

声明

本实验报告由 Costannt 独立撰写，旨在分享实验思路与结果分析。仅供学习参考，严禁全文或部分直接抄袭、复制或提交为个人作业。

学术诚信是每位学生的基本责任，请尊重他人劳动成果，也保护自己的信誉。任何因不当使用本材料所引发的后果，均由使用者自行承担。

实验评分标准

- | | |
|-----------------------|------|
| (1) 被控对象辨识 (系统频率特性测试) | 10 分 |
| (2) 位置单闭环 PID 控制器设计 | 10 分 |
| (3) PID 控制器加干扰观测器设计 | 10 分 |
| (4) 频域控制器设计 | 10 分 |
| (5) 双闭环 PID 控制器设计 | 10 分 |

实验一 被控对象辨识（系统频率特性测试）

一、 系统模型辨识程序（包括代码说明），（不用粘贴保存的数据）

（8 分）

```
clear; clc; close all;

% 设置中文字体
set(0, 'DefaultAxesFontName', '宋体');
set(0, 'DefaultTextFontName', '宋体');
set(0, 'DefaultUicontrolFontName', '宋体');
set(0, 'DefaultUitableFontName', '宋体');
set(0, 'DefaultLegendFontName', '宋体');

% 读取数据
data1 = readtable('1129-1.csv', 'VariableNamingRule', 'preserve');
data2 = readtable('1129-2.csv', 'VariableNamingRule', 'preserve');

% 创建表格
if size(data1, 2) == 6
    data1 = array2table(data1, 'VariableNames', {'input_angle',
'input_voltage', 'current_freq', 'input_speed', 'feedback_speed',
'feedback_angle'});
end
if size(data2, 2) == 6
    data2 = array2table(data2, 'VariableNames', {'input_angle',
'input_voltage', 'current_freq', 'input_speed', 'feedback_speed',
'feedback_angle'});
end

% 合并数据
data = [data1; data2];

if istable(data)
    % 获取列名
    colnames = data.Properties.VariableNames;
    fprintf('检测到的列名:\n');
    for i = 1:length(colnames)
        fprintf('列 %d: %s\n', i, colnames{i});
    end
end
```

```

    % 根据列位置提取数据
    input_angle = data(:,1);
    input_voltage = data(:,2);
    current_freq = data(:,3);
    input_speed = data(:,4);
    feedback_speed = data(:,5);
    feedback_angle = data(:,6);
end

% 处理无效值
idx_input = find(isnan(input_voltage));
idx_feedback = find(isnan(feedback_angle));
input_voltage(idx_input) = 0;
feedback_angle(idx_feedback) = 0;

% 采样参数
fs = 500; % 采样频率 500Hz
Ts = 1/fs; % 采样周期

% 获取所有测试频率
freqs = unique(current_freq);
freqs = freqs(freqs > 0); % 去除 0 频率

% 存储结果
magnitude_ratio = zeros(length(freqs), 1);
phase_diff = zeros(length(freqs), 1);
input_magnitude = zeros(length(freqs), 1);
output_magnitude = zeros(length(freqs), 1);
norm_input_magnitude = zeros(length(freqs), 1);
norm_output_magnitude = zeros(length(freqs), 1);

% 创建图形文件夹
if ~exist('frequency_plots_angle', 'dir')
    mkdir('frequency_plots_angle');
end

% 对每个频率进行分析
for i = 1:length(freqs)
    freq = freqs(i);
    fprintf('处理频率: %.1f Hz\n', freq);

    % 找到当前频率的数据索引

```

```

freq_indices = find(abs(current_freq - freq) < 0.001);

if isempty(freq_indices)
    fprintf('未找到频率 %.1f Hz 的数据\n', freq);
    continue;
end

% 提取当前频率的输入输出数据
input_data = input_voltage(freq_indices);
output_data = feedback_angle(freq_indices);
time = (0:length(input_data)-1)' * Ts;

% 计算周期和样本数
T = 1/freq; % 周期
samples_per_period = round(fs / freq); % 每个周期的样本数

% 判断是否需要扩展为 5 个周期
if length(input_data) < 5 * samples_per_period
    % 不够 5 个周期的情况 - 去除前后各 1 个周期的渐变
    trim_start = samples_per_period;
    trim_end = length(input_data) - samples_per_period;
else
    % 足够 5 个周期的情况 - 去除前后 0.75 秒渐变和最后 1 秒零输入
    trim_start = round(0.75 * fs); % 前 0.75 秒
    trim_end = length(input_data) - round(0.75 * fs) - round(1 * fs); %
    后 0.75 秒渐变 + 1 秒零输入
end

% 确保裁剪范围有效
if trim_end <= trim_start || trim_start < 1 || trim_end >
length(input_data)
    fprintf('频率 %.1f Hz: 裁剪范围无效, 使用全部数据\n', freq);
    trim_start = 1;
    trim_end = length(input_data);
end

% 裁剪数据
input_trimmed = input_data(trim_start:trim_end);
output_trimmed = output_data(trim_start:trim_end);
time_trimmed = time(trim_start:trim_end);

% 绘制原始和裁剪后的数据
figure('Position', [100, 100, 1200, 800]);

```

```

% 原始数据
subplot(2,2,1);
plot(time, input_data, 'b', 'LineWidth', 1.5);
hold on;
plot(time, output_data, 'r', 'LineWidth', 1.5);
title(sprintf('频率 %.1f Hz - 原始数据', freq), 'FontSize', 12);
xlabel('时间 (s)');
ylabel('幅值');
legend('输入电压 (V)', '反馈角度 (°)');
grid on;

% 标记裁剪区域
ylims = ylim;
fill([time(trim_start), time(trim_start), time(trim_end),
time(trim_end)], ...
     [ylims(1), ylims(2), ylims(2), ylims(1)], 'k', 'FaceAlpha',
0.1, 'EdgeColor', 'none');
legend('输入电压 (V)', '反馈角度 (°)', '裁剪区域');

% 裁剪后的数据
subplot(2,2,2);
plot(time_trimmed, input_trimmed, 'b', 'LineWidth', 1.5);
hold on;
plot(time_trimmed, output_trimmed, 'r', 'LineWidth', 1.5);
title(sprintf('频率 %.1f Hz - 裁剪后数据', freq), 'FontSize', 12);
xlabel('时间 (s)');
ylabel('幅值');
legend('输入电压 (V)', '反馈角度 (°)');
grid on;

% 傅里叶变换
N = length(input_trimmed);
f = (0:N-1) * fs / N;

% 输入信号的 FFT (幅值归一化)
input_fft = fft(input_trimmed);
input_mag = abs(input_fft) / N * 2; % 对称频谱, 直流分量除外
input_mag(1) = input_mag(1) / 2;    % 直流分量不需要乘以 2
input_phase = angle(input_fft);

% 输出信号的 FFT (幅值归一化)
output_fft = fft(output_trimmed);

```

```

output_mag = abs(output_fft) / N * 2; % 直流分量除外
output_mag(1) = output_mag(1) / 2; % 直流分量不需要乘以 2
output_phase = angle(output_fft);

% 找到输入信号的主频率分量
[~, input_peak_idx] = max(input_mag(1:floor(N/2)));
input_peak_freq = f(input_peak_idx);

% 在输出信号中找到相同频率的分量
freq_idx = find(abs(f - input_peak_freq) == min(abs(f -
input_peak_freq)), 1);

% 计算幅值比和相位差
input_magnitude(i) = input_mag(freq_idx);
output_magnitude(i) = output_mag(freq_idx);
magnitude_ratio(i) = output_mag(freq_idx) / input_mag(freq_idx);
phase_diff(i) = output_phase(freq_idx) - input_phase(freq_idx);

% 记录归一化幅值
norm_input_magnitude(i) = input_mag(freq_idx);
norm_output_magnitude(i) = output_mag(freq_idx);

% 转换为度
phase_diff_deg = rad2deg(phase_diff(i));

% 频谱图
subplot(2,2,3);
plot(f(1:floor(N/2)), input_mag(1:floor(N/2)), 'b-', 'LineWidth',
1.5);
hold on;
plot(f(1:floor(N/2)), output_mag(1:floor(N/2)), 'r-', 'LineWidth',
1.5);
xline(input_peak_freq, 'k--', 'LineWidth', 2, 'Label', sprintf('
主频: %.2f Hz', input_peak_freq));
title('频谱分析 (幅值归一化)');
xlabel('频率 (Hz)');
ylabel('归一化幅值');
legend('输入频谱', '输出频谱', '主频率');
grid on;
xlim([0, min(50, max(freqs)*2)]); % 限制 x 轴范围

% 显示结果
subplot(2,2,4);

