MECA 482 Lab: Inverted Pendulum

Merry Christmas, 2019

Paul Bartholow

Chase Cibart

Joseph Oliveri

Kyle Riemenschnitter

Introduction:

The purpose of this project was to develop and control a model of an inverted pendulum robot. An inverted pendulum consists of a mass being held above a pivot point with a lever arm. Without any external force interaction, the mass would be unstable and fall over. In order to keep the mass upright a control system needed to be implemented. An input was given to the motor, which is connected to the wheels on the cart, in order to balance the mass connected to the lever arm. An encoder was placed at the pivot point of the extended beam which provided a resulting output simultaneously as the wheels turned. From this, the angle was calculated, and a feedback control system was implemented in order to stabilize and control the position of the cart below. This process can be seen in Figure 1 and Figure 2 below. Our initial goal was to implement a control system into a V-REP model using MATLAB and SIMULINK. After modelling a simple inverted pendulum on V-REP and creating the control system in MATLAB, our group had trouble implementing the control system cross platform. As a result, the scope of our project changed to providing a simulated inverted pendulum using SIMULINK.

Modeling:

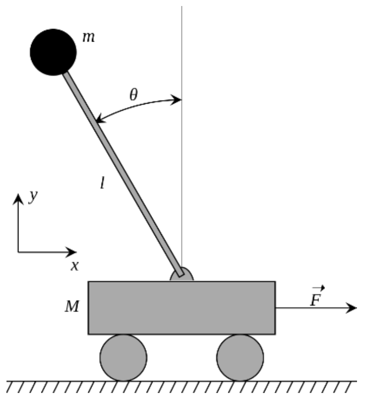


Figure 1 [1]

The initial equations were found based of the simple model of the system:

Cart:

$$i: F - Tsin(\theta) = m_c\ddot{x} \ (1)$$

Pendulum:

$$i: Tsin(\theta) = m_p a_{p_x} \ (2)$$

$$j: -Tcos(\theta) - m_p g = m_p a_{p_y} \ (3)$$

The following equation manipulation allowed us to find the state equations for our system. Utilizing the relative acceleration of the system we were able to solve for the state equations:

Relative Acceleration:

$$a_p = a_c + a_{\frac{p}{c}} = \ddot{x}\hat{\imath} + \left[L\ddot{\theta}\hat{e}_\theta - L\dot{\theta}^2\hat{e}_r\right]$$

$$= \ddot{x}\hat{\imath} + L\ddot{\theta}[-\cos(\theta)\,\hat{\imath} - \sin(\theta)\,\hat{\jmath}] - L\dot{\theta}^2[-\sin(\theta)\,\hat{\imath} - \cos(\theta)\,\hat{\jmath}]$$

$$Tsin(\theta) = m_p\ddot{x} - m_pL\ddot{\theta}\cos(\theta) + m_pL\dot{\theta}^2\sin(\theta) \;\; (4)$$

$$-Tcos(\theta) - m_pg = -m_pL\ddot{\theta}\sin(\theta) + m_pL\dot{\theta}^2\cos(\theta) \;\; (5)$$

$$\boldsymbol{-m_pgsin(\theta) = m_p\ddot{x}\cos(\theta) - m_pL\ddot{\theta}} \; (6)$$

$$\boldsymbol{F + m_pL\ddot{\theta}\cos(\theta) - m_pL\dot{\theta}^2\sin(\theta) = (m_p + m_c)\ddot{x}} \; (7)$$

Fixed Points:

$$\theta = 0, \pi$$

Lagrange's:

$$L = T - V$$

$$V = mgLcos(\theta)$$

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}mv_p^2 = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\left(v_x^2 + v_y^2\right)$$

