) / (y(idx_10) -
y(idx_10 - 1));
    t90 = k(idx_90) - Ts * (y(idx_90) - 0.9 * ref) / (y(idx_90) -
y(idx_90 - 1));

    tr = t90 - t10; % 上升时间
else
    tr = NaN;
end

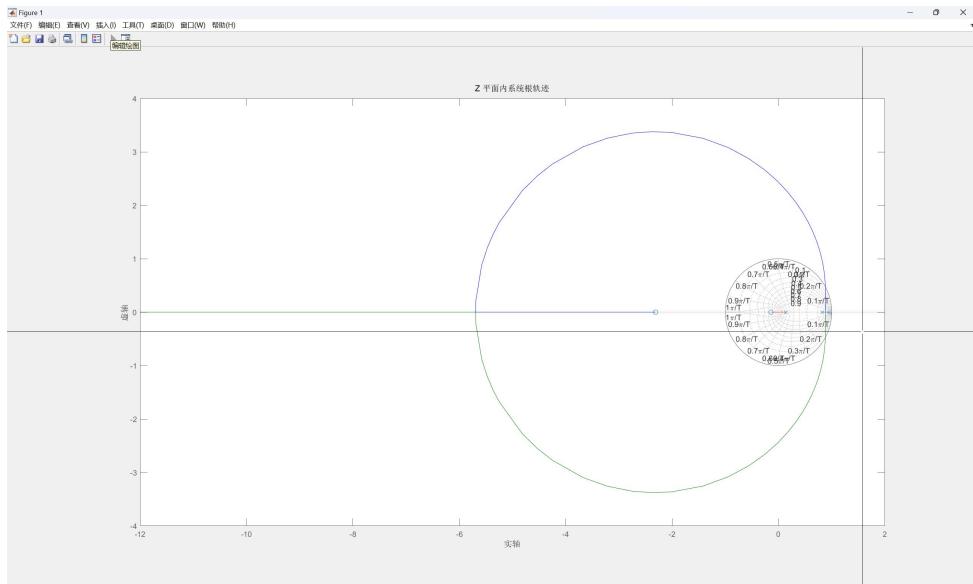
% 调节时间 (进入 2% 区间)
idx_settle = find(abs(y - ref) > 0.02 * ref, 1, 'last');
if isempty(idx_settle) || idx_settle == length(y)
    ts = k(end);
else
    ts = k(idx_settle + 1);
end

% 稳态误差 (以百分比计)
ess = abs(100 * (y(end) - ref) / ref);

end

```

a)



请在根轨迹图中点击你认为系统临界稳定的极点位置...

Select a point in the graphics window

```
selected_point =
```

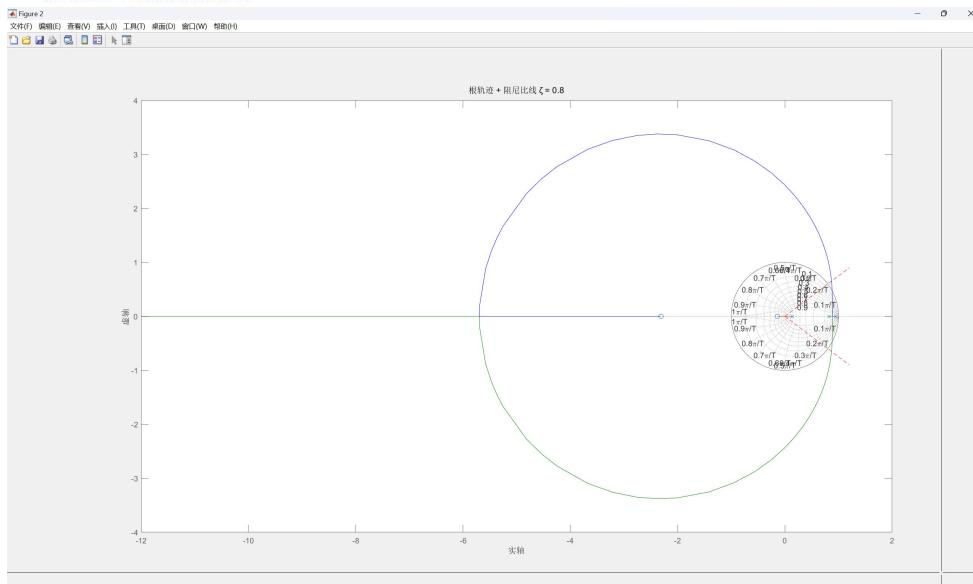
```
0.8869 + 0.4530i
```

你选择的临界极点对应  $K_p = 6.2651$ , 对应极点为:

```
0.8822 + 0.4529i
```

```
0.8822 - 0.4529i
```

```
0.0902 + 0.0000i
```



请点击根轨迹上满足  $\zeta \approx 0.8$  的极点...

Select a point in the graphics window

```
selected_point =
```

```
0.8700 + 0.6589i
```

$\zeta \approx 0.8$  时的  $K_p = 13.6524$ , 对应极点为:

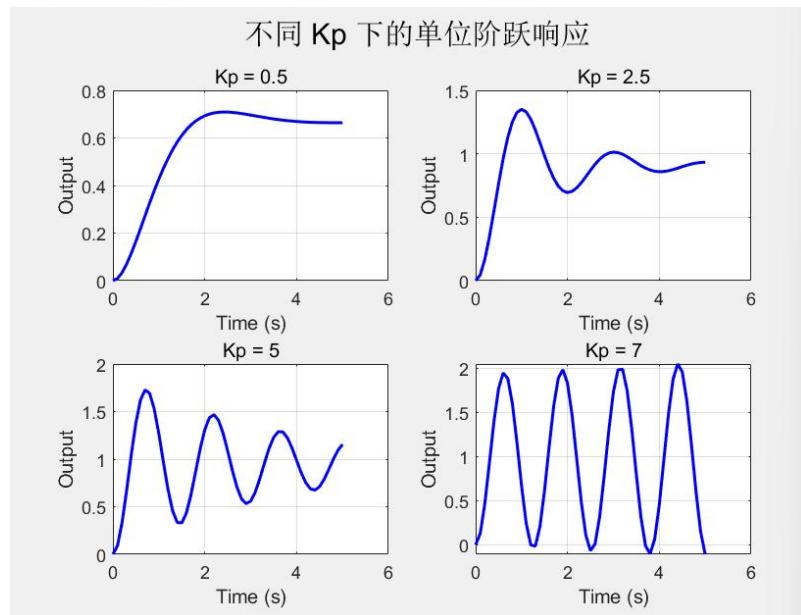
```
0.8684 + 0.6588i
```

```
0.8684 - 0.6588i
```

```
0.0581 + 0.0000i
```

b)

Kp	Overshoot (%)	RiseTime (s)	PeakTime (s)	SteadyState
0.50	0.00	10.00	2.40	0.66
2.50	35.08	0.42	1.00	0.93
5.00	72.81	0.26	0.70	0.84
7.00	104.97	0.22	4.40	-0.11

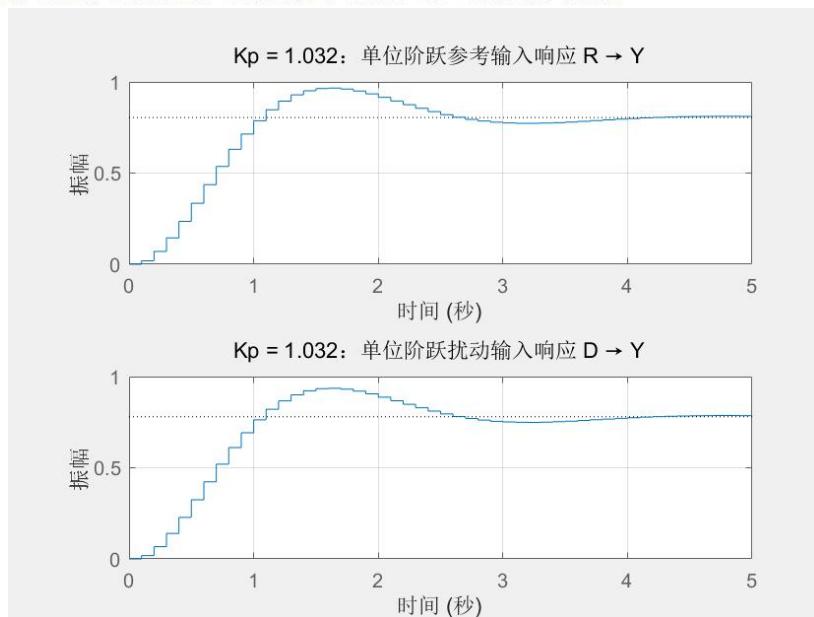


c)

$K_p = 1.032$  时系统性能指标:

参考输入响应: 超调 20.00%, 上升时间 0.70 s, 稳态值 0.97

扰动输入响应: 超调 20.00%, 上升时间 0.70 s, 稳态值 0.94



当比例系数  $K_p=1.032$  时, 系统对单位阶跃的参考输入和扰动输入均表现出较明显的动态特性: 超调为 20%, 上升时间为 0.70 秒, 表明系统响应较为迅速。然而, 稳态输出分别稳定在 0.97 和 0.94, 仍存在较明显的稳态误差。这说明尽管比例控制器能够加快系统响应, 但无

法有效消除静态误差，对扰动的抑制能力也较弱，导致整体控制性能不够理想，仍有较大改进空间。

### 5.3

$$G_c(z) = K_p [1 + K_I T_s \frac{z}{z-1}]$$

对于例 5.2 中的系统，若使用比例积分控制器

(a) 积分增益  $K_I=0.3, 0.5, 1.0, K_p=0.3, 0.5, 0.7, 0.9, 1.1$ ，画出各种情况下参考输入为单位阶跃信号的响应图。确定  $K_I$  什么值时  $K_p$  产生比较理想的阻尼效果和调整时间小于等于 5s (2%)。

(b) 在  $K_I=K_p$  时， $K_p=0.563$ ，并画出单位阶跃参考输入和扰动输入响应。分析：比例积分控制的性能是否满意？

a)

```

clc;clear;
% 离散系统建模
s = tf('s');
Gp = 4 / ((2*s + 1)*(0.5*s + 1)); % 被控对象
H = 1 / (0.05*s + 1); % 传感器
Ts = 0.1;
Gz = c2d(Gp, Ts, 'zoh');
Hz = c2d(H, Ts, 'zoh');
GHz = c2d(Gp * H, Ts, 'zoh');
z = tf('z', Ts);

% 参数列表
Kp_list = [0.3, 0.5, 0.7, 0.9, 1.1];
Ki_list = [0.3, 0.5, 1.0];

% 响应图与性能记录
figure;
idx = 1;
infos = [];

for i = 1:length(Ki_list)
    Ki = Ki_list(i);
    subplot(length(Ki_list), 1, i);
    hold on;
    for j = 1:length(Kp_list)
        Kp = Kp_list(j);

        % 控制器结构: Gc(z) = Kp * (1 + Ki*Ts*z / (z-1))
        Gc = Kp * (1 + Ki * Ts * z / (z - 1));

        % 开环、闭环系统
        CL = (Gc * Gz) / (1 + Gc * GHz);

        % 绘图
        [y, t] = step(CL, 10);
        plot(t, y, 'DisplayName', ['Kp=' , num2str(Kp)]);
    end
end

% 记录性能
[Mo, kp, kr, ks, ess] = kstats(t, y, 1);

```

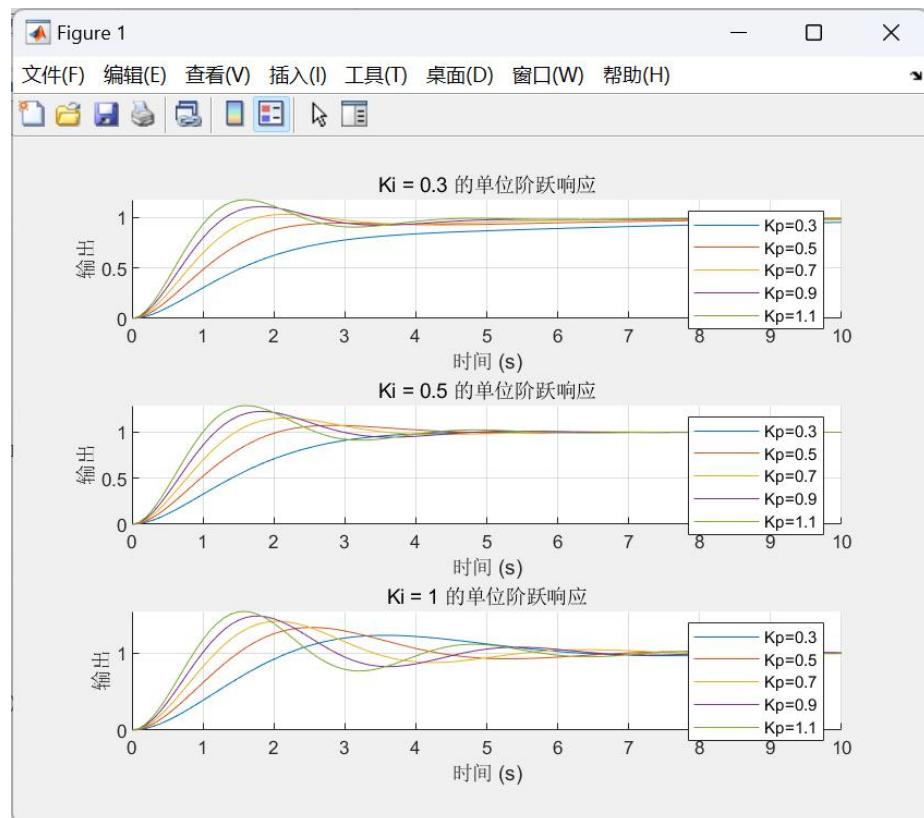
```

info.Kp = Kp;
info.Mo = Mo;
info.tp = kp;
info.tr = kr;
info.ts = ks;
info.ess = ess;
infos = [infos; struct('Kp', Kp, 'Ki', Ki, ...
'Overshoot', info.Mo, 'RiseTime', info.tr, 'PeakTime', info.tp, ...
'SettlingTime', info.ts, 'SteadyState', info.ess )];
end

title(['Ki = ', num2str(Ki), ' 的单位阶跃响应']);
xlabel('时间 (s)'); ylabel('输出');
legend show; grid on;
end

