

一、 Matlab Function 模块练习

第一个信号为从 0 时刻开始的终值为 1 的阶跃信号；第二个信号是幅值为 1，频率为 1rad/s 的正弦信号；第三个信号是从 1 时刻开始的终值为 3 的阶跃信号。

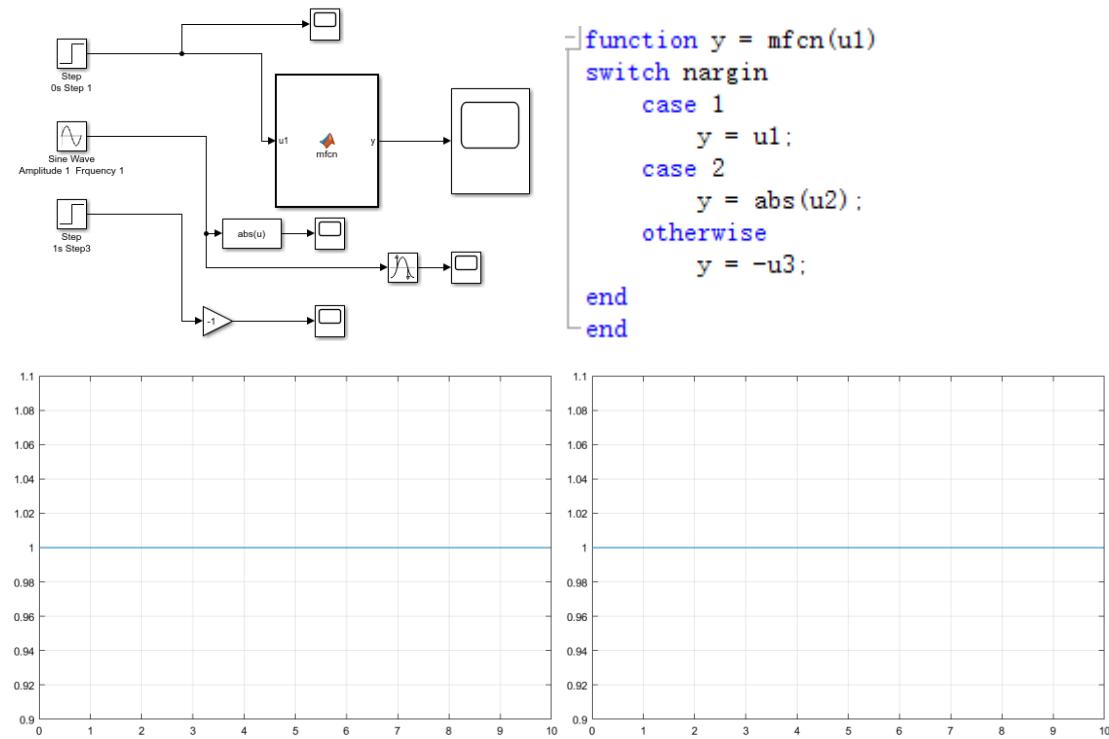
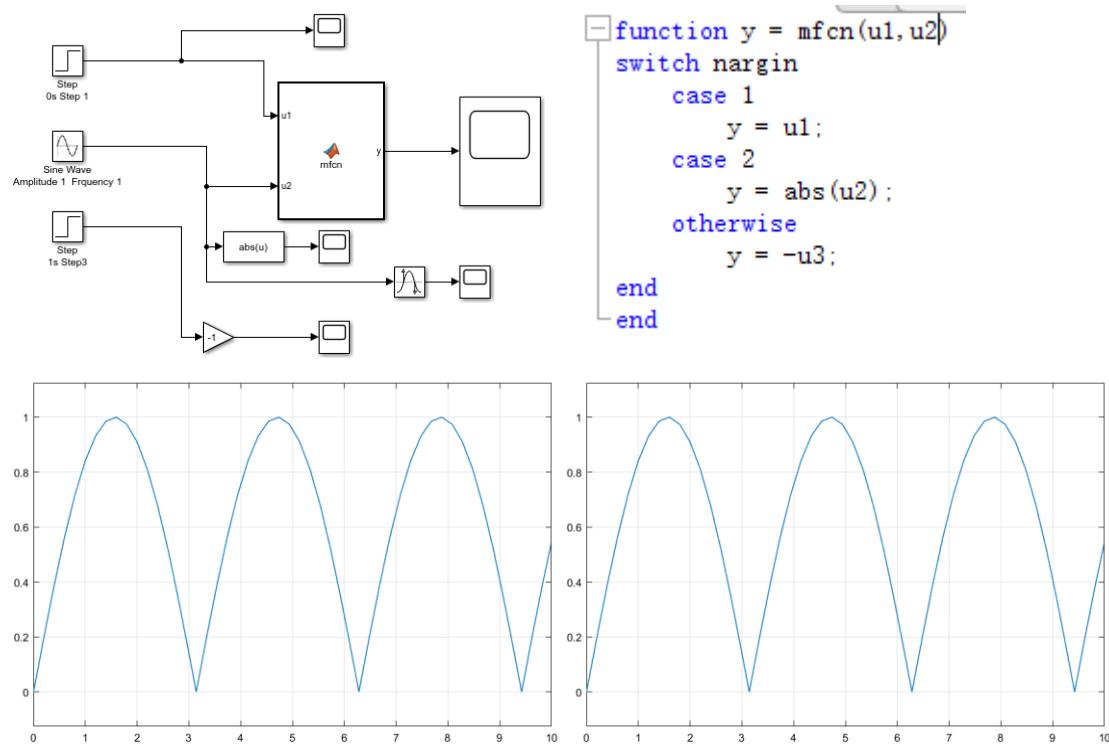


