E102 Inverted Pendulum

Claudia Nanez and Nandini Garg

April 18, 2024

1 Introduction

We are given a cart with an attached inverted pendulum that must move from its initial rest position at time is zero seconds to its final rest position one meter away. Sensors measure the cart position at $s(t)$ and the pendulum angular displacement $\theta(t)$. An model of the system is given below in Figure 1.

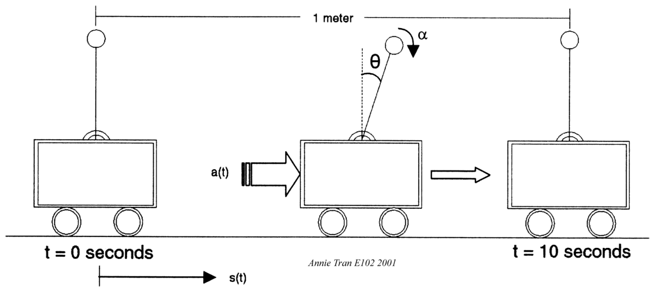


Figure 1: Cart with Attached Inverted Pendulum System Model

The pendulum is subject to an angular acceleration disturbance $\alpha(t) = 0.5rad/sec^2$. A control system will carry out the movement over ten seconds. The control input, acceleration, is limited to $|a(t)| < 0.5m/sec^2$. The cart will not overshoot the final position. The equations for the cart-pendulum system are given below. Let pendulum length L = 0.5m.

$$L\frac{d^2\theta(t)}{dt^2} - gsin\theta(t) = -a(t)cos\theta(t) + L\alpha(t) \tag{1}$$

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

2 State Space Design of a Linear Controller for a Linearized Plant

2.1 Linearized State Space Equations

We will formulate the linearized state space equations of the form $\dot{x} = Ax + Bu$ and $y = Cx + Du$. First, we put equations (1) and (2) into canonical form and apply the small angle approximation $sin\theta(t) = \theta(t)$ and $cos\theta(t) = 1$. The results are in equations (3) and (4) below.

$$\frac{d^2\theta(t)}{dt^2} - \frac{g\theta(t)}{L} = -\frac{a(t)}{L} + \alpha(t) \tag{3}$$

$$\frac{d^2s(t)}{dt^2} = a(t) \tag{4}$$

Then, use state vector $x = \begin{bmatrix}\theta(t)\\ \dot{\theta}(t)\\ s(t)\\ \dot{s}(t)\end{bmatrix}$ and let $u = a(t), w = \alpha(t), y = \begin{bmatrix}\theta(t)\\ s(t)\end{bmatrix}$. We find the matrices $A, B, C,$ and D as follows.

$$A = \begin{bmatrix}0 & 1 & 0 & 0\\ \frac{g}{L} & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0\end{bmatrix} B = \begin{bmatrix}0\\ \frac{-1}{L}\\ 0\\ 1\end{bmatrix} C = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix} D = \begin{bmatrix}0\\ 0\end{bmatrix} \tag{5}$$

2.2 Stability, Controllability, and Observability

The first step in modeling a state space plant is to determine if it is stable, controllable, and observable. A linearized state space system is stable if all of the eigenvalues of matrix A lie in the left hand plane or are complex conjugate pairs with real parts that lie in the left hand plane. We find the poles of matrix A to be 4.4924, -4.4924, 0, 0. These poles do not lie in the left hand plane, so the system is currently unstable. A system is controllable if the found controllability matrix $Mc = \begin{bmatrix}B & AB & ... & A^{n-1}B\end{bmatrix}$ has a rank that is equivalent to the number of columns in matrix A and B. The model was evaluated in MATLAB and the rank of matrix Mc was found to be 4, which equals n. Thus, the system is controllable. Similarly, we find that the rank of observability matrix $Mo = \begin{bmatrix}C & CA & ... & CA^{n-1}\end{bmatrix}^{-1}$ equivalent to n = 4 and the system is observable.

2.3 State Feedback Control System

In order for us to be able to design a state feedback control system with integral action and an observer for the linearized plant, the first step is to recognize the realities of the system and also the constraints of the problem statement. The inverted pendulum problem is inherently unstable. This reality of the problem is further supplemented by section 2.2- Stability, Controllability, and Observability. The biggest constraint of the problem at hand is that the system must have a settling time that is under 10 seconds. This constraint is essentially what the sets the tone for the way in which we solve the problem. Since our settling time is supposed to be under 10 seconds, we initially chose an arbitrary T_s value as 8 seconds. This was later changed to 6 seconds based on values we believed would work best with our guess and check model. This new settling time value was still in line with our constraints and so it worked well for us. We also operated under the assumption that the system is an underdamped system which meant that our ζ values would need to be between 0 and 1. We ended up picking our ζ value as 0.95. These values was then used to compute the ω_n value.There were two ways in which we could compute the ω_n, either by using the settling time formula

$$T_s = \frac{4}{\zeta\omega_n} \tag{6}$$

or by using the natural frequency formula

$$\omega_n = \frac{-\log\left(T_p \times \sqrt{1-\zeta^2}\right)}{\zeta \times T_s} \tag{7}$$

We specifically ended up using equation 7, because we found that it worked better with our iterative guess and check model. After figuring out the ζ and ω_n values we used the 2nd order system's canonical form

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \tag{8}$$

to figure out the poles of our system. The poles that we attained for this system were

$$pole_1 = -0.961520272746210 + 0.316036430458925i$$
$$pole_2 = -0.961520272746210 - 0.316036430458925i$$

Based on these poles that we attained, we used the iterative guess and check model to try and find the best value of orders of magnitude of the original pole1. Using this pole placement method to try and attain the desired poles of the system we were then able transpose the original A, B, C and D matrices into desired formats and attain the integral and observer gains using the place function in Matlab. These gains were situated in variables that were then linked to our Simulink models and run to attain the subsequent graphs.

