专家控制

Author: 章翰宇
ID: 3220104133

Abstract

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。比较离散 PID 控制器以及专家控制方法，针对不同的初始夹角，给出专家 PID 控制的结果。

Keywords: 智能控制，专家控制

如有问题，请访问

模型建立

如下图，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（分别为大车水平分量，小球水平和竖直分量）：

$$\begin{cases} F - N\cos(\theta) = M\ddot{x} \\ N\sin\theta = m\left(\ddot{x} + l\left(\cos\theta\ddot{\theta} - \sin\theta\dot{\theta}^2\right)\right) \\ N\cos\theta - mg = ml\left(-\cos\theta\dot{\theta}^2 - \sin\theta\ddot{\theta}\right) \end{cases}$$

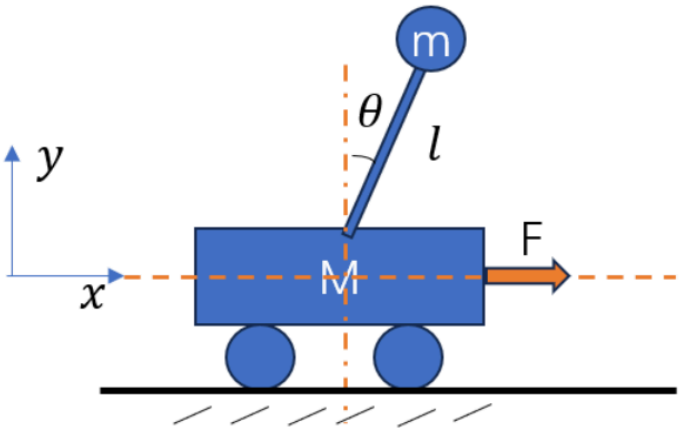


Figure 1: 倒立摆模型

消去变量N，得到：

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

为了便于画方块图，将x的二阶导数移到左边：

$$\begin{cases} \ddot{x} = \left(\frac{M+m}{ml\cos\theta} - \frac{1}{l}\cos\theta\right)^{-1}\left(\tan\theta\dot{\theta}^2 + \frac{F}{ml\cos\theta} - \frac{g}{l}\sin\theta\right) \\ \ddot{\theta} = \frac{g}{l}\sin\theta - \frac{1}{l}\cos\theta\ddot{x} \end{cases}$$

下面是方块图：

Simulink

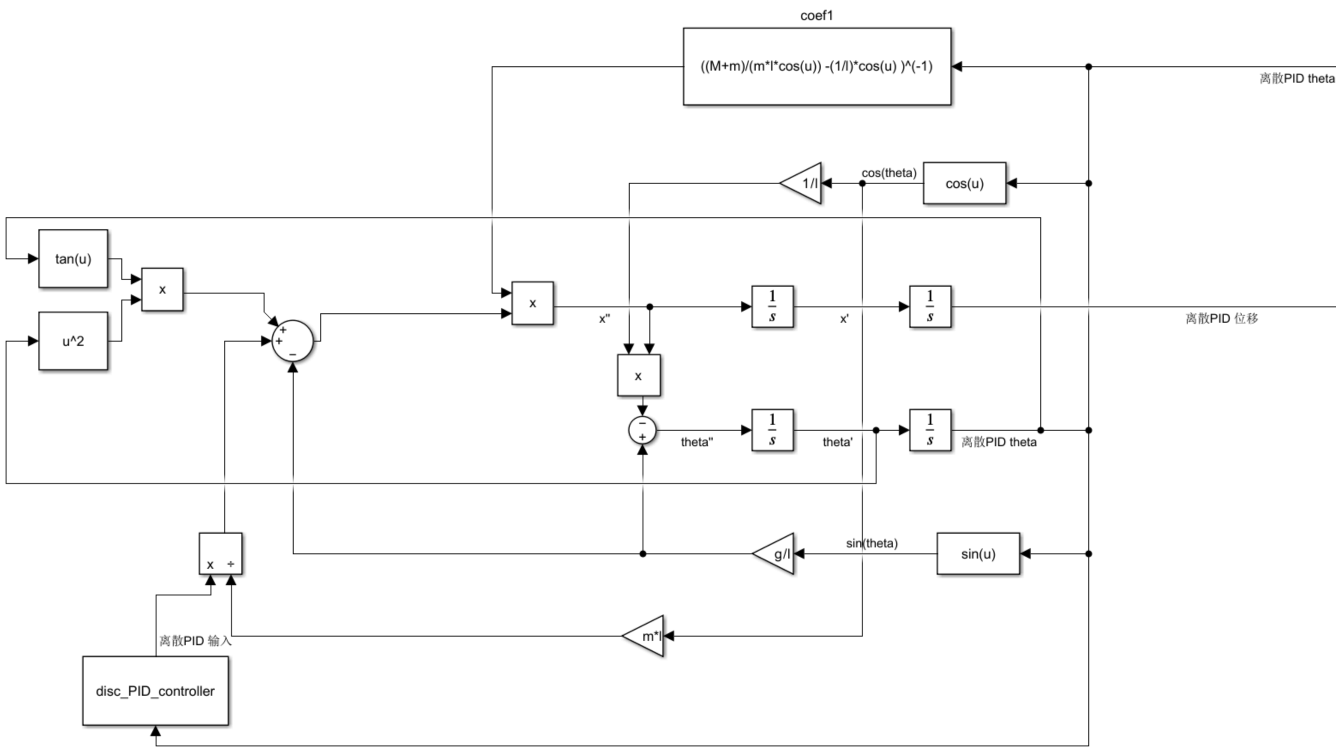


Figure 2: Simulink 模型图（离散 PID 版本）

框图解释：核心是中间两路$\ddot{\theta} \to \dot{\theta} \to \theta$以及$\ddot{x} \to \dot{x} \to x$。

其中，$\ddot{\theta}$的值根据$\frac{g}{l}\sin\theta - \frac{1}{l}\cos\theta\ddot{x}$给出，因此是两部分的相减，下面一部分，来自$\theta$，作用sin后再乘以 Gain 得到；上面一部分来自cos作用后的θ与$\ddot{x}$的乘积。

$\ddot{x}$比较复杂，表达式为$\left(\frac{M+m}{ml\cos\theta} - \frac{1}{l}\cos\theta\right)^{-1}\left(\tan\theta\dot{\theta}^2 + \frac{F}{ml\cos\theta} - \frac{g}{l}\sin\theta\right)$，因此，框图最上方的 coef 块即计算了前面这个很大的系数，而框图左侧的三个部分相加减，即$\tan\theta\dot{\theta}^2$, $\frac{F}{ml\cos\theta}$, $\frac{g}{l}\sin\theta$这三部分。

disc_PID_controller 部分使用 S-Function 实现。

控制器设计

Warning

我使用的均是 Level 2 的 S-Function

以下为离散 PID 的实现代码主要部分，设置了三个内置变量：

```
function DoPostPropSetup(block)

  %% Setup Dwork
  block.NumDworks = 3;

  % 一个内置更新量为 F(k)
```

```
  block.Dwork(1).Name = 'F';
  block.Dwork(1).Dimensions      = 1;
  block.Dwork(1).DatatypeID      = 0;
  block.Dwork(1).Complexity      = 'Real';
  block.Dwork(1).UsedAsDiscState = true;

  % 一个内置更新量为 theta(k)
  block.Dwork(2).Name = 'theta';
  block.Dwork(2).Dimensions      = 1;
  block.Dwork(2).DatatypeID      = 0;
  block.Dwork(2).Complexity      = 'Real';
  block.Dwork(2).UsedAsDiscState = true;

  % 一个内置更新量为 Delta_heta(k)
  block.Dwork(3).Name = 'Delta_theta';
  block.Dwork(3).Dimensions      = 1;
  block.Dwork(3).DatatypeID      = 0;
  block.Dwork(3).Complexity      = 'Real';
  block.Dwork(3).UsedAsDiscState = true;

  %block.DialogPrm(1).Data;

function Output(block)

  block.OutputPort(1).Data = block.Dwork(1).Data; % 输出即为 F(k)

function Update(block)

  % 更新目标：F(k)(即 block.Dwork(1).Data)，然后输出之
  % 更新步骤：
  % 首先，theta(k)输入(为 InputPort(1).Data)，将其与 theta(k-1) (Dwork 变量) 作差，得到
Delta_theta(k)
  Delta_theta_k = block.InputPort(1).Data-block.Dwork(2).Data;
  % 这样，Delta_Delta_theta 也可以算了：
  Delta_Delta_theta = Delta_theta_k - block.Dwork(3).Data;
  % 至此，组成增量 PID 的三个关键，即 theta,Delta_theta,Delta_Delta_theta 都有了

  % 增量 PID 的 F(k)更新公式：
  F_k = block.Dwork(1).Data + K*(K_p*Delta_theta_k + T/T_i*block.InputPort(1).Data + T_d/
T *Delta_Delta_theta);

  % 更新 Dwork 参数们：
  % 力的更新要考虑上下限：
  if F_k> F_m
    block.Dwork(1).Data = F_m;
  elseif F_k< -F_m
    block.Dwork(1).Data = -F_m;
  else
    block.Dwork(1).Data = F_k;
  end
```

```
block.Dwork(2).Data = block.InputPort(1).Data; % theta_k 参数就是输入本身
block.Dwork(3).Data = Delta_theta_k; % Delta_theta 参数更新
```

