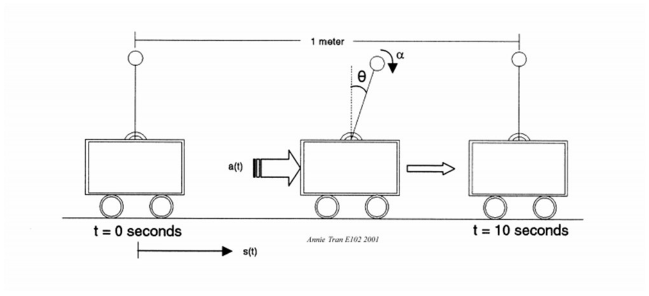
Marco Conati
April 30, 2020

Engineering 102 Final Report

I. Introduction

This report details the steps taken to stabilize an inverted pendulum system on a cart. The problem setup and system dynamics are shown in Figure 1. An inverted pendulum of length 0.5 meters is attached to a cart that must be moved to a location 1 meter away within ten seconds. During this process, cart position and angular displacement are measured and the available control input to the system is the acceleration of the cart. Additionally, there is a disturbance causing an angular acceleration of 0.5 rad/s^2 on the pendulum, no overshoot is allowed, and the maximum control input is $|a(t)| < 0.5$ m/s^2.



$$L\frac{d^2\theta(t)}{dt^2} - g\sin\theta(t) = -a(t)\cos\theta(t) + L\alpha(t)$$

$$\frac{d^2 s(t)}{dt^2} = a(t)$$

Figure 1. Initial problem and System dynamics

The objective was to develop a controller for this system using a Linearized Plant, and then prove the abilities of the controller using simulation with the nonlinear plant.

II. State Space Design of a Linear Controller for a Linearized Plant

To begin the design of the Linear controller, the system must first be linearized. Since the controller should keep the pendulum near vertical to ensure stability, the system was linearized around this fixed point. To do so, the small angle approximation was used ($\sin(\theta) \approx \theta$; $\cos(\theta) \approx 1$). For simplicity, I will refer to θ(t), s(t), and a(t) as θ, s, and a respectively.

$$\ddot{\theta} = \frac{-a + L\alpha + g\theta}{L}$$

$$\ddot{s} = a$$

Now that we had the linearized dynamics for our system, we could put them in state space form. In our case, the state $\overline{x} = \begin{bmatrix} \theta \\ \dot{\theta} \\ s \\ \dot{s} \end{bmatrix}$, output vector $y = \begin{bmatrix} \theta \\ s \end{bmatrix}$, disturbance w = α, and input u = a. Putting our system dynamics in matrix form, we get the following state space representation:

$$\dot{x} = A\overline{x} + Bu$$

$$y = C\overline{x} + Du$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{g}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ -\frac{1}{L} \\ 0 \\ 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

To establish that the system is possible to control and observe, we can calculate our controllability and observability matrices. These matrices dictate whether achieving stability is possible. If the controllability matrix is full rank, the system can reach any state based off the inputs. If the observability matrix is full rank, the current state can be approximated from the outputs. Clearly, both are needed to control a system to reach a desired state:

$$Co = \left[B\ AB\ A^2B \ldots A^{n\text{-}1}B\right] \qquad Ob = transpose\left(\left[C\ CA\ CA^2 \ldots CA^{n\text{-}1}\right]\right)$$

$$Co = \begin{bmatrix} 0 & \text{-}2 & 0 & \text{-}4g \\ \text{-}2 & 0 & \text{-}4g & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad Ob = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 2g & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 2g & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Both matrices are full rank, so the system is controllable and observable. The system will be controlled using integral control in order to eliminate any steady state error. To calculate our integral control gain, we can augment the state vector to contain our gain value. The state is updated by considering the behavior of the plant and controller as one system. The overall integral action system is given by:

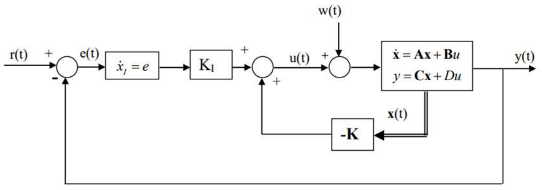

w(t)
r(t)
+
-
e(t)
$\dot{x}_I = e$
K_I
+
+
u(t)
+
$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$
$y = \mathbf{C}\mathbf{x} + Du$
y(t)
x(t)
-K

Figure 2. Block diagram of the system with integral action.

