浙江大學
ZHEJIANG UNIVERSITY

智能控制技术
实验报告

实验名称	HW01 专家控制
实验地点	教 7 304
姓　名	PhilFan
学　号	19260817
实验日期	January 10, 2025
指导老师	刘山

Contents

1 实验目的和要求

如图所示为车载倒立摆系统，一辆小车在水平轨道上移动，小车上有一个可绕固定点转动的倒立摆。控制小车在水平方向的移动可使摆杆维持直立不倒，这和手掌移动可使直立木棒不倒的现象类似。

忽略车轮与地面的摩擦力等阻力，可推导出车载倒立摆的动力学方程如下：

$$\begin{cases} (M+m)\ddot{x} + ml(\ddot{\theta}\cos\theta + ml\dot{\theta}^2\sin\theta) = F \\ ml^2\ddot{\theta} + ml\ddot{x}\cos\theta - mgl\sin\theta = 0 \end{cases} \tag{1}$$

其中的参数如表所示：

增量型离散 PID 控制算法如下：

$$F(k) = F(k-1) + K\left[K_p\Delta\theta(k) + \frac{T}{T_i}\theta(k) + \frac{T}{T_d}(\Delta\theta(k) - \Delta\theta(k-1))\right]$$

其中 T 为采样时间，$\Delta\theta(k) = \theta(k) - \theta(k-1)$

若 $F_m = 25$，取 $T = 0.0001s$，$K_p = 200$，$K_i = 3$，$K_d = 10$

设计 $0 < \theta_1 < \theta_2 < \theta_m$，$0 < K_s < 1 < K_b$

在离散 PID 控制基础上，采用专家 PID 控制方案，规则如下：

- 1. 若 $|\theta(k)| \geq \theta_m$ 时，$F(k) = \mathrm{sgn}(\theta)F_m$
- 2. 若 $\theta_2 \leq |\theta(k)| < \theta_m$ 时，
  - 若 $\theta(k)\Delta\theta(k) > 0$ 时，$K = K_b$
  - 若 $\theta(k)\Delta\theta(k) < 0$ 时，
    - 若 $\Delta\theta(k)\Delta\theta(k-1) > 0$ 时，$K = 1$
    - 若 $\Delta\theta(k)\Delta\theta(k-1) < 0$ 时，$K = K_b$
- 3. 若 $\theta_1 \leq |\theta(k)| < \theta_2$ 时，
  - 若 $\theta(k)\Delta\theta(k) > 0$ 时，$K = 1$
  - 若 $\theta(k)\Delta\theta(k) < 0$ 时，
    - 若 $\Delta\theta(k)\Delta\theta(k-1) > 0$ 时，$K = K_s$
    - 若 $\Delta\theta(k)\Delta\theta(k-1) < 0$ 时，$K = 1$
- 4. 若 $|\theta(k)| < \theta_1$ 时，$K = 1$

若小车和摆杆静止，摆杆与垂直向上方向的初始夹角 $\theta(0) = \frac{\pi}{4}$ rad，请：

Question.1. 给出上述专家 PID 控制方案的合适参数 $\theta_1, \theta_2, \theta_m$ 和 K_s, K_b，通过调节 F 使倒立摆的摆杆夹角 θ 恢复并维持在期望值（$\theta_d = 0$），在 matlab 中进行仿真，给出位移 x、夹角 θ 和水平力 F 的变化曲线，并比较专家 PID 控制与常规 PID 控制的结果（可尝试参数 $\theta_1 = 0.1, \theta_2 = 0.3, \theta_m = 0.5$ 和 $K_s = 1, K_b = 1.3$）。

Question.2. 针对不同的初始夹角 $\theta(0)$，给出专家 PID 控制的结果。（可能需要调整相关参数 $\theta_1, \theta_2, \theta_m$ 和 K_s, K_b）

2 问题分析

经过搜索和阅读题目，我认为本题可以分三个步骤来完成

(1). **建立小车倒立摆的物理模型**

(2). **建立普通 PID 和专家 PID 的控制器**

(3). **优化参数**

对于第一个步骤，由于普通的状态空间模型不能表达非线性模型，所以这里我们采用了 S-function 进行表达

普通 PID 使用了 Matlab 中自带的 PID 模块。

专家 PID 则使用题干中的控制策略，使用 S-function 构造一个输入是 θ，状态变量是 $[\theta, \frac{d\theta}{dt}, \theta_{last}, error_{sum}, error_{last}]$, 输出变量是 F 的控制器

对于调参来讲，我们先使用题目中推荐的参数进行测试，开始使用了观察法，比较普通 PID 和专家 PID 的控制效果；之后我参考机器学习中的损失函数，设计了基于 MSE 和最小超调 $theta$ 值的 $cost\ funtion$，并设计了 `critic` 评分器，用来比较 PID 和专家 PID 的实验结果.

同时，我的代码结构也比较规范，增加了详细的注释和说明，并添加了版本管理系统。

3 算法设计

3.1 S-function 学习

S-function 模块位于 Simulink/User-Defined Functions 模块库中，是使 S-function 图形化的模板工具，用于为 S-function 创建一个定值的对话框和图标。

- **S-function name**：填入 S-function 的函数名称，这样就建立了 S-function 模块与 M 文件形式的 S-function 之间的对应关系；
- **S-function parameters**：填入 S-function 需要输入的外部参数的名称，如果有多个变量，则变量中间用逗号隔开，如 a，b，c；
- **S-function modules**：仅当 S-function 是用 C 语言编写并用 MEX 工具编译的 C-MEX 文件时，才需要填写该参数；

