实验六 直流电机位置监控系统设计

一、实验目的

1. 掌握利用 Matlab GUI 或 APP 进行实验综合设计。
2. 熟练应用 PID 整定方法。

二、实验内容

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。
3. 参考画面

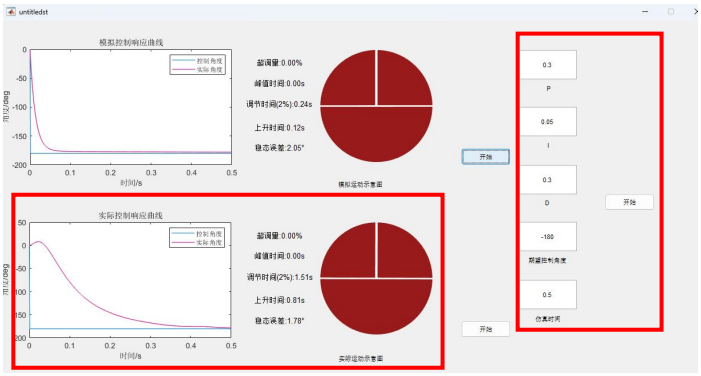

图 1 监控系统参考画面

三、任务解析

1. 利用 GUI 设计电机转角控制虚拟仿真与实际电机控制系统界面

修改实验平台配置

要使用 *.m* 文件直接启动 Qube 硬件的随动控制，首先需要对实验平台的相关配置进行修改。在代码中，*board_type* 变量指定了使用的实验平台类型，*board_identifier* 变量指定了系统中使用的 QUBE 卡的标识符。

```
board_type = 'qube_servo3_usb'; % 使用 QUBE - SERVO3 - USB 实验平台
board_identifier = '0';          % 使用系统中的第一个 QUBE 卡
```

修改上述配置后，程序就可以正确识别并连接到指定的 Qube 实验平台。

读取编码器数据

在硬件控制过程中，需要实时获取硬件的旋转臂角度数据。这可以通过编码器来实现，在代码中使用 *encoder_channels* 变量指定编码器通道。

```
encoder_channels = 0; % 编码器通道 0
```

通过以下代码可以实时读取硬件的旋转臂角度数据：

```
count = task.read_encoder(samples_to_read);
```

这里的 task 是通过 board.task_create_encoder_reader 函数创建的编码器读取任务，samples_to_read 表示每次调用 hil_task_read 时读取的样本数。读取到的 count 值可以通过转换公式将其转换为实际的角度值。

```
position  = double(count) * 360 / 2048; % 将编码器计数转换为角度
```

2、要求系统可实现 PD 或 PID 控制，分虚拟仿真控制和实际控制两部分

实现纯虚拟的角度控制

运行例程可以看到电机跟随正弦函数进行随动控制。在这个过程中，我们可以结合电机的角度转动数学模型和 PID 的参数控制，得到一条纯虚拟的角度控制响应曲线。

在代码中，首先定义被控对象的传递函数 *G(s)* 和 PID 控制器的传递函数 *C(s)*：

```
G = tf(30, [0.1 1 0]);
% 获取 PID 参数
Kp = str2double(get(data.hKp, 'String'));
Ki = str2double(get(data.hKi, 'String'));
Kd = str2double(get(data.hKd, 'String'));
% 定义 PID 控制器传递函数 C(s) = Kp + Ki/s + Kd*s
C = pid(Kp, Ki, Kd);
```

然后计算闭环传递函数：

```
closedLoop = feedback(C*G, 1);
```

接着根据用户输入的运行时间和目标角度，计算阶跃响应：

```
runTime = str2double(get(data.hRunTime, 'String'));
% 仿真时间
t = 0:0.01:runTime;
% 获取目标角度
targetAngle = str2double(get(data.hTargetAngle, 'String'));
% 计算阶跃响应（目标角度为用户设定值）
[y, t] = step(closedLoop, t);
y = targetAngle * y; % 缩放到目标角度
```

通过上述步骤，我们就可以得到一条纯虚拟的角度控制响应曲线 *y*，它表示在给定 PID 参数和目标角度下，系统的理论响应。

让硬件跟随虚拟曲线转动

根据得到的虚拟响应曲线，我们可以通过编程让硬件跟随这条曲线转动，最终间接实现对硬件的角度控制。

在电机控制回调函数 *motorControlCallback* 中，首先加载理论仿真数据：

```
saved_data = load('dynamic_response_data.mat', 't', 'y');
t = saved_data.t;
command_signal = saved_data.y;
```