3 Simulation of Control of Nonlinear Plant with the Linear Controller

We model the plant in Simulink (Appendix 1, section 5.1) to design and further iterate on the observer controller and integral action. The next part of the Simulink model is to design and implement the appropriate observer. An observer will estimate the representations of the larger system and returns state estimates to the controller. Our results from the MATLAB script to be implemented into the Simulink model are given in Figure 2 below.

```
The desired poles of the system are:
  -0.6667 + 0.2191i  -0.6667 - 0.2191i -13.3333 + 0.0000i -20.0000 + 0.0000i  -0.7333 + 0.0000i

The gain matrix K is
 -168.9584  -26.8127   -6.6933  -18.9588

The integral control gain Ki is
    4.9084

The control gains from integral action are
 -188.6211  -33.1278  -20.5964  -30.8557

The observer matrix L is
   19.5505   -1.8533
  114.0146  -18.9826
   -1.8644   18.6162
  -19.0897   84.8552
```

Figure 2: MATLAB Command Window Responses

4 Discussion and Conclusions

First, we had to fine tune the simulink model to meet the desired design specifications. This is because the uncontrolled system results in outputs that are very unstable. The plots of the system before integral control and observer are applied are pictured below in Figure 3.

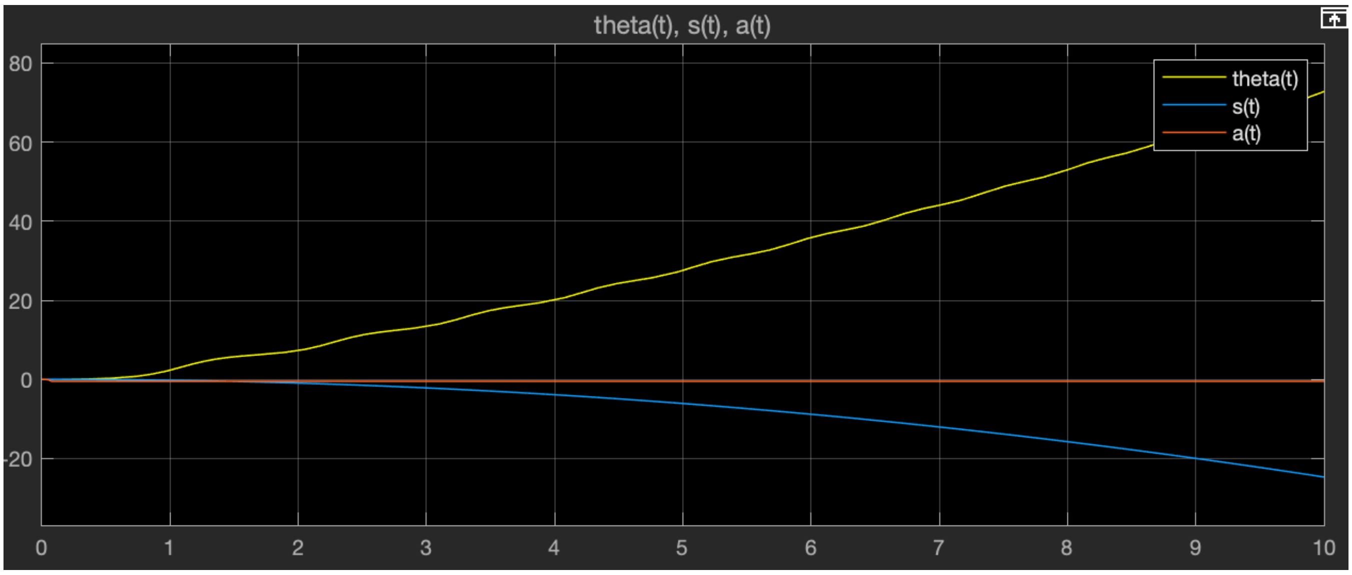


Figure 3: Uncontrolled Response of the System

As explained in the legend, the red line represents the response of the a(t), the yellow represents theta(t), and the blue line s(t). As we can see, the uncontrolled system has no second order step response for s(t), which is what we want our controlled system to have. Furthermore, the plots of θ(t) and s(t) go towards infinity and negativity infinity respectively, which indicates that the system is unstable. Comparatively, the plots of the controlled system are shown in Figure 4 below.