图 1 功能①模块连接与 Matlab Fcn 设置（左：Matlab Fcn 输出；右：对比输出）



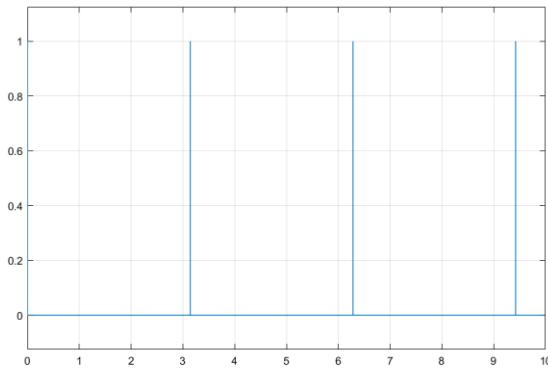


图 2 功能②模块连接与 Matlab Fcn 设置
(左: Matlab Fcn 输出; 右: 对比输出; 下: 过零检测输出)

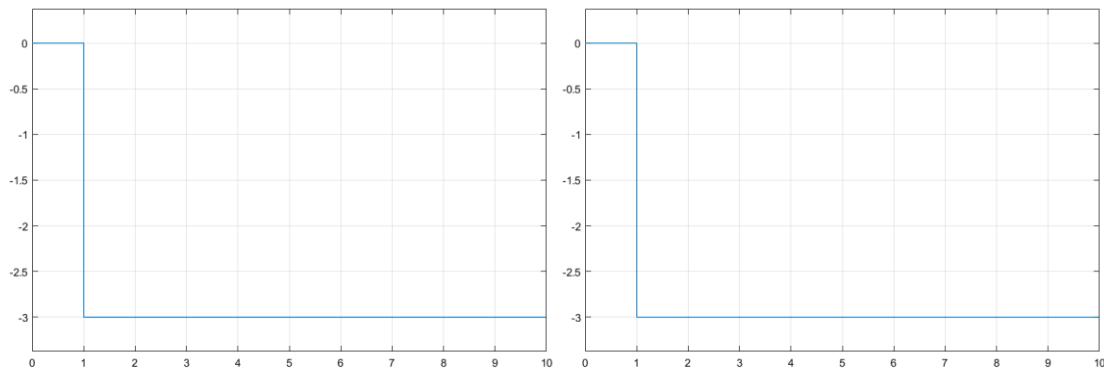
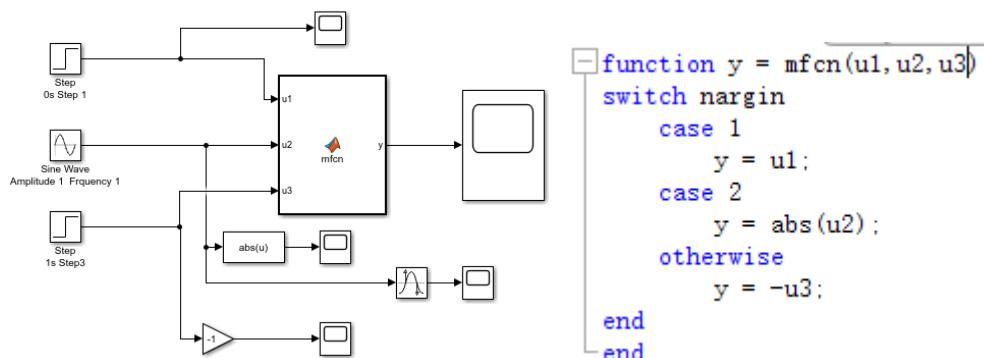


图 3 功能③模块连接与 Matlab Fcn 设置 (左: Matlab Fcn 输出; 右: 对比输出)

二、利用 sfuntmpl 模板创建 level 1 格式的 S 函数的练习

```

case 1
sys=mdlDerivatives(t,x,u,l,m,M);