% 输出性能汇总
fprintf('\n 满足条件 (超调<10% 且调整时间≤5s) 参数如下: \n');
fprintf('\n%-6s %-6s %-12s %-12s %-12s %-15s %-12s\n', ...
'Kp', 'Ki', 'Overshoot(%)', 'RiseTime(s)', 'PeakTime(s)', 'SettlingTime(s)', ...
'SteadyState(%)');
for i = 1:length(infos)
    if infos(i).Overshoot < 10 && infos(i).SettlingTime <= 5
        fprintf('%-6.1f %-6.1f %-12.2f %-12.4f %-12.4f %-15.4f %-
12.4f\n', ...
            infos(i).Kp, infos(i).Ki, infos(i).Overshoot, infos(i).RiseTime, ...
            infos(i).PeakTime, infos(i).SettlingTime, infos(i).SteadyState);
    end
end

```



满足条件（超调<10% 且调整时间≤5s）参数如下：

Kp	Ki	Overshoot (%)	RiseTime (s)	PeakTime (s)	SettlingTime (s)	SteadyState (%)
0.3	0.5	0.17	2.4460	5.6000	4.0000	0.0642
0.5	0.5	7.43	1.3622	2.8000	4.2000	0.0068

由表中数据和图像结果可知，当 Ki=0.5 时，能产生比较理想的阻尼效果和调整时间。

b)

```
Kp_star = 0.563; % 示例值
Ki_star = 0.5; % 示例值

Gc = Kp_star * (1 + Ki_star * Ts * z / (z - 1));
CL_ref = (Gc * Gz) / (1 + Gc * GHz); % 参考输入响应
CL_dist = Gz / (1 + Gc * GHz); % 扰动输入响应

% 仿真时间向量
k = 0:Ts:10;

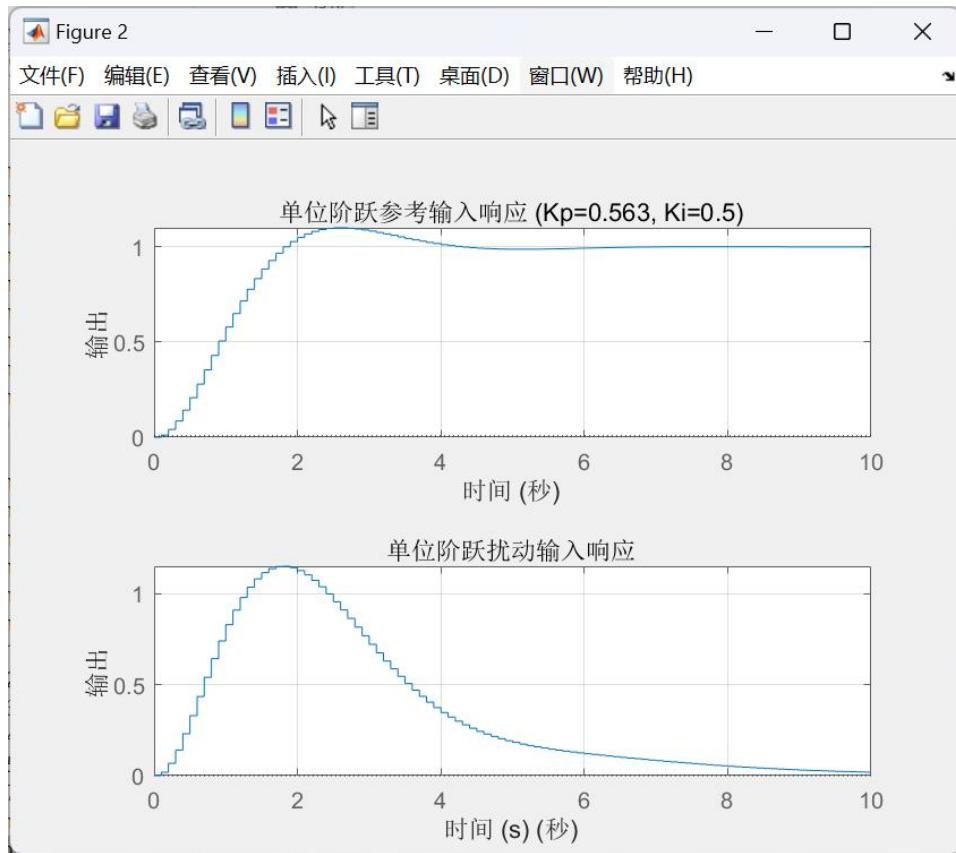
% 获取单位阶跃响应数据
[y_ref, ~] = step(CL_ref, k);

% 提取性能指标
[Mo, kp, kr, ks, ess] = kstats(k, y_ref, 1.0);

% 作图
figure;
subplot(2,1,1);
step(CL_ref, 10);
title(['单位阶跃参考输入响应 (Kp=' , num2str(Kp_star), ', Ki=' ,
num2str(Ki_star), ')']);
ylabel('输出'); grid on;

subplot(2,1,2);
step(CL_dist, 10);
title('单位阶跃扰动输入响应');
xlabel('时间 (s)'); ylabel('输出'); grid on;

% 打印性能指标
fprintf('\n 最优参数对应的性能指标:\n');
fprintf(' Mo = %.2f %%\n', Mo);
fprintf(' tp = %.2f s\n', kp);
fprintf(' tr = %.2f s\n', kr);
fprintf(' ts = %.2f s\n', ks);
fprintf(' ess = %.2f %%\n', ess);
```



性能指标：

```

Mo   = 10.02 %
tp   = 2.60 s
tr   = 1.21 s
ts   = 3.90 s
ess  = 0.02 %

```

在比例积分控制器参数设定为  $K_p=0.563$ ,  $K_i=0.5$  的情况下, 系统对参考输入的跟踪性能表现良好: 超调量控制在合理范围、响应速度较快(上升时间短)、稳态偏差处于可接受水平、基本满足多数性能指标要求。然而, 该控制器在扰动抑制方面存在明显不足: 系统对干扰的响应波动剧烈、存在显著的稳态误差、抗干扰能力较弱。综合评估表明, 该比例积分控制方案在扰动抑制方面的性能有待提升。

## 5.4

**例 5.4** 对于例 5.2 中的系统, 若使用比例积分微分控制器

$G_c(z) = K_p [1 + K_I T_s \frac{z}{z-1} + K_D \frac{z-1}{T_s z}]$ , 使系统单位阶跃响应满足: 超调量 $\leq 10\%$ , 上升时间 $\leq 0.35s$ , 调整时间 $\leq 1.4s$  (2%)。

```

clc; clear;
% 系统建模

```

```

s = tf('s');
Gp = 4 / ((2*s + 1)*(0.5*s + 1)); % 被控对象
H = 1 / (0.05*s + 1); % 传感器
Ts = 0.1;
Gz = c2d(Gp, Ts, 'zoh');
Hz = c2d(H, Ts, 'zoh');
GHz = c2d(Gp * H, Ts, 'zoh');
z = tf('z', Ts);

% 枚举 PID 参数范围
Kp_list = 1.9;
Ki_list = 0.4;
Kd_list = 0.4;

% 存储满足条件的解
results = [];

% 穷举搜索满足条件的参数组合
for Kp = Kp_list
    for Ki = Ki_list
        for Kd = Kd_list

            % PID 控制器公式
            Gc = Kp * (1 + (Ki * Ts * z) / (z - 1) + (Kd * (z - 1)) / (Ts * z));

            % 闭环系统
            CL = (Gc * Gz) / (1 + Gc * GHz);

            % 阶跃响应性能指标
            [y, t] = step(CL, 10);
            [Mo, kp, kr, ks, ess] = kstats(t, y, 1);
            info.Kp = Kp;
            info.Mo = Mo;
            info.tp = kp;
            info.tr = kr;
            info.ts = ks;
            info.ess = ess;

            if info.Mo <= 5 && ...
                info.tr <= 0.35 && ...
                info.ts <= 1.4
                % 记录满足条件的参数并绘图
                figure;
                plot(t, y, 'LineWidth', 1.5);
                grid on;
                % 添加坐标轴标签和标题
                xlabel('时间 (s)', 'FontSize', 10);
                ylabel('系统输出', 'FontSize', 10);
                title(sprintf('PID 控制阶跃响应 (Kp=%.1f, Ki=%.1f, Kd=%.1f)', Kp,
                Ki, Kd), ...
                    'FontSize', 12, 'FontWeight', 'bold');
                results = [results; struct(...,
                    'Kp', Kp, 'Ki', Ki, 'Kd', Kd, ...
                    'Overshoot', info.Mo, 'RiseTime', info.tr, 'PeakTime',
info.tp,... ...
                    'SettlingTime', info.ts, 'SteadyState', info.ess )];
            end
        end
    end
end