也可以使用专家控制方法，和离散 PID 唯一的区别是，增加了一些判断语句，分类讨论K的取值，但是上面的 PID 则是固定的K取值（我默认设置为 1）

```
%% 专家控制额外部分：
  % 若 theta_k 很大，则拉满输出力，不用管 PID 算法
  if abs(theta_k) >= theta_m
     block.Dwork(1).Data = sign(theta_k)* F_m;
  else
  % 若稍微比较大，则分两种情况讨论
     if ( abs(theta_k)>= theta_2 ) && ( abs(theta_k)< theta_m )

         % 1) 若乘积>0，则 K=K_b 再使用 PID
         if theta_k * Delta_theta_k >0
              K = K_b;

         % 2) 若乘积<0，则再进一步分类讨论
         else
              %     i) 若两个 Delta 之乘积为正，则 K=1
              if Delta_Delta_theta*Delta_theta_k > 0
                  K = 1;
              %     ii) 若两个 Delta 之乘积为负，则 K=K_b
              else
                  K = K_b;
              end
          end

      % 若较小，则分两种情况讨论
     elseif ( abs(theta_k)>= theta_1 ) && ( abs(theta_k)< theta_2 )

         % 1) 若乘积>0，则 K=1 再使用 PID
         if theta_k * Delta_theta_k >0
              K = 1;
         % 2) 若乘积<0，则再进一步分类讨论
         else
            %     i) 若两个 Delta 之乘积为正，则 K=K_s
            if Delta_Delta_theta*Delta_theta_k > 0
                K = K_s;
            %     ii) 若两个 Delta 之乘积为负，则 K=1
            else
                K = 1;
            end
         end

      % 若 theta_k 很小，则 K=1 再使用 PID
     else
         K = 1;
```

```
    end
     % 增量 PID 的 F(k)更新公式：
     F_k = block.Dwork(1).Data + K*(K_p*Delta_theta_k + T/T_i*block.InputPort(1).Data +
T_d/T *Delta_Delta_theta);

     % 更新 Dwork 参数们：
     % 最后，力的更新仍然要考虑上下限！
     if F_k> F_m
       block.Dwork(1).Data = F_m;
     elseif F_k< -F_m
       block.Dwork(1).Data = -F_m;
     else
       block.Dwork(1).Data = F_k;
     end
  end
  block.Dwork(2).Data = block.InputPort(1).Data; % theta_k 参数就是输入本身
  block.Dwork(3).Data = Delta_theta_k; % Delta_theta 参数更新
```

效果对比分析

准备测试，使用题中建议的参数列表：

```
%% 离散 PID 参数
m = 0.5; M = 1; l = 0.5; g = 9.8;
T = 0.0001; % sample period
K_p = 200; T_i = 0.001; K = 1;
T_d = 10; F_m =25; % max input
%% 专家控制部分参数
theta_1 = 0.1; theta_2 = 0.3;
theta_m = 0.5; K_s = 1; K_b = 1.3;
```

框图如下图 Figure 3，把离散 PID 系统控制的一个系统与专家控制改进的系统，其输出值汇总到一个 Scope 中进行对比，分析区别：

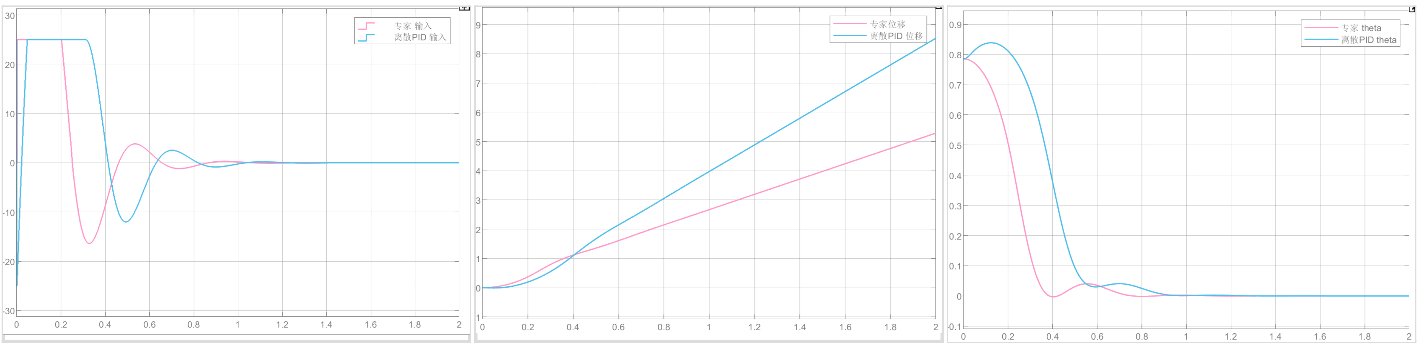


Table 1:
1. 输入 2. 小车位移 3. 倒立摆角度

可见，在该种参数下，两方法都可以使得倒立摆的摆角稳定在 0 度左右，并且调节时间都很快（1 秒以内）。但是，专家控制的倒立摆角度无超调，调节时间更短，而且使用的能耗更小（参见输入曲线），最后由于调节产生的小车速度也更小。因此，从快速性和稳定性来分析，专家控制更胜一筹。