```

```

    text(0.1, 0.9, sprintf('频率: %.1f Hz', freq), 'FontSize', 12);
    text(0.1, 0.7, sprintf('输入幅值: %.4f V', norm_input_magnitude(i)),
'FontSize', 12);
    text(0.1, 0.5, sprintf('输出幅值: %.4f °',
norm_output_magnitude(i)), 'FontSize', 12);
    text(0.1, 0.3, sprintf('幅值比: %.4f', magnitude_ratio(i)),
'FontSize', 12);
    text(0.1, 0.1, sprintf('相位差: %.2f°', phase_diff_deg), 'FontSize',
12);
    axis off;

    % 保存图片
    saveas(gcf, sprintf('frequency_plots_angle/freq_%.1f_Hz.png',
freq));
    close gcf;

    fprintf('频率 %.1f Hz: 幅值比 = %.4f, 相位差 = %.2f°\n', freq,
magnitude_ratio(i), phase_diff_deg);
end

% 将弧度相位差转换为度, 并存储为数组
phase_diff_deg = rad2deg(phase_diff); % 所有频率的相位差 (度)

phase_diff_unwrapped_rad = unwrap(phase_diff); % 弧度解卷绕
phase_diff_unwrapped_deg = rad2deg(phase_diff_unwrapped_rad); % 转为
度
offset = 360;
phase_diff_final = phase_diff_unwrapped_deg - offset;

% 绘制 Bode 图
figure('Position', [100, 100, 1000, 600]);

% 幅频特性
subplot(2,1,1);
semilogx(freqs, 20*log10(magnitude_ratio), 'bo-', 'LineWidth', 1.5,
'MarkerSize', 2, 'MarkerFaceColor', 'b');
grid on;
title('Bode 图 - 幅频特性', 'FontSize', 14);
xlabel('频率 (Hz)');
ylabel('增益 (dB)');
xlim([min(freqs), max(freqs)]);

% 相频特性

```



```

subplot(2,1,2);
semilogx(freqs, phase_diff_final, 'ro-', 'LineWidth', 1.5,
'MarkerSize', 4, 'MarkerFaceColor', 'r');
grid on;
title('Bode 图 - 相频特性', 'FontSize', 14);
xlabel('频率 (Hz)');
ylabel('相位 (度)');
xlim([min(freqs), max(freqs)]);

% 添加相位参考线
hold on;
yline(-90, 'k--', 'LineWidth', 1, 'Label', '-90°');
yline(-180, 'k--', 'LineWidth', 1, 'Label', '-180°');
yline(-270, 'k--', 'LineWidth', 1, 'Label', '-270°');
yline(-360, 'k--', 'LineWidth', 1, 'Label', '-360°');

% 显示结果表格
fprintf('\n 频率响应分析结果 (电压到角度) :\n');
fprintf('频率(Hz)\t输入幅值(V)\t输出幅值(°)\t幅值比\t相位差(度)\n');
for i = 1:length(freqs)
    fprintf('%0.1f\t\t%.4f\t\t%.4f\t\t%.4f\t%.2f\n', ...
        freqs(i), norm_input_magnitude(i),
norm_output_magnitude(i), magnitude_ratio(i), phase_diff_final(i));
end

% 保存结果到 CSV 文件
results = table(freqs, norm_input_magnitude, norm_output_magnitude,
magnitude_ratio, phase_diff_final, ...
    'VariableNames', {'Frequency_Hz', 'Input_Magnitude_V',
'Output_Magnitude_deg', 'Magnitude_Ratio', 'Phase_Difference_deg'});
writetable(results, 'frequency_response_results_angle.csv');
fprintf('\n 结果已保存到 frequency_response_results_angle.csv\n');

fprintf('\n 分析完成！\n');

```

二、 辨识出的系统模型传递函数及 BODE 图（软件截图）（2 分）

Process Models

Transfer Function

$$K$$

$$s(1 + T_{p1}s)(1 + T_{p2}s)$$

Poles

2 All real

☐ Zero

☐ Delay

☒ Integrator

Disturbance Model: None Initial condition: Auto Regularization..

Focus: Simulation Covariance: Estimate Options...

☒ Display progress Continue

Name: P2I Estimate Close Help

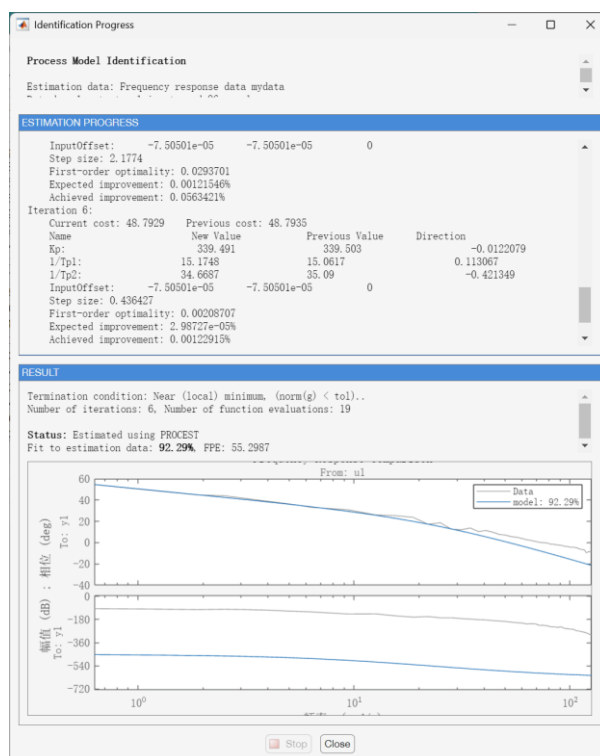
Par	Known	Value	Initial Guess	Bounds
K	<input type="checkbox"/>	339.491	Auto	[-Inf Inf]
Tp1	<input type="checkbox"/>	0.065899	Auto	[0 10000]
Tp2	<input type="checkbox"/>	0.028845	Auto	[0 10000]
Tp3	<input type="checkbox"/>	0	0	[0 Inf]
Tz	<input type="checkbox"/>	0	0	[-Inf Inf]
Td	<input type="checkbox"/>	0	0	[0 Inf]

Initial Guess

☒ Auto-selected

☐ From existing model:

☐ User-defined: Value-->Initial Guess



若使用未简化的电机传递函数，则：

$$G(s) = \frac{339.491}{s(0.065899s + 1)(0.028845s + 1)}$$

自然地考虑使用简化的电机传递函数：

Process Models

Transfer Function

$$\frac{K}{s(1 + T_{p1}s)}$$

Poles

1 All real

☐ Zero

☐ Delay

☒ Integrator

Par	Known	Value	Initial Guess	Bounds
K	<input type="checkbox"/>	341.0215	Auto	[-Inf Inf]
Tp1	<input type="checkbox"/>	0.096351	Auto	[0 10000]
Tp2	<input type="checkbox"/>	0	0	[0 Inf]
Tp3	<input type="checkbox"/>	0	0	[0 Inf]
Tz	<input type="checkbox"/>	0	0	[-Inf Inf]
Td	<input type="checkbox"/>	0	0	[0 Inf]

Initial Guess

☐ Auto-selected

☐ From existing model:

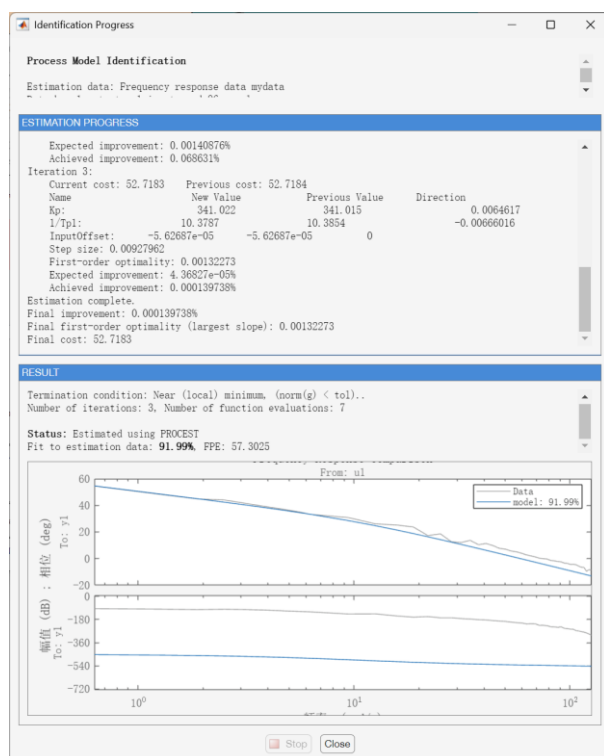
☒ User-defined Value-->Initial Guess

Disturbance Model: None Initial condition: Auto Regularization..

Focus: Simulation Covariance: Estimate Options...

☒ Display progress Continue

Name: P11 Estimate Close Help



得到：

$$G(s) = \frac{341.0215}{s(0.096351s + 1)}$$

分别绘制两个辨识出的传递函数的 Bode 图并与扫频结果比较。

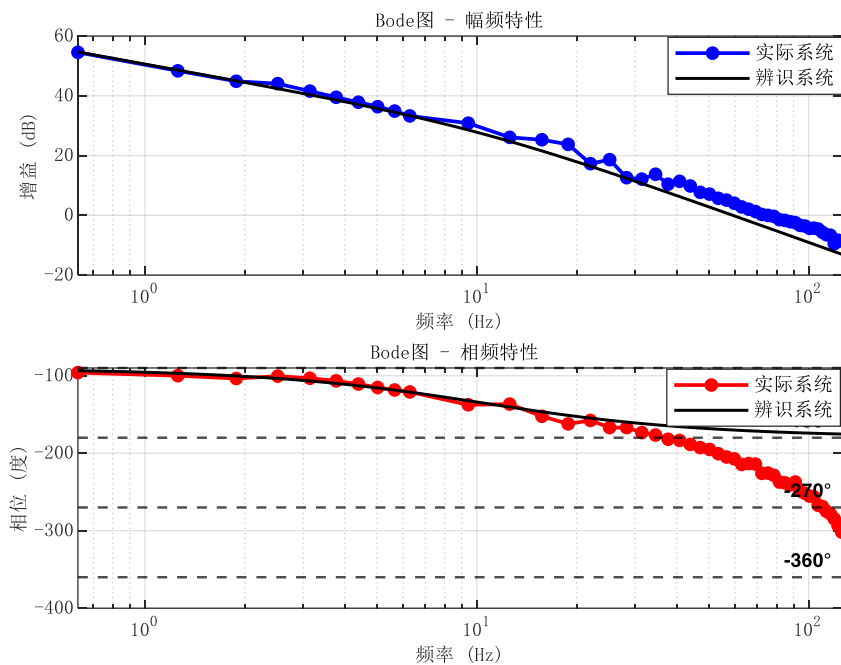


图 1 未简化模型与扫频结果 Bode 图

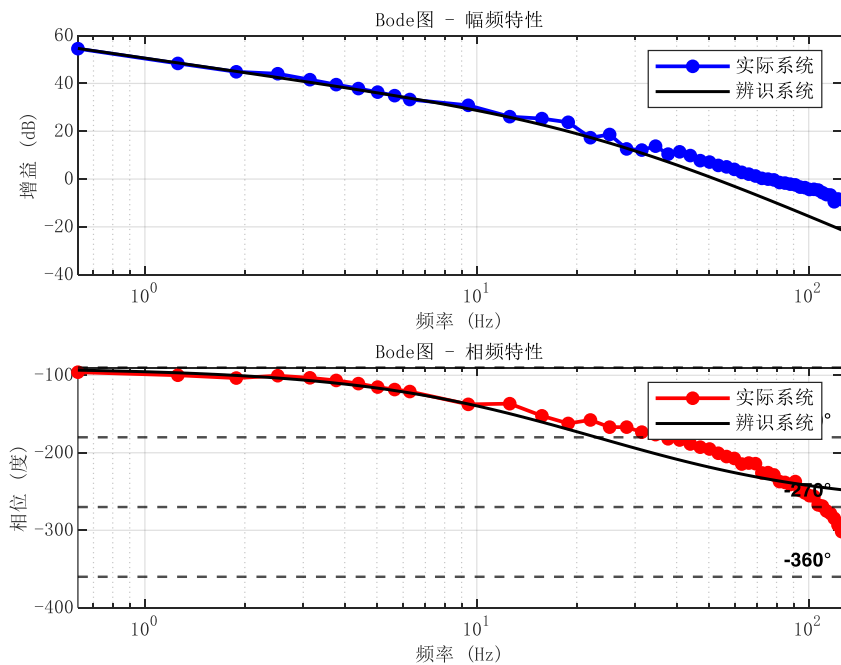


图 2 简化模型与扫频结果 Bode 图

通过 Bode 图和系统辨识报告，可见简化模型对系统的描述效果已很好，因此仿真、控制器设计中可使用简化模型。

实验二 位置单闭环 PID 控制器设计

一、仿真验证

1、仿真程序框图或者代码（包括代码说明）（1 分）

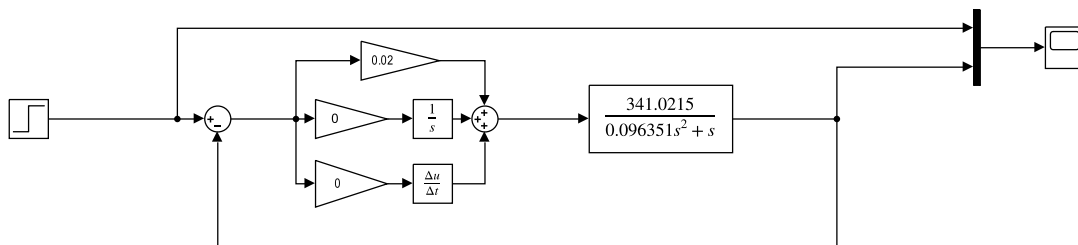


图 3 仿真程序框图

2、PID 控制器参数及仿真响应曲线（1 分）

观察形式，考虑只加入比例控制器 K_p ，构成一个二阶系统。选择 $K_p = 0.02$ ，此时二阶系统 $\omega_n \approx 8.41 \text{ rad/s}$ ， $\zeta \approx 0.62$ 。可以解析地得到 $t_r = 0.34 \text{ s}$ ， $M_p = 8.52\%$ ， $e_{ss} = 0$ 满足设计要求。