In order to augment the state, the feedback behavior of the system is considered. The next state is given by the error in the system: $\dot{x\imath} = e = r - y = r - C\overline{x}$ – Du. Augmenting our state matrix to include xi as a state allows us to add this equation to our state space matrices, as shown below:

$$\overline{x} = \begin{bmatrix} xi \\ \theta \\ \dot{\theta} \\ s \\ \dot{s} \end{bmatrix} \qquad A = \begin{bmatrix} 0 & 0 & 0 & \text{-}1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{g}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{L} \\ 0 \\ 1 \end{bmatrix} \qquad C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

To control this augmented plant, both a controller and observer must be designed. An observer is necessary since only two variables of state are present at the output, meaning the rest must be predicted. However, before designing the controller and observer, the augmented system must be shown to be controllable as well:

$$Co = \left[B\ AB\ A^2B \ldots A^{n\text{-}1}B\right] \qquad Co = \begin{bmatrix} 0 & 0 & \text{-}1 & 0 & 0 \\ 0 & \text{-}2 & 0 & \text{-}4g & 0 \\ \text{-}2 & 0 & \text{-}4g & 0 & -8g * g \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Once again, the controllability matrix is full rank, so it can be controlled. The controller was designed using the pole placement method. The initial controller poles were designed to make the system roughly second order. This was done by setting the two dominant poles to be less negative (-0.11 and -0.12) and the two remaining poles to be more negative (-1 and -1.01). The dominant poles will decay much less quickly, in turn determining the behavior of the system and making it roughly second order. The additional pole (-0.1) is included due to our augmented state and determines Ki. It is placed around the dominant poles of our system so that it does not slow down the system or cause unexpected oscillations. The poles were set to similar, but not repeating, values since the Place command in Matlab can't work with repeated poles. A parameter k was introduced that can speed up or slow down the response. This parameter was initially set to 6 but meant to be tuned during simulation.

$$Pa = [-0.1 - 0.11 - 0.12 - 1 - 1.01] * k$$

The observer poles were designed to be faster than the controller poles by some integer factor. This parameter, h, was initially set to 5, but meant to be tuned in simulation as well.

$$Pobs = Pa[2{:}end] * h$$

Finally, once we had our pole locations, the Place command was used to create our observer and controller gains as shown below. The full code is included in Appendix A.6.

```
%%CONTROLLER POLES
%Place poles for the linear plant
Ka = place(Aa,Ba,Pa);
%separate the Ki value and K gains
Ki = -1*Ka(1);
K = Ka(2:end);
```

```
%%OBSERVER POLES
%calculate the observer poles(same as Pa but scaled by
h)
Pobs = Pa(2:end)*h;
%Place the observer poles and calculate L
L = place(A', C', Pobs);
L = L';
```

Now, all gains and matrices are known, so the system can be simulated.

III. Simulation of Control of Nonlinear Plant with Linear Controller

To simulate the nonlinear system using the linear controller, a Simulink model was built. This model used the controller and observer designed in II with a nonlinear plant dictated by the governing equations of the pendulum system. The overall model, along with all subsystems are shown in Appendix A.1-A.5.

The model was initially run with the guessed controller and observer pole locations. However, the system was clearly unstable and improperly controlling the system. Control effort was capped for most of the run, angle diverged quickly, and position was growing without bound. This is shown in Figure 3.

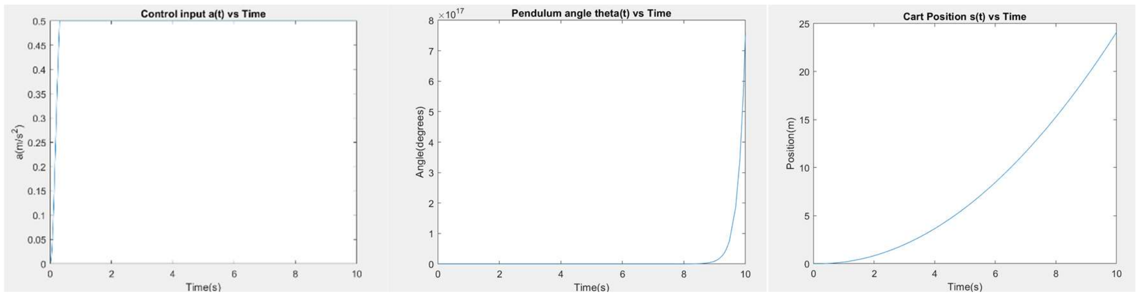


Figure 3. Unstable system with guessed pole locations

In order to meet all the required specifications, the simulation model was iteratively run with different parameters controlling the controller and observer pole locations. The parameter k was used to multiply controller pole guesses to get our controller pole locations. The parameter h was used to multiply controller pole locations to get observer pole locations as described at the end of section II. In both instances, this process made the poles more negative. This had the positive effect of increasing response speed and the negative effect of increasing overshoot.

