
西安交通大學

XI'AN JIAOTONG UNIVERSITY

# 实验报告

课程名称：自动控制原理专题实验

组员姓名：周湛昊

学院：电信学部

班级：自动化 2305

学号：2233712088

联系方式：18663839982

2025 年 12 月 26 日

# 目录

# 实验六 直流电机位置监控系统设计

## 一、任务要求

1. 利用 GUI 或 APP 设计电机位置实时控制与虚拟动态交互界面。

监控界面包含：1）控制系统 PID 参数、目标位置、控制时长等变量设置；2）显示控制响应曲线及性能指标；3）显示实时控制的位置变化虚拟动态过程，及实际位置。

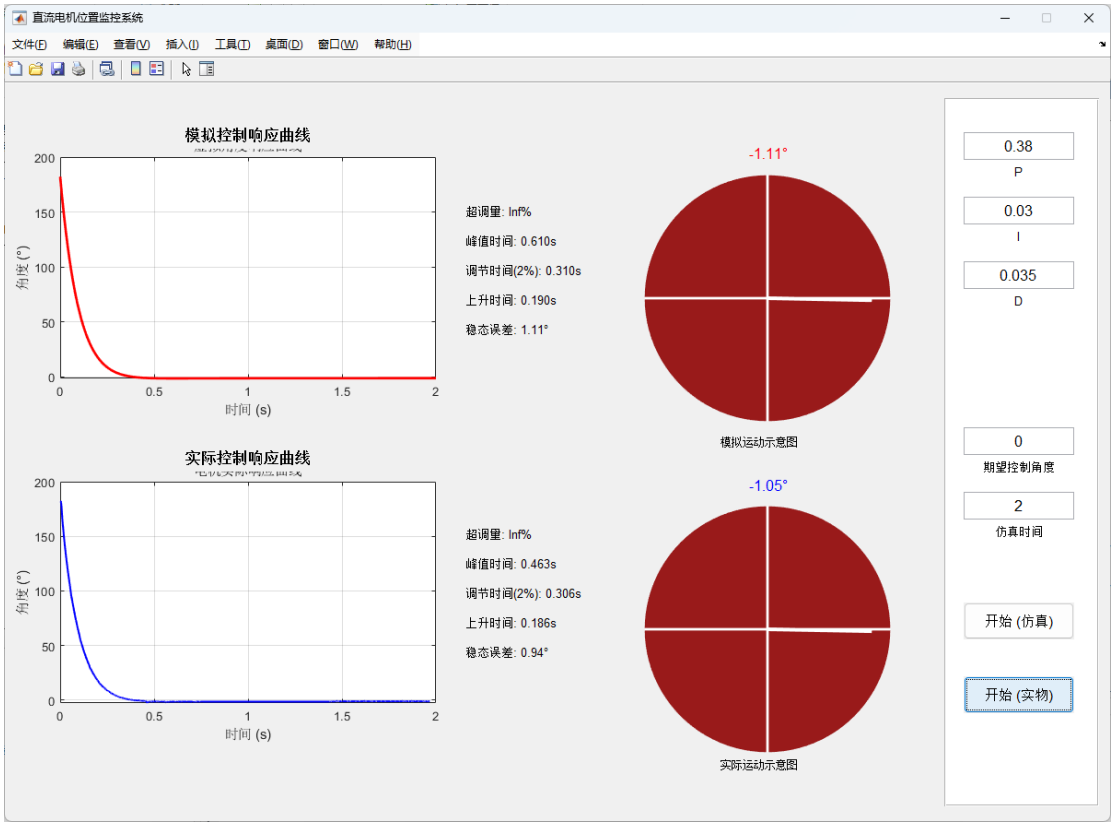
2. 调试实现 PID 或 PD 实时电机控制与虚拟动态过程，满足±180 度控制。