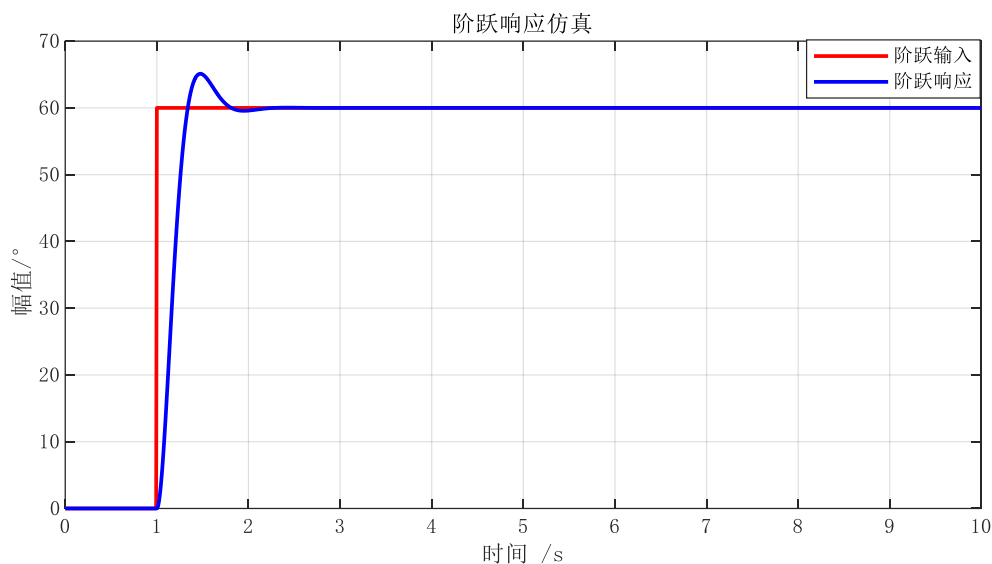


图 4 仿真响应曲线

设计的 PID 参数为：

$$K_p = 0.02, K_i = 0, K_d = 0$$

二、实验验证

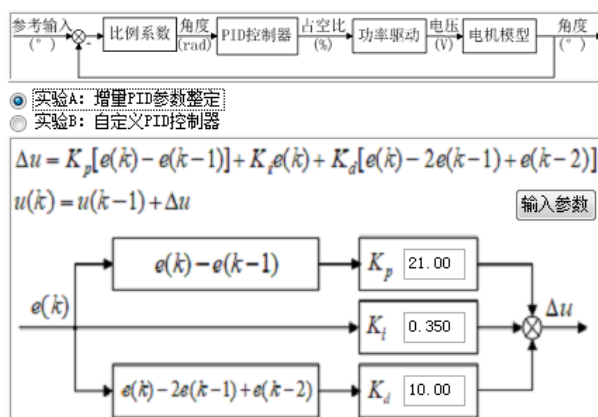
1、描述 PID 控制器设计过程及参数（软件截图）（5 分）

首先尝试应用仿真设计参数。根据实验指导书给出的公式：

$$K_{p\text{Real}} = K_{p\text{Simu}} \times \frac{180}{\pi} \times \frac{100}{24}$$

将 $K_{p\text{Simu}} = 0.02$ 代入计算，得 $K_{p\text{Real}} \approx 4.77$ 。经测试，控制效果很差。本文认为实验效果与仿真不同的原因包括：①仿真模型中未考虑非线性特性。实际系统中存在未建模动态和参数漂移；②实验中有采样率限制，量化噪声和控制时滞等影响；③实验中存在初始条件问题。

通过调整 K_p 来调整响应速度和稳态误差，调整 K_i 以减小甚至消除稳态误差，调整 K_d 以改善超调和系统阻尼。



$$K_p = 21, K_i = 0.350, K_d = 10$$

2、实际调试出的性能指标及 60 度阶跃响应实验曲线（软件截图） （2 分）



软件截图如上所示。实际调试后，系统跟踪输入信号达到稳定，上升时间 0.14s，超调量 8.77%，稳态误差 0.161°，均满足设计要求。

3、使用 MATLAB 根据实验数据复现的 60 度阶跃响应曲线（1 分）

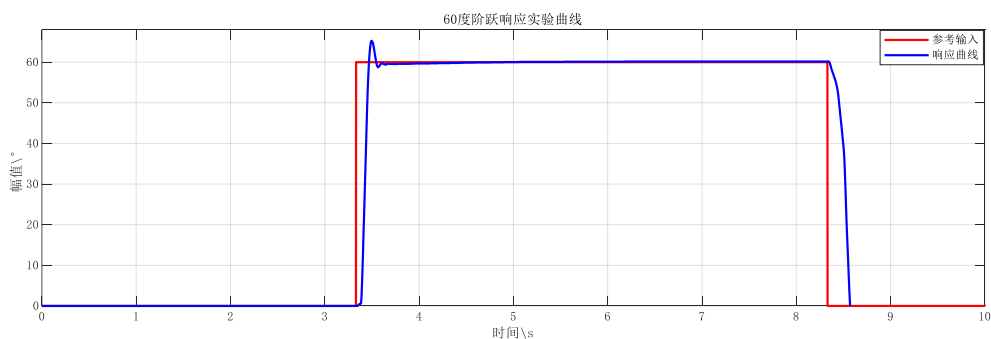


图 5 使用 MATLAB 根据实验数据复现的 60 度阶跃响应曲线

实验三 PID 控制器加干扰观测器设计

一、仿真验证

1、仿真程序框图或者代码（干扰类型可自由选择）（1 分）

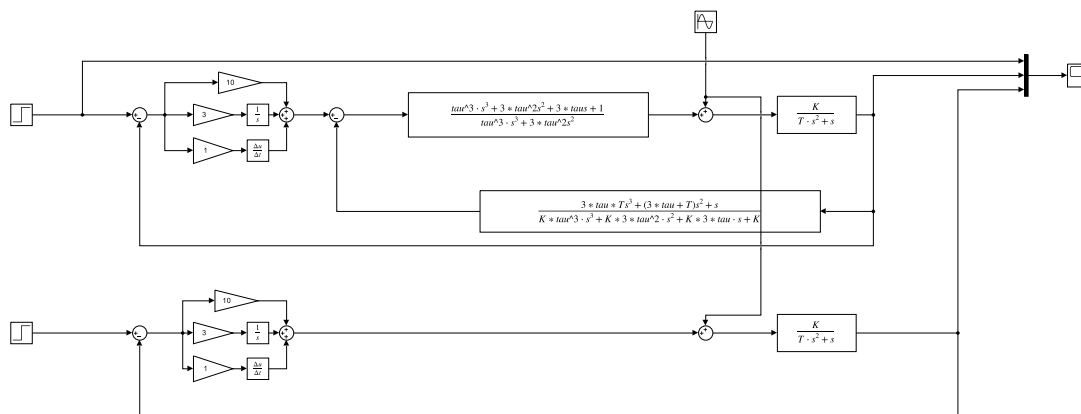


图 6 仿真程序框图

设计了如图的仿真程序以对比干扰观测器的作用，其中加入幅值为 10、频率为 10rad/s 的正弦信号作为干扰。

2、PID 控制器加干扰观测器设计过程、参数和仿真响应曲线（1 分）

通过设计 τ 值来设计低通滤波器的带宽，保证低通滤波器带宽大于被控对象剪切频率。在此基础上，依次调整 K_p, K_i, K_d 以使超调量、上升时间和稳态误差等在允许范围内。调整后得到：

$$K_p = 10, K_i = 3, K_d = 1, \tau = 0.001$$

绘制仿真曲线如图所示：

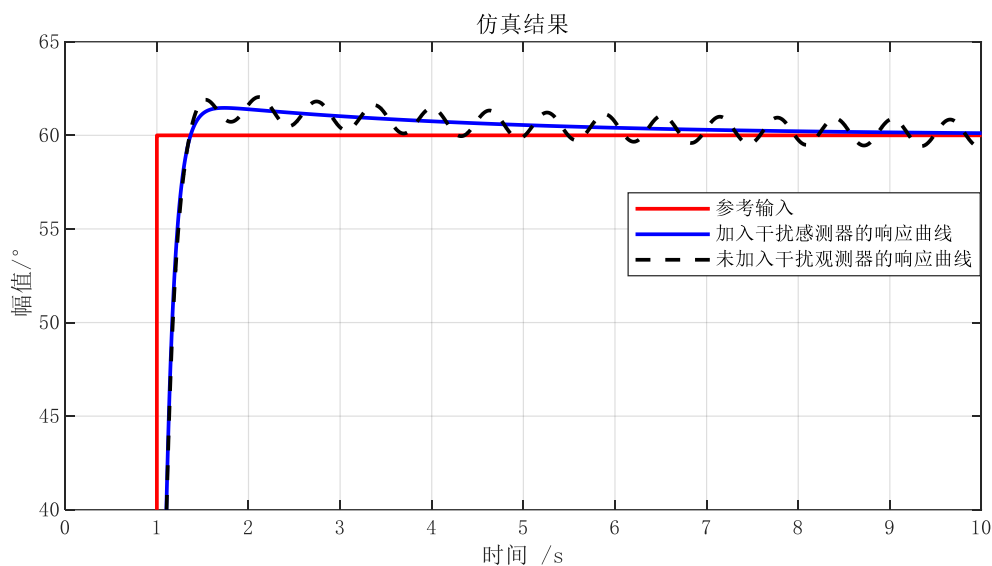
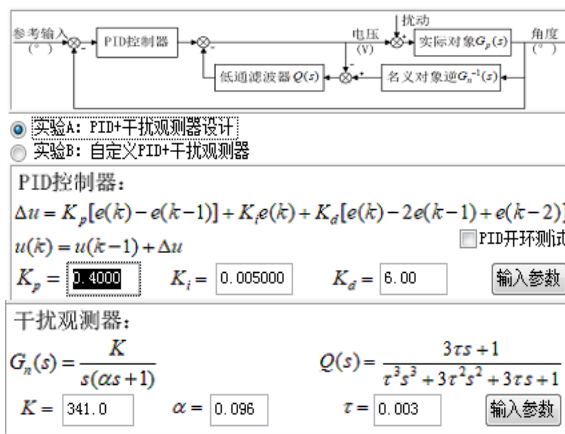