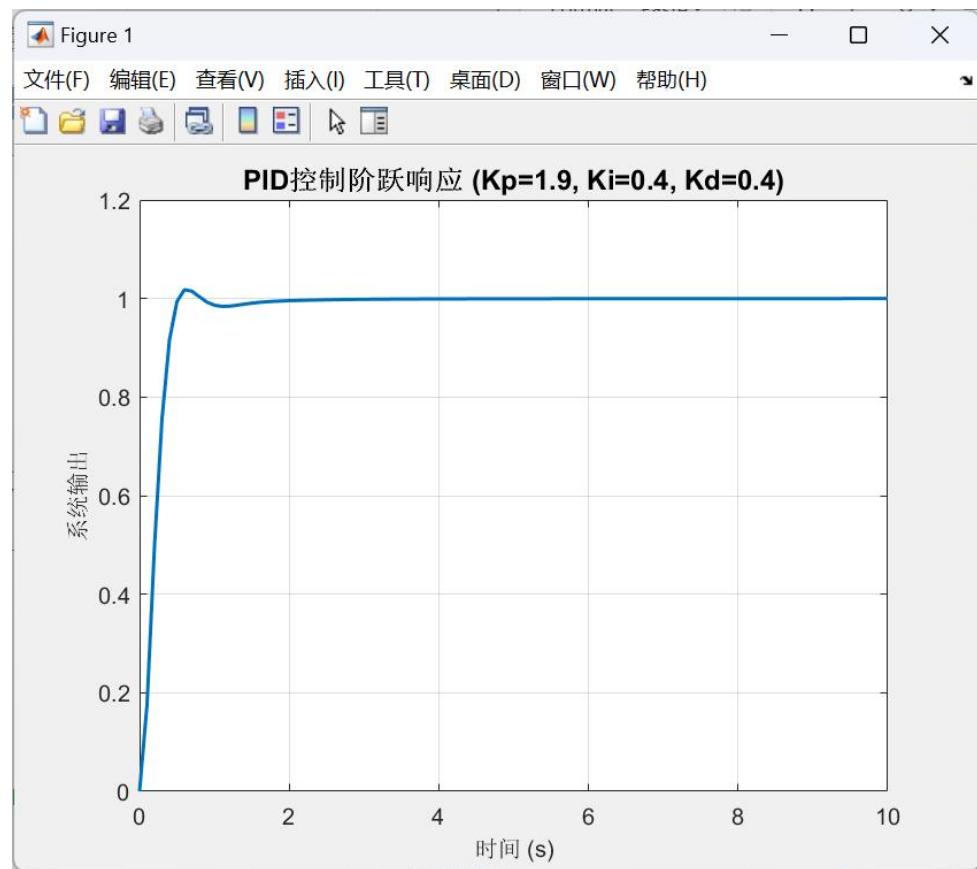
```

```

    end
end

fprintf('\n 满足设计指标的 PID 参数如下: \n');
for i = 1:length(results)
    r = results(i);
    fprintf('Kp = %.1f\n', r.Kp);
    fprintf('Ki = %.1f\n', r.Ki);
    fprintf('Kd = %.1f\n', r.Kd);
    fprintf('Overshoot = %.2f%%\n', r.Overshoot);
    fprintf('Rise Time = %.4fs\n', r.RiseTime);
    fprintf('Peak Time = %.4fs\n', r.PeakTime);
    fprintf('Settling Time = %.4fs\n', r.SettlingTime);
    fprintf('Steady State Error = %.4f\n', r.SteadyState);
end

```



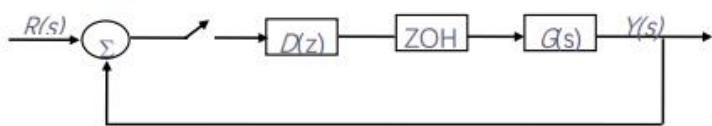
满足设计指标的 PID 参数如下:

Kp = 1.9  
 Ki = 0.4  
 Kd = 0.4  
 Overshoot = 1.78%  
 Rise Time = 0.3331s  
 Peak Time = 0.6000s  
 Settling Time = 0.5000s  
 Steady State Error = 0.0038

由上述输出可知超调量为 1.78%，上升时间为 0.333s，调整时间为 0.500s。满足题目超调量≤10%、上升时间≤0.35s、调整时间≤1.4s 的要求。

## 6.1

例 6.1 图中  $G(s) = \frac{1}{s+1}$ ,  $T = 0.1s$



控制系统框图

用一阶后向离散化方法离散化控制器,  $D(z) = D(s) | s \leftarrow \frac{z-1}{Tz}$ 。

要求:

- 1) 设计比例控制器, 求其阶跃响应, 观察有无静差, 记下  $M_p$ ,  $T_r$ ,  $T_s$  的值。
- 2) 设计比例积分控制器, 求其阶跃响应, 观察有无静差, 记下  $M_p$ ,  $T_r$ ,  $T_s$  的值。和 A 中响应比较, 并分析原因。
- 3) 调节比例积分控制器的比例参数, 得到较满意的一组参数。
- 4) 被控对象加入滞后环节,  $\tau=0.2s$ , 用 C 中控制器控制, 观察结果。
- 5) 调节 C 中原有参数, 如降低  $K_p$  值, 使 C 中超调量值基本不变 (但  $T_s$  要大很多)。
- 6) 用达林算法设计控制器,  $T\tau=0.2$ 。
- 7) 用 Smith 预估器设计控制器。

1)

% 1) 设计比例控制器, 求其阶跃响应

% 参数设置

$T = 0.1$ ; % 采样时间 (s)

$Gp_s = tf(1, [1 1])$ ; % 被控对象连续传递函数  $Gp(s) = 1 / (s + 1)$

% 离散化被控对象 (零阶保持法)

$Gp_z = c2d(Gp_s, T, 'zoh');$

% 比例控制器增益

$Kp = 5$ ;

$D_p = Kp$ ;

% 闭环系统传递函数  $G_{cl} = D(z)*Gp(z)/(1 + D(z)*Gp(z))$

$G_{cl} = feedback(D_p * Gp_z, 1)$ ;

% 仿真并获取阶跃响应 (仿真时间 5 秒)

$[y, t] = step(G_{cl}, 5)$ ;

$steady\_state\_value = y(end)$ ; % 稳态值估计

% 绘制阶跃响应

`figure;`

`stairs(t, y, 'LineWidth', 1.5); % 使用 stairs 表示离散时间响应`

`hold on;`

`plot([t(1), t(end)], [1, 1], 'r--'); % 参考线 (单位阶跃)`

`xlabel('时间 (s)');`

```

ylabel('系统输出');
title('闭环系统阶跃响应 (K_p = 5)');
legend('系统响应', '参考值');
grid on;

% 计算性能指标
Mp = (max(y) - steady_state_value) / steady_state_value * 100; % 最大超调量
(%) 

% 上升时间 Tr (从 10% 到 90%)
y_normalized = y / steady_state_value;
t10 = t(find(y_normalized >= 0.1, 1));
t90 = t(find(y_normalized >= 0.9, 1));
Tr = t90 - t10;

% 调节时间 Ts (进入±2%误差带)
lower_bound = 0.98 * steady_state_value;
upper_bound = 1.02 * steady_state_value;
settling_indices = find(y < lower_bound | y > upper_bound);
if ~isempty(settling_indices)
Ts = t(settling_indices(end));
else
Ts = 0;
end

% 稳态误差分析
steady_state_error = abs(1 - steady_state_value);
if steady_state_error < 0.01
fprintf('系统无静差。\\n');
else
fprintf('系统存在静差, 静差为: %.4f\\n', steady_state_error);
end

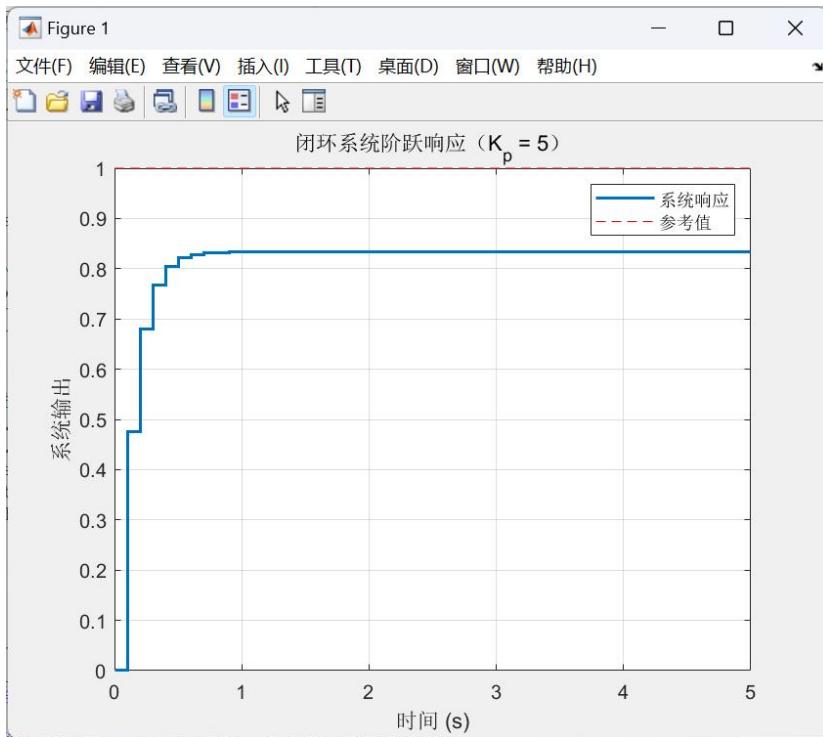
% 打印性能指标
fprintf('闭环系统性能指标 (K_p = %.1f): \\n', Kp);
fprintf('最大超调量 Mp: %.2f%%\\n', Mp);
fprintf('上升时间 Tr: %.4f s\\n', Tr);
fprintf('调节时间 Ts: %.4f s\\n', Ts);
fprintf('稳态输出值: %.4f\\n', steady_state_value);

```

```

>> ex_6_1_1
系统存在静差, 静差为: 0.1667
闭环系统性能指标 (K_p = 5.0):
最大超调量 Mp: 0.00%
上升时间 Tr: 0.2000 s
调节时间 Ts: 0.4000 s
稳态输出值: 0.8333

```



2)

```
% 2) 设计比例积分控制器, 求阶跃响应
% 系统参数
T = 0.1; % 采样时间 (s)
Gp_s = tf(1, [1 1]); % 被控对象连续传递函数 Gp(s) = 1 / (s + 1)

% 离散化对象 (ZOH 零阶保持)
Gp_z = c2d(Gp_s, T, 'zoh');

% 设计比例积分控制器 D(z) = Kp * (1 + Ki * T * z / (z - 1))
Kp = 10;
Ki = 2;
D_pi = Kp * (1 + Ki * T * z / (z - 1));

% 闭环系统传递函数 G_cl = D(z)Gp(z)/(1 + D(z)Gp(z))
G_cl = feedback(D_pi * Gp_z, 1);

% 仿真并获取阶跃响应
[y, t] = step(G_cl, 5); % 仿真时长 5 秒
steady_state_value = y(end); % 稳态值估计

% 绘制阶跃响应曲线
figure;
stairs(t, y, 'LineWidth', 1.5); % 使用 stairs 更适合离散系统响应
hold on;
plot([t(1), t(end)], [1, 1], 'r--'); % 单位阶跃参考线
xlabel('时间 (s)');
ylabel('系统输出');
title('闭环系统阶跃响应 (PI 控制器)');
legend('系统响应', '参考值');
grid on;
```

```

% 计算性能指标
Mp = (max(y) - steady_state_value) / steady_state_value * 100; % 最大超调量
(%)

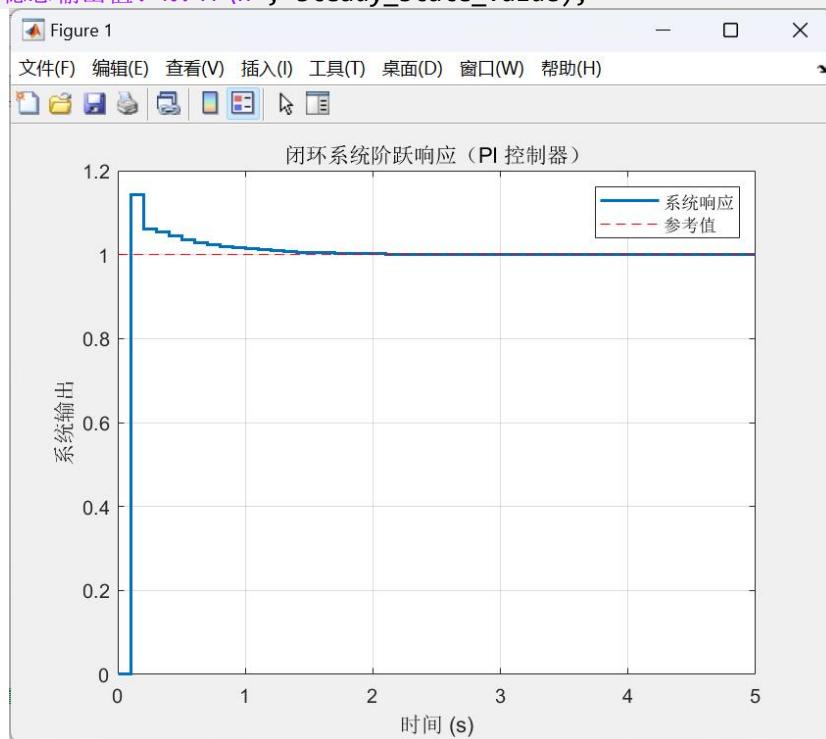
% 上升时间 Tr (从 10%到 90%)
y_normalized = y / steady_state_value;
t10 = t(find(y_normalized >= 0.1, 1));
t90 = t(find(y_normalized >= 0.9, 1));
Tr = t90 - t10;

% 调节时间 Ts (±2%误差带内)
lower_bound = 0.98 * steady_state_value;
upper_bound = 1.02 * steady_state_value;
settling_indices = find(y < lower_bound | y > upper_bound);
if ~isempty(settling_indices)
Ts = t(settling_indices(end));
else
Ts = 0;
end

% 稳态误差判断
steady_state_error = abs(1 - steady_state_value);
if steady_state_error < 0.01
fprintf('系统无静差。\\n');
else
fprintf('系统存在静差，大小为: %.4f\\n', steady_state_error);
end

% 输出性能指标
fprintf('闭环系统性能指标 (PI 控制器: Kp = %.1f, Ki = %.1f) : \\n', Kp, Ki);
fprintf('最大超调量 Mp: %.2f%%\\n', Mp);
fprintf('上升时间 Tr: %.4f s\\n', Tr);
fprintf('调节时间 Ts: %.4f s\\n', Ts);
fprintf('稳态输出值: %.4f\\n', steady_state_value);

```



```
>> ex_6_1_2
系统无静差。
闭环系统性能指标 (PI 控制器: Kp = 10.0, Ki = 2.0) :
最大超调量 Mp: 14.19%
上升时间 Tr: 0.0000 s
调节时间 Ts: 0.7000 s
稳态输出值: 1.0000
```

本小题的比例积分控制器相比纯比例控制器，在阶跃响应中表现出明显不同的现象。首先，PI 控制器消除了系统的静差，稳态输出值达到了理想的 1.0，而纯比例控制器存在较大静差，稳态输出仅为 0.8333，说明其无法完全消除稳态误差。其次，PI 控制器出现了较大的最大超调量 14.19%，而比例控制器则无超调，系统响应更加平稳。此外，PI 控制器的上升时间明显缩短，响应更快，但调节时间增加，系统达到稳态所需时间更长。

产生这些现象的根本原因在于积分环节的引入，积分作用能够累积误差信号，驱动控制器持续调整，从而消除稳态误差，实现无静差控制。但同时，积分作用使系统响应变得更加激进，容易产生超调和振荡，导致调节时间延长。而纯比例控制器仅根据当前误差进行调节，虽然响应稳定且无超调，但因缺乏累积误差的能力，导致存在静差。综上所述，PI 控制器通过积分环节提高了系统稳态性能，但也带来了动态响应上的不稳定因素，需要在设计时权衡二者，合理调整控制参数。

3)

