Inverted Pendulum MATLAB Manual

Sheldon Logan

July 2, 2006

1 Table of Contents

2 Table of Figures

3 Introduction

The following manual contains information on all the MATLAB m-files, GUIs and models associated with the inverted pendulum project. The Manual will be separated into three sections. The first section will include information on the m-files, GUIs and models used to simulate various controllers for the inverted pendulum model. The second section will contain information on the m-files used to implement the various controller algorithms on the PIC microprocessor. The third section will contain information on the GUIs, m-files and models used to control in the inverted pendulum in real time from MATLAB.

4 Inverted Pendulum Simulations

There are six GUIs associated with the simulation of the inverted pendulum system. The first GUI, called StartPage is used to access 5 other GUIs. The second GUI, InvertedPendulum, simulates the inverted pendulum system subject to state space control. The third GUI, InvertedPendulumO simulates the system with an observer that estimates the pendulum and link velocity. The fourth GUI, InvertedPendulumM simulates the system with a Kalman observer that estimates the pendulum and link velocity. The fifth GUI, InvertedPendulumT also simulates the system with state space control but also implements tracking control of various signals. The sixth GUI, PendGainCalc, is used to calculate the Observer, Kalman and State Space gains used in the various simulations.

4.1 StartPage

The StartPage GUI pictured in Figure 1 below, allows the user to access the five other main GUIs by clicking the button associated with each GUI.

Figure 1: StartPage GUI

4.2 InvertedPendulum

The InvertedPendulum GUI is used to simulate state space control of the inverted pendulum system. To run a simulation the user should click the simulate model. The user has the options of changing several variables in model. For example the user can change the initial pendulum and link angles by changing the corresponding edit boxes in the

Simulation variable button group. The user can also change the Sample time, (T) the simulation length, the measurement noise variance (Mnoise) and process noise variance (Wnoise) by changing the edit boxes in the Simulation variables button group. The threshold value is used to switch between energy control and linear control. The pendulum variables button group contains mass and length estimates that are used to calculate the energy gain used in swing up control. The controller gains can be obtained from the workspace by selecting the gain lock radio button. This feature is useful since the gains are best calculated using the gain calculator which stores them to the workspace. Alternately if the user wants to change these gain values they can deselect

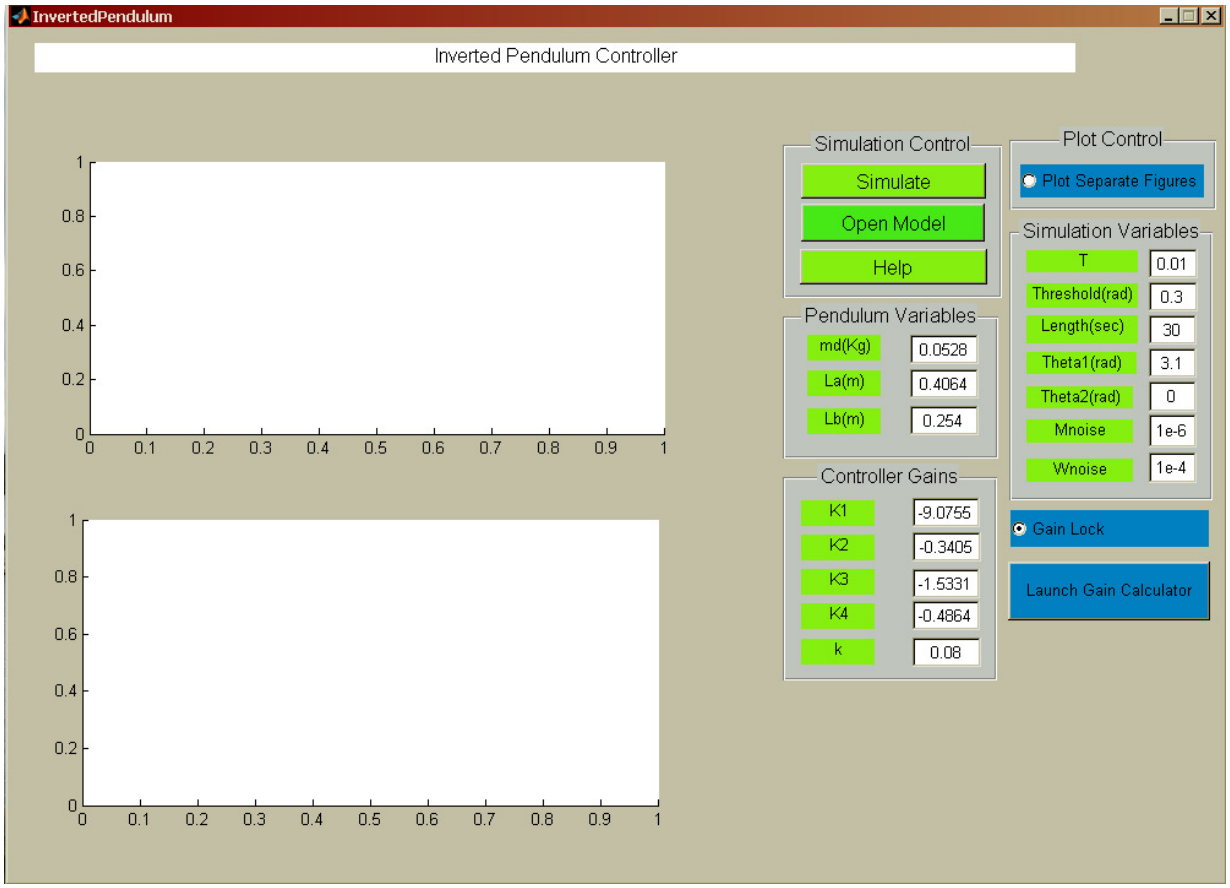

Figure 2: InvertedPendulum GUI

the gain lock radio button and change them in the edit box. The k value is the energy gain constant. The larger the k value the faster the pendulum will swing up, however large k values will lead to instability. The PendGainCalc GUI can be launched by clicking on the Launch Gain Calculator button. If the user desires to plot figures outside of the GUI the user can select the plot separate figures radio button.

4.3 InvertedPendulumO

The InvertedPendulumO GUI is similar to the InvertedPendulum GUI with the main difference being the ability to enter the observer gain matrix L into the Observer Gains button group. As with the Controller Gains button group the Observer Gains can

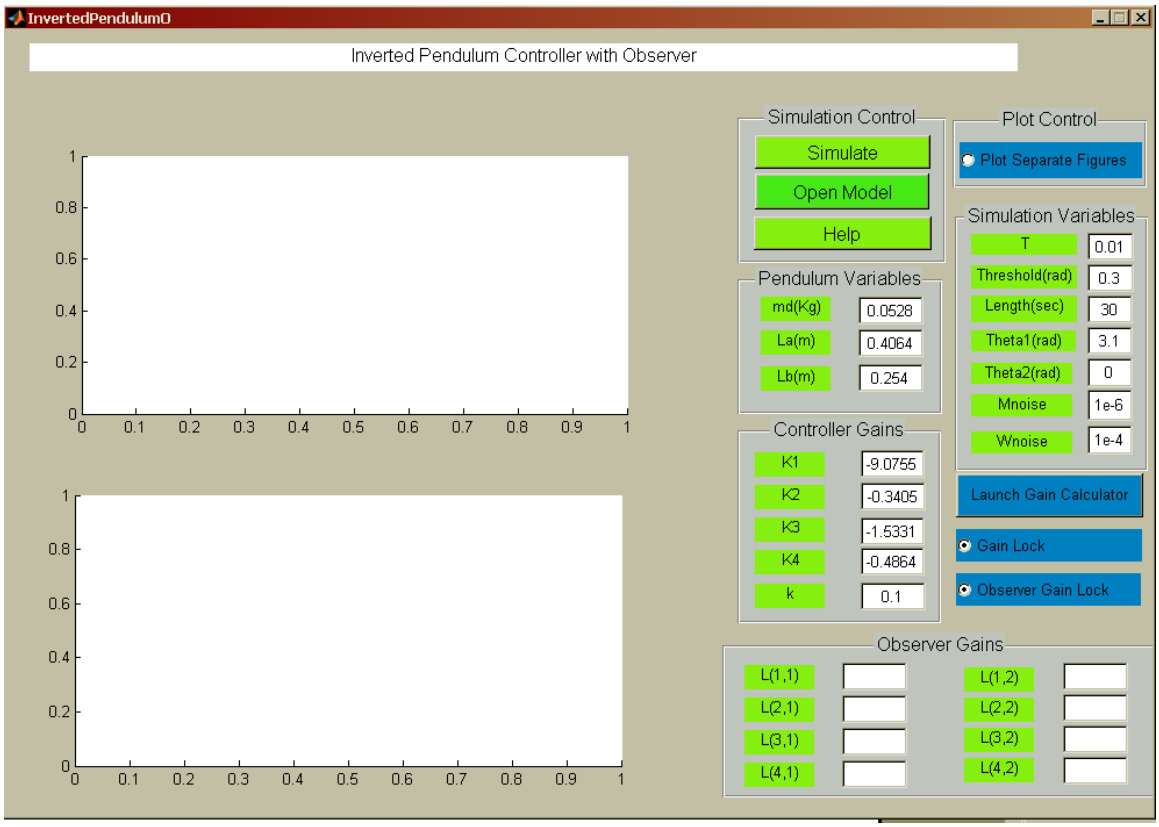

Figure 3: InvertedPendulumO GUI

either be entered in manually or taken from the workspace. Selecting the Observer Gain lock will acquire the L matrix from the workspace. The user can change the L matrix after they have deselected the Observer Gain Lock radio button.

4.4 InvertedPendulumM

The InvertedPendulumM GUI is similar to the InvertedPendulumO GUI with the main difference being the observer gain matrix L is replaced with the Kalman gain matrix M.