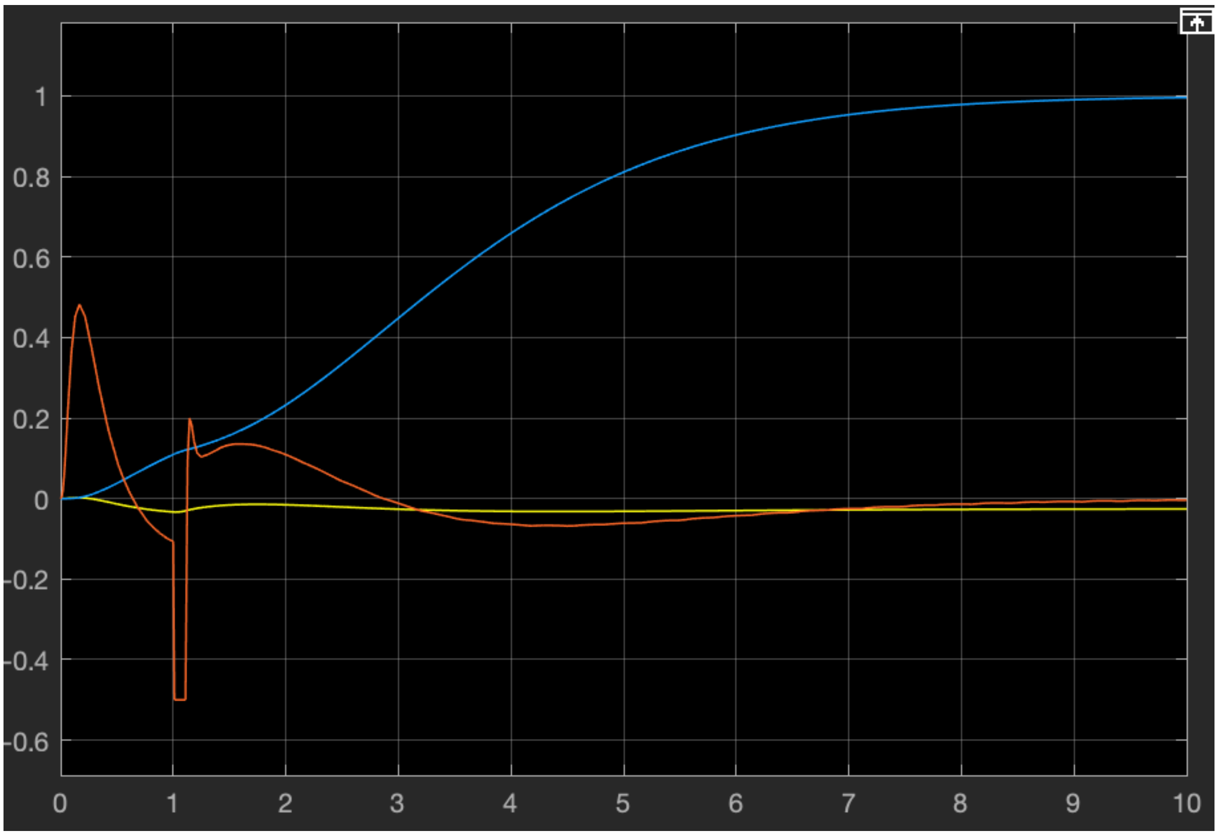


Figure 4: Controlled Response of the System

Once again, the red line represents the response of the a(t), the yellow represents θ(t), and the blue line s(t). Here, we can see that the integral control enforces a second order response on the a(t), or acceleration, plot. In the zoomed in version of the plot of θ(t) shown in Appendix 3, we can see that there is a second order response as well, which is unique from the uncontrolled version. Additionally, the θ(t) response levels out sooner, which indicates that the system is controlled. Thus, the observer and integral control has done its implied function of controlling the system and making it stable.

