Lab 2

Basic Concepts in Control System Design

Name: Mingjun Wu, Matthew Domine

Date: 03/08/2021

Introduction

The goal of this lab:

- to understand some of the basic concepts behind control theory: equilibrium points, stability, feedback, steady-state response, and linearization
- To become more familiar with the MATLAB and Simulink environment.

Theory

Poles and Zeros

Recall that the poles of a transfer function (TF) are those values of the Laplace transform variable, s, for which the denominator of the TF is zero, including any roots shared with the numerator, i.e. roots of the denominator which 'cancel' out with the numerator. Similarly, the zeros of a transfer function (TF) are those values of s that cause the TF to become zero, including roots shared with the denominator.

Stability

The time response of a linear time invariant (LTI) dynamical system $c(t) = c_{forced}(t) + c_{natural}(t)$, where $c_{forced}(t)$ is the forced response (driven by the input applied to the system) and $c_{natural}(t)$ is the natural response (driven by the system's initial states).

A linear, time-invariant system is called:

- stable if the natural response approaches zero as time approaches infinity.
- unstable if the natural response grows without bound as time approaches infinity.
- marginally stable if the natural response neither decays nor grows but remains constant or oscillates as time approaches infinity.

For LTI dynamical systems one can discuss stability easily in terms of the locations of the poles of the system's TF. A system is stable if all poles lie in the left half of the complex plane (LHP). A system is unstable if its TF has at least one pole in the right half of the complex plane (RHP), or a pole of multiplicity greater than one on the imaginary axis. A system is marginally stable if it has no poles in the RHP and only poles of multiplicity one on the imaginary axis.

Analysis & Results

1. Pre Lab

1.1 Simple Feedback System

Consider the feedback system shown in Figure 1.

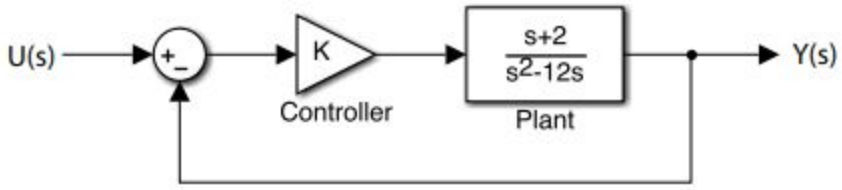
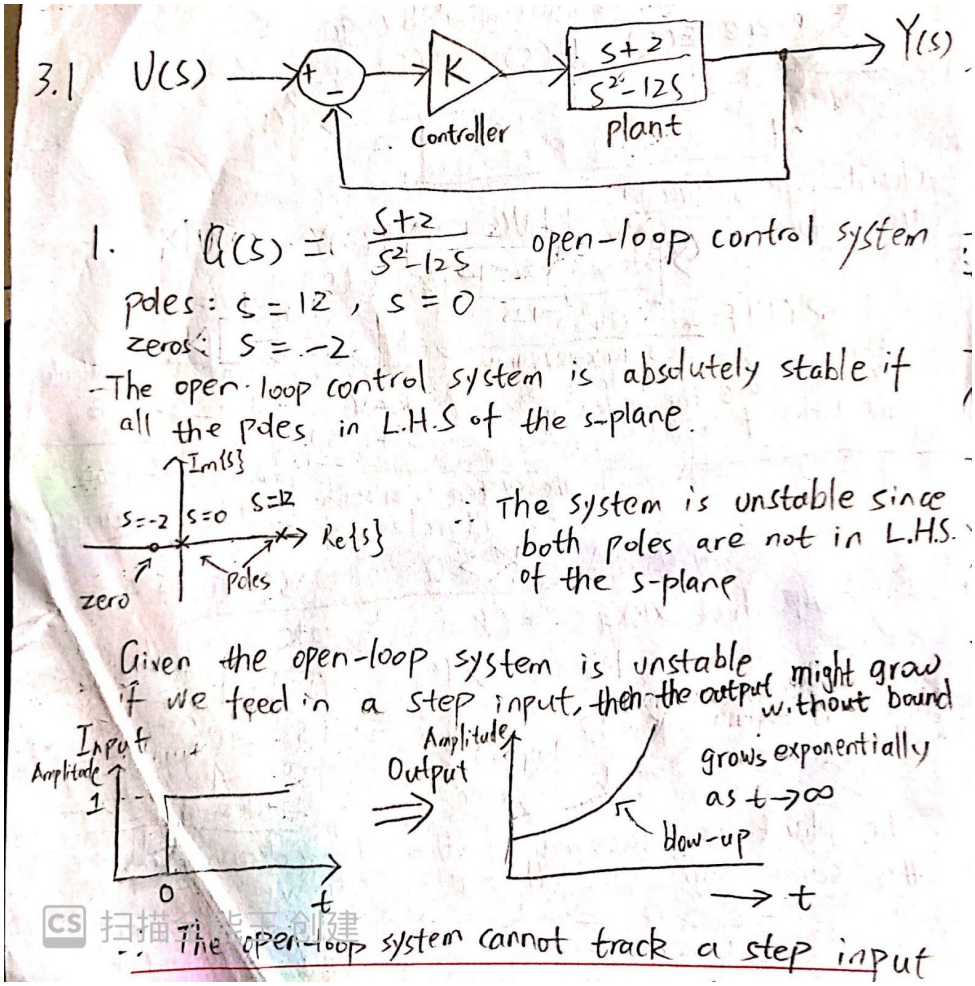


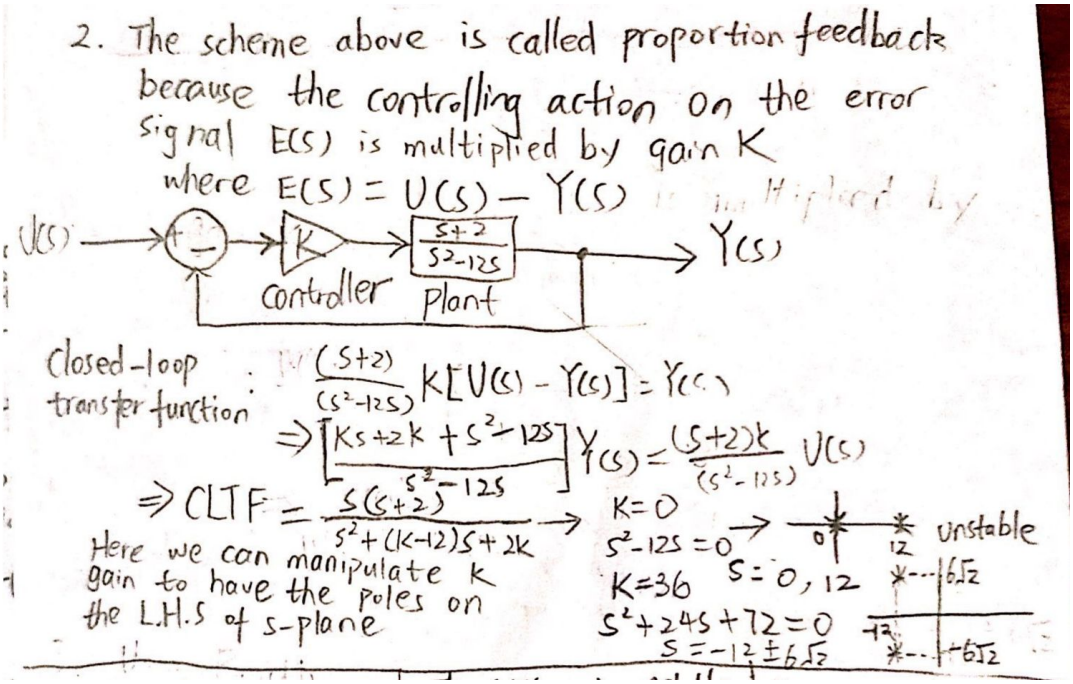
Figure 1: Simple proportional control system

Suppose the control goal is to track a step input.

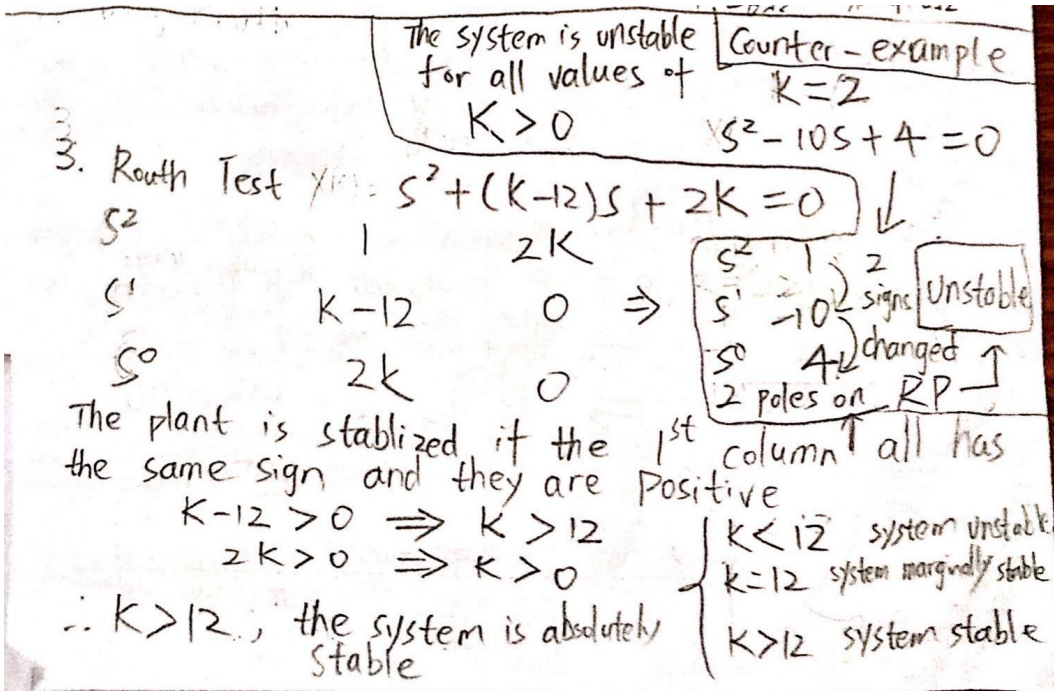
1. What are the poles and zeros of the open-loop system? Is the open-loop system stable? Can the open-loop system track a step input? Explain.