```
% 3) 调节比例积分控制器的比例参数
% 系统参数
T = 0.1; % 采样时间(s)
Gp_s = tf(1, [1 1]); % 连续系统 Gp(s) = 1 / (s + 1)
z = tf('z', T);
Gp_z = c2d(Gp_s, T, 'zoh'); % 离散化被控对象

% PI 控制器参数
Kp = 8;
Ki = 1.5;
D_pi = Kp * (1 + Ki*T*z/(z - 1)); % PI 控制器离散形式

% 闭环系统传递函数
G_cl = (D_pi * Gp_z) / (1 + D_pi * Gp_z);

% 阶跃响应仿真
[y, t] = step(G_cl, 5); % 仿真 5 秒
steady_state_value = y(end); % 稳态值

% 绘图
figure;
stairs(t, y, 'LineWidth', 1.5); % 离散响应
hold on;
plot([t(1), t(end)], [1, 1], 'r--'); % 参考线
xlabel('时间 (s)');
ylabel('输出');
title('闭环系统阶跃响应 (Kp=8, Ki=1.5)');
legend('系统响应', '期望值');
```

```

grid on;

% 计算性能指标
Mp = (max(y) - steady_state_value)/steady_state_value * 100; % 最大超调量 (%)

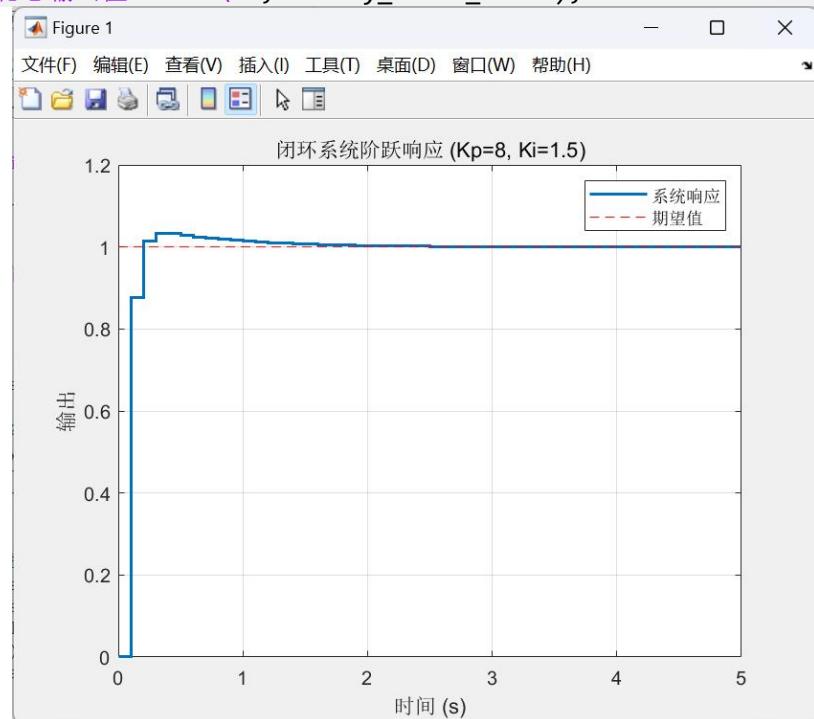
% 上升时间 Tr (10%~90%)
y_normalized = y / steady_state_value;
t10 = t(find(y_normalized >= 0.1, 1));
t90 = t(find(y_normalized >= 0.9, 1));
Tr = t90 - t10;

% 调节时间 Ts (进入±2%误差带)
lower = 0.98 * steady_state_value;
upper = 1.02 * steady_state_value;
idx = find(y < lower | y > upper);
if ~isempty(idx)
Ts = t(idx(end));
else
Ts = 0;
end

% 静态误差分析
steady_state_error = abs(1 - steady_state_value);
if steady_state_error < 0.01
fprintf('系统无静差\n');
else
fprintf('系统存在静差, 静差为: %.4f\n', steady_state_error);
end

% 输出性能参数
fprintf('性能指标 (Kp=%.1f, Ki=%.1f):\n', Kp, Ki);
fprintf('最大超调量 Mp: %.2f%\n', Mp);
fprintf('上升时间 Tr: %.4f s\n', Tr);
fprintf('调节时间 Ts: %.4f s\n', Ts);
fprintf('稳态输出值: %.4f\n', steady_state_value);

```



```
>> ex_6_1_3
系统无静差
性能指标 (Kp=8.0, Ki=1.5):
最大超调量 Mp: 3.37%
上升时间 Tr: 0.1000 s
调节时间 Ts: 0.7000 s
稳态输出值: 1.0000
```

4)

```
% 4) 被控对象加入滞后环节后的闭环响应分析
% 离散控制参数设置
T = 0.1; % 采样周期(s)
L = 0.2; % 被控对象延迟时间(s)
s = tf('s');
z = tf('z', T);

% 被控对象(含延迟)连续传递函数及其离散化
Gp_s = exp(-L*s) / (s + 1);
Gp_z = c2d(Gp_s, T, 'zoh'); % 零阶保持法离散化

% PI 控制器参数
Kp = 8;
Ki = 1.5;
D_pi = Kp * (1 + Ki * T * z / (z - 1)); % 一阶后向差分离散 PI 控制器

% 闭环系统传递函数
G_cl = feedback(D_pi * Gp_z, 1);

% 阶跃响应仿真
[y, t] = step(G_cl, 10); % 模拟 10 秒
steady_state_value = y(end); % 稳态值

% 绘制阶跃响应
figure;
stairs(t, y, 'LineWidth', 1.5);
hold on;
plot([t(1), t(end)], [1, 1], 'r--'); % 理想参考线
xlabel('时间 (s)');
ylabel('系统输出');
title('闭环阶跃响应(含滞后环节)');
legend('系统响应', '期望值');
grid on;

% 性能指标计算
Mp = (max(y) - steady_state_value) / steady_state_value * 100; % 最大超调量(%)

y_norm = y / steady_state_value;
t10 = t(find(y_norm >= 0.1, 1)); % 上升时间起点
t90 = t(find(y_norm >= 0.9, 1)); % 上升时间终点
Tr = t90 - t10; % 上升时间

lower = 0.98 * steady_state_value;
upper = 1.02 * steady_state_value;
settle_idx = find(y < lower | y > upper);
```

```

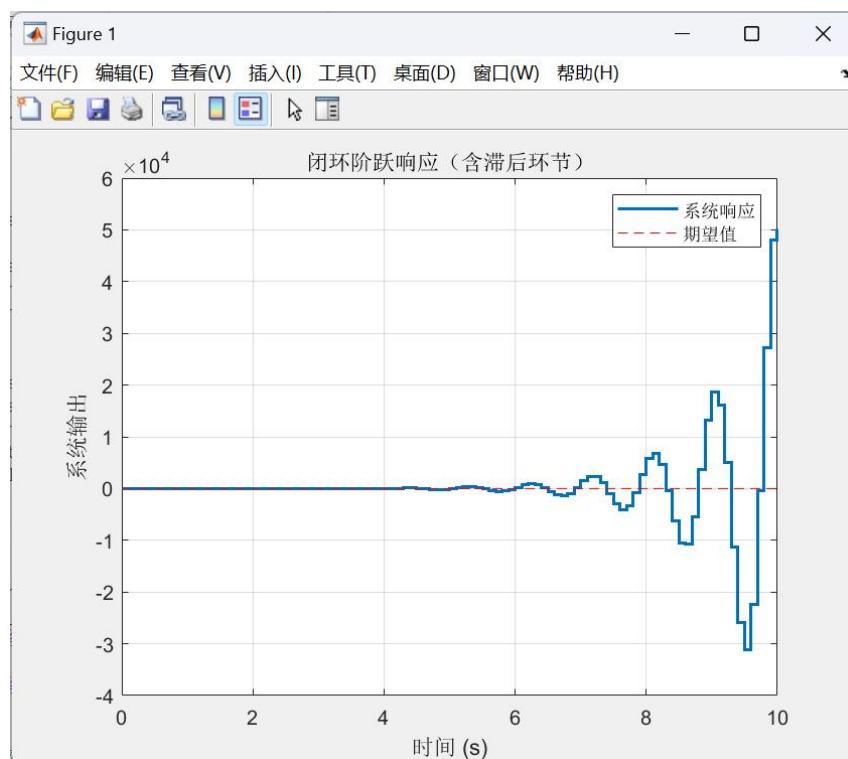
if ~isempty(settle_idx)
Ts = t(settle_idx(end)); % 调节时间
else
Ts = 0;
end

steady_state_error = abs(1 - steady_state_value); % 稳态误差
if steady_state_error < 0.01
fprintf('系统无静差。\\n');
else
fprintf('系统存在静差，大小为: %.4f\\n', steady_state_error);
end

% 输出系统性能指标
fprintf('系统性能指标（含滞后环节）:\\n');
fprintf('最大超调量 Mp: %.2f%%\\n', Mp);
fprintf('上升时间 Tr: %.4f s\\n', Tr);
fprintf('调节时间 Ts: %.4f s\\n', Ts);
fprintf('稳态输出值: %.4f\\n', steady_state_value);

```

>> ex\_6\_1\_4  
警告：使用状态空间表示法实现更高效的离散时滞建模。  
> 位置: tf/feedback\_ (第 570 行)  
位置: InputOutputModel/feedback (第 128 行)  
位置: ex\_6\_1\_4 (第 18 行)  
系统存在静差，大小为: 50135.0215  
系统性能指标（含滞后环节）：  
最大超调量 Mp: 0.00%  
上升时间 Tr: 1.9000 s  
调节时间 Ts: 9.9000 s  
稳态输出值: 50136.0215



我注意到 MATLAB 警告提示“使用状态空间表示法实现更高效的离散时滞建模。”，是因为我使用了带有延迟项  $\exp(-L*s)$  的传递函数  $G_p(s)$ ，然后通过  $c2d$  离散化后再与控制器进行 feedback 闭环连接。在这种情况下，MATLAB 建议改用状态空间  $ss$  的形式进行建模，以避免在使用  $tf$  传递函数时引发效率低下或数值不稳定的问题，尤其是在模型包含延迟或高阶系统时。

为解决以上警告，我改用状态空间的方法实现。

```
% 4) 被控对象加入滞后环节（使用状态空间避免警告）
% 采样周期与延迟
T = 0.1; % 采样周期 (s)
L = 0.2; % 延迟时间 (s)

% 连续系统传递函数
s = tf('s');
Gp_s = exp(-L*s) / (s + 1); % 被控对象带滞后项

% 转换为状态空间以提高建模效率
Gp_ss = ss(Gp_s); % 转换为状态空间
Gp_z = c2d(Gp_ss, T, 'zoh'); % 离散化 (ZOH 法)

% 定义 PI 控制器参数
Kp = 8;
Ki = 1.5;
z = tf('z', T);
D_pi = Kp * (1 + Ki * T * z / (z - 1)); % 离散 PI 控制器（一阶后向差分）

% 闭环系统
G_cl = feedback(D_pi * Gp_z, 1); % 闭环传递函数

% 仿真闭环系统阶跃响应
[y, t] = step(G_cl, 10); % 仿真 10 秒
steady_state_value = y(end); % 稳态值

% 绘图
figure;
stairs(t, y, 'LineWidth', 1.5);
hold on;
plot([t(1), t(end)], [1, 1], 'r--');
xlabel('时间 (s)');
ylabel('输出');
title('闭环阶跃响应（加入滞后环节）');
grid on;
legend('系统响应', '期望值');

% ----- 性能指标计算 -----
Mp = (max(y) - steady_state_value) / steady_state_value * 100; % 最大超调率 %

% 上升时间 Tr (从 10% 到 90%)
y_norm = y / steady_state_value;
t10 = t(find(y_norm >= 0.1, 1));
t90 = t(find(y_norm >= 0.9, 1));
Tr = t90 - t10;

% 调节时间 Ts (进入 ±2% 范围)
```

```

lower = 0.98 * steady_state_value;
upper = 1.02 * steady_state_value;
settle_idx = find(y < lower | y > upper);
if ~isempty(settle_idx)
Ts = t(settle_idx(end));
else
Ts = 0;
end

% 静态误差
steady_state_error = abs(1 - steady_state_value);
if steady_state_error < 0.01
fprintf('系统无静差\n');
else
fprintf('系统存在静差, 静差大小为: %.4f\n', steady_state_error);
end

