**深 圳 大 学**

计算机控制技术大作业论文

题目: **离散时间倒立摆动态系统状态估计与最优控制**

姓名: **方仕展**

专业: **自动化**

学院: **机电与控制工程**

学号: 2021110303

教师: **崔玉康**

2023年11月

目录

[1 动力学分析与连续时间状态空间模型建立 5](#_Toc156311069)

[1.1 问题描述：倒立摆模型 5](#_Toc156311070)

[1.2 MATLAB建立动力学方程 7](#_Toc156311071)

[2 离散时间状态空间模型建立 8](#_Toc156311072)

[2.1 问题描述 8](#_Toc156311073)

[2.2 倒立摆问题背景 8](#_Toc156311074)

[2.3 理论推导 9](#_Toc156311075)

[2.4 使用MATLAB建立离散时间模型下的状态空间方程 11](#_Toc156311076)

[2.5 分析倒立摆系统稳定性、能控性及能观性 11](#_Toc156311077)

[2.5.1 稳定性 11](#_Toc156311078)

[2.5.2 能控性及能观性 13](#_Toc156311079)

[2.6 不加控制器求离散倒立摆系统在特定信号输入下的响应 14](#_Toc156311080)

[2.6.1 通过MATLAB求离散倒立摆系统阶跃响应和脉冲响应 14](#_Toc156311081)

[2.6.2 通过simulink求离散倒立摆系统阶跃响应和脉冲响应 15](#_Toc156311082)

[2.7 实验心得与总结 16](#_Toc156311083)

[3 离散时间倒立摆系统PID控制 17](#_Toc156311084)

[3.1 问题描述 17](#_Toc156311085)

[3.2 理论推导 17](#_Toc156311086)

[3.3 MATLAB建立连续倒立摆传递函数模型 17](#_Toc156311087)

[3.4 设计PID控制器 19](#_Toc156311088)

[3.4.1 使用simulink设计PID控制器 19](#_Toc156311089)

[3.4.2 在MATLAB中编程设计PID控制器 24](#_Toc156311090)

[3.4.3 分析加入PID控制器前后力到角度传递函数特性 26](#_Toc156311091)

[3.4.4 PID控制离散系统状态方程 27](#_Toc156311092)

[3.5总结与心得 28](#_Toc156311093)

[4 基于状态观测器的倒立摆系统最优控制 29](#_Toc156311094)

[4.1 问题描述 29](#_Toc156311095)

[4.2 理论推导 30](#_Toc156311096)

[4.2.1 LQR控制算法 30](#_Toc156311097)

[4.2.2 卡尔曼滤波 32](#_Toc156311098)

[4.3 simulink构建基于卡尔曼滤波器下的倒立摆LQR控制模块 37](#_Toc156311099)

[4.3.1 仿真框图 37](#_Toc156311100)

[4.3.2 高斯白噪声设计 37](#_Toc156311101)

[4.3.3 系统状态空间表达式模块设计 38](#_Toc156311102)

[4.3.4 LQR控制器设计 39](#_Toc156311103)

[4.3.5 卡尔曼滤波器设计 45](#_Toc156311104)

[5 基于卡尔曼滤波及LQR进行simscape刚体仿真 53](#_Toc156311105)

[5.1 simscape多刚体仿真模块搭建 53](#_Toc156311106)

[5.2 simscape多刚体仿真结果分析 55](#_Toc156311107)

[6 基于LQR编写倒立摆系统上位机 58](#_Toc156311108)

[6.1 倒立摆上位机设计流程 59](#_Toc156311109)

[6.1.1 前端页面设计 59](#_Toc156311110)

[6.1.2 后端逻辑设置 60](#_Toc156311111)

[6.2 上位机运行效果 65](#_Toc156311112)

[7 基于平衡小车进行倒立摆PID实验 67](#_Toc156311113)

[7.1 系统结构分析 68](#_Toc156311114)

[7.1.1 小车外设关系图 68](#_Toc156311115)

[7.1.2 电机控制流程图 68](#_Toc156311116)

[7.1.3 程序结构框图 69](#_Toc156311117)

[7.2 PID平衡小车程序设计 69](#_Toc156311118)

[7.2.1 直立环PD控制 69](#_Toc156311119)

[7.2.2 速度环PI控制 70](#_Toc156311120)

[7.2.3 转向环P控制 71](#_Toc156311121)

[7.3 平衡小车控制效果 71](#_Toc156311122)

[8 基于MPC模型预测控制的倒立摆仿真 72](#_Toc156311123)

[8.1 MPC理论推导 72](#_Toc156311124)

[8.1.1 最优控制 72](#_Toc156311125)

[8.1.2 MPC基本概念 73](#_Toc156311126)

[8.1.3 MPC公式推导 73](#_Toc156311127)

[8.2 基于MPC进行倒立摆仿真实验 76](#_Toc156311128)

[8.2.1 MPC模型预测函数编写 76](#_Toc156311129)

[8.2.2 Matlab 进行MPC倒立摆实验 77](#_Toc156311130)

[8.2.3 Simulink 进行MPC倒立摆实验 78](#_Toc156311131)

[9 倒立摆实验总结与心得体会 82](#_Toc156311132)

[9.1 实验总结： 82](#_Toc156311133)

[9.2 心得体会 83](#_Toc156311134)

[10 参考文献 83](#_Toc156311135)

[成绩评定 84](#_Toc156311136)

# 动力学分析与连续时间状态空间模型建立

## 问题描述：倒立摆模型

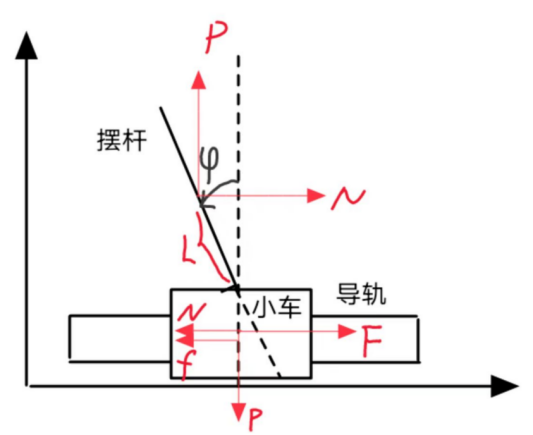


图 1 单级倒立摆模型示意图

忽略空气阻力、各种摩擦之后，可将倒立摆系统抽象为小车和匀质杆组成的系统。单级倒立摆模型结构如图1 所示。其中，小车质量 *M* (kg)，摆杆质量*m*(kg)，小车摩擦系数 *b*(N/m/sec)，摆杆转动轴心到杆质心的长度*l*(m)，摆杆惯量*I*(kg\*2)，加在小车上的力 F(N)，小车位置*x*(m)，摆杆与垂直向上方向的夹角中 (rad)，摆杆与垂直向下方向的夹角 (rad)。设小车与摆杆相互作用力的水平和垂直方向的分量为 *N* 和*P*。分析小车水平方向所受的合力，可以得到以下方程:



由摆杆水平方向的受力进行分析可以得到下面等式:



联立上式，得到系统的第一个运动方程:



对摆杆垂直方向上的合力进行分析，得到下面方程:



力矩平衡方程:



联立以上方程，得到第二个运动方程:



设 (是摆杆与垂直向上方向之间的夹角)，假设中与 1 rad 相比很小，即中<<1，则可以进行近似处理:



用u来代表被控对象的输入力F，线性化后得到两个运动方程



进一步变换得到系统的状态空间方程：





其中：给定模型中各个参数取值：小车质量M=1.096kg,摆杆质量m=0.109kg:小车摩擦系数 b=0.1N/m/sec，摆杆转动轴心到杆质心的长度l=0.25m，摆杆惯量I=0.0034kg\*m2

带入相应参数后获得连续时间状态空间模型。

## MATLAB建立动力学方程

在MATLAB中先定义相关变量大小：

%%%%%%%%%%%%%%%%%%%%%%%模型参数设定%%%%%%%%%%%%%%%%%%%%%

%小车质量

M = 1.096;

%摆杆质量

m = 0.109;

%小车摩擦系数0

b = 0.1;

%摆杆转动惯量

I = 0.0034;

%重力加速度

g = 9.8;

%仿真步长

Time\_step = 0.001;

%仿真时间

Time = 20;

%摆杆转动轴心到杆质心之间的距离

L = 0.25;

根据前面建立的状态空间表达式，在matlab中建立动力学方程，如下图所示，以便用于之后的离散化。

%%%%%%%%%%%%建立状态空间表达式%%%%%%%%%%%

%分母

p = I \* (M + m) + M \* m \* L^2;

% 状态方程相关矩阵

A\_con = [0 1 0 0;

0 -(I + m \* L^2) \* b / p, (m^2 \* g \* L^2) / p, 0;

0 0 0 1;

0. -(m \* L \* b) / p, m \* g \* L \* (M + m) / p, 0];

B\_con = [0; (I + m \* L^2) / p; 0; m \* L / p];

C\_con = eye(4);

D\_con = [0; 0; 0; 0];

sys\_con = ss(A\_con, B\_con, C\_con, D\_con);

运行后输出结果为：

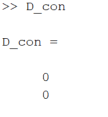
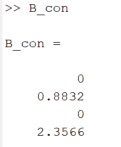
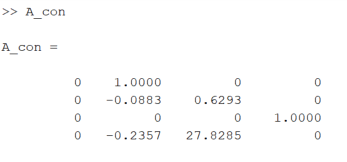


图 2 代入参数获得方程结果

与预期相符合，说明模型建立无误

# 离散时间状态空间模型建立

## 问题描述

对模型进行离散化处理，取采样时间T=0.02，求出离散时间模型（可借助Matlab），并给出离散时间系统在脉冲输入下的响应，用Matlab画出仿真图。

## 倒立摆问题背景

倒立摆系统是一个非线性、 不稳定的快速系统，其控制方式与直立行走的机器人、 飞行中的静不稳定导弹有许多相似之处。倒立摆系统是控制理论实验的典型装置，也是控制理论研究中常用的验证对象。

倒立摆问题有广泛的应用领域，例如机器人控制、自动驾驶、飞行器稳定性控制等。在这些应用中，倒立摆可以被看作是一个控制系统的模型，通过控制摆杆的角度或位置来实现摆锤的稳定。因此，倒立摆问题成为了研究和开发控制系统的重要工具。

在研究倒立摆问题时，需要考虑摆杆的质量、长度、摩擦力等因素，并通过分析其动力学方程来描述摆杆的运动。通常使用数学方法或计算机模拟来分析倒立摆的稳定性和控制方法，并设计出适合的控制策略来实现摆锤的倒立或保持稳定。这些研究对于理解和应用控制理论具有重要意义。

## 理论推导

①代入模型参数得到连续系统的数学表达式

②设定符号，便于连续系统离散化过程的符号表达

设连续系统的状态空间表达式为：

使用零阶保持器方法对系统离散化后的状态空间表达式为：

③根据零阶保持器对系统进行离散化

因为

所以

则有

所以在T=0.02下离散系统的状态矩阵为：

因为

所以在T=0.02下离散系统的输入矩阵为：

倒立摆离散系统的状态空间表达式为：

## 使用MATLAB建立离散时间模型下的状态空间方程

通过Matlab的c2dm函数进行验证，确保离散化状态空间表达式的准确性，由输出结果（精确到四位小数）可知状态空间方程离散化是准确的

%%开环连续系统离散化

sys\_disc = c2d(sys\_con, Time\_step,'zoh');

A\_disc = sys\_disc.A;

B\_disc = sys\_disc.B;

C\_disc = sys\_disc.C;

D\_disc = sys\_disc.D;

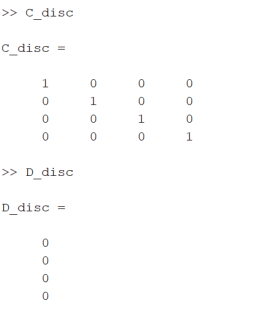
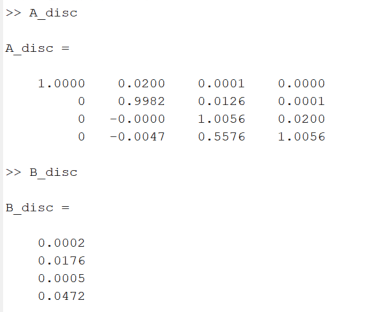


图 3 离散时间模型下的状态空间方程

## 分析倒立摆系统稳定性、能控性及能观性

### 稳定性

对于连续状态空间方程，系统的稳定性取决于矩阵A\_con的特征值。如果所有特征值的实部都是负的，那么系统是稳定的。如果存在任何一个特征值的实部是正的，那么系统是不稳定的。如果有特征值的实部为零，那么系统可能是不稳定的或者是边界稳定的，需要进一步的分析。

对于离散状态空间方程，系统的稳定性分析也是通过矩阵A\_disc的特征值进行。与连续状态空间方程不同的是，离散系统的稳定性与特征值的模是否小于1有关。如果所有特征值的模都小于1，那么系统是稳定的。如果存在任何一个特征值的模大于1，那么系统是不稳定的。

在MATLAB中通过eig获取连续状态空间方程A矩阵A\_con和离散状态空间方程的A矩阵A\_disc特征值。通过real函数获取A\_con实部并判断是否都大于0来判断稳定性，abs函数获取A\_disc的模长并判断是否大于1来判断稳定性，同时通过syms定义特征方程变量S，通过MATLAB表达特征方程并通过vpa精确到4位小数。

syms S

%系统在离散和连续状态下的极点和特征方程

poles\_con = eig(A\_con);

characteristic\_equation\_con = vpa((S - poles\_con(1)) \* (S - poles\_con(2)) \* (S - poles\_con(3)) \* (S - poles\_con(4)), 4);

disp('离散系统特征方程为：');

disp(characteristic\_equation\_con);

if(find(real(poles\_con) > 1))

disp('离散状态下的系统不稳定：');

else

disp('离散状态下的系统稳定：');

end

poles\_disc = eig(A\_disc);

characteristic\_equation\_disc = vpa((S - poles\_disc(1)) \* (S - poles\_disc(2)) \* (S - poles\_disc(3)) \* (S - poles\_disc(4)), 4);

disp('连续系统特征方程为：');

disp(characteristic\_equation\_disc);

if(find(abs(poles\_disc) > 0))

disp('连续状态下的系统不稳定：');

else

disp('连续状态下的系统稳定：');

end

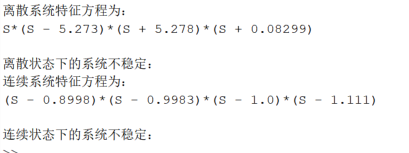


图 4 MATLAB稳定性诊断

### 能控性及能观性

系统的可控性和可观性可以通过系统的状态空间表示中的A和B、C矩阵来判断。

可控性：对于连续和离散系统，如果矩阵M = [B, AB, A^2B, ..., A^{n-1}B] ]（ n 是系统的状态空间维度）的秩等于系统的状态空间维度，则系统是可控的。

可观性：对于连续和离散系统，如果N=[C,CA,...,CA^(n-1)]( n 是系统的状态空间维度)的秩等于系统的状态空间维度，则系统是可观的。

在MATLAB中通过ctrb函数获取系统能控性矩阵，obsv函数获取系统能观性矩阵并通过rank获取矩阵的秩，通过判断秩是否等于系统状态空间维度来确定系统能控性和能观性，由命令行窗口输出可以得知连续和离散系统下系统均是能控和能观的。

%能控性和能观测性分析

disp('判断系统能控性和能观性');

%连续系统

% 判断可控性

rank\_ctrb\_con = rank(ctrb(A\_con, B\_con));

if rank\_ctrb\_con == size(A\_con, 1)

disp('连续系统为可控的');

else

disp('连续系统为不可控的');

end

% 判断可观性

rank\_obsv\_con = rank(obsv(A\_con, C\_con));

if rank\_obsv\_con == size(A\_con, 1)

disp('系统连续为可观的');

else

disp('系统连续为不可观的');

end

%连续系统

% 判断可控性

rank\_ctrb\_disc = rank(ctrb(A\_disc, B\_disc));

if rank\_ctrb\_disc == size(A\_disc, 1)

disp('离散系统为可控的');

else

disp('离散系统为不可控的');

end

% 判断可观性

rank\_obsv\_disc = rank(obsv(A\_disc, C\_disc));

if rank\_obsv\_disc == size(A\_disc, 1)

disp('离散系统为可观的');

else

disp('离散系统为不可观的');

end

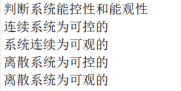


图 5 MATLAB能控能观性分析

## 不加控制器求离散倒立摆系统在特定信号输入下的响应

### 通过MATLAB求离散倒立摆系统阶跃响应和脉冲响应

设置t = 0:0.2:10;即以0.2采样时间仿真10S，通过step和impulse函数求离散倒立摆系统阶跃响应和脉冲响应

%绘制阶跃响应，为发散

figure

t = 0:0.2:10; % 时间向量，步长为 0.1

step(sys\_disc, t);

grid on; title('阶跃响应'); xlabel('Time'); ylabel('Amplitude'); drawnow

%绘制脉冲响应，为发散

figure

t = 0:0.2:10; % 时间向量，步长为 0.1

impulse(sys\_disc, t);

grid on; title('脉冲响应'); xlabel('Time'); ylabel('Amplitude'); drawnow

由运行结果得知，在未加控制器的情况下，系统阶跃响应和脉冲响应均为发散

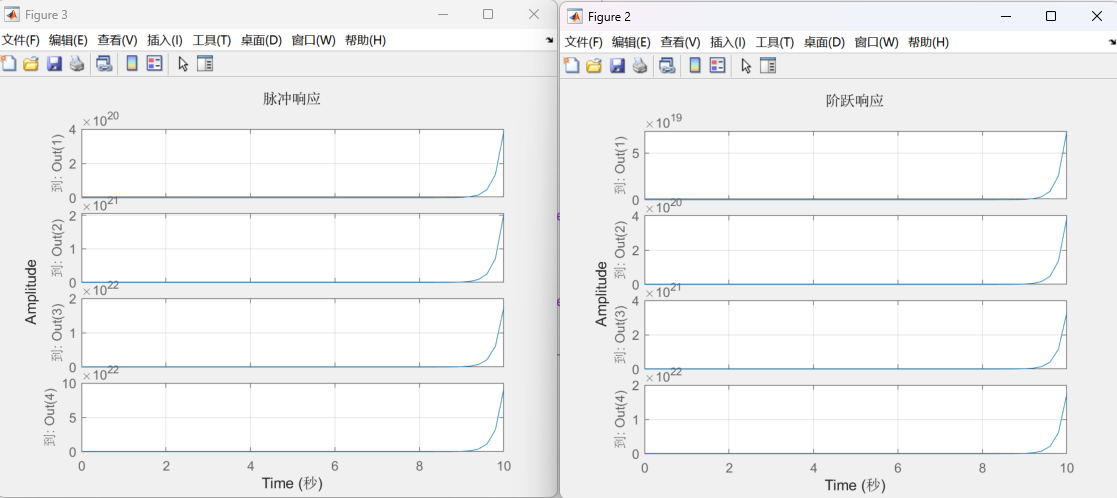


图 6 系统阶跃响应和脉冲响应

### 通过simulink求离散倒立摆系统阶跃响应和脉冲响应

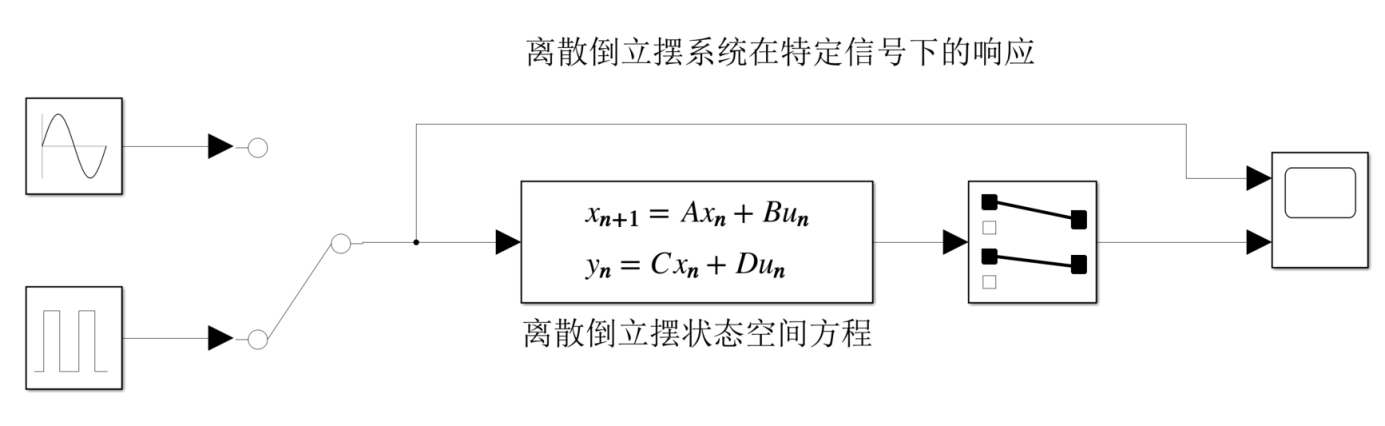


图 7 simulink离散倒立摆系统框图

Simulink框图如上，在“离散倒立摆状态空间方程”元件中，绑定.m文件生成的A\_disc、B\_disc、C\_disc和D\_disc矩阵，通过Selector模块，筛选出位置、角度状态变量。由scope的输出响应得知：对模型进行离散化处理后，无控制器控制的情况下，输入端添加脉冲信号或者正弦信号，倒立摆受到脉冲响应、正弦响应之后将会不再收敛，整个系统变得不再稳定，与实际情况一致。此时需要控制器对系统进行控制。

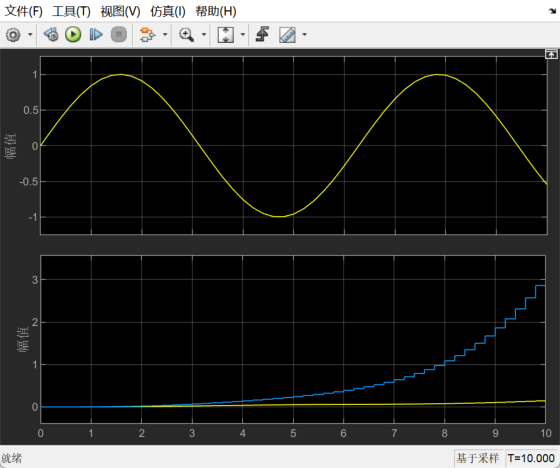
 

图 8 正弦信号及系统响应 图 9 脉冲信号及系统响应

## 实验心得与总结

在建立倒立摆离散时间状态空间模型的过程中，我遇到了一些困难。首先，我发现在对倒立摆系统进行离散化时，需要对系统的非线性动力学方程进行线性化处理后再进行离散化，但这一过程需要较高对差分方程的理解。其次，在构建状态空间模型时，需要确定系统的状态变量和观测变量，而这一步骤需要对系统的物理特性有深入的了解，包括力学和控制理论等方面。最后，在建立数学模型时，需要保证模型的稳定性和准确性，这需要较高的数学建模和仿真能力。

为了解决这些困难，我首先加强了对数学建模和控制理论的学习，深入理解了差分方程和状态空间模型的原理，同时加强了对非线性动力学系统的线性化方法的学习。其次，我加强了对倒立摆系统的物理特性的研究，包括对摩擦、质量、惯性和控制器设计等方面的理论学习，以便确定系统的状态和观测变量。最后，通过实际的数学建模和仿真实验，验证模型的准确性和稳定性，得到符合实际的离散时间状态空间模型。

通过这一过程，我深刻体会到了数学建模和控制理论的重要性，也意识到了在实际工程中，理论知识的应用和实践能力的培养是同等重要的。通过不断研究和实践，我逐渐掌握了倒立摆离散时间状态空间模型的建立方法，也加深了对控制理论的理解和应用。这一过程在促使我不断学习和提高的同时，也增强了我的数学建模和工程实践能力。

# 离散时间倒立摆系统PID控制

## 问题描述

采用PID控制器，对倒立摆的离散时间模型进行控制，并给出脉冲输入和单位阶跃输入下的系统响应仿真曲线；分析上升时间，稳定时间，超调量，稳态误差等控制性能参数。

## 理论推导

在前面通过动力学分析建立连续时间状态空间模型部分，获得如下两条运动方程



对以上式子进行拉普拉斯变换，得：





对上式进行化简，先将第一条运动方程的提取出来，带入下面第二条方程，话讲获得力到角度传递函数，将第二条运动方程的提取出来，带入上面第一条运动方程，得到力到位置传递函数：

****

## MATLAB建立连续倒立摆传递函数模型

在matlab中进行建模，定义s作为拉普拉斯变换的变量，便于后面建立传递函数

% 定义 s 为拉普拉斯变换的变量

s = tf('s');

之后根据前面求得的力到位置、力到角度的传递函数，建立传递函数模型，同时将分子分母分别提取出来，保存到工作区，供simulink的PID模型使用

% 位置传递函数

position\_tf = (((I + m \* L^2) ) \* s^2 - (m \* g \* L)) / ...