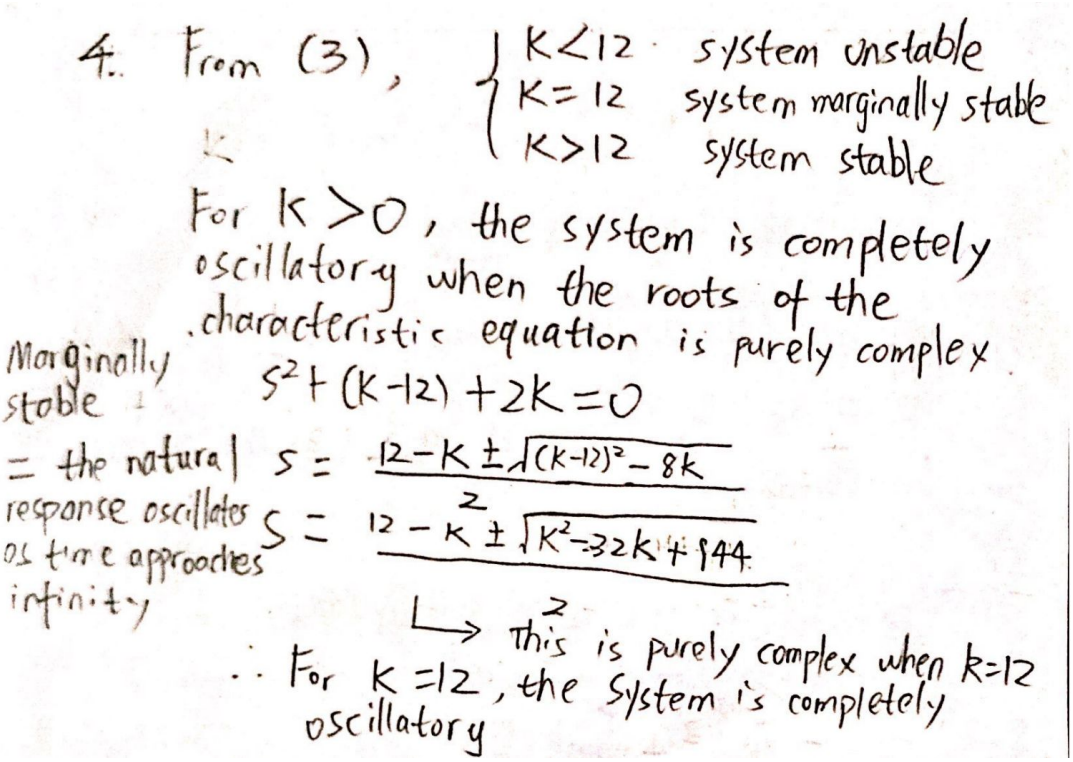
2. Suppose you use proportional feedback control as shown in Figure 1. Explain why the scheme above is called proportional feedback. Explain (mathematically) how this scheme can be used to stabilize the system.



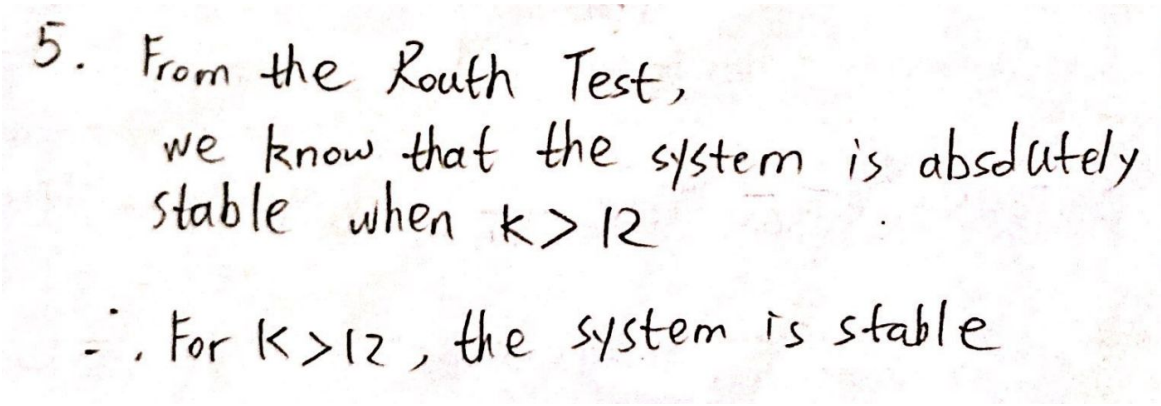
3. Is the system stable for all values of K > 0? Prove or find a counterexample.



4. For what values of K > 0 is the system completely oscillatory?



5. For what values of K > 0 is the system stable?



Nonlinear Damped Pendulum

Consider a damped pendulum as in Figure 2.

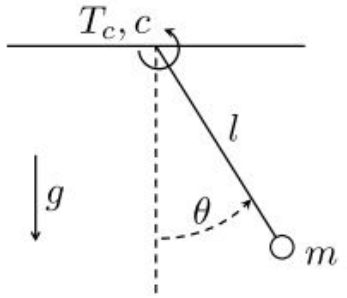


Figure 2: A damped pendulum

The nonlinear equation of motion for the pendulum in Figure 2 is given by:

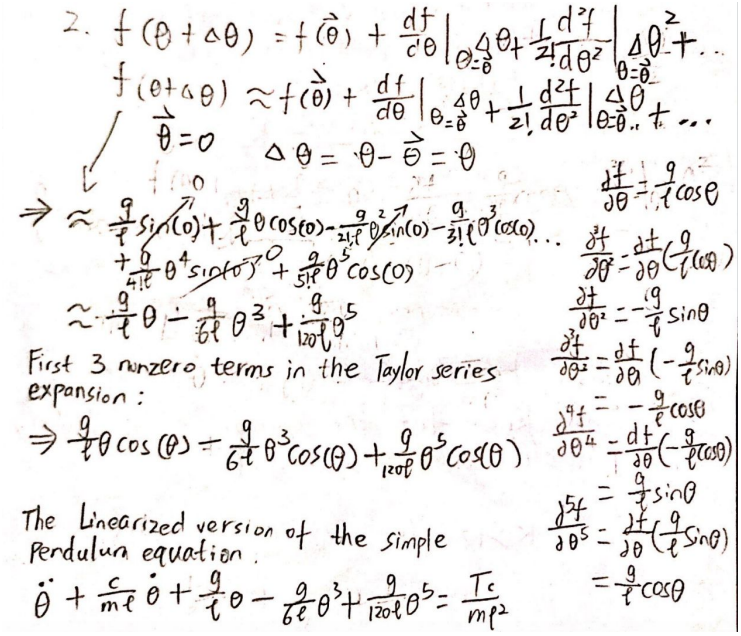
$$\ddot{\theta} + \frac{c}{ml}\dot{\theta} + \frac{g}{l}sin\theta = \frac{T_c}{ml^2} \quad (1)$$

Here θ is the angle of the pendulum from the vertical (in rad), c is the velocity damping term (in s^{-1}), m is the mass of the pendulum (in kg), l is the length (in m), g is the acceleration due to gravity (in ms^{-2}) and Tc is the force input (in N · m). For this problem assume that g = 9.81 ms^{-2}, l = 2.0 m, m = 1.0 kg, and c = 0.25 s^{-1} .

1. What is the nonlinear term in the pendulum equation above?

   Nonlinear Term: $\frac{g}{l}sin\theta$

2. Very often, you "remove" the nonlinearity in the plant by linearizing the plant about an operating point. In this case, we will do a Taylor series expansion around θ = 0 for the nonlinearity and disregard the nonlinear terms in the expansion. What are the first three terms in the Taylor series expansion of the nonlinearity? What is the linearized version of the simple pendulum equation?



2. Lab

2.1 Simple Feedback System

1. Create a Simulink model of the simple feedback system shown in Figure 1.

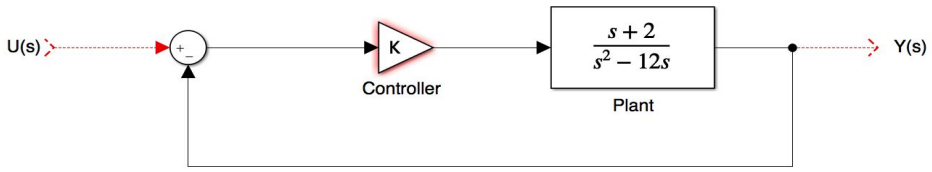


Figure 1: Simulink Model of the Simple Feedback system

2. Verify the system is oscillatory for the value(s) of K that you found in the Pre-Lab. You can test some select values of K in this interval. Plot the step response of the system (i.e. the input signal should be a step input which steps from 0 to 1 at time $t_0 = 0$) for four different values of K.

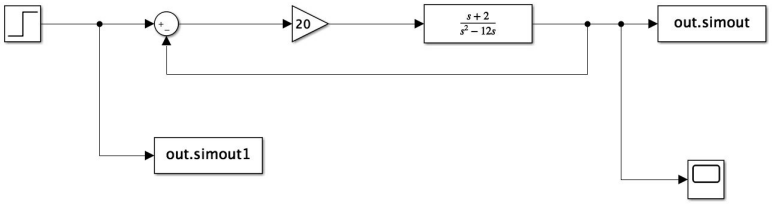


Figure 2: Simulink Model of the Simple Feedback system with a step input which steps from 0 to 1 at time $t_0 = 0$

```
% 4.1.2
plot(out.tout, out.simout,'r');
hold on
plot(out.simout1,'-.y*');
title('Step Response with K = 20')
xlabel('Time(sec)');
ylabel('Theta(rad)     Force Input(N·m)')
legend('Step response of the system','Step Input','Location','NorthEast')
```

Figure 3:Example of the Matlab Code when K = 20

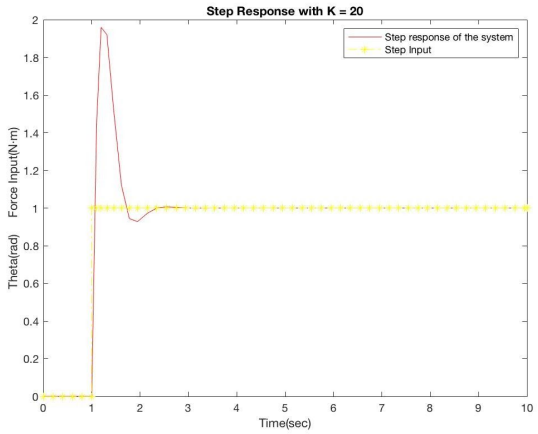


Figure 3:Step Response when K = 20

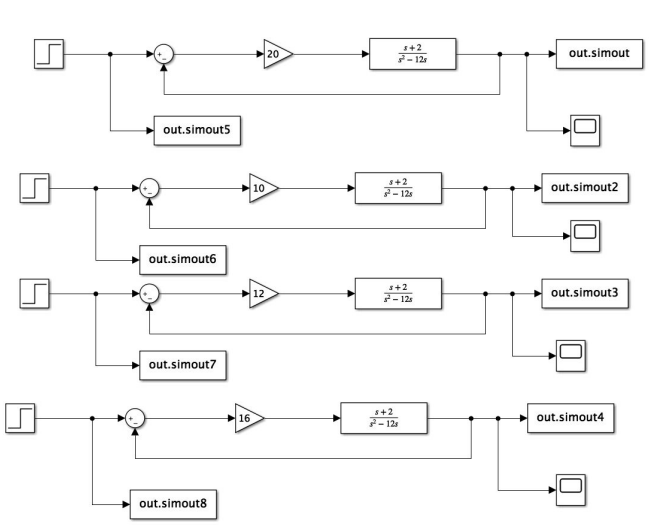


Figure 4:Simulink Models for K= 10, 12, 16, 20 20

```
subplot(2,2,2)
plot(out.tout, out.simout3,'r');
hold on
plot(out.simout7,'-.y*');
title('Step Response with K = 12')
xlabel('Time(sec)');
ylabel('Theta(rad)     Force Input(N·m)')
legend('Step response of the system(rad)','Step Input','Location','NorthEast')

subplot(2,2,3)
plot(out.tout, out.simout4,'r');
hold on
plot(out.simout8,'-.y*');
title('Step Response with K = 16')
xlabel('Time(sec)');
ylabel('Theta(rad)     Force Input(N·m)')
legend('Step response of the system(rad)','Step Input','Location','NorthEast')

subplot(2,2,4)
plot(out.tout, out.simout,'r');
hold on
plot(out.simout5,'-.y*');
title('Step Response with K = 20')
xlabel('Time(sec)');
ylabel('Theta(rad)     Force Input(N·m)')
legend('Step response of the system(rad)','Step Input','Location','NorthEast')
```