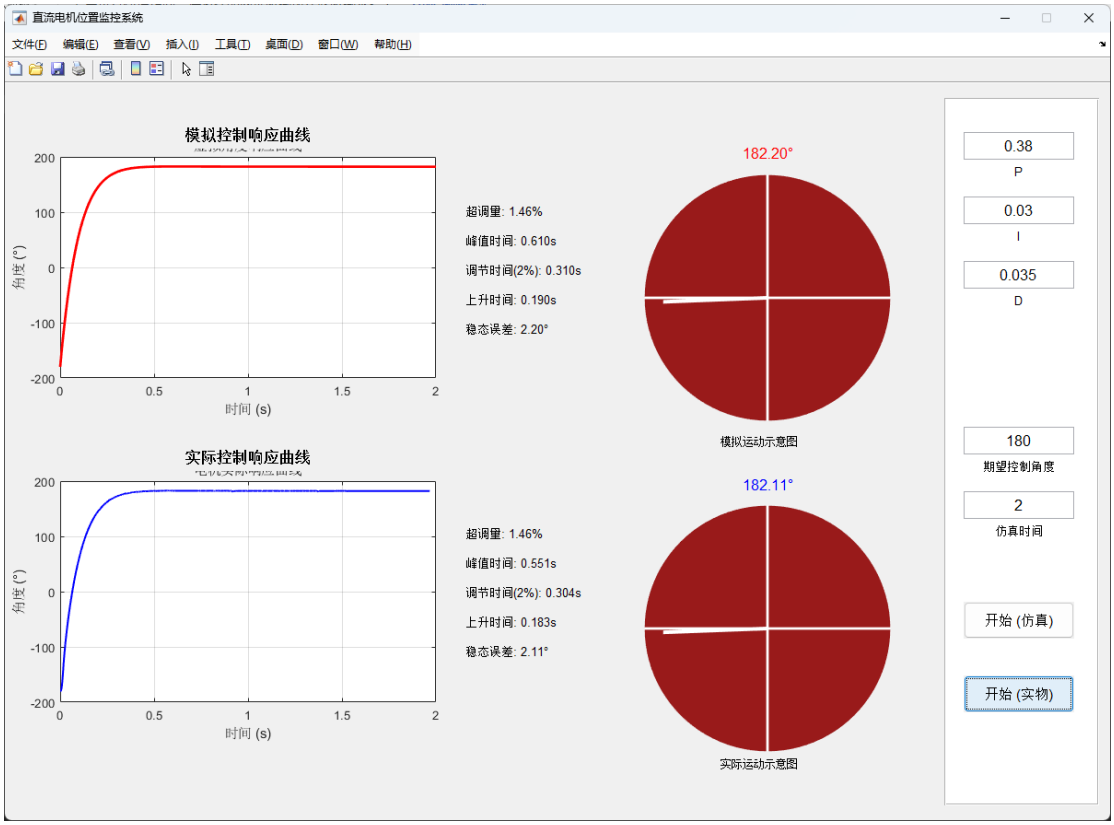
## 二、实验部分

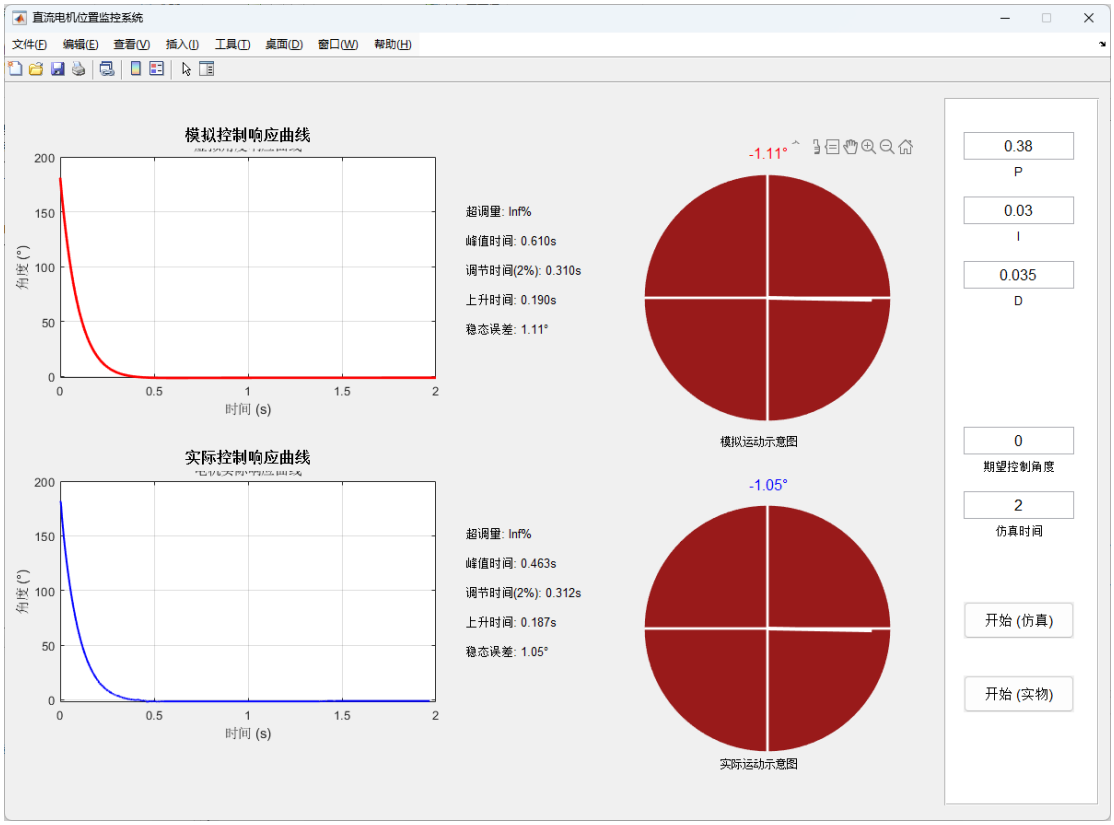
### 1. PID 控制仿真与实控调试效果

水平向右是 x 轴正方向，竖直向上是 y 轴正方向

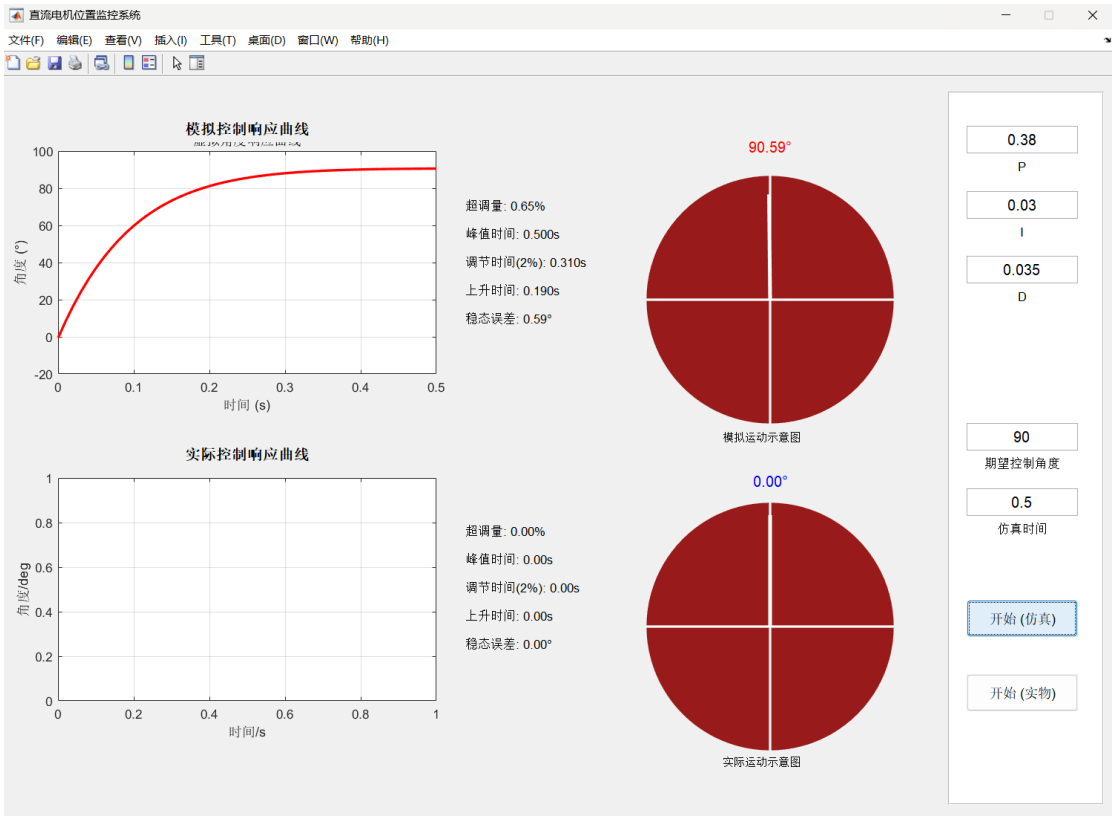
+x 方向为 0°，+y 方向为 90°，-x 方向为±180°



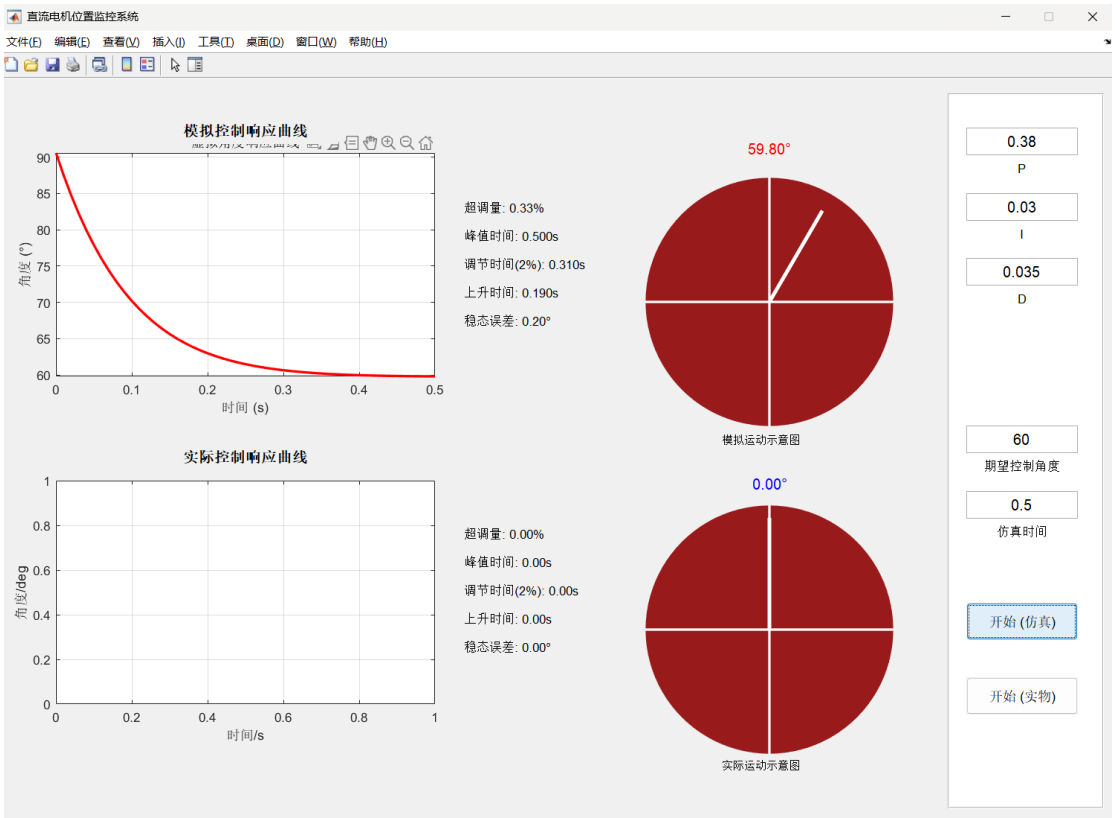

直流电机位置监控系统
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
模拟控制响应曲线
角度 (°)
时间 (s)
超调量: 1.46%
峰值时间: 0.610s
调节时间(2%): 0.310s
上升时间: 0.190s
稳态误差: 2.20°
182.20°
模拟运动示意图
实际控制响应曲线
超调量: 1.46%
峰值时间: 0.551s
调节时间(2%): 0.304s
上升时间: 0.183s
稳态误差: 2.11°
182.11°
实际运动示意图
0.38
P
0.03
I
0.035
D
180
期望控制角度
2
仿真时间
开始 (仿真)
开始 (实物)