```

```

function [sys, x0, str, ts, simStateCompliance]=mdlInitializeSizes
%
% call simsizes for a sizes structure, fill it in and convert it
% sizes array.
%
% Note that in this example, the values are hard coded. This is
% recommended practice as the characteristics of the block are typically
% defined by the S-function parameters.
%
sizes = simsizes;

sizes.NumContStates    = 4;
sizes.NumDiscStates    = 0;
sizes.NumOutputs       = 4;
sizes.NumInputs        = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes   = 1; % at least one sample time is needed

sys = simsizes(sizes);

%
% initialize the initial conditions
%
x0  = [0 0 0.1 0];

function sys=mdlDerivatives(t,x,u,l,m,M)
sys = [ x(2);
        1/(M*cos(x(3)))*((u+m*l*x(4)^2*sin(x(3)))*cos(x(3))-(9.8*sin(x(3))+l*(x(4))^2*sin(x(3))*cos(x(3)))*m;
        x(4);
        1/(M*l*cos(x(3))^2)*((M+m)*(9.8*sin(x(3))+l*x(4)^2*sin(x(3))*cos(x(3)))-cos(x(3))*(u+m*l*sin(x(3)))) ];
%
%=====
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=====
%
function sys=mdlUpdate(t,x,u)
sys = [];

%
%=====
% mdlOutputs
% Return the block outputs.
%=====
%
function sys=mdlOutputs(t,x,u,l)
sys = [x];
%
```

图 4 mS-Fcn 改动处

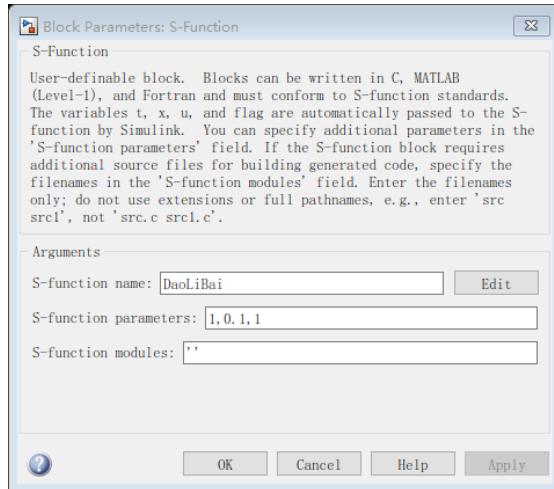


图 5 S-Function 设置

三、利用 Simulink 模块搭建上述倒立摆的动态仿真系统，用练习 2 中所设计的控制器对所搭建的倒立摆动态系统进行控制，检验控制效果

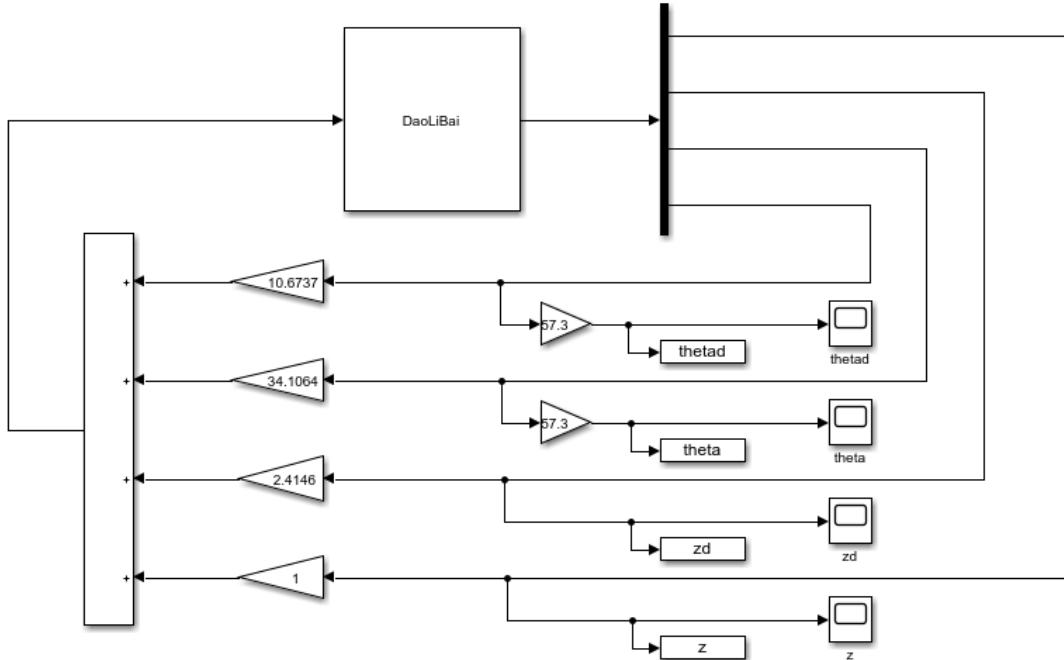


图 6 带全状态反馈控制器的倒立摆模型

倒立摆模型和控制器模型公式如下：

$$\begin{cases} \ddot{z} = \frac{(u + ml\dot{\theta}^2 \sin \theta) \cos \theta - (g \sin \theta + l\dot{\theta}^2 \sin \theta \cos \theta)m}{M \cos \theta} \\ \ddot{\theta} = \frac{(M+m)(g \sin \theta + l\dot{\theta}^2 \sin \theta \cos \theta) - \cos \theta(u + ml \sin \theta)}{Ml \cos^2 \theta} \end{cases} \quad | \quad \text{状态选取为: } \begin{cases} x_1 = z \\ x_2 = \dot{z} \\ x_3 = \theta \\ x_4 = \dot{\theta} \end{cases}, \text{ 初始状态为 } [0, 0, 0.1, 0];$$