5 Appendix

5.1 Appendix 1: Simulink Block Diagrams

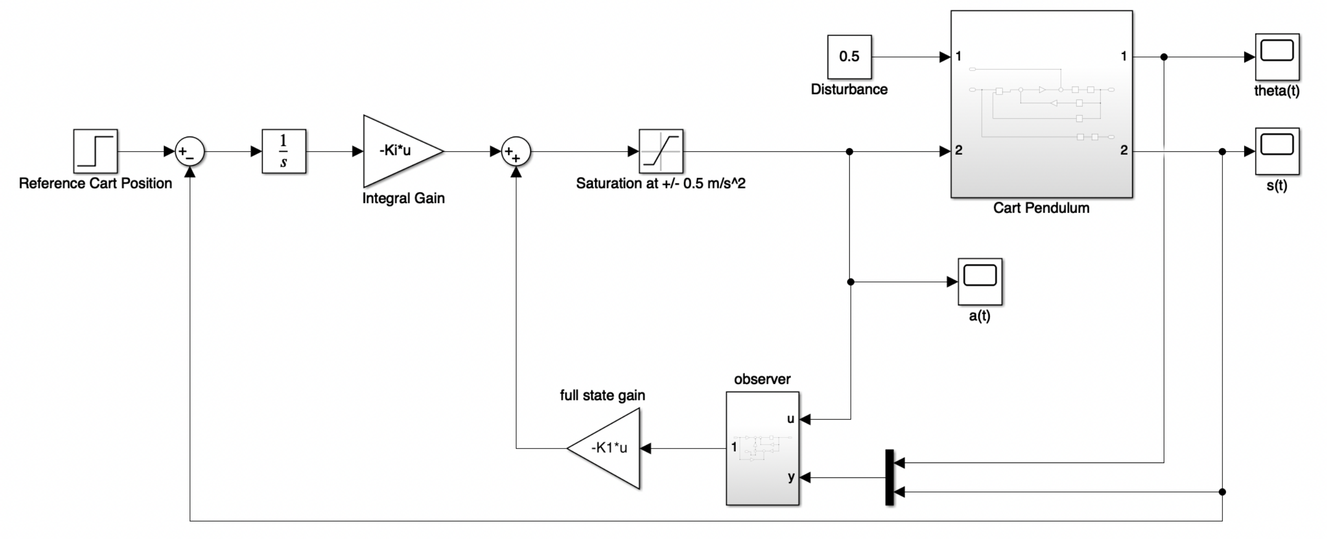


Figure 5: Inverted Pendulum Simulink Model

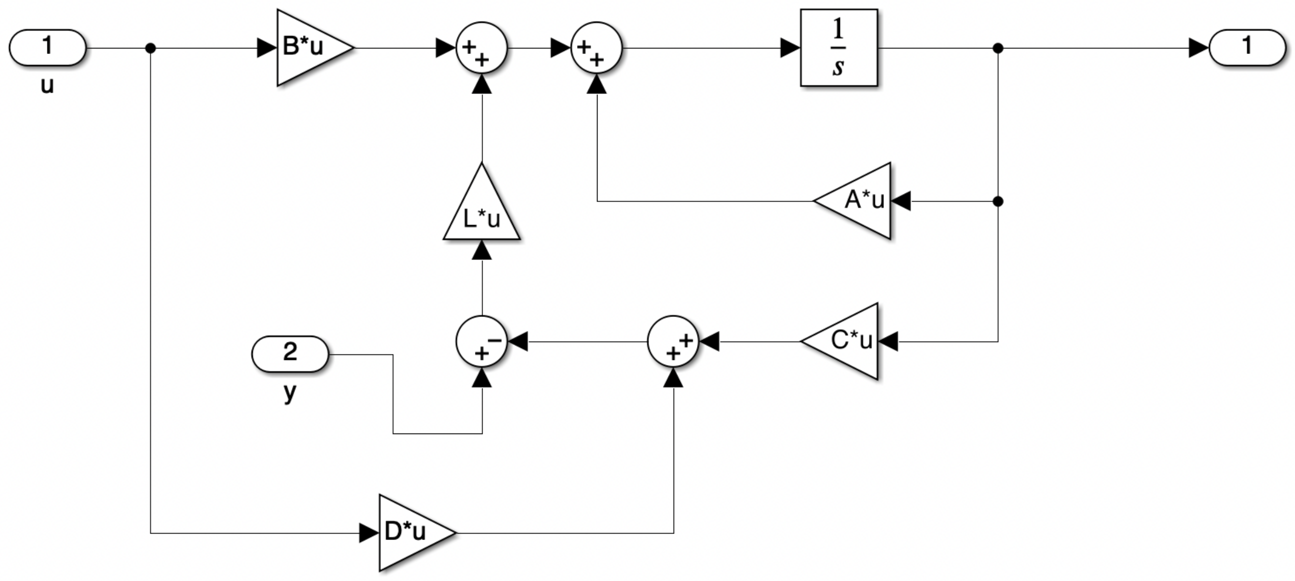


Figure 6: Observer Simulink Model

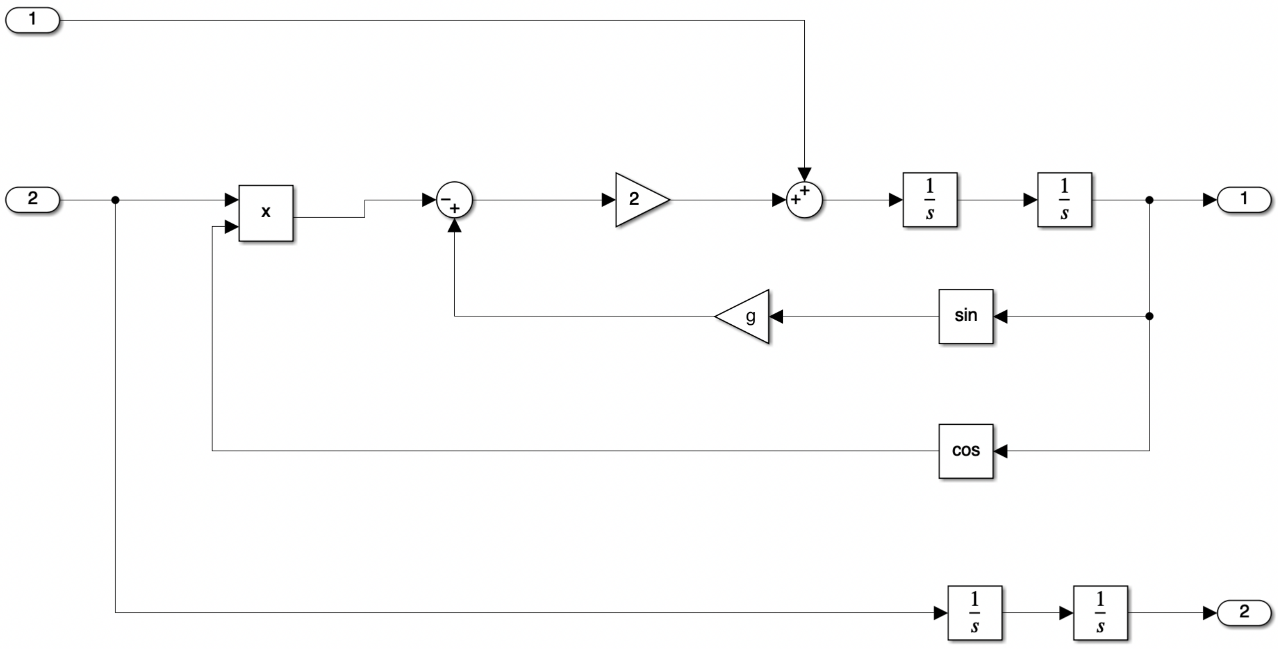
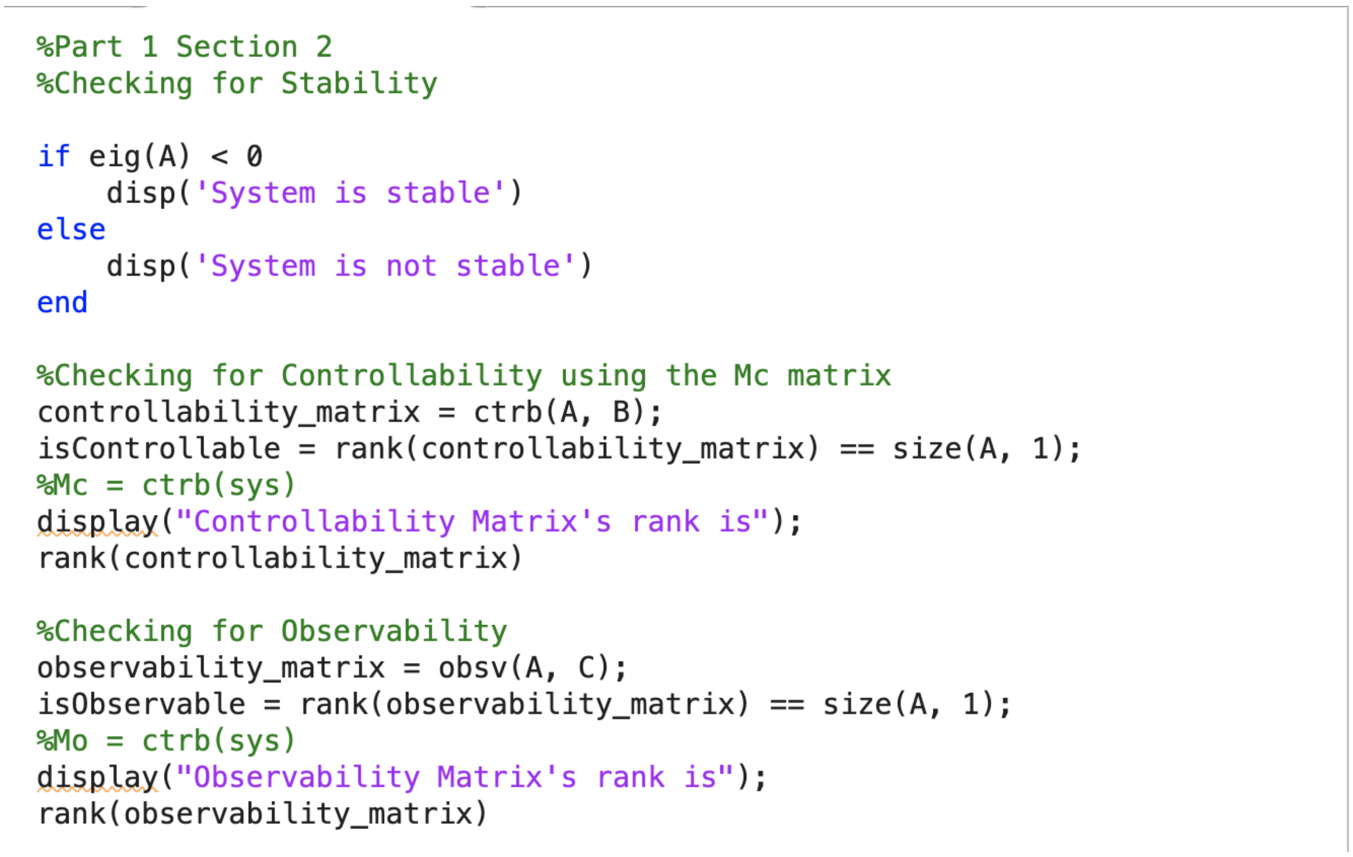