$$= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(v_x^2 + v_y^2\right)$$

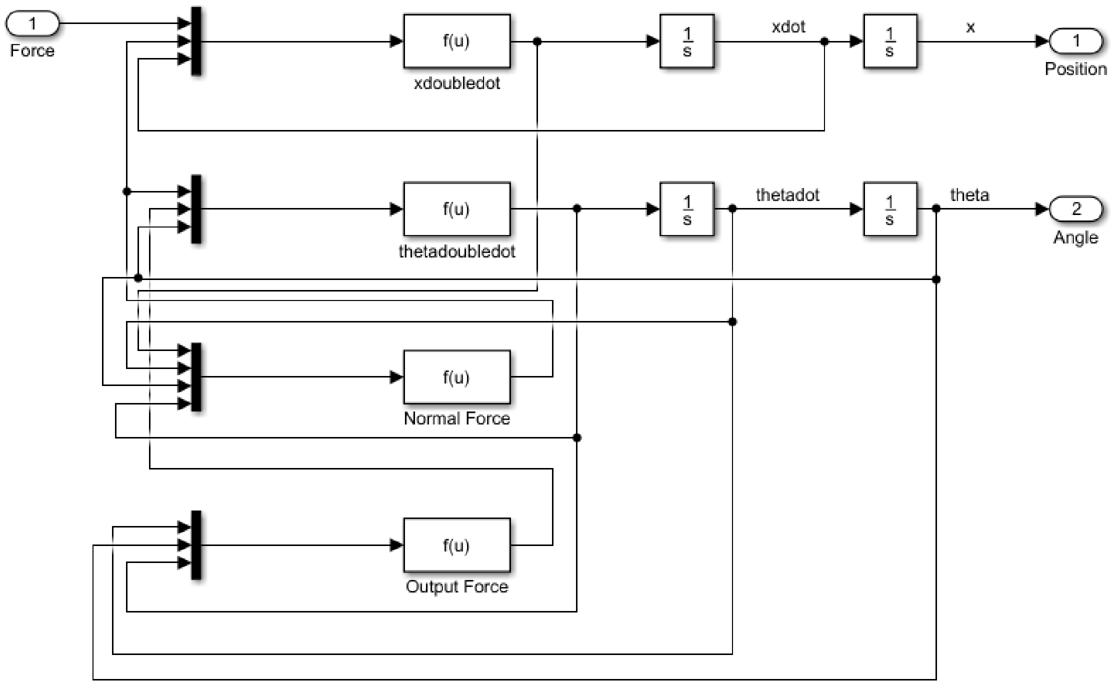
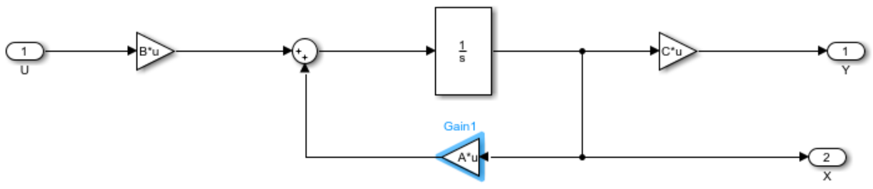


Figure 2 [1]

Model Simplification:

The model was simplified to a closed, negative-feedback loop and required less blocks during the simulation in SIMULINK.

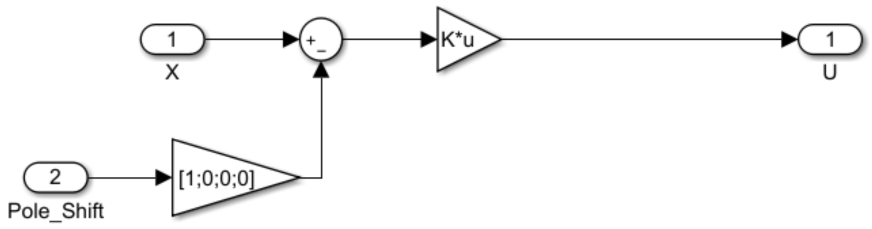


Sensor Calibration:

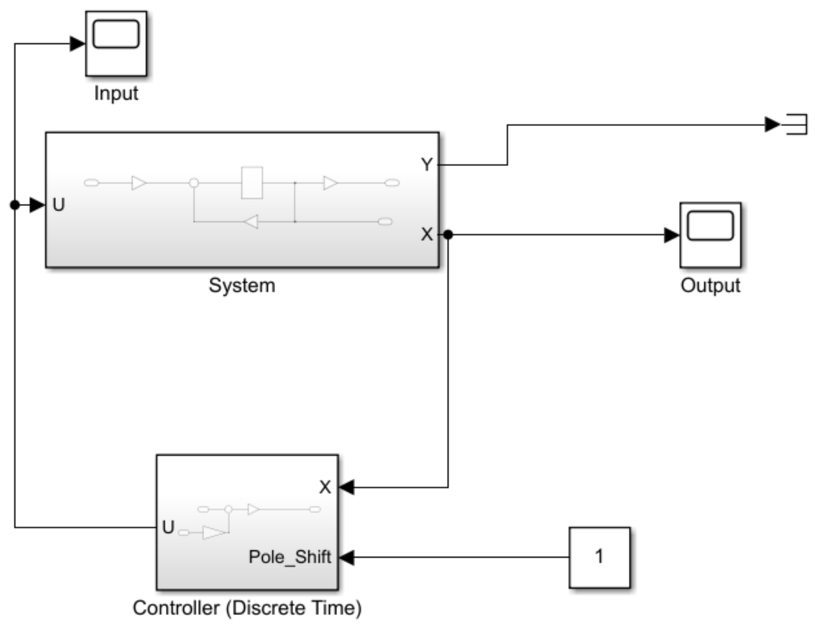
No sensor calibration was required for the implementation of this control system.