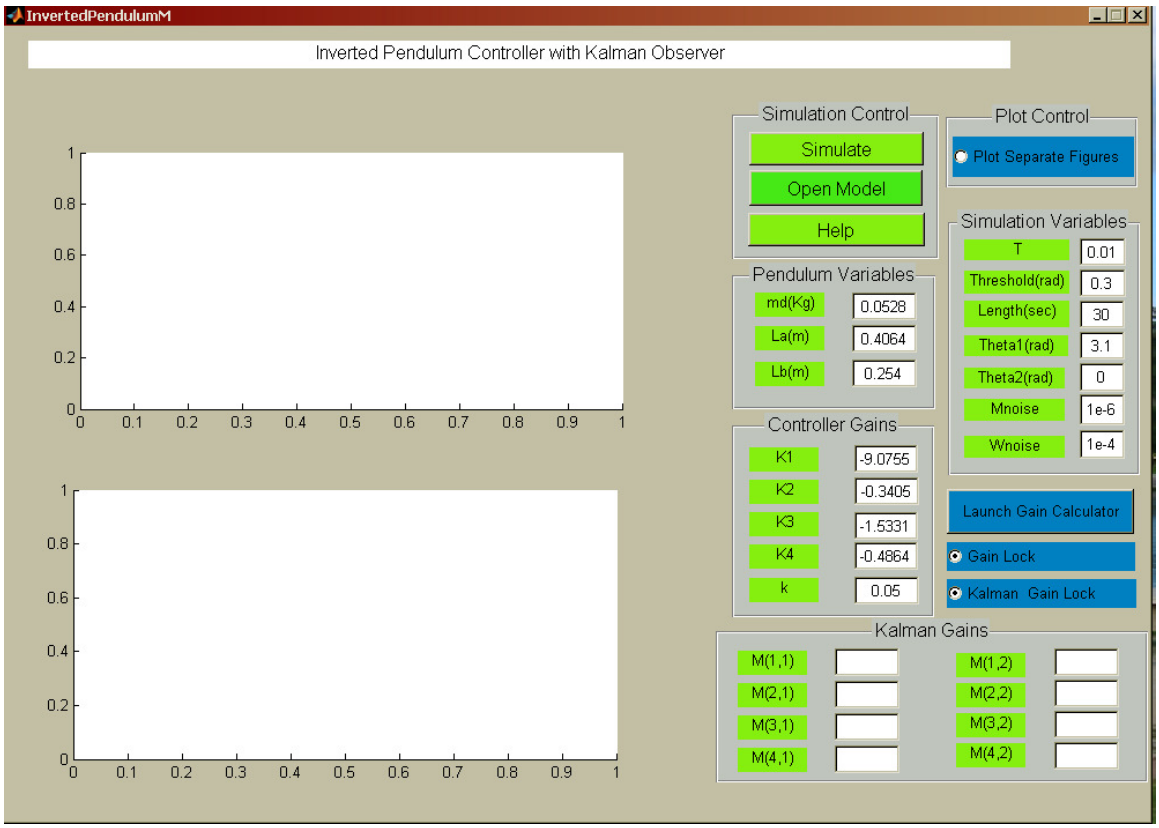

Figure 4: InvertedPendulumM GUI

4.5 InvertedPendulumT

The InvertedPendulumT GUI allows the user to enter tracking signals for either the pendulum or the link angle. The user has a choice between 4 tracking signals. A sine, square, sawtooth wave or a step input. The amplitude and frequency of the waves can be adjusted in the Reference Signal button group. The magnitude and time of the step input can also be adjusted in the GUI. The reference gains for the pendulum and link angle can be changed by the user in the Reference control button group.

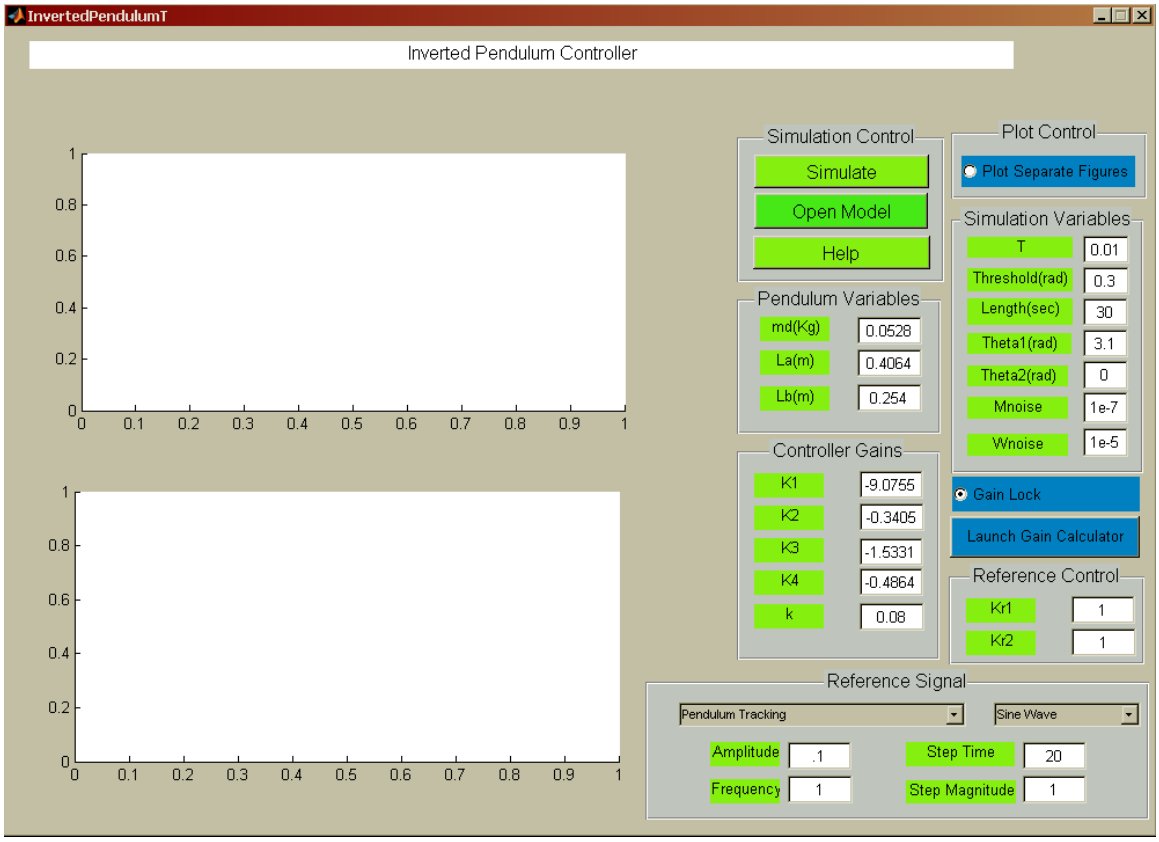

Figure 5: InvertedPendulumT GUI

4.6 PendGainCalc

The Controller, Kalman and Observer gains can be calculated using the PendGainCalc GUI. The user can specify estimates for the pendulum system variables in

the Pendulum Variables button group. The sample time (Ts) can be adjusted in the Controller Variables button group. The Q and R matrix for the linear quadratic regulator can also be adjusted in the Controller Variables button group. Observer poles can be placed to be X times faster than closed loop poles by setting the PoleDiv edit box to X. Kalman gains can be calculated for estimated measurement noise and process noise by changing the corresponding edit boxes in the Kalman Variables button group.

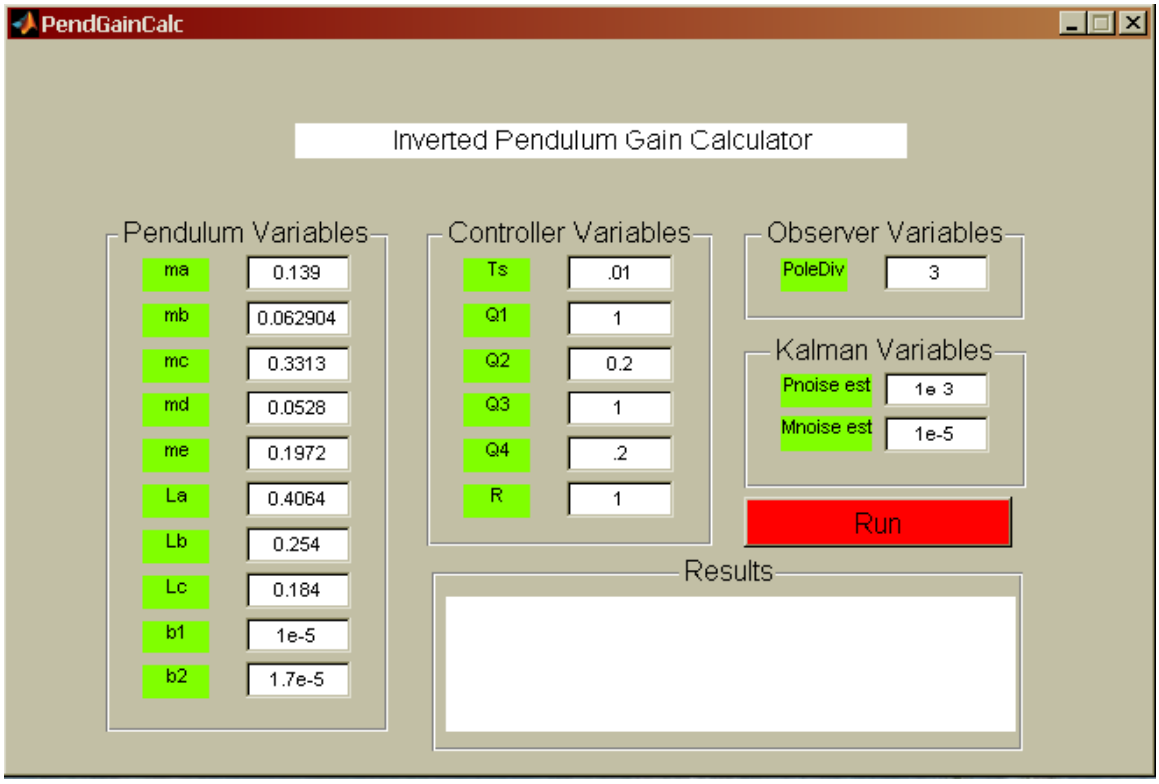

Figure 6: PendGainCalc GUI

5 Appendix A: MATLAB M-files

5.1 Inverted Pendulum System M-Files

5.1.1 arm_dynamicsmodf

This function is used to simulate the governing equations of the inverted pendulum system.

```
function thetadotdot = arm_dynamicsmodf(u)

tao = u(1:2);
theta = u(3:4);
thetadot = u(5:6);

theta1 = theta(1);
theta2 = theta(2);
theta1dot = thetadot(1);
theta2dot = thetadot(2);

La = .4064;
Lb = .254;                                  %lengths in metres
Lc = .184;

mb = 7820*0.25*pi*(0.00635^2)*Lb;           %masses in kg
me = .1972;
ma = 2700*0.25*pi*(0.0127^2)*La;
mc = .3313;
md = .1157-mb;

b1 = 1e-5;                                  %Nms/rad
b2 = 1.7e-5;

g = 9.81;                                   %m/s^2

alpha11 = (1/3)*Lb^2*mb + md*(Lb^2);
alpha12 = 0.5*Lb*mb*(La-Lc)*cos(theta1)+Lb*md*(La-Lc)*cos(theta1);
alpha21 = (1/2)*Lb*mb*cos(theta1)*(La-Lc)+Lb*md*cos(theta1)*(La-Lc);
alpha22 = (1/3)*Lb^2*mb*sin(theta1)^2 + me*(La-Lc)^2+(1/12)*ma*La^2+...
            mc*Lc^2+0.25*ma*(La-2*Lc)^2+mb*(La-Lc)^2+md*(La-
Lc)^2+md*Lb^2*sin(theta1)^2;
alpha = [alpha11 alpha12; alpha21 alpha22];

beta1 = -0.5*Lb*mb*g*sin(theta1)-
(1/3)*Lb^2*mb*sin(theta1)*cos(theta1)*theta2dot^2+...
        b1*theta1dot-Lb*md*g*sin(theta1)-
Lb^2*md*sin(theta1)*cos(theta1)*theta2dot^2;
beta2 = (2/3)*Lb^2*mb*sin(theta1)*cos(theta1)*theta2dot*theta1dot-...
        0.5*Lb*mb*sin(theta1)*(La-Lc)*theta1dot^2+b2*theta2dot+...
        Lb^2*2*md*sin(theta1)*cos(theta1)*theta2dot*theta1dot-...
```

```
        Lb*md*sin(theta1)*(La-Lc)*theta1dot^2;
beta = [beta1; beta2];

thetadotdot = inv(alpha)*(tao-beta);
```