(((M + m) \* (I + m \* L^2) - (m \* L)^2)\*s^4 + ...

(b \* (I + m \* L^2)) \* s^3 - ...

((M + m) \* m \* g \* L) \* s^2 - ...

b \* m \* g \* L \* s );

% 计算分子和分母的系数，保存到工作空间便于PID模型调用

%分子

num\_position = [((I + m \* L^2)), 0, - (m \* g \* L)];

%分母

den\_position = [((M + m) \* (I + m \* L^2) - (m \* L)^2), (b \* (I + m \* L^2)), - ((M + m) \* m \* g \* L), - b \* m \* g \* L , 0];

% 角度传递函数

angle\_tf = (m \* L \* s ) / (((M + m) \* (I + m \* L^2) - (m \* L)^2) \* s^3 + ...

(b \* (I + m \* L^2)) \* s^2 - ...

((M + m) \* m \* g \* L) \* s - ...

b \* m \* g \* L );

% 计算分子和分母的系数，保存到工作空间便于PID模型调用

%分子

num\_angle = [m \* L, 0];

%分母

den\_angle = [((M + m) \* (I + m \* L^2) - (m \* L)^2), (b \* (I + m \* L^2)), - ((M + m) \* m \* g \* L), - b \* m \* g \* L];

% 将小车和吊索的传递函数组合成一个 2x1 的传递函数 sys\_tf

pos\_ang\_tf = [position\_tf ; angle\_tf];

%离散化

angle\_tf\_disc = c2d(angle\_tf, Time\_step, 'zoh');

num\_angle\_disc = cell2mat(angle\_tf\_disc.Numerator);

den\_angle\_disc = cell2mat(angle\_tf\_disc.Denominator);

## 设计PID控制器

### 使用simulink设计PID控制器

#### 3.4.1.1模型搭建

将num\_angle\_disc和den\_angle\_disc保存到工作区，并在simulink建立PID控制模型，如下图所示：

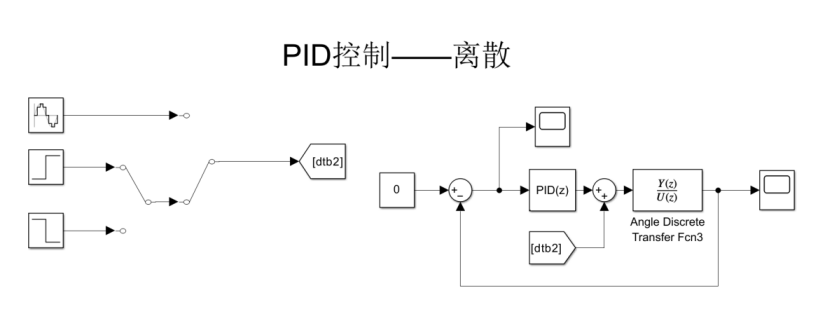


图 10 离散系统PID框图

在Angle Transfer Fun模块中将分子分母定位到工作空间的num\_angle\_disc

和den\_angle\_disc



图 11 simulink传递函数设计

使用PID模块的tunner，获取一组PID参数，并在此基础上进行调节，在PID调节器中，阶跃仿真的结果如下，其中，上升时间为0.073S，稳定时间为0.785S，超调量为44.6%，峰值为1.46。

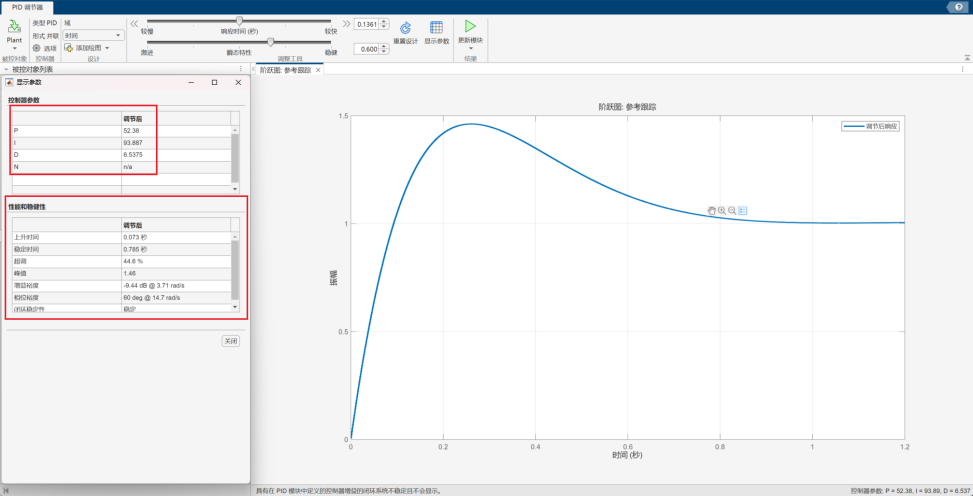


图 12 PID模块的tunner设定

将阶跃信号作为目标值接入simulink PID 模型,求阶跃响应，其simulink框图如下，主要改动为：将目标输入改为阶跃信号，并删除干扰

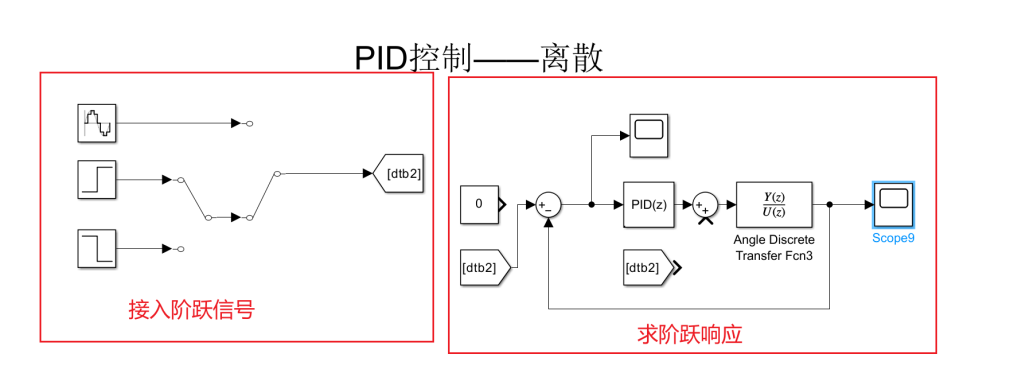


图 13 求解系统阶跃响应

运行simulink模型，通过scope得到如下响应，可知峰值约为1.45，稳定时间在0.8S，与PID调节器中仿真结果相同

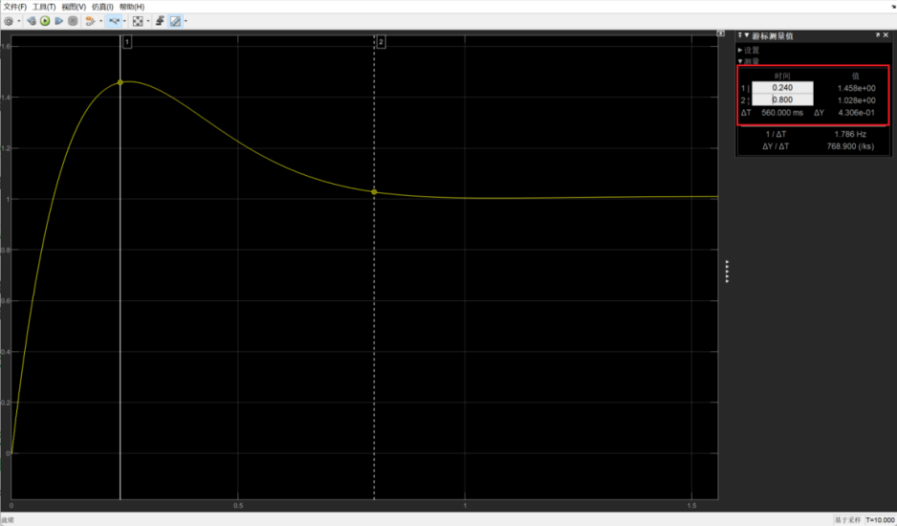


图 14 阶跃响应波形

#### 3.4.1.2 添加PID控制器下系统在特定力干扰下的输出响应

接下来将目标值设置为0，并加入阶跃、脉冲及正弦干扰，验证力到角度的传递函数在添加完PID控制器后角度能否快速收敛到0。

（1）干扰为阶跃信号

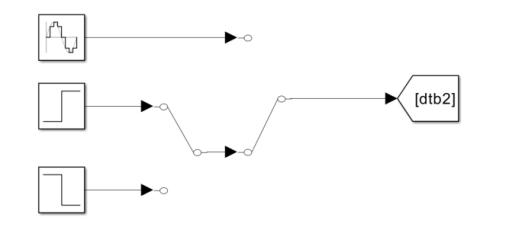


图 15 阶跃干扰信号

将干扰力修改为阶跃信号，终值为1，即给1N的力，其波形如下：

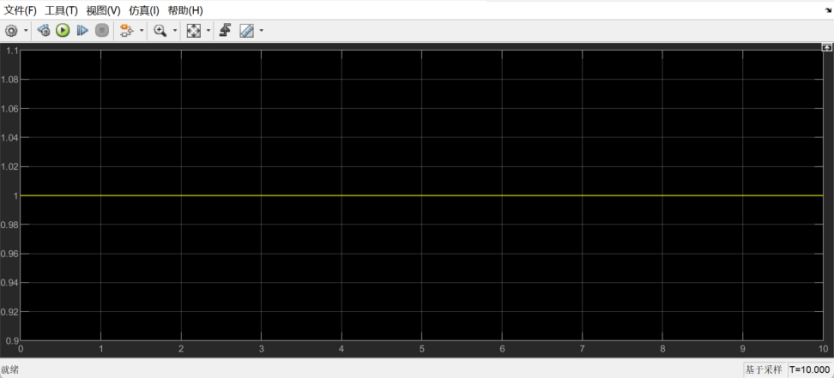


图 16 干扰信号波形

给定恒定为1N的干扰力后，其响应波形如下，可以看到，超调量为0.0024rad，约0.13°，稳定时间为1.2S。

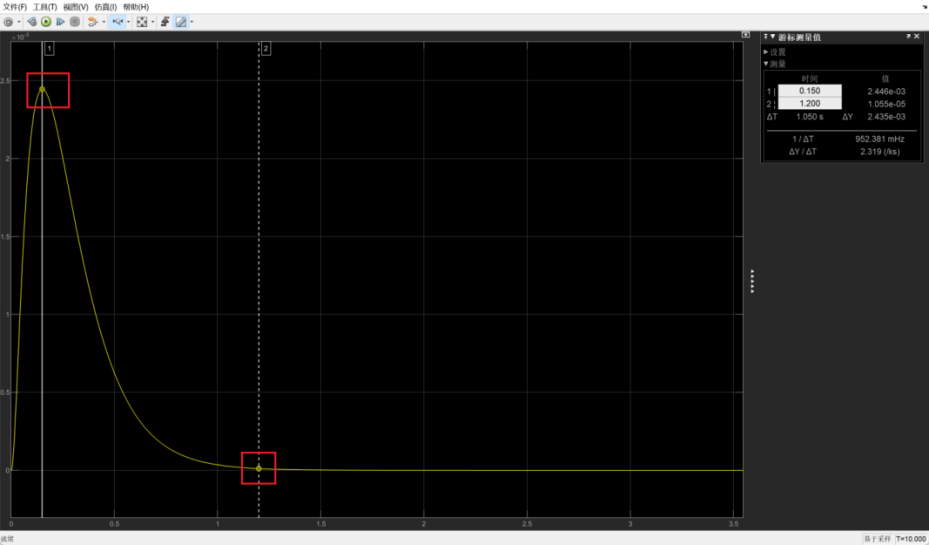


图 17 阶跃信号干扰响应

（2）干扰为脉冲信号

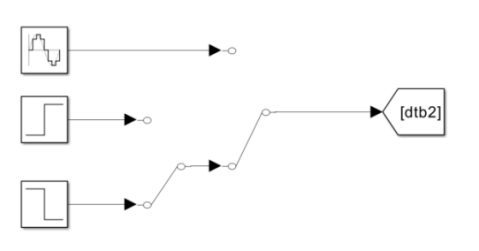


图 18 脉冲干扰信号

将干扰力修改为脉冲，在0时刻给10N的力，并在0.1时置为0，其波形如下：

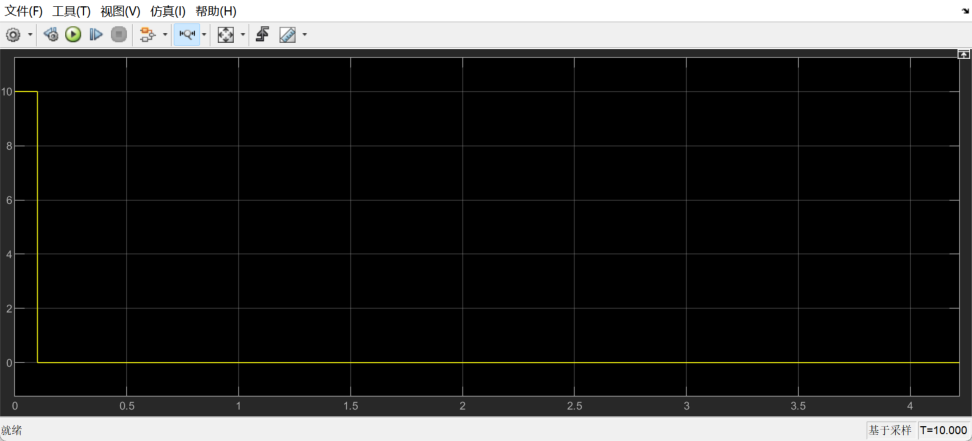


图 19 脉冲干扰信号波形

给定10N，持续0.1S的干扰力后，其响应波形如下，可以看到，超调量为0.022rad，约1.26°，稳定时间为1.4S。

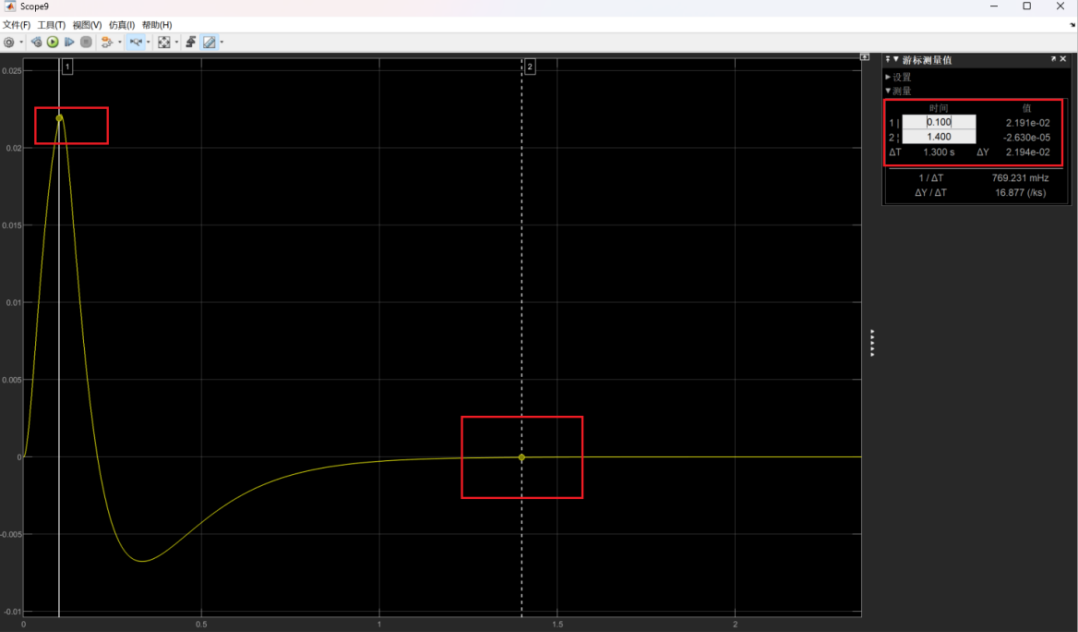


图 20 脉冲干扰信号响应

（3）干扰为正弦信号

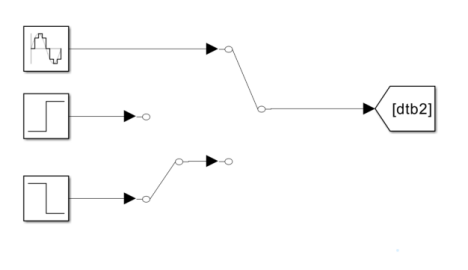


图 21 正弦干扰信号

将干扰力修改为正弦信号，峰值为1N，其波形如下：

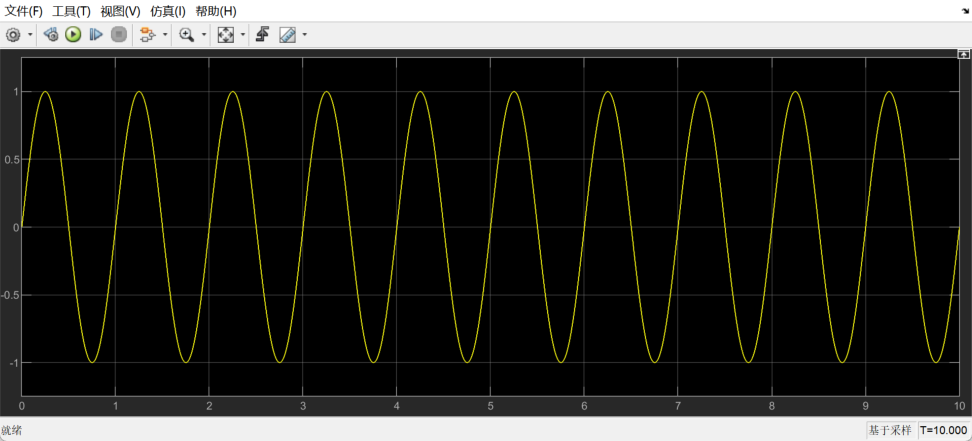


图 22 正弦干扰信号波形

给定峰值为1N的正弦干扰力后，其响应波形如下，可以看到，超调量为0.0004rad，约0.023°，存在一定的震荡，但是幅度比较小，在可接收范围内，接近稳定。

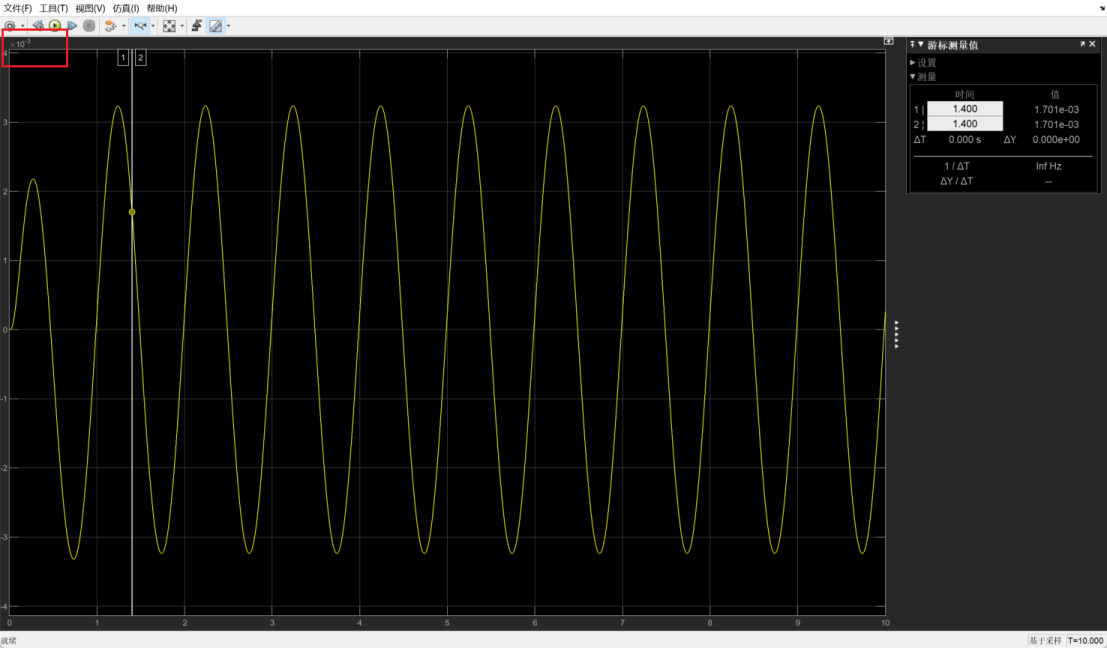


图 23 正弦干扰信号响应

### 在MATLAB中编程设计PID控制器



图 24 未添加控制器系统响应

在Matlab中通过get\_param获取simulink文件PID\_Ctr中的Angle\_disc\_PID元件的PID参数，通过series函数将PID控制器和力到角度的传递函数串联，同时使用feedback函数添加负反馈，最后使用impulse和step函数求得力到角度的传递函数的阶跃响应和脉冲响应，使用stepinfo函数可以获取系统阶跃响应下的上升时间、稳定时间以及超调量等性能参数。