直接馈通

如果输出函数（`mdlOutputs` 或 `flag==3`）是输入 `u` 的函数，即，如果输入 `u` 在 `mdlOutputs` 中被访问，则存在直接馈通。例如：

$$y = k \cdot u$$

采样时间与偏移量

采样时间是按照固定格式成对指定的：`[采样时间偏移时间]`。

采样时间表示	意义
`[0 0]`	连续采样时间
`[-1 0]`	继承 S-function 输入信号或父层模型的采样时间
`[0.5 0.1]`	离散采样时间，从 0.1s 开始每 0.5s 采样一次

Table 1: 采样时间与偏移量

函数分析

S-function 包括主函数和 6 个功能子函数，包括 `mdlInitializeSizes`（初始化）、`mdlDerivatives`（连续状态微分）、`mdlUpdate`（离散状态更新）、`mdlOutputs`（模块输出）、`mdlGetTimeOfNextVarHit`（计算下次采样时刻）和 `mdlTerminate`（仿真结束）。

在 S-function 仿真过程中，利用 `switch-case` 语句，根据不同阶段对应的 `flag` 值（仿真流程标志向量）来调用 S-function 的不同子函数，以完成对 S-function 模块仿真流程的控制。

3.2 Mask

如果我们不想每次修改 S-function 的参数都要打开 S-function 的编辑窗口，我们可以使用 Mask 功能。

第一步：增加 Mask

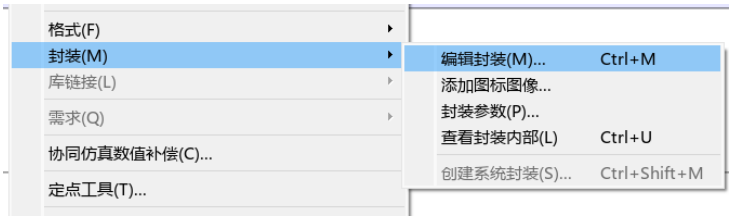

Figure 1: 增加 Mask

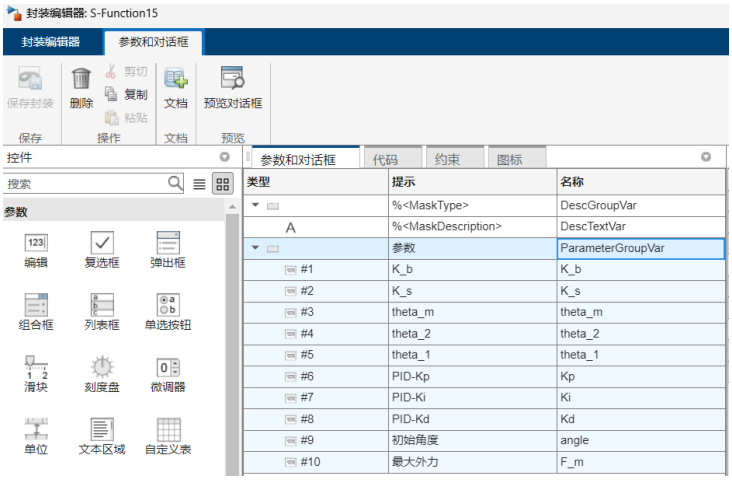

Figure 2: 点击添加封装

提示框可以随便写，但是名称需要和代码中的变量名称对齐。

第二步：在 S-function 初始化当中加入 Mask 参数

```
1  function [sys,x0,str,ts,simStateCompliance] = expert_control(t,x,u,flag,K
       _b,K_s,theta_m,theta_2,theta_1,Kp,Ki,Kd,angle,F_m) %这里需要加上需要的
       参数
2      switch flag
3          case 0
4              [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(angle); %
                   注意这里要写上需要的参数
5          case 1
6              sys=mdlDerivatives(t,x,u);
7          case 2
8              sys=mdlUpdate(t,x,u,K_b,K_s,theta_m,theta_2,theta_1,Kp,Ki,Kd,F_
                   m); % 注意这里要写上需要的参数
```

```
        case 3
            sys=mdlOutputs(t,x,u);
        case 4
            sys=mdlGetTimeOfNextVarHit(t,x,u);
        case 9
            sys=mdlTerminate(t,x,u);
        otherwise
            DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
    end

%% 子函数定义部分
function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(angle) %
    注意这里要写上需要的参数
```

第三步：在 s-function 的模块参数中，添加需要预置的参数

在参数这一行把需要的参数填进去，按照顺序来

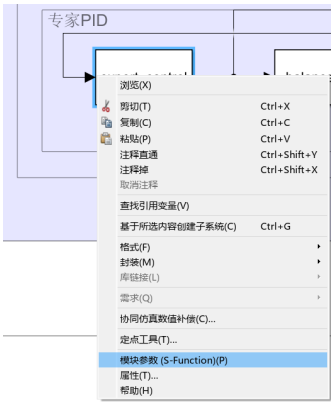

Figure 3: 添加需要预置的参数

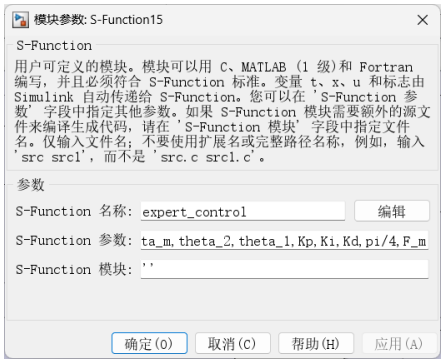

Figure 4: 参数设置

3.3 普通 PID 建模

对于小车，我们使用 s-function 进行建模，这里使用 $x, \dot{x}, \theta, \dot{\theta}$ 作为状态变量。

核心的 S-function 函数为：

```
function sys=mdlDerivatives(t,x,u)
    % 系统参数定义
    m = 0.5;                        % 摆杆质量(kg)
    M = 1;                          % 小车质量(kg)
    l = 0.5;                        % 摆杆半长(m)
    g = 9.8;                        % 重力加速度(m/s^2)
    % 计算状态导数
    dx1 = x(2);                     % 小车位置的导数
    dx3 = x(4);                     % 摆杆角度的导数
    dx2 = (u - m^2*l^2*x(4)^2*sin(x(3)) - m*g*sin(x(3))*cos(x(3))) / (M +
        m*sin(x(3))^2); % 小车速度的导数
    dx4 = ( m*g*l*sin(x(3)) - m*l*cos(x(3))*dx2) / (m*l^2); % 摆杆角速度的
        导数，注意这里可以使用已经计算过的简化表达
    sys = [dx1; dx2; dx3; dx4];   % 返回导数向量
```