图 7 倒立摆模型公式

在 $l=1, m=0.1, M=1$ 的情况下，保证倒立摆稳定的全状态反馈控制器为

$$u = K_1 z + K_2 \dot{z} + K_3 \theta + K_4 \dot{\theta}$$

$$u = z + 2.4146\dot{z} + 34.1064\theta + 10.6737\dot{\theta}$$

图 8 控制器模型与默认模型参数的控制器增益

此时控制效果如下（画图代码见附录或源文件）：

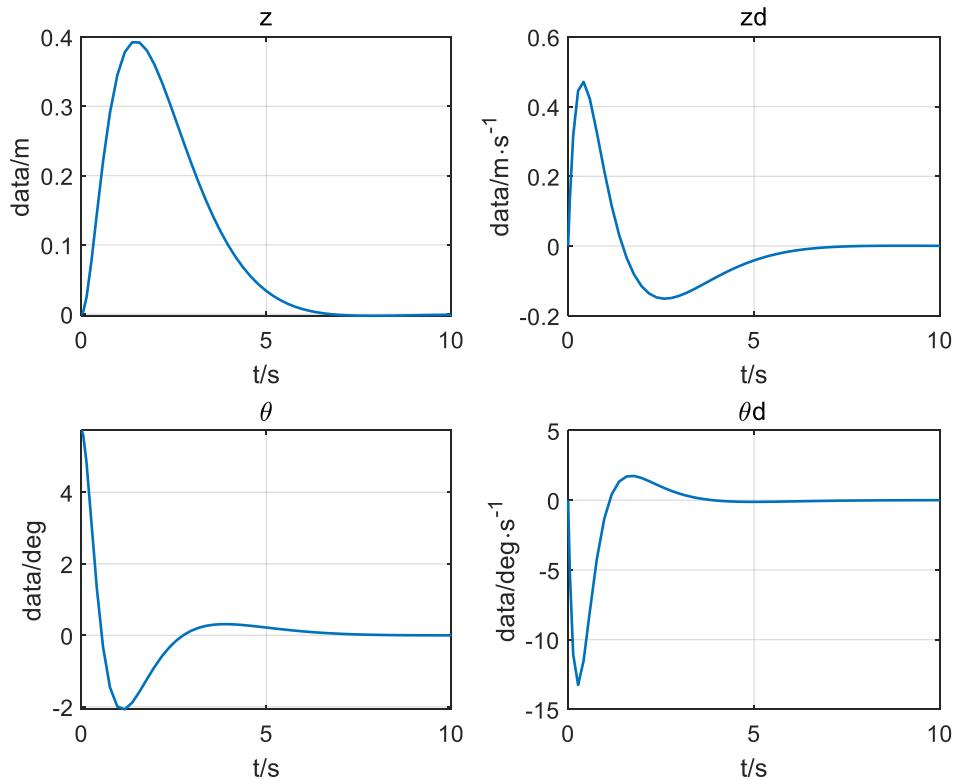


图 9 默认模型参数与控制器增益的控制效果（状态输出）

改变摆杆长度 $l = 1$ ，摆杆质量 $m = 0.2$ ，小车质量 $M = 2$ ，此时的控制效果如下：

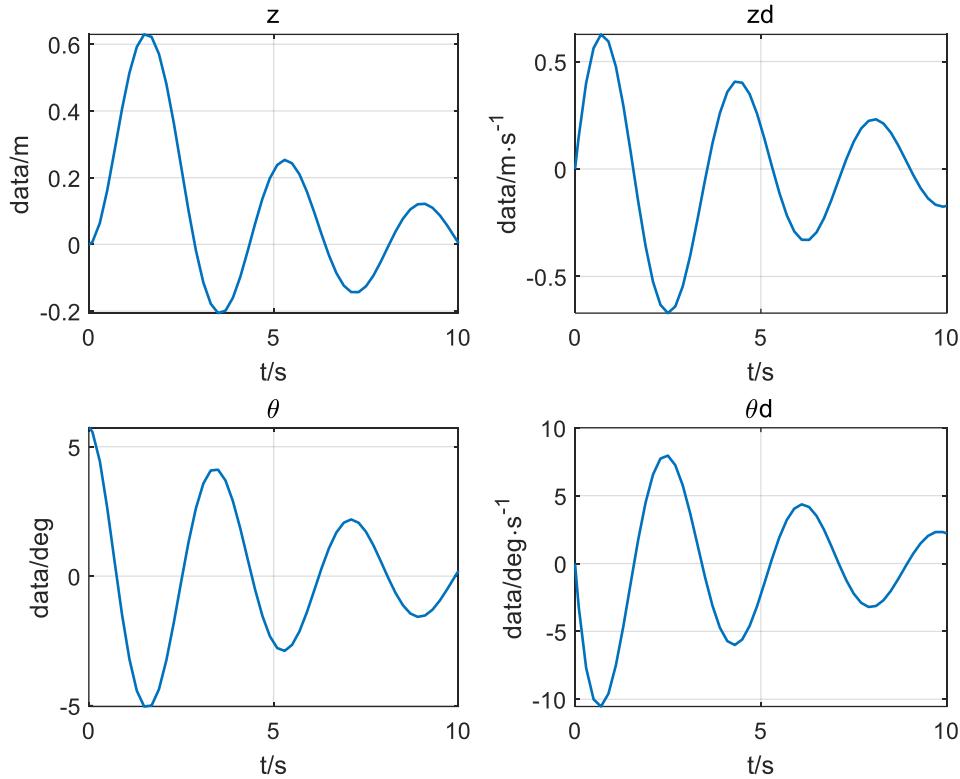