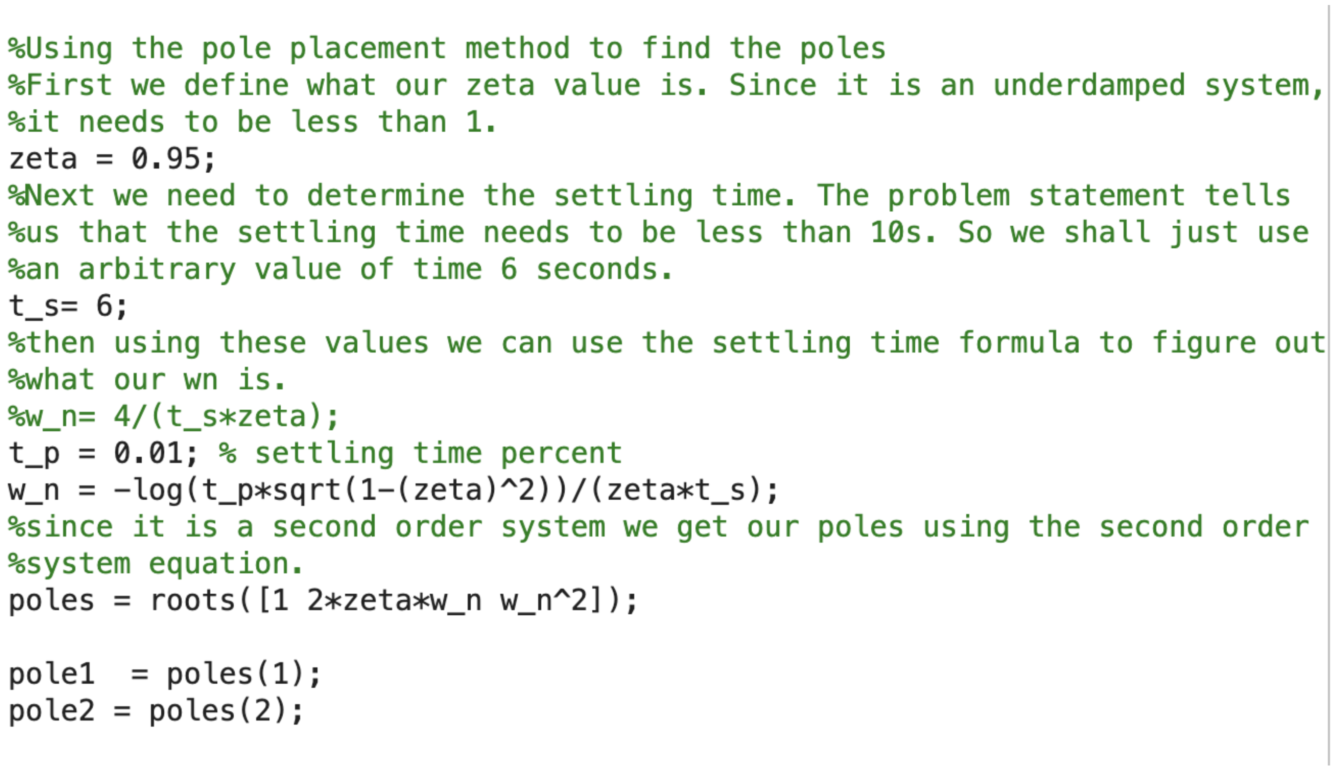
Figure 7: Cart Pendulum Simulink Model

5.2 Appendix 2: MATLAB Code

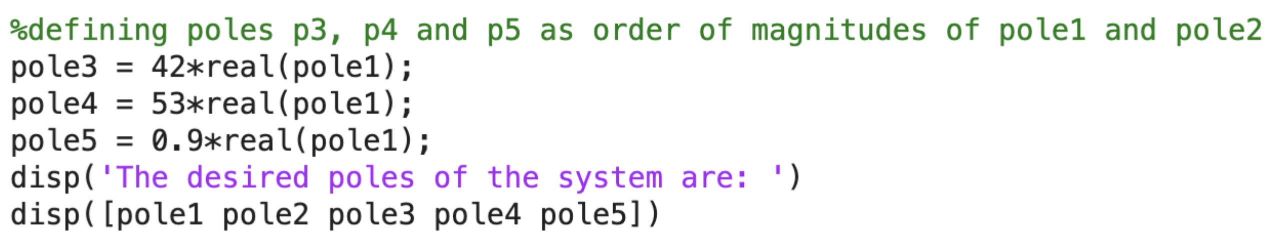
```
%Nandini Garg and Claudia Nanez
%E102 Inverted Pendulum Project
%Thursday April 18, 9:30 am

clear all

%Defining the given constants
g = 9.81; % Acceleration due to gravity (m/s^2)
pL = 0.5; %length of the pendulum in m

%Defining the disturbance parameters
alpha= 0.5;

%Defining the state-space matrices
%These state space matrices were found by using the given equations and
%representing it in terms of the subsequent variable. So if x1(t)= theta(t)
%then dx1(t)/dt=x2(t)and so on and so forth. Then we frame it in terms of
%A, B, C and D from the AxBu and CxDu equations.
A = [0 1 0 0; g/pL 0 0 0; 0 0 0 1; 0 0 0 0];
B = [0; -1/pL; 0; 1];
C = [1 0 0 0; 0 0 1 0];
D = [0; 0];

%Creating the state-space model
sys = ss(A, B, C, D);
```