图 7 仿真响应曲线

二、实验验证

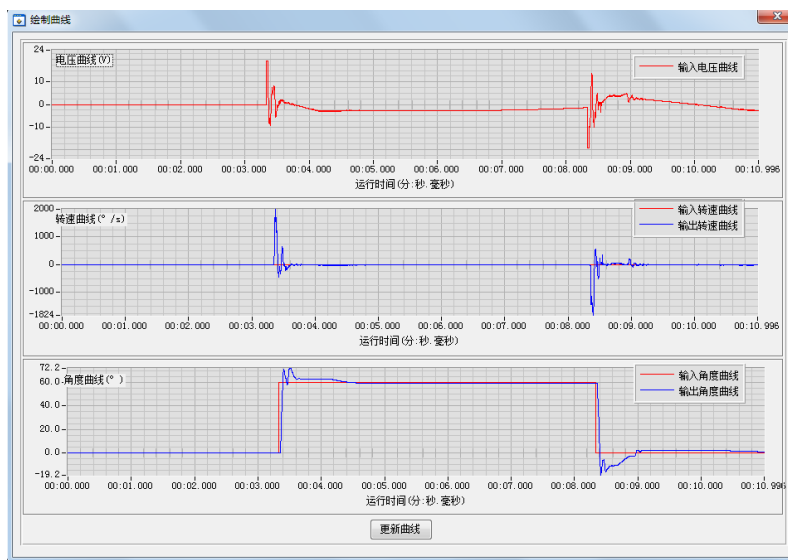
1、描述干扰观测器、PID 控制器调试过程及参数（软件截图）（5 分）

打开软件后，勾选 PID 开环测试，调整 τ 直至电压曲线中不出现剧烈的震荡和饱和现象。最终选择 $\tau = 0.003$ 。取消勾选后，通过调整 K_p 来调整响应速度和稳态误差，调整 K_i 以减小甚至消除稳态误差，调整 K_d 以改善超调和系统阻尼。使超调量、上升时间和稳态误差等在允许范围内。



$$K_p = 0.4, K_i = 0.005, K_d = 6$$

2、实际调试出的性能指标及 60 度阶跃响应实验曲线（软件截图） （2 分）



软件截图如上所示。实际调试后，系统跟踪输入信号达到稳定，上升时间 0.06s，超调量 20.3%，稳态误差 0.45°。

3、使用 MATLAB 根据实验数据复现的 60 度阶跃响应曲线（1 分）

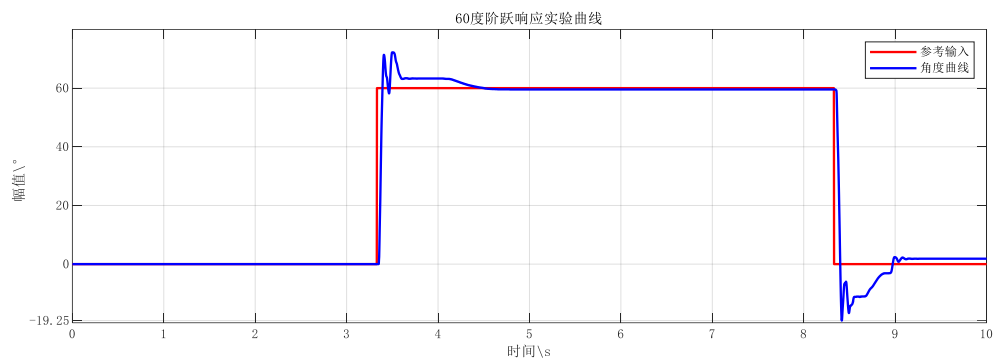


图 8 使用 MATLAB 根据实验数据复现的 60 度阶跃响应曲线

实验四 频域控制器设计

一、仿真验证

1、仿真程序框图或者代码（包括代码说明）（1 分）

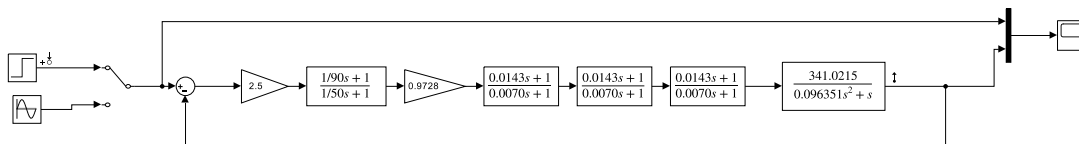


图 9 仿真程序框图

2、频域控制器设计过程、参数和仿真响应曲线（1 分）

根据实验要求，设计实验开环剪切频率 $15\text{Hz} \approx 94.2478 \text{ rad} \cdot \text{s}^{-1}$ ，为了方便，选择开环剪切频率为 $100\text{rad} \cdot \text{s}^{-1}$ 。

经计算，被控对象在期望剪切频率处相角为 -174.075° ，考虑相位裕度 60° ，需要补偿 53° 左右。在此分三次补偿 60° 。经计算，此时应添加增益 0.9728 以确保剪切频率。计算发现此时开环传递函数在 5Hz 处增益为 11.704dB ，根据经验，要使系统在此处满足双十指标，应保证其开环增益达到 18dB 及以上，设计滞后环节并进行增益补偿后运行仿真实验。使用的频域控制器：

超前校正环节 1、2、3: $G_{\text{Lead}}(s) = \frac{0.0143s + 1}{0.0070s + 1}$ ，增益 $K_c = 0.9728$ 。

滞后校正环节 1: $G_{\text{Lag}}(s) = \frac{0.0111s + 1}{0.0200s + 1}$ ，增益 $K_c = 2.5$ 。