5.1.2 wrap

This function is used to constrain the pendulum and link angles to be between -π and π.

```
function xx = wrap(u)

xx = mod(u+pi,2*pi)-pi;
```

5.2 GUI M-Files

5.2.1 initval

This function is used called when the StartPage GUI is opened. It places values for controller, observer and Kalman gains into the workspace.

```
K = [-9.0755 -0.3405 -1.5331 -0.4864];
M = [0.2661 -0.0012;-0.0012 0.2522;4.0740 -0.0017;-0.0774 3.6196];
Lo = [1.4110 0.0918;0.0283 1.4115;50.1533 6.1586;1.8113 49.6816];
```

5.2.2 Pendtestsim

This is the callback function for the InvertedPendulum GUI

```
clc;

handles = guihandles(gcbo);

if(get(handles.gl,'Value')==1)
    set(handles.K1,'String',num2str(K(1),'%5.4f'));
    set(handles.K2,'String',num2str(K(2),'%5.4f'));
    set(handles.K3,'String',num2str(K(3),'%5.4f'));
    set(handles.K4,'String',num2str(K(4),'%5.4f'));
end

K = [str2double(get(handles.K1, 'String'))...
     str2double(get(handles.K2, 'String'))...
     str2double(get(handles.K3, 'String'))...
     str2double(get(handles.K4, 'String'))];

k = str2double(get(handles.k, 'String'));
```

```
Vsat = 6;            %Volts
R = 1.6;             %Ohms
L = 4.1e-3;          %Henry
Ke  = 9.7403e-2;     %V sec/rad
Kt = 0.0923;         %Nm/A

Lb = str2double(get(handles.Lb, 'String'));
La = str2double(get(handles.La, 'String'));
md = str2double(get(handles.md, 'String'));
mb = 7820*0.25*pi*(0.00635^2)*Lb;
ma = 2700*0.25*pi*(0.0127^2)*La;

T1 = 0.5*9.81*Lb*(mb+2*md);
T2 = (.1667*mb*Lb^2) + (0.5*md*Lb^2);
T3 = 0.04166667*ma*(La^2);

theta10 = str2double(get(handles.theta10, 'String'));
theta20 = str2double(get(handles.theta20, 'String'));
T = str2double(get(handles.T, 'String'));
thres = str2double(get(handles.thres, 'String'));

wn = str2double(get(handles.wn, 'String'));
mn = str2double(get(handles.mn, 'String'));

tspan = str2double(get(handles.tspan, 'String'));

[t,x,y]=sim('PendtestDiscrete',tspan,simset('solver','ode45'));

psfigval  = get(handles.psfig,'Value');

if (psfigval == 0)
    axes(handles.axes1)
    plot(t,y(:,1))
    title('Pendulum angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_1')

    axes(handles.axes2)
    plot(t,y(:,2))
    title('Horizantal link angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_2')
else
    figure(1)
    plot(t,y(:,1))
    title('Pendulum angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_1')

    figure(2)
    plot(t,y(:,2))
    title('Horizantal link angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_2')
```

```
end
```

5.2.3 Pendtestsimoberv

This is the callback function for the InvertedPendulumO GUI

```
clc;

handles = guihandles(gcbo);

if(get(handles.gl,'Value')==1)
    set(handles.K1,'String',num2str(K(1),'%5.4f'));
    set(handles.K2,'String',num2str(K(2),'%5.4f'));
    set(handles.K3,'String',num2str(K(3),'%5.4f'));
    set(handles.K4,'String',num2str(K(4),'%5.4f'));
end

K = [str2double(get(handles.K1, 'String'))...
     str2double(get(handles.K2, 'String'))...
     str2double(get(handles.K3, 'String'))...
     str2double(get(handles.K4, 'String'))];

k = str2double(get(handles.k, 'String'));

Ad =[1.0026         0    0.0100    0.0000;...
   -0.0004    1.0000   -0.0000    0.0100;...
    0.5294         0    1.0026    0.0000;...
   -0.0879         0   -0.0004    1.0000];

Bd = [-0.0021;0.0021;-0.4188;0.4185];
C = [1 0 0 0;0 1 0 0];

if(get(handles.ogl,'Value') == 1)
    set(handles.L11,'String',num2str(Lo(1,1),'%5.4f'));
    set(handles.L21,'String',num2str(Lo(2,1),'%5.4f'));
    set(handles.L31,'String',num2str(Lo(3,1),'%5.4f'));
    set(handles.L41,'String',num2str(Lo(4,1),'%5.4f'));
    set(handles.L12,'String',num2str(Lo(1,2),'%5.4f'));
    set(handles.L22,'String',num2str(Lo(2,2),'%5.4f'));
    set(handles.L32,'String',num2str(Lo(3,2),'%5.4f'));
    set(handles.L42,'String',num2str(Lo(4,2),'%5.4f'));
end

Lo = [str2double(get(handles.L11, 'String'))...
      str2double(get(handles.L12, 'String'));...
      str2double(get(handles.L21, 'String'))...
      str2double(get(handles.L22, 'String'));...
      str2double(get(handles.L31, 'String'))...
      str2double(get(handles.L32, 'String'));...
      str2double(get(handles.L41, 'String'))...
      str2double(get(handles.L42, 'String'))];
```

```
Vsat = 6;            %Volts
R = 1.6;             %Ohms
L = 4.1e-3;          %Henry
Ke  = 9.7403e-2;     %V sec/rad
Kt = 0.0923;         %Nm/A

Lb = str2double(get(handles.Lb, 'String'));
La = str2double(get(handles.La, 'String'));
md = str2double(get(handles.md, 'String'));
mb = 7820*0.25*pi*(0.00635^2)*Lb;
ma = 2700*0.25*pi*(0.0127^2)*La;

T1 = 0.5*9.81*Lb*(mb+2*md);
T2 = (.1667*mb*Lb^2) + (0.5*md*Lb^2);
T3 = 0.04166667*ma*(La^2);

theta10 = str2double(get(handles.theta10, 'String'));
theta20 = str2double(get(handles.theta20, 'String'));
T = str2double(get(handles.T, 'String'));
thres = str2double(get(handles.thres, 'String'));

wn = str2double(get(handles.wn, 'String'));
mn = str2double(get(handles.mn, 'String'));

tspan = str2double(get(handles.tspan, 'String'));

[t,x,y]=sim('PendtestDiscreteObserv',tspan,simset('solver','ode45'));

psfigval  = get(handles.psfig,'Value');

if (psfigval == 0)
    axes(handles.axes1)
    plot(t,y(:,1))
    title('Pendulum angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_1')

    axes(handles.axes2)
    plot(t,y(:,2))
    title('Horizantal link angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_2')
else
    figure(1)
    plot(t,y(:,1))
    title('Pendulum angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_1')

    figure(2)
    plot(t,y(:,2))
    title('Horizantal link angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_2')
end
```

5.2.4 PendtestsimKalman

This is the callback function for the InvertedPendulumM GUI

```
clc;

handles = guihandles(gcbo);

if(get(handles.gl,'Value')==1)
    set(handles.K1,'String',num2str(K(1),'%5.4f'));
    set(handles.K2,'String',num2str(K(2),'%5.4f'));
    set(handles.K3,'String',num2str(K(3),'%5.4f'));
    set(handles.K4,'String',num2str(K(4),'%5.4f'));
end

K = [str2double(get(handles.K1, 'String'))...
     str2double(get(handles.K2, 'String'))...
     str2double(get(handles.K3, 'String'))...
     str2double(get(handles.K4, 'String'))];

k = str2double(get(handles.k, 'String'));

Ad =[1.0026         0    0.0100    0.0000;...
   -0.0004    1.0000   -0.0000    0.0100;...
    0.5294         0    1.0026    0.0000;...
   -0.0879         0   -0.0004    1.0000];
Bd = [-0.0021;0.0021;-0.4188;0.4185];
C = [1 0 0 0;0 1 0 0];

if(get(handles.kgl,'Value') == 1)
    set(handles.M11,'String',num2str(M(1,1),'%5.4f'));
    set(handles.M21,'String',num2str(M(2,1),'%5.4f'));
    set(handles.M31,'String',num2str(M(3,1),'%5.4f'));
    set(handles.M41,'String',num2str(M(4,1),'%5.4f'));
    set(handles.M12,'String',num2str(M(1,2),'%5.4f'));
    set(handles.M22,'String',num2str(M(2,2),'%5.4f'));
    set(handles.M32,'String',num2str(M(3,2),'%5.4f'));
    set(handles.M42,'String',num2str(M(4,2),'%5.4f'));
end

M = [str2double(get(handles.M11, 'String'))...
      str2double(get(handles.M12, 'String'));...
      str2double(get(handles.M21, 'String'))...
      str2double(get(handles.M22, 'String'));...
      str2double(get(handles.M31, 'String'))...
      str2double(get(handles.M32, 'String'));...
      str2double(get(handles.M41, 'String'))...
      str2double(get(handles.M42, 'String'))];

Vsat = 6;           %Volts
R = 1.6;            %Ohms
L = 4.1e-3;         %Henry
Ke  = 9.7403e-2;    %V sec/rad
Kt = 0.0923;        %Nm/A
```

```
Lb = str2double(get(handles.Lb, 'String'));
La = str2double(get(handles.La, 'String'));
md = str2double(get(handles.md, 'String'));
mb = 7820*0.25*pi*(0.00635^2)*Lb;
ma = 2700*0.25*pi*(0.0127^2)*La;

T1 = 0.5*9.81*Lb*(mb+2*md);
T2 = (.1667*mb*Lb^2) + (0.5*md*Lb^2);
T3 = 0.04166667*ma*(La^2);

theta10 = str2double(get(handles.theta10, 'String'));
theta20 = str2double(get(handles.theta20, 'String'));
T = str2double(get(handles.T, 'String'));
thresGain = str2double(get(handles.thres, 'String'));

wn = str2double(get(handles.wn, 'String'));
mn = str2double(get(handles.mn, 'String'));

tspan = str2double(get(handles.tspan, 'String'));
thresLin = 0.1;

[t,x,y]=sim('PendtestDiscreteKalman',tspan,simset('solver','ode45'));

psfigval  = get(handles.psfig,'Value');

if (psfigval == 0)
    axes(handles.axes1)
    plot(t,y(:,1))
    title('Pendulum angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_1')

    axes(handles.axes2)
    plot(t,y(:,2))
    title('Horizantal link angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_2')
else
    figure(1)
    plot(t,y(:,1))
    title('Pendulum angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_1')

    figure(2)
    plot(t,y(:,2))
    title('Horizantal link angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_2')
end
```