然后在控制循环中，根据当前时间从理论仿真数据中获取对应的指令信号：

```
for time = 0:period:tf
    % 从理论仿真数据中获取当前时间的指令信号
    command_index = find(t >= time, 1);
    if ~isempty(command_index)
        command = command_signal(command_index);
    end
    % 读取编码器数据
    count = task.read_encoder(samples_to_read);
    % 将编码器计数转换为角度
    position  = double(count) * 360 / 2048;
    % 计算位置误差
    pos_error = command - position;
    % 应用比例控制
    voltage   = gain * pos_error;
    % 发送控制信号
    err = board.write_analog(analog_channels, voltage);
    ...
end
```

在每个控制周期内，根据当前时间从理论仿真数据中获取对应的指令信号 *command*，然后读取编码器数据得到当前硬件的实际位置 *position*，计算位置误差 *pos_error*，并应用比例控制计算控制电压 *voltage*，最后将控制电压发送到硬件，从而实现硬件跟随虚拟曲线转动。

3、控制界面要求有 PD 或 PID 参数变量、期望控制角度变量

制作 GUI 集成所有功能(以下程序仅供参考，需自己修改)

为了方便用户操作和观察实验结果，我们可以将上述硬件控制和 PID 仿真功能集成到一个 GUI 中。

在 *Quanser_PID_Control* 函数中，首先创建主窗口和各种控件，包括 PID 参数输入框、目标角度输入框、运行时间输入框、虚拟仿真按钮、电机控制按钮、实时角度显示框、性能指标显示框以及绘图区域等：

```
fig = figure('Name', 'Quanser 角度 PID 控制', ...
             'Position', [150 150 1200 800], ...
             'NumberTitle', 'off', ...
             'Resize', 'off');

% 创建 PID 参数输入框和标签
% kp
uicontrol('Style', 'text', 'Position', [50 730 40 40], 'String', 'Kp', 'FontSize', 10, '
HorizontalAlignment', 'left');
hKp = uicontrol('Style', 'edit', 'Position', [80 740 50 40], 'String', '5');
% ki
uicontrol('Style', 'text', 'Position', [150 730 40 40], 'String', 'Ki', 'FontSize', 10, '
HorizontalAlignment', 'left');
hKi = uicontrol('Style', 'edit', 'Position', [180 740 50 40], 'String', '0');
% kd
uicontrol('Style', 'text', 'Position', [250 730 40 40], 'String', 'Kd', 'FontSize', 10, '
HorizontalAlignment', 'left');
hKd = uicontrol('Style', 'edit', 'Position', [280 740 50 40], 'String', '0.05');

% 目标角度输入
uicontrol('Style', 'text', 'Position', [350 730 100 40], 'String', '目标角度 (°):', 'Fo
ntSize', 10, 'HorizontalAlignment', 'left');
hTargetAngle = uicontrol('Style', 'edit', 'Position', [430 740 100 40], 'String', '30
');

% 运行时间输入
uicontrol('Style', 'text', 'Position', [550 730 100 40], 'String', '运行时间 (s):', 'Fo
ntSize', 10, 'HorizontalAlignment', 'left');
hRunTime = uicontrol('Style', 'edit', 'Position', [630 740 100 40], 'String', '1');

% 虚拟仿真
hStartSim = uicontrol('Style', 'pushbutton', 'Position', [750 740 100 40], ...
            'String', '虚拟仿真', 'FontSize', 12, 'Callback', @startSimulation);

% 虚拟实时角度显示
uicontrol('Style', 'text', 'Position', [940 740 150 30], 'String', '虚拟仿真实时角度
', 'FontWeight', 'bold', 'FontSize', 12);
hRealTimeAngle = uicontrol('Style', 'text', 'Position', [970 710 100 30], 'String', '
0.00 度', 'FontSize', 12);

% 虚拟性能指标显示
```

```
uicontrol('Style', 'text', 'Position', [940 670 150 30], 'String', '虚拟仿真性能指标', 'FontWeight', 'bold', 'FontSize', 12);
hOvershoot = uicontrol('Style', 'text', 'Position', [920 620 130 30], 'String', '超调量: ');
hSettlingTime = uicontrol('Style', 'text', 'Position', [920 580 130 30], 'String', '调节时间: ');
hSteadyError = uicontrol('Style', 'text', 'Position', [920 540 130 30], 'String', '稳态误差: ');

% 创建虚拟动态响应的绘图区域
axesResponse = axes('Parent', fig, 'Position', [0.05 0.58 0.4 0.3]);
title('虚拟角度动态响应曲线');
xlabel('时间 (s)');
ylabel('角度 (°)');