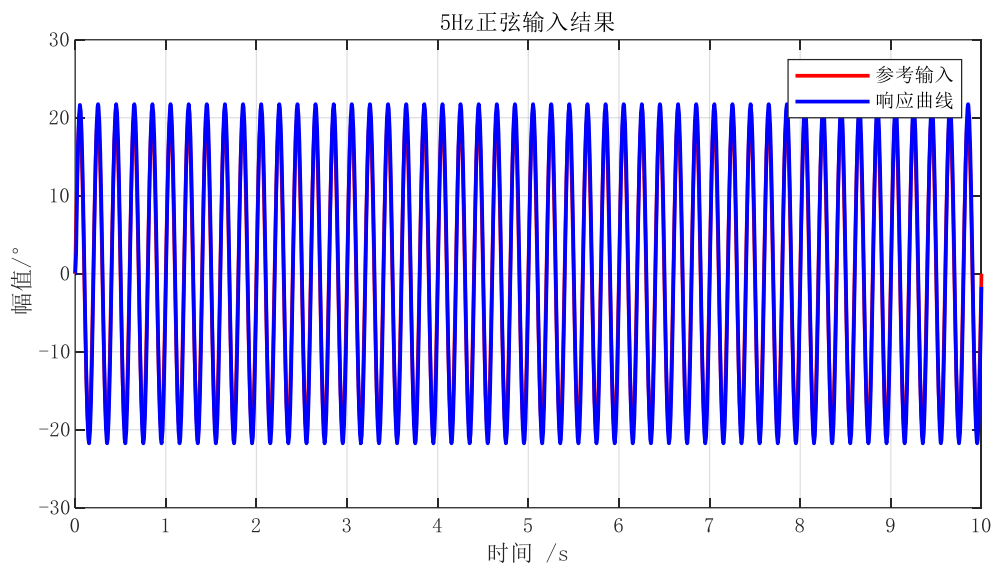


图 10 5Hz 正弦仿真响应曲线

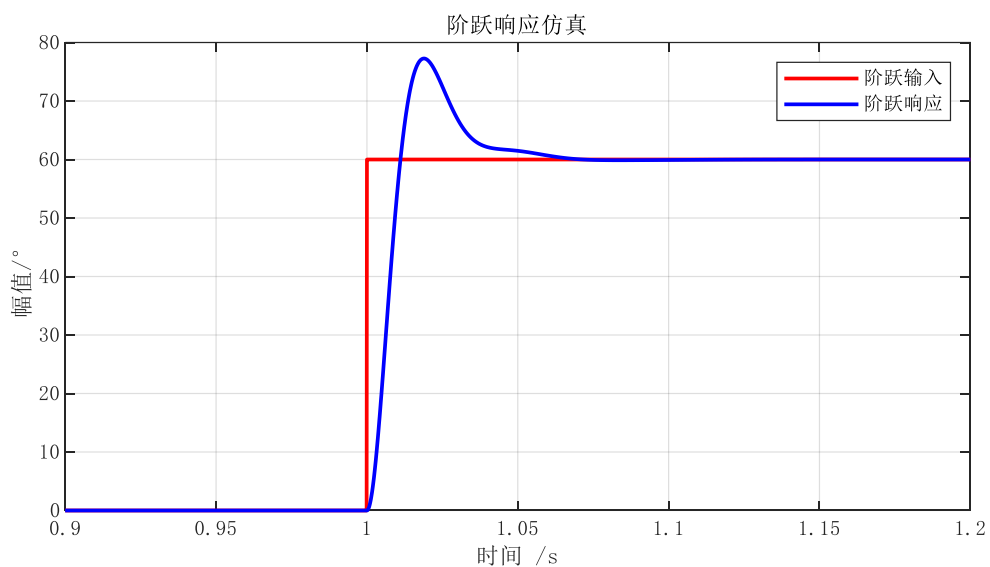
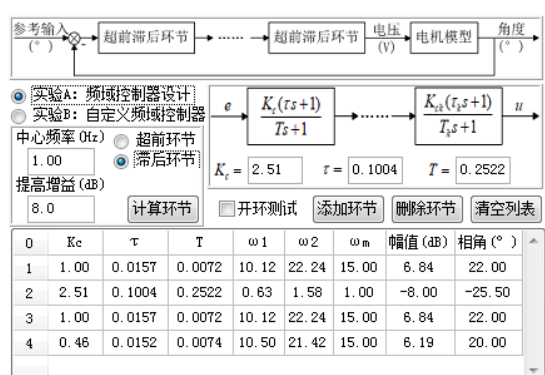


图 11 阶跃仿真响应曲线

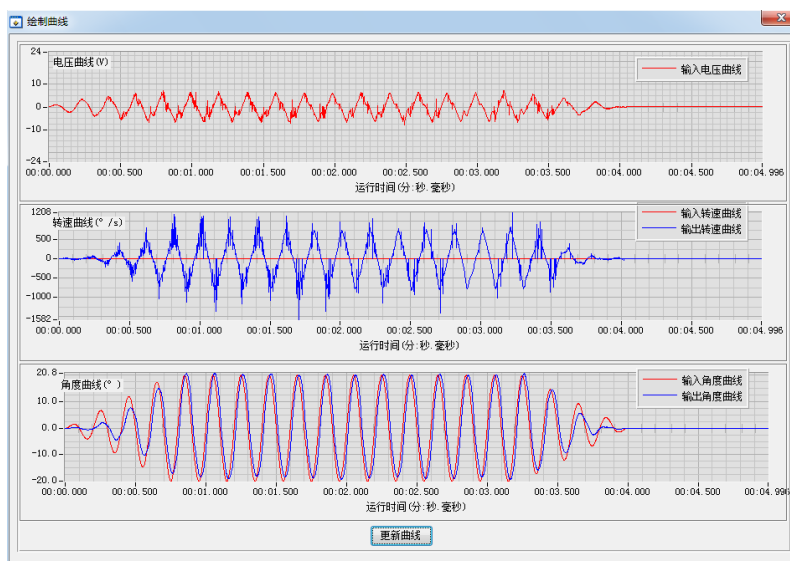
二、实验验证

1、描述频域控制器的调试过程及参数（软件截图）（5 分）



通过将输入信号频率设定为预期剪切频率，基于输出输入的相位差确认当前相位裕度。完成裕度设计后，添加合适的增益以使电机电压不饱和、输出输入曲线的幅值比接近于 1，此时，剪切频率已经设定完毕。进行闭环测试，发现未能对目标频率信号满足双十指标，低频增益不足，增加滞后校正环节及合适的 K_c 来提高系统低频增益。

2、实际调试出的性能指标及 5HZ 正弦跟踪和 60 度阶跃响应实验曲线（软件截图）（2 分）





软件截图如上所示。实际调试后，系统跟踪输入信号达到稳定。输出与输入幅值比在 95.1%~103.4%之间，相位差为 9.6° 。阶跃响应上升时间0.06s，超调量约 12.4%，稳态误差约 1.5° 。均满足设计要求。