% ----- 输出性能指标 -----
fprintf('性能指标 (加入滞后环节) :\n');
fprintf('最大超调量 Mp: %.2f%%\n', Mp);
fprintf('上升时间 Tr: %.4f s\n', Tr);
fprintf('调节时间 Ts: %.4f s\n', Ts);
fprintf('稳态输出值: %.4f\n', steady_state_value);

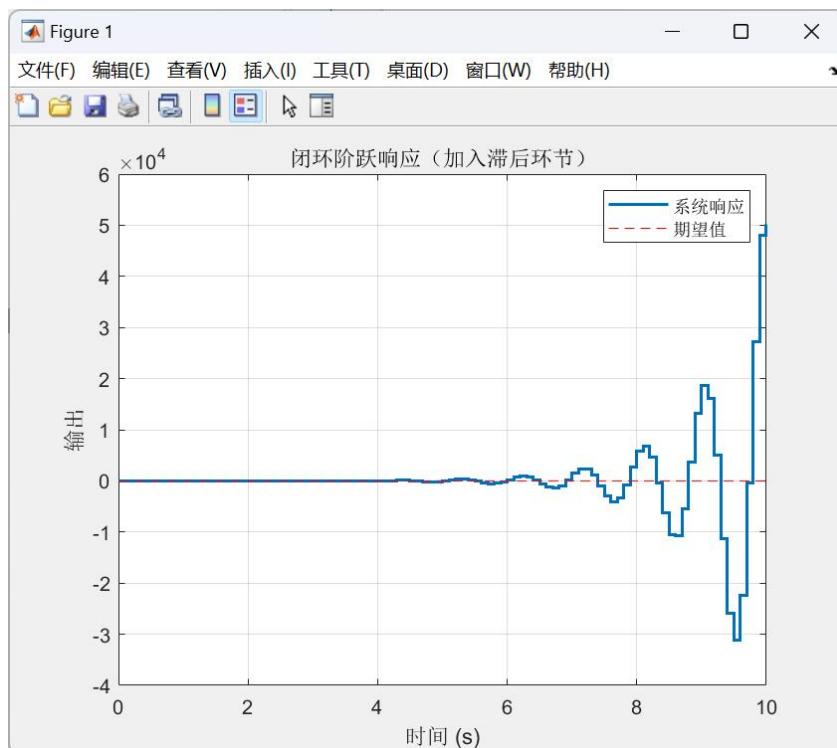
```

得到结果如下。

```

>> ex_6_1_4
系统存在静差, 静差大小为: 50135.0215
性能指标 (加入滞后环节) :
最大超调量 Mp: 0.00%
上升时间 Tr: 1.9000 s
调节时间 Ts: 9.9000 s
稳态输出值: 50136.0215

```



```

5)
% 5) 调节参数以保持最大超调量基本不变
T = 0.1; % 采样周期 (秒)
s = tf('s');
z = tf('z', T);

Gp_s = tf(1, [1 1]); % 连续系统  $G_p(s) = 1 / (s + 1)$ 
Gp_z = c2d(Gp_s, T, 'zoh'); % 离散化 (ZOH)

% PI 控制器设计参数
Kp = 1;
Ki = 1.65;
D_pi = Kp * (1 + Ki * T * z / (z - 1)); % 后向差分离散 PI 控制器

% 闭环传递函数
G_cl = feedback(D_pi * Gp_z, 1);

% 获取阶跃响应
[y, t] = step(G_cl, 10);
steady_value = y(end); % 稳态值

% 绘图
figure;
stairs(t, y, 'LineWidth', 1.5);
hold on;
plot([t(1), t(end)], [1 1], 'r--'); % 期望输出参考线
xlabel('时间 (s)');
ylabel('输出');
title('闭环阶跃响应 (调整后参数)');
legend('系统响应', '期望值');
grid on;

% 性能指标计算
Mp = (max(y) - steady_value) / steady_value * 100; % 最大超调(%)

% 上升时间 Tr (10% -> 90%)
y_norm = y / steady_value;
t10 = t(find(y_norm >= 0.1, 1));
t90 = t(find(y_norm >= 0.9, 1));
Tr = t90 - t10;

% 调节时间 Ts ( $\pm 2\%$  带内)
lower = 0.98 * steady_value;
upper = 1.02 * steady_value;
idx = find(y < lower | y > upper);
if ~isempty(idx)
Ts = t(idx(end));
else
Ts = 0;
end

% 静态误差
ess = abs(1 - steady_value);
if ess < 0.01
fprintf('系统无明显静态误差。\\n');
else
fprintf('系统存在静态误差，误差为: %.4f\\n', ess);

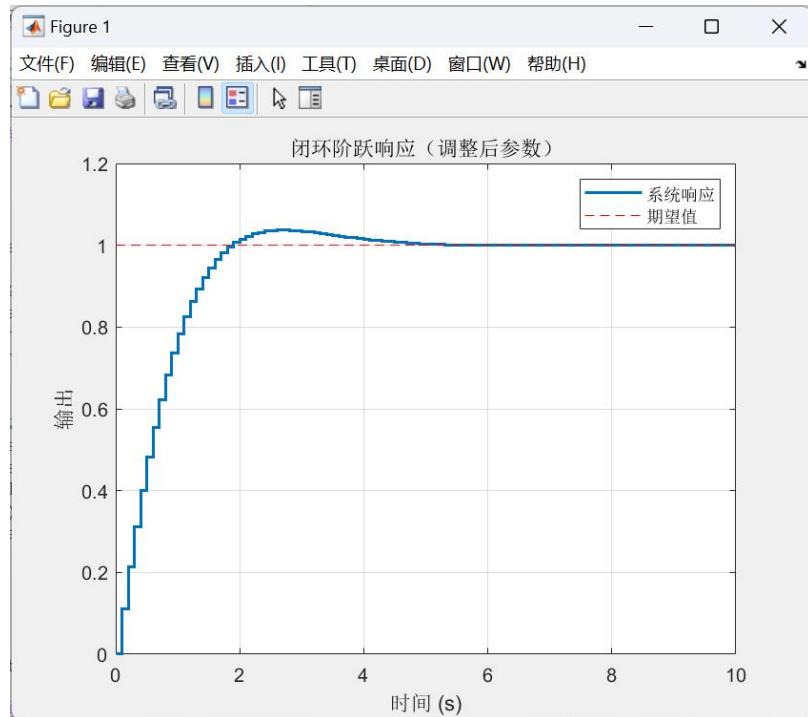
```

```

end

% 输出性能指标
fprintf('性能指标 (Kp = %.2f, Ki = %.2f) :\n', Kp, Ki);
fprintf('最大超调量 Mp: %.2f%\n', Mp);
fprintf('上升时间 Tr: %.4f s\n', Tr);
fprintf('调节时间 Ts: %.4f s\n', Ts);
fprintf('稳态输出值: %.4f\n', steady_value);

```



```

>> ex_6_1_5
系统无明显静态误差。
性能指标 (Kp = 1.00, Ki = 1.65) :
最大超调量 Mp: 3.67%
上升时间 Tr: 1.3000 s
调节时间 Ts: 3.7000 s
稳态输出值: 1.0000

```

3) 中的超调量为 3.77%，本小题修改完参数后超调量基本没有变化。

6)  
% 6) 达林算法设计控制器

```

T = 0.1; % 采样周期
T_0 = 0.2; % 延迟时间
s = tf('s');
z = tf('z', T);
N = round(T_0 / T);

% 用 Pade 近似代替时滞
delay_order = 1; % 一阶 Pade 近似即可
[num_delay, den_delay] = pade(T_0, delay_order);
Delay_approx = tf(num_delay, den_delay);

```

```

% 被控对象传递函数，代替 exp(-T_0*s)用 Pade 近似
G_s = Delay_approx * (1 / (s + 1));

% 离散化采用状态空间方法，避免警告
sys_ss = ss(G_s);
sys_d = c2d(sys_ss, T, 'zoh');

% 达林算法控制器设计
alpha = exp(-T / T_0);

D_z = (1 - (1 / z) * exp(-T)) * (1 - alpha) / ...
(1 - exp(-T)) / ...
(1 - alpha / z - (1 - alpha) * z^(-N-1));

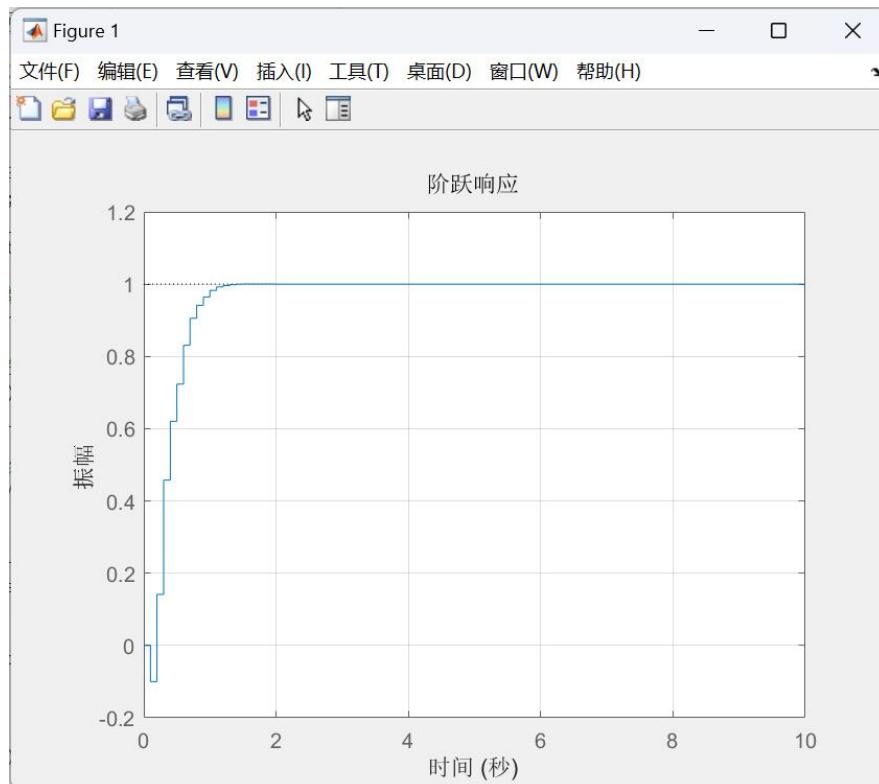
% 闭环系统
sys_cl = feedback(D_z * sys_d, 1);

% 绘制阶跃响应
figure;
step(sys_cl, 10);
grid on;

% 计算性能指标
info6 = stepinfo(sys_cl);
fprintf('最大超调量 Mp = %.2f%%\n', info6.Overshoot);
fprintf('上升时间 Tr = %.2f s\n', info6.RiseTime);
fprintf('调节时间 Ts = %.2f s\n', info6.SettlingTime);

>> ex_6_1_6
最大超调量 Mp = 0.06%
上升时间 Tr = 0.50 s
调节时间 Ts = 1.00 s

```



本小题，我发现图上最大幅值看起来是 1，而计算得到的最大超调量为 0.06%。这是因为图像上显示的是近似值，而 MATLAB 的 stepinfo 是基于数值精度计算的实际最大值。

7)

```
% 7) Smith 预估器控制器设计
T = 0.1; % 采样周期
L = 0.2; % 延迟时间
s = tf('s');
z = tf('z', T);

% 被控对象 Gp(s) 带延迟项
Gp_s = 1 / (s + 1);
Gp_z = c2d(exp(-L*s) * Gp_s, T, 'zoh'); % 离散化含延迟系统

% PI 控制器参数 (可替换为调优值)
Kp = 8;
Ki = 1.5;

% 后向差分离散 PI 控制器
D_pi_base = Kp * (1 + Ki * T * z / (z - 1));

% Smith 预估器部分，预测延迟带来的影响
D_z = c2d((1 - exp(-L*s)) * Gp_s, T, 'zoh');
D_pi = D_pi_base / (1 + D_pi_base * D_z); % Smith 校正器结构

% 闭环系统
G_cl = feedback(D_pi * Gp_z, 1);

% 阶跃响应仿真
[y, t] = step(G_cl, 10);
y_final = y(end);

% 绘制响应图
figure;
stairs(t, y, 'LineWidth', 1.5);
hold on;
plot([t(1), t(end)], [1, 1], 'r--'); % 目标值参考线
xlabel('时间 (s)');
ylabel('输出');
title('Smith 预估器闭环响应');
legend('系统响应', '期望值');
grid on;

% 计算性能指标
Mp = (max(y) - y_final) / y_final * 100; % 最大超调量 (%)

% 上升时间 Tr (10% 到 90%)
y_norm = y / y_final;
t10 = t(find(y_norm >= 0.1, 1));
t90 = t(find(y_norm >= 0.9, 1));
Tr = t90 - t10;

% 调节时间 Ts (±2% 误差带)
```

```

lower_bound = 0.98 * y_final;
upper_bound = 1.02 * y_final;
out_of_band = find(y < lower_bound | y > upper_bound);
if ~isempty(out_of_band)
Ts = t(out_of_band(end));
else
Ts = 0;
end