Simulink

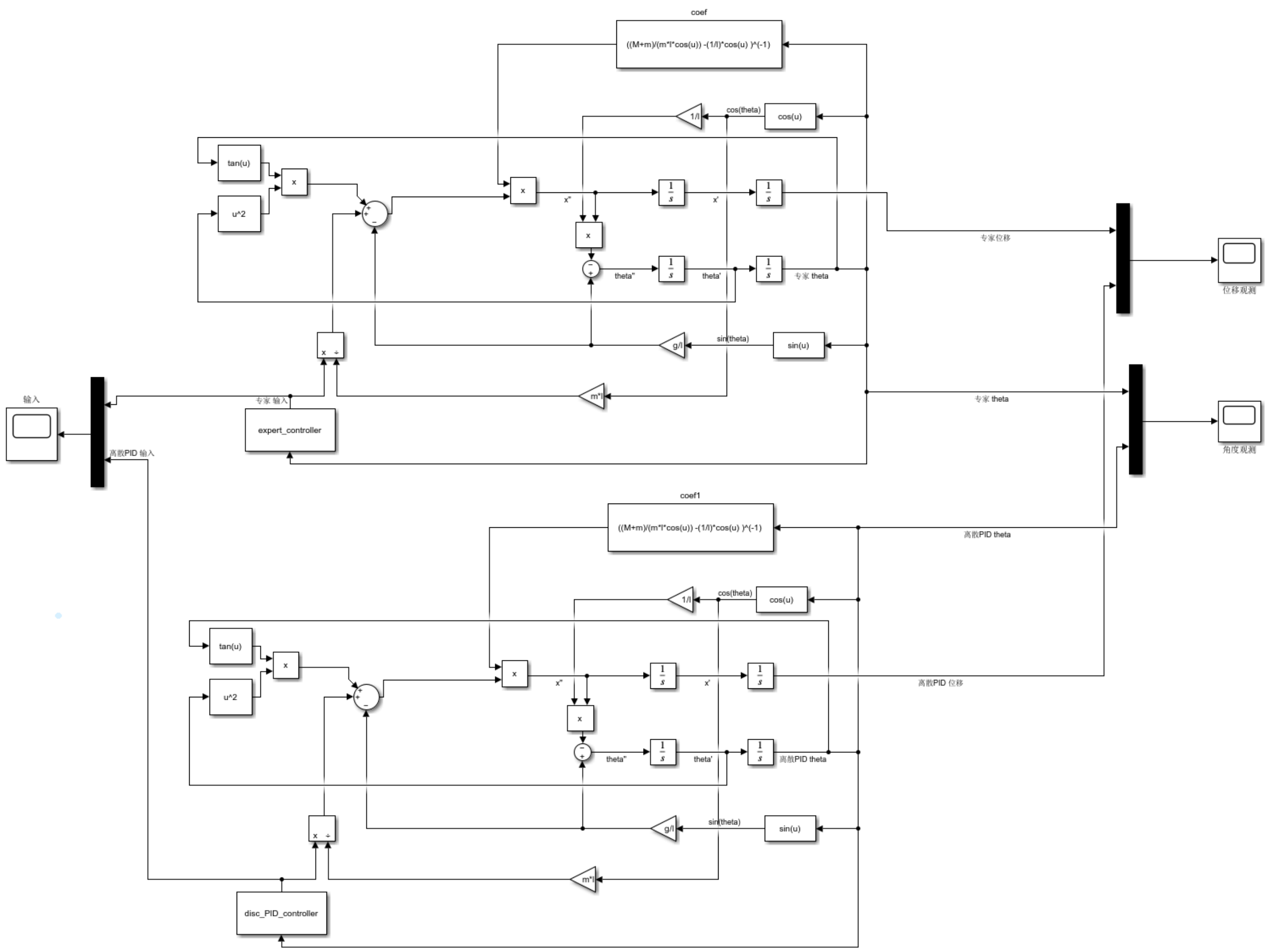


Figure 3: 对比框图

不同初始条件的专家控制

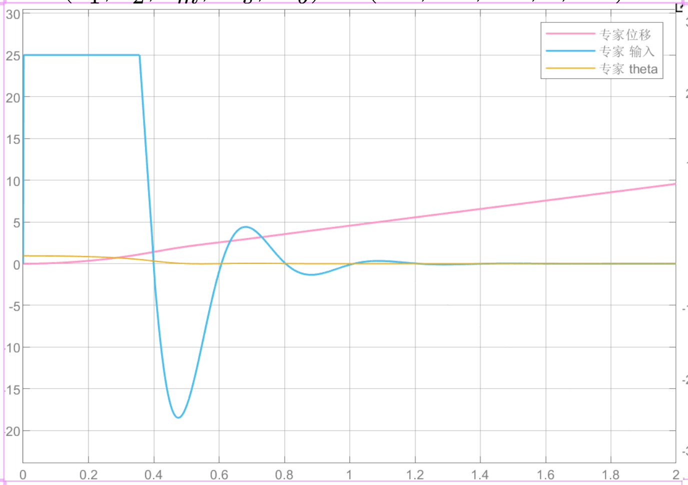
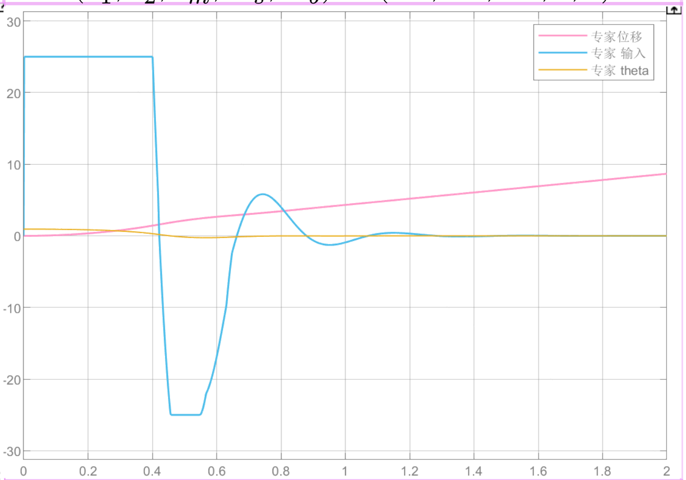
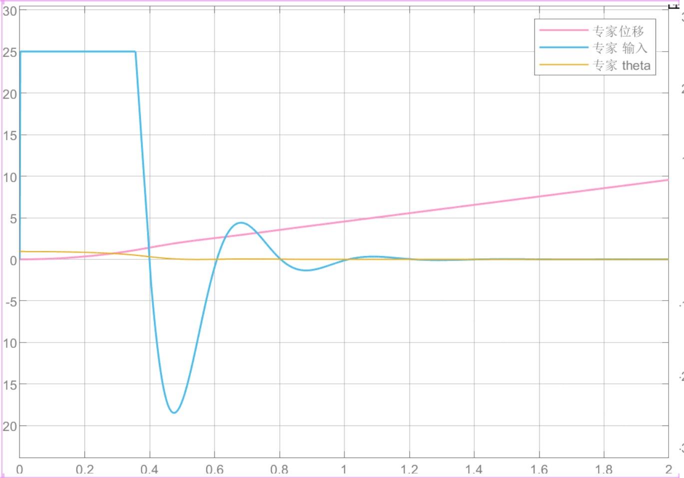
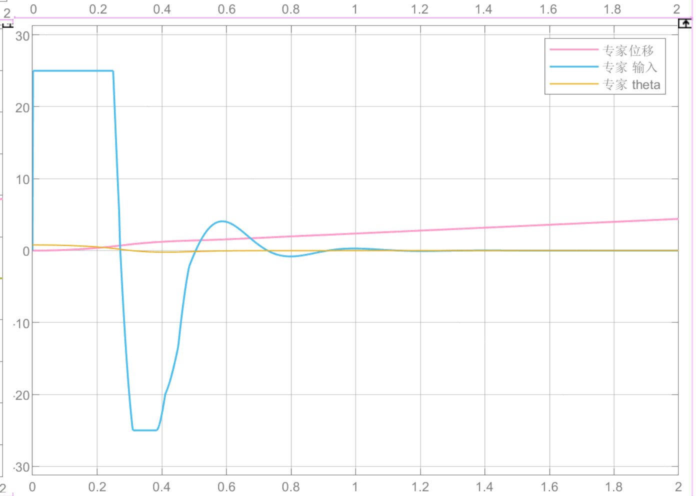
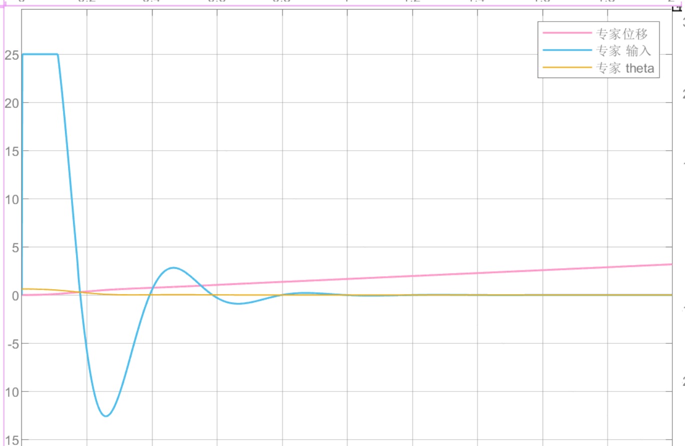
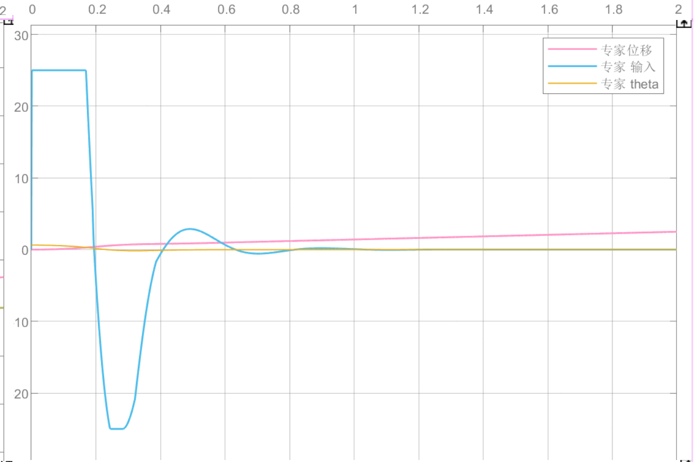
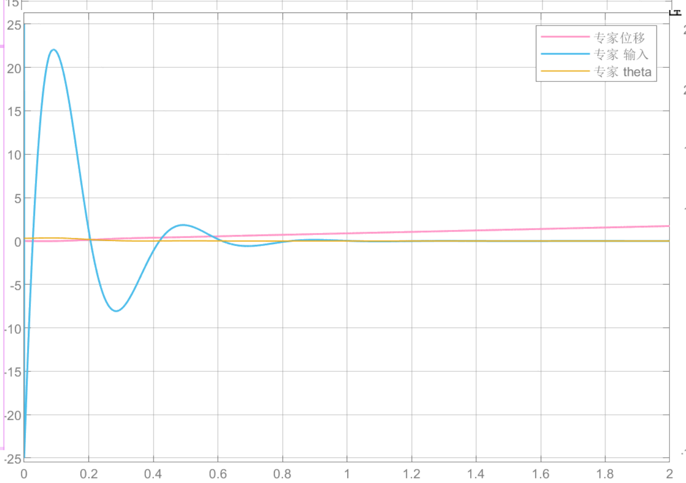
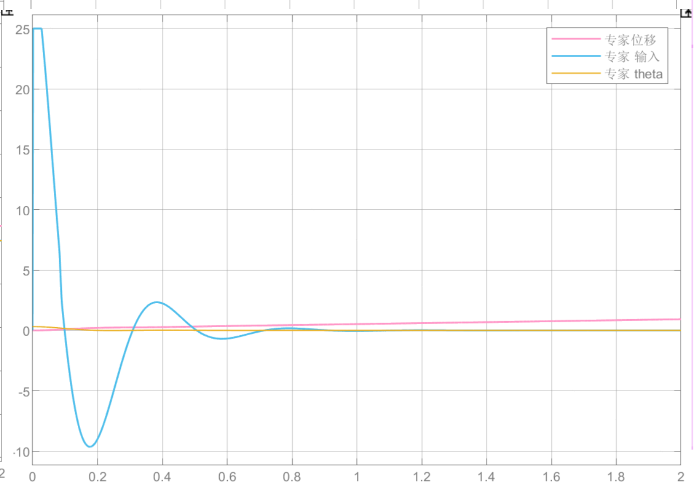
下面，探索不同初值$\theta(0)$下，专家控制效果是否仍然较好。首先证明如下断言：