Figure 5: Matlab Code for plotting the step responses when K= 10, 12, 16, 20

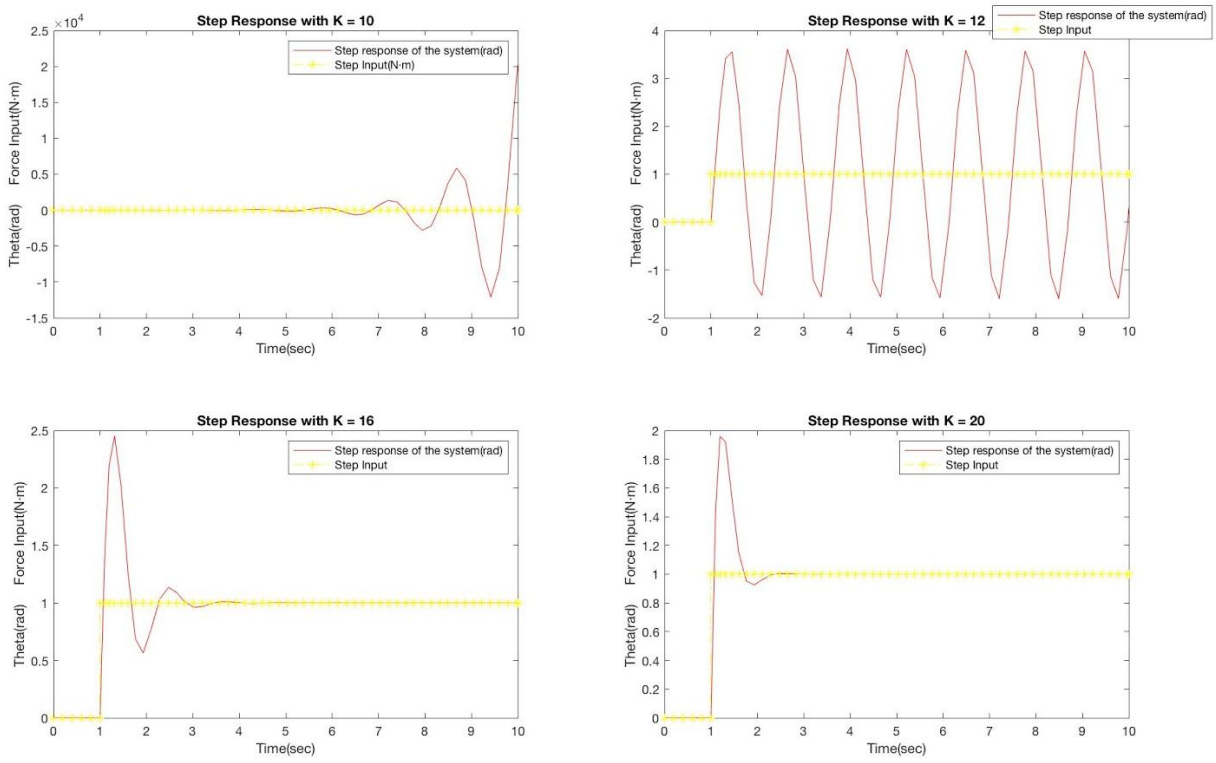


Figure 7: Plot of the step responses when K= 10, 12, 16, 20

From figure 7 above, it is verified that the system is oscillatory for the value of K found in the Pre-Lab 1.

3. Verify the system is stable for values of K in the corresponding interval you found in the Pre-Lab. You can test some select values in this interval. Plot the step response of the system for four different values of K in this interval (on the same graph). **Report the location of the poles of the closed loop system for each value of K.**

```
%4.1.3
plot(out.simout1,'-.y*');
hold on
plot(out.tout, out.simout,'b');
hold on
plot(out.tout, out.simout2,'g');
hold on
plot(out.tout, out.simout3,'r');
hold on
plot(out.tout, out.simout4,'k');
title('Step Response When K > 12 And It Is Stable')
xlabel('Time(sec)');
ylabel('Theta(rad)     Force Input(N·m)')
legend('Step Input(N·m)','K=13(rad)','K=16(rad)','K=24(rad)','K=32(rad)','Location','NorthEast')
```

Figure 8: Matlab Code for plotting the step responses of the system for the four K values which K > 12

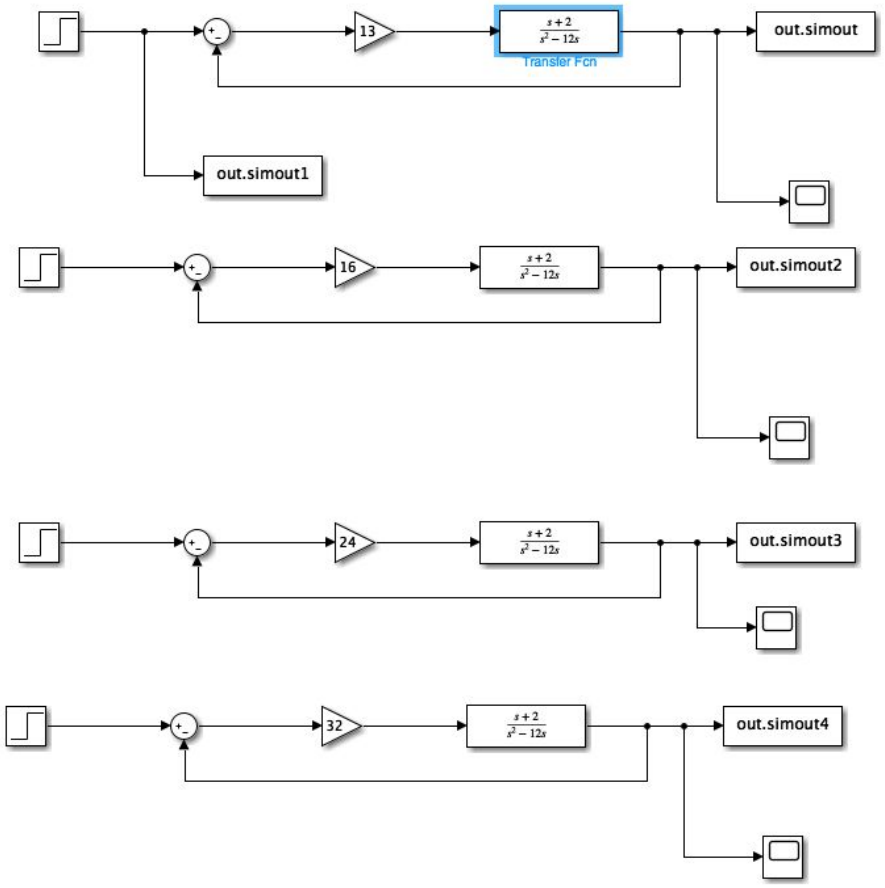


Figure 9: Simulink Models for the system when K = 13, 16, 24, 32

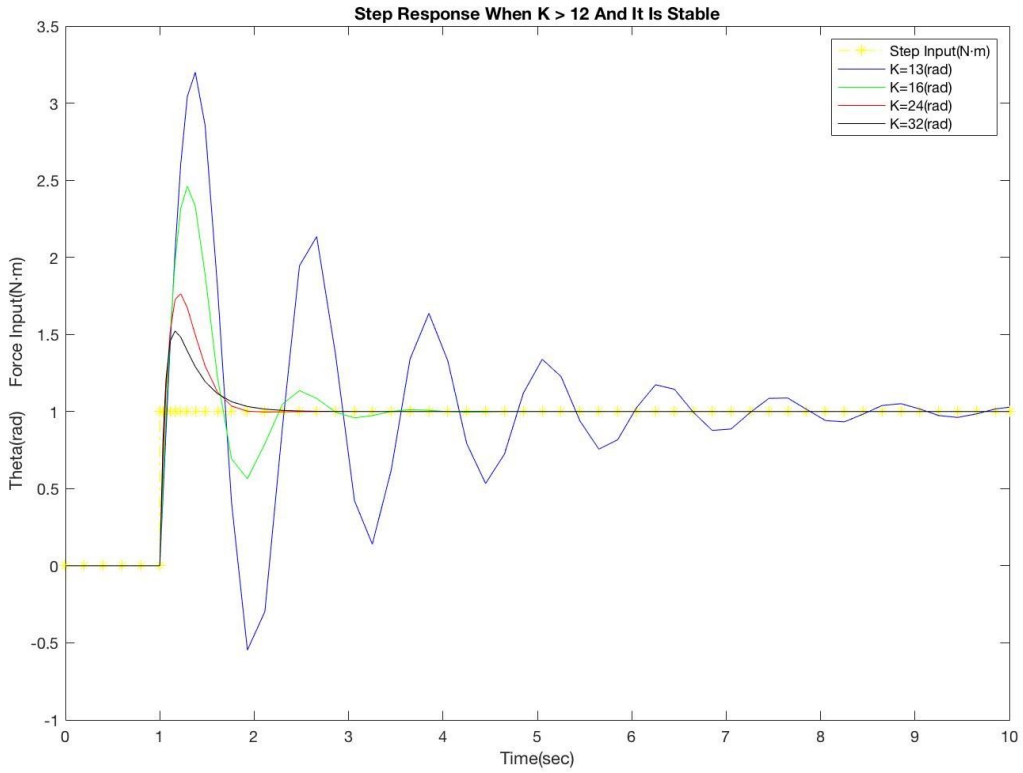


Figure 10: Plot of the step responses of the system for K = 13, 16, 24, 32 on the same graph

```
%4.1.3
% K =13
K1 = 13;
sys1 = tf([K1 2*K1],[1 K1-12 2*K1]);
P1 = pole(sys1)

%K = 16
K2 = 16;
sys2 = tf([K2 2*K2],[1 K2-12 2*K2]);
P2 = pole(sys2)

%K = 24
K3 = 24;
sys3 = tf([K3 2*K3],[1 K3-12 2*K3]);
P3 = pole(sys3)

%K = 32
K4 = 32;
sys4 = tf([K4 2*K4],[1 K4-12 2*K4]);
P4 = pole(sys4)
```

Figure 11: Matlab code for finding the poles when K = 13, 16, 24, 32

```
P1 =

  -0.5000 + 5.0744i
  -0.5000 - 5.0744i


P2 =

  -2.0000 + 5.2915i
  -2.0000 - 5.2915i


P3 =

  -6.0000 + 3.4641i
  -6.0000 - 3.4641i


P4 =

   -16
    -4
```

Figure 12: Output Display of the locations of the poles

Table 1: Report of the location of poles

Value of K	*Location of Poles (round to 2 decimals)*
13	$-0.5 \pm 5.07j$
16	$-2 \pm 5.29j$
24	$-6 \pm 3.46j$
32	$-4, -16$

From figure 10 above, it is verified that the system is stable for the values of K > 12 found in the Pre-Lab 1.