The iterative process was performed as follows. The simulation was run with values of h at integer increments from 5 to 10. As a nested for-loop, these different h values were then run with values of k ranging from 6 to 12 and a resolution of 0.01. From these 3000 runs of the simulation, all combinations where there was no overshoot were saved as possible solutions. From hundreds of solutions, the combination (h = 9, k = 6.14) was selected since this combination had the largest final position value without overshooting 1. This run of the simulation kept control effort, position, and angle within specifications as shown in Figure 4:

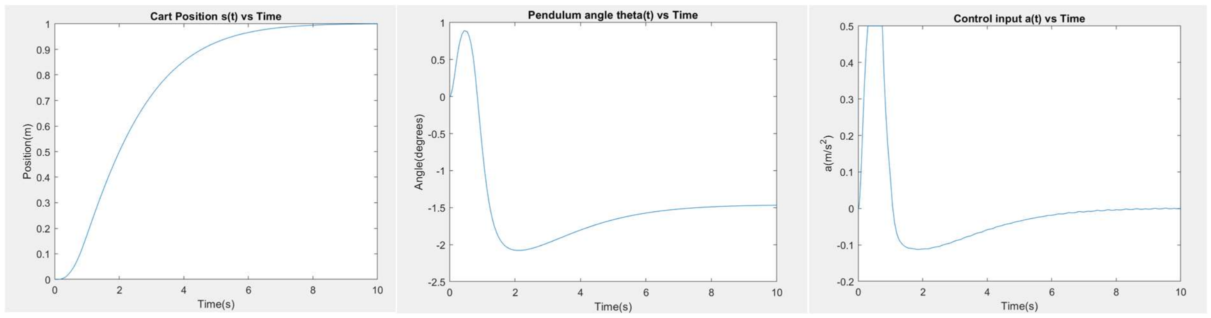


Figure 4. Controlled inverted pendulum outputs

IV. Discussion and Conclusions

Overall, the process of designing a linear controller for the inverted pendulum system went smoothly. The result was within specifications, and the output graphs show desired behaviors. Specifically, the control effort $|a(t)| < 0.5$ m/s^2 and there was no overshoot of the ending position. Additionally, the resting angle of the pendulum makes sense, and is stable at -1.466 degrees. The resting angle of the pendulum must be slightly negative to offset the disturbance α. Additionally, plugging into our system dynamics equations shows that with the conditions at t=10s, there is no angular acceleration of the pendulum:

$$\ddot{\theta} = \frac{-a\cos(\theta) + L\alpha + g\sin(\theta)}{L} \qquad \ddot{\theta} = \frac{-(0) + (0.5)^2 + 9.8\sin(-1.466)}{L} \approx 0$$

To test the limitations of this result, α was incremented to find the point where the system went unstable. With this set of parameters, the system went unstable at $\alpha = 0.54$ rad/s^2. This result is slightly disappointing, as it leaves an exceedingly small margin for any variance in the disturbance input. However, this is expected with how the system was designed. When I designed the controller to reach one meter without any overshoot, I did not consider its resistance to larger input disturbances. To test how the resilience could be improved with slightly relaxed position requirements, I iterated by increasing values of α and searching for poles that could stabilize the system at each α. The table below shows the poles used earlier in the report, along with two other noteworthy points that I found:

#	Maximum alpha	Controller poles(Pa)	Ki	Observer Poles(Pobs)	Position at 10s(m)
1	0.54	[-0.68 -0.74 -6.14 -6.20]	-0.614	9*Pa	1
2	0.79	[-1.32 -1.44 -12 -12.12]	-1.2	8*Pa	0.9981
3	0.81	[-1.32 - 1.44 -11.99 -12.11]	-1.199	9*Pa	0.9813

This process was by no means exhaustive, but it reflects the importance of situational control. Depending on the objective, a different controller setup may be optimal, even if the plant remains the same. As shown above, allowing for a slight deviation from the position specification in point 2 vastly increased the tolerance to disturbance. Taking it even farther with point 3, I showed that the system can be very resistant to disturbance if that becomes the design objective. This principle highlights part of the importance of a Systems Engineers' role. It is clearly important to both design a solution that closely meets specifications, but also consider any adverse side effects of the design process to ensure that the design functions as intended.