直流电机位置监控系统
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
模拟控制响应曲线
角度 (°)
时间 (s)
超调量: Inf%
峰值时间: 0.610s
调节时间(2%): 0.310s
上升时间: 0.190s
稳态误差: 1.11°
-1.11°
模拟运动示意图
实际控制响应曲线
超调量: Inf%
峰值时间: 0.463s
调节时间(2%): 0.312s
上升时间: 0.187s
稳态误差: 1.05°
-1.05°
实际运动示意图
0.38
P
0.03
I
0.035
D
0
期望控制角度
2
仿真时间
开始 (仿真)
开始 (实物)

## 2. 各角度切换控制效果




*90 度*




*60 度*

## 3. 程序讲解与流程图

### 详细代码

```
function Quanser_PID_Control
fig = figure('Name', '直流电机位置监控系统', ...
    'Position', [100 100 1200 800], ...
    'NumberTitle', 'off', ...
    'Resize', 'off', ...
    'Color', [0.94 0.94 0.94]);

%% 右侧参数设置区（侧边栏）
panelRight = uipanel('Parent', fig, 'Position', [0.85 0.02 0.14 0.96], ...
    'BackgroundColor', 'white', 'BorderType', 'etchedin');

% PID 参数
hKp = uicontrol('Parent', panelRight, 'Style', 'edit', 'Position', [20 700
120 30], ...
    'String', '0.38', 'FontSize', 12);
uicontrol('Parent', panelRight, 'Style', 'text', 'Position', [20 670 120
25], ...
    'String', 'P', 'FontSize', 10, 'BackgroundColor', 'white');
hKi = uicontrol('Parent', panelRight, 'Style', 'edit', 'Position', [20 630
120 30], ...
    'String', '0.03', 'FontSize', 12);
uicontrol('Parent', panelRight, 'Style', 'text', 'Position', [20 600 120
25], ...
    'String', 'I', 'FontSize', 10, 'BackgroundColor', 'white');
hKd = uicontrol('Parent', panelRight, 'Style', 'edit', 'Position', [20 560
120 30], ...
    'String', '0.035', 'FontSize', 12);
uicontrol('Parent', panelRight, 'Style', 'text', 'Position', [20 530 120
25], ...
    'String', 'D', 'FontSize', 10, 'BackgroundColor', 'white');

% 仿真按钮
uicontrol('Parent', panelRight, 'Style', 'pushbutton', 'Position', [20 180
120 40], ...
    'String', '开始 (仿真)', 'FontSize', 11, 'Callback', @startSimulation);

% 期望角度
hTargetAngle = uicontrol('Parent', panelRight, 'Style', 'edit', 'Position',
[20 380 120 30], ...
    'String', '-180', 'FontSize', 12);
uicontrol('Parent', panelRight, 'Style', 'text', 'Position', [10 350 140
25], ...
    'String', '期望控制角度', 'FontSize', 10, 'BackgroundColor', 'white');
```

```
% 仿真时间
hRunTime = uicontrol('Parent', panelRight, 'Style', 'edit', 'Position', [20
310 120 30], ...
    'String', '0.5', 'FontSize', 12);
uicontrol('Parent', panelRight, 'Style', 'text', 'Position', [20 280 120
25], ...
    'String', '仿真时间', 'FontSize', 10, 'BackgroundColor', 'white');

% 实物按钮
hMotorControl = uicontrol('Parent', panelRight, 'Style', 'pushbutton',
'Position', [20 100 120 40], ...
    'String', '开始 (实物)', 'FontSize', 11, 'Callback',
@motorControlCallback, 'Enable', 'off');

%% 左侧绘图与显示区
% 上半区：模拟
uicontrol('Style', 'text', 'Position', [165 730 200 25], 'String', '模拟控制
响应曲线', ...
    'FontSize', 12, 'FontWeight', 'bold', 'BackgroundColor', [0.94 0.94
0.94]);

axesResponse = axes('Position', [0.05 0.60 0.34 0.30]);
grid on; hold on; box on; xlabel('时间/s'); ylabel('角度/deg');

hSimMetrics = uicontrol('Style', 'text', 'Position', [500 520 150 150], ...
    'String', sprintf('超调量: 0.00%%\n\n 峰值时间: 0.00s\n\n 调节时间(2%%):
0.00s\n\n 上升时间: 0.00s\n\n 稳态误差: 0.00°'), ...
    'HorizontalAlignment', 'left', 'FontSize', 10, 'BackgroundColor', [0.94
0.94 0.94]);

axSimPtr = axes('Position', [0.55 0.50 0.28 0.40]);
axis equal; xlim([-1.2 1.2]); ylim([-1.2 1.2]); hold on; axis off;
rectangle('Position',[-1 -1 2 2],'Curvature',[1 1],'FaceColor',[0.6 0.1
0.1],'EdgeColor','none');
line([-1 1], [0 0], 'Color', 'w', 'LineWidth', 2);
line([0 0], [-1 1], 'Color', 'w', 'LineWidth', 2);
hPointer = plot(axSimPtr, [0 0], [0 0.9], 'w', 'LineWidth', 3);
uicontrol('Style', 'text', 'Position', [770 400 100 20], 'String', '模拟运动