对于 PID 算法，我直接采用了 Simulink 中自带的 PID 模块使用。需要设计的参数是，采样时间 $t = 0.0001$，$K_P = 200, K_i = 0.001, K_d = 10$。得到了如图7,8,9的曲线。可以发现，只采用单环 PID 控制，θ 收敛，但是 x 发散，这和朴素的理解也是相符的。

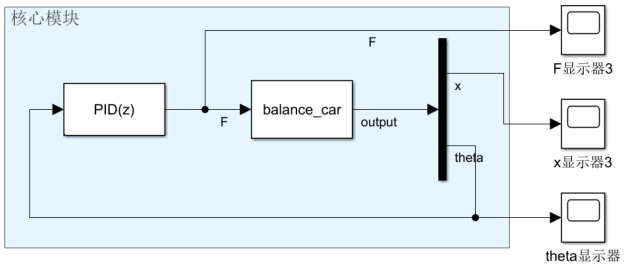

Figure 5: 核心模块的结构

Figure 6: 参数设计

Figure 7: F 的图像

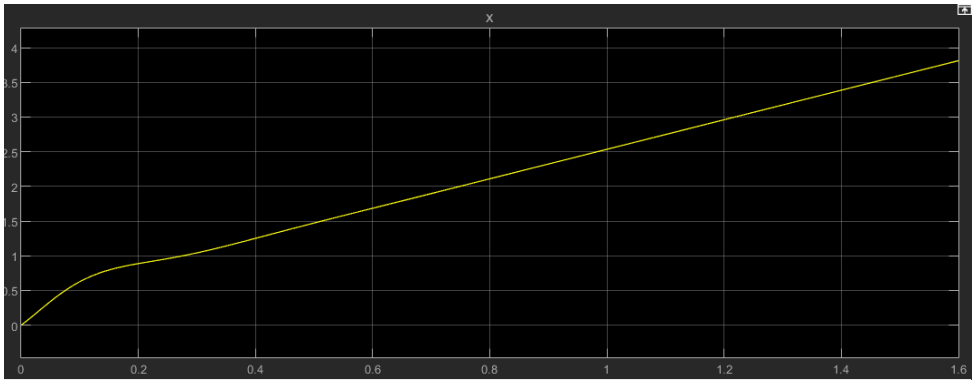

Figure 8: x 的图像

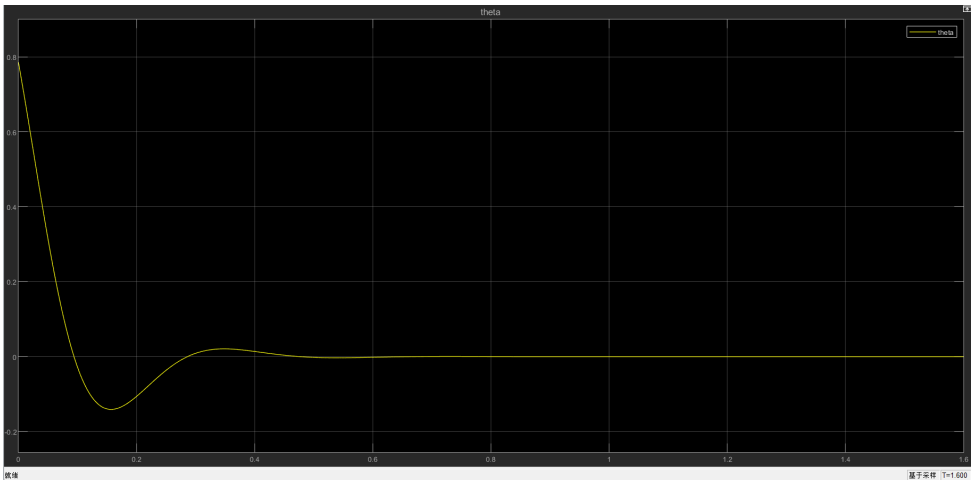

Figure 9: θ 的图像

3.4 专家 PID 策略分析

专家 PID 控制策略是一种基于规则的 PID 控制方法，它根据被控对象的状态和变化趋势来动态调整 PID 控制器的参数，以达到更好的控制效果。具体来说，专家 PID 控制策略包括以下几个规则：

(1). **大角度控制规则**：

- 当角度绝对值 $|\theta(k)|$ 大于或等于最大允许角度 θ_m 时，控制器输出 $F(k)$ 为最大控制力 F_m，并且方向与角度 $\theta(k)$ 的符号相同。这种规则确保了在角度过大时，控制器能够迅速采取强有力的措施来纠正偏差。

(2). **中角度控制规则**：

- 当角度绝对值 $|\theta(k)|$ 在 θ_2 和 θ_m 之间时，控制器根据角度的变化趋势来调整增益 K：
  - 如果角度 $\theta(k)$ 和其变化率 $\Delta\theta(k)$ 同号（即角度在增加且变化率正，或者角度在减小且变化率负），则增益 K 为基本增益 K_b。
  - 如果角度 $\theta(k)$ 和其变化率 $\Delta\theta(k)$ 异号，则进一步根据变化率的符号变化来调整增益：
    - 如果当前变化率 $\Delta\theta(k)$ 和前一时刻的变化率 $\Delta\theta(k-1)$ 同号，则增益 K 为 1。
    - 如果当前变化率 $\Delta\theta(k)$ 和前一时刻的变化率 $\Delta\theta(k-1)$ 异号，则增益 K 为基本增益 K_b。