5.2.5 PendtestsimTrack

This is the callback function for the InvertedPendulumT GUI

```
clc;

handles = guihandles(gcbo);

if(get(handles.gl,'Value')==1)
    set(handles.K1,'String',num2str(K(1),'%5.4f'));
    set(handles.K2,'String',num2str(K(2),'%5.4f'));
    set(handles.K3,'String',num2str(K(3),'%5.4f'));
    set(handles.K4,'String',num2str(K(4),'%5.4f'));
end

K = [str2double(get(handles.K1, 'String'))...
     str2double(get(handles.K2, 'String'))...
     str2double(get(handles.K3, 'String'))...
     str2double(get(handles.K4, 'String'))];

k = str2double(get(handles.k, 'String'));

Vsat = 6;           %Volts
R = 1.6;            %Ohms
L = 4.1e-3;         %Henry
Ke  = 9.7403e-2;    %V sec/rad
Kt = 0.0923;        %Nm/A

Lb = str2double(get(handles.Lb, 'String'));
La = str2double(get(handles.La, 'String'));
md = str2double(get(handles.md, 'String'));
mb = 7820*0.25*pi*(0.00635^2)*Lb;
ma = 2700*0.25*pi*(0.0127^2)*La;

T1 = 0.5*9.81*Lb*(mb+2*md);
T2 = (.1667*mb*Lb^2) + (0.5*md*Lb^2);
T3 = 0.04166667*ma*(La^2);

theta10 = str2double(get(handles.theta10, 'String'));
theta20 = str2double(get(handles.theta20, 'String'));
T = str2double(get(handles.T, 'String'));
thres = str2double(get(handles.thres, 'String'));

wn = str2double(get(handles.wn, 'String'));
mn = str2double(get(handles.mn, 'String'));

tspan = str2double(get(handles.tspan, 'String'));

swti = get(handles.whotrack,'Value'); %1 = pendulum tracking,2 = link
tracking

if  isempty(find_system('Name','PendtestDiscreteTrack')),
    open_system('PendtestDiscreteTrack');
```

```
end

stime  = str2double(get(handles.stime, 'String'));      %step time
sfinal = str2double(get(handles.sfinal, 'String'));     %step final
value

Kr1 = str2double(get(handles.Kr1, 'String'));
Kr2 = str2double(get(handles.Kr2, 'String'));

atrack = get(handles.atrack, 'String');
ftrack = get(handles.ftrack, 'String');

signtype = get(handles.genmenu,'Value'); %1 = sine, 2 = square, 3 =
sawtooth, 4 = step

switch(signtype)

    case 1
        swti2 = 1;
        set_param('PendtestDiscreteTrack/Controller/Link Tracking
Input/Signal Generator',...
        'WaveForm','sine','Amplitude',atrack,'Frequency',ftrack)
        set_param('PendtestDiscreteTrack/Controller/Pendulum Tracking
Input/Signal Generator',...
        'WaveForm','sine','Amplitude',atrack,'Frequency',ftrack)

    case 2
        swti2 = 1;
        set_param('PendtestDiscreteTrack/Controller/Link Tracking
Input/Signal Generator',...
        'WaveForm','square','Amplitude',atrack,'Frequency',ftrack)
        set_param('PendtestDiscreteTrack/Controller/Pendulum Tracking
Input/Signal Generator',...
        'WaveForm','square','Amplitude',atrack,'Frequency',ftrack)

    case 3
        swti2 = 1;
        set_param('PendtestDiscreteTrack/Controller/Link Tracking
Input/Signal Generator',...
        'WaveForm','sawtooth','Amplitude',atrack,'Frequency',ftrack)
        set_param('PendtestDiscreteTrack/Controller/Pendulum Tracking
Input/Signal Generator',...
        'WaveForm','sawtooth','Amplitude',atrack,'Frequency',ftrack)

    case 4
        swti2 = 2;
    otherwise

end

save_system('PendtestDiscreteTrack');
close_system('PendtestDiscreteTrack');

[t,x,y]=sim('PendtestDiscreteTrack',tspan,simset('solver','ode45'));
```

```
psfigval  = get(handles.psfig,'Value');

if (psfigval == 0)
    axes(handles.axes1)
    plot(t,y(:,1))
    title('Pendulum angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_1')

    axes(handles.axes2)
    plot(t,y(:,2))
    title('Horizantal link angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_2')
else
    figure(1)
    plot(t,y(:,1))
    title('Pendulum angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_1')

    figure(2)
    plot(t,y(:,2))
    title('Horizantal link angle vs Time')
    xlabel('time(s)')
    ylabel('\theta_2')
end
```

5.2.6 part1fdiscretesim

This is the callback function for the PendGainCalc GUI

```
clc;

handles = guihandles(gcbo);

syms theta1 theta2 theta1dot theta2dot
syms ma mb mc md me La Lb Lc g b1 b2
syms tao2
tao = [0; tao2];

alpha11 = (1/3)*Lb^2*mb + md*(Lb^2);
alpha12 = 0.5*Lb*mb*(La-Lc)*cos(theta1)+Lb*md*(La-Lc)*cos(theta1);
alpha21 = (1/2)*Lb*mb*cos(theta1)*(La-Lc)+Lb*md*cos(theta1)*(La-Lc);
alpha22 = (1/3)*Lb^2*mb*sin(theta1)^2 + me*(La-Lc)^2+(1/12)*ma*La^2+...
            mc*Lc^2+0.25*ma*(La-2*Lc)^2+mb*(La-Lc)^2+md*(La-
Lc)^2+md*Lb^2*sin(theta1)^2;
alpha = [alpha11 alpha12; alpha21 alpha22];

beta1 = -0.5*Lb*mb*g*sin(theta1)-
(1/3)*Lb^2*mb*sin(theta1)*cos(theta1)*theta2dot^2+...
```

```
        b1*theta1dot-Lb*md*g*sin(theta1)-
Lb^2*md*sin(theta1)*cos(theta1)*theta2dot^2;
beta2 = (2/3)*Lb^2*mb*sin(theta1)*cos(theta1)*theta2dot*theta1dot-...
        0.5*Lb*mb*sin(theta1)*(La-Lc)*theta1dot^2+b2*theta2dot+...
        Lb^2*2*md*sin(theta1)*cos(theta1)*theta2dot*theta1dot-...
        Lb*md*sin(theta1)*(La-Lc)*theta1dot^2;
beta = [beta1; beta2];

thetadotdot = inv(alpha)*(tao-beta);

A = jacobian(thetadotdot,[theta1 theta2 theta1dot theta2dot]);
A = [0 0 1 0; 0 0 0 1; A];
A = subs(A,[theta1, theta2, theta1dot, theta2dot],[0,0,0,0]);
A =
subs(A,[ma,mb,mc,md,me,La,Lb,Lc,b1,b2,g],[str2double(get(handles.ma,
'String'))...

str2double(get(handles.mb, 'String'))...

str2double(get(handles.mc, 'String'))...

str2double(get(handles.md, 'String'))...

str2double(get(handles.me, 'String'))...

str2double(get(handles.La, 'String'))...

str2double(get(handles.Lb, 'String'))...

str2double(get(handles.Lc, 'String'))...

str2double(get(handles.b1, 'String'))...

str2double(get(handles.b2, 'String'))...
                                              9.81]);

B = jacobian(thetadotdot,tao2);
B = [0;0;B];
B = subs(B,[theta1, theta2, theta1dot, theta2dot],[0,0,0,0]);
B =
subs(B,[ma,mb,mc,md,me,La,Lb,Lc,b1,b2,g],[str2double(get(handles.ma,
'String'))...

str2double(get(handles.mb, 'String'))...

str2double(get(handles.mc, 'String'))...

str2double(get(handles.md, 'String'))...

str2double(get(handles.me, 'String'))...

str2double(get(handles.La, 'String'))...

str2double(get(handles.Lb, 'String'))...

str2double(get(handles.Lc, 'String'))...
```

```
str2double(get(handles.b1, 'String'))...

str2double(get(handles.b2, 'String'))...
                                                    9.81]);
B = eval(B);

C = [1 0 0 0;0 1 0 0];
D = [0;0];
sys = ss(A,B,C,D);

Ts = str2double(get(handles.Ts, 'String'));

[Ad,Bd] = c2d(A,B,Ts);
observrk = rank([(Ad^3)*Bd (Ad^2)*Bd Ad*Bd Bd]);
contrork = rank([C*(Ad^3) C*(Ad^2) C*Ad C]);

Q = diag([str2double(get(handles.Q1, 'String'))...
          str2double(get(handles.Q2, 'String'))...
          str2double(get(handles.Q3, 'String'))...
          str2double(get(handles.Q4, 'String'))]);
R = str2double(get(handles.R, 'String'));

[K,S,e] = dlqr(Ad,Bd,Q,R);
Kstr =['K = [ ', num2str(K,'%5.4f'),' ]'];
set(handles.rbox,'String',Kstr)


Qn = str2double(get(handles.Qn, 'String'))*eye(4);
Rn = str2double(get(handles.Rn, 'String'))*eye(2);

[M,P,Z,ke] = dlqe(Ad,diag(Bd),C,Qn,Rn);

Lp = e/(str2double(get(handles.obsvdiv, 'String')));
Lo = (place(Ad',C',Lp)).';
```