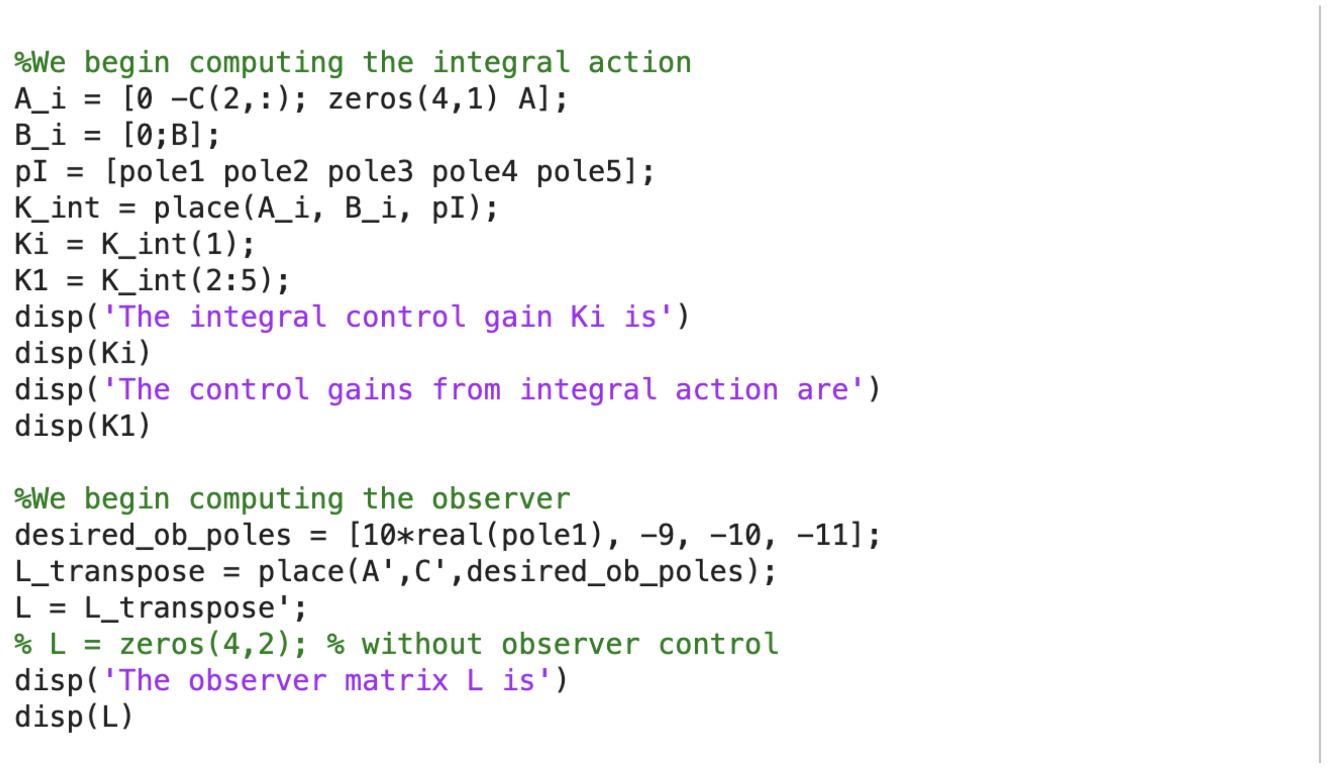
```
%Part 1 Section 2
%Checking for Stability

if eig(A) < 0
    disp('System is stable')
else
    disp('System is not stable')
end

%Checking for Controllability using the Mc matrix
controllability_matrix = ctrb(A, B);
isControllable = rank(controllability_matrix) == size(A, 1);
%Mc = ctrb(sys)
display("Controllability Matrix's rank is");
rank(controllability_matrix)

%Checking for Observability
observability_matrix = obsv(A, C);
isObservable = rank(observability_matrix) == size(A, 1);
%Mo = ctrb(sys)
display("Observability Matrix's rank is");
rank(observability_matrix)
```



```
%Using the pole placement method to find the poles
%First we define what our zeta value is. Since it is an underdamped system,
%it needs to be less than 1.
zeta = 0.95;
%Next we need to determine the settling time. The problem statement tells
%us that the settling time needs to be less than 10s. So we shall just use
%an arbitrary value of time 6 seconds.
t_s= 6;
%then using these values we can use the settling time formula to figure out
%what our wn is.
%w_n= 4/(t_s*zeta);
t_p = 0.01; % settling time percent
w_n = -log(t_p*sqrt(1-(zeta)^2))/(zeta*t_s);
%since it is a second order system we get our poles using the second order
%system equation.
poles = roots([1 2*zeta*w_n w_n^2]);

pole1  = poles(1);
pole2 = poles(2);
```



```
%defining poles p3, p4 and p5 as order of magnitudes of pole1 and pole2
pole3 = 42*real(pole1);
pole4 = 53*real(pole1);
pole5 = 0.9*real(pole1);
disp('The desired poles of the system are: ')
disp([pole1 pole2 pole3 pole4 pole5])

%Now on to designing state feedback control
%We begin doing so by computing the state feedback gain matrix K
p = [pole1 pole2 pole3 pole4];
K = place(A,B,p);
disp('The gain matrix K is')
disp(K)
```



```
%We begin computing the integral action
A_i = [0 -C(2,:); zeros(4,1) A];
B_i = [0;B];
pI = [pole1 pole2 pole3 pole4 pole5];
K_int = place(A_i, B_i, pI);
Ki = K_int(1);
K1 = K_int(2:5);
disp('The integral control gain Ki is')
disp(Ki)
disp('The control gains from integral action are')
disp(K1)

%We begin computing the observer
desired_ob_poles = [10*real(pole1), -9, -10, -11];
L_transpose = place(A',C',desired_ob_poles);
L = L_transpose';
% L = zeros(4,2); % without observer control
disp('The observer matrix L is')
disp(L)
```

5.3 Appendix 3: Final Images

For ease of inspection, we have decided to include the final images of the s(t), a(t), and θ(t) separate below.