(3). **小角度控制规则**：

- 当角度绝对值 $|\theta(k)|$ 在 θ_1 和 θ_2 之间时，控制器同样根据角度的变化趋势来调整增益 K：
  - 如果角度 $\theta(k)$ 和其变化率 $\Delta\theta(k)$ 同号，则增益 K 为 1。
  - 如果角度 $\theta(k)$ 和其变化率 $\Delta\theta(k)$ 异号，则进一步根据变化率的符号变化来调整增益：
    - 如果当前变化率 $\Delta\theta(k)$ 和前一时刻的变化率 $\Delta\theta(k-1)$ 同号，则增益 K 为阻尼增益 K_s。
    - 如果当前变化率 $\Delta\theta(k)$ 和前一时刻的变化率 $\Delta\theta(k-1)$ 异号，则增益 K 为 1。

(4). **小角度控制规则**：

- 当角度绝对值 $|\theta(k)|$ 小于 θ_1 时，增益 K 为 1。这种规则适用于角度接近目标值时，采用较小的增益以避免过度调整。

根据这些规则，我设计了我的专家控制 S-function，核心代码如下

Listing 1: 专家控制核心代码

```
% 专家控制规则
if abs(theta_k) >= theta_m
    % 规则1: 角度大于等于theta_m,施加最大外力
    F = sign(theta_k) * F_m;
elseif abs(theta_k) >= theta_2
    % 规则2: theta_2到theta_m区间的控制
    if theta_k * delta_theta_k > 0
        K = K_b;
    elseif theta_k * delta_theta_k < 0
        if delta_theta_k * delta_theta_k_1 > 0
            K = 1;
        elseif delta_theta_k * delta_theta_k_1 < 0
            K = K_b;
        end
    end
elseif abs(theta_k) >= theta_1
    % 规则3: theta_1到theta_2区间的控制
    if theta_k * delta_theta_k > 0
        K = 1;
    else
        if delta_theta_k * delta_theta_k_1 > 0
            K = K_s;
        elseif delta_theta_k * delta_theta_k_1 < 0
            K = 1;
        end
    end
else
    K = 1;
end

% 计算控制力
F = F + K * (Kp * delta_theta_k + (T / T_i) * theta_k + (T_d / T) * (
    delta_theta_k - delta_theta_k_1));
```

3.5 critic 评分器设计

为了更好的量化普通 PID 和专家 PID 的调整结果，参考在《机器学习》课程中的思想，我设计了一个 Critic 评价函数，采用 MSE($\sum u^2$) 和最大化最小 $\theta(max\ \theta_{min})$ 的方式。

为了权衡这两种不同的评价指标，我设计了一个 cost 的计算公式：

$$cost = \alpha \cdot (ref - u)^2 + \beta \cdot abs(\theta_{min}) \tag{2}$$

在这个 critic 评价体系下，我们的参数调整就变成了一个优化问题：

$$cost^* = \underset{u \in U}{arg\ min} \left[\alpha \cdot (ref - u)^2 + \beta \cdot abs(\theta_{min})\right] \tag{3}$$

当然，这种评价方式带有了一定的归纳偏置，所以这个 critic 函数还有很多优化的空间。

对于 critic 评价器，我也是使用 S-function 实现的，核心代码如下：

Listing 2: critic 核心代码

```
function sys = mdlUpdate(t, x, u)
    % 状态更新回调子函数
    % 更新最小值theta_min，保存当前theta_k与历史最小值比较
    theta_k = u; % 当前的theta值
    theta_min = min(x(1), theta_k); % 更新最小值
    cost_function = x(2) + u * u ; % 使用MSE计算成本函数
    sys = [theta_min, cost_function];
end
```