% 设置离散控制器的参数 Kp、Ki 和 Kd

Kp\_disc = str2double(get\_param('PID\_Ctr/Angle\_disc\_PID','P'));

Ki\_disc = str2double(get\_param('PID\_Ctr/Angle\_disc\_PID','I'));

Kd\_disc = str2double(get\_param('PID\_Ctr/Angle\_disc\_PID','D'));

PID\_Ctr\_disc = pid(Kp\_disc, Ki\_disc, Kd\_disc, Ts = Time\_step);

% 使用 feedback 函数求离散系统添加PID控制器后的响应

angle\_tf\_OL\_PID\_disc = series(PID\_Ctr\_disc, angle\_tf\_disc);

angle\_tf\_PID\_disc = feedback(angle\_tf\_OL\_PID\_disc, 1);

if draw\_state

figure

impulse(angle\_tf\_PID\_disc, T\_PID);

title('PID控制器下离散系统的脉冲响应'); drawnow

figure

step(angle\_tf\_PID\_disc, T\_PID);

title('PID控制器下离散系统的阶跃响应'); drawnow

[angle\_tf\_PID\_disc\_step\_info]= stepinfo(angle\_tf\_PID\_disc);

fprintf('上升时间：%d\n', angle\_tf\_PID\_disc\_step\_info.RiseTime);

fprintf('峰值时间：%d\n', angle\_tf\_PID\_disc\_step\_info.PeakTime);

fprintf('调节时间：%d\n', angle\_tf\_PID\_disc\_step\_info.SettlingTime);

fprintf('峰值：%d\n', angle\_tf\_PID\_disc\_step\_info.Peak);

end

由输出结果可以得知上升时间约为0.04S，峰值时间约为0.12S，调节时间为0.82S，峰值为1.2，即超调量为0.2.具有较好的控制效果。

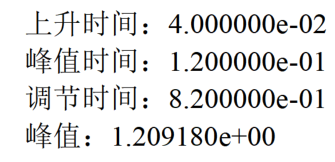


图 25 PID控制器下系统性能



图 26 PID控制器下的离散系统阶跃和脉冲响应

### 分析加入PID控制器前后力到角度传递函数特性

#### 3.4.3.1 PID控制下极点变化

使用eig可以获取系统极点，通过abs函数获取极点的模长，分析几点是否处在单位圆内，极点均处在单位元内说明系统稳定。由图27可以看到未添加PID控制器前极点存在模长大于1的情况，即此时系统不稳定，添加完PID之后，极点均在单位圆内。

%PID控制添加前后极点分析

angle\_tf\_disc\_poles = eig(angle\_tf\_disc);

if(find(abs(angle\_tf\_disc\_poles) > 1))

fprintf("未加PID控制器下的力到角度的传递函数是不稳定的\n");

else

fprintf("未加PID控制器下的力到角度的传递函数是稳定的\n");

end

angle\_tf\_PID\_disc\_poles = eig(angle\_tf\_PID\_disc);

if(find(abs(angle\_tf\_PID\_disc\_poles) > 1))

fprintf("PID控制器下的力到角度的传递函数是不稳定的\n");

else

fprintf("PID控制器下的力到角度的传递函数是稳定的\n");

end



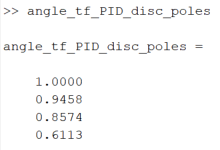
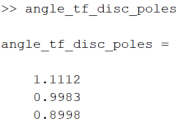


图 27 添加PID控制器前后极点变化

#### 3.4.3.2 PID控制下系统频率响应

使用margin函数可以获取系统的相位裕度、幅值裕度、相位穿越频率、幅值穿越频率，由函数输出结果可以得知，以上系统动态特性均有富余，即系统稳定。

%PID控制下的频域分析

[Gm, Pm, Wcg, Wcp] = margin(angle\_tf\_OL\_PID\_disc);

fprintf("相位裕度：%d\n 幅值裕度: %d\n 相位穿越频率: %d\n 幅值穿越频率:%d\n", Gm, Pm, Wcg, Wcp);

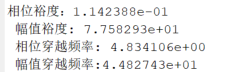


图 28 系统动态特性参数

### PID控制离散系统状态方程

使用并联PID对离散状态空间矩阵进行控制，其simulink框图如下：

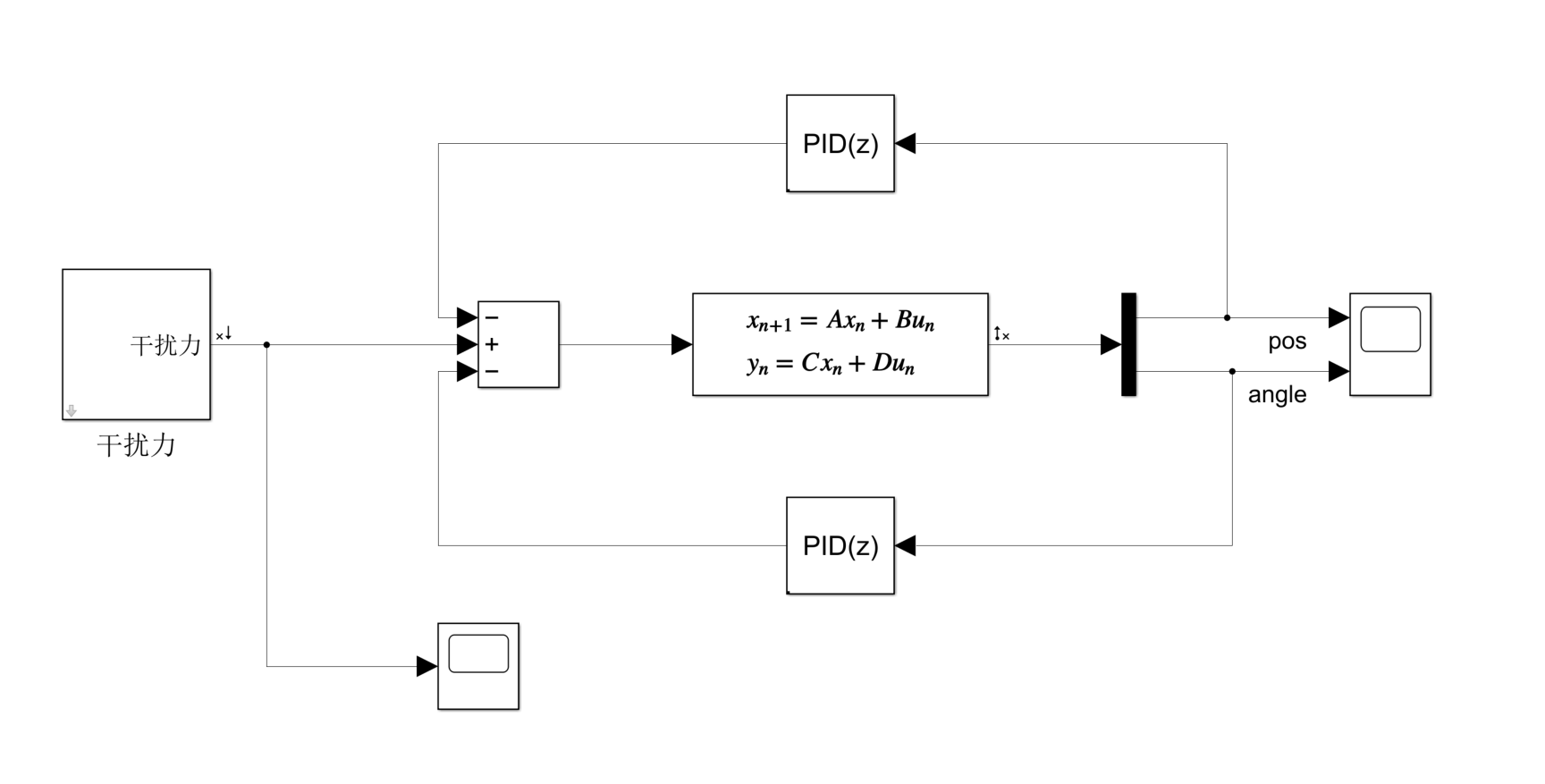


图 29 PID控制离散系统状态方程

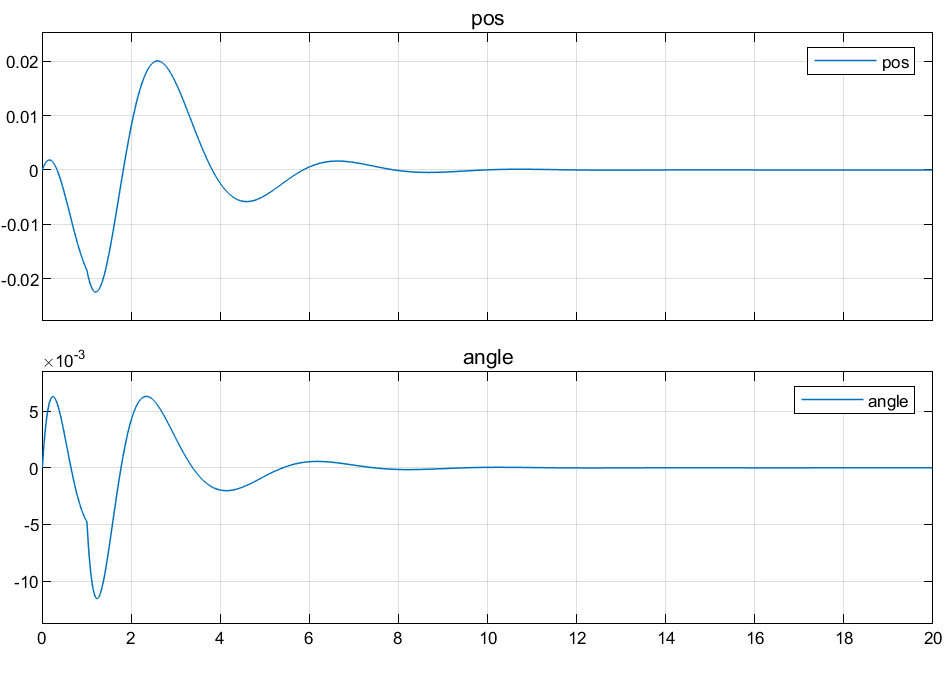


图 30 PID控制离散系统状态方程

由输出波形可以看到，稳定时间约为7S，且小车位置及角度均有较好控制。

3.5总结与心得

在这次实验中，我主要研究了如何使用PID控制器对倒立摆系统进行控制。通过离散时间模型，我模拟了系统的响应，并针对脉冲输入和单位阶跃输入进行了仿真。实验过程中，我深入分析了系统的控制性能参数，如上升时间、稳定时间、超调量和 稳态误差。

在实验初期，我首先对倒立摆系统进行了数学建模，将其转化为离散时间模型。这个过程加深了我对控制系统理论的理解。随后，我选择了PID控制器，并通过simulink PID模块的tunner获取并调整其三个参数——比例增益、积分增益和微分增 益，以优化系统的控制性能。

在仿真过程中，我分别对脉冲输入和单位阶跃输入进行了模拟。通过这两种输入方式，我观察了系统在不同输入下的响应特性。对于脉冲输入，我观察到了系统的快速响应和稳定的能力；而对于单位阶跃输入，系统的响应更加平滑，且超调量较小。

在分析控制性能参数时，我发现通过合适的参数调整，系统对于脉冲输入的上升时间和稳定时间都得到了较好的控制。而对于单位阶跃输入，虽然超调量有所增加，但系统的稳态误差非常小。这些分析让我对PID控制器在倒立摆控制中的应用有了更加深入的了解。

通过PID控制器实验，我不仅掌握了如何使用PID控制器对倒立摆系统进行控制，还学会了如何分析系统的控制性能参数。这次实验加深了我对控制系统理论的理解，也提高了我的实际操作和问题解决能力。

# 基于状态观测器的倒立摆系统最优控制

## 问题描述

考虑观测位置和角度时传感器存在噪声，控制器施加力矩时存在噪声，如图5所示。为系统设计卡尔曼最优滤波器，并采用LQR最优控制器（也可以是其他控制器，但不能使用PID控制器），对倒立摆的离散时间模型进行控制，并给出脉冲输入和单位阶跃输入下系统响应仿真曲线；分析上升时间，稳定时间，超调量，稳态误差等控制性能参数。

墙上的钟表

描述已自动生成

图 31 收白噪声干扰的对象

## 理论推导

### LQR控制算法

[LQR](https://so.csdn.net/so/search?q=LQR&spm=1001.2101.3001.7020)(Linaer Quadratic Regulator)，即线性二次型调节器，是一种现代控制理论中设计状态反馈控制器(State Variable Feedback,SVFB）的方法。

对于一个系统，假设我们要设计一个线性反馈控制器，则此时状态方程可以写为：，由于让系统稳定的条件是矩阵的特征值的实部均为负数，因此可以手动选择几个满足上述条件的特征值，然后反解出K，从而得到控制器。为了选择特征值，使得控制器的控制效果最好，定义一种代价函数(cost function)J：



其中，Q 和R是两个对角参数矩阵，分别决定了状态向量x和输入向量u的重要性。J是一个二次型函数，需在满足系统稳定的前提下 ，设计合适K，使代价函数J最小。

考虑一个双变量系统，即，我们希望设计的控制器可以表示为，此时代价函数可以写成：



设置，，代价函数可写成：



如果令，则状态变量在代价函数中的占比就更大，这意味着如希望代价函数最小，必须更小。又因为越大意味着闭环系统矩阵的极点在s平面中更偏左，因此收敛得更快。若 ，则希望输入量收敛得更快，也就是以更小得代价实现系统稳定，通常意味着更加节省能量。

因为对象是线性的，并且代价函数是二次型，因此这种选择K设计状态反馈控制器以最小化代价函数J 的方法被称为“线性二次型调节器(LQR)”。Regular意味着这种反馈的功能是将系统状态调节为0，这在一些追踪问题中会受到约束，有时更希望稳定状态是给定的非零值，即目标状态。

求解K使得J最小，定义一个辅助常量矩阵P，使得



代入代价函数得：



由于假设t趋近于无穷时，x趋近于0，则等于0，所以



即可得J与P和系统初始状态有关，让P最小也就是让代价函数最小。即找出满足条件的P。将式左边的微分项展开可得：



又,设，则上述方程可以化为：



对于任意x均满足，则说明括号中等式恒为0，带入得：



设，则变为：



获得Algebraic Riccati Equation (ARE)方程。ARE是一个矩阵二次方程，对于给定的(A,B,Q,R)可以解出辅助矩阵P 。之后，优化反馈控制器的K 就可通过式( 10 )得出，代价函数的最小值可以用式( 7 )得到。而在MATLAB中，使用LQR可以使用lqr(A,B,Q,R)函数获得矩阵K。

### 卡尔曼滤波

#### 4.2.2.1卡尔曼滤波简介

卡尔曼滤波是一种在动态系统中用于估计状态的方法。它利用系统的动态模型和测量数据来估计系统的状态，同时考虑测量误差和过程噪声，以此来提高状态估计的精度。

卡尔曼滤波的基本思想是通过结合系统的动态模型和实际的测量数据，来提供对系统状态的最优估计。它主要包括两个阶段：预测和更新。在预测阶段，根据系统的动态模型和当前状态的估计，推断系统的下一个状态。这一步考虑了系统的动态特性和可能存在的过程噪声。在更新阶段，结合实际的测量数据和预测的状态估计，计算系统的最优状态估计。这一步考虑了测量数据的精度和可能存在的测量误差。过交替进行预测和更新，卡尔曼滤波可以提供对系统状态的最优估计，同时考虑了系统的动态特性和测量数据的精度，从而在动态系统中实现较精确的状态估计。

#### 4.2.2.2 卡尔曼公式推导

连续线性定常系统下的状态空间表达式为：



但是在实际工程之中我们并非总能得到一个连续的系统，有可能是离散的，此时更新状态空间表达式为：



由于我们的模型建立可能不准确，测量也存在误差，所以式中和是过程噪音(process noise)和测量噪音(measure noise)。现在要做到的就是根据以上方程估计一个。

先定义先验估计：



根据输出方程，可以根据得到当前的状态：



则获得一个测出来的值和一个算出来的值，即测量值和预测值，综合它们来获得真值， 即数据融合得到：



令得：



此时，的取值范围是，当取0的时候，即相信预测值，当取的时候， ，即相信测量值。这时问题转换为选取卡尔曼增益，使得后验的更接近于实际值。引入误差：



后验的越接近实际值，协方差矩阵P的迹越小。



则问题转换为：为何值时，有最小值



计算协方差矩阵P



使得协防差矩阵的迹最小：



令可得：



从而得到卡尔曼增益为：



由推导的结果可以分析，当测量误差较大时， 接近于0，即根据(3)式，后验

更相信先验的，而当测量误差较小时， ，根据(3)式，后验更相信测得的结果。因此，卡尔曼增益推导符合理论。

而卡尔曼增益公式中，唯一不知道的是 ，其求解方式如下：



其中先验误差为：



​所以先验协方差矩阵为：





将卡尔曼增益带入上式的，可将项消除，则可得：



至此，得到了卡尔曼滤波的五条重要公式：

①预测：

先验估计：

先验误差协方差：

②矫正：

卡尔曼增益：

后验估计：

更新误差协方差：

## simulink构建基于卡尔曼滤波器下的倒立摆LQR控制模块

### 仿真框图

基于卡尔曼滤波器下的倒立摆LQR控制模块包含了“力干扰模块”、“系统状态空间表达式模块”、“卡尔曼滤波”模块、“LQR计算力输入”模块，通过Switch模块可以选择是否进行卡尔曼滤波， simulink仿真框图如图29所示



图 32 基于卡尔曼滤波器下的倒立摆LQR控制模块框图

### 高斯白噪声设计

对位移测量值添加高斯白噪声，其方差为 (0.02m)^2，相当于测量误差为2cm；而对角度测量值添加高斯白噪声，其方差为 (0.2°)^2，即相当于测量误差为0.5°。假设速度和角速度的测量误差为无穷小，设置为1e-6。MATLAB代码如下：

%%

angle\_to\_radian = pi/180;

Measure\_noise\_position = 0.02^2;%位置测量存在2cm误差

Measure\_noise\_speed = 1e-6;%速度测量误差

Measure\_noise\_angle = (0.2\*angle\_to\_radian)^2;%角度测量存在0.2°误差

Measure\_noise\_angle\_vel = 1e-6;%角速度测量误差

%过程噪声方差

Wk = 0.001;

%测量噪声方差

Vk = [Measure\_noise\_position, ...

Measure\_noise\_speed, ...

Measure\_noise\_angle, ...

Measure\_noise\_angle\_vel];

### 系统状态空间表达式模块设计

“系统状态空间表达式模块”采用simulink封装功能，其子系统框图如下，在输出处加入过程噪声，同时在输出处引入测量噪声，并通过四路信号输出，通过交互页面可以实时修改状态空间相关矩阵，同时可以为系统引入在MATLAB定义的过程噪声及测量噪声。

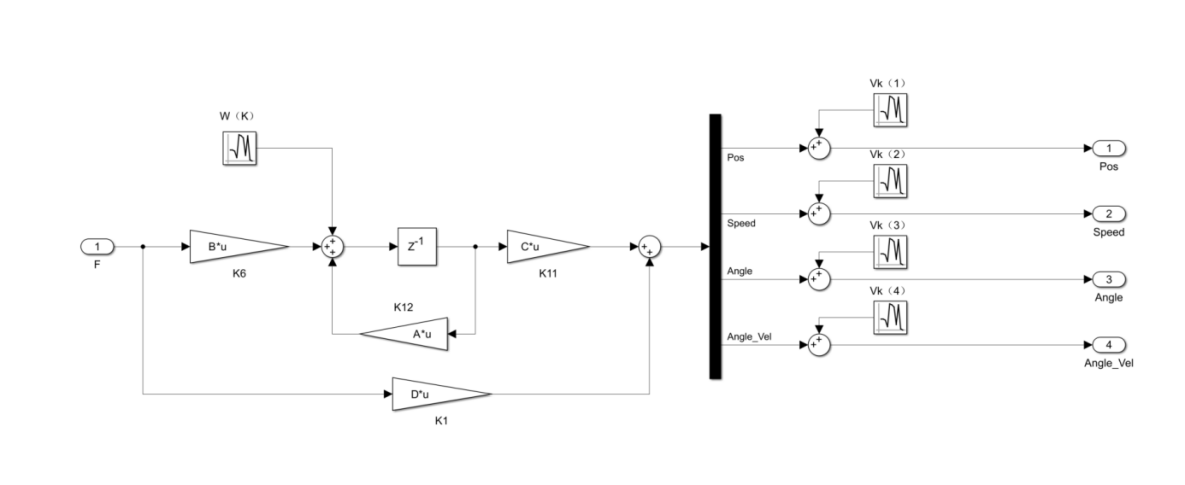


图 33 系统状态空间表达式模块框图



图 34 系统状态空间表达式模块交互页面

### LQR控制器设计

#### 4.3.4.1 利用LQR建立闭环系统

在LQR控制器中，Q矩阵是状态权重矩阵，维度与A矩阵一致，代表了对系统状态变量的重视程度。不同的 Q 矩阵会导致控制器对状态变量的重视程度不同，Q矩阵为对角阵，对角线上值越大，表明希望对应的状态变量收敛速度越快，从而影响系统的响应速度、稳定性和能耗等方面的性能。R 矩阵是控制输入权重矩阵，维度与U矩阵一致,代表了对控制输入的重视程度。不同的 R 矩阵会导致控制器对控制输入的重视程度不同，从而影响系统对控制输入的调节速度、稳定性和能耗等方面的性能。

%设置LQR矩阵

Q = diag([400 10 500 10]);

R = 0.4;

%使用LQR计算K

[K, P] = dlqr(A\_disc,B\_disc,Q,R);

%计算加入反馈后的状态空间方程矩阵

A\_disc\_feed = A\_disc - B\_disc \* K;

B\_disc\_feed = B\_disc;

C\_disc\_feed = C\_disc;

D\_disc\_feed = D\_disc;

sys\_feedback\_LQR = ss(A\_disc\_feed, ...