V. Appendix

References:

Bright, Anthony. E102 Modern Control Engineering: Class Notes.

Tsai, TJ. E102 Lecture notes.

Simulink Diagrams:

Figure A.1. Overall System Diagram:

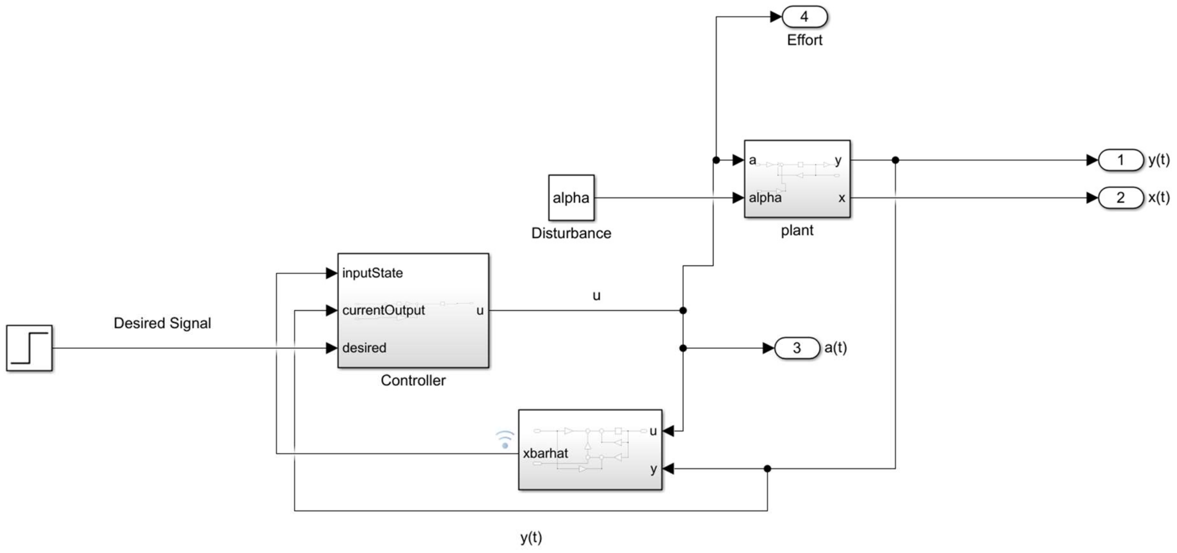
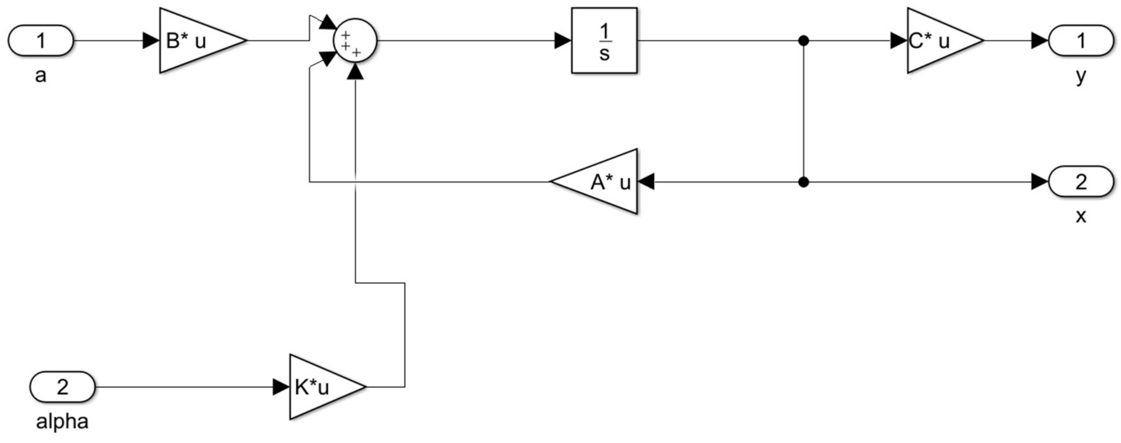