图 10 改变模型参数与默认控制器增益的控制效果

可以看出，此时在 10s 的仿真时间内，状态响应未收敛，需要重新设计控制器参数，用 Check Custom Bounds 模块进行控制器增益整定，设置 8 秒开始边界为 [-0.01 0.01]，整定得到

$$K_1 = 0.994475318464968$$

$$K_2 = 2.365734779560844$$

$$K_3 = 40.772814639268900$$

$$K_4 = 10.736019334899419$$

控制器增益整定过程与其控制效果如下：

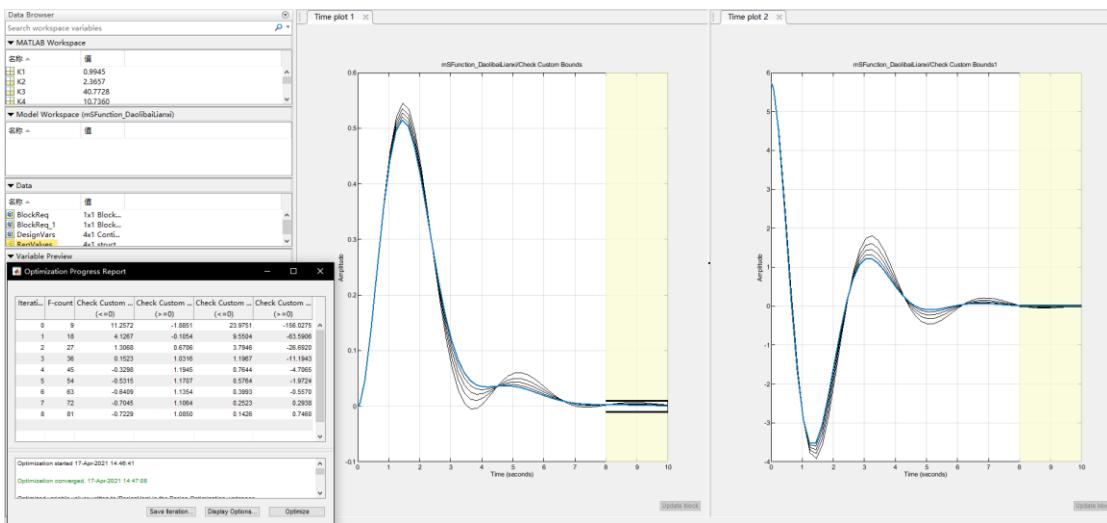


图 11 控制器增益整定过程 (Time plot 1: z; Time plot 2: theta)