Claim

$\exists \theta_0^* \in [0, \frac{\pi}{2}]$，使得，若初始条件满足 $\theta(0) = \theta_0$，其中$\forall \theta_0 > \theta_0^*$，则，不存在一组合适的参数，使得$\theta(\infty) = 0$

Proof 以车为参考系，分析小球受力，由于非惯性系中惯性力水平方向向左，且最大为$(mF_m)/M$，又受到竖直向下的重力mg，小球所受杆子的支撑力沿杆径向，因此，如果重力与惯性力合力，与竖直方向的夹角大于杆子与竖直方向的夹角，则所受合力不可能使得小球回归平衡，由几何知识得，此临界角$\theta_0^* = \arctan\left(\frac{F_m}{Mg}\right)$

因此，无需测试初始角度过大的倒立摆系统，因为在此题框架下，存在一些过大初值，不能控制成功的情况。

<table>
<tr><th>$\theta(0)$</th><th>$(\theta_1;\theta_2;\theta_m;K_s;K_b)=(0.1,0.3,0.5,1,1.3)$</th><th>$(\theta_1;\theta_2;\theta_m;K_s;K_b)=(0.1,0.2,0.3,2,3)$</th></tr>
<tr><td>$\frac{3\pi}{10}$</td><td></td><td></td></tr>
<tr><td>$\frac{\pi}{4}$</td><td></td><td></td></tr>
<tr><td>$\frac{\pi}{5}$</td><td></td><td></td></tr>
<tr><td>$\frac{\pi}{10}$</td><td></td><td></td></tr>
</table>

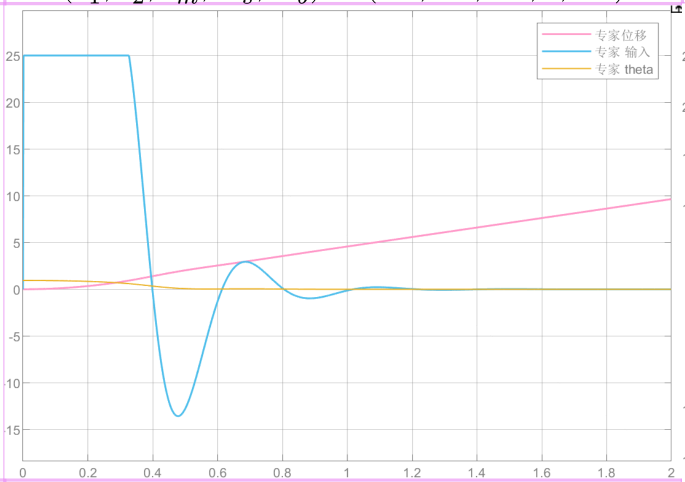
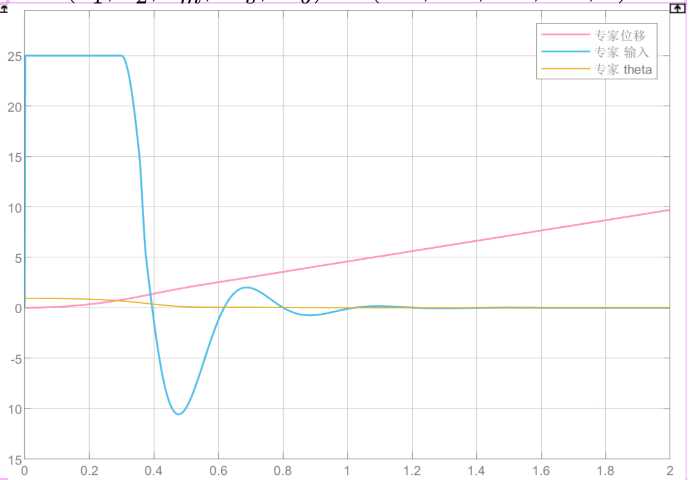
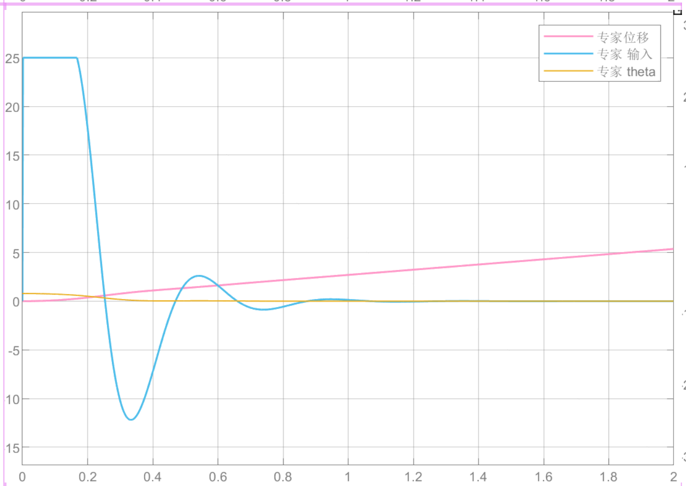
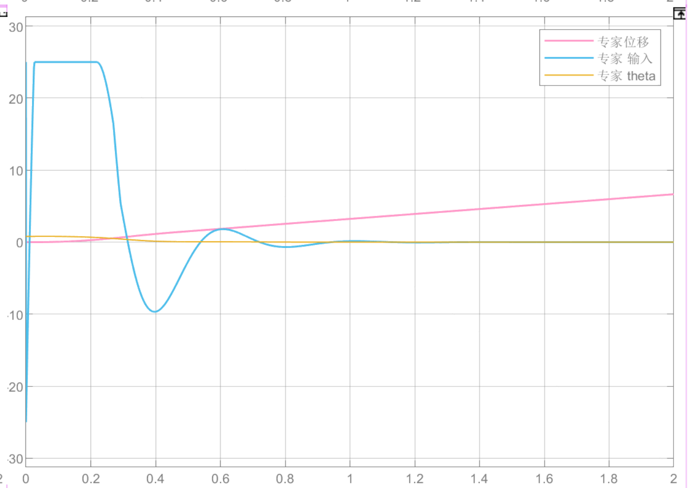
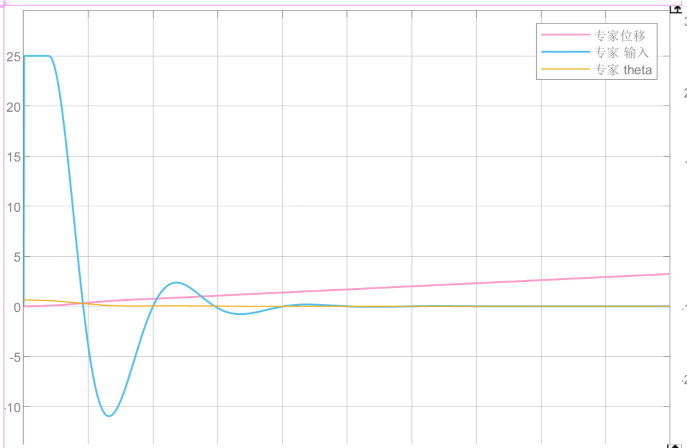
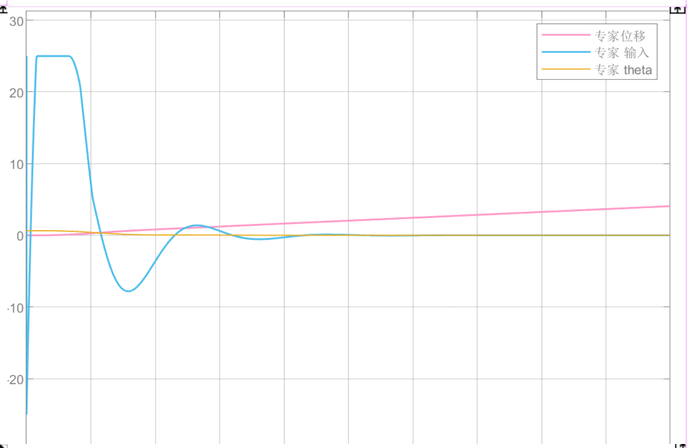
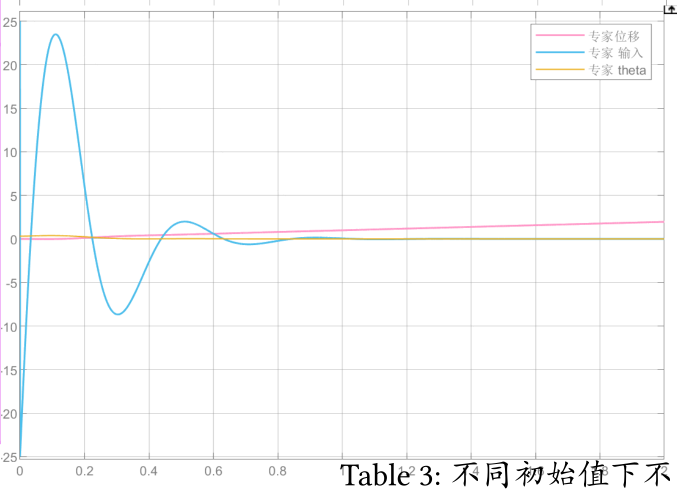
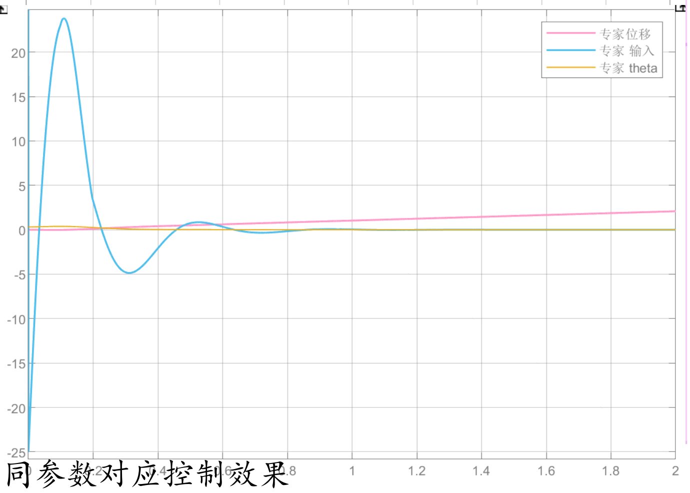
$\theta(0)$	$(\theta_1;\theta_2;\theta_m;K_s;K_b) = (0.2, 0.4, 0.6, 1, 1.2)$	$(\theta_1;\theta_2;\theta_m;K_s;K_b) = (0.2, 0.5, 0.8, 1.8, 2)$
$\frac{3\pi}{10}$		
$\frac{\pi}{4}$		
$\frac{\pi}{5}$		
$\frac{\pi}{10}$		