4 实验结果表现与分析

根据上面的三个模块，我实现了 PID 和专家 PID 系统，并集成到了一个子系统中进行比较。

4.1 $\theta = \frac{\pi}{4}$ 时

首先来看最基础的任务

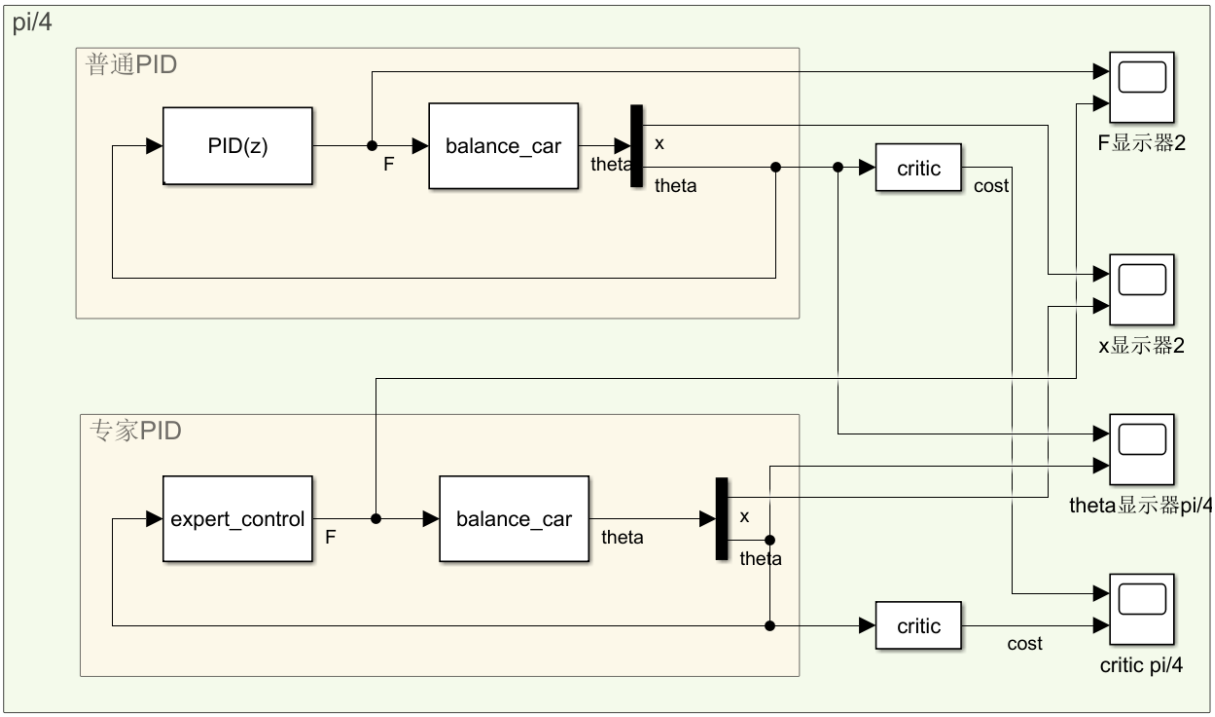

Figure 10: $\theta = \frac{\pi}{4}$

我设置的参数如下图所示。在这样的参数下，普通 PID 和专家 PID 的 cost 值分别为 16.0421 和 10.2557，在这个角度下，专家 PID 的效果是更为优秀的。

