附录：实验用MATLAB代码

主程序

Attention: 当您运行以下代码的时候，请通过注释其余模块，一个一个模块的运行。因为每个模块都会大量计算传递函数的阶跃响应，并且作10张以上的时域响应曲线。一次性全部运行很可能导致MATLAB卡顿、崩溃。

```
main.m
```

```
clc;clear;close all;

%{
<program>  Copyright (C) <2023>  <Huaqian Chin>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
%}

%% 初始默认参数
Kp_s = 2.8;
Ki_s = 0.15;
Kd_s = 3;
p = 2;
t = 0:0.05:100;  % 定义时间区间

%% Pump Function
n = [1 0];  % s
gp = tf(1, n);

%% Patient Function
n = [1, 2*p, p*p];
g = tf(1, n);

%% H(s)
H = tf(1, 1);

row(Kp_s,Ki_s,Kd_s,t,gp,g,H);


%% --------------case 1.1.1 :  PD  ------------------
%
%        R = 10 (t>10);  N(s) = 0;  Td(s) = 0;
%
%  在这个模块中，将PD接入了系统的Gc(s)的位置
%  实现了对两个参数 (Kp & Kd) 进行了两组分析
%   1. 保持Kp不变，调整Kd的值----以1为步长，在[1,5]的范围内生成5张系统时域图
%   2. 保持Kd不变，调整Kp的值----以2为步长，在[2,10]的范围内生成5张系统时域图
% --------------------------------------------------
PD(Kp_s,Ki_s,Kd_s,t,gp,g,H);
```

```
%% --------------case 1.1.2 :  PI  ------------------
%
%       R = 10 (t>10);  N(s) = 0;  Td(s) = 0;
%
%  在这个模块中，将PI接入了系统的Gc(s)的位置
%  实现了对两个参数 (Kp & Ki) 进行了两组分析
%   1. 保持Kp不变，调整Ki的值----以0.15为步长，在[0.15,0.75]的范围内生成5张系统时域图
%   2. 保持Ki不变，调整Kp的值----以0.8为步长，在[0.8,4]的范围内生成5张系统时域图
% --------------------------------------------------
PI(Kp_s,Ki_s,Kd_s,t,gp,g,H)




%% --------------case 1.2 :  PID  ------------------
%
%       R = 10 (t>10);  N(s) = 0;  Td(s) = 0;
%
%  在这个模块中，将PID接入了系统的Gc(s)的位置
%  实现了对三个参数 (Kp & Ki) 进行了三组分析
%   1. 保持Kp & Ki不变，调整Kd的值----以0.8为步长，在[0.8,4]的范围内生成5张系统时域图
%   2. 保持Kp & Kd不变，调整Ki的值----以0.15为步长，在[0.15,0.75]的范围内生成5张系统时域图
%   2. 保持Ki & Kd不变，调整Kp的值----以0.6为步长，在[0.6,3]的范围内生成5张系统时域图
% --------------------------------------------------
PID(Kp_s,Ki_s,Kd_s,t,gp,g,H)


%% --------------case 2 :  PID with Noise  ------------------
%
%           R = 0;  N(s) = 0;  Td(s) = 50/s ;
%
%  在这个模块中，将PID接入了系统的Gc(s)的位置
%  实现了对三个参数 (Kp & Ki & Kd) 进行了三组分析
%   1. 保持Kp & Ki不变，调整Kd的值----以0.8为步长，在[0.8,4]的范围内生成5张系统时域图
%   2. 保持Kp & Kd不变，调整Ki的值----以0.15为步长，在[0.15,0.75]的范围内生成5张系统时域图
%   2. 保持Ki & Kd不变，调整Kp的值----以0.6为步长，在[0.6,3]的范围内生成5张系统时域图
% ------------------------------------------------------------
PID_with_noise(Kp_s,Ki_s,Kd_s,t,gp,g,H)
```

原系统——无控制器接入

```
row.m
```

```
function row(Kp_s,Ki_s,Kd_s,t,gp,g,H)

%{
<program>  Copyright (C) <2023>  <Huaqian Chin>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
%}
```

```
num = 10;
r = zeros(size(t));  % 初始化为全零
r(t > 10 & t <= 11) = r(t > 10 & t <= 11) + (max(0, min((t(t > 10 & t <= 11) - 10), 1)) * num);
r(t > 11) = num;
% 创建n序列
noise = zeros(size(t));  % 创建n序列
% 创建Td序列
Td = zeros(size(t));  % 创建Td序列

%%
figure(1);

gc = 1;
fai_r = (gc*gp*g)/(1+H*gc*gp*g);
fai_Td = (g)/(1+H*gc*gp*g);
fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
output = output_r + output_Td + output_n; % 得到总输出

% 计算调节时间
final_value = 10;  % 假设最终稳定值为10
settling_index = find(output >= 0.98 * final_value & output <= final_value, 1, 'first');
settling_time = t(settling_index) - 10;  % 调节时间从t=10开始计算

plot(t, output);  % 画出时域图像

% 添加输入开始的标记
hold on;
line([10, 10], ylim, 'Color', 'green', 'LineStyle', '--');
text(10, min(output), '', 'VerticalAlignment','top', 'HorizontalAlignment','left');

% 添加调节时间标记
line([settling_time + 10, settling_time + 10], ylim, 'Color', 'red', 'LineStyle', '--');