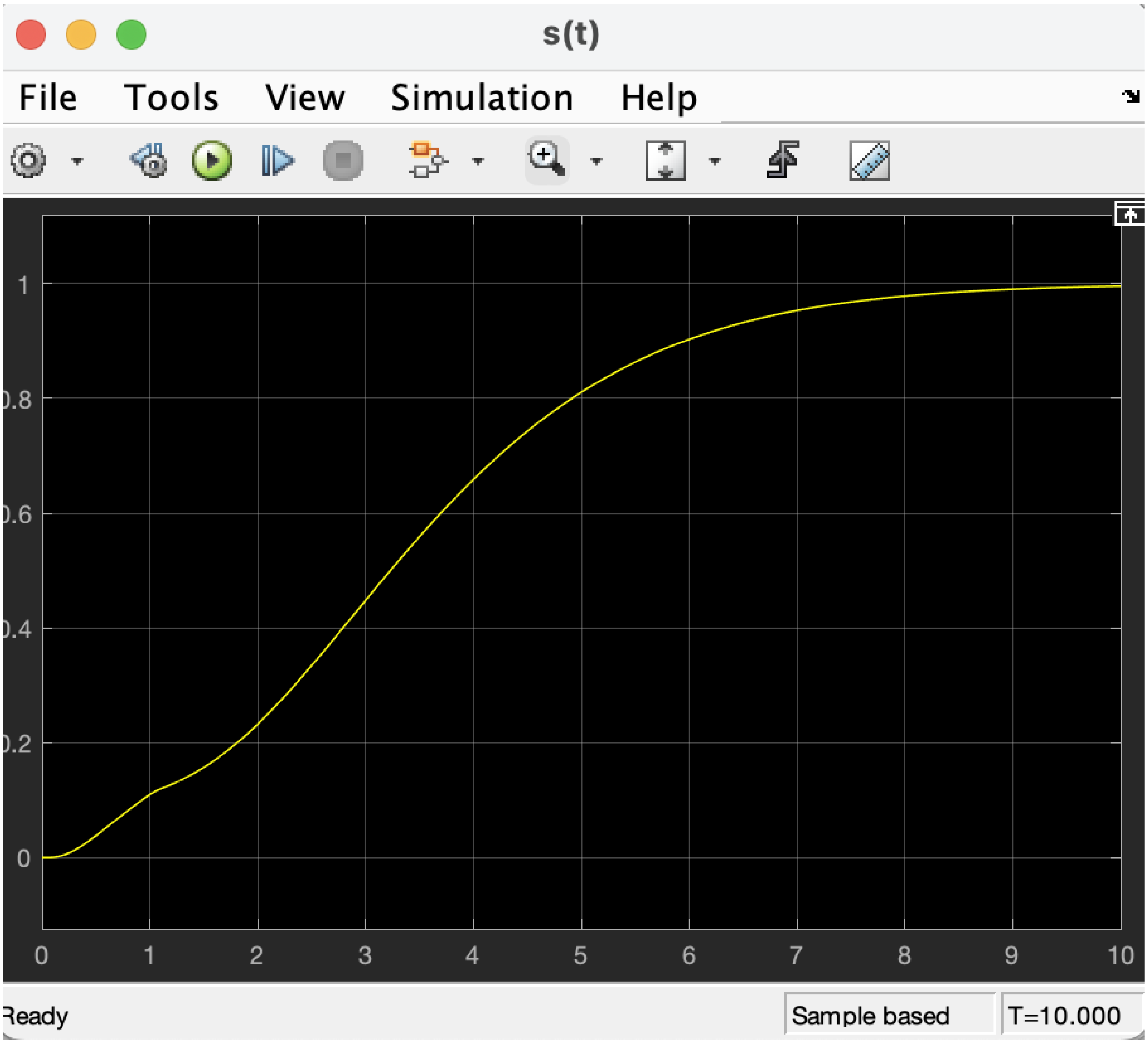


Figure 8: Controlled s(t)