B\_disc\_feed, ...

C\_disc\_feed, ...

D\_disc\_feed);

#### 4.3.4.2 分析闭环系统性能

建立具有LQR控制的系统之后，对系统进行性能分析，使用eig函数获取在LQR控制下，系统的极点分布，发现极点均在复平面左半边，同时，使用margin函数分析系统响应频率，由输出结果可以得知，添加LQR控制器后，系统相位裕度、幅值裕度、相位穿越频率、幅值穿越频率均处在理论范围，说明在LQR控制器能够使得系统稳定。

%% 系统性能分析

%系统在离散和连续状态下的极点和特征方程

state\_space\_poles\_disc\_feedback\_LQR = eig(A\_disc\_feed);

characteristic\_equation\_con = vpa((S - state\_space\_poles\_disc\_feedback\_LQR(1))\*...

(S - state\_space\_poles\_disc\_feedback\_LQR(2))\* ...

(S - state\_space\_poles\_disc\_feedback\_LQR(3))\* ...

(S - state\_space\_poles\_disc\_feedback\_LQR(4)), 4);

disp('LQR控制下离散状态空间表达式特征方程为：');

disp(characteristic\_equation\_con);

if(find(real(state\_space\_poles\_disc\_feedback\_LQR) > 1))

disp('LQR控制下离散状态下的系统不稳定：');

else

disp('LQR控制下离散状态下的系统稳定：');

end

%LQR控制下的频域分析

[Gm, Pm, Wcg, Wcp] = margin(angle\_tf\_OL\_PID\_disc);

fprintf("相位裕度：%d\n 幅值裕度: %d\n 相位穿越频率: %d\n 幅值穿越频率:%d\n",...

Gm, Pm, Wcg, Wcp);



图 35 LQR闭环系统动态特性

#### 4.3.4.3 LQR闭环系统在特定干扰信号下的响应

使用simulink模块封装功能，设计力干扰模块，通过交互页面传递对应信号给input变量，在子系统中通过input控制输出对应信号，特定信号包含：①无：没有干扰，input为0 ②正弦：正弦信号，input为1 ③脉冲：脉冲信号，input为2

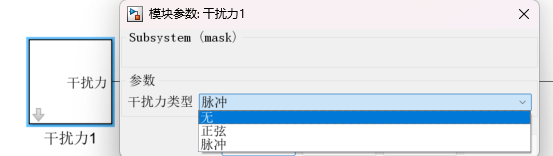


图 36 干扰力模块

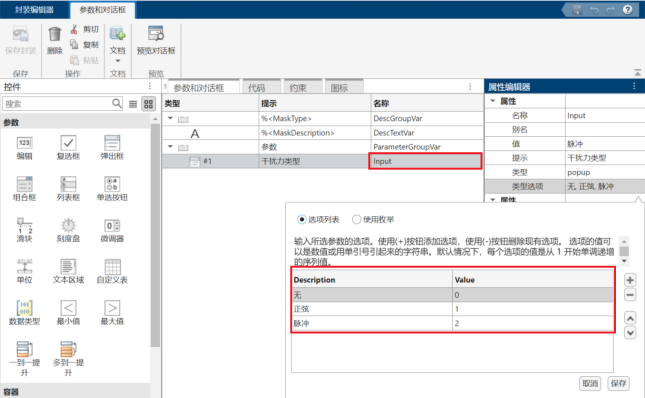


图 37 干扰力模块设计

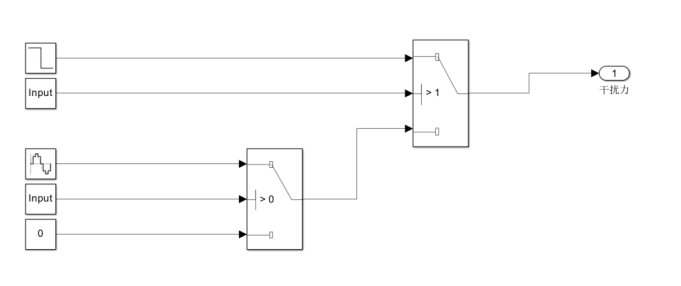


图 38 干扰力模块子系统框图

**（1）正弦信号干扰下**

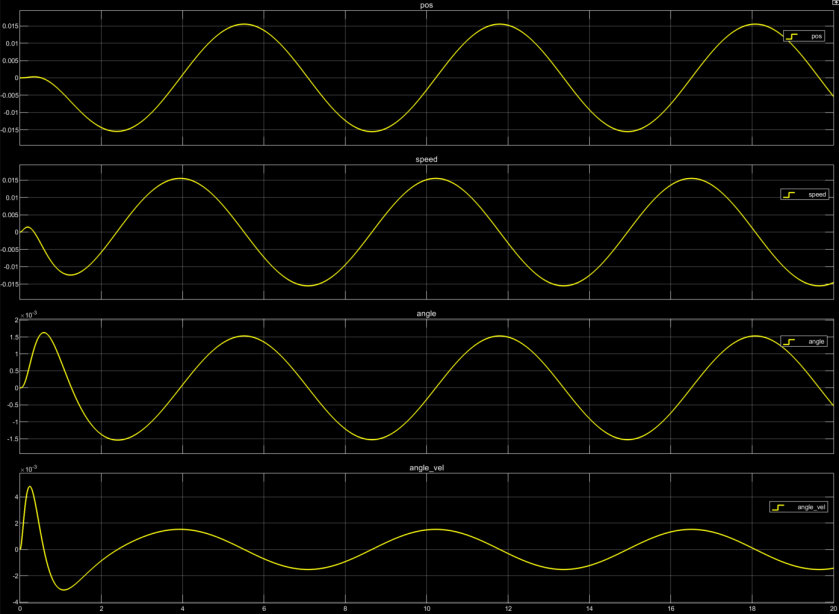


图 39 正弦信号干扰下系统响应

给定峰值为1N的正弦干扰力后，其响应波形如下，可以看到，超调量为0.0015rad，约0.84°，存在一定的震荡，但是幅度比较小，在可接收范围内，接近稳定。

**（2）脉冲信号干扰下**

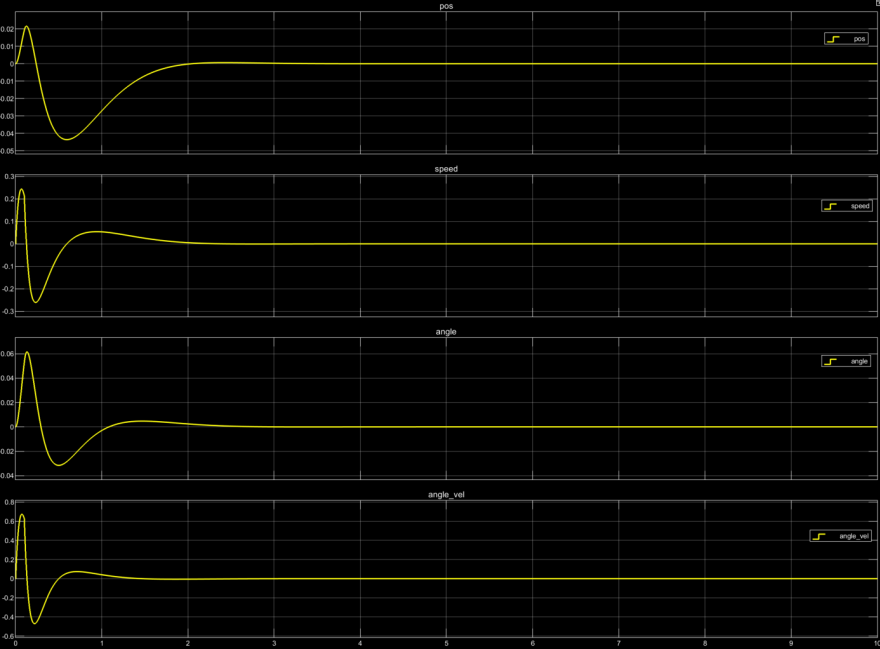


图 40 脉冲信号干扰下系统响应

给定10N，持续0.1S的干扰力后，其响应波形如下，可以看到，稳定时间为1S。相比于PID控制器，收敛速度更快，且收敛过程幅度更小

#### 4.3.4.4 目标状态设定

先前设计的闭环系统，其期望状态为X0 = [0; 0; 0; 0]，但是大部分时刻更多希望能够指定特定的目标状态，因而需要对系统进行一定修改，使得系统能够收敛到设定目标状态。将状态方程修改为：

****

即当目标状态未达到设定的时，反馈将一直存在直到收敛于目标状态，其MATLAB代码如下：

%%

%设置离散时间及步长

T = 0 : Time\_step : Time;

%设置系统初始状态

X0 = [5; 0; 0; 0];

X\_tar = [2; 0; 0; 0];

if(X\_tar(2) ~= 0)

K(1) = 0;

end

X\_err = X\_tar - 0;

%设置输入

U = ones(size(T));

% U(1) = 0;

U\_LQR = K \* X\_err \* ones(size(T));

%使用lsim求离散系统在U输入下的响应

[Y\_op, X\_op] = dlsim(A\_disc, B\_disc, C\_disc, D\_disc, U, X0);

[Y\_LQR\_cl, X\_LQR\_cl] = dlsim(A\_disc\_feed, B\_disc\_feed, C\_disc\_feed, D\_disc\_feed, U\_LQR, X0);

if draw\_state

%绘制图案

figure

subplot(2,1,1);

plot(T, X\_LQR\_cl); % 闭环响应

legend('x1', 'x2', 'x3', 'x4');

xlabel('Time');

ylabel('State Response');

title('LQR控制下系统响应');

axis([0 15 -10 15]);

subplot(2,1,2);

plot(T, X\_op); % 开环响应

legend('x1', 'x2', 'x3', 'x4');

xlabel('Time');

ylabel('State Response');

title('无LQR控制下系统响应');

axis([0 15 -10 15]);

end

将系统初始状态设置为即期望位置为2，设置输入为，使用dlsim函数求解系统阶跃响应，结果如下：

****

图 41 LQR控制下系统响应

****

图 42 无LQR控制下系统响应

可以看到、在添加LQR控制器下，系统收敛，且由x1曲线可以得知，x1由初始的2收敛到期望目标位置5，需要注意的是，当期望速度不为0时，需要将位置的增益设置为0，因为在期望速度不为0情况下，位置是实时改变的，位置不会收敛于某个值，因而需要将位置反馈增益设置为0，即：

if(X\_tar(2) ~= 0)

K(1) = 0;

end

将速度设置为，获得系统响应如下，可以看到位置x1实时以1m/s速度前进，且速度收敛到2，角度和角速度收敛到0

****

图 43 LQR控制下系统响应

### 卡尔曼滤波器设计

#### 4.3.5.1 使用simulink自带卡尔曼滤波元件进行设计

（1）卡尔曼滤波器参数设定

设置卡尔曼滤波器为离散时间型，A、B、C、D矩阵为系统离散化后的矩阵系数。初始状态设置为[0 0 0 0]，Q矩阵为过程噪声，为模型的误差程度；R矩阵表示测量噪声，为测量值下的误差程度。Simulink下卡尔曼滤波器模块设置如下图所示



图 44 卡尔曼滤波器参数设定

Q、R矩阵设置如下：

%%

angle\_to\_radian = pi/180;

Measure\_noise\_position = 0.02^2;%位置测量存在2cm误差

Measure\_noise\_speed = 1e-6;%速度测量误差

Measure\_noise\_angle = (0.2\*angle\_to\_radian)^2;%角度测量存在0.2°误差

Measure\_noise\_angle\_vel = 1e-6;%角速度测量误差

%模型误差

Q\_kalman = 0.000001;

%测量误差

R\_kalman = [Measure\_noise\_position, ...

Measure\_noise\_speed, ...

Measure\_noise\_angle, ...

Measure\_noise\_angle\_vel];

（2）卡尔曼滤波仿真

将干扰力输入设置为脉冲信号，同时将switch开关选择到卡尔曼滤波，使用卡尔曼滤波后的数据做反馈，这里只使用了位置及角度的卡尔曼滤波。各个状态变量的响应曲线；表示加了噪声后的状态变量曲线，由响应可以的字，卡尔曼滤波器能够滤除大量噪声，获取平滑的测量值，使得LQR控制器能够更加稳定控制系统。



图 45 将卡尔曼滤波接入反馈

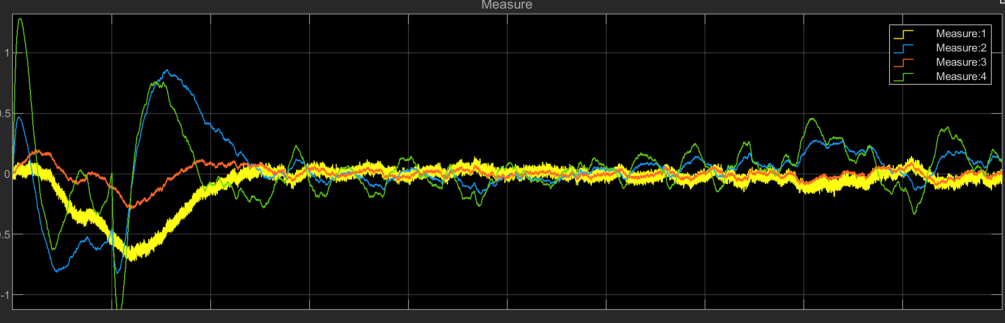


图 46 卡尔曼滤波前波形

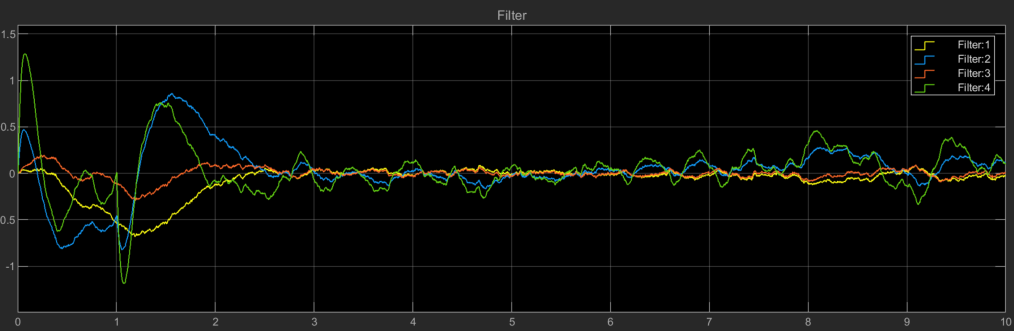


图 47 卡尔曼滤波后波形

使用scope单独查看位置及角度滤波状况，蓝色均为滤波前的波形图，黄色为滤波后的波形图，由波形图可知，相比于滤波前，滤波后效果更为平滑，达到预期效果。

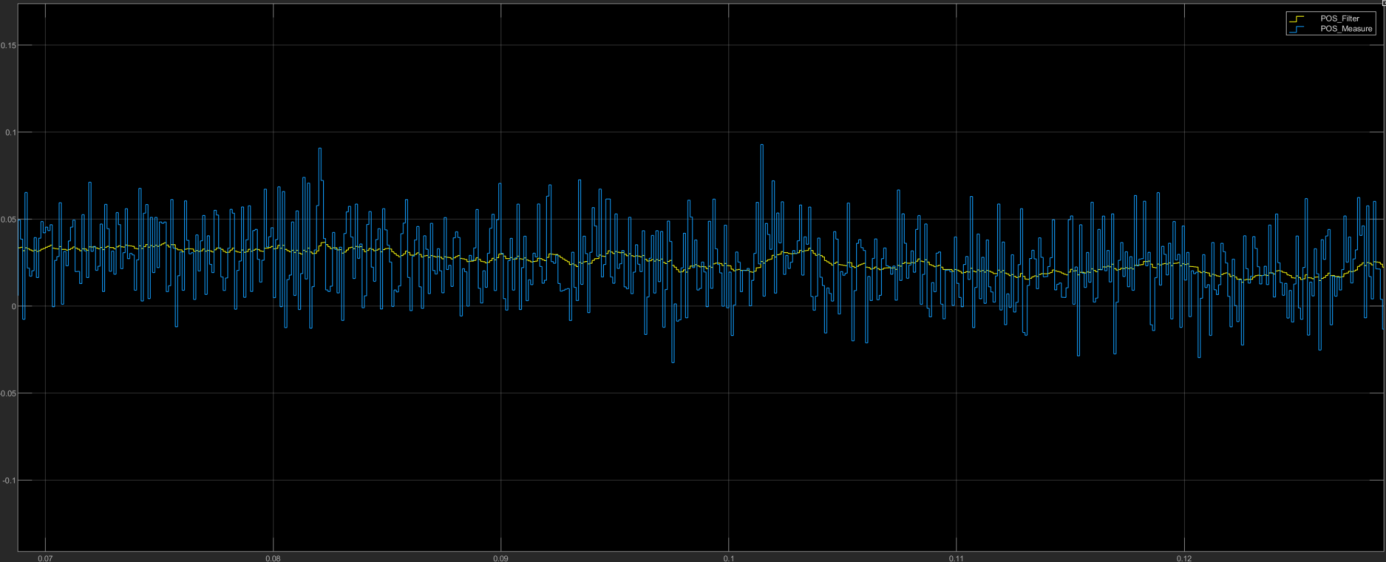


图 48 位置状态变量滤波前后对比

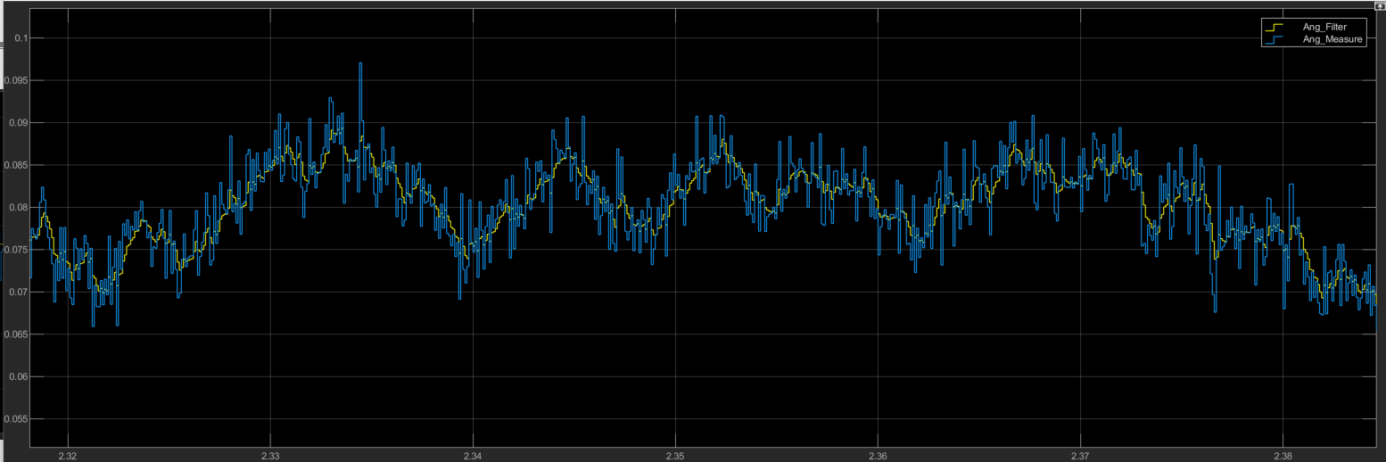


图 49 角度状态变量滤波前后对比

#### 4.3.5.2 自主搭建卡尔曼滤波模块

（1）自定义卡尔曼滤波器理论公式

前面对卡尔曼滤波理论推导得到卡尔曼滤波的五条重要公式：

①预测：

先验估计：

先验误差协方差：

②矫正：

卡尔曼增益：

后验估计：

更新误差协方差：

根据以上公式自主搭建卡尔曼滤波模块：

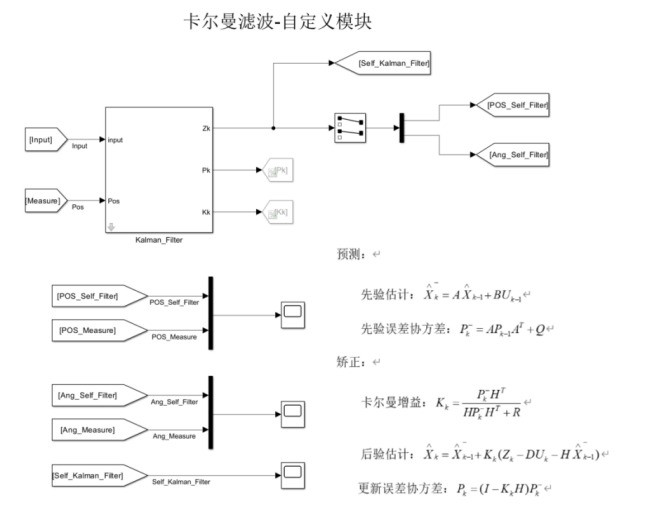


图 50 自定义卡尔曼滤波模块

（2）自定义卡尔曼滤波器模块内部设计

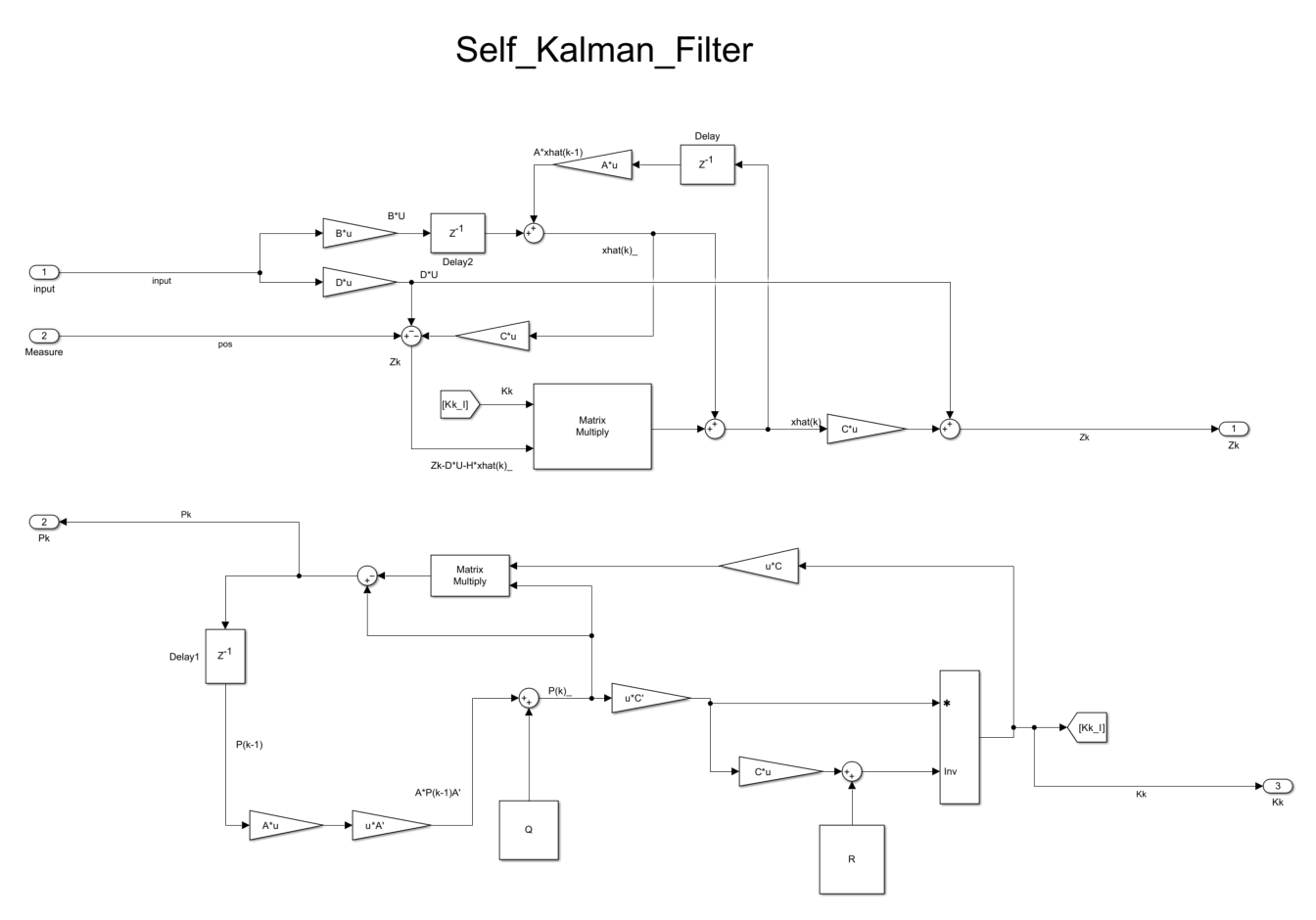


图 51 自定义卡尔曼滤波器模块内部设计

模块包含两输入三输出，输入分别为：Input、Measure，输出为：Zk、Pk、Kk，模块分成两部分，上面半部分对应卡尔曼滤波的前两个公式，也就是预测值更新的过程，下面部分都是卡尔曼增益更新的迭代过程，卡尔曼滤波的五个等式参数非常多，但大多数参数都在不断的循环更新中，整个系统真正能调整的参数只有三个，就是系统开始循环的初始值P0、X0，需要初始化。还有就是最重要的两个参数，过程误差和测量误差的协方差，以及系统的状态空间表达式，因而在封装模块时，设计交互页面输入以上参数，交互页面设计如下，可以通过点击模块左下角的箭头查看模块内部设计。