示意图', 'FontSize', 9, 'BackgroundColor', [0.94 0.94 0.94]);
hSimAngleDisp = uicontrol('Style', 'text', 'Position', [780 710 100 25],
'String', '0.00°', ...
    'FontSize', 12, 'ForegroundColor', 'r', 'BackgroundColor', [0.94 0.94
0.94]);
```

```
% 下半区：实物
uicontrol('Style', 'text', 'Position', [165 380 200 25], 'String', '实际控制响应曲线', ...
    'FontSize', 12, 'FontWeight', 'bold', 'BackgroundColor', [0.94 0.94 0.94]);

axesMotorResponse = axes('Position', [0.05 0.16 0.34 0.30]);
grid on; hold on; box on; xlabel('时间/s'); ylabel('角度/deg');

hMotorMetrics = uicontrol('Style', 'text', 'Position', [500 170 150 150], ...
    'String', sprintf('超调量: 0.00%%\n\n 峰值时间: 0.00s\n\n 调节时间(2%%): 0.00s\n\n 上升时间: 0.00s\n\n 稳态误差: 0.00°'), ...
    'HorizontalAlignment', 'left', 'FontSize', 10, 'BackgroundColor', [0.94 0.94 0.94]);

axRealPtr = axes('Position', [0.55 0.06 0.28 0.40]);
axis equal; xlim([-1.2 1.2]); ylim([-1.2 1.2]); hold on; axis off;
rectangle('Position',[-1 -1 2 2],'Curvature',[1 1],'FaceColor',[0.6 0.1 0.1],'EdgeColor','none');
line([-1 1], [0 0], 'Color', 'w', 'LineWidth', 2);
line([0 0], [-1 1], 'Color', 'w', 'LineWidth', 2);
hMotorPointer = plot(axRealPtr, [0 0], [0 0.9], 'w', 'LineWidth', 3);
uicontrol('Style', 'text', 'Position', [770 50 100 20], 'String', '实际运动示意图', 'FontSize', 9, 'BackgroundColor', [0.94 0.94 0.94]);
hMotorAngleDisp = uicontrol('Style', 'text', 'Position', [780 350 100 25], 'String', '0.00°', ...
    'FontSize', 12, 'ForegroundColor', 'b', 'BackgroundColor', [0.94 0.94 0.94]);

% 句柄
handles = guihandles(fig);
handles.hKp = hKp; handles.hKi = hKi; handles.hKd = hKd;
handles.hTargetAngle = hTargetAngle; handles.hRunTime = hRunTime;
handles.axesResponse = axesResponse; handles.hPointer = hPointer;
handles.hSimMetrics = hSimMetrics; handles.hSimAngleDisp = hSimAngleDisp;
handles.hMotorControl = hMotorControl; handles.axesMotorResponse = axesMotorResponse; handles.hMotorPointer = hMotorPointer;
handles.hMotorMetrics = hMotorMetrics; handles.hMotorAngleDisp = hMotorAngleDisp;
handles.lastSimEnd = 0; handles.lastMotorEnd = 0;
guidata(fig, handles);

%% 仿真回调
```

```
function startSimulation(hObject, ~)
    data = guidata(hObject);
    target_delta = str2double(data.hTargetAngle.String);
    t_len = str2double(data.hRunTime.String);
    start_pos = data.lastSimEnd;

    if target_delta == 0
        final_target = 0;
    else
        final_target = target_delta;
    end

    G = tf(30, [0.1 1 0]);
    C = pid(str2double(data.hKp.String), str2double(data.hKi.String),
str2double(data.hKd.String));
    sys = feedback(C*G, 1);

    t_sim = 0:0.01:t_len;
    [y_raw, ~] = step(sys, t_sim);
    y_sim = start_pos + (final_target - start_pos) * y_raw;

    [ov, tr, tp, ts] = calculate_performance(t_sim, y_sim, start_pos,
final_target);

    save('dynamic_response_data.mat', 't_sim', 'y_sim', 'start_pos',
'final_target', 'ov', 'tr', 'tp', 'ts', 'target_delta');
    cla(data.axesResponse);
    plot(data.axesResponse, t_sim, y_sim, 'r', 'LineWidth', 2);
    xlabel(data.axesResponse, '时间 (s)');
    ylabel(data.axesResponse, '角度 (°)');
    title(data.axesResponse, '虚拟角度响应曲线');
    grid(data.axesResponse, 'on');

    for i = 1:length(t_sim)
        theta = y_sim(i);
        set(data.hPointer, 'XData', [0 0.85*cosd(theta)], 'YData', [0
0.85*sind(theta)]);
        data.hSimAngleDisp.String = sprintf('%.2f°', theta);
        drawnow
        if mode(i,3) == 1
            drawnow;
        end
    end
    drawnow;
```

```