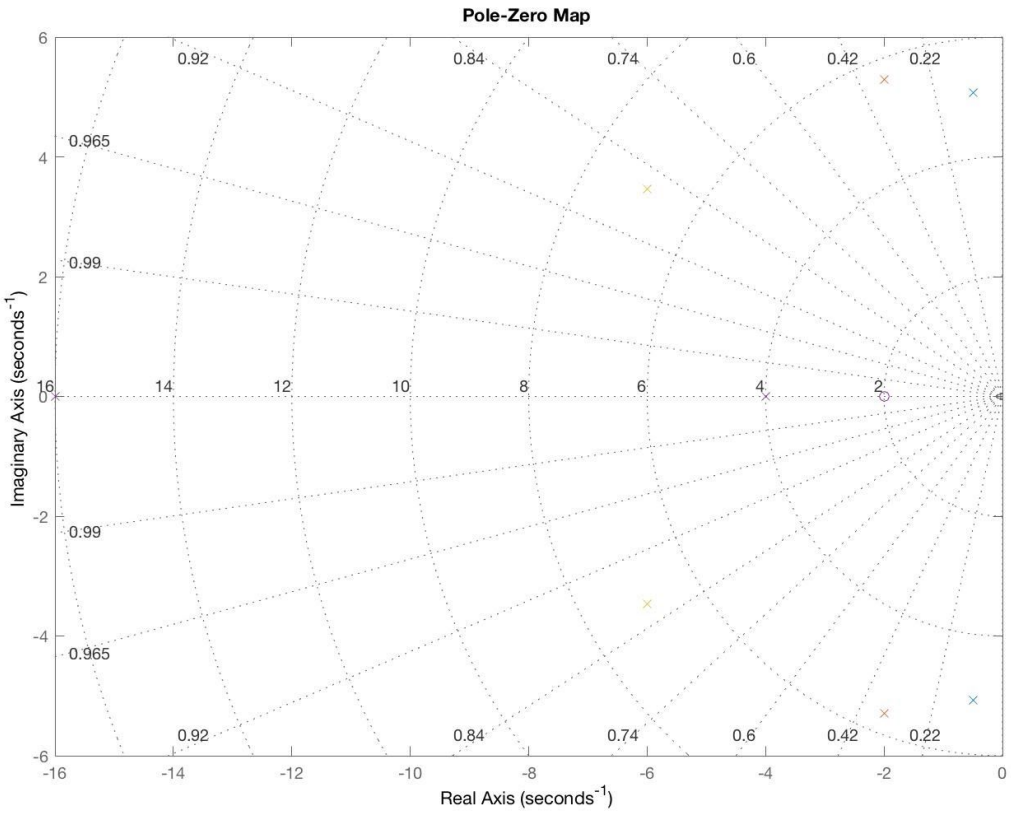


Figure 13: Using the pzplot function and graphically examine the location of the poles

From figure 13 above, it is verified that all the poles are in the left-hand plane of the s-plane and hence the systems are stable for the four values of K.

4. Suppose we want our output response to not significantly exceed the final value of our input (step) signal. We can set a bound on our response using the maximum percent overshoot (%OS) – the percent by which the maximum value of the response exceeds the steady state (final) value. In this case, let us specify that we want the %OS to be less or equal than 25% for all $t \geq t_0$.
   Using simulation (not math), determine for what values of K (find a range – precision of 0.1 for bounds is good enough) the %OS is less or equal than 25%? Also show graphically that your bounds on K are correct (needs to be evident on the plot). Report the location of the closed loop poles for this value of K.

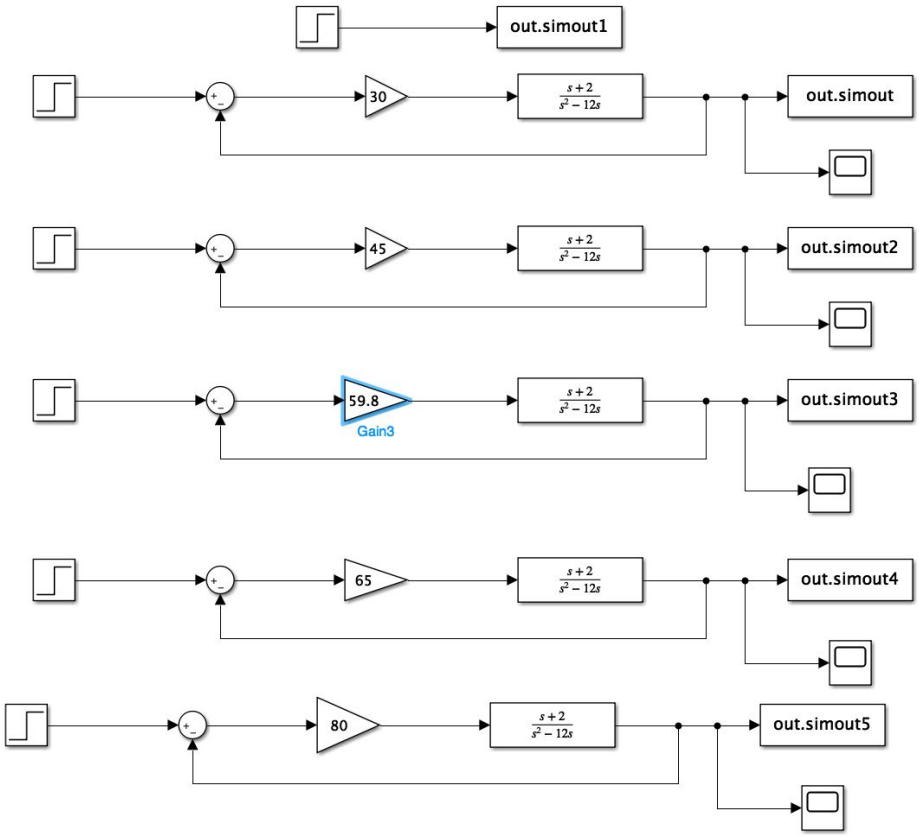


Figure 14: Simulink Models for the system when K = 30, 45, 59.8, 65, 80

The maximum percent overshoot is 25%, which is equal to a step input with a value of 1.25, and hence a step response which steps from 0 to 1.25 at time $t_0 = 0$ is used as the reference bound. Though it is a bit of a silly approach, the K value with the precision of 0.1 is found by tuning the K values until reaching the smallest K value that do not exceed the amplitude of 1.25 since we only have to rely on Simulink.

```
%4.1.4
plot(out.simout1,'y');
hold on
plot(out.tout, out.simout,'m');
hold on
plot(out.tout, out.simout2,'b');
hold on
plot(out.tout, out.simout3,'k');
hold on
plot(out.tout, out.simout4,'r');
hold on
plot(out.tout, out.simout5,'g');

title('Percent Overshoot for Various Values of K')
xlabel('Time(sec)');
ylabel('Theta(rad)')
legend('25%OS Bound(rad)','K=30(rad)','K=45(rad)','K=59.8(rad)','K=65(rad)','K=80(rad)','Location','NorthEast')
```