% 静态误差判断
ess = abs(1 - y_final);
if ess < 0.01
fprintf('系统无明显静态误差。\\n');
else
fprintf('系统存在静态误差，误差为: %.4f\\n', ess);
end

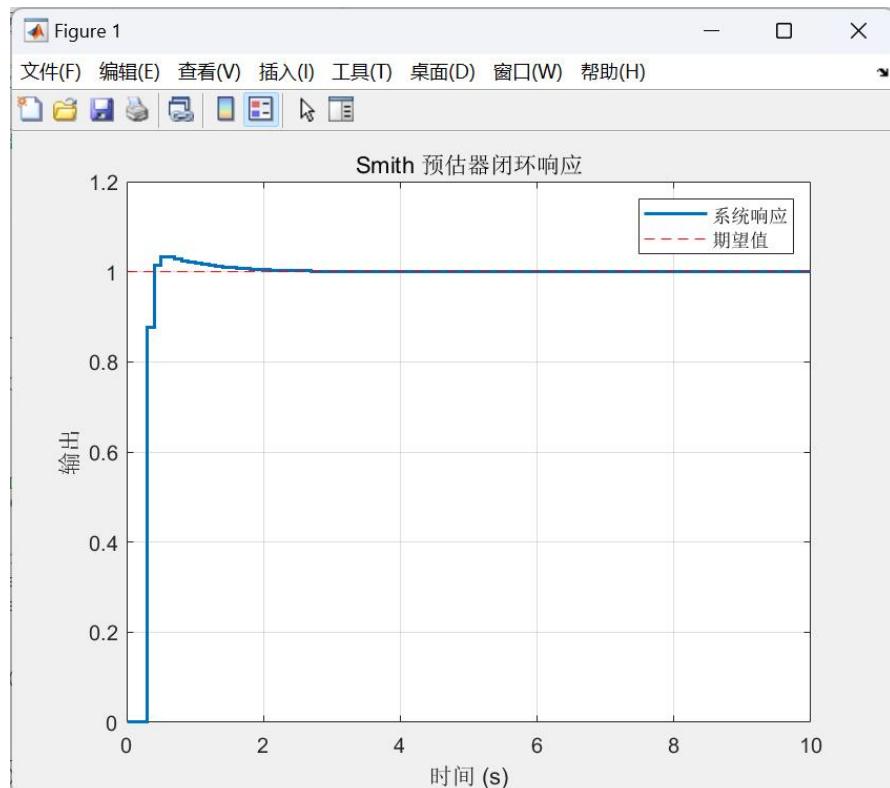
% 输出性能指标
fprintf('性能指标 (Smith 预估器) :\n');
fprintf('最大超调量 Mp: %.2f%%\\n', Mp);
fprintf('上升时间 Tr: %.4f s\\n', Tr);
fprintf('调节时间 Ts: %.4f s\\n', Ts);
fprintf('稳态输出值: %.4f\\n', y_final);

```

```

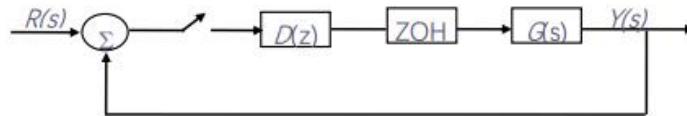
>> ex_6_1_7
系统无明显静态误差。
性能指标 (Smith 预估器) :
最大超调量 Mp: 3.37%
上升时间 Tr: 0.1000 s
调节时间 Ts: 0.9000 s
稳态输出值: 1.0000

```



## 7.1

例 7.1 图中  $G(s) = \frac{10}{s(s+1)}$ ,  $T = 1s$  时



控制系统框图

1) 阶跃输入信号作用下, 有纹波最小拍控制器、无纹波最小拍控制器设计;

2) 使 A 中采样点间增加 4 个点进行显示, 观察有纹波、无纹波控制器的区别;

当  $G(s) = \frac{1}{s}$ ,  $T = 1s$  时

3) 阶跃输入信号作用下, 无纹波最小拍控制器设计  $D_1(z)$ ;

4) 斜坡输入信号作用下, 无纹波最小拍控制器设计  $D_2(z)$ ;

5) 选作: 折衷设计  $D(z)$ , 要求阶跃响应超调小于 30%, 斜坡响应 12 拍达到稳态 ( $\pm 5\%$ )。

1)

```
% 阶跃输入信号作用下, 有纹波/无纹波最小拍控制器设计
% 参数定义
s = tf('s');
T = 1;
z = tf('z', T);

% 被控对象及参考输入
Gs = 10 / (s * (s + 1));
Rz = 1 / (1 - 1 / z);
Gz = c2d(Gs, T, 'zoh');

%% 有纹波最小拍控制器
WBz1 = 1 / z;
Yz1 = WBz1 * Rz;
[num_Yz1, den_Yz1] = tfdata(Yz1, 'v');
result_Yz1 = polynomial_process(num_Yz1, den_Yz1);

Uz1 = Yz1 / Gz;
[num1, den1] = tfdata(Uz1, 'v');
result_Uz1 = polynomial_process(num1, den1);

%% 无纹波最小拍控制器
zeros_Gz = zero(Gz);
WBz2 = 1/(1 - zeros_Gz) * 1 / z * (1 - zeros_Gz * 1 / z);

Yz2 = WBz2 * Rz;
[num_Yz2, den_Yz2] = tfdata(Yz2, 'v');
result_Yz2 = polynomial_process(num_Yz2, den_Yz2);

Uz2 = Yz2 / Gz;
```

```

[num2, den2] = tfdata(Uz2, 'v');
result_Uz2 = polynomial_process(num2, den2);

%% 离散时间向量
t_discrete = 0:1:12;

% 有纹波控制器零阶保持连续信号
[t_continuous1, u_continuous1] = zero_order_hold(t_discrete, result_Uz1);
% 无纹波控制器零阶保持连续信号
[t_continuous2, u_continuous2] = zero_order_hold(t_discrete, result_Uz2);

% 系统响应仿真
y_step1 = lsim(Gs, u_continuous1, t_continuous1);
y_step2 = lsim(Gs, u_continuous2, t_continuous2);

%% 绘图
figure('Position', [100, 100, 1000, 500]);

% 子图 1: 有纹波控制器
subplot(1, 2, 1);
hold on;
plot(t_continuous1, y_step1, 'b-', 'LineWidth', 2, 'DisplayName', '连续响应');
stem(t_discrete, result_Yz1, 'r-o', 'LineWidth', 1.5, 'MarkerFaceColor', 'r', 'DisplayName', '离散序列');
title('有纹波最小拍控制器响应', 'FontSize', 14);
xlabel('时间 (秒)', 'FontSize', 12);
ylabel('输出响应', 'FontSize', 12);
ylim([0 1.4]);
legend('Location', 'best');
grid on;
hold off;

% 子图 2: 无纹波控制器
subplot(1, 2, 2);
hold on;
plot(t_continuous2, y_step2, 'b-', 'LineWidth', 2, 'DisplayName', '连续响应');
stem(t_discrete, result_Yz2, 'r-o', 'LineWidth', 1.5, 'MarkerFaceColor', 'r', 'DisplayName', '离散序列');
title('无纹波最小拍控制器响应', 'FontSize', 14);
xlabel('时间 (秒)', 'FontSize', 12);
ylabel('输出响应', 'FontSize', 12);
ylim([0 1.4]);
legend('Location', 'best');
grid on;
hold off;

%% 函数
function [t_continuous, y_continuous] = zero_order_hold(t_discrete,
y_discrete, dt_continuous)
if nargin < 3
    dt_continuous = (t_discrete(2) - t_discrete(1)) / 50;
end
t_continuous = t_discrete(1):dt_continuous:t_discrete(end);
y_continuous = zeros(size(t_continuous));
for i = 1:length(t_continuous)

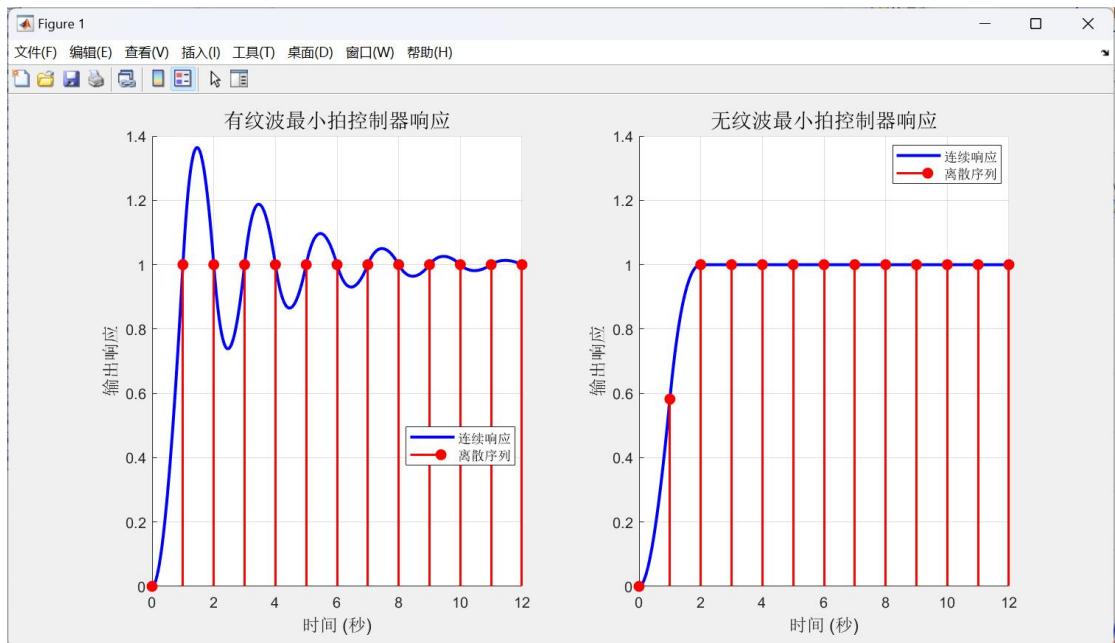
```

```

    idx = find(t_discrete <= t_continuous(i), 1, 'last');
    if ~isempty(idx)
        y_continuous(i) = y_discrete(idx);
    end
end
end

function result = polynomial_process(num, den)
if nargin < 2
    error('需要提供分子和分母多项式系数');
end
temp = zeros(1, length(den));
result = zeros(1, 13);
for i = 1:13
    result(i) = num(1) / den(1);
    for j = 1:length(den)
        temp(j) = den(j) * result(i);
        num(j) = num(j) - temp(j);
    end
    for k = 1:length(den)-1
        num(k) = num(k + 1);
    end
    num(length(den)) = 0;
end
end

```



```

2)
clc; clear;

% 参数定义
s = tf('s');
T = 1;
z = tf('z', T);

% 被控对象
Gs = 10 / (s*(s+1));

```

```

% 离散系统及参考输入
Gz = c2d(Gs, T, 'zoh');
Rz = 1 / (1 - 1/z);

%% 有纹波最小拍控制器设计

WBz1 = 1 / z; % 有纹波控制器的闭环传递函数

% 计算 Y(z) 和 U(z)
Yz1 = WBz1 * Rz;
[num_Yz1, den_Yz1] = tfdata(Yz1, 'v');
result_Yz1 = polynomial_process(num_Yz1, den_Yz1);

Uz1 = Yz1 / Gz;
[num1, den1] = tfdata(Uz1, 'v');
result_Uz1 = polynomial_process(num1, den1);

% 离散时间向量
t_discrete = 0:1:12;

% 离散控制信号用零阶保持器转换为连续信号
[t_continuous1, u_continuous1] = zero_order_hold(t_discrete, result_Uz1);

% 系统响应仿真
y_step1 = lsim(Gs, u_continuous1, t_continuous1);

% 选取部分离散点绘制
t_select1 = zeros(1, 4);
y_select1 = zeros(1, 4);
for i = 1:4
    t_select1(i) = t_continuous1(51 + 10*i);
    y_select1(i) = y_step1(51 + 10*i);
end

%% 无纹波最小拍控制器设计

% 计算无纹波控制器的闭环传递函数 WBz2
zeros_Gz = zero(Gz);
WBz2 = 1 / (1 - zeros_Gz) * 1 / z * (1 - zeros_Gz * 1 / z);

% 计算 Y(z) 和 U(z)
Yz2 = WBz2 * Rz;
[num_Yz2, den_Yz2] = tfdata(Yz2, 'v');
result_Yz2 = polynomial_process(num_Yz2, den_Yz2);

Uz2 = Yz2 / Gz;
[num2, den2] = tfdata(Uz2, 'v');
result_Uz2 = polynomial_process(num2, den2);

% 离散控制信号用零阶保持器转换为连续信号
[t_continuous2, u_continuous2] = zero_order_hold(t_discrete, result_Uz2);

% 系统响应仿真
y_step2 = lsim(Gs, u_continuous2, t_continuous2);