Figure A.2. Linear Plant:



Gain block to change alpha to dimension [0;1;0;0]

Figure A.3. Nonlinear Plant:

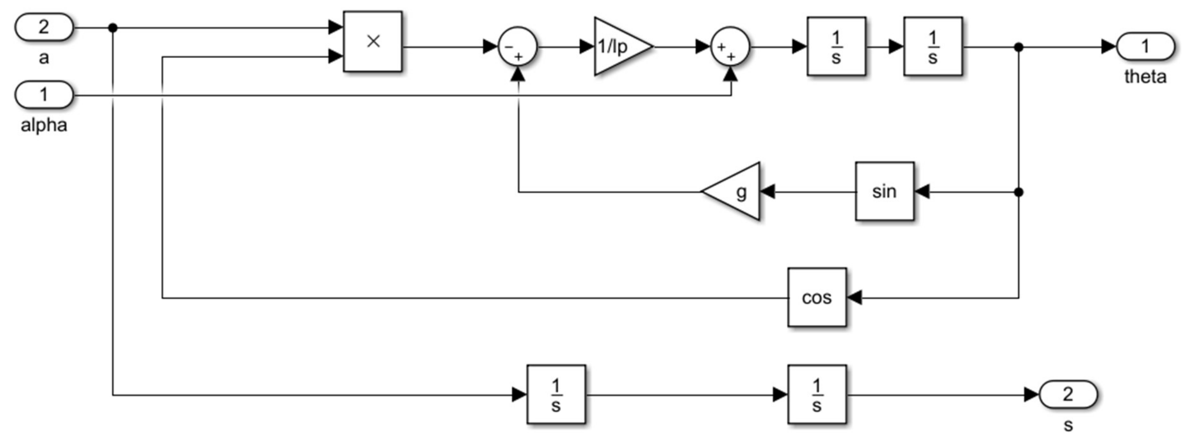


Figure A.4. Observer:

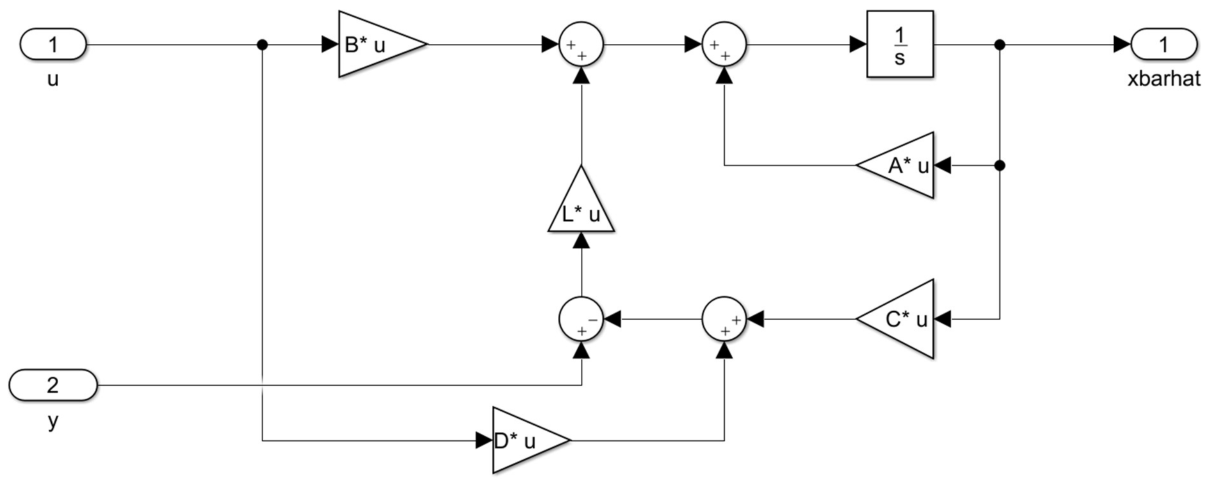
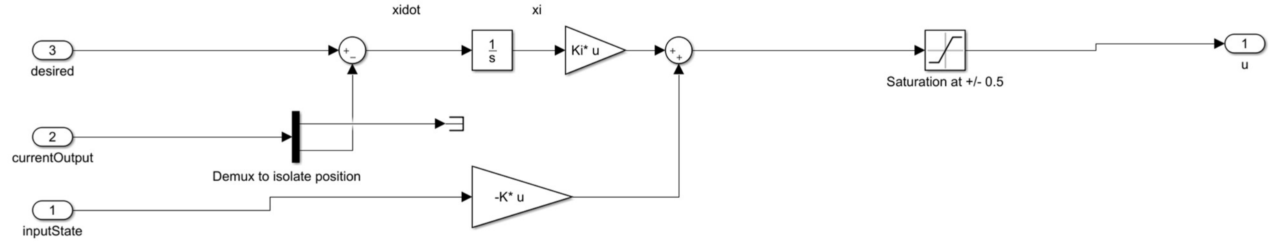