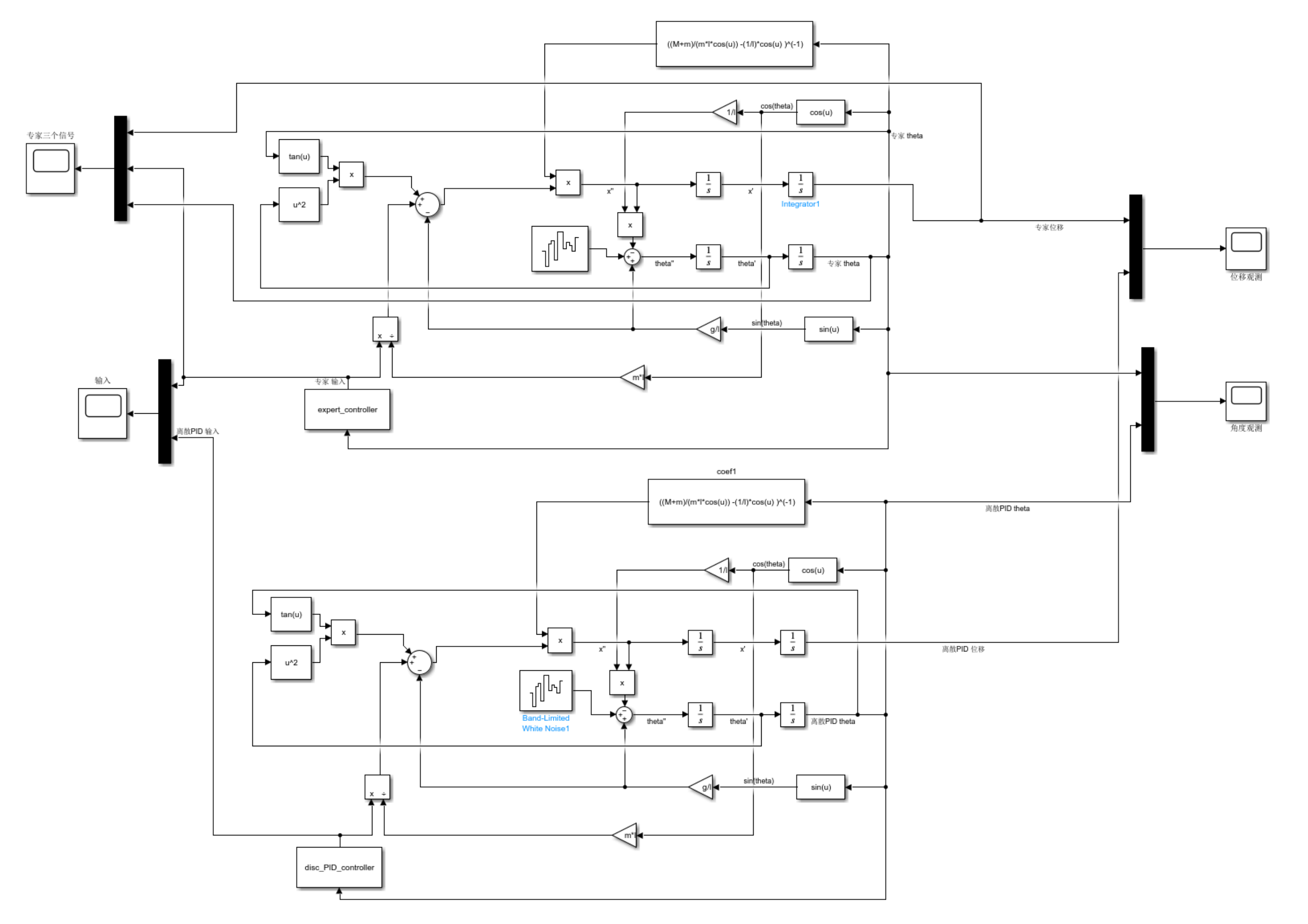
Table 3: 不同初始值下不同参数对应控制效果

抗扰动测试

白噪声

模拟当小车在运动时，$\ddot{\theta}$收到高斯白噪声的影响后，是否还能够较好维持在$\theta=0$附近

Simulink



此处，取的专家控制参数为：

$$(\theta_1;\theta_2;\theta_m;K_s;K_b)=(0.1,0.3,0.5,1,1.3)$$

取的初始角为

$$\theta(0)=\frac{\pi}{4}$$

模型如上图，$\ddot{\theta}$处加入了噪声，下表为不同噪声功率时，两种控制方法的效果对比。

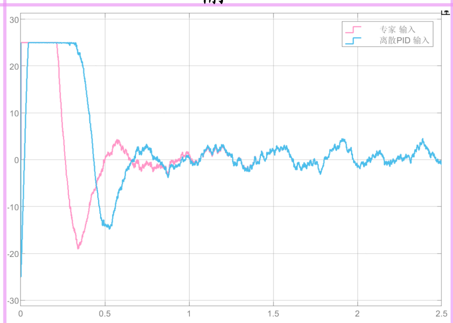
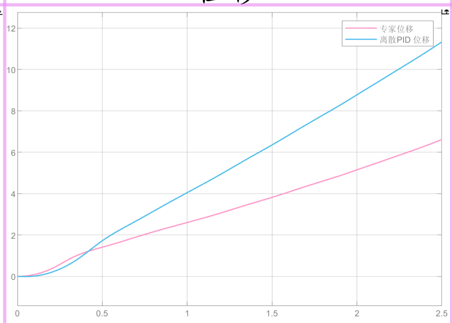
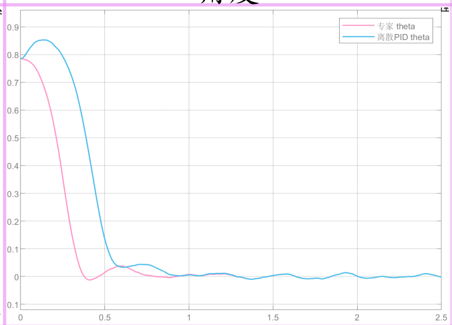
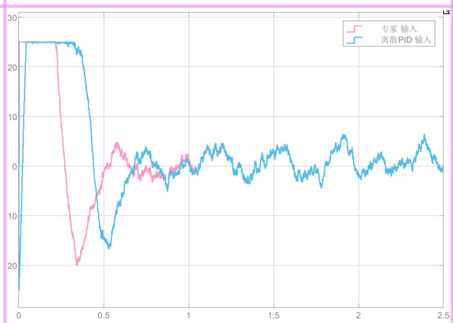
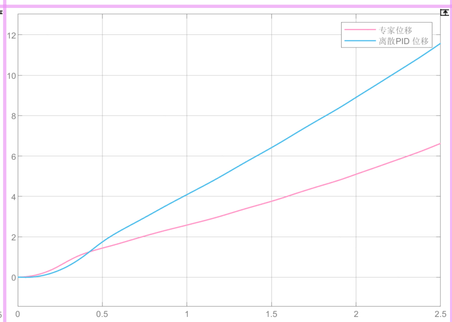
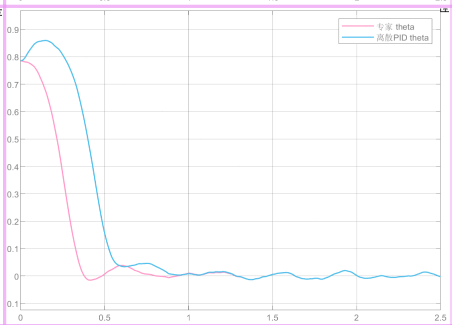
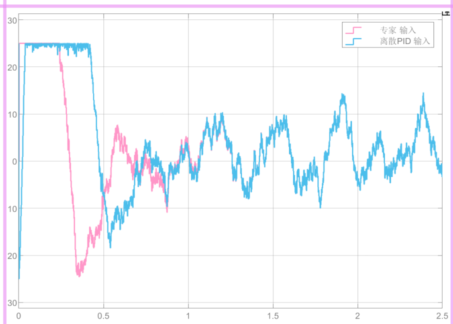
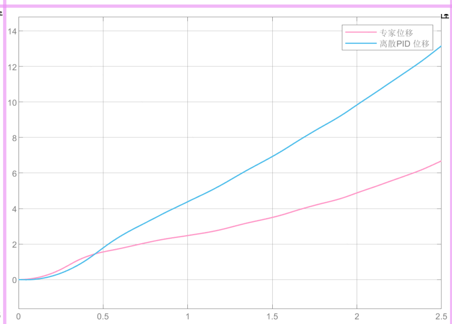
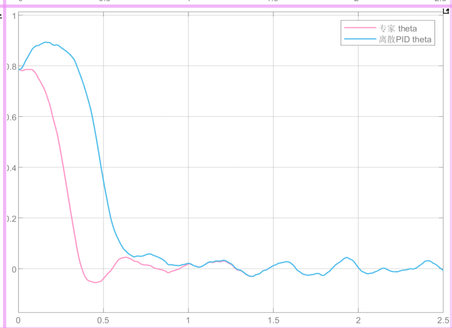
功率	输入	位移	角度
0.5			
1			
5			