Figure 15: Matlab code for finding the poles when K =30, 45, 59.8, 65, 80

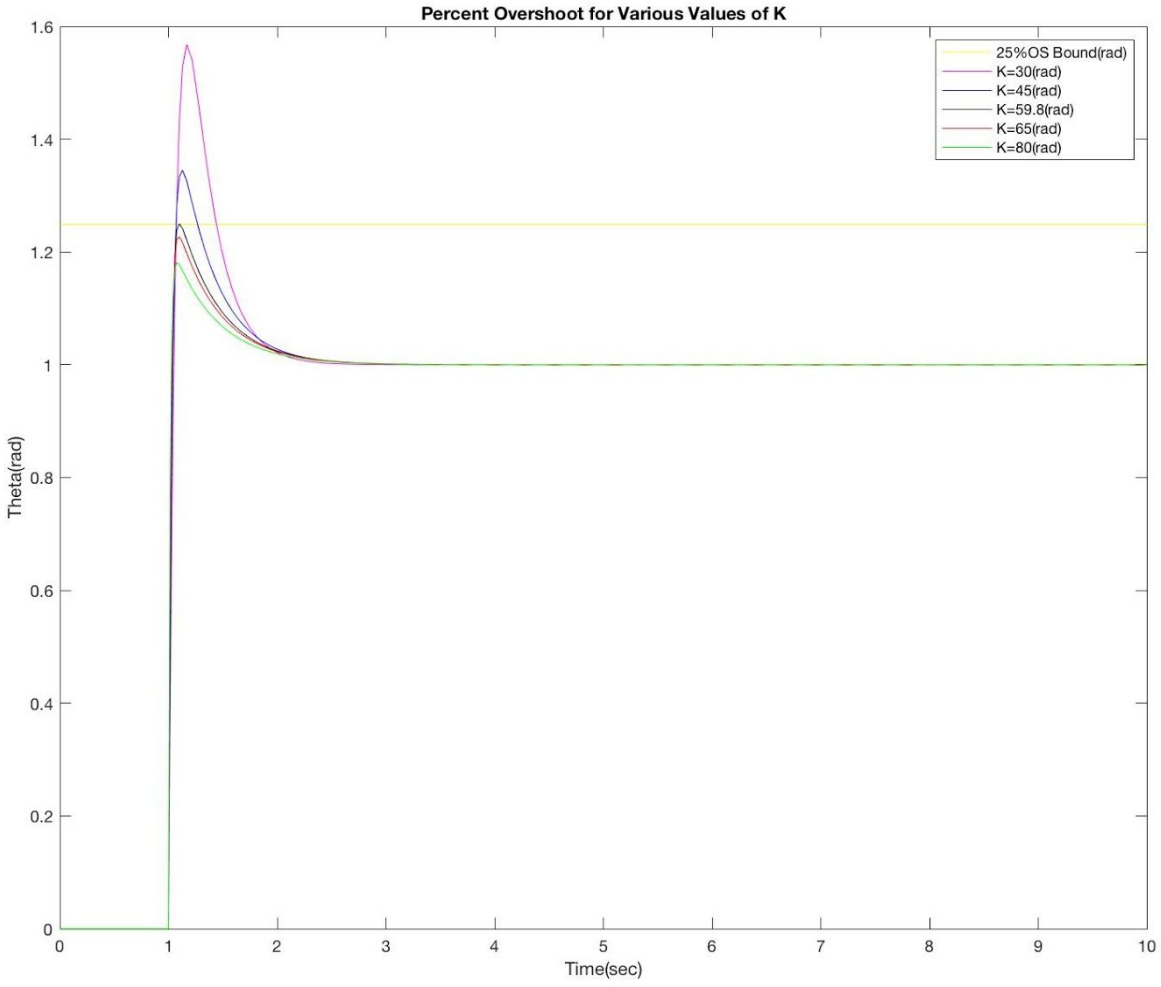


Figure 16: plot of the step responses when K =30, 45, 59.8, 65, 80

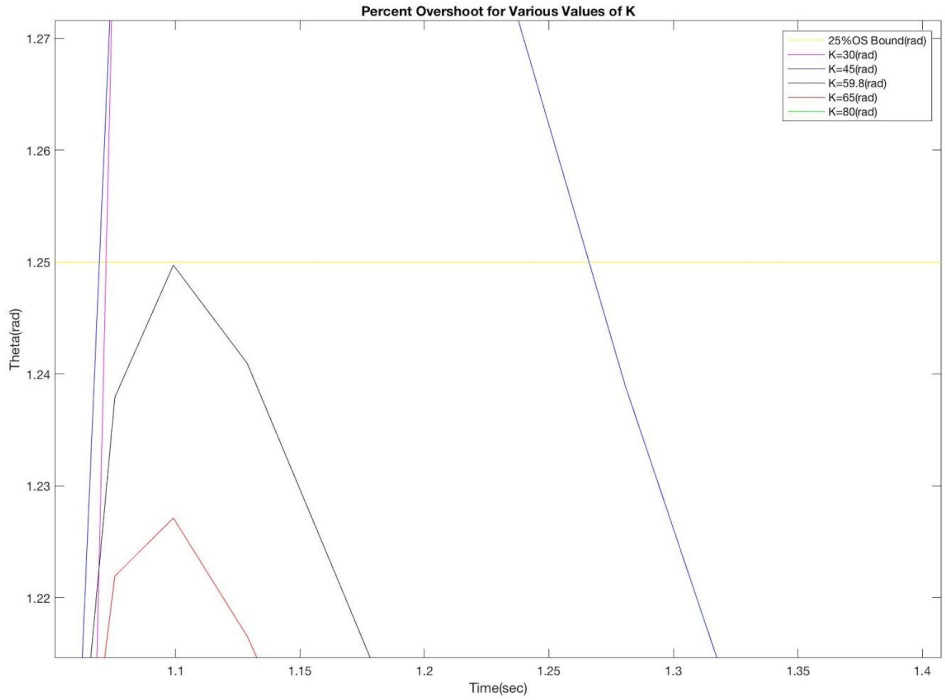


Figure 17: Graphically verified that the bound on K hold

From figure 17 above, it is verified that K = 59.8 is the bound that the maximum value of the response do not exceed the steady state value for more than 25%, as the peak of the step response when K = 59.7 would go over 1.25.

```
%K=30
K1 = 30;
sys1 = tf([K1 2*K1],[1 K1-12 2*K1]);
P1 = pole(sys1)

%K = 45
K2 = 45;
sys2 = tf([K2 2*K2],[1 K2-12 2*K2]);
P2 = pole(sys2)

%K = 59.8
K3 = 59.8;
sys3 = tf([K3 2*K3],[1 K3-12 2*K3]);
P3 = pole(sys3)

%K = 65
K4 = 65;
sys4 = tf([K4 2*K4],[1 K4-12 2*K4]);
P4 = pole(sys4)


%K = 80
K5 = 80;
sys5 = tf([K5 2*K5],[1 K5-12 2*K5]);
P5 = pole(sys5)
```

Figure 18: Matlab code for finding the poles when K = 30, 45, 59.8, 65, 80

```
P1 =

  -13.5826
   -4.4174


P2 =

   -30
    -3


P3 =

  -45.1511
   -2.6489


P4 =

  -50.4217
   -2.5783


P5 =

  -65.5595
   -2.4405
```

Figure 19: Output Display of the locations of the poles

Table 2: Report of the location of poles

Value of K	*Location of Poles (round to 2 decimals)*
30	*-13.58, -4.42*
45	*-30, -3*
59.8	$-45.15,\ -2.65$
65	$-50.42,\ -2.58$
80	*-65.56, -2.44*

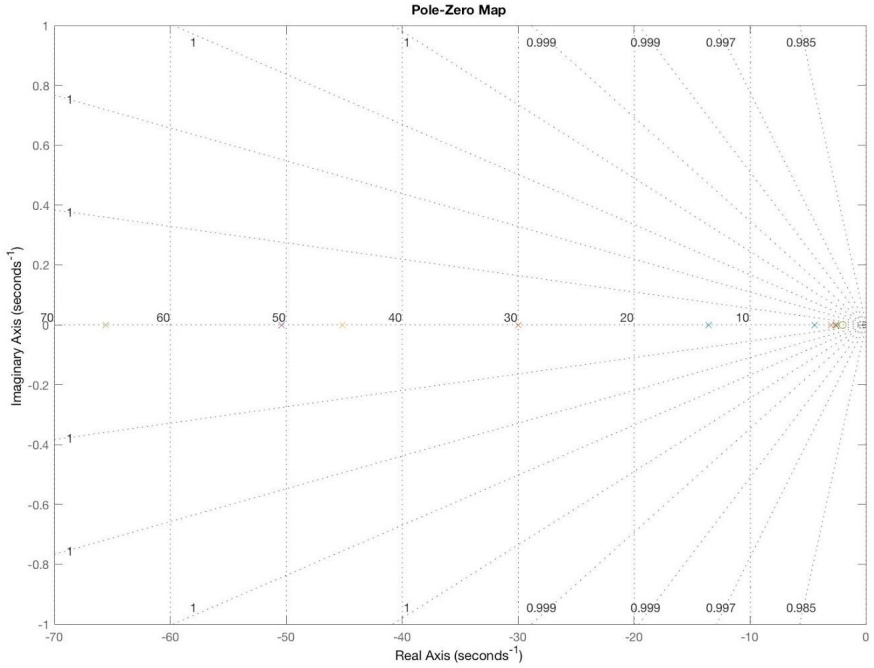


Figure 20: Using the pzplot function and graphically examine the location of the poles

5. How do the poles of the system change as K increases?
   It is observed from table 1 and table 2, that the real part of the poles gets larger and larger while the imaginary part gets smaller and smaller as K increases. This makes sense because the step response is faster, in other words, less time to reach steady-state and hence the step response has less oscillations as K increases.
   The distance between the two poles on the real axis is further and further and the pole closer to the origin is the dominant pole. The further pole corresponds to the exponential term in the step response which will die out very quickly in relation to the exponential terms corresponding to the dominant pole. Thus, the system effectively behaves as a first order system when K gets large enough.

2.2 Nonlinear Damped Pendulum

1. Model the nonlinear damped pendulum in Simulink. Note that this is not a feedback control system – you are building a block diagram representation of a plant! Instead of using derivatives, generate the necessary θ signals "backwards" by using integrators. Why might using numerical derivatives be bad? To simplify updating the model, define variables and use them as parameters in the Simulink blocks (e.g. instead of directly using the value of the pendulum length l in the block, define a variable l in Matlab and use it in the Simulink block). Include a figure of the Simulink model in your answer.

Numerical derivatives might be a bad idea since the derivative block outputs might be sensitive to the dynamics of the entire model. The nonlinear term $\frac{g}{l}sin(\theta)$ might result in unexpected fluctuations in the output which will cause the output to be inaccurate since the derivative block is sensitive to inputs. In addition, the time step size which determines the smoothness of the plot (where the smoother the plot the more accurate the output is) cannot be controlled and is based solely on the inputs. Therefore if the inputs to the block change rapidly the solver will not take the smaller steps that are taken by continuous blocks such as the integrator blocks which will return an inaccurate output.

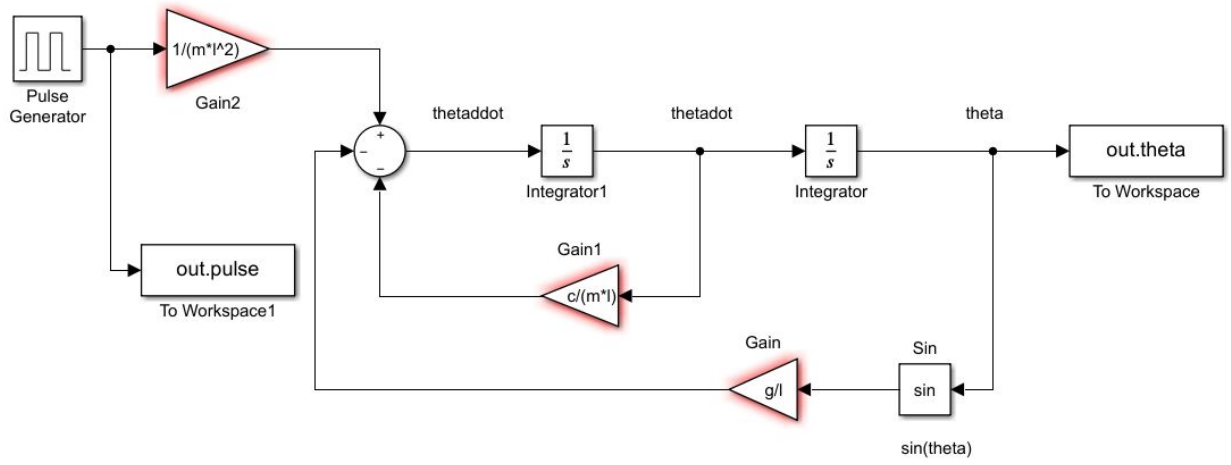


Figure 21: Simulink block diagram model of the nonlinear damped pendulum

```
%% Part 4.2
clear, clc, close all;
%Define Parameters
g = 9.81; %acceleration due to gravity [m/s^2]
m = 1; %mass of pendulum[kg]
c = 0.25; %velocity damping term[s^-1]

%Define Variables
l = 2; %length[m]
```

Figure 22: Parameters and Variables for the Simulink model defined in MATLAB

```
%Simulate
tstop = 70; %Simulation Time [s]
simout = sim('nlpendulum',tstop); %Outputs: Angle[rad], Pulse Input [N*m]
simout2 = sim('nlpendulum2',tstop); %Outputs: Angle[rad], Pulse Input [N*m]
simout3 = sim('nlpendulum3',tstop); %Outputs: Angle[rad], Pulse Input [N*m]
simout4 = sim('nlpendulum4',tstop); %Outputs: Angle[rad], Pulse Input [N*m]

%Plot Output
figure
plot(simout.theta)
ylabel('Theta (rad)')
title('4.2.2 Angle Theta over Time')

figure
plot(simout2.theta)
ylabel('Theta (rad)')
title('4.2.3 Angle Theta over Time')

figure
plot(simout3.theta)
ylabel('Theta (rad)')
title('4.2.4 Angle Theta over Time')

figure
plot(simout4.theta)
ylabel('Theta (rad)')
title('4.2.5 Angle Theta over Time')
```