% 选取部分离散点绘制
t_select2 = zeros(1, 4);
y_select2 = zeros(1, 4);

```

```

for i = 1:4
    t_select2(i) = t_continuous2(51 + 10*i);
    y_select2(i) = y_step2(51 + 10*i);
end

%% 绘图
figure('Position', [100, 100, 1000, 500]);

% 子图 1: 有纹波最小拍控制器响应
subplot(1,2,1);
hold on;
plot(t_continuous1, y_step1, 'b-', 'LineWidth', 2, 'DisplayName', '连续信号');
stem(t_discrete, result_Yz1, 'r-o', 'LineWidth', 1.5, 'MarkerFaceColor', 'r', 'DisplayName', '离散序列');
stem(t_select1, y_select1, 'go', 'LineWidth', 1.5, 'MarkerSize', 8, 'MarkerFaceColor', 'g', 'DisplayName', '选取点');
ylim([0 1.4]);
grid on;
xlabel('时间 (秒)', 'FontSize', 12);
ylabel('信号幅度', 'FontSize', 12);
title('有纹波最小拍控制器响应', 'FontSize', 14);
legend('Location', 'best');
hold off;

% 子图 2: 无纹波最小拍控制器响应
subplot(1,2,2);
hold on;
plot(t_continuous2, y_step2, 'b-', 'LineWidth', 2, 'DisplayName', '连续信号');
stem(t_discrete, result_Yz2, 'r-s', 'LineWidth', 1.5, 'MarkerFaceColor', 'r', 'DisplayName', '离散序列');
stem(t_select2, y_select2, 'go', 'LineWidth', 1.5, 'MarkerSize', 8, 'MarkerFaceColor', 'g', 'DisplayName', '选取点');
ylim([0 1.4]);
grid on;
xlabel('时间 (秒)', 'FontSize', 12);
ylabel('信号幅度', 'FontSize', 12);
title('无纹波最小拍控制器响应', 'FontSize', 14);
legend('Location', 'best');
hold off;

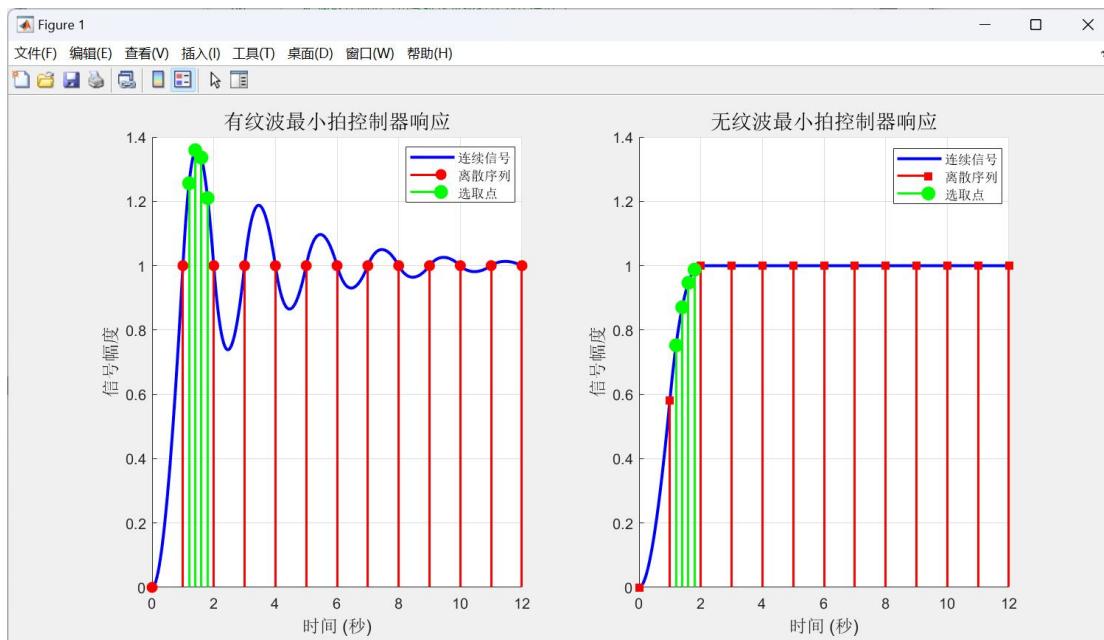
%% 函数
function [t_continuous, y_continuous] = zero_order_hold(t_discrete,
y_discrete, dt_continuous)
    if nargin < 3
        dt_continuous = (t_discrete(2) - t_discrete(1)) / 50;
    end
    t_continuous = t_discrete(1):dt_continuous:t_discrete(end);
    y_continuous = zeros(size(t_continuous));
    for i = 1:length(t_continuous)
        idx = find(t_discrete <= t_continuous(i), 1, 'last');
        if ~isempty(idx)
            y_continuous(i) = y_discrete(idx);
        end
    end
end

```

```

function result = polynomial_process(num, den)
    if nargin < 2
        error('需要提供分子和分母多项式系数');
    end
    temp = zeros(1, length(den));
    result = zeros(1, 13);
    for i = 1:13
        result(i) = num(1) / den(1);
        for j = 1:length(den)
            temp(j) = den(j) * result(i);
            num(j) = num(j) - temp(j);
        end
        for k = 1:length(den) - 1
            num(k) = num(k + 1);
        end
        num(length(den)) = 0;
    end
end

```



对于有纹波的控制器，其输出曲线在达到稳态值前呈现出明显的振荡特性，即在目标值附近存在多次上下波动。尽管最终能收敛到稳态，但过程中的过冲与下冲使得系统存在不必要的能量消耗与不稳定性，体现为图中采样点之间插值曲线出现多次起伏。相对而言，无纹波的控制器响应则更加平稳，其输出在达到稳态值前基本不发生明显振荡，通常呈现出单调上升或轻微超调后快速收敛的特性。采样点之间插值的曲线平滑且无明显波峰波谷，说明系统具有良好的阻尼性能和稳态控制能力。这类控制器虽然在响应速度上可能略慢于有纹波的控制器，但在稳态误差和控制稳定性方面表现更优，适用于对平稳性和精度要求较高的系统。

3)  
%  $G(s)=1/s$ ,  $T=1s$ , 阶跃输入信号作用下, 无纹波最小拍控制器设计  
clc; clear;

```

% 参数定义
s = tf('s');
T = 1;
z = tf('z', T);

% 被控对象 G(s) = 1/s (纯积分环节)
Gs = 1 / s;
Rz = 1 / (1 - 1 / z);
Gz = c2d(Gs, T, 'zoh'); % Gz = Z 变换后得到的离散模型
WBz = 1 / z; % 无纹波情况

% 无纹波最小拍控制器设计
Dz = WBz / (Gz * (1 - WBz));
Yz = WBz * Rz;

% 提取 Yz 序列
[num_Yz, den_Yz] = tfdata(Yz, 'v');
result_Yz = polynomial_process(num_Yz, den_Yz);

% Uz 序列 (控制器输出序列)
Uz = Yz / Gz;
[num_Uz, den_Uz] = tfdata(Uz, 'v');
result_Uz = polynomial_process(num_Uz, den_Uz);

% 离散到连续转换
t_discrete = 0:1:12;
[t_continuous, u_continuous] = zero_order_hold(t_discrete, result_Uz);
y_step = lsim(Gs, u_continuous, t_continuous);

% 选取 4 个用于标记的点
for i = 1:4
    t_select(i) = t_continuous(51 + 10 * i);
    y_select(i) = y_step(51 + 10 * i);
end

%% 绘图
figure('Position', [100, 100, 800, 500]);
subplot(1,1,1);
hold on;

% 连续响应
plot(t_continuous, y_step, 'b-', 'LineWidth', 2, 'DisplayName', '连续信号');

% 离散输出序列
stem(t_discrete, result_Yz, 'r-s', 'LineWidth', 1.5, 'MarkerFaceColor', 'r',
'DisplayName', '离散序列');

% 标记点
stem(t_select, y_select, 'go', 'LineWidth', 1.5, 'MarkerSize', 8,
'MarkerFaceColor', 'g', 'DisplayName', '选取点');

grid on;
xlabel('时间 (秒)', 'FontSize', 12);
ylabel('信号幅度', 'FontSize', 12);
title('无纹波最小拍控制器响应', 'FontSize', 14);
legend('Location', 'best');

```

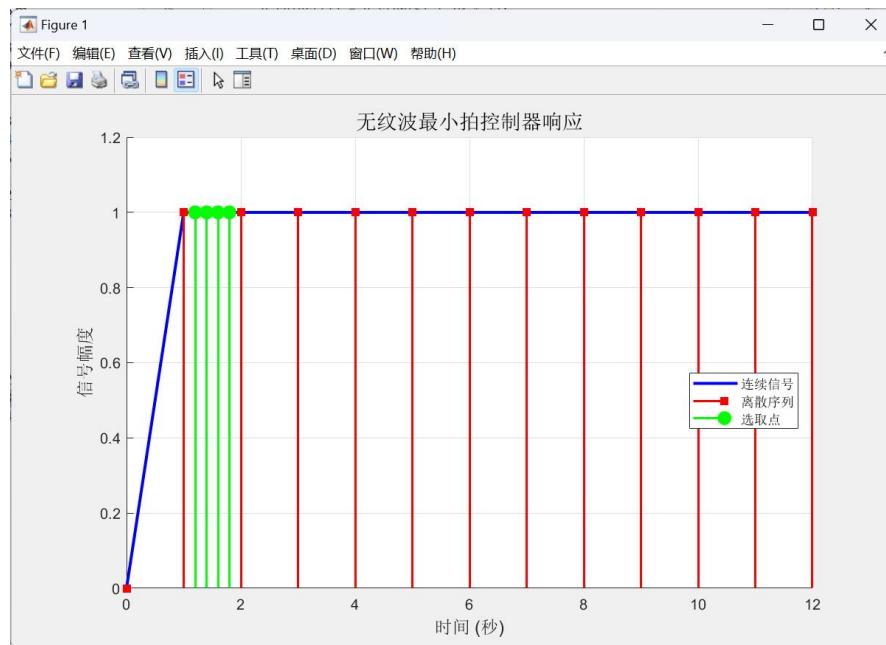
```

hold off;

%% 函数
function [t_continuous, y_continuous] = zero_order_hold(t_discrete,
y_discrete, dt_continuous)
if nargin < 3
    dt_continuous = (t_discrete(2) - t_discrete(1))/50;
end
t_continuous = t_discrete(1):dt_continuous:t_discrete(end);
y_continuous = zeros(size(t_continuous));
for i = 1:length(t_continuous)
    idx = find(t_discrete <= t_continuous(i), 1, 'last');
    if ~isempty(idx)
        y_continuous(i) = y_discrete(idx);
    end
end
end

function result = polynomial_process(num, den)
if nargin < 2
    error('需要提供分子和分母多项式系数');
end
temp = zeros(1, length(den));
result = zeros(1, 13);
for i = 1:13
    result(i) = num(1)/den(1);
    for j = 1:length(den)
        temp(j) = den(j)*result(i);
        num(j) = num(j) - temp(j);
    end
    for k = 1:length(den)-1
        num(k) = num(k+1);
    end
    num(length(den)) = 0;
end
end

```



```

4)
% G(s)=1/s, T=1s, 斜坡输入信号作用下, 无纹波最小拍控制器设计
clc; clear;

s = tf('s');
T = 1;
z = tf('z', T);

% 被控对象 G(s) = 1/s
Gs = 1 / s;
Gz = c2d(Gs, T, 'zoh'); % 离散模型
WBz = 1 / z; % 无纹波条件
Rz = T * z / (z - 1)^2; % 斜坡输入

% 控制器设计
Dz = WBz / (Gz * (1 - WBz));
Yz = WBz * Rz;
[num_Yz, den_Yz] = tfdata(Yz, 'v');
result_Yz = polynomial_process(num_Yz, den_Yz);

Uz = Yz / Gz;
[num_Uz, den_Uz] = tfdata(Uz, 'v');
result_Uz = polynomial_process(num_Uz, den_Uz);

% 连续时间响应
t_discrete = 0:1:12;
[t_continuous, u_continuous] = zero_order_hold(t_discrete, result_Uz);
y_step = lsim(Gs, u_continuous, t_continuous);

% 标记 4 个点
for i = 1:4
    t_select(i) = t_continuous(51 + 10 * i);
    y_select(i) = y_step(51 + 10 * i);
end

% 绘图
figure('Position', [100, 100, 800, 500]);
hold on;
plot(t_continuous, y_step, 'b-', 'LineWidth', 2, 'DisplayName', '连续信号');
stem(t_discrete, result_Yz, 'r-s', 'LineWidth', 1.5, 'MarkerFaceColor', 'r',
'DisplayName', '离散序列');
stem(t_select, y_select, 'go', 'LineWidth', 1.5, 'MarkerSize', 8,
'MarkerFaceColor', 'g', 'DisplayName', '选取点');
grid on;
xlabel('时间 (秒)', 'FontSize', 12);
ylabel('信号幅度', 'FontSize', 12);
title('无纹波最小拍控制器响应', 'FontSize', 14);
legend('Location', 'best');
hold off;