Table 4: 白噪音下的控制效果对比

可见，专家控制在有噪音的情况下，控效果也比离散 PID 要更好一些

阶跃扰动

模拟当小车在运动时，球体收到榔头的冲激导致速度$\dot{\theta}$突变的情况，此种扰动$\ddot{\theta}$可视为无穷，比高斯白噪声影响更大。

此处，取的专家控制参数为：

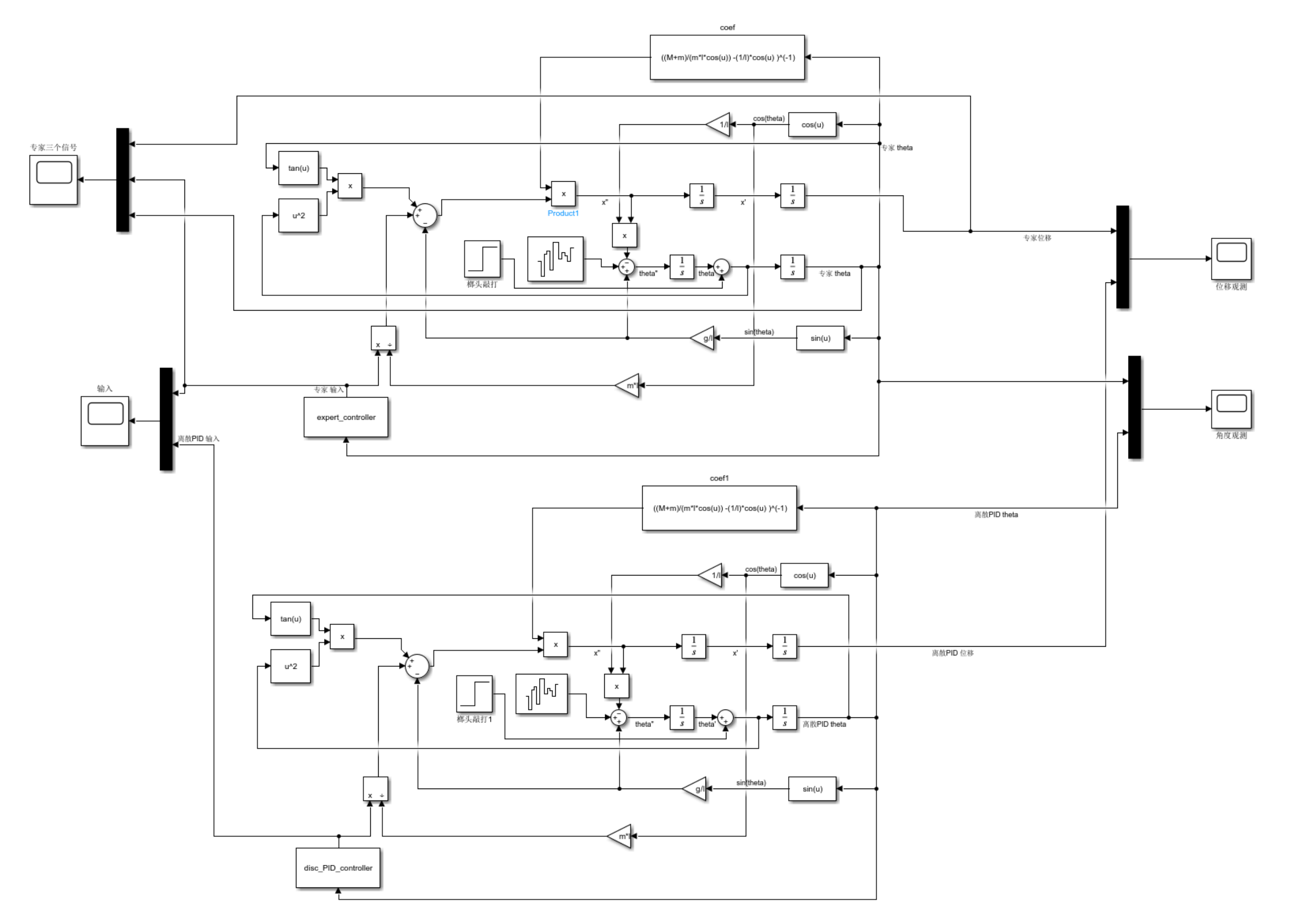
$$(\theta_1;\theta_2;\theta_m;K_s;K_b) = (0.1, 0.3, 0.5, 1, 1.3)$$