Figure 23: MATLAB code for obtaining the Output Theta Plots

```
%% Check 4.2
%4.2.2
figure
plot(simout.theta)
hold on
plot(simout.pulse)
ylabel('F(t)')
title('4.2.2 Angle Theta vs Pulse Input Tc over Time')
legend('Theta (rad)','Pulse Input Tc (N*m)')

figure
plot(simout2.theta)
hold on
plot(simout2.pulse)
ylabel('F(t)')
title('4.2.3 Angle Theta vs Pulse Input Tc over Time')
legend('Theta(rad)','Pulse Input Tc (N*m)')

figure
plot(simout3.theta)
hold on
plot(simout3.pulse)
ylabel('F(t)')
title('4.2.4 Angle Theta vs Pulse Input Tc over Time')
legend('Theta(rad)','Pulse Input Tc (N*m)')

figure
plot(simout4.theta)
hold on
plot(simout4.pulse)
ylabel('F(t)')
title('4.2.5 Angle Theta vs Pulse Input Tc over Time')
legend('Theta(rad)','Pulse Input Tc (N*m)')
```

Figure 24: MATLAB code for obtaining the Output Theta and Pulse Input Plots

2. Plot the θ response of this system to a single pulse having amplitude 10 and width 0.2 seconds. There are several ways to generate such a pulse; one easy way is to use a "pulse generator" with a large period (e.g. 100 s) and suitable pulse width (0.2% when using a period of 100 s). When you plot the graph, show the response for 70 seconds of time. To what value is the pendulum angle converging to? Hint: At some point you should also plot your pulse input to verify that it is the correct amplitude and width. This does not need to be included in your lab report, but will help you avoid mistakes.

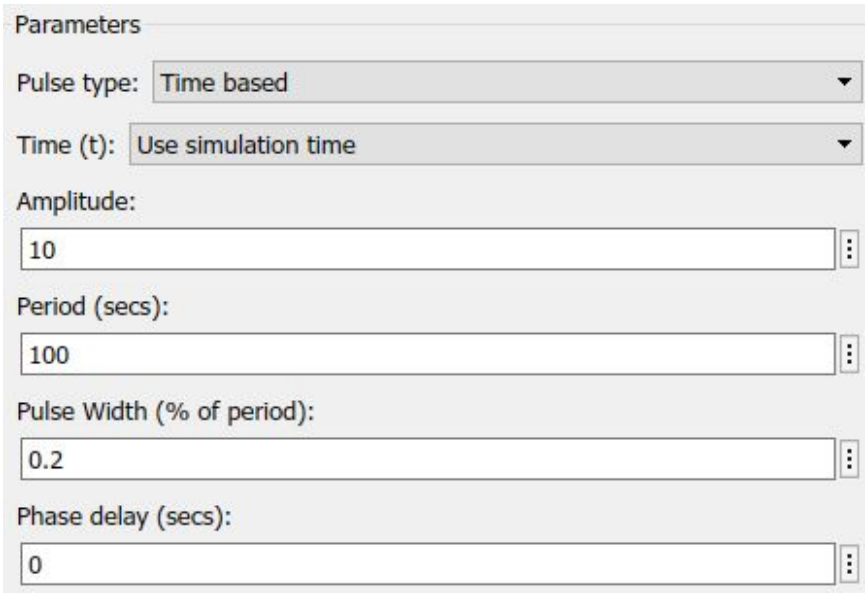


Figure 25: Parameter for Pulse Generator Block (Amplitude = 10, Period 100s, Pulse Width 0.2s)

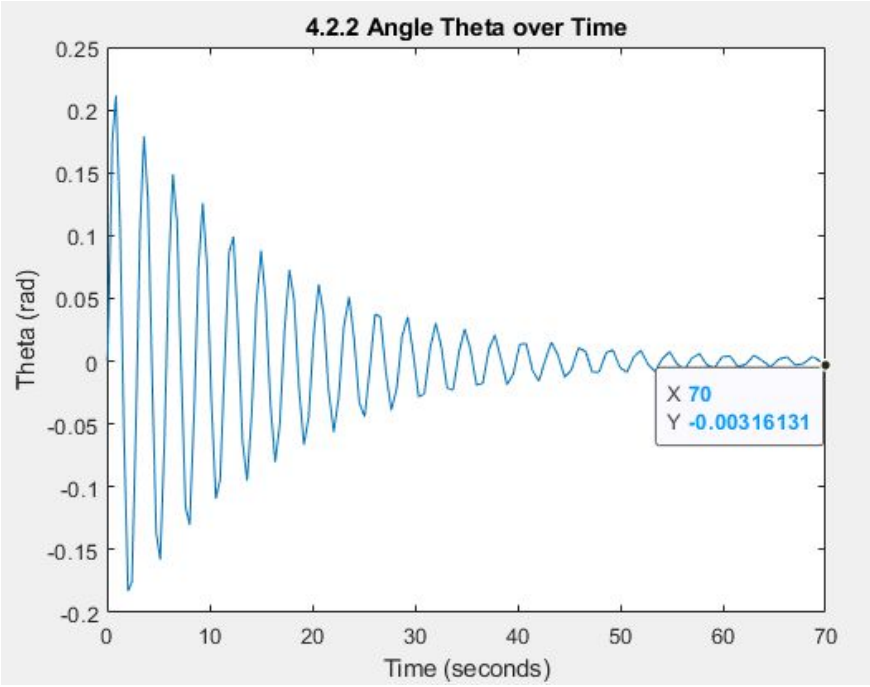


Figure 26: Nonlinear Pendulum Theta Response over Time to a single pulse having amplitude 10 and width 0.2 seconds

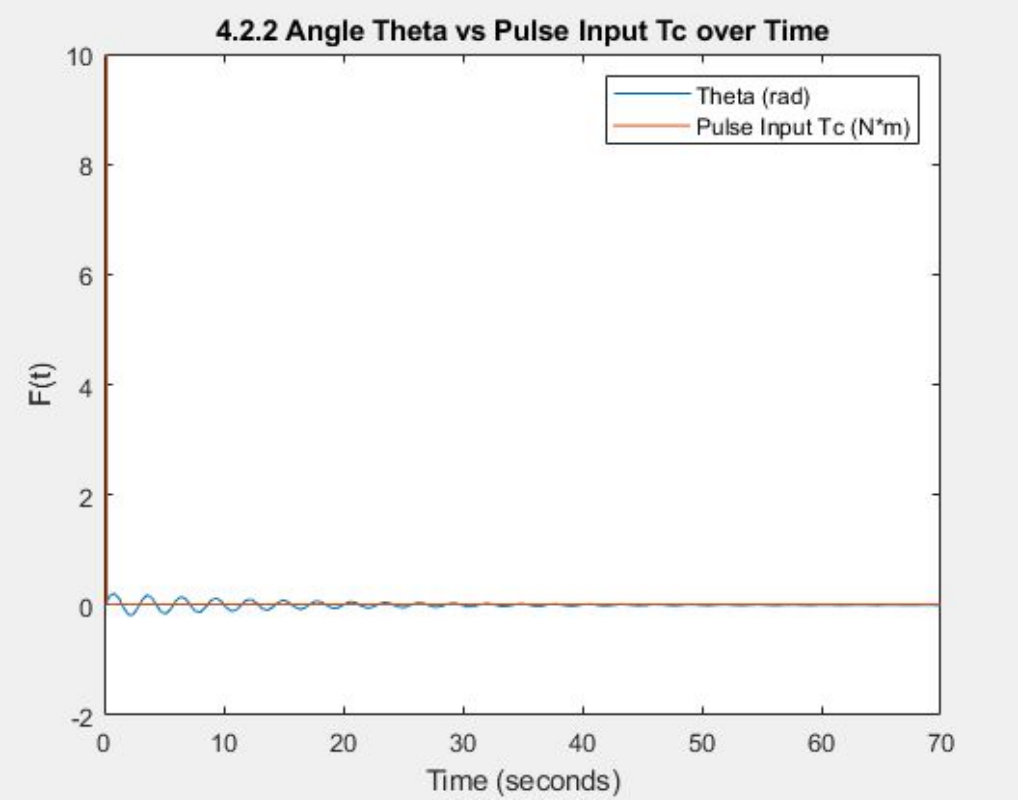


Figure 27: Nonlinear Pendulum Theta Response with Pulse Input over Time, where the Pulse input has amplitude 10 and width 0.2 seconds

When you plot the graph, show the response for 70 seconds of time. To what value is the pendulum angle converging to?

In Figure 26, it can be observed that the value that the pendulum angle is converging to is 0 rad.

3. Plot the θ response of this system to a pulse having amplitude 250 and width 0.2 seconds. To what new value does the pendulum angle converge? Explain why.

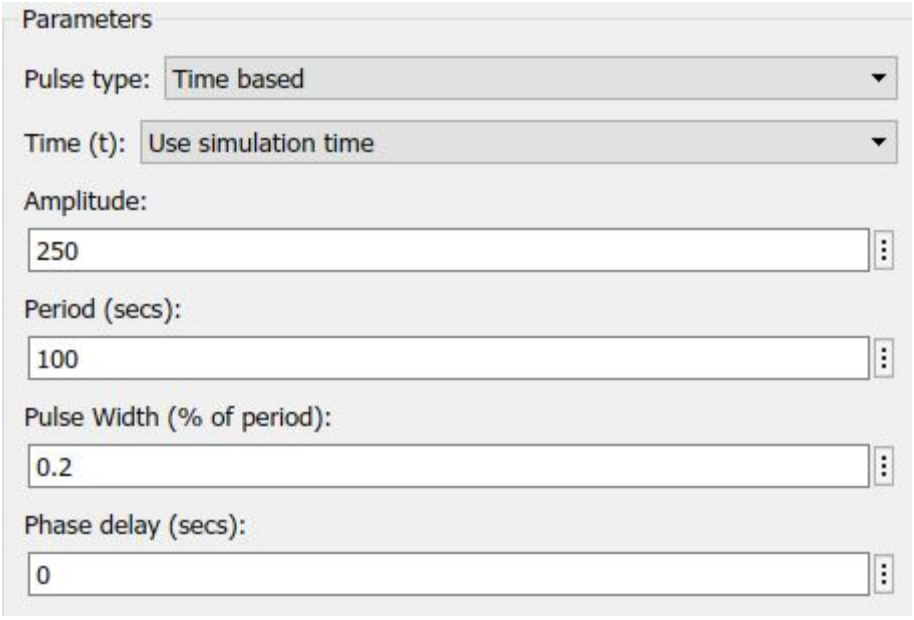


Figure 28: Parameter for Pulse Generator Block (Amplitude = 250, Period 100s, Pulse Width 0.2s)