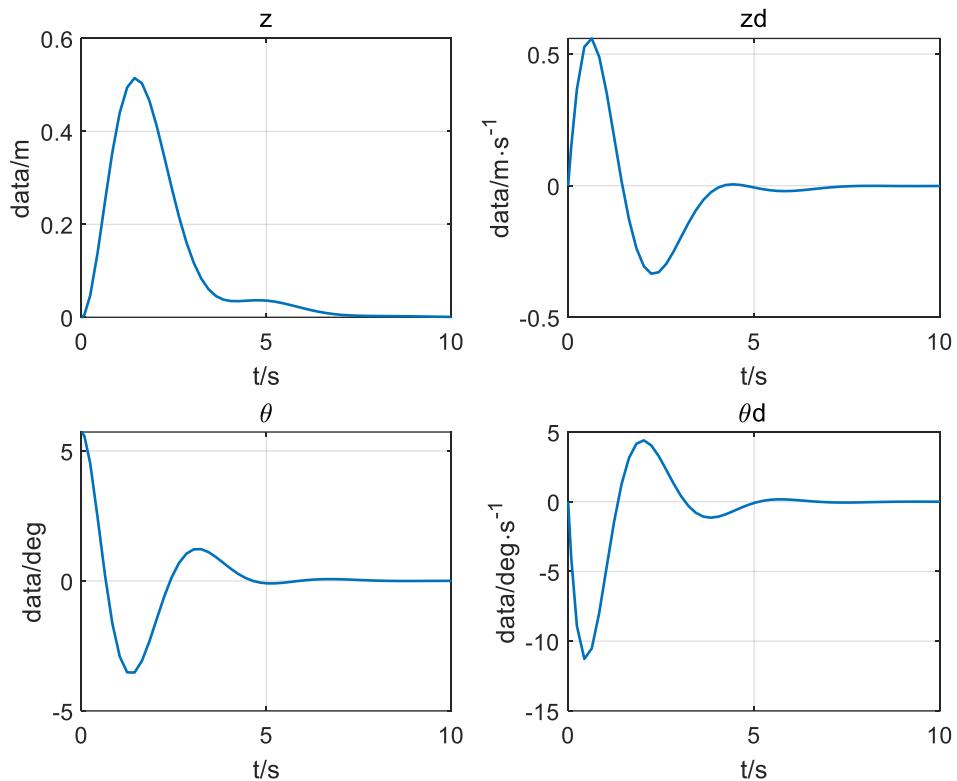


图 12 改变模型参数与整定控制器增益的控制效果
可以看出，此时在 10s 的仿真时间内，状态响应收敛，满足设计要求。

四、连续系统、离散系统、混合系统、变步长系统的 S 函数的编写

1. 连续状态 S-Function

连续系统状态方程和 A、B、C、D 矩阵参数如下：

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

$$A = \begin{bmatrix} -2 & -4 \\ 4 & -3 \end{bmatrix}, B = \begin{bmatrix} 1 & -7 \\ 0 & -2 \end{bmatrix}, C = \begin{bmatrix} 0 & 2 \\ 1 & -5 \end{bmatrix}, D = \begin{bmatrix} -3 & 0 \\ 1 & 0 \end{bmatrix}$$

mS-Fcn 需要改动的地方如下：

```

case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);

%%%%%
% Derivatives %
%%%%%
case 1,
    sys=mdlDerivatives(t,x,u,A,B,C,D);

%%%%%
% Outputs %
%%%%%
case 3,
    sys=mdlOutputs(t,x,u,A,B,C,D);

%%%%%
% Unhandled flags %
%%%%%
case { 2, 4, 9 },
    sys = [];

function [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D)
    sizes = simsizes;
    sizes.NumContStates = 2;
    sizes.NumDiscStates = 0;
    sizes.NumOutputs = 2;
    sizes.NumInputs = 2;
    sizes.DirFeedthrough = 1;
    sizes.NumSampleTimes = 1;

    sys = simsizes(sizes);
    x0 = zeros(2,1);
    str = [];
    ts = [0 0];

function sys=mdlDerivatives(t,x,u,A,B,C,D)
    sys = A*x + B*u;

    % end mdlDerivatives
    %

%=====
% mdlOutputs
% Return the block outputs.
%=====

%
function sys=mdlOutputs(t,x,u,A,B,C,D)
    sys = C*x + D*u;

% end mdlOutputs

```

图 13 my_csfnc.m 中的改动

2. 离散状态 S-Function

离散系统状态方程和 A、B、C、D 矩阵参数如下：

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Du(k) \end{cases}$$

$$A = \begin{bmatrix} -1.3839 & -0.5097 \\ 1 & 0 \end{bmatrix}, B = \begin{bmatrix} -2.5559 & -1 \\ 1 & 4.2382 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 2.0761 \\ 0 & 7.7891 \end{bmatrix}, D = \begin{bmatrix} -0.8141 & -2.9334 \\ 1.2426 & 0 \end{bmatrix}$$

mS-Fcn 需要改动的地方如下：

```

switch flag

%%%%%
% Initialization %
%%%%%
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D);

%%%%%
% Update %
%%%%%
case 2,
    sys = mdlUpdate(t,x,u,A,B,C,D);

%%%%%
% Output %
%%%%%
case 3,
    sys = mdlOutputs(t,x,u,A,C,D);

%%%%%
% Terminate %
%%%%%
case 9,
    sys = [];% do nothing
end % end mdlInitializeSizes

function sys = mdlUpdate(t,x,u,A,B,C,D)
    sys = A*x+B*u;

%end mdlUpdate

%
%=====
% mdlOutputs
% Return the output vector for the S-function
%=====
%
function sys = mdlOutputs(t,x,u,A,C,D)
    sys = C*x+D*u;

%end mdlOutputs

```

图 14 my_dsfuc.m 中的改动

3. 混合系统 S-Function

S-Function 对混合系统的处理十分直接，通过参数 flag 来控制对于系统中的连续和离散部分调用正确的 S-Function 子程序。混合系统和变步长系统 S-Funciton 的一个特点就是在所有的采样时间上，Simulink 都会调用 mdlUpdate, mdlOutputs 以及 mdlGetTimeOfNextVarHit 程序。这意味着在这些程序中，必须进行测试以确定正在处理哪个采样点以及哪些采样点只执行相应的更新。