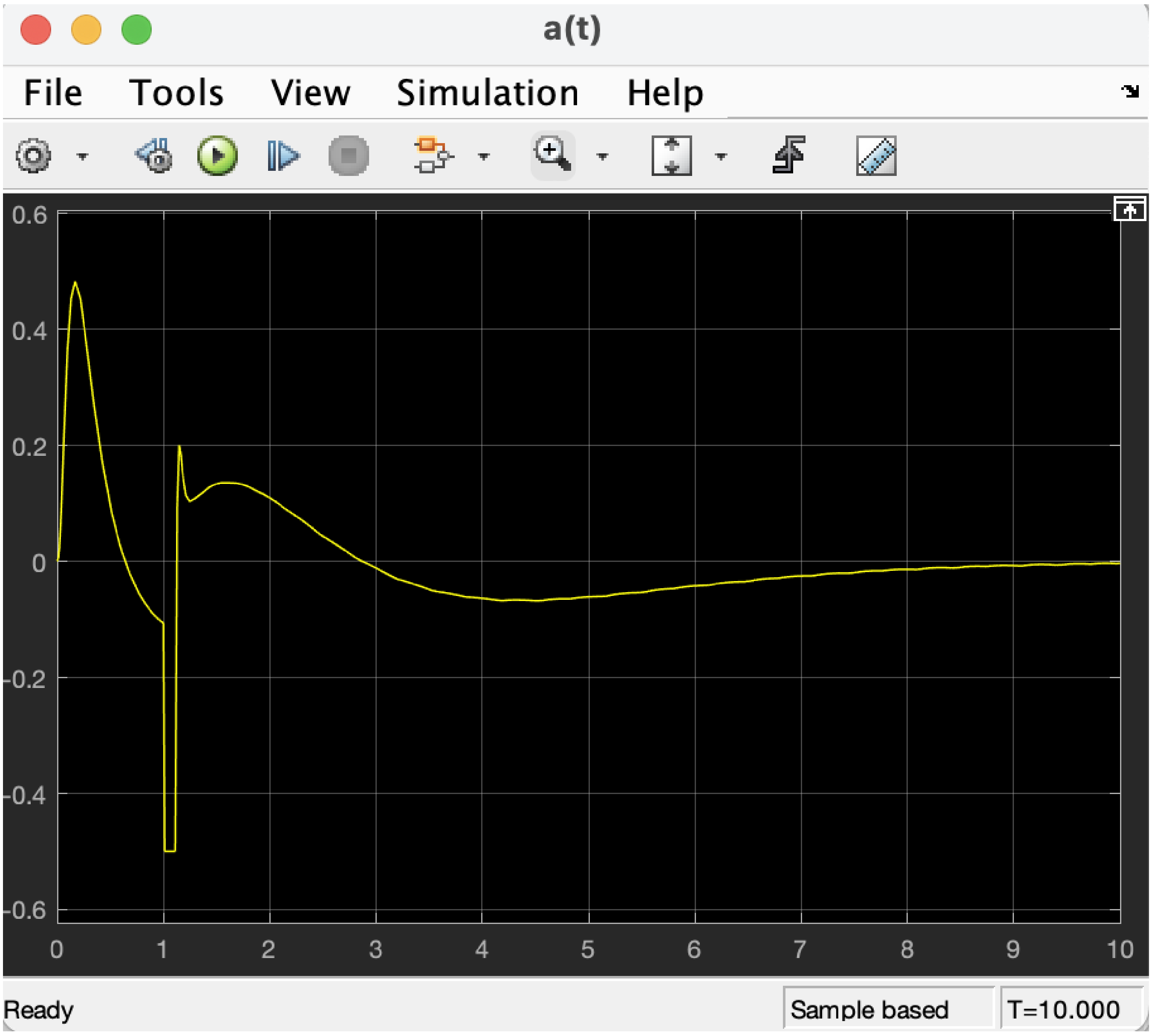


Figure 9: Controlled a(t)

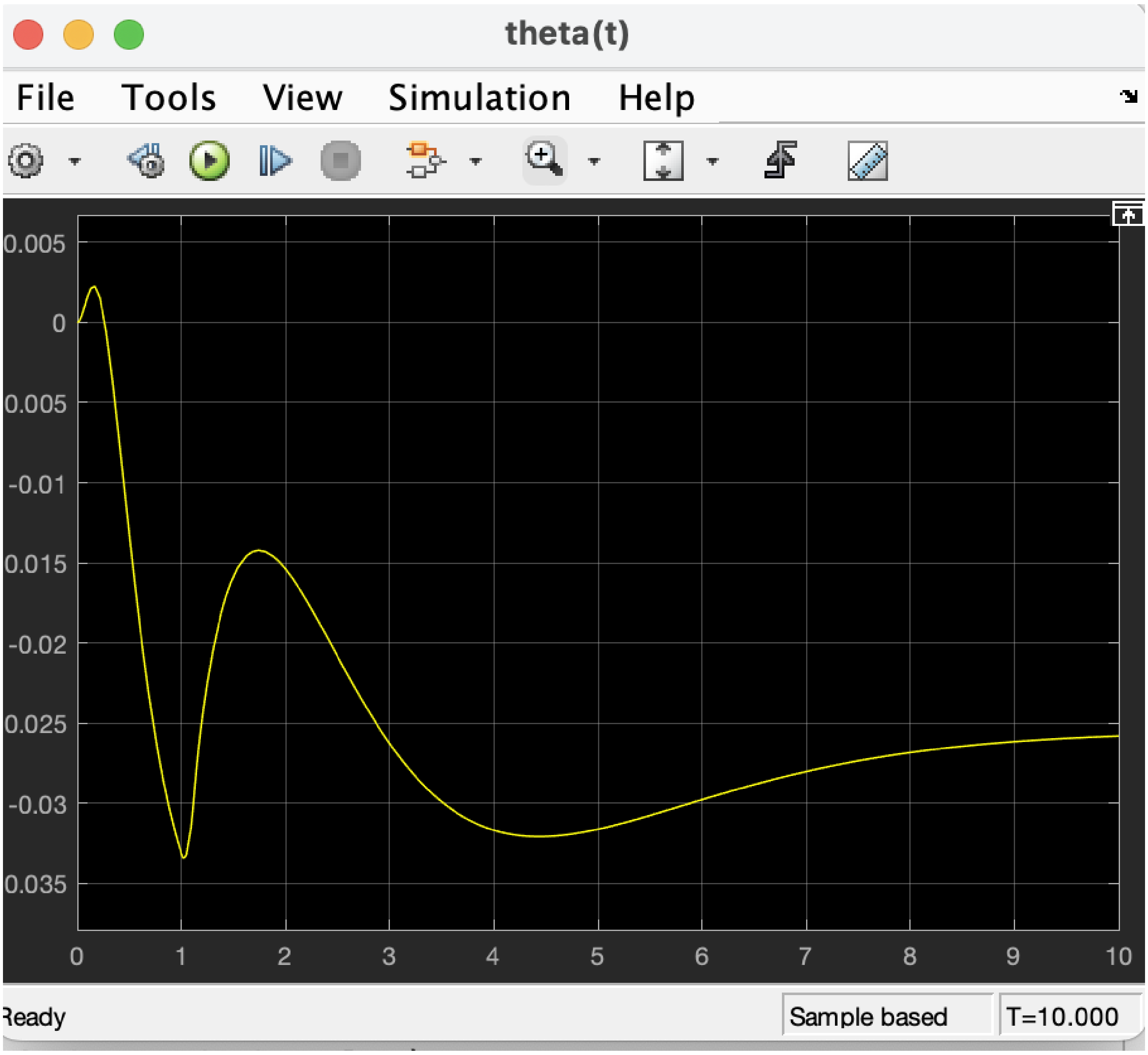


Figure 10: Controlled (θ (t)