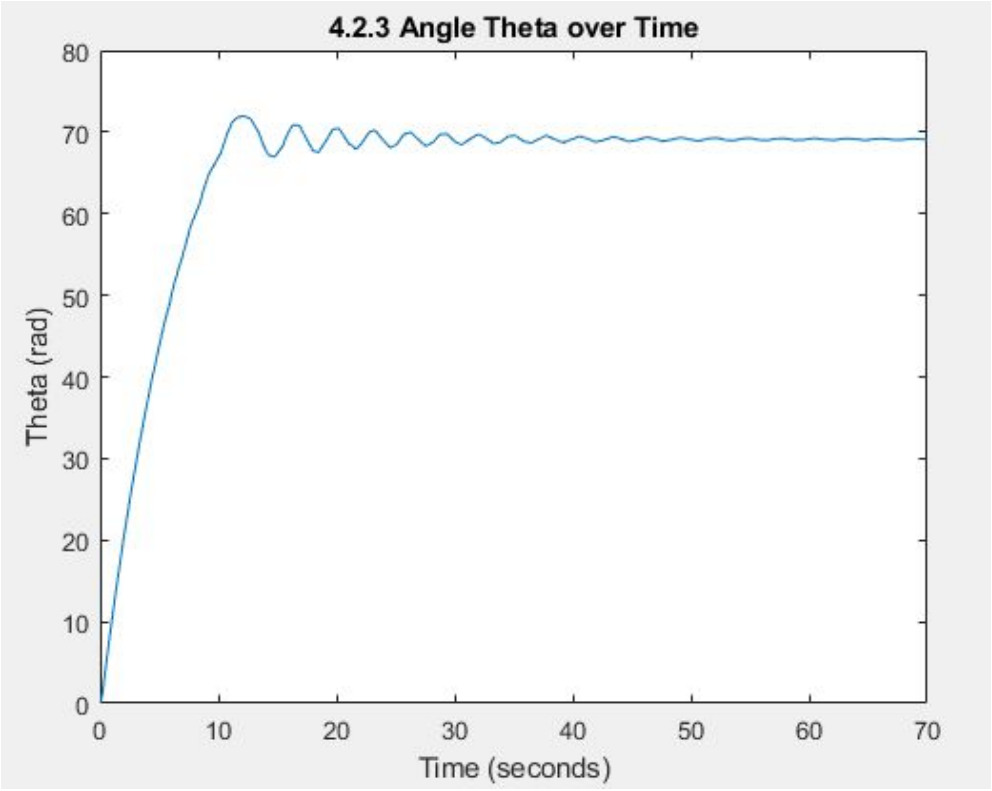


Figure 29: Nonlinear Pendulum Theta Response over Time to a single pulse having amplitude 250 and width 0.2 seconds

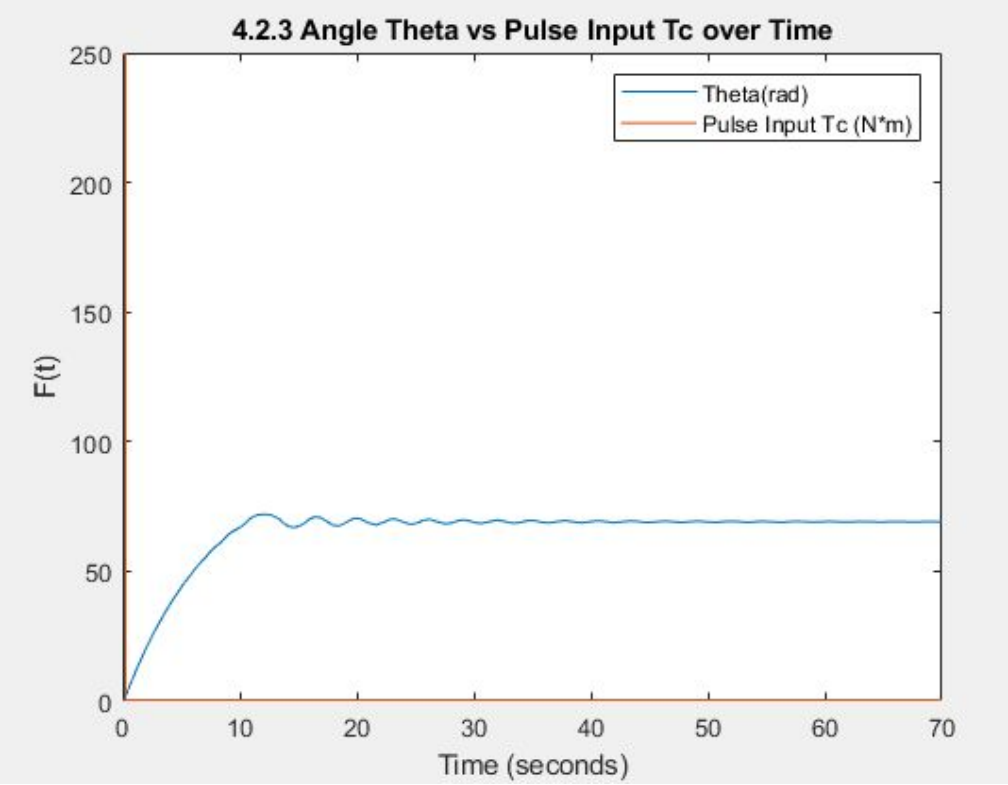


Figure 30: Nonlinear Pendulum Theta Response with Pulse Input over Time, where the Pulse input has amplitude 250 and width 0.2 seconds

To what new value does the pendulum angle converge? Explain why.

From Figure 29, the pendulum angle converges to around 70 rad. The reason behind this is that the force input is increased in amplitude from 10 N • m to 250 N • m, which is much larger as an initial input. Instead of the pendulum converging around an angle of 0 rad, it will converge around 70 rad. This is possibly caused by the steady state error which the pendulum is smacked with a large and the system overshoots too much.

4. Replace the nonlinear term in your Simulink model with the linear version. Repeat the experiment for the pulse with amplitude 10 and width 0.2 seconds. Do the nonlinear and linear versions agree for the angle response?

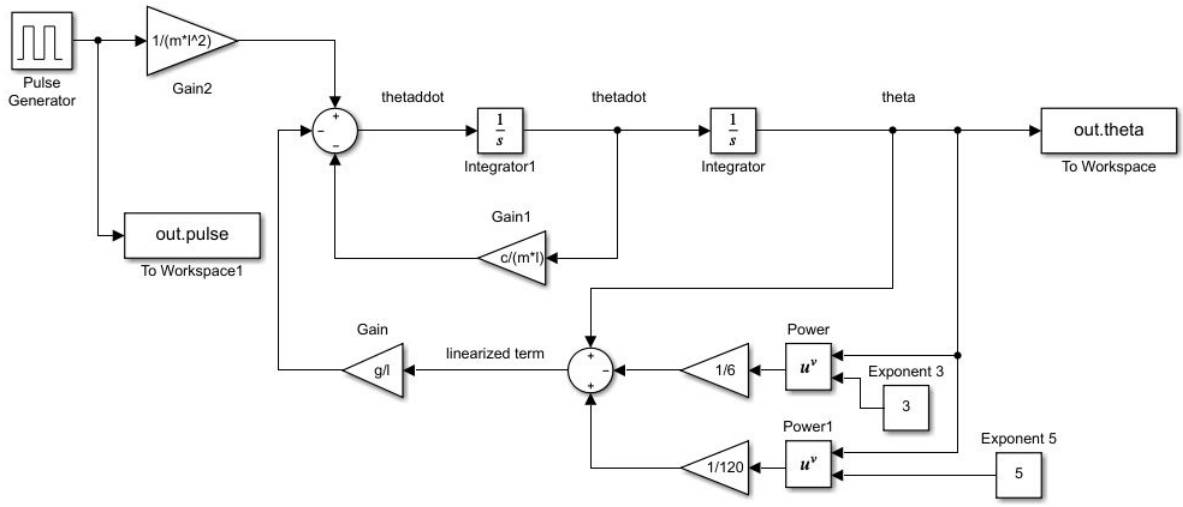


Figure 31: Simulink block diagram model of the linearized nonlinear damped pendulum

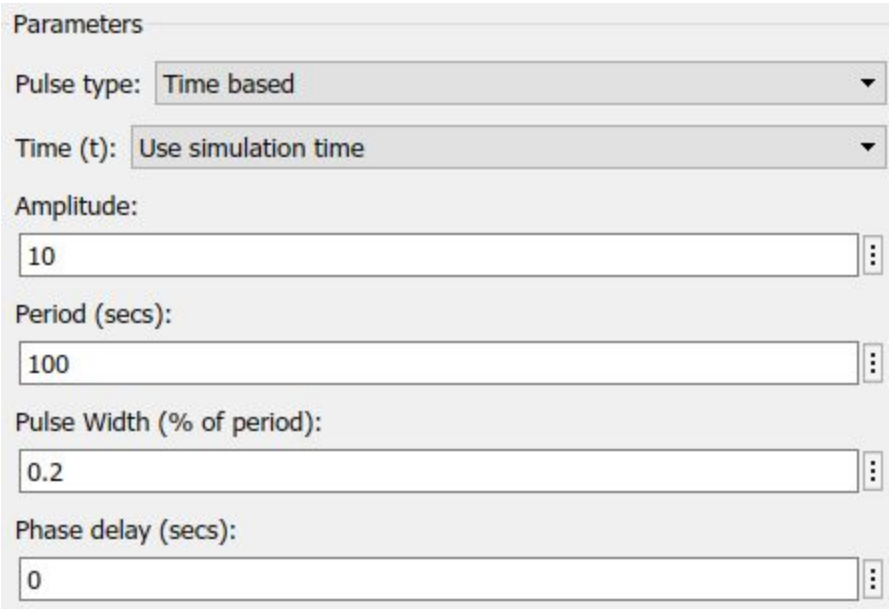


Figure 32: Parameters for Pulse Generator Block (Amplitude = 10, Period 100s, Pulse Width 0.2s)

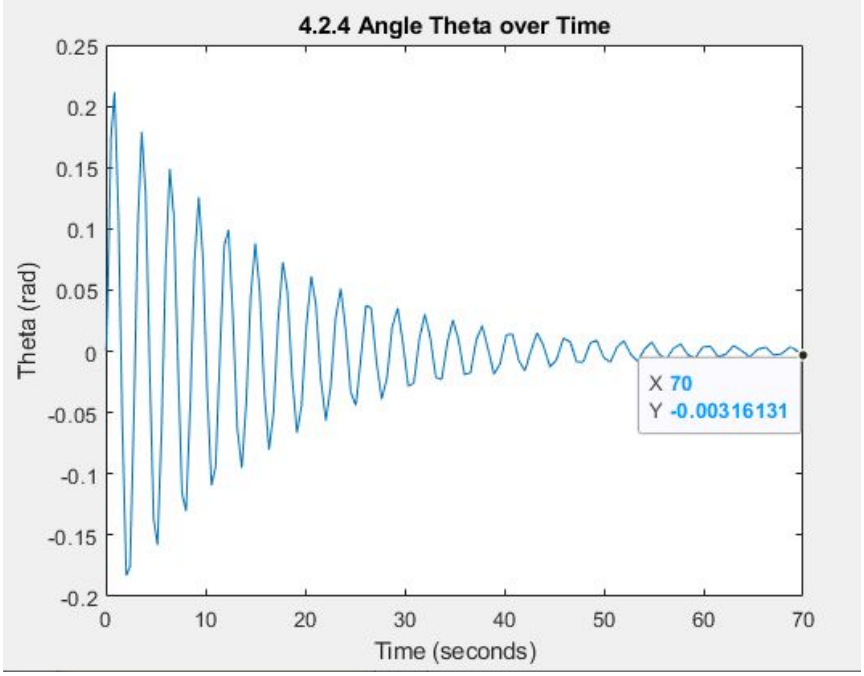


Figure 33: Linearized Nonlinear Pendulum Theta Response over Time to a single pulse having amplitude 10 and width 0.2 seconds