text(settling_time + 10, min(output), sprintf('t_s = %.2f', settling_time), 'VerticalAlignment','top', 'HorizontalAlignment','right');
hold off;

title('Response for row Controller')
xlabel('Time (s)')
ylabel('Response')


%%
figure(1);

num = -50;  % 修改num为-50
Td = zeros(size(t));  % 初始化为全零
Td(t > 50 & t <= 51) = Td(t > 50 & t <= 51) + (max(0, min((t(t > 50 & t <= 51) - 50), 1)) * num);
```

```
Td(t > 51) = num;

gc = 1;
fai_r = (gc*gp*g)/(1+H*gc*gp*g);
fai_Td = (g)/(1+H*gc*gp*g);
fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
output = output_r + output_Td + output_n; % 得到总输出

% 计算超调量
overshoot = max(0, output - 10);  % 假设最终稳定值为10

% 计算超调量变化比例
overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10

% 找到最大超调量变化比例及其对应的时间
[max_overshoot_ratio, idx] = max(overshoot_ratio);
max_overshoot_ratio_time = t(idx);

plot(t, output);  % 画出时域图像
hold on;
plot(t, overshoot_ratio, 'r--');  % 画出超调量变化比例曲线
text(max_overshoot_ratio_time, max_overshoot_ratio, sprintf('Max overshoot
ratio: %.2f%%', max_overshoot_ratio*100), 'VerticalAlignment','bottom',
'HorizontalAlignment','right');
hold off;

title('Response and Overshoot Ratio for row Controller---Td_input')
xlabel('Time (s)')
ylabel('Response / Overshoot Ratio')
legend('Response', 'Overshoot Ratio')

end
```

阶跃输入信号——PD控制器

```
PD.m
```

```
function PD(Kp_s,Ki_s,Kd_s,t,gp,g,H)

%{
<program>  Copyright (C) <2023>  <Huaqian Chin>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
%}

% N(s) = 0;  Td(s) = 0;
num = 10;
r = zeros(size(t));  % 初始化为全零
```

```
r(t > 10 & t <= 11) = r(t > 10 & t <= 11) + (max(0, min((t(t > 10 & t <= 11) -
10), 1)) * num);
r(t > 11) = num;
% 创建n序列
noise = zeros(size(t));  % 创建n序列
% 创建Td序列
Td = zeros(size(t));  % 创建Td序列

%% Kp不动Kd动
figure('Position', [10 10 900 900])
Kp = Kp_s;
for i = 1:5
    Kd = i*1;

    % PD Block
    n = [Kd, Kp, 0];  %分子多项式系数向量
    d = [1 0];  %分母多项式向量
    gc_pd = tf(n, d);% 构建传递函数对象
    gc = gc_pd;

    fai_r = (gc*gp*g)/(1+H*gc*gp*g);
    fai_Td = (g)/(1+H*gc*gp*g);
    fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
    output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
    output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
    output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
    output = output_r + output_Td + output_n; % 得到总输出

    % 计算超调量
    overshoot = max(0, output - 10);  % 假设最终稳定值为10
    % 计算超调量变化比例
    overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10
    % 找到最大超调量变化比例及其对应的时间
    [max_overshoot_ratio, idx] = max(overshoot_ratio);
    max_overshoot_ratio_time = t(idx);

    % 计算调节时间
    final_value = 10;  % 假设最终稳定值为10
    settling_index = find(output >= 0.98 * final_value & output <= final_value,
1, 'first');
    settling_time = t(settling_index) - 10;

    subplot(5,1,i)
    plot(t, output);  % 画出时域图像
    hold on;
    % 添加超调量标记
    text(max_overshoot_ratio_time, output(idx)-0.5, sprintf('Overshoot: %.2f%%',
max_overshoot_ratio*100), 'VerticalAlignment','top',
'HorizontalAlignment','left');
    % 添加调节时间标记
    line([settling_time + 10, settling_time + 10], ylim, 'Color', 'red',
'LineStyle', '--');
    text(settling_time + 10, output(settling_index)-0.5, sprintf('t_s = %.2f',
settling_time), 'VerticalAlignment','top', 'HorizontalAlignment','right');
    hold off;
```

```
    title(['Response for PD Controller with Kp=',num2str(Kp) ,'  Kd=',