Controller Design and Simulations:

The controller was designed for operation in discrete time for the simulation process. A pole shift was required to place them to the left side of the imaginary axis. Using a rank comparison, it was also determined that there was an unobservable output even though both outputs were controllable. This unobservable output was omitted from the simulation.



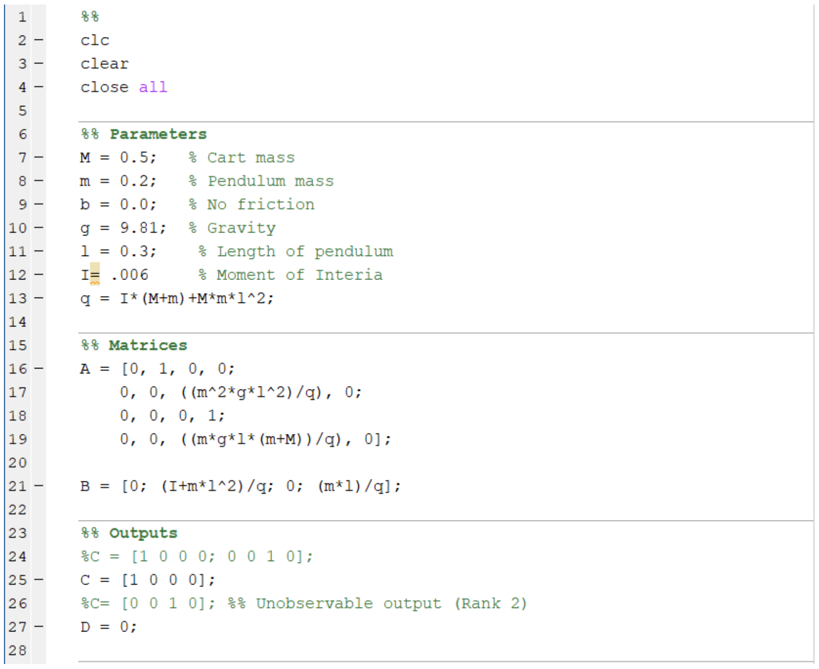
The entire model is shown below. Two scopes were added for viewing capabilities of the input and output responses.



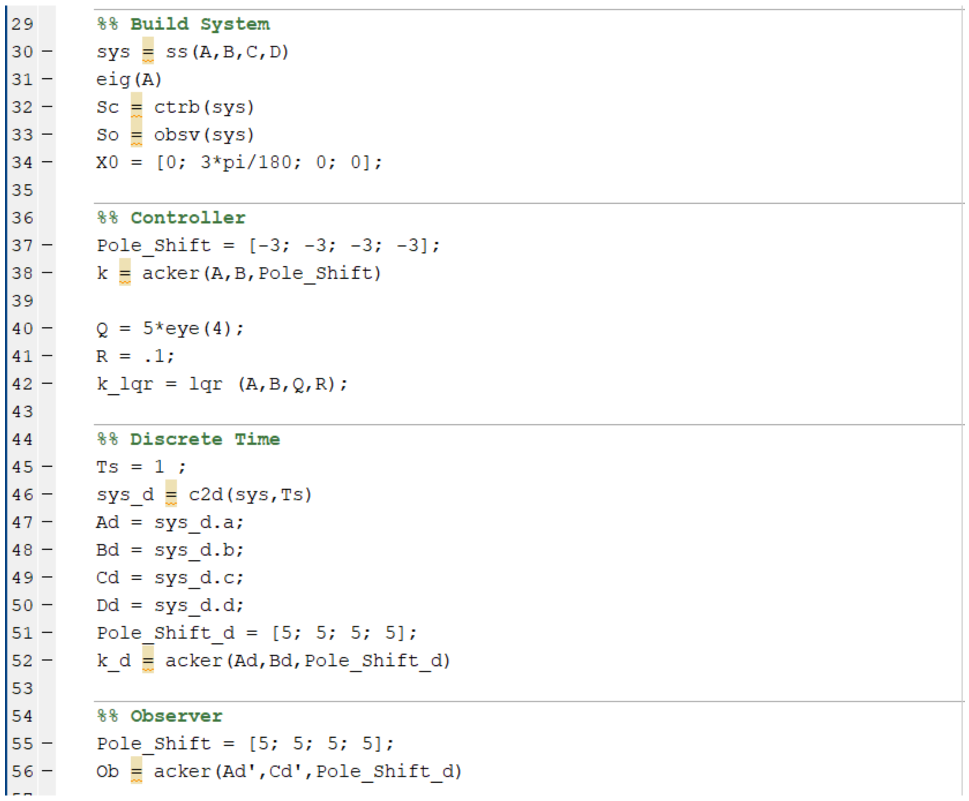
Unfortunately, the simulation process did not yield the desired results. While an input and output were achieved and displayed, they did not represent stability or accuracy.