5.3 M-files for Microprocessor Control

5.3.1 gaincalc

This function is used to calculate the gains to be used in the microprocessor. The user should copy and past the K vector into the function. The function then outputs the gains to be copied to the c code.

```
clc

Lb = .254;
mb = 7820*0.25*pi*(0.00635^2)*Lb;
La = .4064;
```

```
ma = 2700*0.25*pi*(0.0127^2)*La;
md = .1157-mb;

k = .12;
Kt = 0.0923;
R = 1.6;
t = 0.01;                       %sample time
V = 6;


K = [-9.0755   -0.3405   -1.5331   -0.4864];

Pe2 = k*(R/Kt)*0.5*9.81*(mb+2*md)*Lb*((2*pi/8192)^2)*(1/V)*255

Ke1 =
k*(R/Kt)*(1/(t^2))*((1/6)*mb+0.5*md)*(Lb^2)*((2*pi/8192)^2)*(1/V)*255

Ke2 = k*(R/Kt)*(1/(t^2))*
0.04166667*ma*(La^2)*((2*pi/4000)^2)*(1/V)*255

K1 = (R/Kt)*255*K(1)*2*pi/(8192*V)
K2 = (R/Kt)*255*K(2)*2*pi/(4000*V)
K3 = (R/Kt)*255*K(3)*2*pi/(8192*t*V)
K4 = (R/Kt)*255*K(4)*2*pi/(4000*V*t)
```

5.3.2 gaincalcobs

This function is used to calculate the K gains to be used in the observer c code. The user should copy and past the K vector into the function. The function then outputs the K gains to be copied to the observer c code.

```
clc

Lb = .254;
mb = 7820*0.25*pi*(0.00635^2)*Lb;
La = .4064;
ma = 2700*0.25*pi*(0.0127^2)*La;
md = .0528;

k = .3;
Kt = 0.0923;
R = 1.6;
t = 0.01;
V = 6;

K = [-9.0755   -0.3405   -1.5331   -0.4864];

Pe2 = k*(R/Kt)*0.5*(mb+2*md)*9.81*Lb*(1/V)*255
Ke1 = k*(R/Kt)*((1/6)*mb+0.5*md)*(Lb^2)*(1/V)*255
Ke2 = k*(R/Kt)*0.04166667*ma*(La^2)*(1/V)*255
```

```
K1 = (R/Kt)*255*K(1)/V
K2 = (R/Kt)*255*K(2)/V
K3 = (R/Kt)*255*K(3)/V
K4 = (R/Kt)*255*K(4)/V
torqgain = V*Kt/(R*255)
pendposgain = 2*pi/8192
motorposgain = 2*pi/4000
```

5.3.3 Kalmcoecalc

This function is used to calculate the coefficients of the Kalman observer in the microprocessor C code. The user should paste the M matrix into the function. The function will then perform the Kalman new estimate calculation. The user can then look at the various estimate equations and copy and past the coefficients into the C code.

```
clc;

Ad =[1.0026          0    0.0100    0.0000;...
   -0.0004    1.0000   -0.0000    0.0100;...
    0.5294          0    1.0026    0.0000;...
   -0.0879          0   -0.0004    1.0000];
Bd = [-0.0021;0.0021;-0.4188;0.4185];

M = [0.2661 -0.0012;-0.0012 0.2522;4.0740 -0.0017;-0.0774 3.6196];

syms theta1est theta2est theta1dotest theta2dotest torq theta1 theta2

xnest = [theta1est;theta2est;theta1dotest;theta2dotest];
yn = [theta1;theta2];
ynest = [theta1est;theta2est];

xnnewestM = (Ad*xnest)+(Bd*torq)+(Ad*M*(yn-ynest));
```

5.3.4 obscoecalc

This function is used to calculate the coefficients of the observer in the microprocessor C code. The user should paste the Lo matrix into the function. The function will then perform the observer new estimate calculation. The user can then look at the various estimate equations and copy and past the coefficients into the C code.

```
clc;
```

```
syms theta1est theta2est theta1dotest theta2dotest torq theta1 theta2

Ad =[1.0026         0    0.0100    0.0000;...
   -0.0004    1.0000   -0.0000    0.0100;...
    0.5294         0    1.0026    0.0000;...
   -0.0879         0   -0.0004    1.0000];
Bd = [-0.0021;0.0021;-0.4188;0.4185];

%2X poles
% Lo = [1.4154 0.0934;0.0288 1.4159;50.4511 6.2638;1.8439 49.9801];

%3X poles
% Lo = [1.1140 0.1378;0.0429 1.1171;31.2951 6.9650;2.0714 30.9170];

%5X poles
% Lo = [1.6487 0.0551;0.0167 1.6469;68.3836 4.4226;1.2548 67.7620];

%10X poles
Lo = [1.8270 0.0275;0.0079 1.8235;83.8897 2.4844;0.6343 83.1129];

xnest = [theta1est;theta2est;theta1dotest;theta2dotest];
yn = [theta1;theta2];
ynest = [theta1est;theta2est];

xnnewestO = (Ad*xnest)+(Bd*torq)+(Lo*(yn-ynest));
```

6 Appendix B: MATLAB Models

6.1 Inverted Pendulum Simulation Models

6.1.1 Basic Inverted Pendulum Model

A Simulink model of the inverted pendulum system is shown below in Figure 7. The governing equations of the system are contained in the m-file arm_dynamicsmodf.

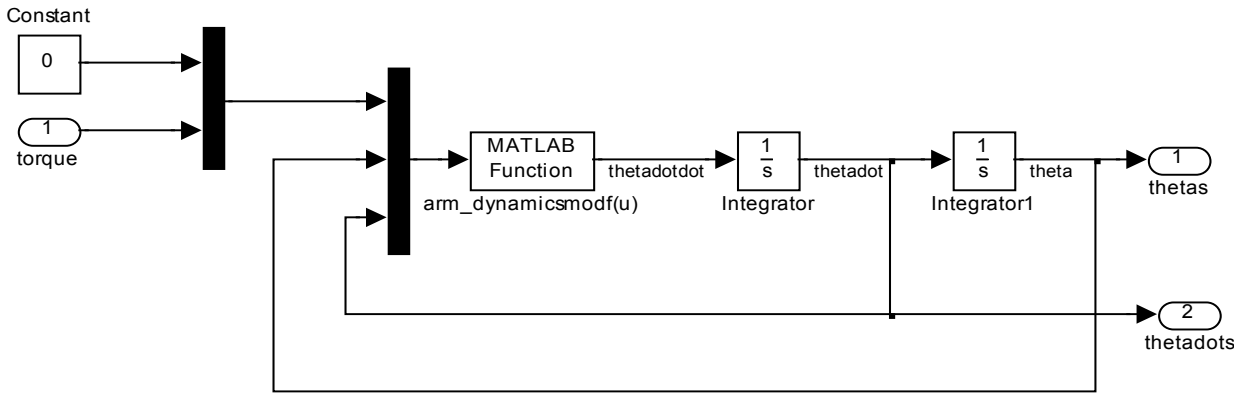

Figure 7: Inverted Pendulum System

A Simulink model of the motor system used to stabilize the inverted pendulum is shown below in Figure 8.

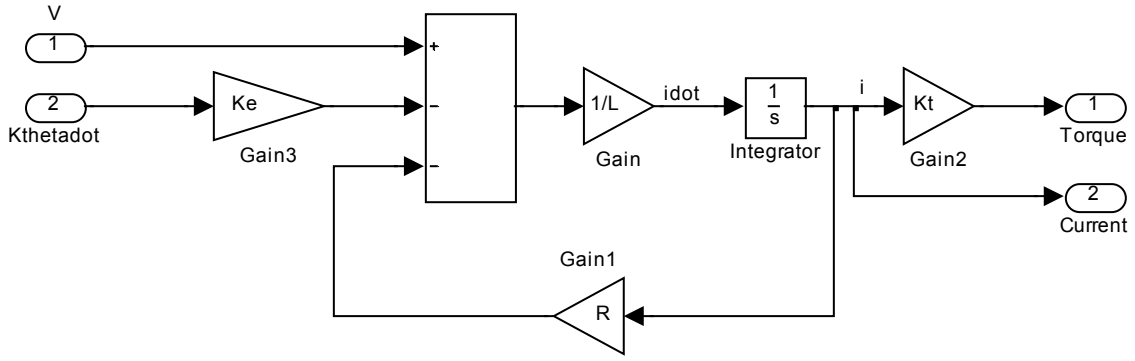

Figure 8: Motor Drive System

A Simulink model of the combined motor and inverted pendulum system is shown below in Figure 9.

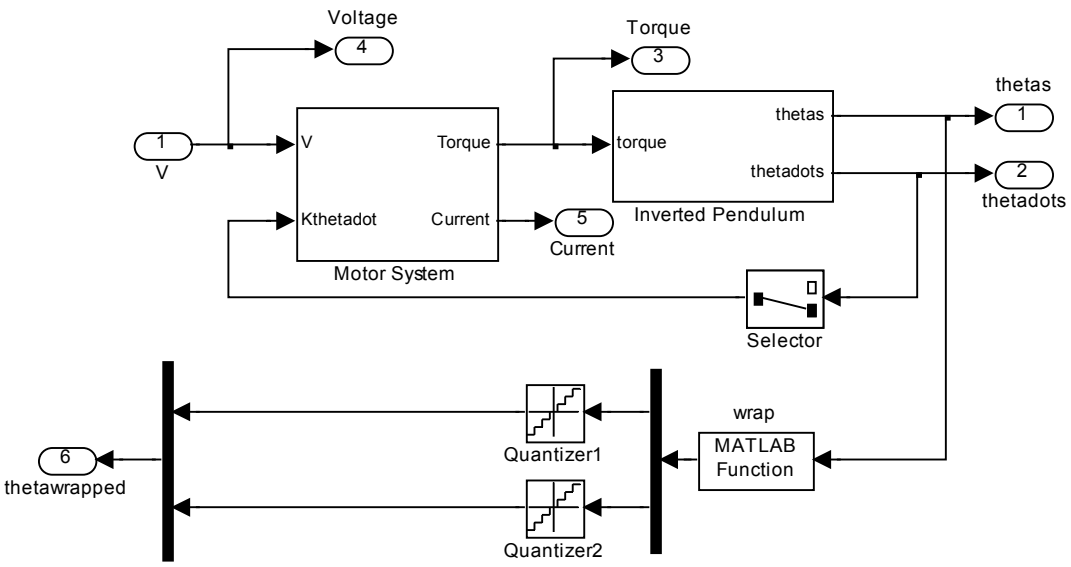

Figure 9: Combined Motor and Inverted Pendulum System

The m-file wrap, is used to constrain the pendulum and link angles to be between $-\pi$ and π radians. The quantizers are used to simulate the quantizing behavior of the encoders.

6.1.2 State Space Controller

A Simulink model of the state space controller for the inverted pendulum system is shown below. The model simulates the A/D and D/A conversions as zero order holds. Measurement and Process Noise is also added to model to ensure the results of the simulation are realistic.