num2str(Kd)])
    xlabel('Time (s)')
    ylabel('Response')
end

%% Kd不动Kp动
figure('Position', [10 10 900 900])
Kd = Kd_s;
for i = 1:5
    Kp = i*2;

    % PD Block
    n = [Kd, Kp, 0];  %分子多项式系数向量
    d = [1 0];  %分母多项式向量
    gc_pd = tf(n, d);% 构建传递函数对象
    gc = gc_pd;

    fai_r = (gc*gp*g)/(1+H*gc*gp*g);
    fai_Td = (g)/(1+H*gc*gp*g);
    fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
    output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
    output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
    output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
    output = output_r + output_Td + output_n; % 得到总输出

    % 计算超调量
    overshoot = max(0, output - 10);  % 假设最终稳定值为10
    % 计算超调量变化比例
    overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10
    % 找到最大超调量变化比例及其对应的时间
    [max_overshoot_ratio, idx] = max(overshoot_ratio);
    max_overshoot_ratio_time = t(idx);

    % 计算调节时间
    final_value = 10;  % 假设最终稳定值为10
    settling_index = find(output >= 0.98 * final_value & output <= final_value,
1, 'first');
    settling_time = t(settling_index) - 10;

    subplot(5,1,i)
    plot(t, output);  % 画出时域图像
    hold on;
    % 添加超调量标记
    text(max_overshoot_ratio_time, output(idx)-0.5, sprintf('Overshoot: %.2f%%',
max_overshoot_ratio*100), 'VerticalAlignment','top',
'HorizontalAlignment','left');
    % 添加调节时间标记
    line([settling_time + 10, settling_time + 10], ylim, 'Color', 'red',
'LineStyle', '--');
    text(settling_time + 10, output(settling_index)-0.5, sprintf('t_s = %.2f',
settling_time), 'VerticalAlignment','top', 'HorizontalAlignment','right');
    hold off;
```

```
    title(['Response for PD Controller with Kp=',num2str(Kp) ,'  Kd=',
num2str(Kd)])
    xlabel('Time (s)')
    ylabel('Response')
end

end
```

阶跃输入信号——PI控制器

```
PI.m
```

```
function PI(Kp_s,Ki_s,Kd_s,t,gp,g,H)

%{
<program>  Copyright (C) <2023>  <Huaqian Chin>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
%}

num = 10;
r = zeros(size(t));  % 初始化为全零
r(t > 10 & t <= 11) = r(t > 10 & t <= 11) + (max(0, min((t(t > 10 & t <= 11) -
10), 1)) * num);
r(t > 11) = num;
% 创建n序列
noise = zeros(size(t));  % 创建n序列
% 创建Td序列
Td = zeros(size(t));  % 创建Td序列

%% Kp不动Ki动
figure('Position', [10 10 900 900])
Kp = Kp_s;
for i = 1:5
    Ki = i*0.15;

    % PI function
    n = [0, Kp, Ki];  %分子多项式系数向量
    d = [1 0];  %分母多项式向量
    gc_pi = tf(n, d);% 构建传递函数对象
    gc = gc_pi;

    fai_r = (gc*gp*g)/(1+H*gc*gp*g);
    fai_Td = (g)/(1+H*gc*gp*g);
    fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
    output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
    output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
    output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
    output = output_r + output_Td + output_n; % 得到总输出

    % 计算超调量
```

```
    overshoot = max(0, output - 10);  % 假设最终稳定值为10
    % 计算超调量变化比例
    overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10
    % 找到最大超调量变化比例及其对应的时间
    [max_overshoot_ratio, idx] = max(overshoot_ratio);
    max_overshoot_ratio_time = t(idx);

    % 计算调节时间
    final_value = 10;  % 假设最终稳定值为10
    settling_index = find(output >= 0.98 * final_value & output <= final_value,
1, 'first');
    settling_time = t(settling_index) - 10;

    subplot(5,1,i)
    plot(t, output);  % 画出时域图像