Appendix A:

MATLAB Code

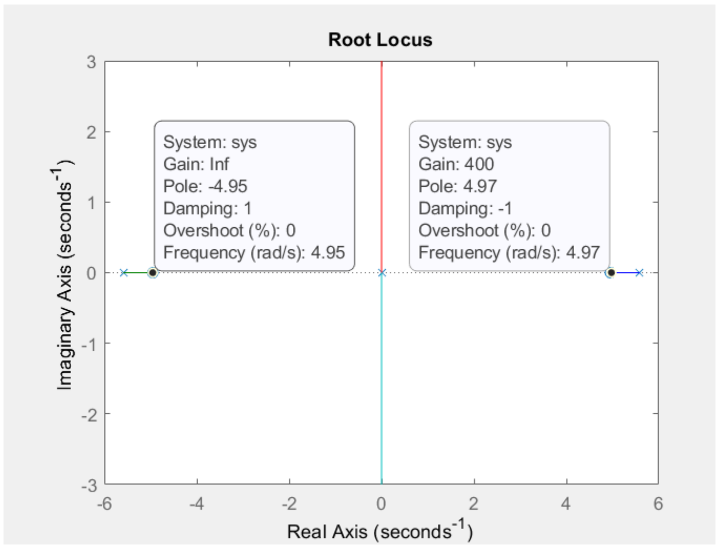


```
%%
clc
clear
close all

%% Parameters
M = 0.5;    % Cart mass
m = 0.2;    % Pendulum mass
b = 0.0;    % No friction
g = 9.81;  % Gravity
l = 0.3;     % Length of pendulum
I= .006      % Moment of Interia
q = I*(M+m)+M*m*l^2;

%% Matrices
A = [0, 1, 0, 0;
    0, 0, ((m^2*g*l^2)/q), 0;
    0, 0, 0, 1;
    0, 0, ((m*g*l*(m+M))/q), 0];

B = [0; (I+m*l^2)/q; 0; (m*l)/q];

%% Outputs
%C = [1 0 0 0; 0 0 1 0];
C = [1 0 0 0];
%C= [0 0 1 0]; %% Unobservable output (Rank 2)
D = 0;