3、使用 MATLAB 根据实验数据复现的 5HZ 正弦跟踪和 60 度阶跃响应曲线（1 分）

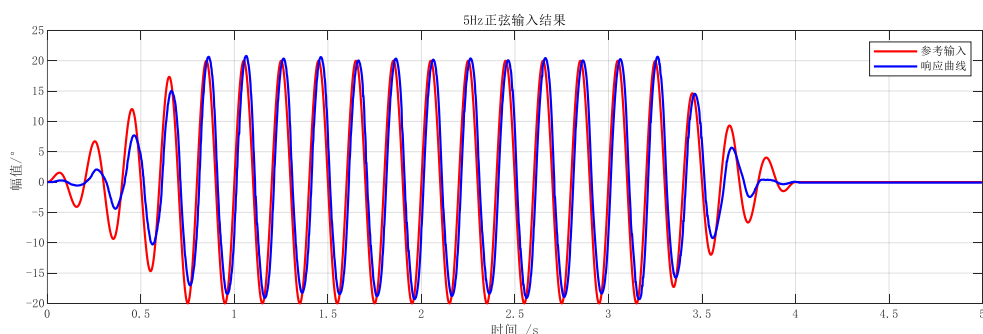


图 12 复现的 5HZ 正弦跟踪曲线

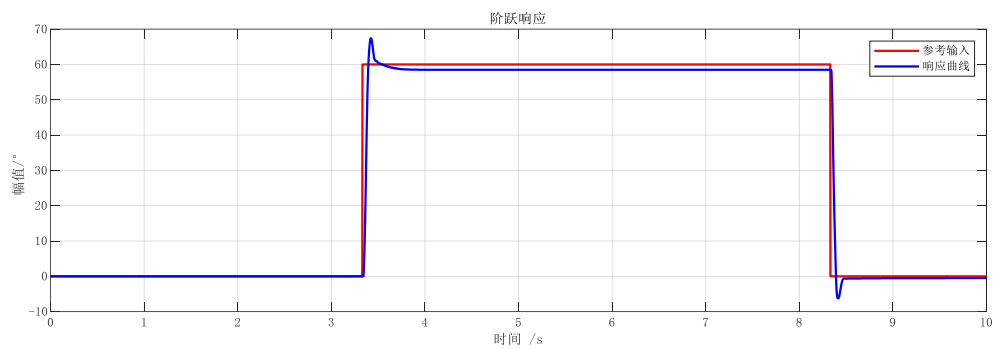


图 13 复现的 60 度阶跃响应曲线

实验五 双闭环 PID 控制器设计

一、仿真验证

1、仿真程序框图或者代码（包括代码说明）（1 分）

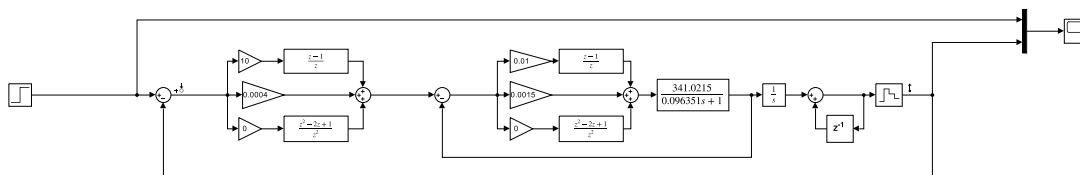


图 14 仿真程序框图

2、双闭环 PID 控制器设计过程、参数及仿真的响应曲线（1 分）

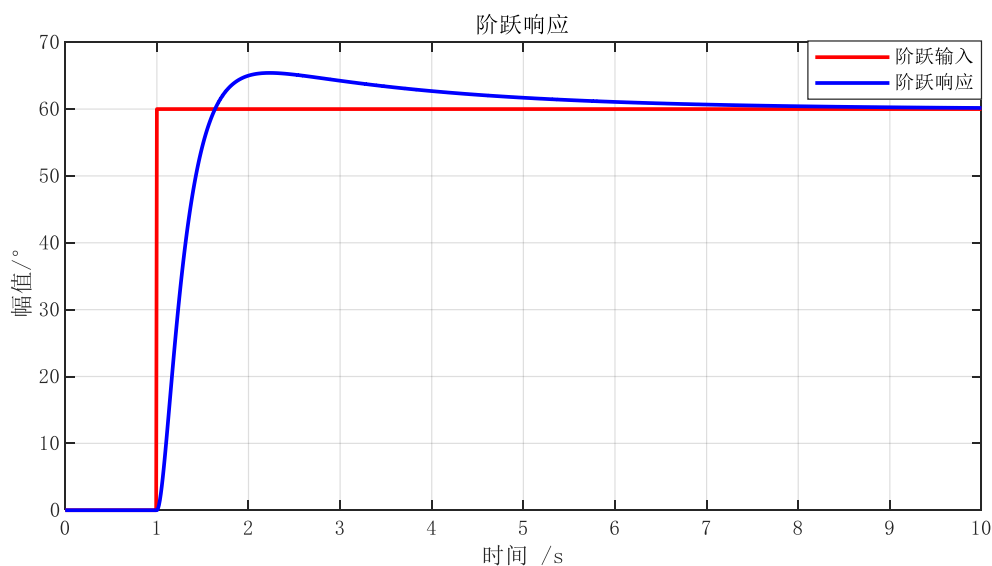


图 15 仿真阶跃响应曲线

在幅值 60° ，时长 15s 的阶跃信号输入下，系统跟踪输入信号达到稳定，上升时间约 0.54s，超调量 8.24%，稳态误差 0.2° 。

设计仿真对应的内回路参数与实物实验一致，外回路参数修改了 K_i 。

给出设计内、外回路闭环 Bode 图：

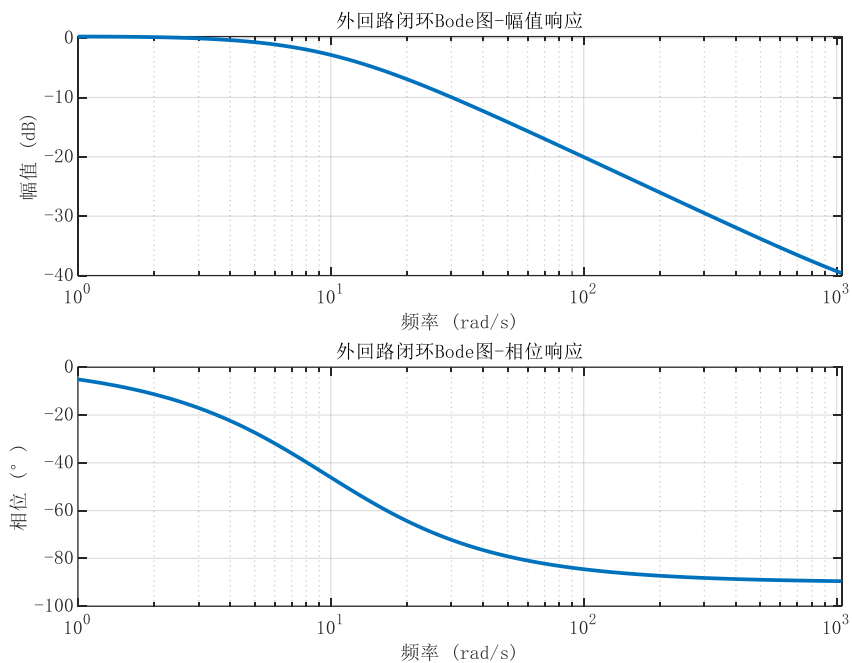


图 16 仿真外回路闭环 Bode 图

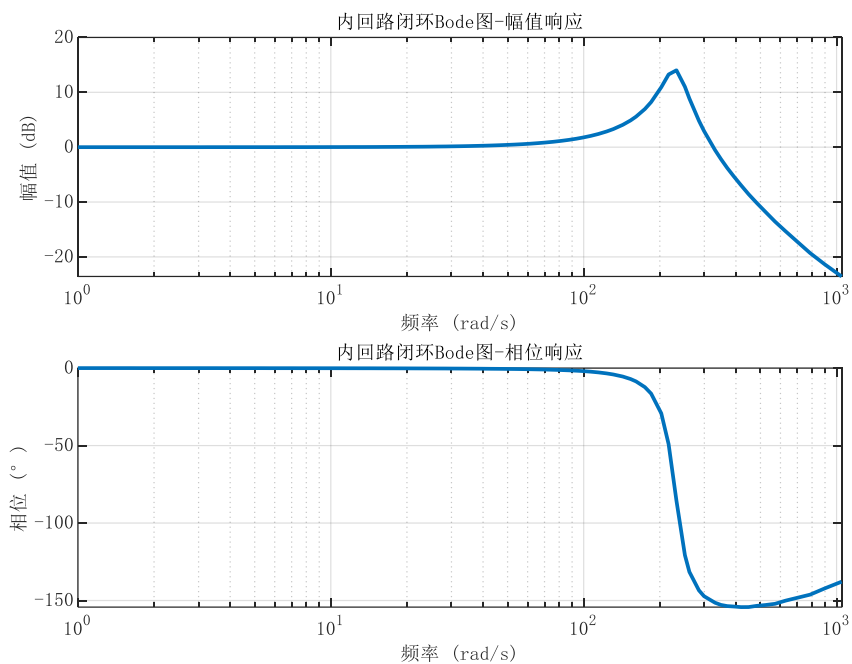


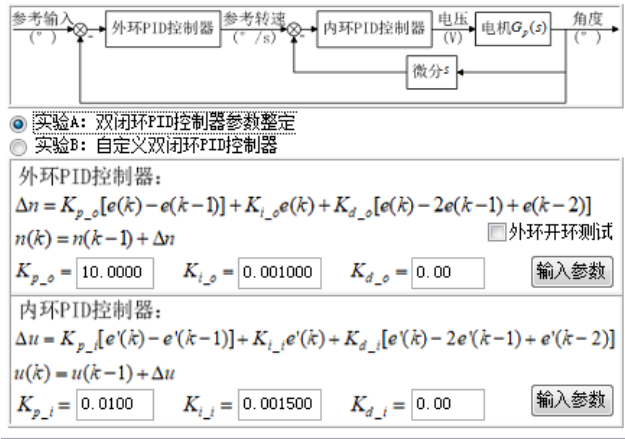
图 17 仿真内回路闭环 Bode 图