    hold on;
    % 添加超调量标记
    text(max_overshoot_ratio_time, output(idx)-0.5, sprintf('Overshoot: %.2f%%',
max_overshoot_ratio*100), 'VerticalAlignment','top',
'HorizontalAlignment','left');
    % 添加调节时间标记
    line([settling_time + 10, settling_time + 10], ylim, 'Color', 'red',
'LineStyle', '--');
    text(settling_time + 10, output(settling_index)-0.5, sprintf('t_s = %.2f',
settling_time), 'VerticalAlignment','top', 'HorizontalAlignment','right');
    hold off;

    title(['Response for PI Controller with Kp=',num2str(Kp) ,'  Ki=',
num2str(Ki)])
    xlabel('Time (s)')
    ylabel('Response')
end

%% Ki不动Kp动
figure('Position', [10 10 900 900])
Ki = Ki_s;
for i = 1:5
    Kp = i*0.8;

    % PI function
    n = [0, Kp, Ki];  %分子多项式系数向量
    d = [1 0];  %分母多项式向量
    gc_pi = tf(n, d);% 构建传递函数对象
    gc = gc_pi;

    fai_r = (gc*gp*g)/(1+H*gc*gp*g);
    fai_Td = (g)/(1+H*gc*gp*g);
    fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
    output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
    output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
    output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
    output = output_r + output_Td + output_n; % 得到总输出

    % 计算超调量
    overshoot = max(0, output - 10);  % 假设最终稳定值为10
```

```
    % 计算超调量变化比例
    overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10
    % 找到最大超调量变化比例及其对应的时间
    [max_overshoot_ratio, idx] = max(overshoot_ratio);
    max_overshoot_ratio_time = t(idx);

    % 计算调节时间
    final_value = 10;  % 假设最终稳定值为10
    settling_index = find(output >= 0.98 * final_value & output <= final_value,
1, 'first');
    settling_time = t(settling_index) - 10;

    subplot(5,1,i)
    plot(t, output);  % 画出时域图像
    hold on;
    % 添加超调量标记
    text(max_overshoot_ratio_time, output(idx)-0.5, sprintf('Overshoot: %.2f%%',
max_overshoot_ratio*100), 'VerticalAlignment','top',
'HorizontalAlignment','left');
    % 添加调节时间标记
    line([settling_time + 10, settling_time + 10], ylim, 'Color', 'red',
'LineStyle', '--');
    text(settling_time + 10, output(settling_index)-0.5, sprintf('t_s = %.2f',
settling_time), 'VerticalAlignment','top', 'HorizontalAlignment','right');
    hold off;

    title(['Response for PI Controller with Kp=',num2str(Kp) ,'  Ki=',
num2str(Ki)])
    xlabel('Time (s)')
    ylabel('Response')
end

end
```

阶跃输入信号——PID控制器

```
PID.m
```

```
function PID(Kp_s,Ki_s,Kd_s,t,gp,g,H)

%{
<program>  Copyright (C) <2023>  <Huaqian Chin>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
%}

num = 10;
r = zeros(size(t));  % 初始化为全零
r(t > 10 & t <= 11) = r(t > 10 & t <= 11) + (max(0, min((t(t > 10 & t <= 11) -
10), 1)) * num);
r(t > 11) = num;
```

```
% 创建n序列
noise = zeros(size(t));  % 创建n序列
% 创建Td序列
Td = zeros(size(t));  % 创建Td序列

%% Kp&Ki不动Kd动
figure('Position', [10 10 900 900])
Kp = Kp_s;
Ki = Ki_s;
for i = 1:5
    Kd = i*0.8;

    % PID function
    n = [Kd, Kp, Ki];  %分子多项式系数向量
    d = [1 0];  %分母多项式向量
    gc_pid = tf(n, d); % 构建传递函数对象
    gc = gc_pid;

    fai_r = (gc*gp*g)/(1+H*gc*gp*g);
    fai_Td = (g)/(1+H*gc*gp*g);
    fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
    output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
    output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
    output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