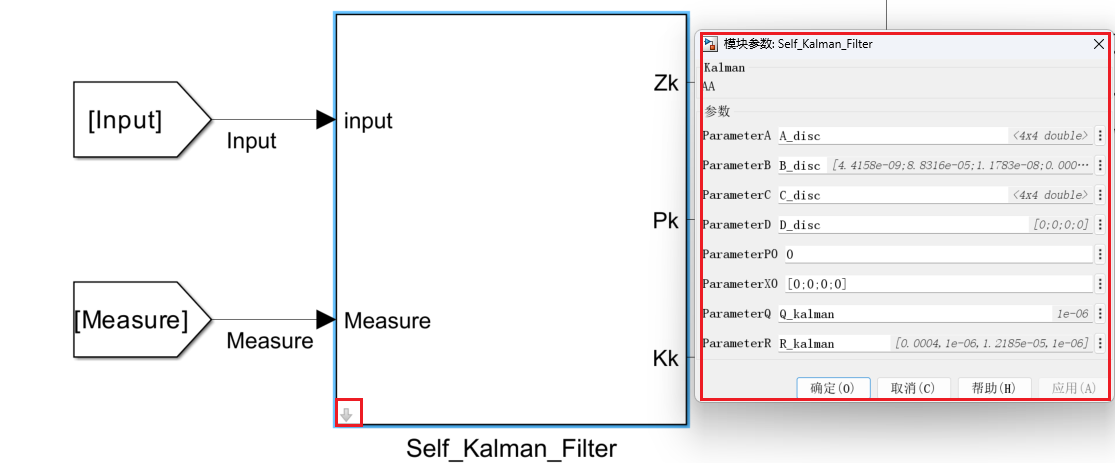


图 52 自定义卡尔曼滤波模块交互界面

（3）自定义卡尔曼滤波器滤波效果

调整switch模块，将系统反馈信号接入自定义卡尔曼滤波模块滤出的信号



图 53 接入自定义卡尔曼滤波模块

各个状态变量的响应曲线如下：图 表示加了噪声后的状态变量曲线，可见卡尔曼滤波器确实能够滤除大量噪声，获取平滑的测量值，让LQR控制器更加精准的控制倒立摆系统。

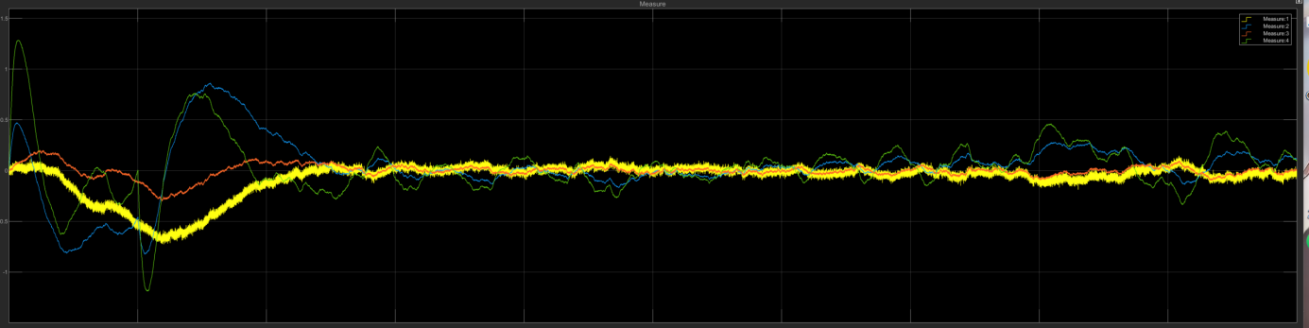


图 54 自定义卡尔曼滤波前波形

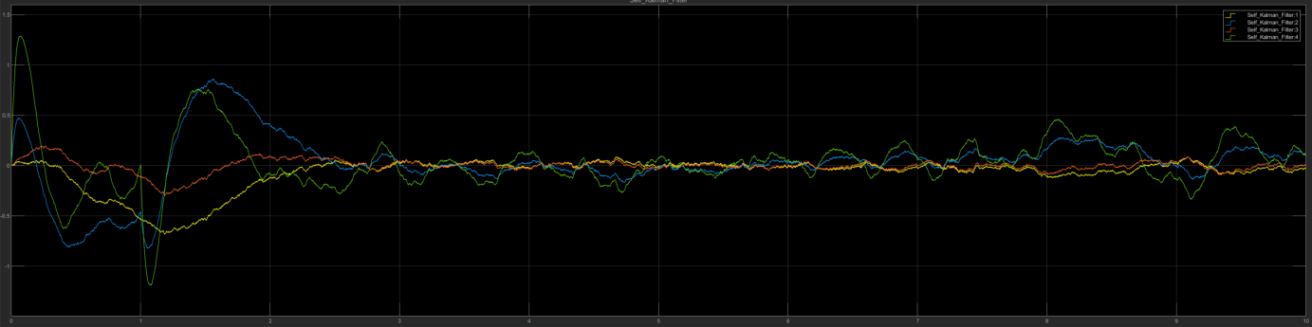


图 55 自定义卡尔曼滤波后波形

使用scope单独查看位置及角度滤波状况，蓝色均为滤波前的波形图，黄色为滤波后的波形图，由波形图可知，相比于滤波前，滤波后效果更为平滑，达到预期效果。自定义卡尔曼滤波器实现期望功能，能够较好的进行滤波。



图 56 位置状态变量滤波前后对比

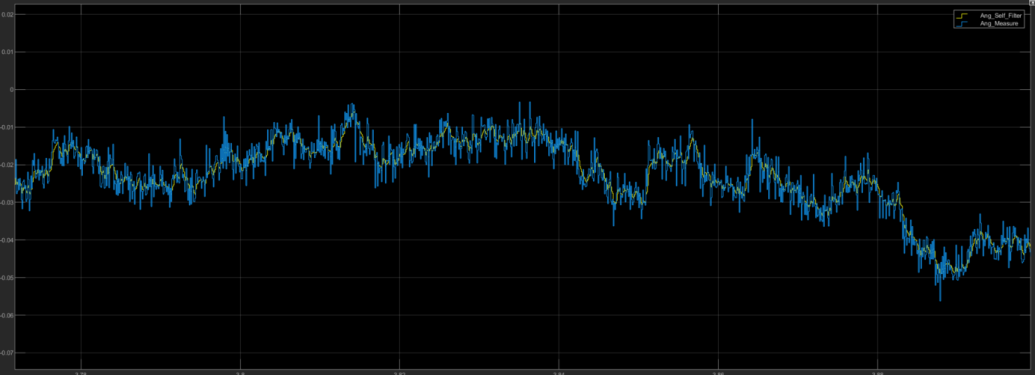


图 57 角度状态变量滤波前后对比

# 基于卡尔曼滤波及LQR进行simscape刚体仿真

## simscape多刚体仿真模块搭建

在“多刚体仿真模块”中，使用了“Rigid Transform”坐标转换、“Prismatic Joint”滑动关节、“Revolute Joint”旋转关节，“Spherical Solid”球体、“Brick Solid”、长方体体等模块，模块各个参数依据“动力学分析与连续时间状态空间模型建立”部分给定参数进行设立及连接，同时将滑动关节的位置、速度，旋转关节角度、角速度输出、为系统的响应输出。输入为给滑动关节的力。

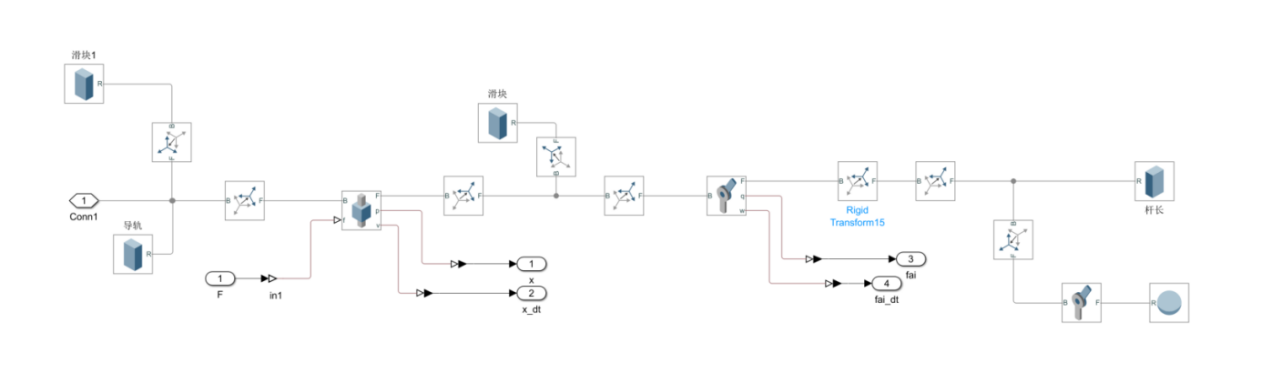


图 58 simscape多刚体仿真模块

其中最主要的是“Prismatic Joint”滑动关节和“Revolute Joint”旋转关节，滑动关节能够沿着指定直线滑动，即模拟倒立摆小车前进后退，且滑动关节能实时输出滑动的位置和速度，即输出小车位置，旋转关节能绕锚点进行旋转，即模拟倒立摆摆杆尾部关节，能够实时输出旋转关节的位置和速度，即对应摆杆的角度和角速度。

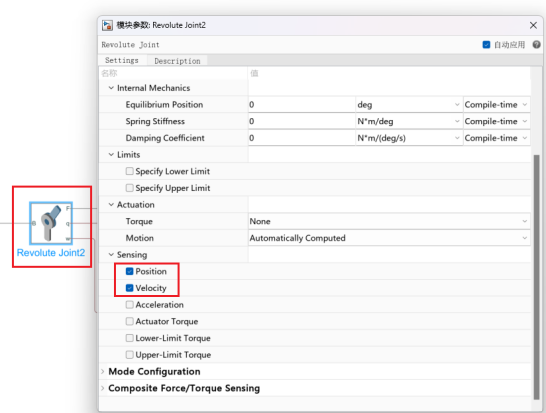
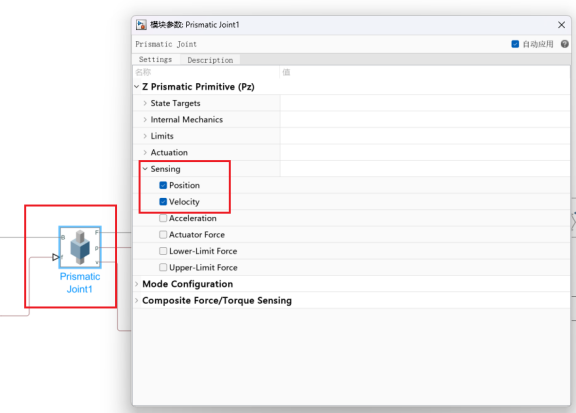