读图知内、外回路-3dB 带宽分别约为360rad/s, 10rad/s。

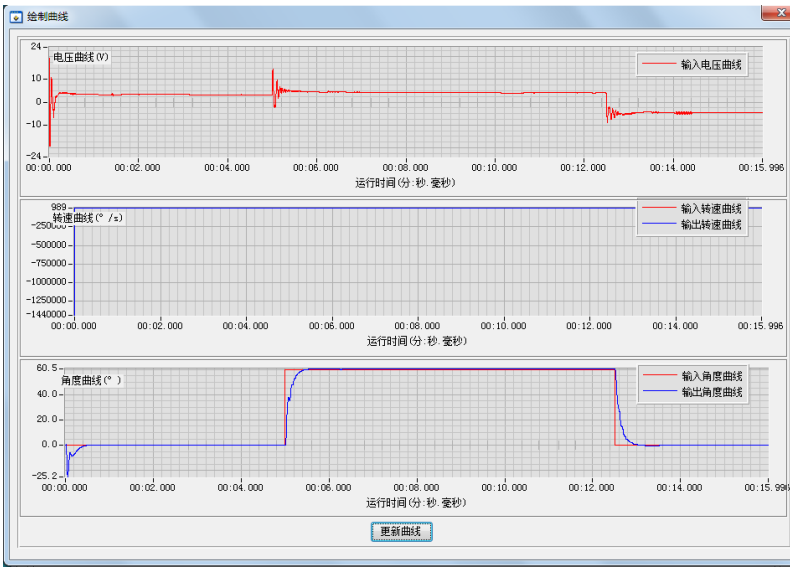
二、实验验证

1、描述双闭环 PID 控制器调试过程及参数（软件截图）（5 分）

由于扰动未知，初始直接使用仿真参数，通过实验调试来确定内环的控制器参数。设计好内环以后，进行外环的设计。取消勾选“外环开环测试”，输入幅值为 60° ，波形时长 15s 的阶跃信号，按照性能指标及外环带宽尽量小的要求设计。



2、实际调试出的性能指标及实验曲线（软件截图）（2 分）



软件截图如上所示。实际调试后，系统跟踪输入信号达到稳定，上升时间 0.48s，超调量 0.85%，稳态误差 0.3° ，均满足设计要求。使用实验中获得的参数在 Simulink 中进行仿真。测试内回路、外回路闭环 Bode 图如下：

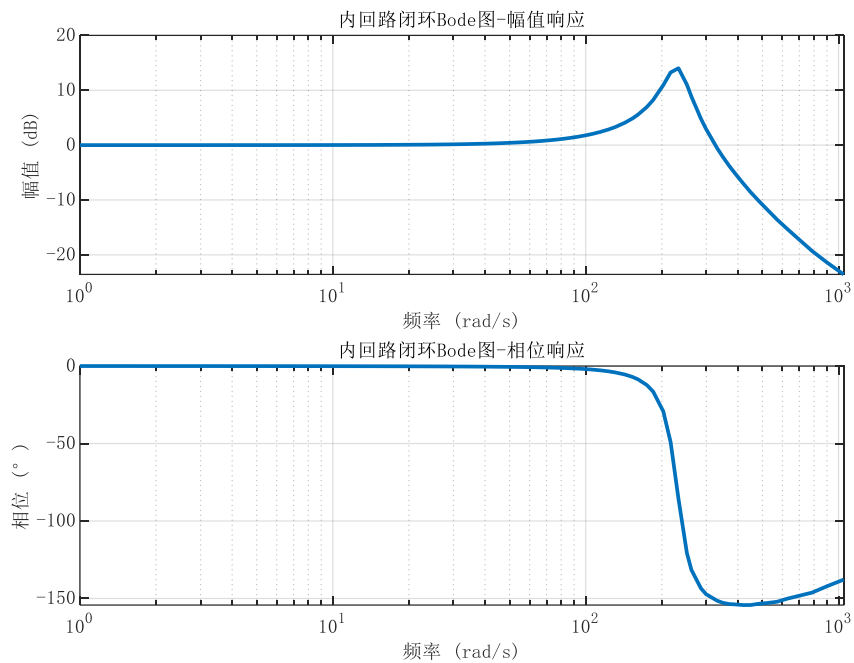


图 18 使用实物实验参数的仿真内回路闭环 Bode 图

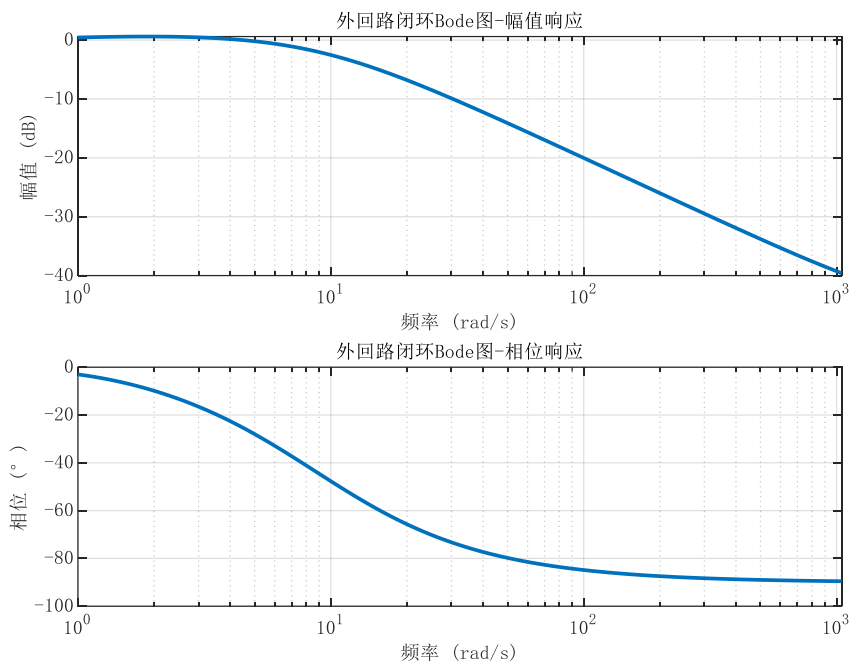


图 19 使用实物实验参数的仿真外回路闭环 Bode 图

读图知内、外回路 -3dB 带宽分别约为 360rad/s, 10rad/s，与预设设计结果一致。

3、使用 MATLAB 根据实验数据复现的曲线（1 分）

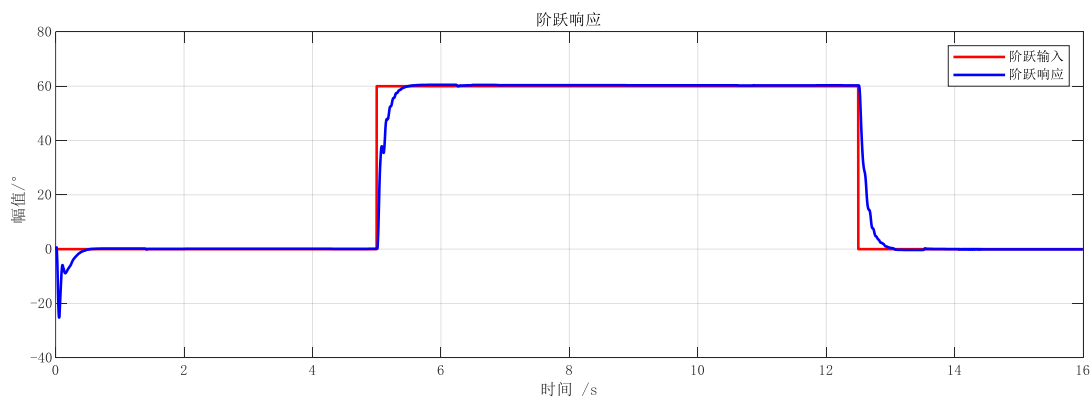


图 20 使用 MATLAB 根据实验数据复现的曲线