```

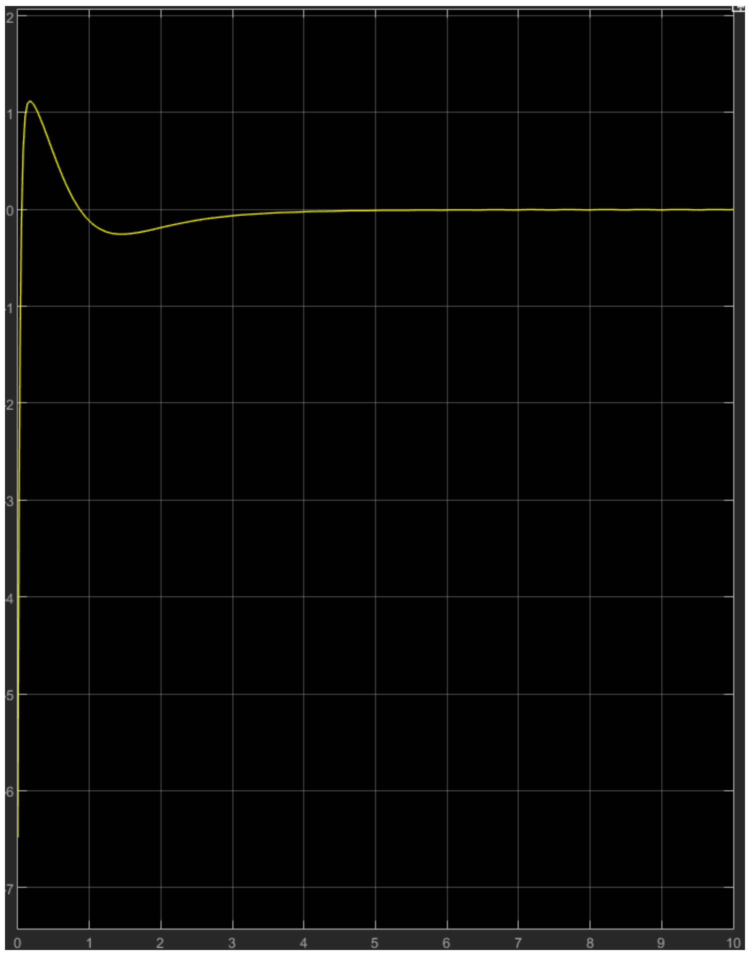


```
%% Build System
sys = ss(A,B,C,D)
eig(A)
Sc = ctrb(sys)
So = obsv(sys)
X0 = [0; 3*pi/180; 0; 0];

%% Controller
Pole_Shift = [-3; -3; -3; -3];
k = acker(A,B,Pole_Shift)

Q = 5*eye(4);
R = .1;
k_lqr = lqr (A,B,Q,R);

%% Discrete Time
Ts = 1 ;
sys_d = c2d(sys,Ts)
Ad = sys_d.a;
Bd = sys_d.b;
Cd = sys_d.c;
Dd = sys_d.d;
Pole_Shift_d = [5; 5; 5; 5];
k_d = acker(Ad,Bd,Pole_Shift_d)

%% Observer
Pole_Shift = [5; 5; 5; 5];
Ob = acker(Ad',Cd',Pole_Shift_d)
```