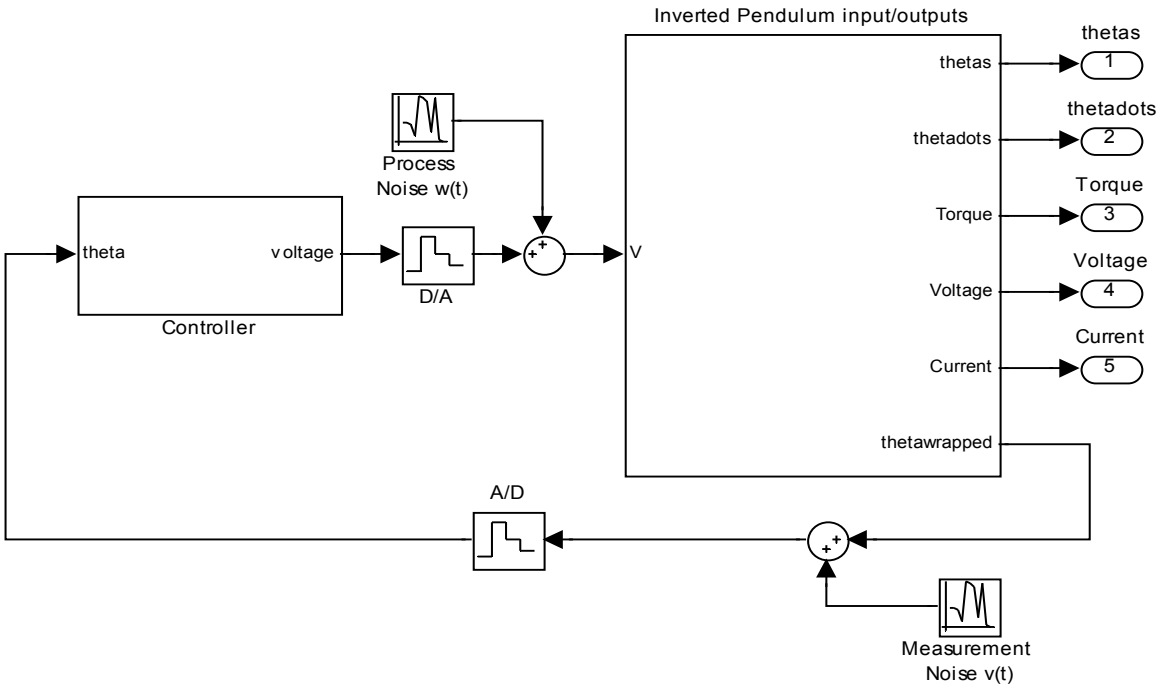

Figure 10: State Space Controller

The controller subsystem is shown below in Figure 11 . Due to the limitations of the motor and the H-Bridge used to drive the motor, the voltage was allowed to saturate at ± 6V. A discrete differentiation of the pendulum and link positions was used to obtain the velocities of the pendulum and link respectively. A Switch was used to alternate between swing up (energy control) and linear control. The threshold for the switch was determined by angle of the pendulum.

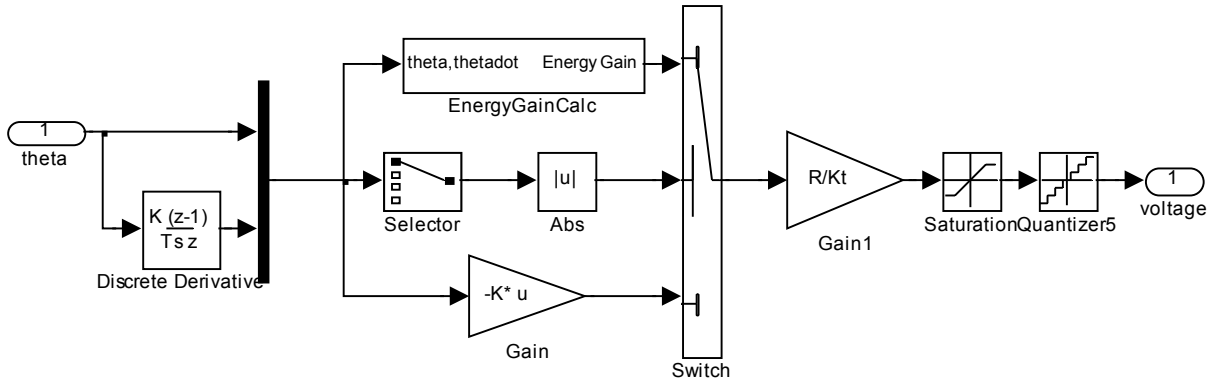

Figure 11: Controller Subsystem

The EnergyGainCalc subsystem is pictured below. The Tselect subsystem contains a demux so as to separate the various states for the IEnergy subsystem.

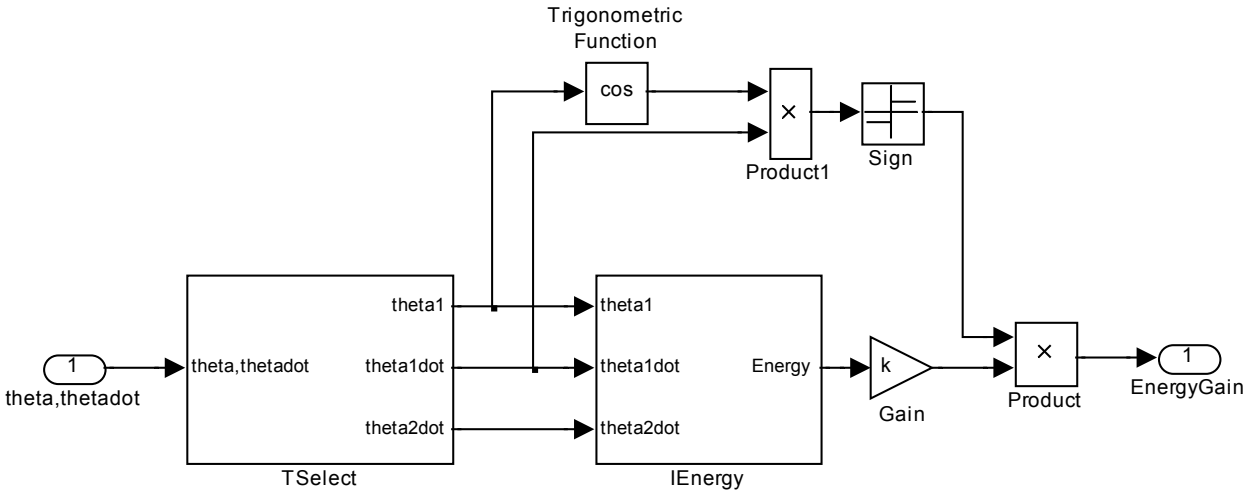

Figure 12: EnergyGainCalc subsystem

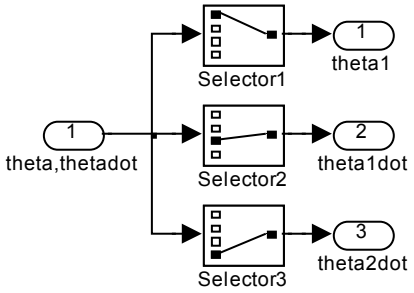

Figure 13: Tselect subsystem

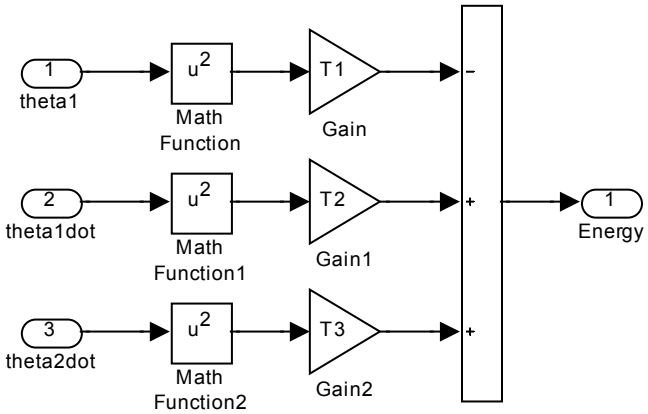

Figure 14: IEnergy subsystem

6.1.3 State Space Controller with Observer

A simulink model of the state space controller with observer is shown below in Figure 15. The Kt/R gain is used to convert the voltage output of the controller to a torque to be used in the observer estimates.

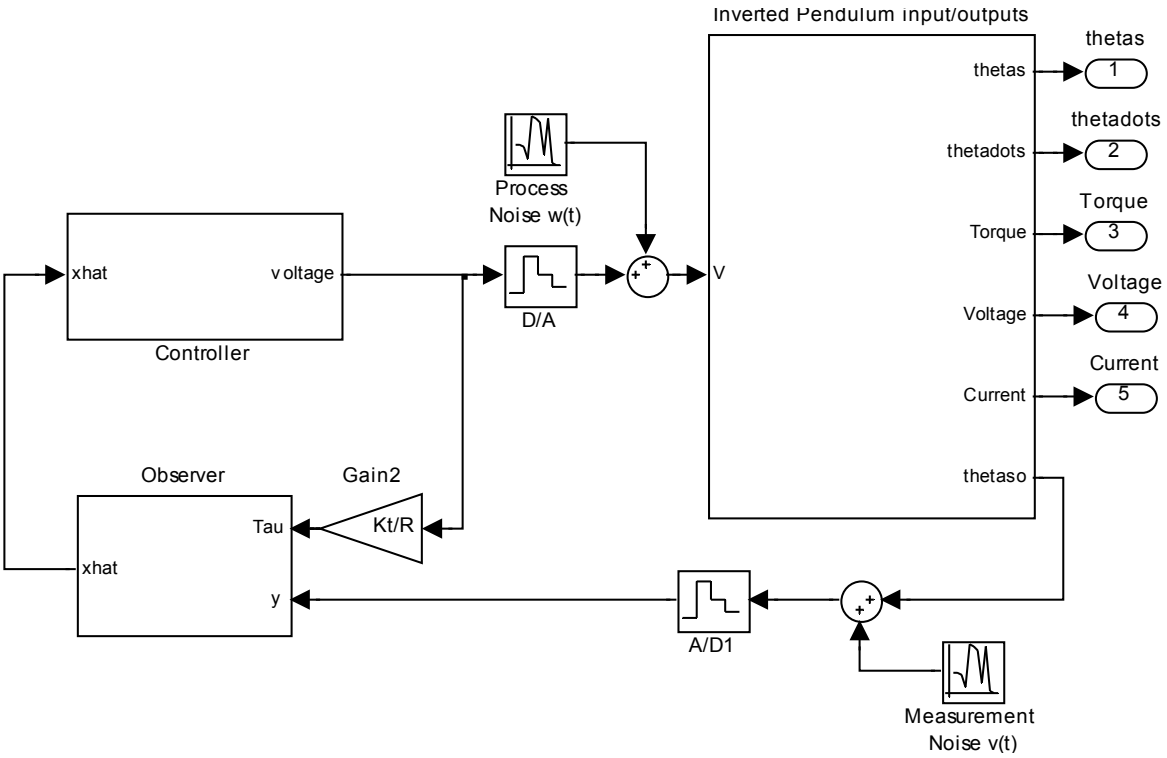

Figure 15: State Space Controller with Observer

A detailed model of the observer subsystem can be found below.

