Lab Assignment 3: Introduction to Simulink

Theodore Janson

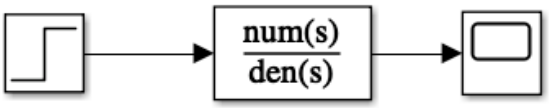
260868223

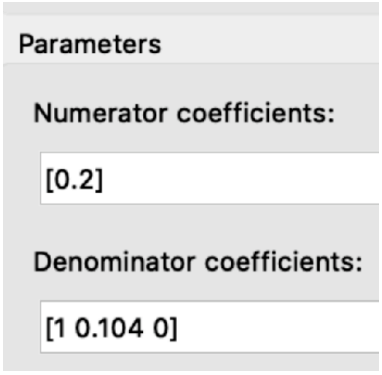
ECSE 403

October 12, 2020

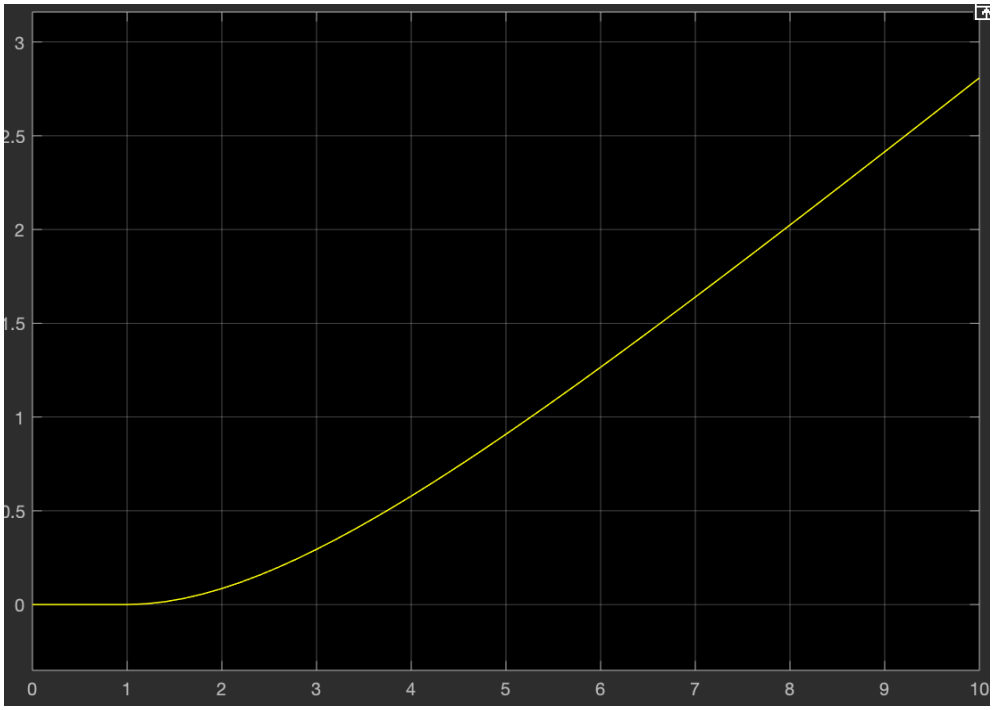
3.1 Simulink

1. Implement the transfer function you derived in Lab 1 θ(s)/V(s) in Simulink.