图 59 滑动关节和旋转关节设置

simscape多刚体仿真模型如下，黑色为滑轨，蓝色为滑块、棕色为摆杆、绿色为小球、浅绿色为滑轨中心，即期望目标值，为倒立摆最终回到的位置。

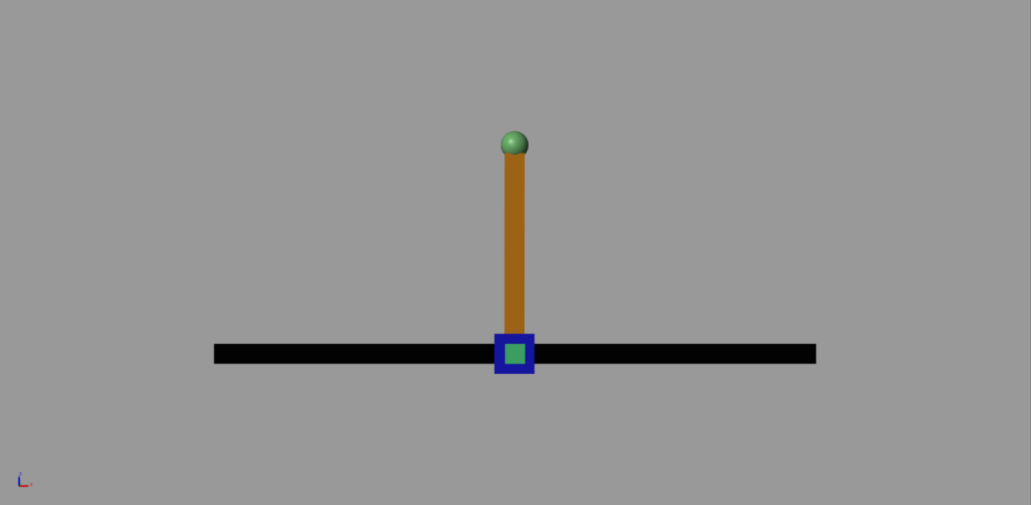


图 60 simscape多刚体仿真模块效果示意图

将该模块替换原先的状态空间表达式模块，进行simscape多刚体仿真

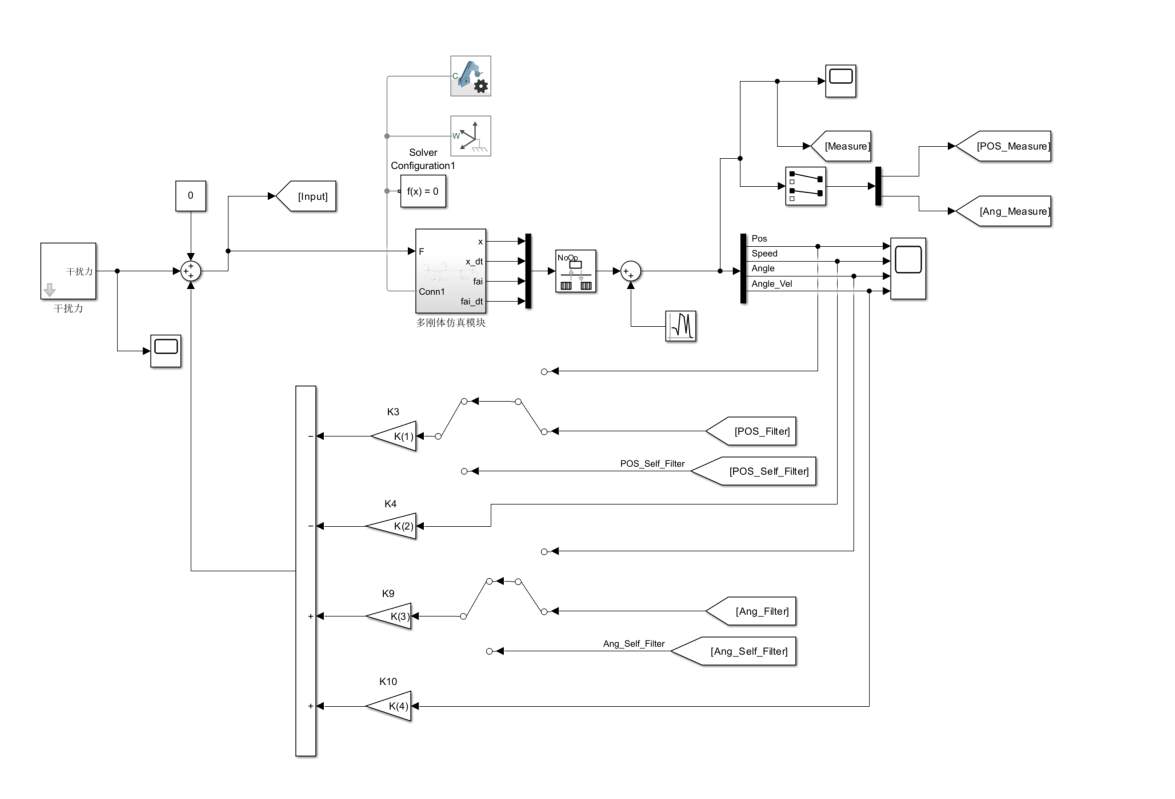


图 61 simscape多刚体仿真实验框图

## simscape多刚体仿真结果分析

运行simulink文件，在MATLAB将弹出对应的动画窗口，运行结果如下所示，下图每个画面间隔0.5S，可以看到倒立摆在第六张图时开始稳定，稳定时间为是3S。

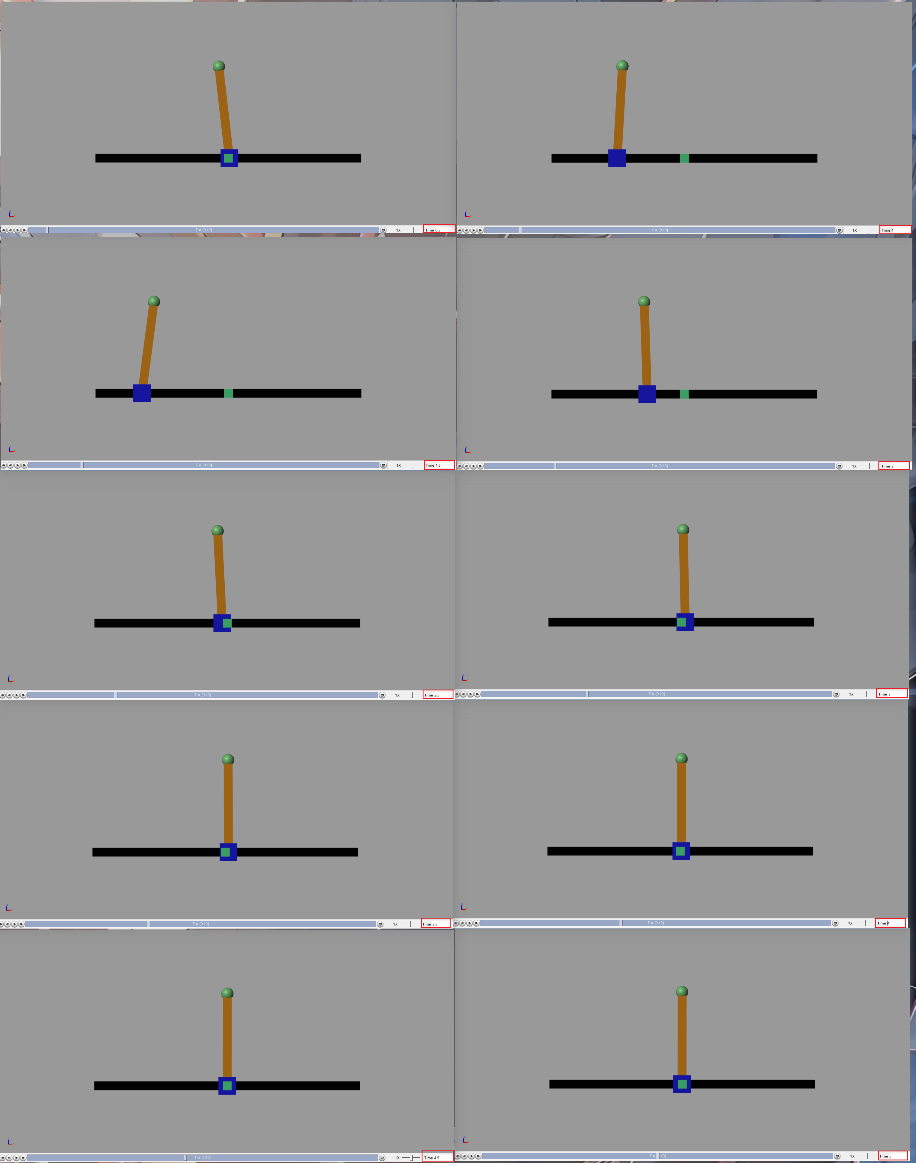


图 62 simscape多刚体仿真结果分析

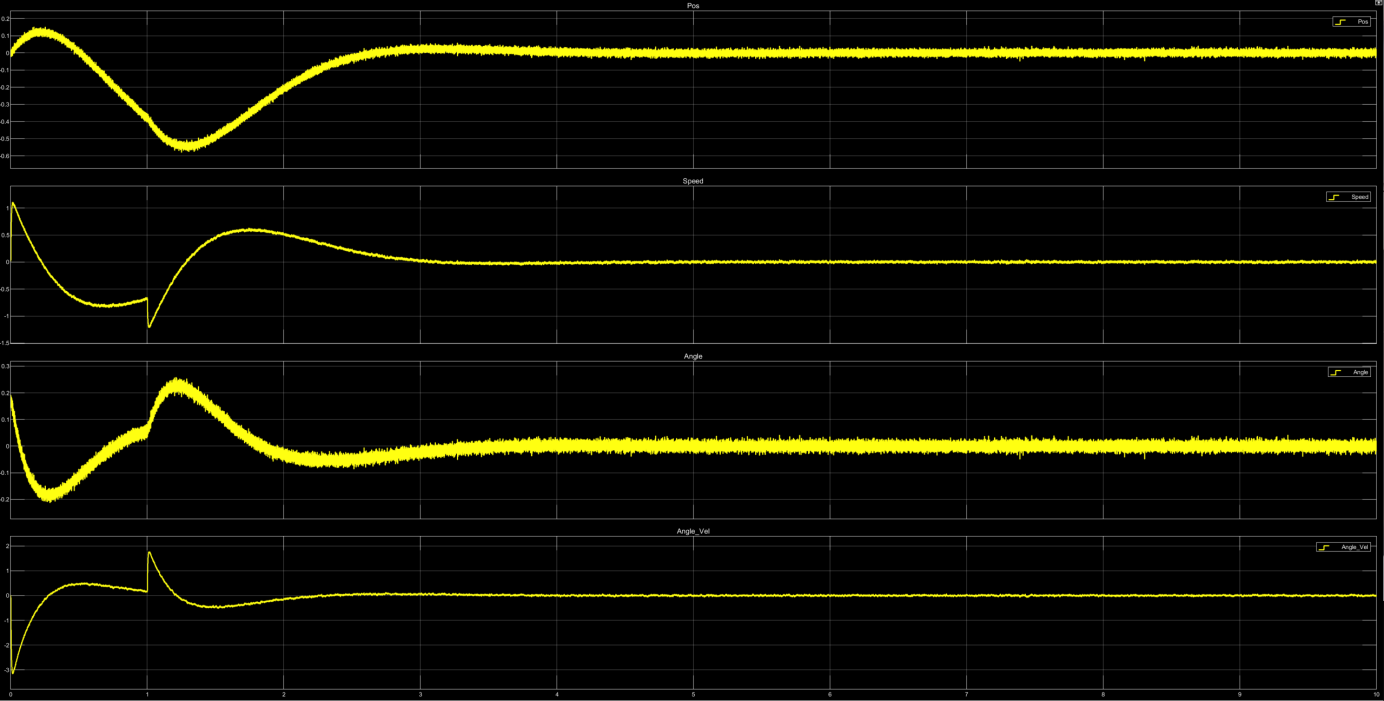


图 63 simscape多刚体仿真输出波形

通过scope查看波形，可以看到系统也是在3S左右达到稳定，达到预期效果

将系统接入系统自带卡尔曼滤波模块

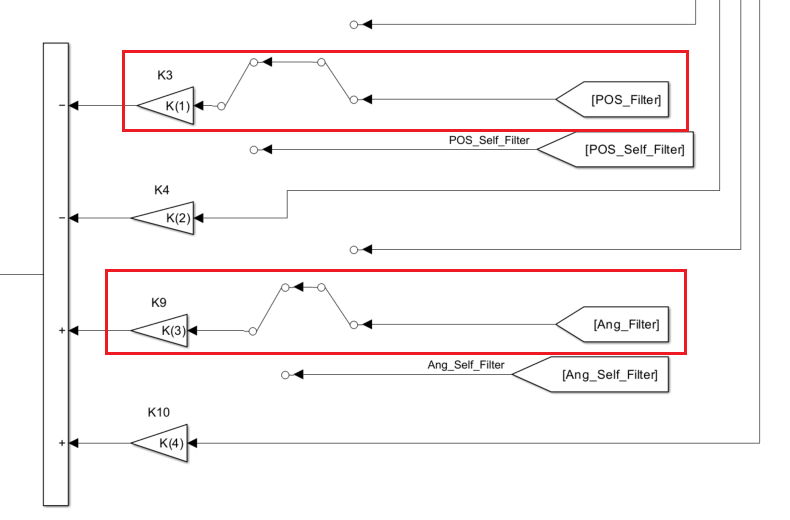


图 64 接入卡尔曼滤波模块

其运行结果如图62~图65所示，卡尔曼滤波实现能够较好的滤除噪声，获得较平滑的状态响应曲线，减少噪声对系统的影响，减少系统在原点的震荡。使用自定义卡尔曼滤波效果类似，遂不在此进行过多赘述。

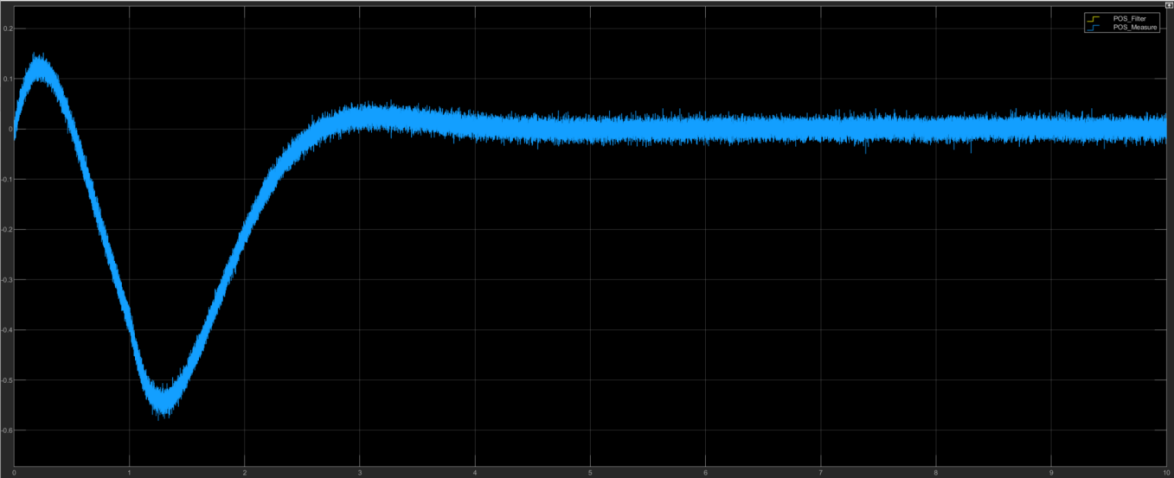


图 65 位置状态变量滤波前波形

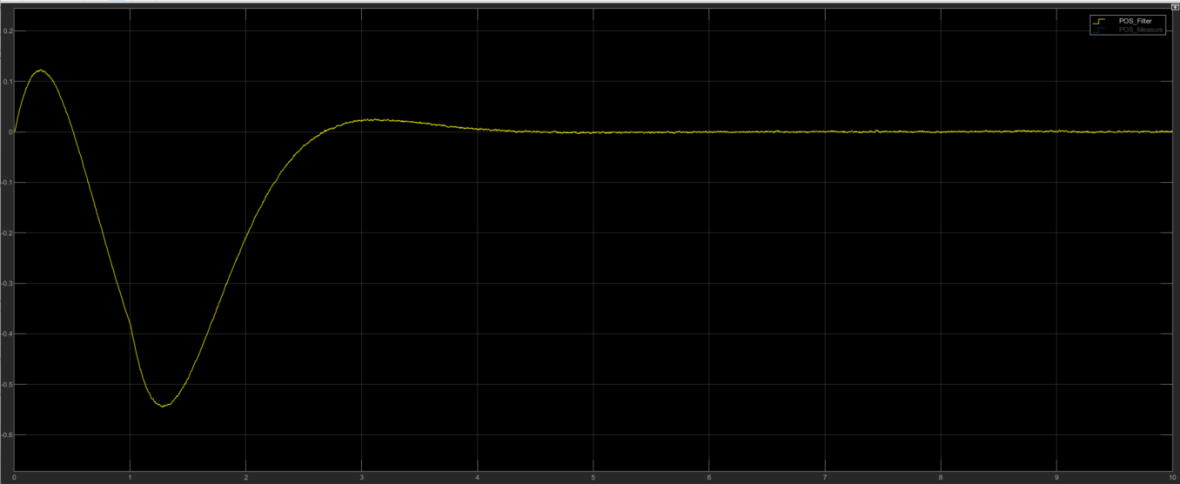


图 66 位置状态变量滤波后波形

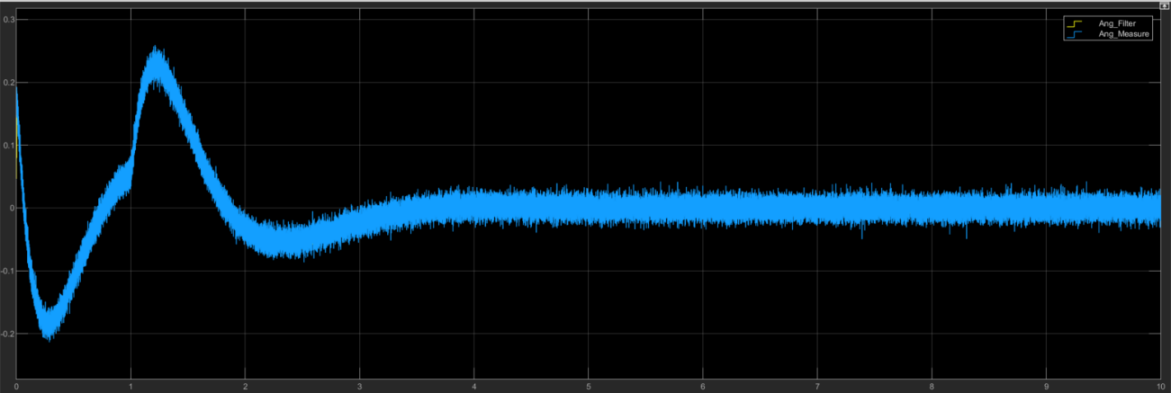


图 67 角度状态变量滤波前波形

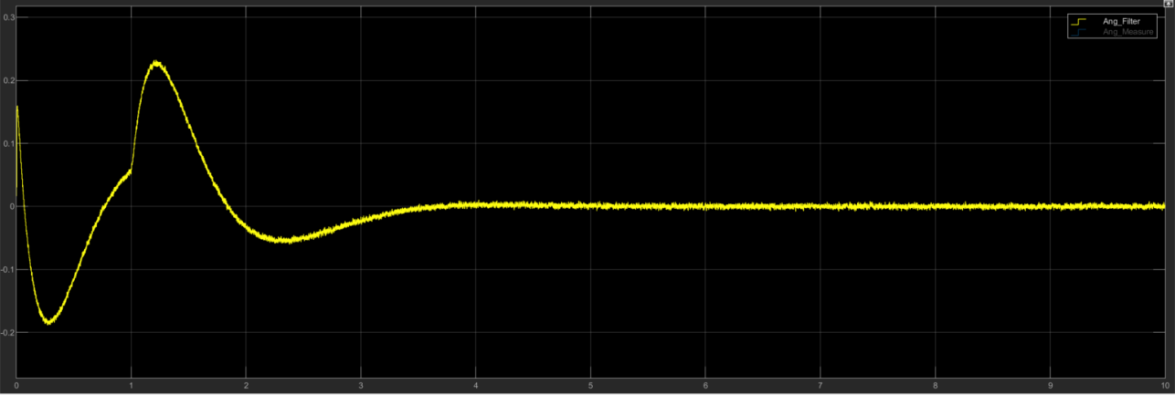


图 68 角度状态变量滤波后波形

# 基于LQR编写倒立摆系统上位机

使用MATLAB代码编写或者Simulink进行仿真均能实现对倒立摆进行控制，但是修改参数都得找到相应的代码或者模块设置相关参数，若是工程量比较大，一个参数多处使用，每次修改得花费很时间寻找参数，因而通过MATLAB的APP-Designer功能，可以将各个模块进行封装，并将接口显示在前端页面，便可以快速进行参数调整，同时，可以将响应结果以图像化方式显示到前端页面，使得初学者能够更直观看到参上修改对系统的影响。

此外，使用MATLAB的APP-Designer还可以将上位机打包成APP，使用者能直接进行下载，并在如图66所示进行安装，便可不了解内部实现机制下进行LQR实验，便于入门者快速理解LQR不同Q、R矩阵对系统收敛带来的影响。

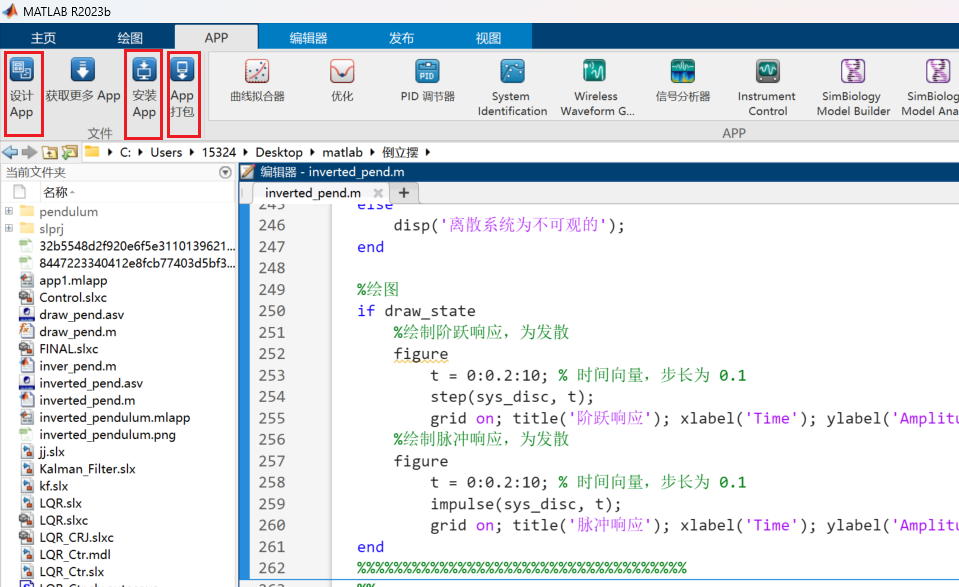


图 69 APP打包及安装

其大致框图如下：

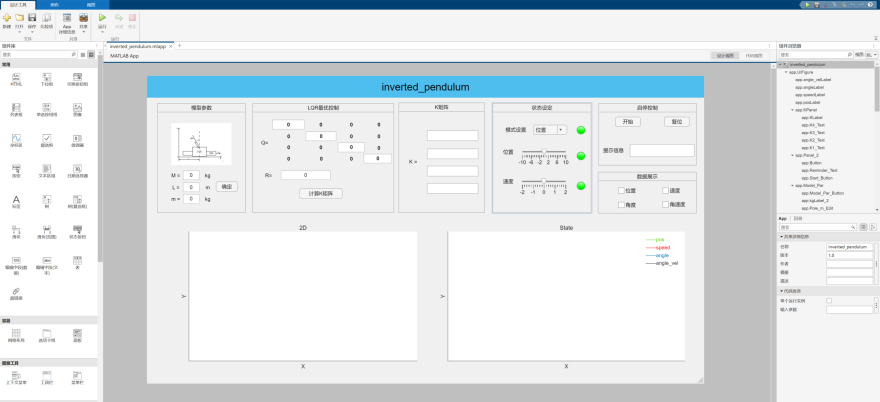


图 70 倒立摆系统上位机框图

## 倒立摆上位机设计流程

### 前端页面设计

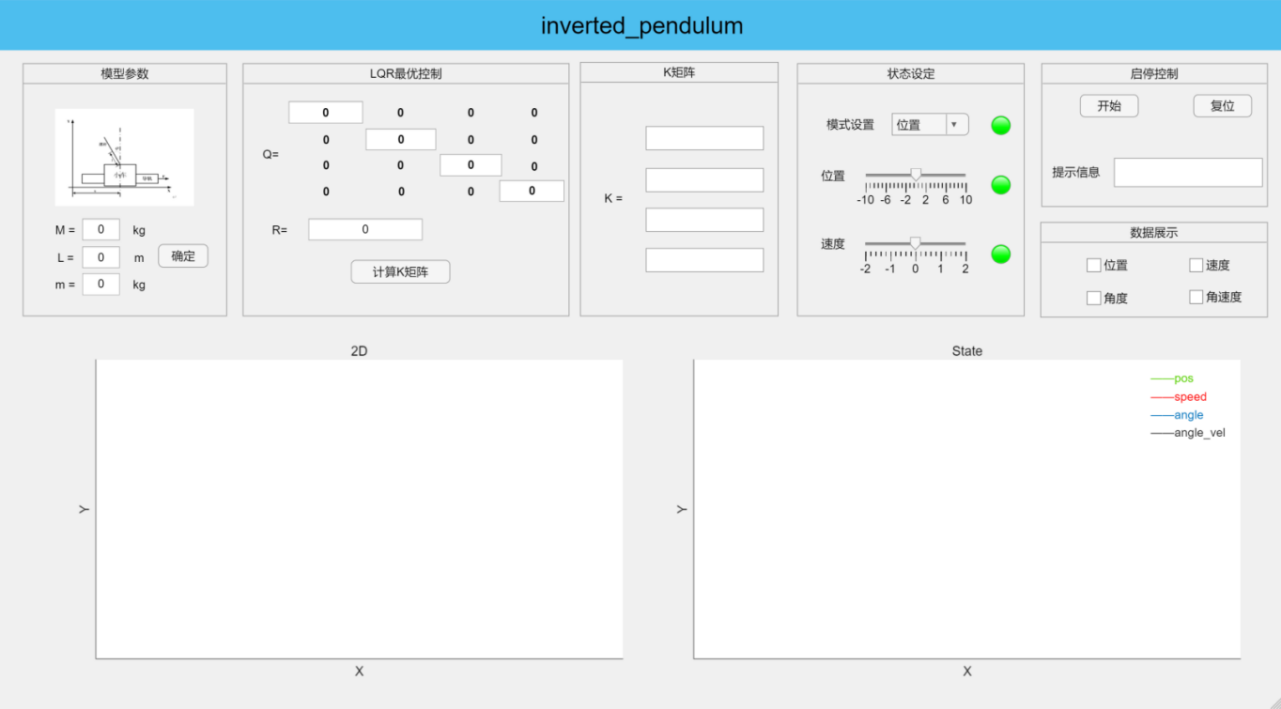


图 71 倒立摆上位机前端页面设计

前端页面包含七个部分，分别是：

①“模型参数”：设定倒立摆系统小车重量、摆杆重量以及摆杆长度

②“LQR最优控制”：输入期望的Q、R矩阵，在后台通过lqr函数进行K矩阵计算，同时将结果输出到“K矩阵”控件

③“K矩阵”：显示计算所得的K矩阵

④“状态设定”：有两种模式，位置模式及速度模式，位置模式即可通过滑动下面滑动条设置指定位置，速度模式即可通过滑动下面滑动条设置指定速度

⑤“启停设置”：设置系统开始与结束，同时打印系统信息，提示系统目前状态

⑥“数据展示”：选择是否展示相应数据

⑦“坐标轴区”：将显示倒立摆动画及系统状态响应

### 后端逻辑设置

#### 6.1.2.1 APP全局参数：

①模型参数

%模型参数

Cart\_m = 1.906;

Pole\_m = 0.109;

Pole\_len = 0.25;

②系统状态空间相关参数

%状态空间方程相关矩阵

%连续系统

A = [0 1 0 0;

0 -0.0883 0.6293 0;

0 0 0 1;

0 -0.2357 27.8285 0];

B = [0; 0.8832; 0; 2.3566];

C = [1 0 0 0;

0 0 1 0];

D = [0; 0];

G = [

1.0000 0.0499 0.0008 0.0000

0 0.9956 0.0318 0.0008

0 -0.0003 1.0350 0.0506

0 -0.0119 1.4074 1.0350 ];

H =[0.0011;

0.0441;

0.0030;

0.1189]

Cd = [1 0 0 0;

0 0 1 0];

Dd = [0; 0];

X\_tar = [2; 0; 0; 0];

X0 = [0; 0; 0; 0];

X\_last\_state = [0; 0; 0; 0];

X\_pre\_state = [0; 0; 0; 0]

③LQR的Q、R矩阵

%LQR

Q = [10 0 0 0;

0 1 0 0;

0 0 200 0;

0 0 0 1];

R = 0.0001;

④控件相关参数

%图像范围

x\_2D\_ran = [-10.0, 10.0];

y\_2D\_ran = [-1.0, 9.0];

x\_Data\_ran = [0, 20.0];

y\_Data\_ran = [-10, 10];

draw\_data\_state = [0 0 0 0 0];

%绘画对象

plot\_obj\_pos

plot\_obj\_speed

plot\_obj\_angle

plot\_obj\_angle\_vel

%绘图状态

plot\_state = [0 0 0 0];

%系统状态

system\_state = 0;

%模式状态

mode\_state = 1;

%滑动条状态

slider\_state = 0;

#### 6.1.2.2 坐标区图像显示函数编写

①绘制小车函数

function draw\_pend(app,y)

%获取位置、角度数据

L = 1.5;

x = y(1);

th = y(3);

% 车子宽度

W = 1 \* sqrt(1.906);

% 车子高度

h = .5 \* sqrt(1.906);

% 轮胎半径

wr = .5;

% 小球半径

mr = .8 \* sqrt(0.109);

% 位置高度

y = wr/2 + h/2;

%两个轮子起始点坐标

w1x = x - 0.9 \* W/2;

w1y = 0;

w2x = x + 0.9 \* W/2 - wr;

w2y = 0;

%杆终点坐标

px = x + L \* sin(th);

py = y + L \* cos(th);

plot(app.UIAxes\_2D,[-20 20], [0 0], 'k', 'LineWidth', 2)

hold(app.UIAxes\_2D,'on');

%绘制小车

rectangle(app.UIAxes\_2D,'Position', [x - W/2, y - h/2, W, h], 'Curvature', 0.1, 'FaceColor', [0.8 1 0.8], 'LineWidth', 1.5)

rectangle(app.UIAxes\_2D,'Position', [w1x, w1y, wr, wr], 'Curvature', 1, 'FaceColor', [0.5 0.5 0.7], 'LineWidth', 1.5)

rectangle(app.UIAxes\_2D,'Position', [w2x,w2y,wr,wr], 'Curvature', 1, 'FaceColor', [0.5 0.5 0.7], 'LineWidth', 1.5)