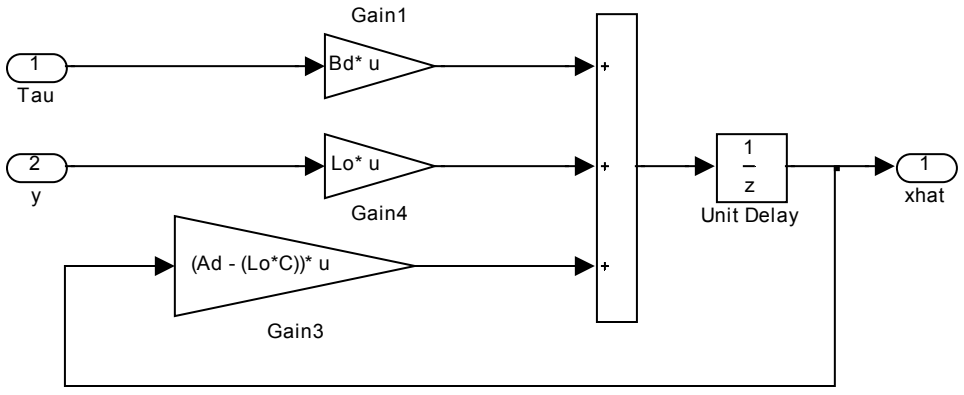

Figure 16: Observer subsystem

6.1.4 State Space Controller with Kalman Observer

A simulink model of the state space controller with observer is shown below in Figure 15. The output switch subsystem contains a switch that alternates between the Kalman observer state estimates and the measured states estimates depending on the pendulum angle value. This switch is necessary since the Kalman Observer will work in the linear range of the inverted pendulum system.

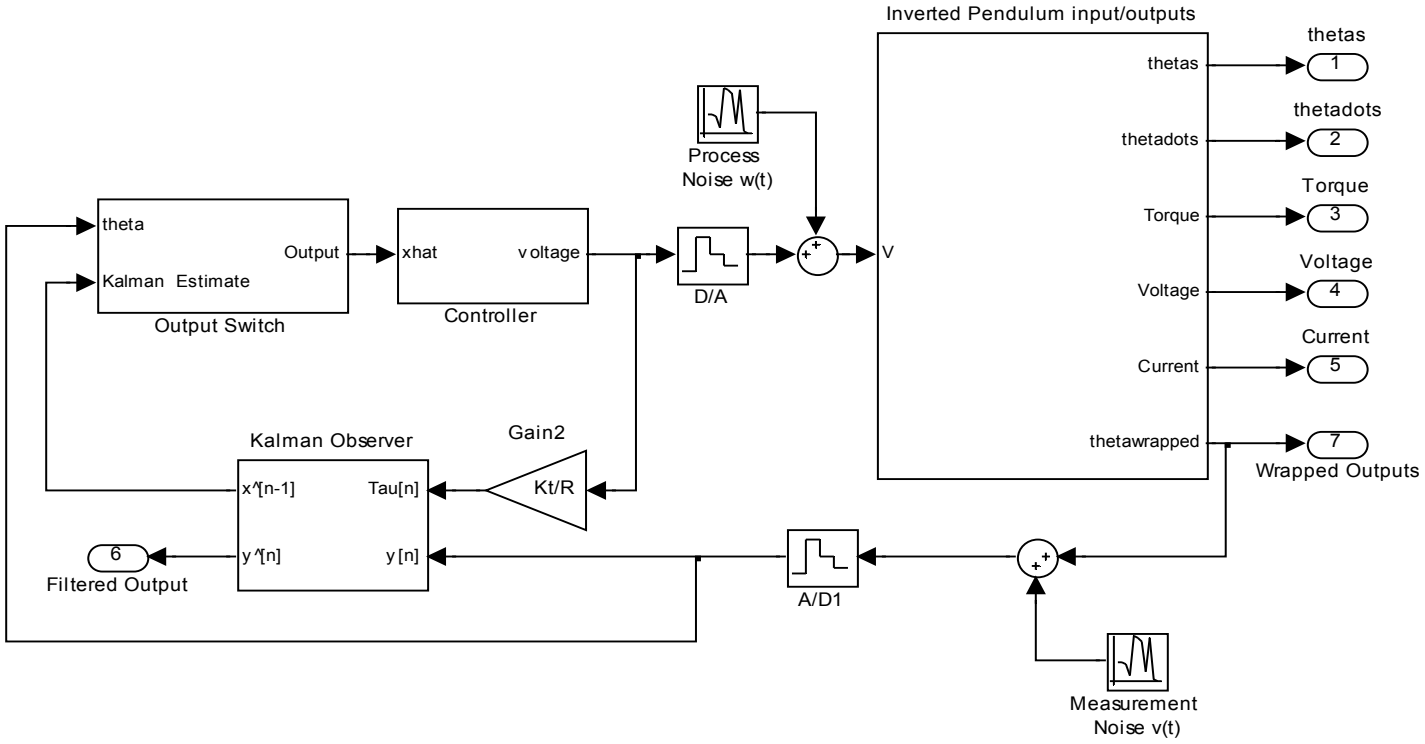

Figure 17: State Space Controller with Kalman Observer

A Simulink model of the output switch subsystem can be found below. The threshold for the switch is determined by the pendulum angle.

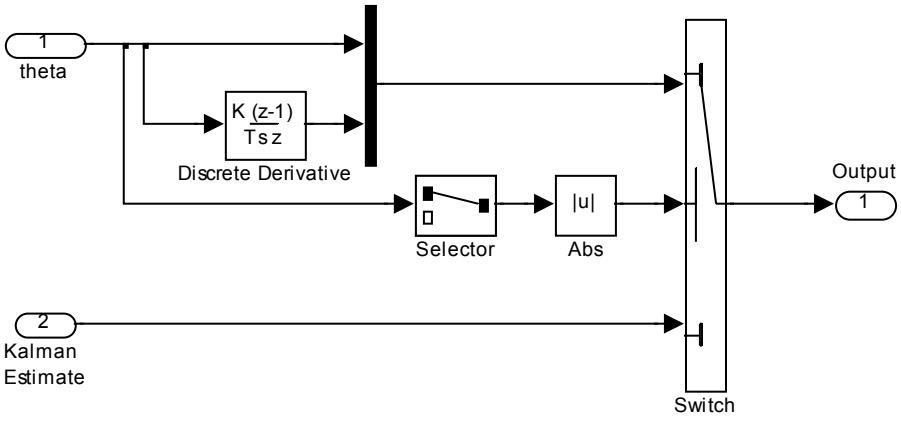

Figure 18: Output Switch Subsystem

A detailed model of the Kalman observer subsystem can be found below.

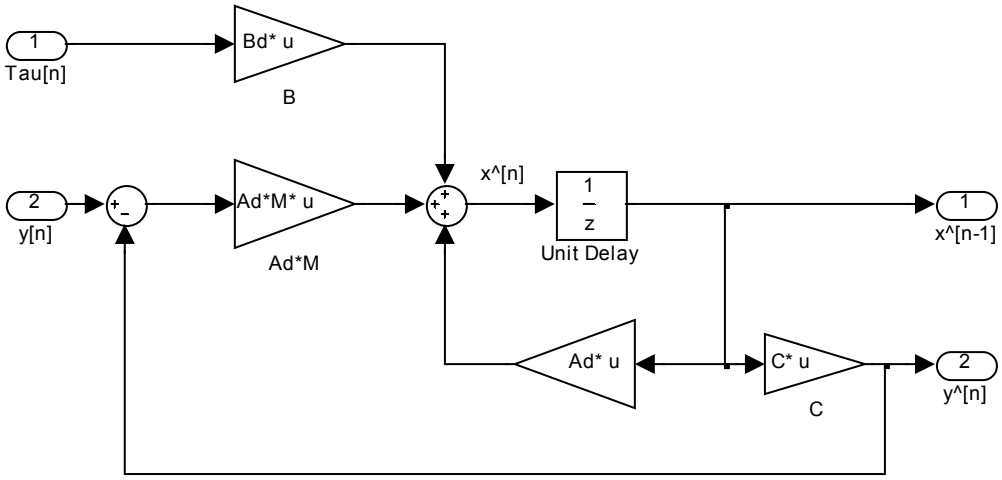

Figure 19: Kalman Observer Model

6.1.5 State Space Controller with Tracking Control

The Controller used for the tracking control of the inverted pendulum system is shown in Figure 20. The model can switch between pendulum tracking and link tracking by changing the value of swti.

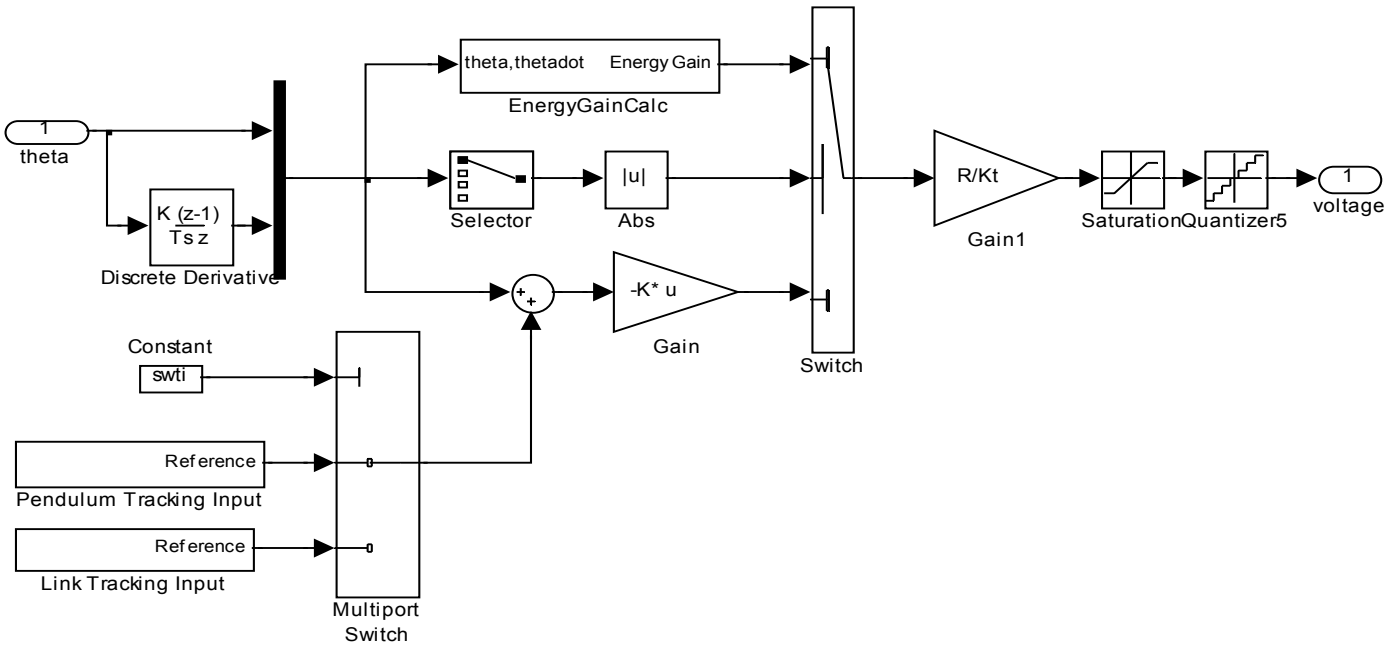

Figure 20: Tracking Controller

A detailed view of the Pendulum Tracking input subsystem is shown in Figure 21 below. The model can track four different reference signals. A sine, square and sawtooth wave or a step input.