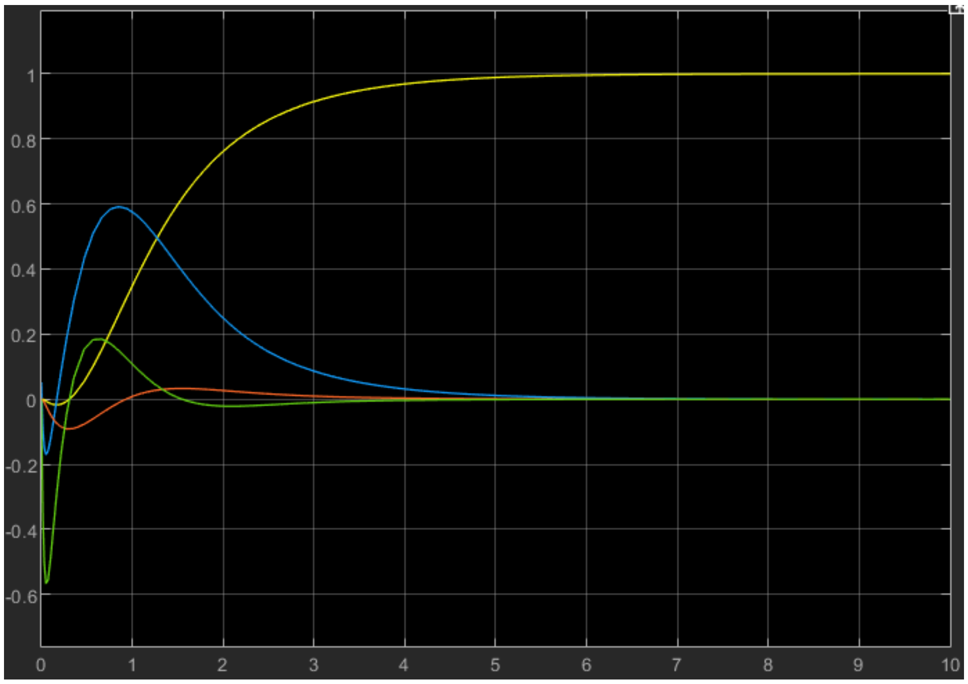
Root Locus



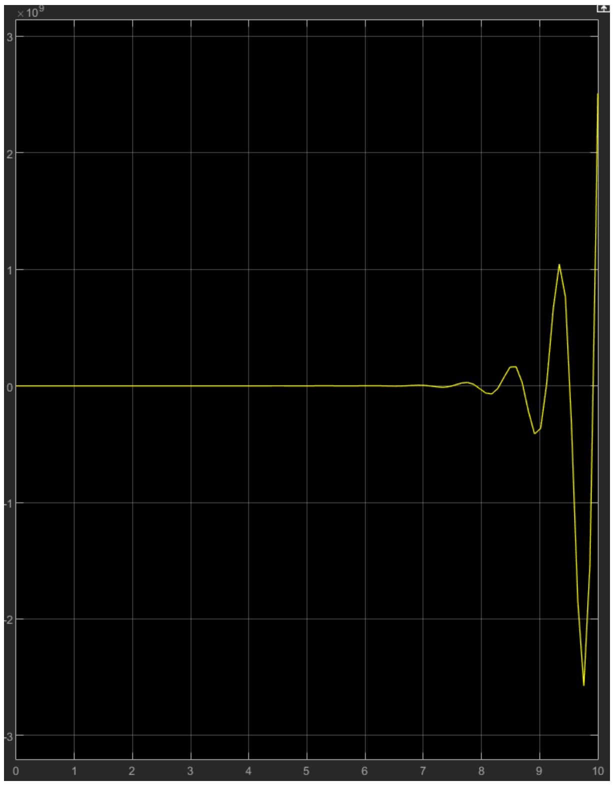
Scope of LQR Input Response



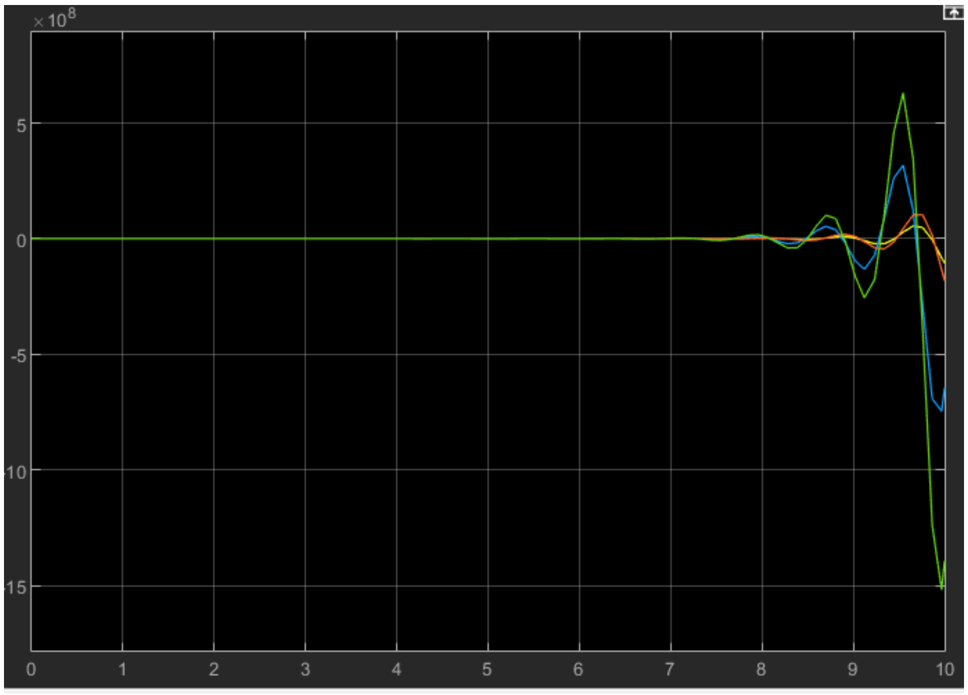
Scope of LQR Output Response



Scope of Discrete Time Input Response



<u>Scope of Discrete Time Output Response</u>



References:

Control Tutorials for MATLAB and Simulink -,

http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum§ion.