Figure 11: 我的参数设置

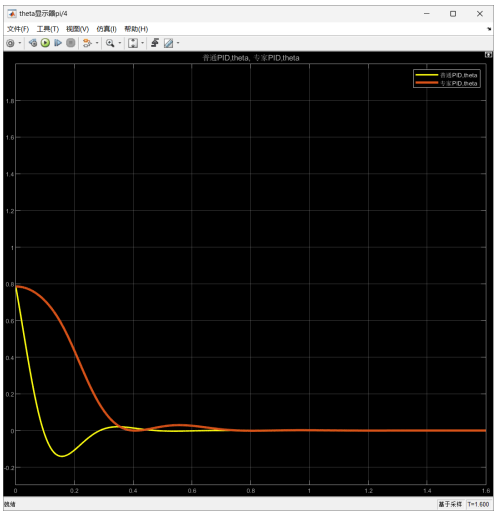

Figure 12: theta 对比图

Figure 13: F 对比图

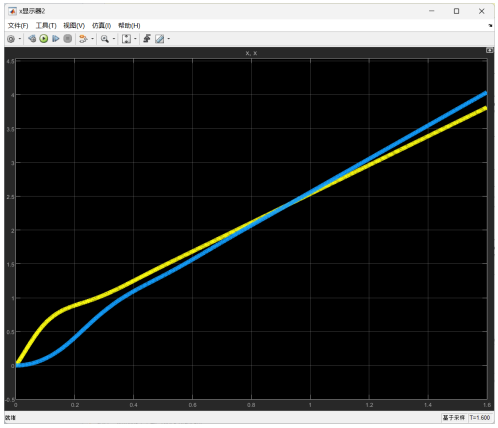

Figure 14: x 对比图

4.2 $\theta = \frac{\pi}{3}$ 时

在 $\theta = \frac{\pi}{3}$ 时，普通 PID 可以直接稳定，在调整专家 PID 参数后，专家 PID 也可以稳定。

并且在图15所示的参属下，普通 PID 的 cost 为 12.4184，专家 PID 的 cost 为 12.1391。可以达到给定目标。

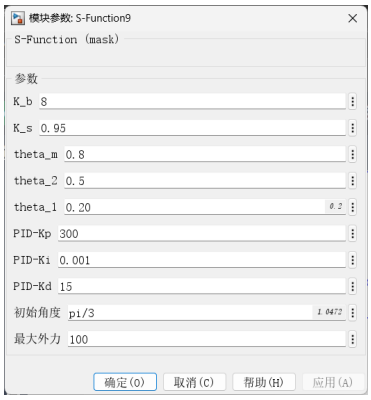

Figure 15: 参数设置

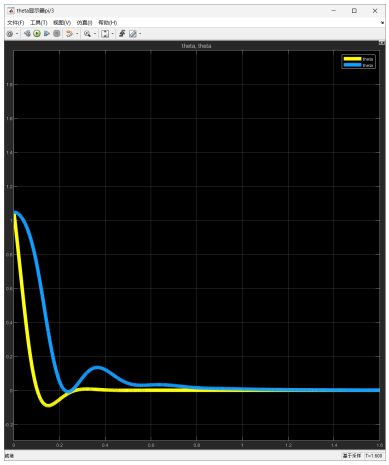

Figure 16: θ 的对比

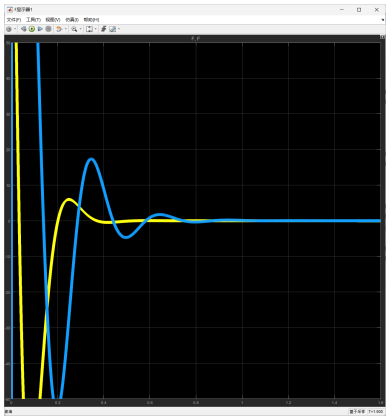

Figure 17: F 对比图

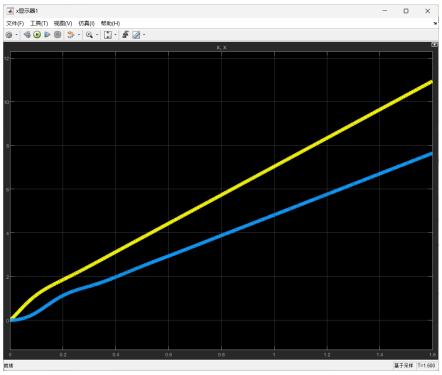

Figure 18: x 的对比

4.3 $\theta = \frac{\pi}{6}$ 时

对于 $\theta = \frac{\pi}{6}$ 的情况，我们应该对应地减小相应参数的大小

经过调整以后，我的参数如下图所示，在这个参数设置下，普通 PID 的 cost 为 6.3159，专家 PID 为 3.3014

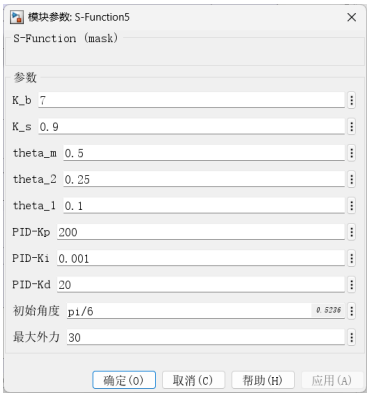

Figure 19: 参数设置

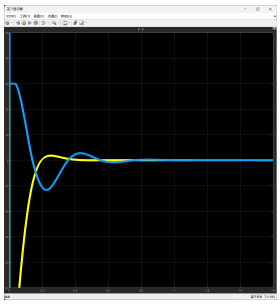

Figure 20: F 结果

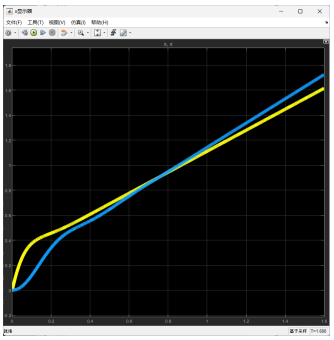

Figure 21: x 对比

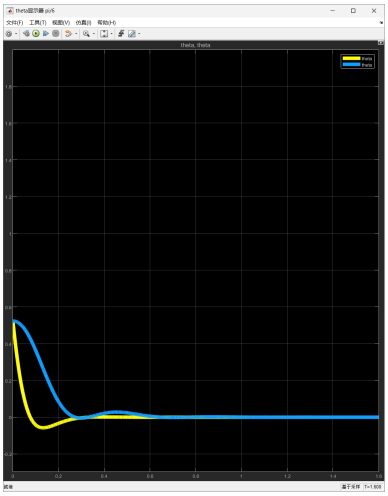

Figure 22: θ 结果, 蓝色为专家控制

5 实验探究与可优化方向

总体来说，这次作业让我体会到了专家控制的整体思路和过程。在这次作业当中，我学习了 s-function 与 mask 的使用方法。并使用了一种 critic 函数来定量评价指标。

我认为还可以在以下几个方面做出优化：

- 增加根据 critic 函数自动优化求解的方法
- 增加 3D animation 界面，可视化呈现
- 自动化地调节 PID 参数，或者通过可视化地调节；目前调节参数只能调完一次重新运行，效率比较低
- 完善系统文档

A 附录 1：文档结构

```
.
    expert_control.prj  # 项目文件
    exp.slx             # 专家控制系统模型
    expert_control.m    # 专家控制器主程序
    critic.m           # 系统评价器
    balance_car.m      # 倒立摆小车模型
    README.md          # 项目说明文档
```

B 附录 2：源程序代码

B.1 balance_car.m

C 附录 3：版本更新记录

```
% Date: 2024-11-30
% Author: PhilFan

% 倒立摆小车物理系统的S-function文件
% 输入参数:
%   t - 当前时间
%   x - 状态变量
%   u - 输入变量
%   flag - 仿真标志
%   angle - 初始角度
% 输出参数:
%   sys - 返回值
%   x0 - 初始状态
%   str - 保留参数
%   ts - 采样时间
%   simStateCompliance - 仿真状态


function [sys,x0,str,ts,simStateCompliance] = balance_car(t,x,u,flag,
    angle)
    switch flag,
```

```
        case 0,
            [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(angle);
        case 1,
            sys=mdlDerivatives(t,x,u);
        case 2,
            sys=mdlUpdate(t,x,u);
        case 3,
            sys=mdlOutputs(t,x,u);
        case 4,
            sys=mdlGetTimeOfNextVarHit(t,x,u);
        case 9,
            sys=mdlTerminate(t,x,u);
        otherwise
            DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag))
                ;
    end

%% 子函数定义部分

function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(angle)
    % 初始化函数
    % 功能：初始化系统的状态、输入输出维度等基本参数
    % 当flag=0时被调用

    sizes = simsizes;              % 生成sizes数据结构
    sizes.NumContStates = 4;       % 连续状态数量：4个
    sizes.NumDiscStates = 0;       % 离散状态数量：0个
    sizes.NumOutputs    = 2;       % 输出数量：2个
    sizes.NumInputs     = 1;       % 输入数量：1个
    sizes.DirFeedthrough = 0;      % 是否存在直接馈通：否
    sizes.NumSampleTimes = 1;      % 采样时间个数：1个

    sys = simsizes(sizes);         % 返回初始化信息
    x0  = [0 0 angle 0];           % 设置初始状态值
    str = [];                      % 保留参数置空
    ts  = [0 0];                   % 设置采样时间
    simStateCompliance = 'UnknownSimState';