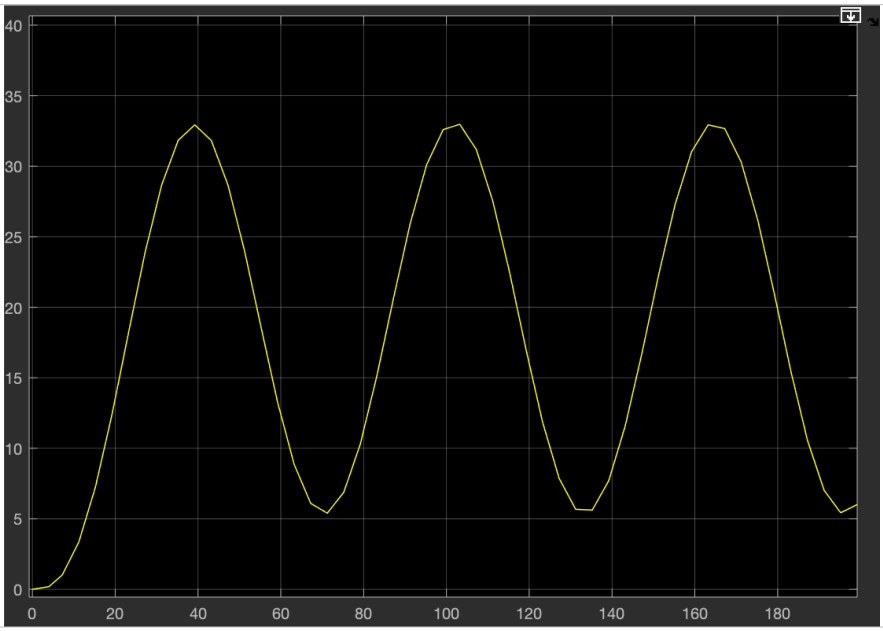


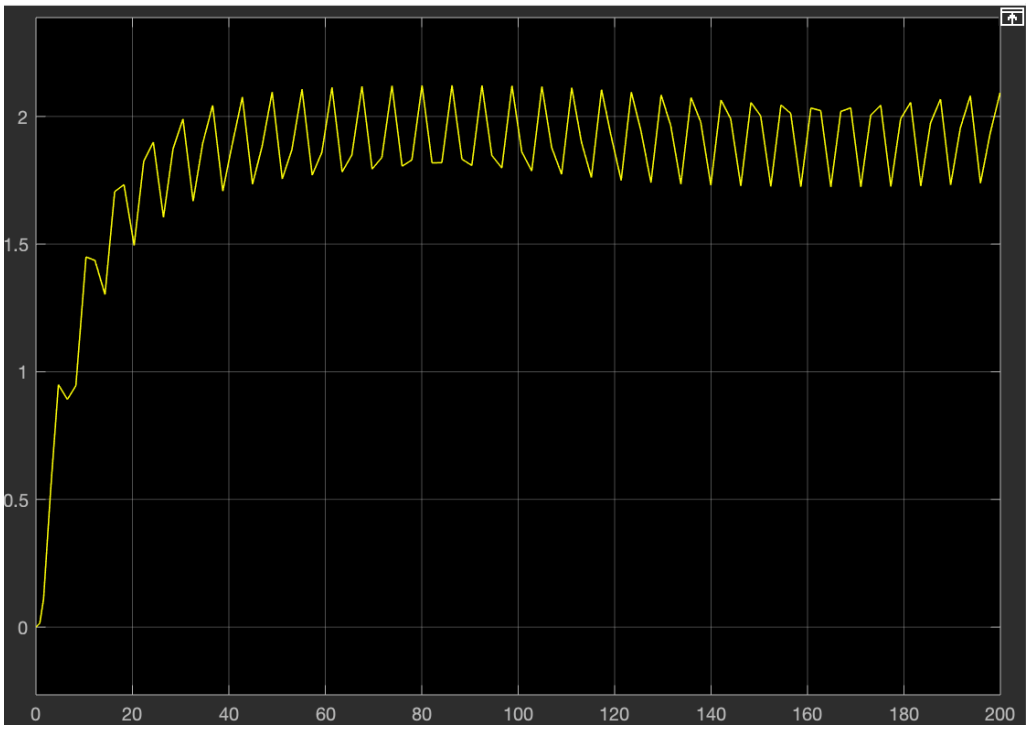
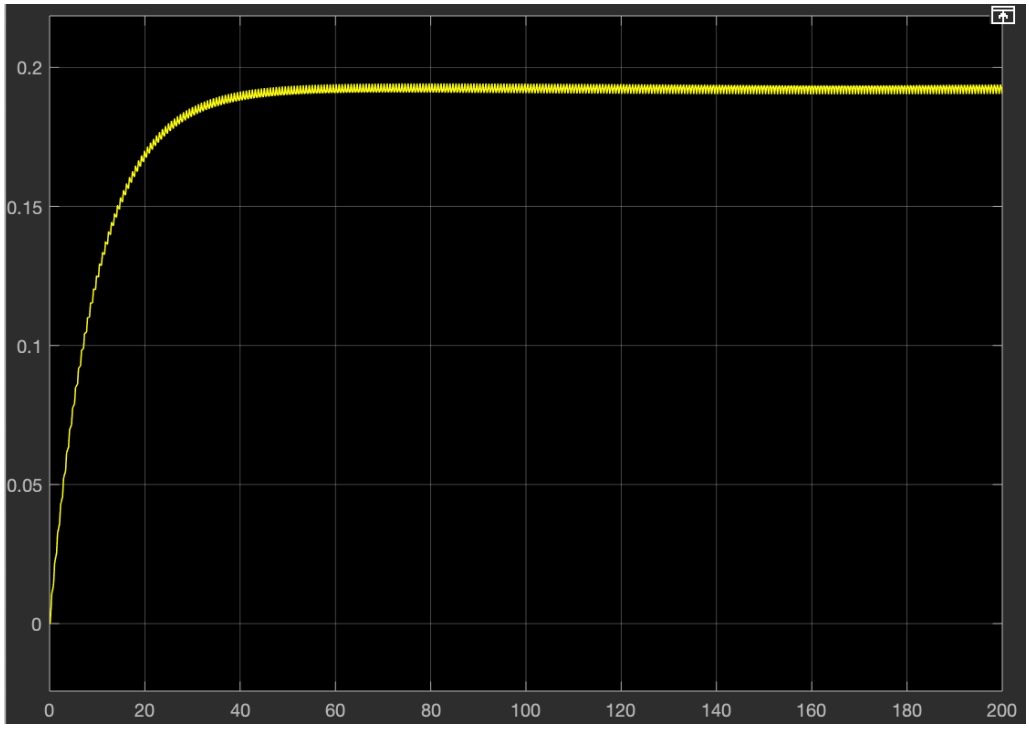
2. Using Step, find the step response of the system. In the case of existence of these numbers find the rise time and steady state of the system. [5 marks]