我编写的混合系统 S-Function 模拟了连续一阶积分器和随后的离散 2 个单位延时，总模型中给出了对应的 Simulink 模块实现。

mS-Fcn 需要改动的地方如下：

```

function [sys,x0,str,ts]=mdlInitializeSizes(dperiod,doffset)
    sizes = simsizes;
    sizes.NumContStates = 1;
    sizes.NumDiscStates = 1;
    sizes.NumOutputs = 1;
    sizes.NumInputs = 1;
    sizes.DirFeedthrough = 0;
    sizes.NumSampleTimes = 2; %两种采样时间

    sys = simsizes(sizes);
    x0 = ones(2,1);
    str = [];
    ts = [0 0]; % sample time
        dperiod doffset];

% end mdlInitializeSizes

% Sampling period and offset for unit delay.
dperiod = 2;
doffset = 0;

switch flag
    %%%%%%%%%%%%%%
    % Initialization %
    %%%%%%%%%%%%%%
    case 0
        [sys,x0,str,ts]=mdlInitializeSizes(dperiod,doffset);
    %%%%%%%%%%%%%%
    % Derivatives %
    %%%%%%%%%%%%%%
    case 1
        sys=mdlDerivatives(t,x,u);
    %%%%%%%%%%%%%%
    % Update %
    %%%%%%%%%%%%%%
    case 2
        sys=mdlUpdate(t,x,u,dperiod,doffset);
    %%%%%%%%%%%%%%
    % Output %
    %%%%%%%%%%%%%%
    case 3
        sys=mdlOutputs(t,x,u,doffset,dperiod);
    %%%%%%%%%%%%%%
    % Terminate %
    %%%%%%%%%%%%%%
    case 9
        sys = []; % do nothing
end

%=====%
% mdlDerivatives %
% Handle discrete state updates, sample time hits, and major time st
% requirements.
%=====%
function sys=mdlDerivatives(t,x,u)
    sys = u;
    % end mdlDerivatives

%
%=====
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time st
% requirements.
%=====
function sys=mdlUpdate(t,x,u,dperiod,doffset)
    % next discrete state is output of the integrator
    if abs(round((t - doffset)/dperiod) - (t - doffset)/dperiod) < 1e-8
        sys = x(1);
    else
        sys = [];
    end
    % end mdlUpdate

%
%=====
% mdlOutputs
% Return the output vector for the S-function
%=====
function sys=mdlOutputs(t,x,u,doffset,dperiod)
    % Return output of the unit delay if we have a
    % sample hit within a tolerance of 1e-8. If we
    % don't have a sample hit, then return [] indicating
    % that the output shouldn't change.
    if abs(round((t - doffset)/dperiod) - (t - doffset)/dperiod) < 1e-8
        sys = x(2);
    else
        sys = [];
    end
    % end mdlOutputs

```

图 15 my_mixedm.m 中的改动

4. 变步长 S-Function

我编写的变步长系统 S-Function 模拟了第二个输入信号+1 为延时单位，输出第一个信号的延时，总模型中给出了对应的 Simulink 模块实现。

mS-Fcn 需要改动的地方如下：

```

function sys=mdlUpdate(t,x,u)
    sys = u(1);

    % end mdlUpdate

    %

%=====mdlOutputs=====
% Return the block outputs.
%=====mdlOutputs=====

function sys=mdlOutputs(t,x,u)
    sys = x(1);

    % end mdlOutputs

    %

%=====mdlGetTimeOfNextVarHit=====
% Return the time of the next hit for this t
% absolute time.
%=====mdlGetTimeOfNextVarHit=====

function sys=mdlGetTimeOfNextVarHit(t,x,u)
    sys = t + u(2)+1;

    % end mdlGetTimeOfNextVarHit