```

```
58      % 状态变量说明:
59      % x(1) = x    - 小车位置
60      % x(2) = dx/dt - 小车速度
61      % x(3) =      - 摆杆角度
62      % x(4) = d /dt - 摆杆角速度
63
64  function sys=mdlDerivatives(t,x,u)
65      % 系统连续状态方程
66      % 功能: 计算系统状态导数
67      % 当flag=1时被调用
68
69      % 系统参数定义
70      m = 0.5;                        % 摆杆质量(kg)
71      M = 1;                          % 小车质量(kg)
72      l = 0.5;                        % 摆杆半长(m)
73      g = 9.8;                        % 重力加速度(m/s^2)
74
75      % 计算状态导数
76      dx1 = x(2);                     % 小车位置的导数
77      dx3 = x(4);                     % 摆杆角度的导数
78      dx2 = (u - m^2*l^2*x(4)^2*sin(x(3)) - m*g*sin(x(3))*cos(x(3))) / (M +
            m*sin(x(3))^2); % 小车速度的导数
79      dx4 = ( m*g*l*sin(x(3)) - m*l*cos(x(3))*dx2) / (m*l^2);
                                % 摆杆角速度的导数
80
81      sys = [dx1; dx2; dx3; dx4];   % 返回导数向量
82
83  function sys=mdlUpdate(t,x,u)
84      % 离散状态更新函数
85      % 功能: 更新系统的离散状态
86      % 当flag=2时被调用
87      sys = [];                       % 本系统无离散状态
88
89  function sys=mdlOutputs(t,x,u)
90      % 输出方程
91      % 功能: 计算系统输出
92      % 当flag=3时被调用
93      sys = [x(1);x(3)];              % 输出小车位置和摆杆角度
```

```
94
95  function sys=mdlGetTimeOfNextVarHit(t,x,u)
96      % 变采样时间计算函数
97      % 功能: 计算下一个采样时间点
98      % 当使用变采样时间时被调用
99      sampleTime = 1;                 % 设置采样间隔为1秒
100     sys = t + sampleTime;           % 计算下一个采样时间点
101
102 function sys=mdlTerminate(t,x,u)
103     % 终止函数
104     % 功能: 完成仿真结束时的必要工作
105     % 当flag=9时被调用
106     sys = [];                       % 清空系统
```

C.1 critic.m

```
1   % Date: 2024-12-03
2   % Author: PhilFan
3
4   % 评价网络定义
5   % 输入参数:
6   %   t - 当前时间
7   %   x - 状态变量 [theta_min, cost_function]
8   %   u - 输入变量 theta (摆杆角度)
9   %   flag - 仿真标志
10  % 输出参数:
11  %   sys - 返回值
12  %   x0 - 初始状态
13  %   str - 保留参数
14  %   ts - 采样时间
15  %   simStateCompliance - 仿真状态
16  % 功能说明:
17  %   该评价网络用于评估倒立摆系统的性能
18  %   通过最小化超调来优化控制效果
19  %   cost_function基于MSE进行计算
20
21
22  function [sys, x0, str, ts, simStateCompliance] = critic(t, x, u, flag)
```

```
% Critic S-Function
% 输入：theta，目标值theta_target（无实际意义），
% 输出：cost_function（基于最小theta的最大化）

switch flag
    case 0
        [sys, x0, str, ts, simStateCompliance] = mdlInitializeSizes();
    case 1
        sys = mdlDerivatives(t, x, u);
    case 2
        sys = mdlUpdate(t, x, u);
    case 3
        sys = mdlOutputs(t, x, u);
    case 4
        sys = mdlGetTimeOfNextVarHit(t, x, u);
    case 9
        sys = mdlTerminate(t, x, u);
    otherwise
        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag))
            ;
end

%% 子函数定义
function [sys, x0, str, ts, simStateCompliance] = mdlInitializeSizes()
    % 初始化回调子函数

    sizes = simsizes;
    sizes.NumContStates = 0; % 连续状态数
    sizes.NumDiscStates = 2; % 离散状态数：[theta_min, cost_function]
    sizes.NumOutputs = 1;   % 输出个数（成本函数）
    sizes.NumInputs = 1;    % 输入个数（theta）
    sizes.DirFeedthrough = 1; % 允许直馈通道
    sizes.NumSampleTimes = 1; % 采样时间数
    sys = simsizes(sizes);  % 返回sizes数据结构
    x0 = [Inf, 0];          % 初始状态：[theta_min, cost_function]
    str = [];               % 保留参数
    ts = [0 0];             % 采样时间
    simStateCompliance = 'UnknownSimState'; % 仿真状态合规性
```

```
60      end
61
62      function sys = mdlDerivatives(t, x, u)
63          % 导数回调子函数（不使用，空实现）
64          sys = [];
65      end
66
67      function sys = mdlUpdate(t, x, u)
68          % 状态更新回调子函数
69          % 更新最小值theta_min，保存当前theta_k与历史最小值比较
70          theta_k = u; % 当前的theta值
71          theta_min = min(x(1), theta_k); % 更新最小值
72          cost_function = x(2) + u * u ; % 使用MSE计算成本函数
73          sys = [theta_min, cost_function];
74      end
75
76      function sys = mdlOutputs(t, x, u)
77          % 输出回调子函数
78          % 输出成本函数
79          alpha = 100;
80          beta = 0.01;
81          cost = x(2);
82          min = x(1);
83          fi  = beta*cost + alpha * abs(min);
84          sys = [fi];  % 输出成本函数
85      end
86
87      function sys = mdlGetTimeOfNextVarHit(t, x, u)
88          % 计算下一个采样时间（定时采样）
89          sampleTime = 1; % 固定采样时间（1秒）
90          sys = t + sampleTime; % 下一次采样时间
91      end
92
93      function sys = mdlTerminate(t, x, u)
94          % 仿真结束时的回调（输出成本函数）
95          alpha = 100;
96          beta = 0.01;
97          %disp(['min: ', num2str(x(1))]);
```

```
98
99          cost = x(2);
100         min = x(1);
101         fi  = beta*cost + alpha * abs(min);
102         disp(['cost: ', num2str(fi)]);
103         sys = [];
104     end
105 end
```