    data.hSimMetrics.String = sprintf(['超调量: %.2f%%\n\n' ...
        '峰值时间: %.3fs\n\n' ...
        '调节时间(2%%): %.3fs\n\n' ...
        '上升时间: %.3fs\n\n' ...
        '稳态误差: %.2f°'], ...
        ov, tp, ts, tr, abs(final_target - y_sim(end)));

    data.lastSimEnd = y_sim(end);
    guidata(hObject, data);
    data.hMotorControl.Enable = 'on';
end

%% 实物控制回调
function motorControlCallback(hObject, ~)
    data = guidata(hObject);

    if ~exist('dynamic_response_data.mat', 'file')
        errordlg('请先运行虚拟仿真！');
        return;
    end

    load('dynamic_response_data.mat', 't_sim', 'y_sim', 'start_pos',
'final_target', 'ov', 'tr', 'tp', 'ts', 'target_delta');

    board_type = 'qube_servo3_usb';
    freq = 1000;
    dt = 1 / freq;

    [board, err] = quanser.hardware.hil.open(board_type, '0');
    if err ~= 0
        errordlg('硬件连接失败');
        return;
    end

    try
        board.write_digital(0, 1);
        board.write_analog(0, 0);

        [task, err] = board.task_create_encoder_reader(1000, 0);
        if err ~= 0
            error('创建编码器任务失败');
        end

        task.start(0, freq, -1);
```

```
counts = task.read_encoder(10);
current_motor_pos = double(mean(counts)) * 360 / 2048;

if target_delta == 0
    motor_target = 0;
else
    motor_target = final_target;
end

control_time = t_sim(end);
actual_t = 0:dt:control_time;
y_sim_interp = interp1(t_sim, y_sim, actual_t, 'pchip', 'extrap');

actual_y = zeros(size(actual_t));
previous_error = 0;
integral_error = 0;

Kp = str2double(data.hKp.String);
Ki = str2double(data.hKi.String);
Kd = str2double(data.hKd.String);

cla(data.axesMotorResponse);
hLine = plot(data.axesMotorResponse, 0, 0, 'b', 'LineWidth', 1.5);
xlabel(data.axesMotorResponse, '时间 (s)');
ylabel(data.axesMotorResponse, '角度 (°)');
title(data.axesMotorResponse, '电机实际响应曲线');
grid(data.axesMotorResponse, 'on');

last_plot_time = 0;
plot_interval = 0.05;

for k = 1:length(actual_t)
    curr_t = actual_t(k);
    cmd = y_sim_interp(k);

    counts = task.read_encoder(3);
    pos = double(mean(counts)) * 360 / 2048;
    actual_y(k) = pos;

    error = cmd - pos;

    P_term = Kp * error;
```

```
            integral_error = integral_error + error * dt;
            max_integral = 10;
            if integral_error > max_integral
                integral_error = max_integral;
            elseif integral_error < -max_integral
                integral_error = -max_integral;
            end
            I_term = Ki * integral_error;

            derivative = (error - previous_error) / dt;
            alpha = 0.1;
            filtered_derivative = alpha * derivative;
            D_term = Kd * filtered_derivative;

            voltage = P_term + I_term + D_term;
            if voltage > 3
                voltage = 3;
            elseif voltage < -3
                voltage = -3;
            end

            board.write_analog(0, voltage);
            previous_error = error;

            if curr_t - last_plot_time >= plot_interval
                set(hLine, 'XData', actual_t(1:k), 'YData', actual_y(1:k));
                set(data.hMotorPointer, 'XData', [0 0.85*cosd(pos)],
'YData', [0 0.85*sind(pos)]);
                data.hMotorAngleDisp.String = sprintf('%.2f°', pos);
                drawnow;
                last_plot_time = curr_t;
            end

            pause(0.001);
        end

        task.stop;
        task.close;

        if target_delta == 0
            [motor_ov, motor_tr, motor_tp, motor_ts] =
calculate_performance(actual_t, actual_y, current_motor_pos, 0);
        else
```