    output = output_r + output_Td + output_n; % 得到总输出

    % 计算超调量
    overshoot = max(0, output - 10);  % 假设最终稳定值为10
    % 计算超调量变化比例
    overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10
    % 找到最大超调量变化比例及其对应的时间
    [max_overshoot_ratio, idx] = max(overshoot_ratio);
    max_overshoot_ratio_time = t(idx);

    % 计算调节时间
    final_value = 10;  % 假设最终稳定值为10
    settling_index = find(output >= 0.98 * final_value & output <= final_value,
1, 'first');
    settling_time = t(settling_index) - 10;

    subplot(5,1,i)
    plot(t, output);  % 画出时域图像
    hold on;
    % 添加超调量标记
    text(max_overshoot_ratio_time, output(idx)-0.5, sprintf('Overshoot: %.2f%%',
max_overshoot_ratio*100), 'VerticalAlignment','top',
'HorizontalAlignment','left');
    % 添加调节时间标记
    line([settling_time + 10, settling_time + 10], ylim, 'Color', 'red',
'LineStyle', '--');
    text(settling_time + 10, output(settling_index)-0.5, sprintf('t_s = %.2f',
settling_time), 'VerticalAlignment','top', 'HorizontalAlignment','right');
    hold off;
```

```
    title(['Response for PID Controller with Kp=',num2str(Kp) ,'  Ki=',
num2str(Ki),'  Kd=', num2str(Kd)])
    xlabel('Time (s)')
    ylabel('Response')
end

%% Kp&Kd不动Ki动
figure('Position', [10 10 900 900])
Kp = Kp_s;
Kd = Kd_s;
for i = 1:5
    Ki = i*0.15;

    % PID function
    n = [Kd, Kp, Ki];  %分子多项式系数向量
    d = [1 0];  %分母多项式向量
    gc_pid = tf(n, d); % 构建传递函数对象
    gc = gc_pid;

    fai_r = (gc*gp*g)/(1+H*gc*gp*g);
    fai_Td = (g)/(1+H*gc*gp*g);
    fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
    output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
    output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
    output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
    output = output_r + output_Td + output_n; % 得到总输出

    % 计算超调量
    overshoot = max(0, output - 10);  % 假设最终稳定值为10
    % 计算超调量变化比例
    overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10
    % 找到最大超调量变化比例及其对应的时间
    [max_overshoot_ratio, idx] = max(overshoot_ratio);
    max_overshoot_ratio_time = t(idx);

    % 计算调节时间
    final_value = 10;  % 假设最终稳定值为10
    settling_index = find(output >= 0.98 * final_value & output <= final_value,
1, 'first');
    settling_time = t(settling_index) - 10;

    subplot(5,1,i)
    plot(t, output);  % 画出时域图像
    hold on;
    % 添加超调量标记
    text(max_overshoot_ratio_time, output(idx)-0.5, sprintf('Overshoot: %.2f%%',
max_overshoot_ratio*100), 'VerticalAlignment','top',
'HorizontalAlignment','left');
    % 添加调节时间标记
    line([settling_time + 10, settling_time + 10], ylim, 'Color', 'red',
'LineStyle', '--');
    text(settling_time + 10, output(settling_index)-0.5, sprintf('t_s = %.2f',
settling_time), 'VerticalAlignment','top', 'HorizontalAlignment','right');
    hold off;
```

```
    title(['Response for PID Controller with Kp=',num2str(Kp) ,'  Ki=', 
num2str(Ki),'  Kd=', num2str(Kd)])
    xlabel('Time (s)')
    ylabel('Response')
end

%% Ki&Kd不动Kp动
figure('Position', [10 10 900 900])
Ki = Ki_s;
Kd = Kd_s;
for i = 1:5
    Kp = i*0.6;

    % PID function
    n = [Kd, Kp, Ki];  %分子多项式系数向量
    d = [1 0];  %分母多项式向量
    gc_pid = tf(n, d); % 构建传递函数对象
    gc = gc_pid;

    fai_r = (gc*gp*g)/(1+H*gc*gp*g);
    fai_Td = (g)/(1+H*gc*gp*g);
    fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