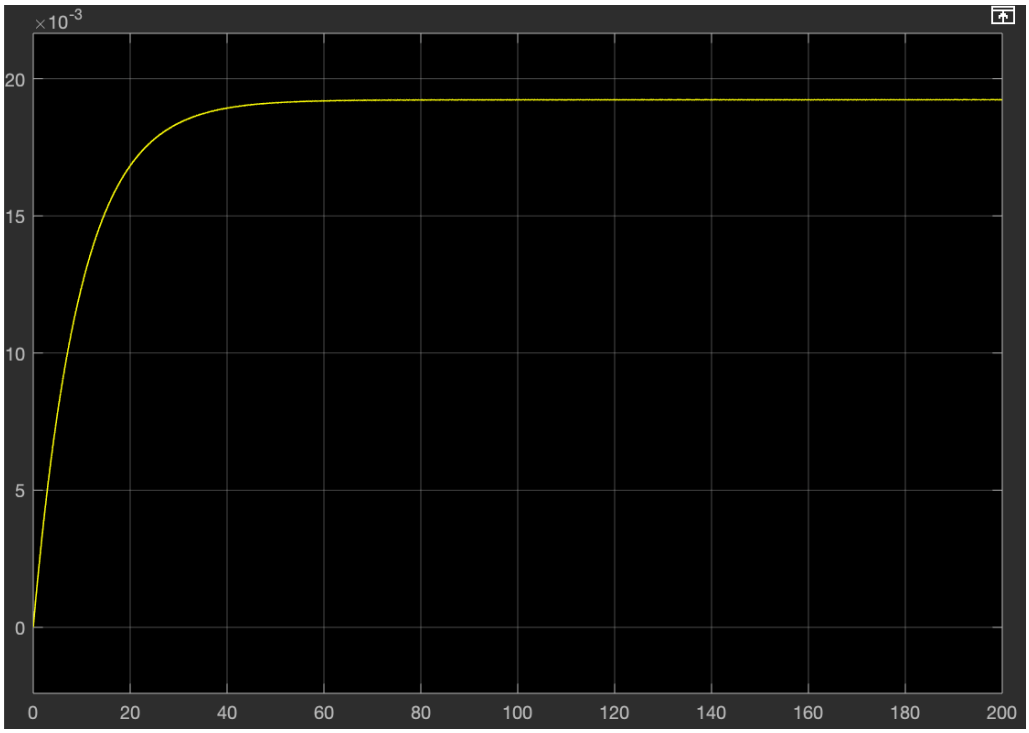
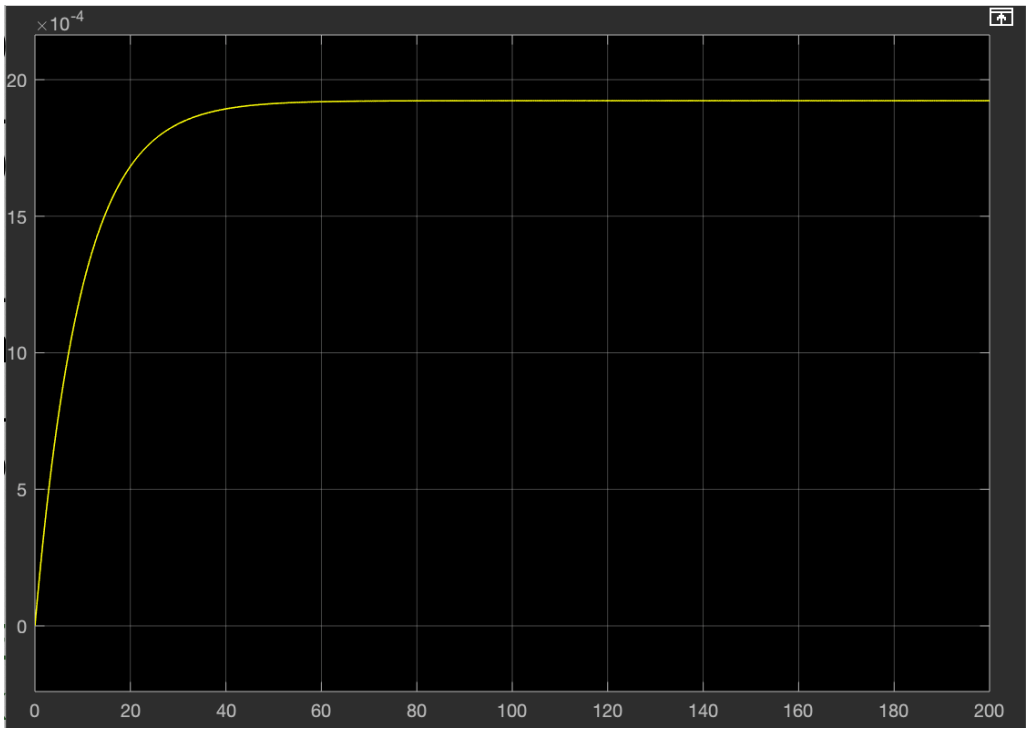