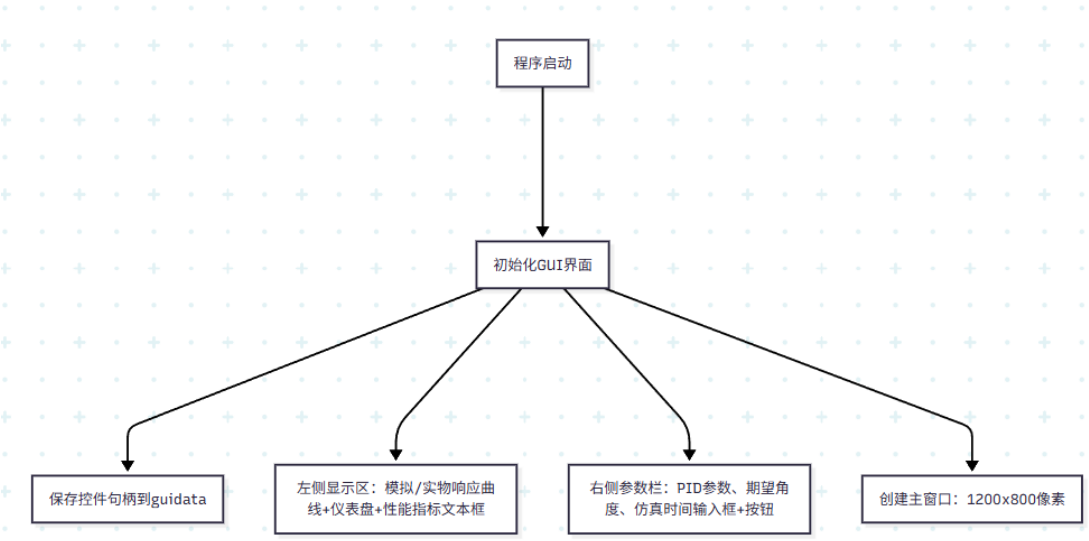
```
            [motor_ov, motor_tr, motor_tp, motor_ts] =
calculate_performance(actual_t, actual_y, current_motor_pos, motor_target);
        end

        data.hMotorMetrics.String = sprintf(['超调量: %.2f%%\n\n' ...
            '峰值时间: %.3fs\n\n' ...
            '调节时间(2%%): %.3fs\n\n' ...
            '上升时间: %.3fs\n\n' ...
            '稳态误差: %.2f°'], ...
            motor_ov, motor_tp, motor_ts, motor_tr, abs(motor_target -
actual_y(end)));

        data.lastMotorEnd = actual_y(end);
        guidata(hObject, data);

    catch ME
        fprintf('控制异常: %s\n', ME.message);
        errordlg(sprintf('控制异常: %s', ME.message));
    end

    board.write_analog(0, 0);
    board.write_digital(0, 0);
    board.close;
end

end

%% 性能指标计算函数
function [ov, tr, tp, ts] = calculate_performance(t, y, start_val,
target_val)
if abs(target_val - start_val) < 1e-6
    ov = 0; tr = 0; tp = 0; ts = 0;
    return;
end

y_norm = (y - start_val) / (target_val - start_val);

if target_val > start_val
    max_val = max(y);
    ov = (max_val - target_val) / (abs(target_val)) * 100;
else
    min_val = min(y);
    ov = (target_val - min_val) / (abs(target_val)) * 100;
end
```

```
if ov < 0, ov = 0; end

idx_10 = find(y_norm >= 0.1, 1);
idx_90 = find(y_norm >= 0.9, 1);
if ~isempty(idx_10) && ~isempty(idx_90)
    tr = t(idx_90) - t(idx_10);
else
    tr = 0;
end

if target_val > start_val
    [~, idx_max] = max(y);
else
    [~, idx_max] = min(y);
end
tp = t(idx_max);

error_band = 0.02;
ts = 0;
for i = 1:length(y)
    if abs(y(i) - target_val) <= abs(target_val - start_val) * error_band
        if i + 20 <= length(y)
            all_in_band = all(abs(y(i:i+20) - target_val) <= abs(target_val
- start_val) * error_band);
            if all_in_band
                ts = t(i);
                break;
            end
        else
            ts = t(i);
            break;
        end
    end
end
if ts == 0
    ts = t(end);
end
end
```