%% 函数
function [t_continuous, y_continuous] = zero_order_hold(t_discrete,
y_discrete, dt_continuous)
if nargin < 3
    dt_continuous = (t_discrete(2) - t_discrete(1))/50;
end
t_continuous = t_discrete(1):dt_continuous:t_discrete(end);

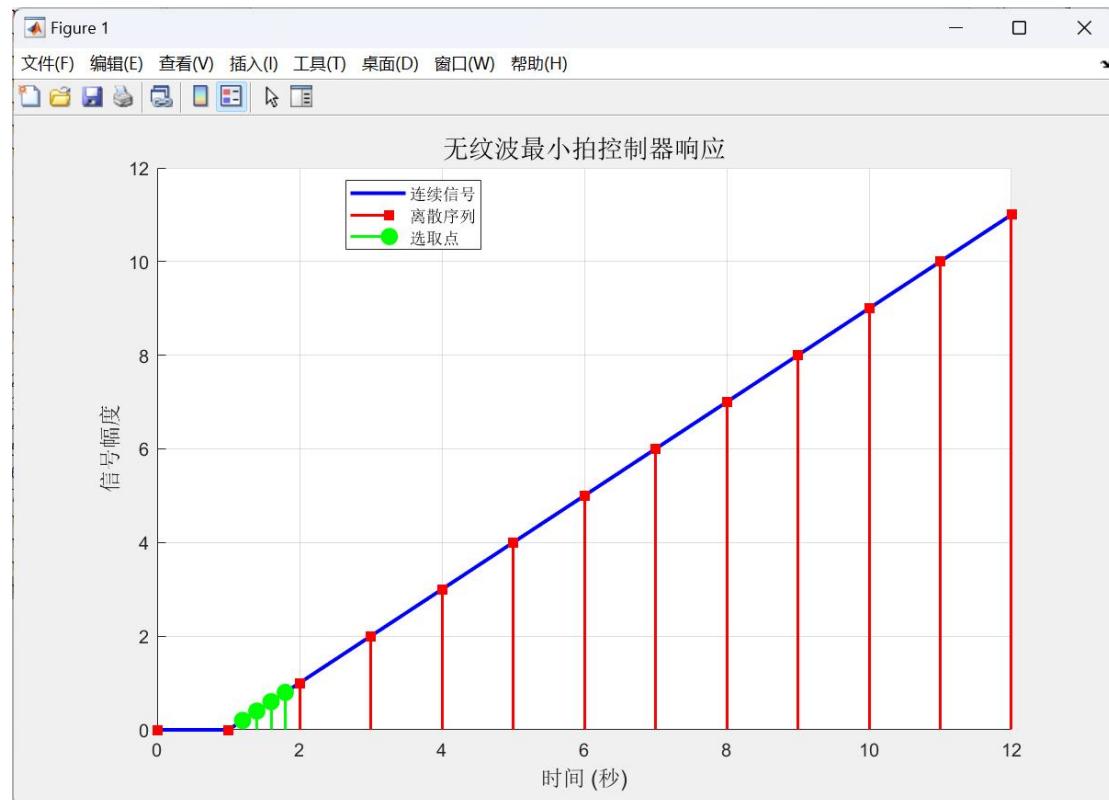
```

```

y_continuous = zeros(size(t_continuous));
for i = 1:length(t_continuous)
    idx = find(t_discrete <= t_continuous(i), 1, 'last');
    if ~isempty(idx)
        y_continuous(i) = y_discrete(idx);
    end
end

function result = polynomial_process(num, den)
if nargin < 2
    error('需要提供分子和分母多项式系数');
end
temp = zeros(1, length(den));
result = zeros(1, 13);
for i = 1:13
    result(i) = num(1)/den(1);
    for j = 1:length(den)
        temp(j) = den(j)*result(i);
        num(j) = num(j) - temp(j);
    end
    for k = 1:length(den)-1
        num(k) = num(k+1);
    end
    num(length(den)) = 0;
end
end

```



5)

相比于“最小拍控制器”强调最快响应，折衷设计可以在控制器分子引入零点或对极点位置适当调整，使响应速度略降、但鲁棒性与性能更稳。控制器设计为如下表示形式，

$$D(z) = \frac{1 - \alpha z^{-1}}{1 - \beta z^{-1}}$$

其中， $\alpha$  主要控制零点位置，影响系统的超调和响应速度； $\beta$  主要控制极点位置，影响系统的稳定性和收敛速度。降低超调一般是让系统阻尼比变大，闭环极点远离单位圆上的虚轴（降低系统振荡）。增大 $\beta$ 会让极点更靠近单位圆内，增强稳定性，通常减小超调。减小 $\alpha$ 相当于减小零点对系统响应的加速作用，也会减少超调。

最终取 $\alpha = 0.55$ ,  $\beta = 0.84$ , 代码如下。

```
% 参数定义
s = tf('s');
T = 1;
z = tf('z', T);

% 被控对象 G(s) = 1/s (纯积分环节)
Gs = 1 / s;
Gz = c2d(Gs, T, 'zoh'); % ZOH 离散模型
t_discrete = 0:1:12;

%% 控制器 D(z) (满足超调<30%, 斜坡 12 拍稳态)
alpha = 0.55;
beta = 0.84;
Dz = (1 - alpha*z^(-1)) / (1 - beta*z^(-1));

% 闭环传递函数 W(z)
Wz = (Dz * Gz) / (1 + Dz * Gz);

%% --- 阶跃响应分析 ---
Rz_step = 1 / (1 - z^(-1)); % Z 域阶跃输入
Yz_step = Wz * Rz_step; % 输出 Y(z)
[num_Ys, den_Ys] = tfdata(Yz_step, 'v');
result_Ys = polynomial_process(num_Ys, den_Ys); % 得到 y[k]

Uz_step = Yz_step / Gz;
[num_Us, den_Us] = tfdata(Uz_step, 'v');
result_Us = polynomial_process(num_Us, den_Us); % 得到 u[k]

[t_continuous_s, u_continuous_s] = zero_order_hold(t_discrete, result_Us);
y_step = lsim(Gs, u_continuous_s, t_continuous_s);

% --- 计算超调量 ---
steady_value = mean(y_step(end-5:end)); % 稳态值取末尾均值
peak_value = max(y_step);
overshoot = (peak_value - steady_value) / steady_value * 100;
fprintf('阶跃响应超调量: %.2f%\n', overshoot);

%% --- 斜坡响应分析 ---
Rz_ramp = z / (z - 1)^2; % Z 域斜坡输入
Yz_ramp = Wz * Rz_ramp;
[num_Yr, den_Yr] = tfdata(Yz_ramp, 'v');
result_Yr = polynomial_process(num_Yr, den_Yr); % y[k]

Uz_ramp = Yz_ramp / Gz;
[num_Ur, den_Ur] = tfdata(Uz_ramp, 'v');
result_Ur = polynomial_process(num_Ur, den_Ur);
```

```

[t_continuous_r, u_continuous_r] = zero_order_hold(t_discrete, result_Ur);
y_ramp = lsim(Gs, u_continuous_r, t_continuous_r);

% --- 自动判断斜坡稳态是否在 12 拍内达到 ±5% ---
ramp_ref = 0:12; % 理想参考: r[k] = k
y12 = result_Yr(1:13); % y[0] 到 y[12]
err = abs(y12 - ramp_ref);
tolerance = 0.05 * ramp_ref;
tolerance(ramp_ref == 0) = 0.05; % 避免 t=0 除 0 错误
within_bounds = err <= tolerance;

satisfied = all(within_bounds(9:end)); % 例如从第 9 拍到第 12 拍都满足
if satisfied
    disp('斜坡响应在第 12 拍前达到±5%稳态。');
else
    disp('斜坡响应未在第 12 拍达到稳态。');
end

%% --- 标注点 ---
for i = 1:4
    t_select(i) = t_continuous_s(51 + 10 * i);
    y_select(i) = y_step(51 + 10 * i);
end

%% --- 绘图 ---
figure('Position', [100, 100, 1000, 450]);

% 阶跃响应图
subplot(1,2,1);
plot(t_continuous_s, y_step, 'b-', 'LineWidth', 2, 'DisplayName', '连续信号');
hold on;
stem(t_discrete, result_Ys, 'r-s', 'LineWidth', 1.5, 'MarkerFaceColor', 'r',
'DisplayName', '离散序列');
stem(t_select, y_select, 'go', 'LineWidth', 1.5, 'MarkerSize', 8,
'MarkerFaceColor', 'g', 'DisplayName', '选取点');
ylim([0 1.4]);
xlabel('时间 (秒)', 'FontSize', 12);
ylabel('信号幅度', 'FontSize', 12);
title('阶跃响应 (折衷控制器)', 'FontSize', 14);
legend('Location', 'best'); grid on; hold off;

% 斜坡响应图
subplot(1,2,2);
plot(t_continuous_r, y_ramp, 'b-', 'LineWidth', 2, 'DisplayName', '连续信号');
hold on;
stem(t_discrete, result_Yr, 'r-s', 'LineWidth', 1.5, 'MarkerFaceColor', 'r',
'DisplayName', '离散序列');
yline(12 * 1.05, 'k--', '5%上界'); yline(12 * 0.95, 'k--', '5%下界');
ylim([0 15]);
xlabel('时间 (秒)', 'FontSize', 12);
ylabel('信号幅度', 'FontSize', 12);
title('斜坡响应 (折衷控制器)', 'FontSize', 14);
legend('Location', 'best'); grid on; hold off;

%% --- 支持函数 ---

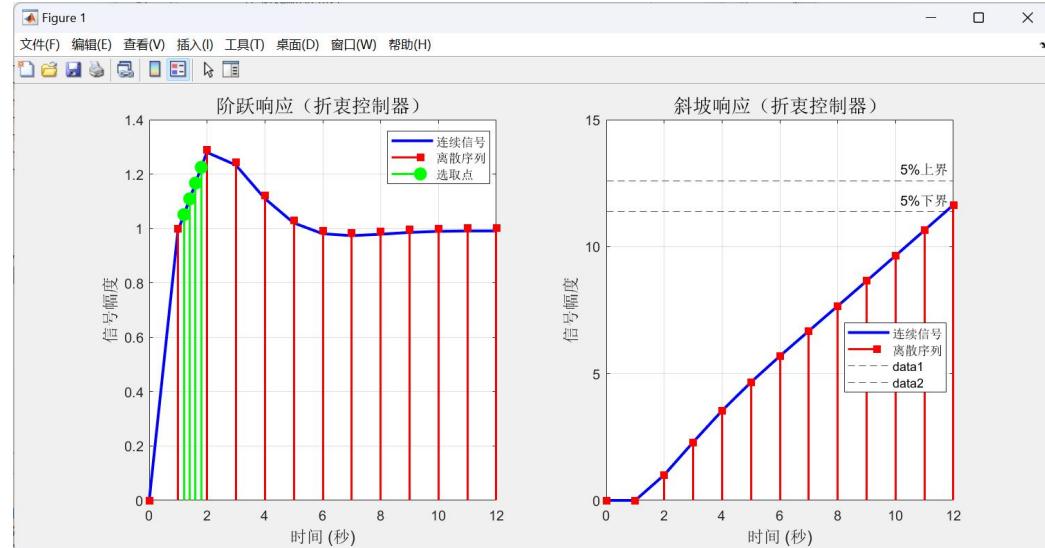
```

```

function [t_continuous, y_continuous] = zero_order_hold(t_discrete,
y_discrete, dt_continuous)
    if nargin < 3
        dt_continuous = (t_discrete(2) - t_discrete(1)) / 50;
    end
    t_continuous = t_discrete(1):dt_continuous:t_discrete(end);
    y_continuous = zeros(size(t_continuous));
    for i = 1:length(t_continuous)
        idx = find(t_discrete <= t_continuous(i), 1, 'last');
        if ~isempty(idx)
            y_continuous(i) = y_discrete(idx);
        end
    end
end

function result = polynomial_process(num, den)
    if nargin < 2
        error('需要提供分子和分母多项式系数');
    end
    temp = zeros(1, length(den));
    result = zeros(1, 13);
    for i = 1:13
        result(i) = num(1)/den(1);
        for j = 1:length(den)
            temp(j) = den(j)*result(i);
            num(j) = num(j) - temp(j);
        end
        for k = 1:length(den)-1
            num(k) = num(k+1);
        end
        num(length(den)) = 0;
    end
end

```



>> ex\_7\_1\_5

阶跃响应超调量: 29.13%

斜坡响应在第 12 拍前达到±5% 稳态。

阶跃响应部分增加了超调量计算，斜坡响应部分会自动判断在第 12 拍之前是否满足稳态（ $\pm 5\%$ ）的要求。从输出结果看，满足要求。