取的初始角为

$$\theta(0) = \frac{\pi}{4}$$

模型如下图，保留了$\ddot{\theta}$处的功率为 1 的高斯白噪声，并新增加$\dot{\theta}$处，0.8 秒时加入了大小为π的角速度阶跃。

Simulink



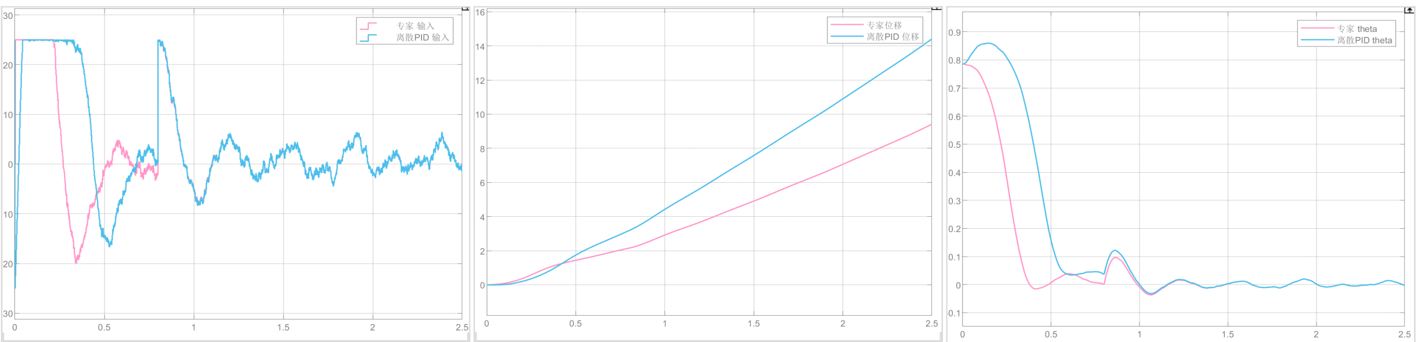


Table 5: 榔头敲击下，1. 输入 2. 小车位移 3. 倒立摆角度

可见，抗冲激效果还可以。

不同平衡位置测试

之前已经测试了平衡点为$\theta = 0$的情况，下面测试平衡点为$\theta = 0.1\pi$与$\theta = -0.1\pi$的情形。

取的专家控制参数为：$(\theta_1;\theta_2;\theta_m;K_s;K_b) = (0.1, 0.3, 0.5, 1, 1.3)$

初始角度为 $\theta(0) = \frac{\pi}{4}$

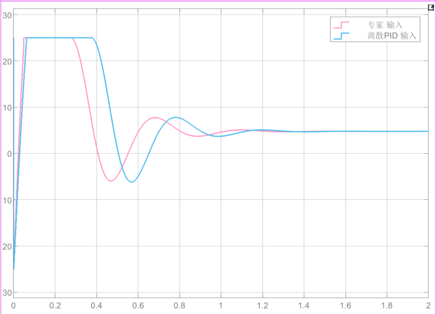
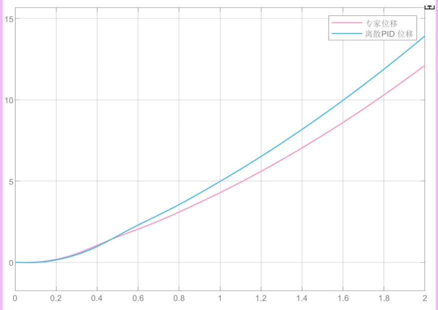
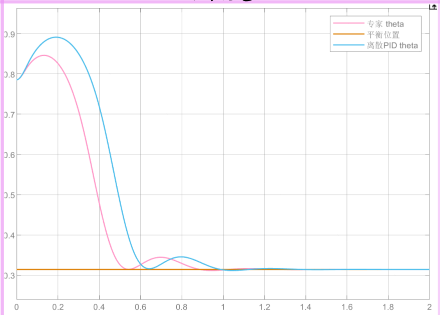
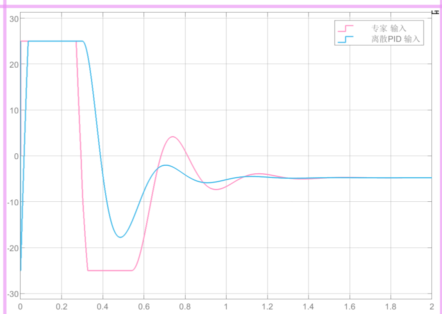
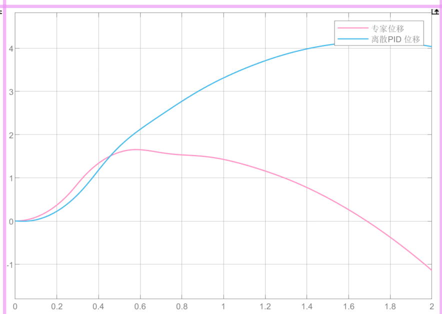
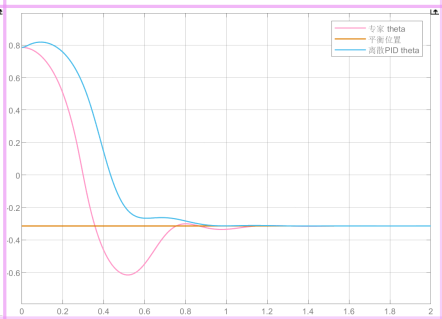
平衡点	输入	位移	角度
0.1π			
-0.1π			

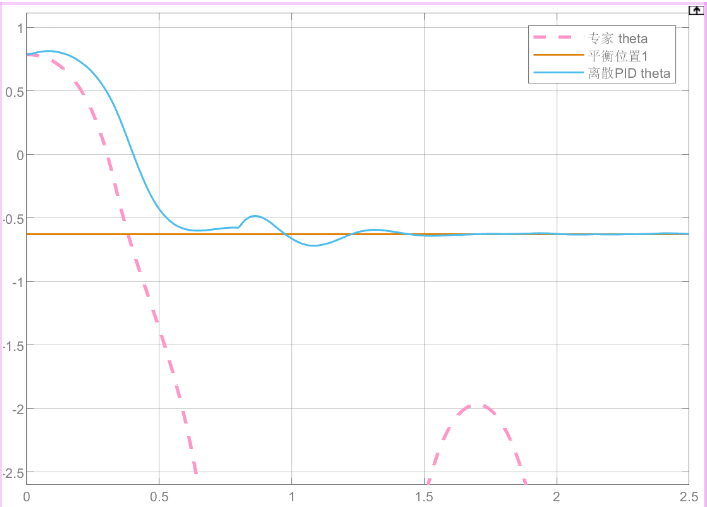
Table 6: 不同平衡点

可见，该专家控制规则也有缺点，在平衡角度为正时没问题，但平衡角度为负时，由于初始角度为正，因此一开始差距很大，导致输入过大，会出现极大的超调，后来刹车已经来不及了，因此稳定性低于 PID，事实上，当平衡角度过负时（比如-0.2π），专家控制就会失效，但是 PID 仍然能够成功（不考虑任何噪声）

新专家控制

动机分析

目前的专家控制和离散 PID 控制均对于较大的平衡点做的不好，尤其是专家控制。虽然目前的专家控制在最终平衡点为 0 或附近时，能有非常好的控制效果（无论是速度方面还是能耗方面），但是对于偏离 0 较大的负角度，就无法平衡了，出现很大超调（见 Table 6）即不稳定的预兆，事实上，当设置$\theta(0)=\frac{\pi}{4}$但是最终平衡位置为-0.18π时，则专家控制失效，离散 PID 仍能成功；当最终平衡位置为0.18π时，两者均失败：（注：此处均加入了白噪音和阶跃冲激，没噪音时这个角度是能成功的）



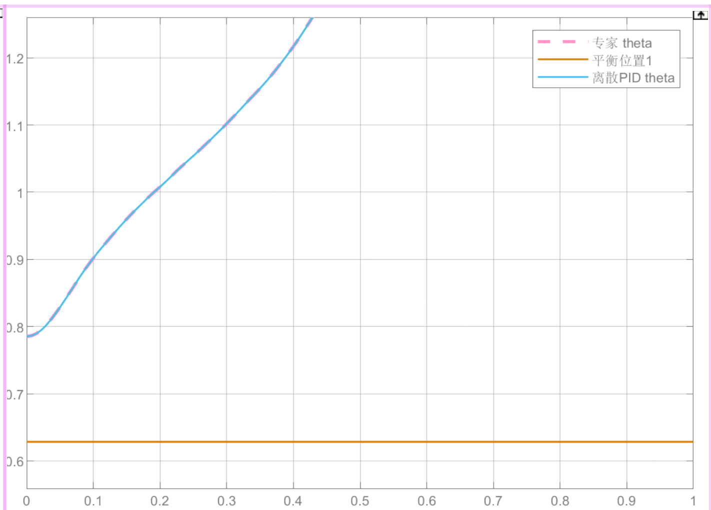


Table 7: 当前两种方案控制的失败案例

新专家控制思路

之前的专家控制之所以失效，是因为在差的很远时调大了K值，这会导致输入过头了无法刹车。因此，为保险起见，使用普通 PID 较好，但是普通 PID 的问题在于，误差太大时，输入还不够大，这又招致调节能力不足。

因此，新的方案是，在误差较大时，仅仅使用比例控制而禁用积分微分，但是调大比例系数，这一方面能避免旧专家控制的输入过头问题，又缓解了普通 PID 控制的输入不足问题。控制器设计如下，仅仅在离散 PID 的基础上做如下修改：

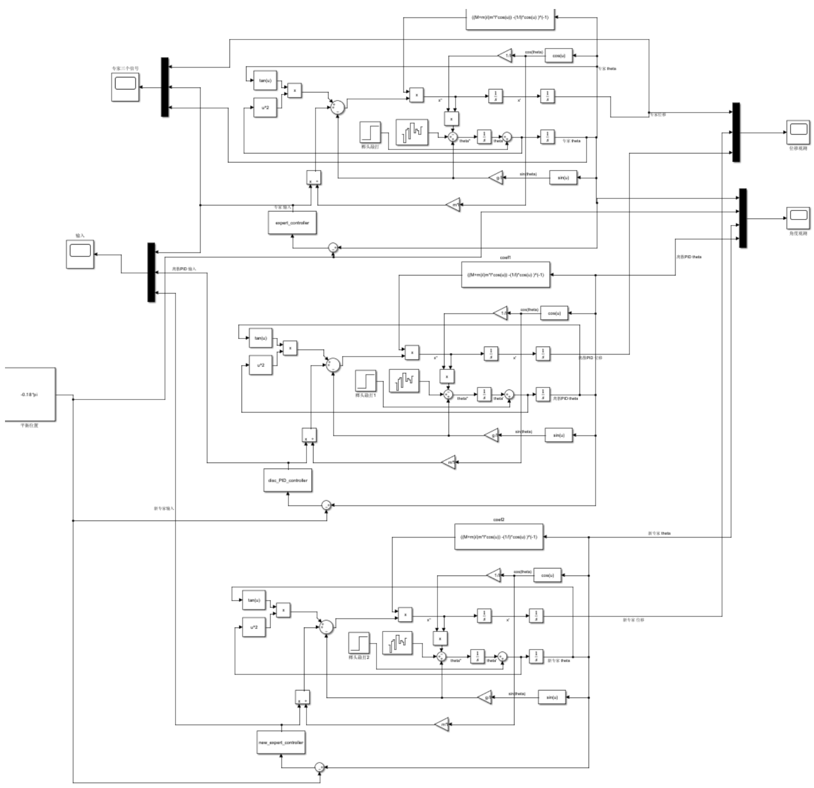
```
% 增量 PID 的 F(k) 更新公式:
  if Delta_Delta_theta+Delta_theta_k >= 0.0001
      F_k = block.Dwork(1).Data + 100*K*(K_p*Delta_theta_k);
  else
      F_k = block.Dwork(1).Data + K*(K_p*Delta_theta_k + T/T_i*block.InputPort(1).Data +
T_d/T *Delta_Delta_theta);
  end
```