%绘制摆臂

plot(app.UIAxes\_2D,[x px], [y py], 'k', 'LineWidth', 2)

%绘制

rectangle(app.UIAxes\_2D,'Position', [px - mr/2, py - mr/2, mr, mr], 'Curvature', 1, 'FaceColor', [1 0.1 .1], 'LineWidth', 1.5)

%设置坐标及刻度

x\_tic = app.x\_2D\_ran(1) : 0.5 : app.x\_2D\_ran(2);

y\_tic = app.y\_2D\_ran(1) : 0.5 : app.y\_2D\_ran(2);

xlim(app.UIAxes\_2D, app.x\_2D\_ran);

ylim(app.UIAxes\_2D, app.y\_2D\_ran);

xticks(app.UIAxes\_2D, x\_tic);

yticks(app.UIAxes\_2D, y\_tic);

%axis equal

drawnow; hold(app.UIAxes\_2D, 'off');

end

②绘制函数

function hold\_data(app)

delete(app.plot\_obj\_pos);

delete(app.plot\_obj\_speed);

delete(app.plot\_obj\_angle);

delete(app.plot\_obj\_angle\_vel);

if(app.plot\_state(1) == 1)

app.plot\_obj\_pos = draw\_data(app, app.Xcl(:,1), 'g', 'pos');

hold(app.UIAxes\_data, "on");

end

if(app.plot\_state(2) == 1)

app.plot\_obj\_speed = draw\_data(app, app.Xcl(:,2), 'r', 'speed');

hold(app.UIAxes\_data, "on");

end

if(app.plot\_state(3) == 1)

app.plot\_obj\_angle = draw\_data(app, app.Xcl(:,3), 'b', 'angle');

hold(app.UIAxes\_data, "on");

end

if(app.plot\_state(4) == 1)

app.plot\_obj\_angle\_vel = draw\_data(app, app.Xcl(:,4), 'k', 'angle\_vel');

hold(app.UIAxes\_data, "on");

end

end

#### 6.1.2.3 其余控件逻辑

由于上位机使用了较多控件，因而仅列举LQR最优控制控件逻辑编写，其余控件不在此进行过多赘述。



图 72 LQR最优控制模块

LQR最优控制模块包含四个“编辑字段”控件，和一个按键控件。通过如下方式可以设置其回调函数，其逻辑代码如图所示，通过“控件名.Value”的方式，获取该控件的数值，并将其传递给前面参数定义的Q矩阵第一个元素，后面一次类推。按键也是如下作用，一旦按键按下便触发回调函数，在其中进行K矩阵计算。



图 73 回调函数设置

% 编辑字段回调函数

function Q1\_EditValueChanged(app, event)

value = app.Q1\_Edit.Value;

app.Q(1,1) = value;

end

% 按键回调函数

function Cal\_K\_ButtonPushed(app, event)

if app.system\_state == 0

msgbox("模型参数未设置");

else

%计算K矩阵

app.system\_state = 2;

if any(diag(app.Q)~=0) && app.R~=0 %判断是否为正定矩阵

[app.K,~] = dlqr(app.G, app.H, app.Q, app.R);

app.K1\_Text.Value = string(app.K(1));

app.K2\_Text.Value = string(app.K(2));

app.K3\_Text.Value = string(app.K(3));

app.K4\_Text.Value = string(app.K(4));

else

msgbox("Q或者R为空");

app.K1\_Text.Value = '0';

app.K2\_Text.Value = '0';

app.K3\_Text.Value = '0';

app.K4\_Text.Value = '0';

end

end

end

## 上位机运行效果

在模型参数设定完参数之后，设定Q、R矩阵，同时按下按键计算K矩阵，之后便可以在启停控制开启系统仿真，注意，上位机使用状态机进行状态控制，若操作以上步骤过程有些疏漏，导致部分参数未能正确设置，请按照弹出窗口进行重新设置。点击运行之后，可以根据需要在数据展示部分查看数据（图中四个状态变量一同查看），在状态设定部分，设定位置模式，下面滑动条后的绿灯表示可以使用位置滑动条进行位置控制，如下图所示，将目标位置设置到6处，在2D图像位置将实时展示小车从原位置到目标位置（图中为位置6）的动画演示（具体视频请移至压缩包查看）。

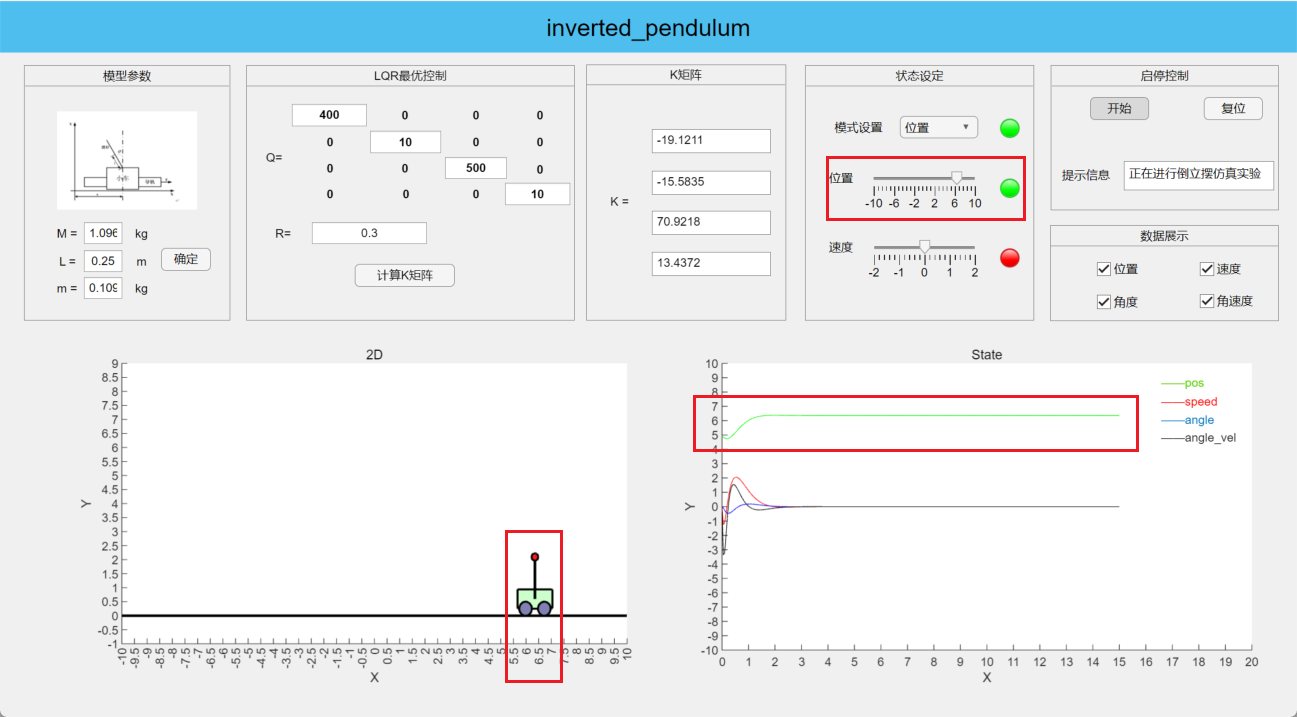


图 74 位置模式运行结果

可以看到，小车移动到指定位置，同时由数据展示可以看到小车位置由初始的5位置变换到6位置，接下来使用速度模式对倒立摆进行控制

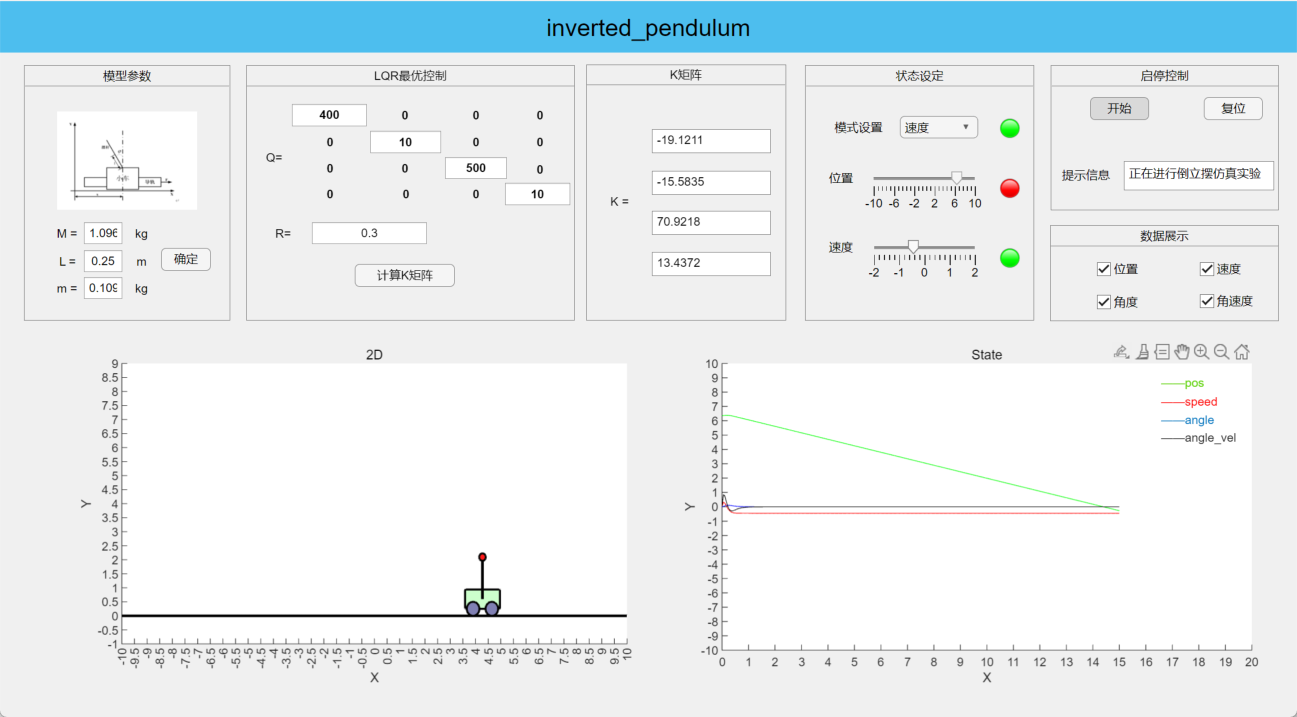


图 75 速度模式运行结果

由上述图像可以看到，红色的Speed线为负值，即小车的恒定向负方向移动，POS位置线持续下降，实现控制小车速度的功能。

# 基于平衡小车进行倒立摆PID实验

前面基于MATLAB进行了倒立摆的PID、LQR仿真，现将其运用在平衡小车上，平衡小车可以简化成以下模型，与一阶倒立摆相似。因而使用平衡小车可以进行倒立摆PID实验。

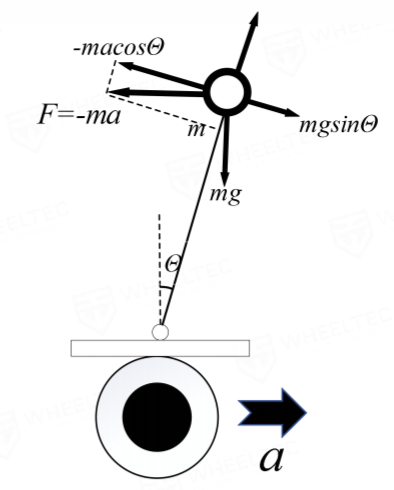


图 76 平衡小车等效模型

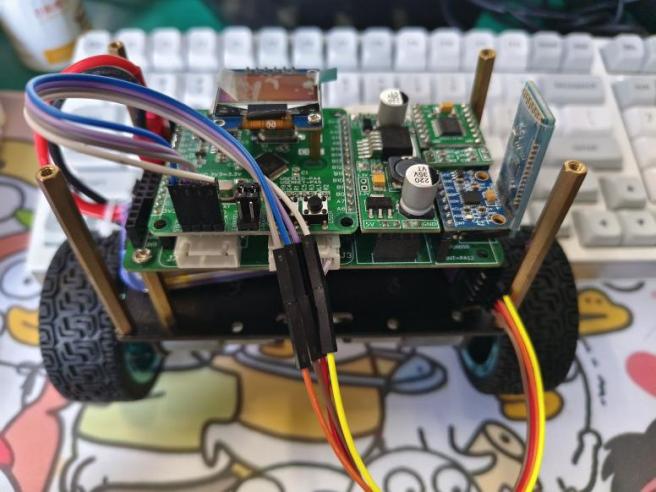


图 77 平衡小车

## 系统结构分析

### 小车外设关系图

平衡小车为主控STM32C8T6芯片，带有MPU6050陀螺仪模块、LED显示屏模块、带编码器的直流减速电机模块、蓝牙模块，外设与控制器之间的连接如图所示

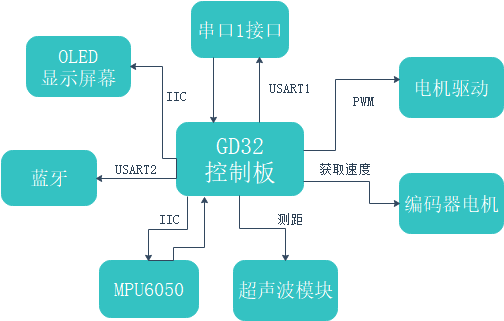


图 78 小车外设关系图

### 电机控制流程图

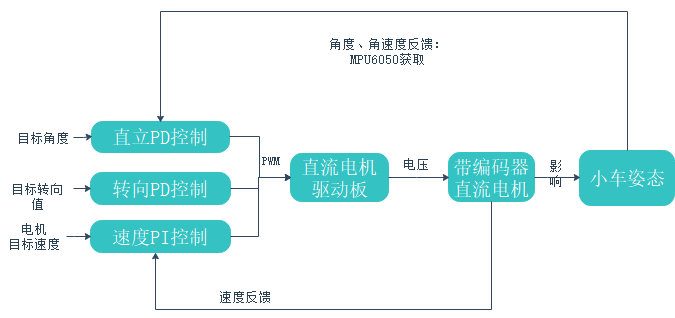


图 79 电机控制流程图

小车的直立与运动通过控制电机来实现，电机通过给定小车当前的角度与速度来进行反馈闭环控制。转向的实现是通过两个电机的差速来实现的，故左右轮转向部分叠加到电机的 PWM 的极性应该相反。

### 程序结构框图

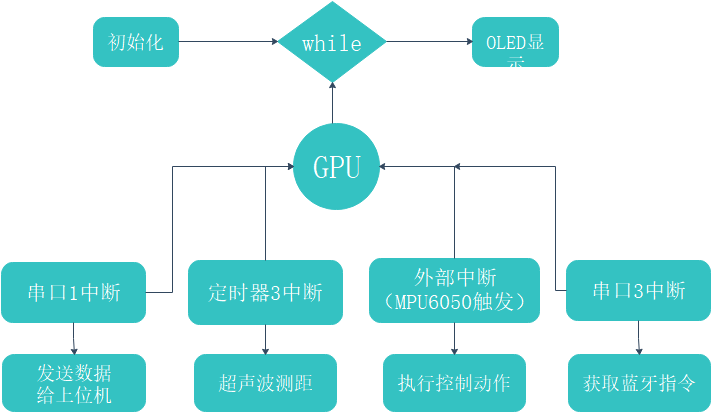


图 80 程序结构框图

程序经过一系列的初始化，进入到 while 死循环里， 其中 OLED 显示函数放在里面。小车的全部控制动作都在由MPU6050触发的中断里面。串口3中断用于接收蓝牙指令，串口 1 中断为发送数据。

## PID平衡小车程序设计

平衡小车核心在于PID控制器的设计，因而这里只展示KEIL工程中PID控制器部分代码

### 直立环PD控制

/\*\*注释

功能：直立环PD控制器

参数：Angle：测得的角度

Gyro：测得的角速度

返回值：Balance：PWM

\*\*/

int Balance(float Angle,short Gyro)

{

float Balance\_Kp=255,Balance\_Kd=1,Balance\_Ki=1.08;

float Angle\_bias,Gyro\_bias,acculate;

int Balance;

Angle\_bias=Middle\_abgle-Angle;

Gyro\_bias=0-Gyro;

Balance=Balance\_Kp\*Angle\_bias+Gyro\_bias\*Balance\_Kd;

return Balance;

}

### 速度环PI控制

/\*\*注释

功能：速度PI控制器

参数：encoder\_left左侧编码器数值，右侧编码器数值

返回值：velocity：PWM值

备注：

\*\*/

int Velocity(int encoder\_left,int encoder\_right,float Angle\_Balance)

{

float Velocity\_Kp=160,Velocity\_Ki=0.8;//速度环PI参数

static float velocity,Encoder\_Least,Encoder\_bias;

static float Encoder\_Integral;

//=============速度PI控制器=======================//

Encoder\_Least=target\_Speed-(encoder\_left+encoder\_right);

Encoder\_bias \*= 0.8; //

Encoder\_bias += Encoder\_Least\*0.2; //一阶低通滤波器

Encoder\_Integral +=Encoder\_bias; //积分出位移 积分时间：10ms

if(Encoder\_Integral>10000) Encoder\_Integral=10000; //积分限幅

if(Encoder\_Integral<-10000) Encoder\_Integral=-10000; //积分限幅

velocity=-Encoder\_bias\*Velocity\_Kp-Encoder\_Integral\*Velocity\_Ki; //速度

if(Turn\_Off(Angle\_Balance)==1||Flag\_Stop==1)

Encoder\_Integral=0; //电机关闭后清除积分

return velocity;

}

### 转向环P控制

int turn(float turn\_Yaw)

{

static float Turn\_Target；

static float turn；

static float Turn\_Amplitude=54;

float turn\_Kp=50,turn\_Kd=0.8;

if(Flag\_Left==1)

Turn\_Target=-Turn\_Amplitude/2;

else if (Flag\_Right==1)

Turn\_Target=Turn\_Amplitude/2;

else

Turn\_Target=0;

//turn=Turn\_Target\*turn\_Kp+turn\_Yaw\*turn\_Kd;

turn=Turn\_Target\*turn\_Kp;

return turn;

}

## 平衡小车控制效果

平衡小车最终能够通过PID实现平衡，稳定时间大致在3S，同时给小车施加干扰的力，小车能够快速平衡。具体演示视频可观看压缩包中的视频。

# 基于MPC模型预测控制的倒立摆仿真

## MPC理论推导

### 最优控制

研究动机（Motivation）——在约束条件（物理限制）下达到最优的系统表现（Get the best performace within certain limitation ）

**①SISO单输入单输出系统**

**轨迹跟踪**

 越小，追踪越好， 越小，输入越小

目标代价函数（Cost Function）：,求代价函数最小

：看重误差

：看重输入

**②MIMO多数入多输出系统**





X状态变量、y输出

目标/代价函数：

例子：













、调节矩阵，、、、权重系数与LQR思想相同

### MPC基本概念

通过模型来预测系统在某一时间段内的表现来进行优化控制，多用于数位控制，常用于离散型状态空间表达，其分为3步骤，在K时刻：

Step1：估计/测量读取当前系统状态

Step2：基于,…来进行最优化控制，此时代价函数：



其中考虑了约束（Condition），为预测最后时刻的误差代价函数，

Step3：只取进行控制，移步时刻，为滚动优化控制（Receding Horizon Control），对控制器性能要求极高，每次运算都是一次优化问题

### MPC公式推导

**①二次规划（Quadratic Programming）**

一般形式，前面为二次型，后面为线性，没有前面部分则变成线性规划，若Q是对角矩阵，则变成最小二乘问题，因而转变目标：将模型化为该形式进行最优化求解

**②MPC公式推导**

设一个系统： 

在时刻：

预测输出（后面的表示在时刻预测的）：



为预测区间（Predictive Horizon）



假设系统输出，参考，得到误差，则代价函数：



其中：误差加权和输入加权和终端误差，想使用二次规划需要先将替换成，







………













是一个1\*1矩阵，则，又

所以



设，，则：



得到该函数后便可使用二次规划进行最优化控制，matlab可使用quadprog函数进行求解

## 基于MPC进行倒立摆仿真实验

### MPC模型预测函数编写

function [E , H]=MPC\_Matrices(A,B,Q,R,F,N)

n=size(A,1);% A 是 n x n 矩阵, 得到 n

p=size(B,2);% B 是 n x p 矩阵, 得到 p

%%%%%%%%%%%%

M=[eye(n);zeros(N\*n,n)]; % 初始化 M 矩阵. M 矩阵是 (N+1)n x n的，它上面是 n x n 个 "I", 这一步先把下半部 % 分写成 0

C=zeros((N+1)\*n,N\*p); % 初始化 C 矩阵, 这一步令它有 (N+1)n x NP 个 0

% 定义M 和 C

tmp=eye(n);%定义一个n x n 的 I 矩阵

%　更新Ｍ和C

for i=1:N % 循环，i 从 1到 N

rows =i\*n+(1:n); %定义当前行数，从i x n开始，共n行

C(rows,:)=[tmp\*B,C(rows-n, 1:end-p)]; %将c矩阵填满

tmp= A\*tmp; %每一次将tmp左乘一次A

M(rows,:)=tmp; %将M矩阵写满

End

% 定义Q\_bar和R\_bar

Q\_bar = kron(eye(N),Q);

Q\_bar = blkdiag(Q\_bar,F);

R\_bar = kron(eye(N),R);

% 计算G, E, H

G=M'\*Q\_bar\*M; % G: n x n

E=C'\*Q\_bar\*M; % E: NP x n

H=C'\*Q\_bar\*C+R\_bar; % NP x NP

end

MPC\_Matrices函数传入参数为系统状态空间方程A、B矩阵以及Q、R、F调节矩阵，以及N预测空间、输出为中的E、H矩阵，由于项只与k时刻有关，优化过程中为定值，因而无需进行考虑。

function u\_k= Prediction(x\_k,E,H,N,p)

U\_k = zeros(N\*p,1); % NP x 1

U\_k = quadprog(H,E\*x\_k);

u\_k = U\_k(1:p,1) % 取第一个结果

end

Prediction函数则根据传入的当前值，以及MPC\_Matrices函数求得的E、H矩阵，N预测区间，输入u维度P，使用quadprog进行二次规划求解，获得在未来N个时刻的，之后只取第一个返回，进行控制，该函数还可用于simulink中matlab function函数块，在simulink进行MPC倒立摆仿真实验。

### Matlab 进行MPC倒立摆实验

%% 定义Q矩阵，n x n 矩阵

Q = diag([200 10 200 10]);