C.2 expert_control.m

```
1   % Date: 2024-12-01
2   % Author: PhilFan
3
4   % 倒立摆专家控制系统的S-function文件
5   % 输入参数:
6   %   t - 当前时间
7   %   x - 状态变量 [theta, dtheta/dt, theta_last, error_sum, error_last]
8   %   u - 输入变量 theta (摆杆角度)
9   %   flag - 仿真标志
10  %   K_b - 基本控制器增益
11  %   K_s - 切换控制器增益
12  %   theta_m - 最大角度阈值
13  %   theta_2 - 第二角度阈值
14  %   theta_1 - 第一角度阈值
15  %   Kp - PID控制器比例增益
16  %   Ki - PID控制器积分增益
17  %   Kd - PID控制器微分增益
18  %   angle - 初始角度
19  %   F_m - 最大控制力
20  % 输出参数:
21  %   sys - 返回值
22  %   x0 - 初始状态
23  %   str - 保留参数
24  %   ts - 采样时间
25  %   simStateCompliance - 仿真状态
26  % 功能说明:
27  %   该专家控制系统根据摆杆角度的不同区域,
```

```
28  %   自适应切换不同的控制策略(PID控制和能量控制),
29  %   实现倒立摆的平衡控制
30
31
32  function [sys,x0,str,ts,simStateCompliance] = expert_control(t,x,u,flag,
        K_b,K_s,theta_m,theta_2,theta_1,Kp,Ki,Kd,angle,F_m)
33      % 主函数,包含四个输出:
34      % sys - 包含某个子函数返回的值
35      % x0 - 所有状态的初始化向量
36      % str - 保留参数,总是一个空矩阵
37      % ts - 返回系统采样时间
38      % simStateCompliance - 仿真状态合规性
39
40      switch flag
41          case 0
42              [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(angle);
43          case 1
44              sys=mdlDerivatives(t,x,u);
45          case 2
46              sys=mdlUpdate(t,x,u,K_b,K_s,theta_m,theta_2,theta_1,Kp,Ki,Kd,
                    F_m);
47          case 3
48              sys=mdlOutputs(t,x,u);
49          case 4
50              sys=mdlGetTimeOfNextVarHit(t,x,u);
51          case 9
52              sys=mdlTerminate(t,x,u);
53          otherwise
54              DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag))
                    ;
55      end
56
57  %% 子函数定义部分
58  function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(angle)
59      % 初始化回调子函数
60      % 提供状态、输入输出、采样时间数目和初始状态的值
61      % 初始化阶段,标志变量flag首先被置为0,S-function首次被调用时该子函数被调
            用
```

```
62
63      sizes = simsizes;               % 生成sizes数据结构
64      sizes.NumContStates = 0;        % 连续状态数
65      sizes.NumDiscStates = 5;        % 离散状态数
66      sizes.NumOutputs = 1;           % 输出个数
67      sizes.NumInputs = 1;            % 输入个数
68      sizes.DirFeedthrough = 1;       % 是否存在直馈通道
69      sizes.NumSampleTimes = 1;       % 采样时间个数
70
71      sys = simsizes(sizes);          % 返回size数据结构
72      x0 = [angle 0 angle 0 0];       % 设置初始状态
73      str = [];                       % 保留变量置空
74      ts = [0 0];                     % 设置采样时间
75      simStateCompliance = 'UnknownSimState';
76
77  function sys=mdlDerivatives(t,x,u)
78      sys = [];
79
80  function sys=mdlUpdate(t,x,u,K_b,K_s,theta_m,theta_2,theta_1,Kp,Ki,Kd,F_m
        )
81      % 状态更新回调子函数
82      % 给定t、x、u计算离散状态的更新
83
84      theta_k = u;
85      theta_k_1 = x(1);
86      delta_theta_k = theta_k - theta_k_1;
87      delta_theta_k_1 = x(1)-x(2);
88      F = x(5);
89
90      % PID控制参数
91      K = 1;
92      T = 0.0001;
93      T_i = 0.001;
94      T_d = 10;
95
96      % 专家控制规则
97      if abs(theta_k) >= theta_m
98          % 规则1: 角度大于等于theta_m,施加最大外力
```

```
99          F = sign(theta_k) * F_m;
100     elseif abs(theta_k) >= theta_2
101         % 规则2: theta_2到theta_m区间的控制
102         if theta_k * delta_theta_k > 0
103             K = K_b;
104         elseif theta_k * delta_theta_k < 0
105             if delta_theta_k * delta_theta_k_1 > 0
106                 K = 1;
107             elseif delta_theta_k * delta_theta_k_1 < 0
108                 K = K_b;
109             end
110         end
111     elseif abs(theta_k) >= theta_1
112         % 规则3: theta_1到theta_2区间的控制
113         if theta_k * delta_theta_k > 0
114             K = 1;
115         else
116             if delta_theta_k * delta_theta_k_1 > 0
117                 K = K_s;
118             elseif delta_theta_k * delta_theta_k_1 < 0
119                 K = 1;
120             end
121         end
122     else
123         K = 1;
124     end
125
126     % 计算控制力
127     F = F + K * (Kp * delta_theta_k + (T / T_i) * theta_k + (T_d / T) * (
        delta_theta_k - delta_theta_k_1));
128
129     % 更新系统状态
130     sys = [theta_k, theta_k_1, delta_theta_k, delta_theta_k_1, F];
131
132 function sys=mdlOutputs(t,x,u)
133     % 计算输出回调函数
134     % 输出当前控制力F
135     sys = [x(5)];
```

```
136
137 function sys=mdlGetTimeOfNextVarHit(t,x,u)
138     % 计算下一个采样时间
139     % 仅在系统是变采样时间系统时调用
140     sampleTime = 1;
141     sys = t + sampleTime;
142
143 function sys=mdlTerminate(t,x,u)
144     % 仿真结束时的回调函数
145     sys = [];
```

D 附录 3：版本更新记录

版本号	更新日期	更新内容	备注
v1.0.0	**2024-11**	• 实现基础倒立摆控制系统 • 包含专家控制器 (expert_control.m)	初始版本
v1.1.0	**2024-11**	增加评价器功能 (critic.m)，用于判断模型的好坏 改进控制器性能参数	性能优化
v1.2.0	**2024-12-01**	优化评价器评估方法 提升系统稳定性，增加对比	参数调节
v1.2.1	**2024-12-03**	优化代码逻辑，增加注释和说明文档	Docs 优化