```

图 16 my_vsfunc.m 中的改动

5. 模型仿真结果

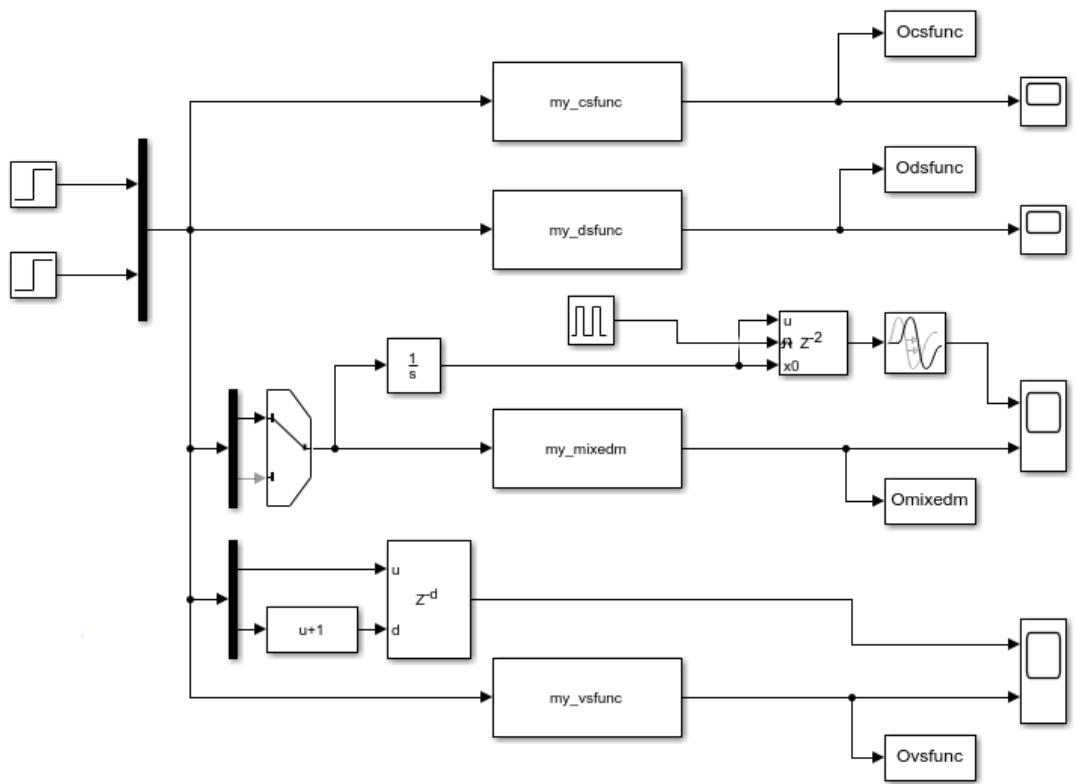


图 17 mS-Fcn 范例编写仿真模型

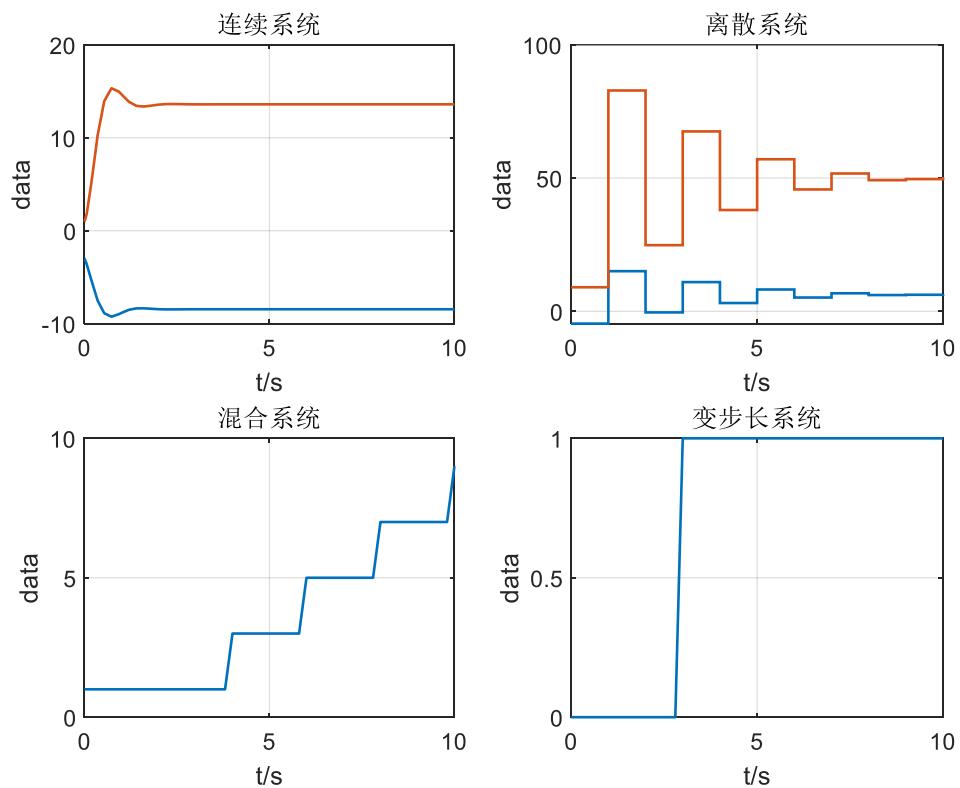


图 18 m-SFcн 范例编写模型输出响应曲线

附录：

mSFunctionHuatu.m

%% 倒立摆模型输出画图

clc, close all

```
sim('mSFunction_DaolibaiLianxi');
figure('Name','倒立摆状态输出响应曲线','NumberTitle','off')
subplot(2,2,1)
plot(z.time,z.data,'LineWidth',1)
xlabel('t/s'), ylabel('data/m')
grid on, title('z')
subplot(2,2,2)
plot(zd.time,zd.data,'LineWidth',1)
xlabel('t/s'), ylabel('data/m\cdot s^{-1}')
grid on, title('zd')
subplot(2,2,3)
plot(theta.time,theta.data,'LineWidth',1)
xlabel('t/s'), ylabel('data/deg')
grid on, title('\theta')
subplot(2,2,4)
plot(thetad.time,thetad.data,'LineWidth',1)
xlabel('t/s'), ylabel('data/deg\cdot s^{-1}')
grid on, title('{\theta}d')
```

%% m-SFcn范例模型输出画图

```
sim('mSFunction_Fanli');
figure('Name','m-SFcn范例模型输出响应曲线','NumberTitle','off')
subplot(2,2,1)
plot(Ocsfunc.time,Ocsfunc.data,'LineWidth',1)
xlabel('t/s'), ylabel('data')
grid on, title('连续系统')
subplot(2,2,2)
stairs(0dsfunc.time,0dsfunc.data,'LineWidth',1)
xlabel('t/s'), ylabel('data')
grid on, title('离散系统')
subplot(2,2,3)
plot(0mixedm.time,0mixedm.data,'LineWidth',1)
xlabel('t/s'), ylabel('data')
grid on, title('混合系统')
subplot(2,2,4)
plot(Ovsfunc.time,Ovsfunc.data,'LineWidth',1)
xlabel('t/s'), ylabel('data')
grid on, title('变步长系统')
```