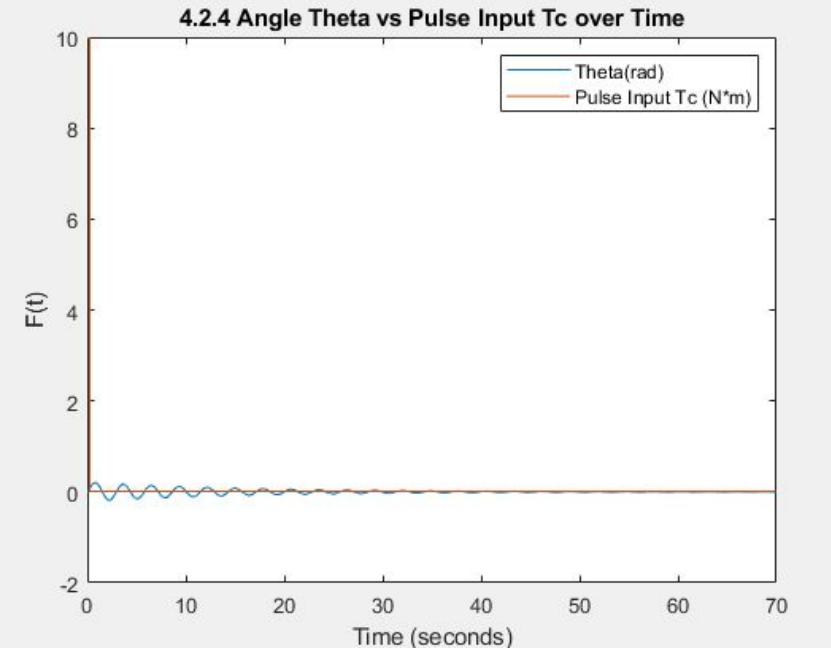


Figure 34: Linearized Nonlinear Pendulum Theta Response with Pulse Input over Time, where the Pulse input has amplitude 10 and width 0.2 seconds

Do the nonlinear and linear versions agree for the angle response?

In comparing Figure 19 & Figure 26, it can be seen that they both converge to 0 rad. Since the graphs obtained from both nonlinear and the linear model are very close, and this means that the nonlinear and linear versions agree for the angle response.

5. Now try the linear version with the pulse having amplitude 250 and width 0.2 seconds. Do the nonlinear and linear versions agree for the angle response? Why is there a discrepancy?

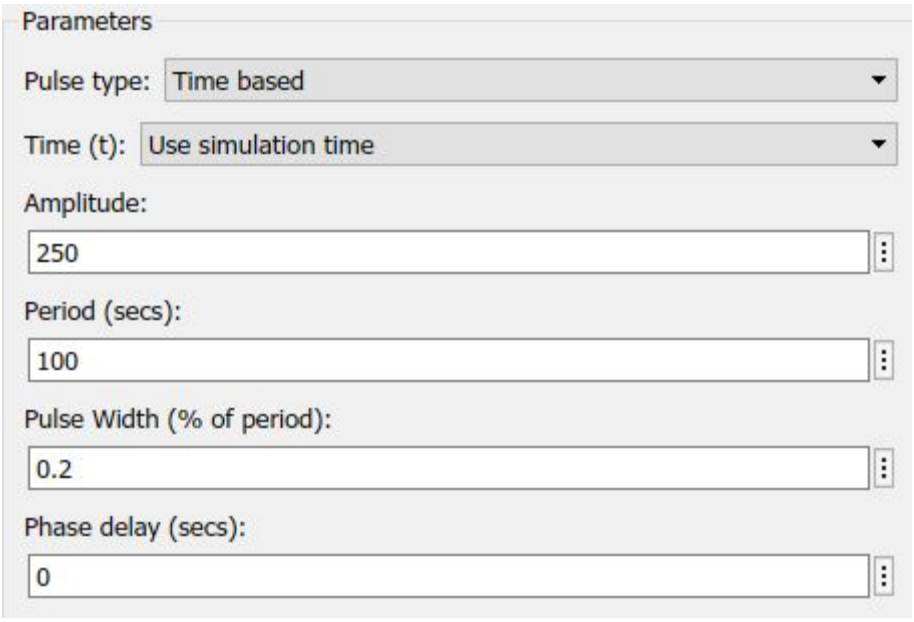


Figure 35: Parameters for Pulse Generator Block (Amplitude = 250, Period 100s, Pulse Width 0.2s)

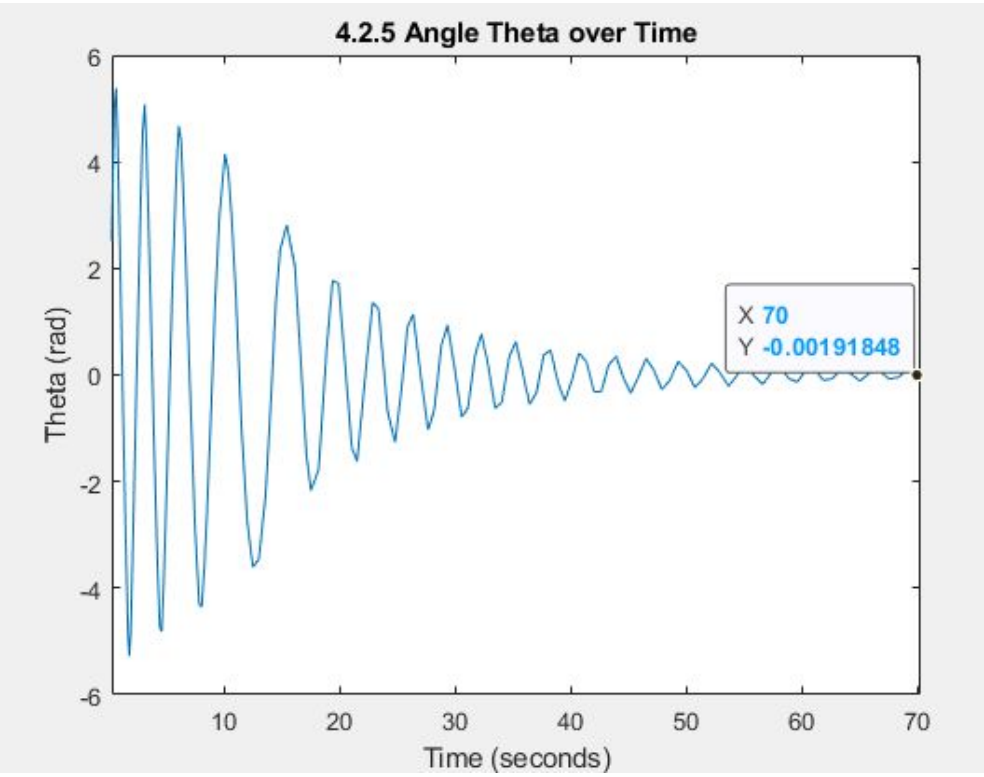


Figure 36: Linearized Nonlinear Pendulum Theta Response over Time to a single pulse having amplitude 250 and width 0.2 seconds

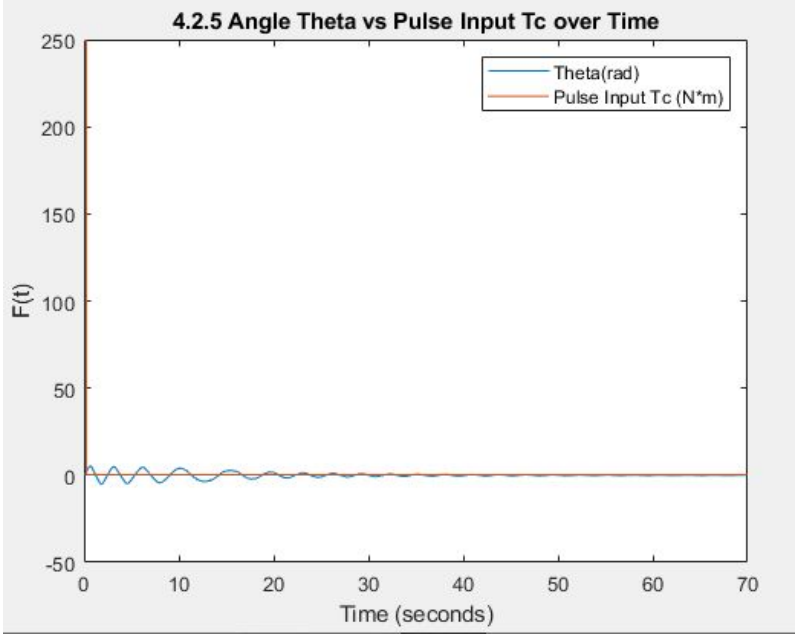


Figure 37: Linearized Nonlinear Pendulum Theta Response with Pulse Input over Time, where the Pulse input has amplitude 250 and width 0.2 seconds

Do the nonlinear and linear versions agree for the angle response? Why is there a discrepancy?

The nonlinear and linear versions do not agree for the angle response as the nonlinear version convergerges to 70 rad while the linear version converges to 0 rad. In the linearization of the nonlinear term $\frac{g}{l}sin\theta$, the Taylor series expansion around $\theta = 0$ for the nonlinearity is found. There is a discrepancy possibly because taylor series expansion no longer holds with a massive input of 250 because a taylor series is valid only within a restricted domain. In general, small oscillations in any nonlinear oscillator can be approximately treated in terms of simple harmonic motion, but large input follows with great oscillations and hence linearizing the nonlinear oscillator using the Taylor series expansion is not possible. This can possibly explain why nonlinear and linear versions no longer agree for the angle response here.

Based on your results from parts 4 and 5, when is the linearized system a “good” approximation of the original system?

The linearized system is a “good” approximation when the input to the system is small (i.e. for the pulse generator we use an amplitude that is 10 as opposed to an amplitude of 250). This can be seen in comparing Figure 23 and Figure 30 which have the same pulse input (Amp =250 N*m and width = 0.2s) where the theta response for the nonlinear version converges to 70 rad, while the theta response for the linear version converges to 0 rad. Since the output from the linearized system is an approximation of the nonlinear system, this means that when the linearized system experiences higher perturbations it will not be a good approximation.