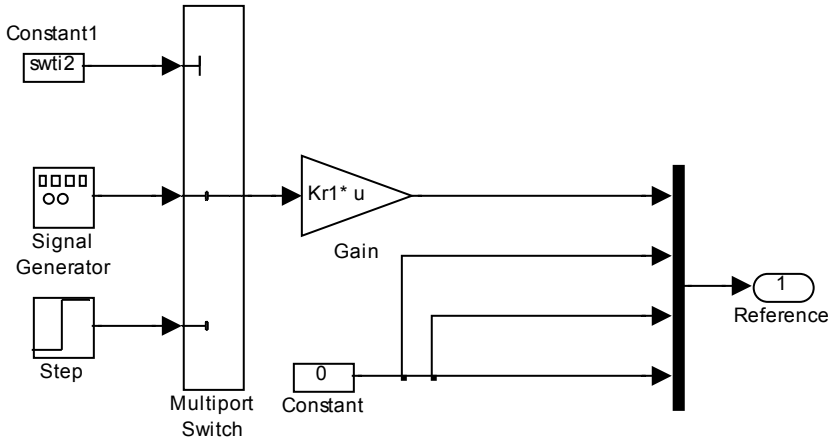

Figure 21: Pendulum Tracking Input subsystem

The Link Tracking input subsystem shown in Figure 22 below can also track 4 different reference signals.

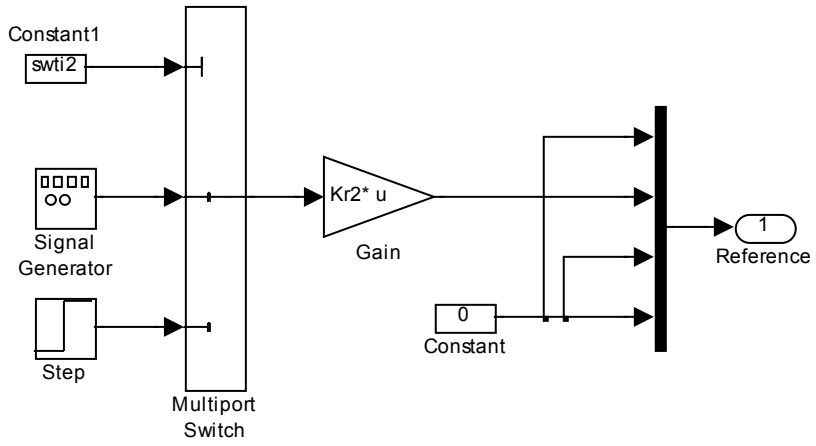

Figure 22: Link Tracking Input subsystem

6.2 Inverted Pendulum Real Time Control Models

6.2.1 State Space Controller

To control the inverted pendulum in real time the following Simulink model was used. The controller block is taken directly from the inverted pendulum simulation models. The pendulum and link angles are measured using an analog input. These voltages are scaled so that they go between $-\pi$ and π by the angle scaler subsystem. The microprocessor cannot output negative voltages. Consequently the motor drive subsystem is used to make the voltage output of the controller always positive and also to specify the sign of the voltage.

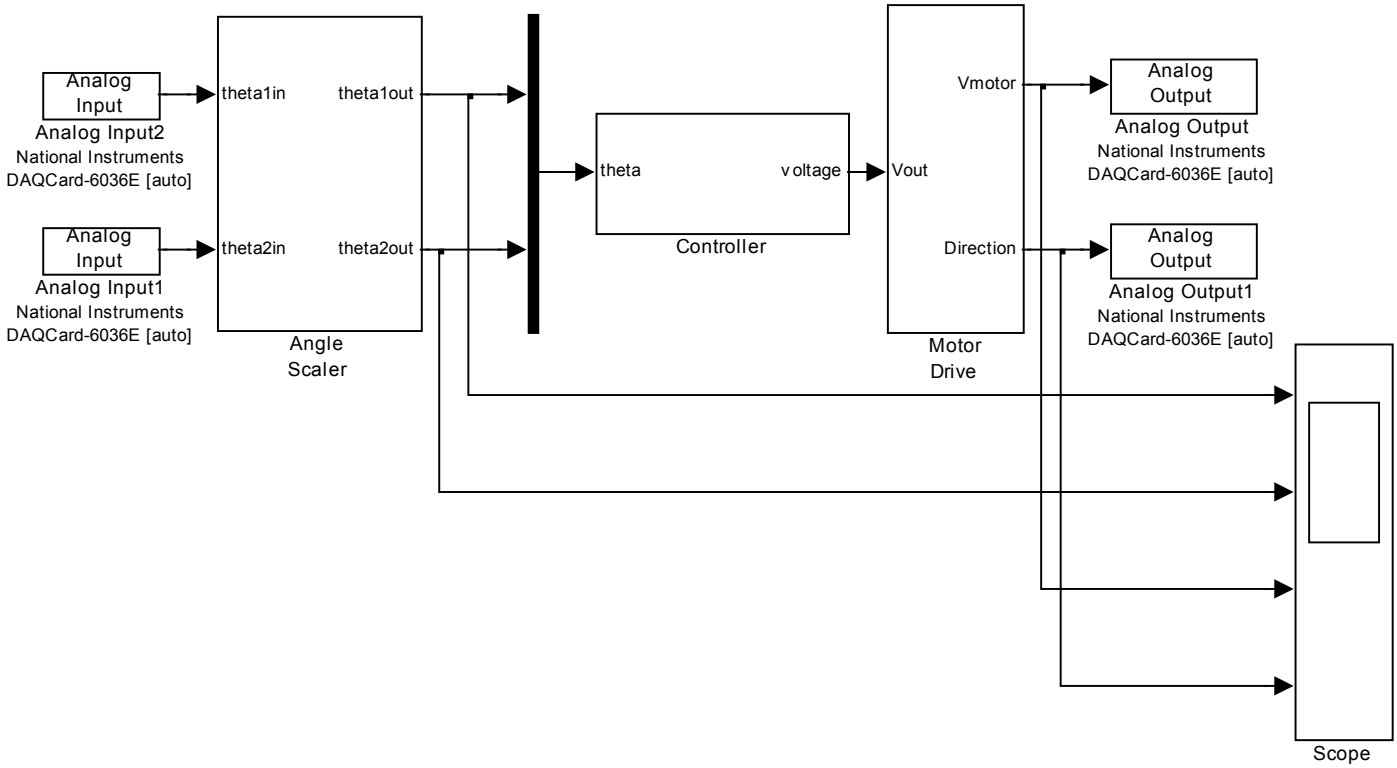

Figure 23: Inverted Pendulum Real Time State Space Controller model

The angle scalar subsystem is shown below in Figure 24. The first block is used to scale the pendulum angle. The second is used to scale the motor angle.

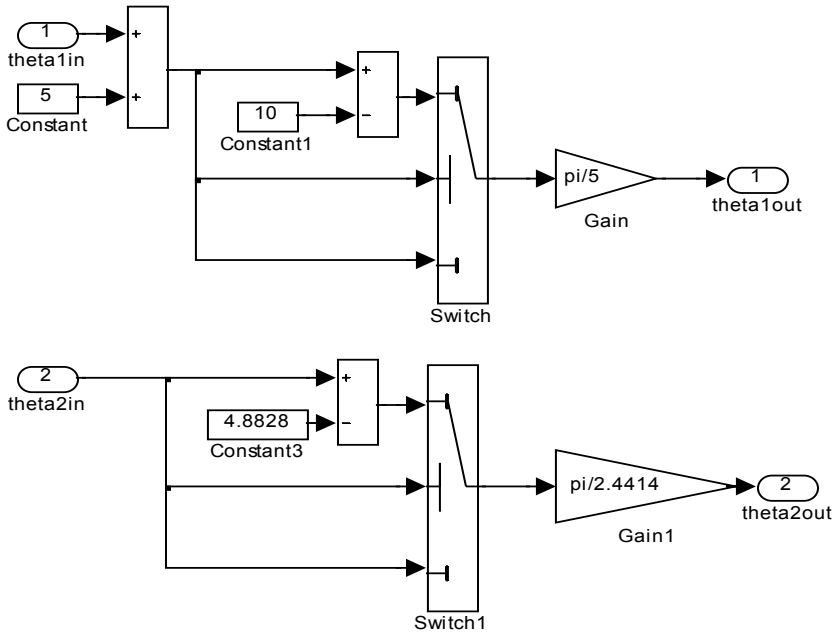

Figure 24: Angle Scaler subsystem

The motor drive subsystem pictured in Figure 25 below is used to take the absolute value of the voltage and also to specify the sign of the voltage.

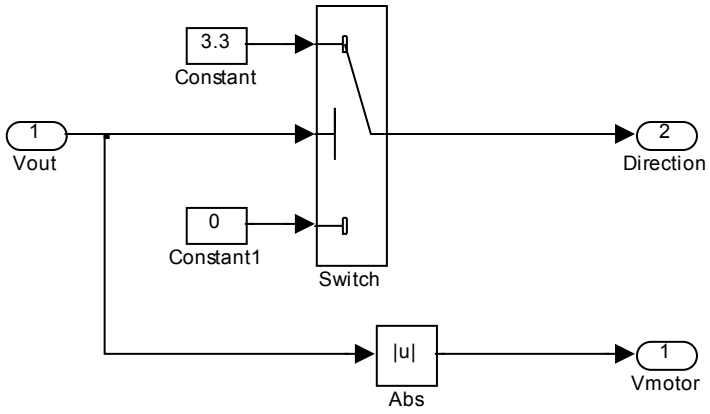

Figure 25: Motor Drive Subsystem