Figure A.5. Controller:



A.6. Matlab Code for the overall system:

```
%%Declare model parameters
%--lp is length of pendulum (m)
%--g is acceleration due to gravity (rad/s^2)
%--alpha is the disturbance in rad/s^2
lp = 0.5;
g = 9.81;
alpha = 0.5;
%% Create the state space representation of the system
%Create the A,B,C,D matricies for the cart-pendulum system
A = [0 1 0 0; g/lp 0 0 0; 0 0 0 1; 0 0 0 0];
B = [0; -1/lp; 0; 1];
C = [1 0 0 0; 0 0 1 0];
D = [0;0];

%Check controllability and observability
sys = ss(A,B,C,D);
controllable = rank(ctrb(sys));
observable = rank(obsv(sys));
if observable == 4 && controllable ==4
    disp('The initial system is controllable and observable!')
end
%% Augment the state. Create Aa,Ba,Ca,Da for the system with integral control
%Create a matrix Cs representing the position related component of the C
%matrix
Cs = [0 0 1 0];
Aa = [0 -Cs; [0;0;0;0] A];
Ba = [0; B];
Ca = [[0;0] C];
Da = D;
%Check controllability and observability
sys = ss(Aa,Ba,Ca,Da);
controllable = rank(ctrb(sys));
observable = rank(obsv(sys));
if observable == 4 && controllable ==4
    disp('The augmented system is controllable and observable!')
end

%% Place poles for the system
% I chose two poles closer to the jw axis and two farther
% away to make the system behave more like a second
% order system. The first 'pole' is
% Ki, which should be placed around the dominant
% pole of the system
Pa =  [-0.1 -0.11 -0.12 -1 -1.01];
%I created two constants h and k. k is multiplied by my poles to change
%their speed and create Pa. h is multiplied by Pa to create the observer
%poles. To calculate potential values, I iterated over values of k between
%6 and 12(with a resolution of 0.01) and values of h between 5 and 10(with
%a resolution of 1). I found solutions for the Nonlinear and linear system
%that did not overshoot the endgoal and did not have a maximum control
%effort over 0.5. I chose the following combinations of h and k from my
%solutions:

%effort    overshoot    h       k
%0.5000    0%        9.0000   6.14  %lowest overshoot for nonlinear plant

%set h and k
k = 6.14;
h = 9;
Pa =  [-0.1 -0.11 -0.12 -1 -1.01]*k;

%Place poles for the linear plant
Ka = place(Aa,Ba,Pa);
%separate the Ki value and K gains
Ki = -1*Ka(1);
K = Ka(2:end);
```

```
%calculate the observer poles(same as Pa but scaled by h)
Pobs = Pa(2:end)*h;
%Place the observer poles and calculate L
L = place(A', C', Pobs);
L = L';

%% Check the system requirements.
sim('pendulum.slx', 10) %Simulates the system with a  10 second window
%get the control effort a
effort = get(yout,3);
%get the output
output = get(yout, 1);
%extract values from each
output = output.Values.Data;
effort = effort.Values.Data;
%Plot the position, angle, and control effort
figure(1)
plot(tout,output(:,2))
title('Cart Position s(t) vs Time')
xlabel('Time(s)')
ylabel('Position(m)')
figure(2)
plot(tout,output(:,1).*180/pi)
title('Pendulum angle theta(t) vs Time')
xlabel('Time(s)')
ylabel('Angle(degrees)')
figure(3)
plot(tout, effort)
title('Control input a(t) vs Time')
xlabel('Time(s)')
ylabel('a(m/s^2)')
```