    output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
    output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
    output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
    output = output_r + output_Td + output_n; % 得到总输出

    % 计算超调量
    overshoot = max(0, output - 10);  % 假设最终稳定值为10
    % 计算超调量变化比例
    overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10
    % 找到最大超调量变化比例及其对应的时间
    [max_overshoot_ratio, idx] = max(overshoot_ratio);
    max_overshoot_ratio_time = t(idx);

    % 计算调节时间
    final_value = 10;  % 假设最终稳定值为10
    settling_index = find(output >= 0.98 * final_value & output <= final_value, 
1, 'first');
    settling_time = t(settling_index) - 10;

    subplot(5,1,i)
    plot(t, output);  % 画出时域图像
    hold on;
    % 添加超调量标记
    text(max_overshoot_ratio_time, output(idx)-0.5, sprintf('Overshoot: %.2f%%', 
max_overshoot_ratio*100), 'VerticalAlignment','top', 
'HorizontalAlignment','left');
    % 添加调节时间标记
    line([settling_time + 10, settling_time + 10], ylim, 'Color', 'red', 
'LineStyle', '--');
    text(settling_time + 10, output(settling_index)-0.5, sprintf('t_s = %.2f', 
settling_time), 'VerticalAlignment','top', 'HorizontalAlignment','right');
    hold off;
```

```
    title(['Response for PID Controller with Kp=',num2str(Kp) ,'  Ki=', num2str(Ki),'  Kd=', num2str(Kd)])
    xlabel('Time (s)')
    ylabel('Response')
end

end
```

阶跃扰动信号——PID控制器

```
PID_with_noise.m
```

```
function PID_with_noise(Kp_s,Ki_s,Kd_s,t,gp,g,H)

%{
<program>  Copyright (C) <2023>  <Huaqian Chin>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
%}

% N(s) = 0;  Td(s) = 50/s;
r = 10 * ones(size(t));  % 初始化为全零
% 创建n序列
noise = zeros(size(t));
% 创建Td序列
num = -50;  % 修改num为-50
Td = zeros(size(t));  % 初始化为全零
Td(t > 50 & t <= 51) = Td(t > 50 & t <= 51) + (max(0, min((t(t > 50 & t <= 51) - 50), 1)) * num);
Td(t > 51) = num;

%% Kp&Ki不动Kd动
figure('Position', [10 10 900 900])
Kp = Kp_s;
Ki = Ki_s;
for i = 1:5
    Kd = i*0.8;

    % PID function
    n = [Kd, Kp, Ki];  %分子多项式系数向量
    d = [1 0];  %分母多项式向量
    gc_pid = tf(n, d); % 构建传递函数对象
    gc = gc_pid;

    fai_r = (gc*gp*g)/(1+H*gc*gp*g);
    fai_Td = (g)/(1+H*gc*gp*g);
    fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
    output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
    output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
    output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
    output = output_r + output_Td + output_n; % 得到总输出
```

```
    % 计算超调量
    overshoot = max(0, output - 10);  % 假设最终稳定值为10
    % 计算超调量变化比例
    overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10
    % 找到最大超调量变化比例及其对应的时间
    [max_overshoot_ratio, idx] = max(overshoot_ratio);
    max_overshoot_ratio_time = t(idx);

    subplot(5,1,i)
%     plot(t, output);  % 画出时域图像
%     hold on;
%     % 添加超调量标记
%     text(max_overshoot_ratio_time, output(idx)-0.5, sprintf('Overshoot:
%.2f%%', max_overshoot_ratio*100), 'VerticalAlignment','top',
'HorizontalAlignment','left');
    plot(t, output);  % 画出时域图像
    hold on;
    plot(t, overshoot_ratio, 'r--');  % 画出超调量变化比例曲线
    text(max_overshoot_ratio_time, max_overshoot_ratio, sprintf('Max overshoot
ratio: %.2f%%', max_overshoot_ratio*100), 'VerticalAlignment','bottom',
'HorizontalAlignment','right');