%% 定义F矩阵，n x n 矩阵

F = Q;

%% 定义R矩阵，p x p 矩阵

R=0.1;

%% 定义step数量k

k\_steps=5000;

%% 定义矩阵 X\_K， n x k 矩 阵

X\_K = zeros(n,k\_steps);

%% 初始状态变量值， n x 1 向量

X\_K(:,1) =[0.1;0;0;0];

%% 定义输入矩阵 U\_K， p x k 矩阵

U\_K=zeros(p,k\_steps);

%% 定义预测区间K

N=20;

%% Call MPC\_Matrices 函数 求得 E,H矩阵

[E, H]=MPC\_Matrices(A\_disc,B\_disc,Q,R,F,N);

%% 计算每一步的状态变量的值

for k = 1 : k\_steps

%% 求得U\_K(:,k)

U\_K(:,k) = Prediction(X\_K(:,k),E,H,N,p);

%% 计算第k+1步时状态变量的值

X\_K(:,k+1)=(A\_disc\*X\_K(:,k)+B\_disc\*U\_K(:,k));

end

%% LQR控制

%设置LQR矩阵

%Q = diag([400 10 500 10]);

Q = diag([100 10 20 10]);

R = 0.8;

%使用LQR计算K

[K, P] = dlqr(A\_disc,B\_disc,Q,R);

%% 绘制状态变量和输入的变化

subplot(2, 1, 1);

hold;

for i =1 :size (X\_K,1)

plot (X\_K(i,:));

end

legend("x1","x2","x3","x4")

hold off;

subplot (2, 1, 2);

hold;

for i =1 : size (U\_K,1)

plot (U\_K(i,:));

end

legend("u1")



图 81 系统响应



图 82 系统输入

可以看到，系统在经过100次（0.03\*100=3S）MPC控制后，趋于稳定，收敛迅速，且基本无稳态误差，控制性能优秀，与此同时系统输入最大接近1N，也有较低的损耗

### Simulink 进行MPC倒立摆实验

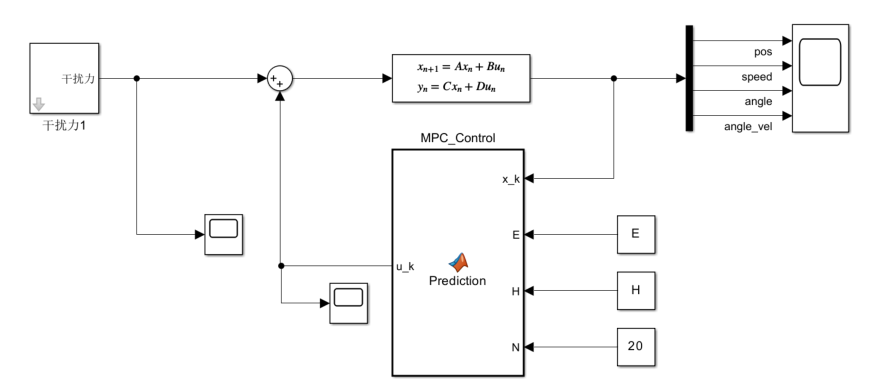


图 83 Simulink仿真框图

Simulink仿真框图如上，使用simulink中的Matlab Function函数编写MPC控制器，通过sum进行相加反馈，以此实现MPC控制，其中MPC控制器代码如下，相比于Matlab中的Prediction函数，需要多指定对应的active-set算法，而active-set算法需要初始点x0，因而还需多定义x0，除了H，E \* x\_k外，其余参数设置为空（不可直接忽略，否则Simulink会报错）

function u\_k = Prediction(x\_k, E, H, N)

% 初始化输出

U\_k = zeros(N\*1, 1); % NP x 1

u\_k = zeros(1, 1); % 指定 u\_k 为固定大小 p x 1

% 获取 H 矩阵的行列数

[m, n] = size(H);

% 根据 H 矩阵的列数定义初始向量 x0

x0 = zeros(n, 1);

% 使用quadprog函数求解优化问题

opts = optimoptions('quadprog','Algorithm','active-set');

[U\_k, ~, ~] = quadprog(H, E \* x\_k, [], [], [], [], [], [], x0, opts);

% 取第一个结果

u\_k = U\_k(1, 1);

end

将干扰设置为脉冲干扰，为持续1S、20N的强脉冲干扰，测试系统性能，由运行结果可以看到即使在强干扰下，系统任然能够在4S左右达到稳定，控制效果优秀

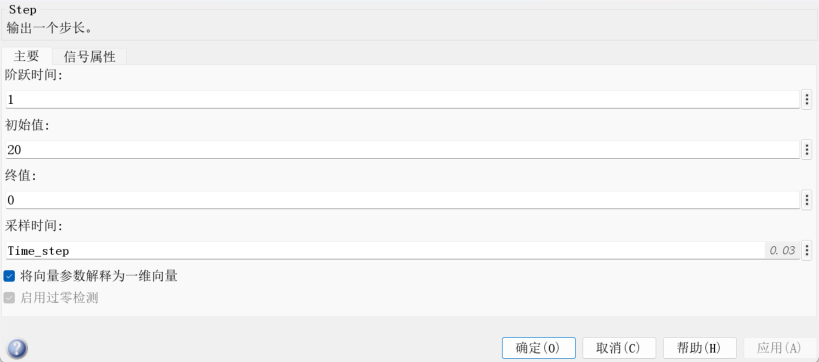


图 84 干扰力设置

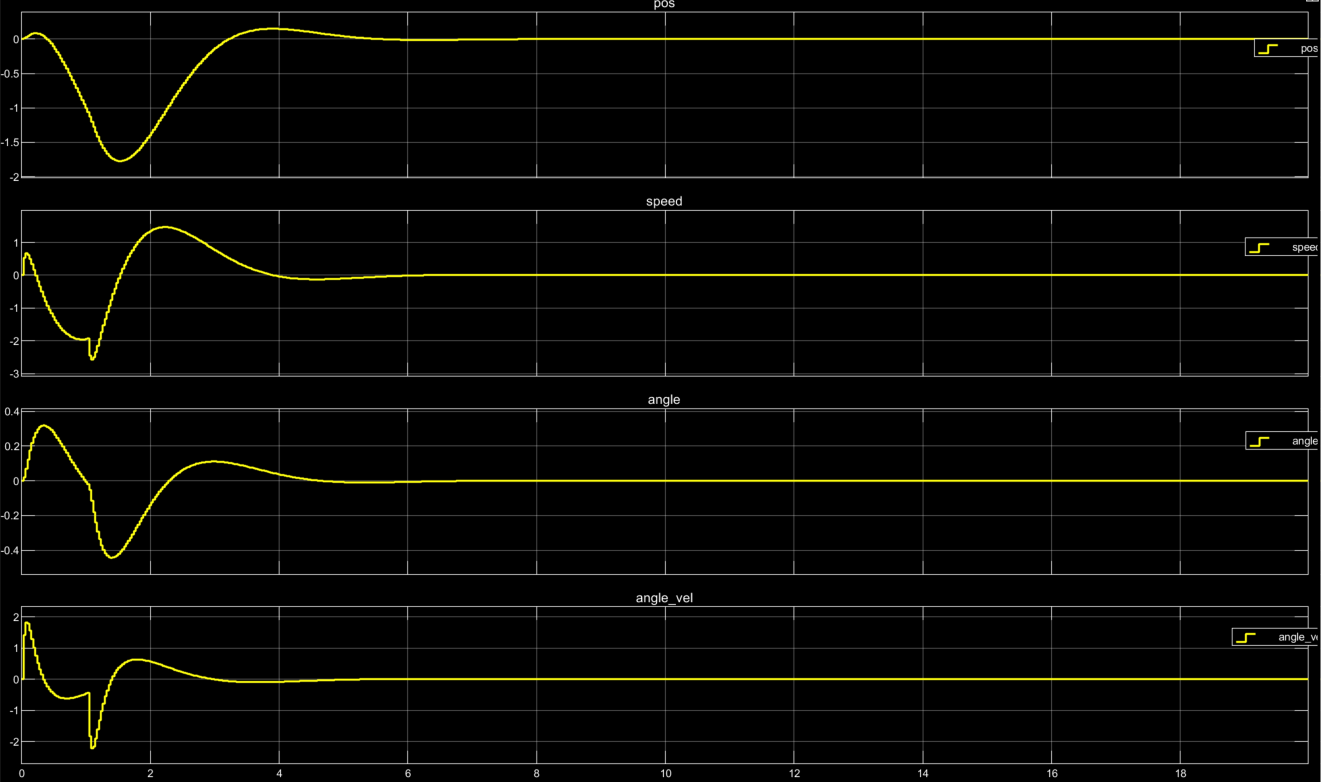


图 85 系统响应

将干扰设置为正弦干扰，测试系统性能，由运行结果可以看到位置及速度在幅度为0.01周边震荡，角度角速度则在0.00005附近震荡，误差在可接受范围内。



图 86 正弦干扰信号

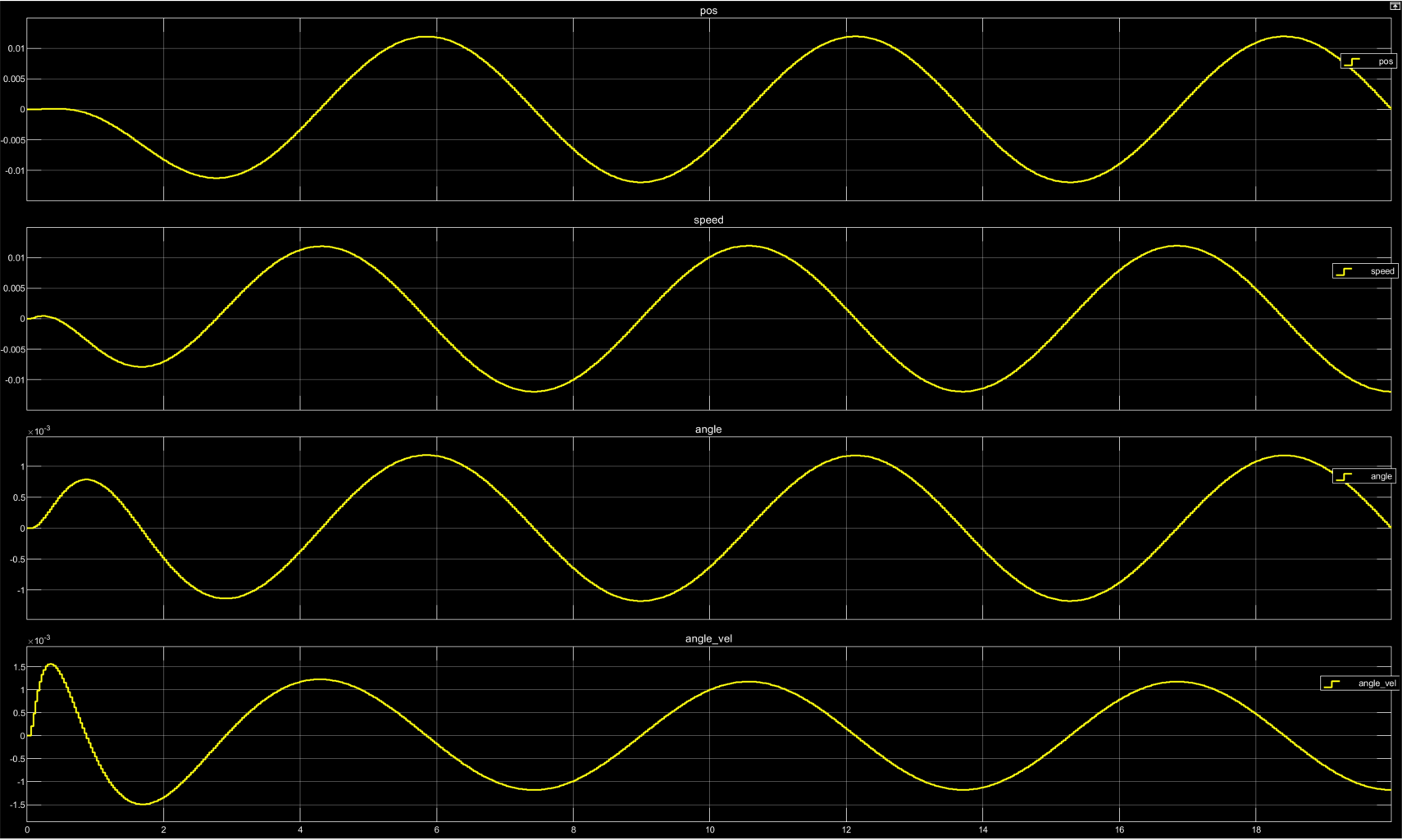


图 87 系统响应

与LQR控制进行对比

在相同脉冲干扰下，进行LQR实验，其简易仿真框图如下

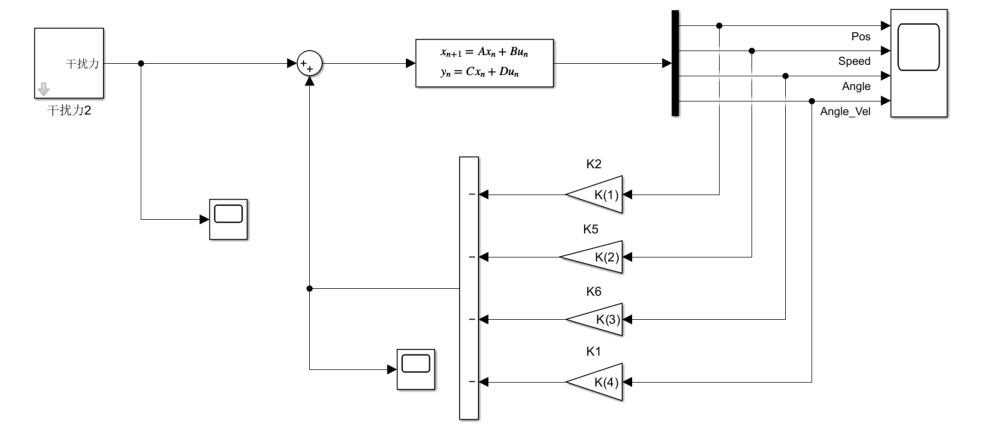
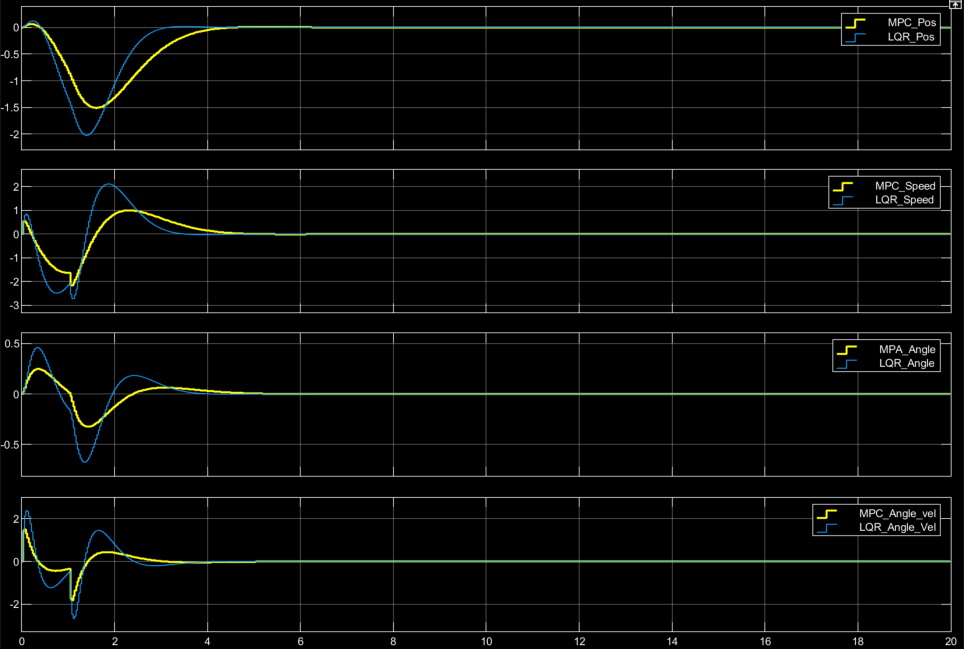


图 88 LQR仿真框图

将干扰设置为1S，20N的脉冲干扰，运行结果如下



对比MPC控制及LQR控制，可以看到两者稳定时间基本一致，系统均在4S左右收敛，但是相比于LQR，MPC在整个过程中，系统响应幅度比LQR小，也就是说，MPC更不容易收到外界干扰，这是由于MPC在进行控制决策时是基于对未来系统行为的预测，以选择当前时刻的控制输入。在每个控制周期内，MPC通过求解一个优化问题来选择最优的控制输入序列，以使系统在未来的一段时间内达到最佳性能。MPC在每个控制周期内都可以重新计算控制输入，以适应系统可能出现的变化 即对系统扰动的处理，MPC能够通过预测模型对系统可能的扰动进行预测，并在控制决策中考虑这些扰动。这使得MPC在系统扰动较大的情况下能够更加稳健地进行控制。能够更加灵活地处理各种系统特性，尤其是对于时变、非线性和不确定系统，MPC能够通过对未来系统行为的预测来实现更优秀的控制性能。这也是MPC在许多实际工程应用中被广泛采用的原因之一。

# 倒立摆实验总结与心得体会

## 实验总结：

在本次实验中，我首先进行了动力学分析和连续时间状态空间模型的建立，侧重于倒立摆模型的问题描述和使用MATLAB建立动力学方程。接着，我转向离散时间状态空间模型的建立，包括倒立摆问题的背景介绍、理论推导以及使用MATLAB建立离散时间模型下的状态空间方程。我还分析了倒立摆系统的稳定性、能控性和能观性，并在不加控制器的情况下求解了离散倒立摆系统在特定信号输入下的响应。通过MATLAB和simulink，我得出了离散倒立摆系统的阶跃响应和脉冲响应。

接下来，我对离散时间倒立摆系统进行了PID控制，包括理论推导、MATLAB建立连续倒立摆传递函数模型和设计PID控制器。在这一部分，我使用simulink和MATLAB编程分析了加入PID控制器前后的力到角度传递函数特性，并总结了实验心得。

在第四部分中，我基于状态观测器的倒立摆系统进行了最优控制，包括LQR控制算法和卡尔曼滤波的理论推导。通过simulink构建基于卡尔曼滤波器下的倒立摆LQR控制模块，我进行了仿真框图、高斯白噪声设计、系统状态空间表达式模块设计、LQR控制器设计以及卡尔曼滤波器设计。

此外，我还进行了基于卡尔曼滤波及LQR进行simscape刚体仿真的实验，并对simscape多刚体仿真模块搭建和仿真结果进行了分析。同时，我还基于LQR编写了倒立摆系统上位机，并对倒立摆上位机设计流程和运行效果进行了详细描述。

最后，我进行了基于平衡小车进行倒立摆PID实验，包括系统结构分析、PID平衡小车程序设计和平衡小车控制效果。通过这一部分实验，我得出了关于平衡小车控制效果的结论。

总的来说，本次实验涵盖了从动力学分析到最优控制的多个方面，通过MATLAB和simulink的应用，得以深入了解倒立摆系统的建模、分析和控制方法，为今后的研究和实践提供了宝贵的参考。

## 心得体会

整个实验历时2个月，在计控大作业还没发布前，已经按照其他方案进行建模控制，发布作业后，便跟上了大作业的步伐，按照大作业要求进行设计，期间遇到了很多困难，最后通过查找资料、询问老师，和同学交流的方式解决了很多问题，最终完成了这次实验。

通过本次实验，我对控制理论的应用和实际系统的控制有了更深入的认识。首先，我意识到了建立系统模型的重要性。在实验中，需要对倒立摆系统进行建模，包括动力学方程、状态空间表示等，这些都是理解系统特性和设计控制器的基础。其次，我学会了使用MATLAB和simulink进行系统分析和控制器设计。通过仿真实验，我能够直观地看到不同控制算法对系统响应的影响，从而更好地理解控制理论的具体应用。

在实验中，我还学习了离散时间状态空间模型、PID控制和LQR控制等基本概念和方法。特别是在应用卡尔曼滤波器进行状态观测和LQR控制算法进行系统控制方面，让我对最优控制有了更深入的了解。同时，通过基于平衡小车进行倒立摆PID实验的学习，我了解了在实际系统中控制算法的设计和调试过程，对系统的稳定性和性能进行评估和优化。这些经验对我的未来学习和工作都将产生积极的影响，使我能够更好地应用控制理论解决实际问题。通过这次实验，我也意识到了控制理论对现代科学技术发展的重要性，我会继续深入学习，不断提升自己的控制理论水平，为未来的科研和工程实践做好充分准备。

# 参考文献

[MATLAB\_APPDesigner基础教程](https://www.bilibili.com/video/BV16f4y147x9/?p=3&spm_id_from=pageDriver&vd_source=b7810ef92710cfbae9c5ced42144fa24)

[LQR原理详解](https://blog.csdn.net/qq_36133747/article/details/123413115)

[卡尔曼原理推导](https://blog.csdn.net/weixin_53313040/article/details/128876114?spm=1001.2014.3001.5502)

[卡尔曼滤波原理详解](https://blog.csdn.net/weixin_43942325/article/details/103416681?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522170391955216800182170179%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=170391955216800182170179&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-4-103416681-null-null.142%5ev99%5epc_search_result_base8&utm_term=%E5%8D%A1%E5%B0%94%E6%9B%BC%E6%BB%A4%E6%B3%A2&spm=1018.2226.3001.4187)

# 成绩评定

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **大作业评分细则表（满分100分）：**   |  |  |  |  | | --- | --- | --- | --- | | 评分项目 | 评分标准 | 成绩 | 权重 | | 理论推导 | 数学推导详细准确、尽量多应用在课堂上学到的计算机控制理论知识，构建离散控制器，无推导错误 |  | 25％ | | 仿真模拟 | 有流程图、伪代码，响应仿真图详尽、能反应出理论结果有效性。（仿真图等可以参考示例；在系统响应仿真图的基础上，给出倒立摆控制前后的模拟动画演示） |  | 25％ | | 控制效果 | 根据理论推导建立的控制器，可以通过数值仿真展现出优良的控制效果。给出并分析能反映控制性能各类参数 |  | 20％ | | 撰写质量 | 公式书写符合规范，文字流畅书面正式，无明显文字错误，有必要的参考文献引用 |  | 20％ | | 分析总结 | 有倒立摆问题背景介绍，针对控制器设计和仿真结果，有合理详尽的分析与总结 |  | 10％ | | 最终成绩 | 综合以上各点给出总成绩 |  | 100% |   指导教师签字：  年 月 日 |