新专家控制方法效果

测试时，提供三种方法（离散 PID、之前的专家控制、新专家控制）对比，发现新的专家控制在另外两种方法都发散的情形下，能够镇定系统。

注意，测试时加入了功率为 0.1 的白噪音和在 0.8 秒时施加的大小为π的角速度阶跃冲激，见下图在 0.8 秒处遭受较大的冲激后仍能恢复，证明新专家控制方案可行。下图中，由上到下的三个系统分别为旧专家、PID、新专家，共用一个 scope 查看信号。

Simulink



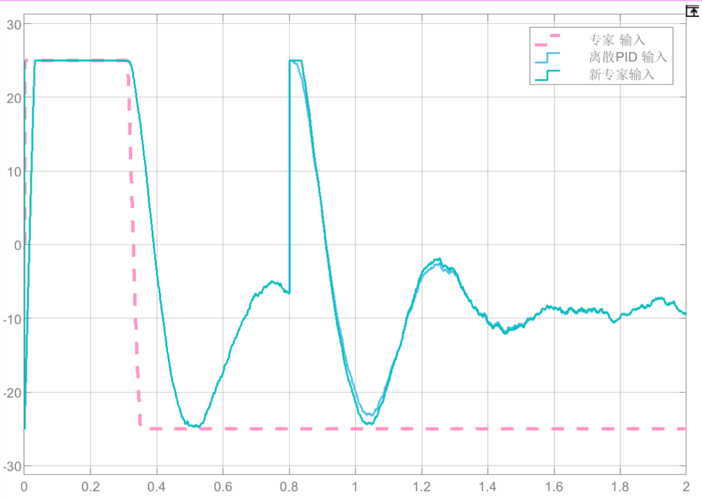
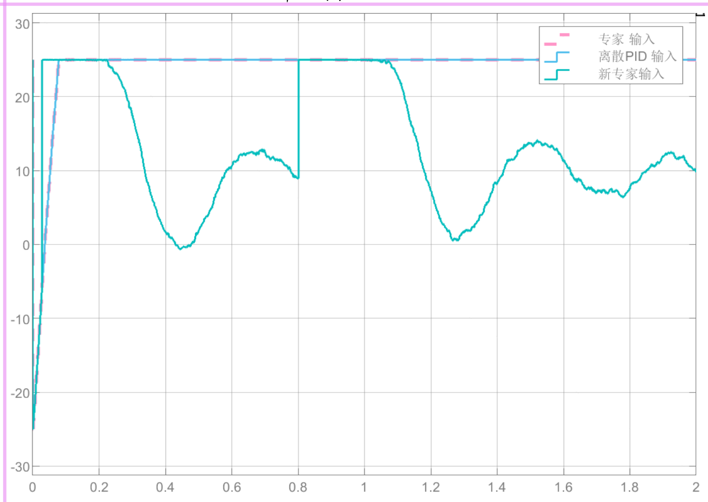
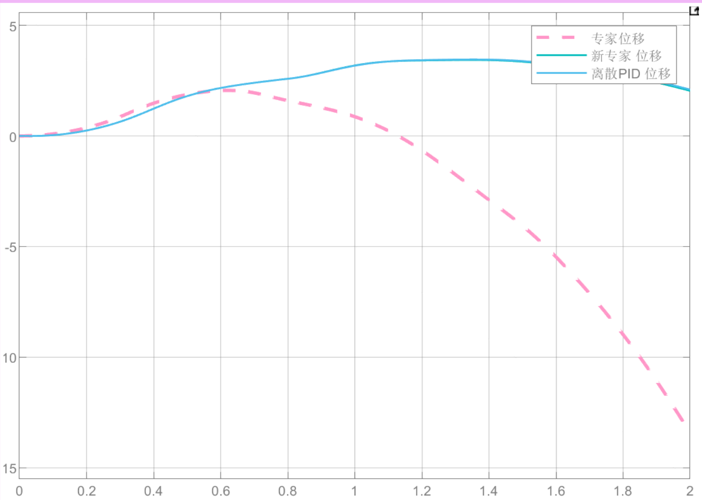
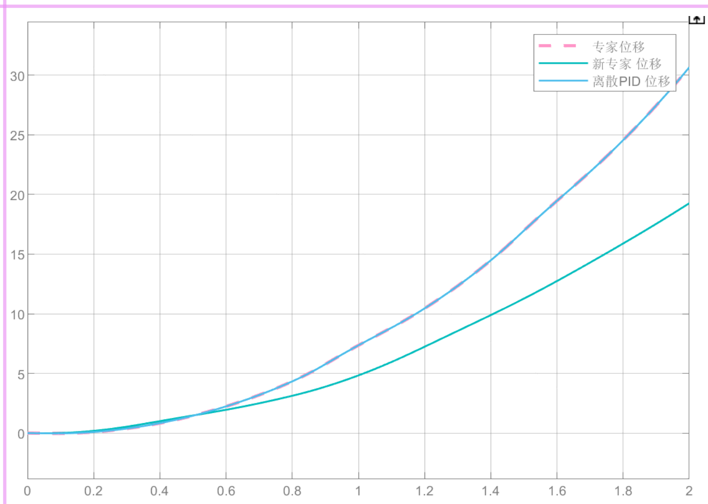
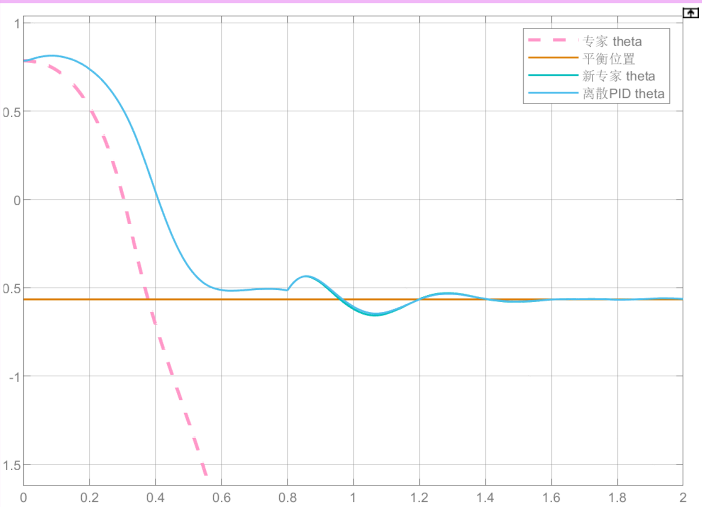
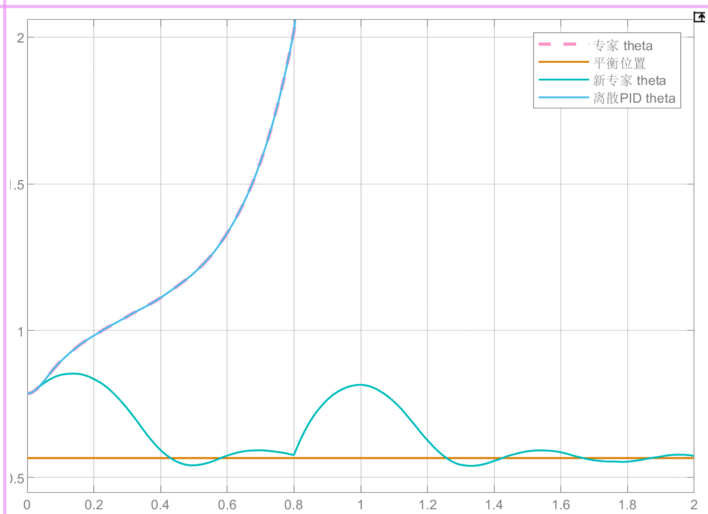
平衡点-0.18π	平衡点0.18π
	
	
	

Table 8: 前两种方案控制的失败时，新专家控制可行

代码附件与运行说明

> 使用的 Matlab 版本为 2023b

附件含有：

> cart_pendulum.slx 仿真模型

> param_for_cart.m 含有参数
> expert_controller.m、new_expert_controller.m、disc_PID_controller.m 三个不同的控制器 S-function 文件

测试时，请先运行 param_for_cart.m 文件（使得 workspace 中有参数的数据，这样 Simulink 以及 S-function 文件才能读到参数数值）然后即可运行 slx 文件。