## 流程图与详细内容解析

四个函数：

Quanser_PID_Control 用于主程序与界面初始化：包括保存控件到句柄，创建窗口、按钮、曲线、转盘和性能指标的文本框等 UI 组件

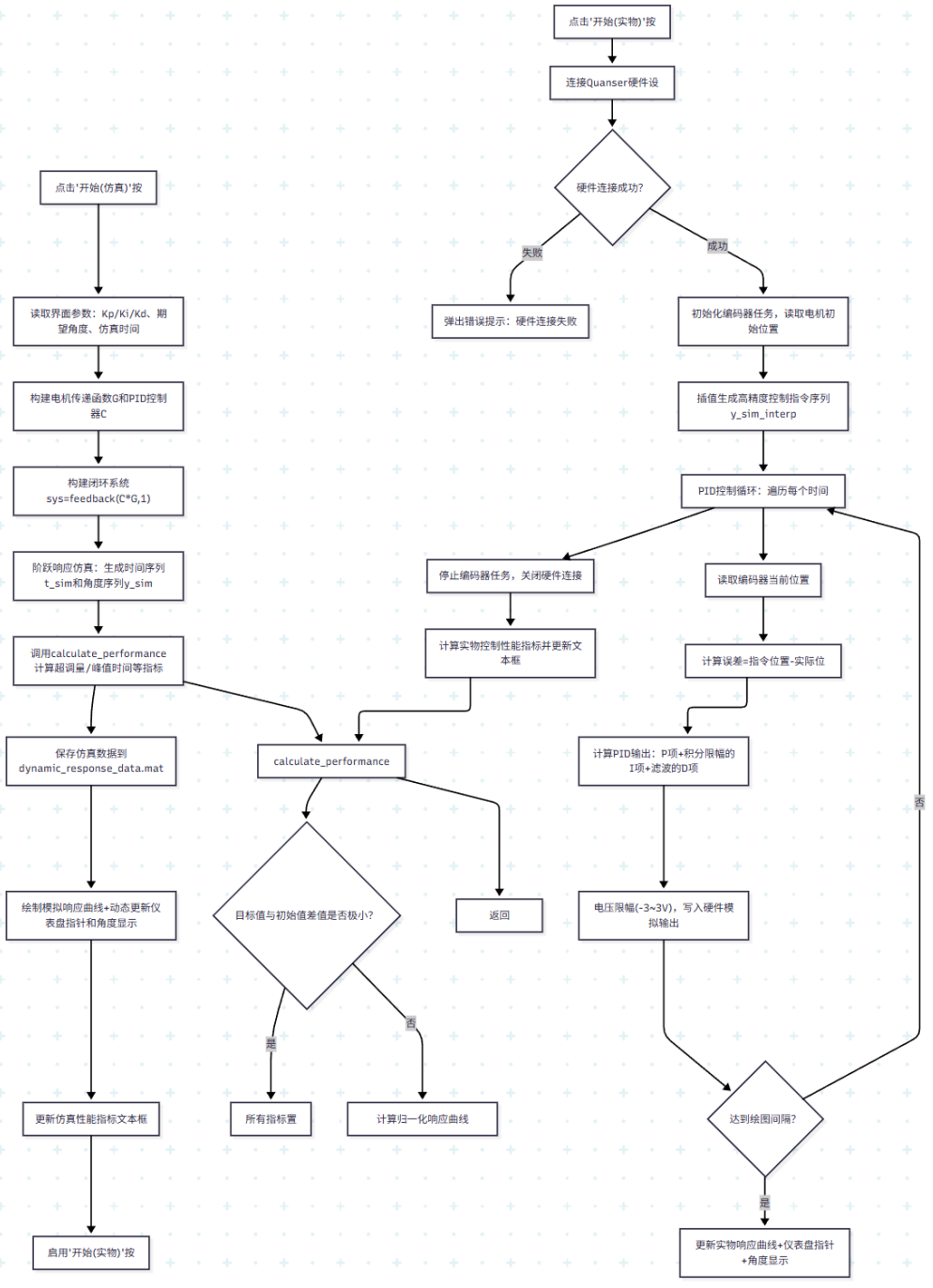



startSimulation 是仿真回调函数

motorControlCallback 是实物控制

calculate_performance 是计算性能指标。

这三部分的详细功能可见下方流程图




## 4. 附加功能

默认程序启动时的角度为 0°

添加滤波器减小系统响应曲线以及实际电机的抖动

实际应用中还有过零点的检测，这个在目前位置控制中不是特别能看的出来，因为范围只有-180 到 180°，如果使用超过超过 360° 的目标位置，这还需要计算转过了多少“圈”

### 5. 总结 PID 作用与调节规律

#### PID 作用

比例环节 P：成比例地反映控制系统的偏差信号，增大 Kp 可提高响应速度，但 t 太大会导致系统不稳定

积分环节 I：能消除稳态误差，但会一定程度上降低系统的稳定性。

微分环节 D：反映偏差信号的变化趋势，能改善系统的动态性能，但会使系统对噪声敏感

#### PID 调节规律

先调 P，从小到大增加 Kp，然后调微分，为了抑制 P 带来的震荡并减少超调，逐渐增加 Kd，直到超调量减小到合适范围，最后加积分 I，如果系统稳定后与目标值仍有稳态误差存在，就从小到大增加 Ki，不过 Ki 的值偏小，太大的 Ki 会引起振荡。

# 三、实验总结

#### 实验收获、遇到的问题和解决办法

通过 MATLAB GUI 实现了一个直流电机位置监控系统，包含仿真和实物两种监控模式，能够实时调节 PID 观察电机转动情况和响应曲线，也能够计算出每次响应的性能指标。

除了主要实现的功能以外，实际的代码编写和调试过程中也有许多实际的问题与解决方案：

利用 MATLAB 的 GUI 界面的编辑绘图功能，能够实时编辑位置，能大大提高在放置 UI 时的效率。

调节 PID 的过程中，在代码中添加一个滤波器，设置合适的参数可以减小电机的抖动，更容易调节至稳定。

适度的增加指针的更新频率能够使动画更加丝滑，同时适度增加采样频率能够使实物调节更丝滑，同时电机如果抖动过大，出现较大的震动声音，也有可能是因为采样频率太低导致的。

最后，通过这个实验，学习到了如何用 MATLAB 进行 UI 界面的设计，与其他

代码的UI设计如Python的Qt不同，MATLAB的代码更加通俗简洁，同时具备仿真上的功能，在控制相关的领域更有优势。同时PID调节相关的内容，虽然之前有过PID的实验内容，但是在实物电机上使用与完全在仿真环境里进行也有明显不同，通过实物的调节，加深了对PID作用与调节方式的理解。

**关于如何用AI大模型解决实验中遇到的问题**

大语言模型具备世界知识，能够很好的解决我们遇到的问题，我们可以利用这一点让它来帮我们写代码、提出问题解决方案、给出如何调整PID的具体方法或经验。

但是我在实验中明显发现了一些问题：

1. 有的AI大模型的多模态能力不行，其实就是它识图能力弱，例如Deepseek这种仅具备语言能力的模型，因此在模型帮忙搭建好代码框架之后还需要自己去手动调节。
2. 对于模型写的大量代码，有无从下手的感觉，这也是现在vibecoding的问题之一，这种情况下需要对于AI写的代码要了解其结构和作用，而不能仅靠与AI对话完成代码的编写
3. 这种基于人类反馈强化学习微调出来的大语言模型，通常对于实验中所涉及到的这种嵌入式代码、仿真代码不具备很好的理解能力，因为它不能够从实际结果中获得反馈，也就是说它无法自己 debug，而需要我们去告诉它出现了什么问题，但是我们的描述并不像代码那样成体系。要解决这个问题，一个方式是要有具体、详细、成体系的描述，也就是prompt，另一个方式就是要靠自己去debug，也就是第2点所说的，要对代码有理解