grid on;
hold on;

% 创建虚拟动态指针的绘图区域
axesPointer = axes('Parent', fig, 'Position', [0.4 0.58 0.5 0.3]);
title('实时虚拟角度变化');
axis equal;
xlim([-1.5 1.5]);
ylim([-1.5 1.5]);
hold on;
plot([-1.2 1.2], [0 0], 'k--', 'Parent', axesPointer);  % 水平基准线
plot([0 0], [-1.2 1.2], 'k--', 'Parent', axesPointer);  % 垂直基准线
hPointer = plot(axesPointer, 0, 0, 'r', 'LineWidth', 3); % 指针初始

% 电机控制
hMotorControl = uicontrol('Style', 'pushbutton', 'Position', [960 450 140 40], ...
                'String', '电机控制', 'FontSize', 12, 'Callback', @motorControlCallback, 'Enable', 'off');

% 创建电机角度跟随曲线绘图区域
axesMotorResponse = axes('Parent', fig, 'Position', [0.05 0.18 0.4 0.3]);
title('电机实际角度响应曲线');
xlabel('时间 (s)');
ylabel('角度 (°)');
grid on;
hold on;

% 创建电机实时角度变化绘图区域
axesMotorPointer = axes('Parent', fig, 'Position', [0.4 0.18 0.5 0.3]);
title('电机实时角度变化');
```

```
axis equal;
xlim([-1.5 1.5]);
ylim([-1.5 1.5]);
hold on;
plot([-1.2 1.2], [0 0], 'k--', 'Parent', axesMotorPointer);  % 水平基准线
plot([0 0], [-1.2 1.2], 'k--', 'Parent', axesMotorPointer);  % 垂直基准线
hMotorPointer = plot(axesMotorPointer, 0, 0, 'r', 'LineWidth', 3); % 指针初始

% 电机实时角度显示
uicontrol('Style', 'text', 'Position', [940 390 150 30], 'String', '电机实时角度', 'F
ontWeight', 'bold', 'FontSize', 12);
hMotorRealTimeAngle = uicontrol('Style', 'text', 'Position', [970 360 100 30], 'St
ring', '0.00 度', 'FontSize', 12);

% 电机性能指标显示
uicontrol('Style', 'text', 'Position', [940 330 150 30], 'String', '电机性能指标', 'F
ontWeight', 'bold', 'FontSize', 12);
hMotorOvershoot = uicontrol('Style', 'text', 'Position', [920 300 130 30], 'String',
 '超调量: ');
hMotorSettlingTime = uicontrol('Style', 'text', 'Position', [920 270 130 30], 'Strin
g', '调节时间: ');
hMotorSteadyError = uicontrol('Style', 'text', 'Position', [920 240 130 30], 'Strin
g', '稳态误差: ');

% 存储控件句柄
handles = struct('hKp', hKp, 'hKi', hKi, 'hKd', hKd, ...
                 'hTargetAngle', hTargetAngle, ...
                 'hRunTime', hRunTime, ...
                 'axesResponse', axesResponse, ...
                 'axesPointer', axesPointer, ...
                 'hPointer', hPointer, ...
                 'hOvershoot', hOvershoot, ...
                 'hSettlingTime', hSettlingTime, ...
                 'hSteadyError', hSteadyError, ...
                 'hRealTimeAngle', hRealTimeAngle, ...
                 'hStartSim', hStartSim, ...
                 'hMotorControl', hMotorControl, ...
                 'axesMotorResponse', axesMotorResponse, ...
                 'axesMotorPointer', axesMotorPointer, ...
                 'hMotorPointer', hMotorPointer, ...
                 'hMotorRealTimeAngle', hMotorRealTimeAngle, ...
                 'hMotorOvershoot', hMotorOvershoot, ...
                 'hMotorSettlingTime', hMotorSettlingTime, ...
```

'hMotorSteadyError', hMotorSteadyError);
guidata(fig, handles);

然后定义虚拟仿真回调函数 *startSimulation* 和电机控制回调函数 *motorControlCallback*，分别实现虚拟仿真和电机控制功能。

1. 当用户点击虚拟仿真按钮时，会调用 *startSimulation* 函数进行虚拟仿真，计算并显示虚拟响应曲线和性能指标；
2. 当用户点击电机控制按钮时，会调用 *motorControlCallback* 函数让**硬件跟随虚拟曲线转动**，并显示电机的实际响应曲线和性能指标。

通过以上步骤，我们就可以实现一个完整的 GUI 程序，方便用户进行 Qube 硬件的 PID 控制实验。