    hold off;

    title(['Response for PID Controller with Kp=',num2str(Kp) ,'  Ki=',
num2str(Ki),'  Kd=', num2str(Kd)])
    xlabel('Time (s)')
    ylabel('Response')
end

%% Kp&Kd不动Ki动
figure('Position', [10 10 900 900])
Kp = Kp_s;
Kd = Kd_s;
for i = 1:5
    Ki = i*0.15;

    % PID function
    n = [Kd, Kp, Ki];  %分子多项式系数向量
    d = [1 0];  %分母多项式向量
    gc_pid = tf(n, d); % 构建传递函数对象
    gc = gc_pid;

    fai_r = (gc*gp*g)/(1+H*gc*gp*g);
    fai_Td = (g)/(1+H*gc*gp*g);
    fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
    output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
    output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
    output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
    output = output_r + output_Td + output_n; % 得到总输出

    % 计算超调量
    overshoot = max(0, output - 10);  % 假设最终稳定值为10
    % 计算超调量变化比例
    overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10
```

```
    % 找到最大超调量变化比例及其对应的时间
    [max_overshoot_ratio, idx] = max(overshoot_ratio);
    max_overshoot_ratio_time = t(idx);

    subplot(5,1,i)
%     plot(t, output);  % 画出时域图像
%     hold on;
%     % 添加超调量标记
%     text(max_overshoot_ratio_time, output(idx)-0.5, sprintf('Overshoot:
%.2f%%', max_overshoot_ratio*100), 'VerticalAlignment','top',
'HorizontalAlignment','left');
    plot(t, output);  % 画出时域图像
    hold on;
    plot(t, overshoot_ratio, 'r--');  % 画出超调量变化比例曲线
    text(max_overshoot_ratio_time, max_overshoot_ratio, sprintf('Max overshoot
ratio: %.2f%%', max_overshoot_ratio*100), 'VerticalAlignment','bottom',
'HorizontalAlignment','right');
    hold off;

    title(['Response for PID Controller with Kp=',num2str(Kp) ,'  Ki=',
num2str(Ki),'  Kd=', num2str(Kd)])
    xlabel('Time (s)')
    ylabel('Response')
end

%% Ki&Kd不动Kp动
figure('Position', [10 10 900 900])
Ki = Ki_s;
Kd = Kd_s;
for i = 1:5
    Kp = i*0.6;

    % PID function
    n = [Kd, Kp, Ki];  %分子多项式系数向量
    d = [1 0];  %分母多项式向量
    gc_pid = tf(n, d); % 构建传递函数对象
    gc = gc_pid;

    fai_r = (gc*gp*g)/(1+H*gc*gp*g);
    fai_Td = (g)/(1+H*gc*gp*g);
    fai_n = (H*gc*gp*g)/(1+H*gc*gp*g);
    output_r = lsim(fai_r, r, t);  % 对应fai_r的输出
    output_Td = lsim(fai_Td, Td, t);  % 对应fai_Td的输出
    output_n = lsim(fai_n, noise, t);   % 对应fai_n的输出
    output = output_r + output_Td + output_n; % 得到总输出

    % 计算超调量
    overshoot = max(0, output - 10);  % 假设最终稳定值为10
    % 计算超调量变化比例
    overshoot_ratio = overshoot ./ 10;  % 假设最终稳定值为10
    % 找到最大超调量变化比例及其对应的时间
    [max_overshoot_ratio, idx] = max(overshoot_ratio);
    max_overshoot_ratio_time = t(idx);

    subplot(5,1,i)
```

```
%     plot(t, output);  % 画出时域图像
%     hold on;
%     % 添加超调量标记
%     text(max_overshoot_ratio_time, output(idx)-0.5, sprintf('Overshoot: %.2f%%', max_overshoot_ratio*100), 'VerticalAlignment','top', 'HorizontalAlignment','left');
    plot(t, output);  % 画出时域图像
    hold on;
    plot(t, overshoot_ratio, 'r--');  % 画出超调量变化比例曲线
    text(max_overshoot_ratio_time, max_overshoot_ratio, sprintf('Max overshoot ratio: %.2f%%', max_overshoot_ratio*100), 'VerticalAlignment','bottom', 'HorizontalAlignment','right');
    hold off;

    title(['Response for PID Controller with Kp=',num2str(Kp) ,'  Ki=', num2str(Ki),'  Kd=', num2str(Kd)])
    xlabel('Time (s)')
    ylabel('Response')
end

end
```