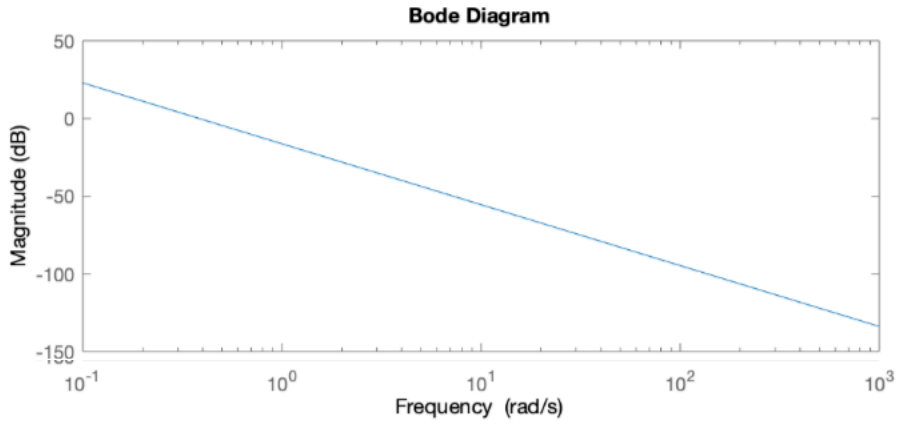
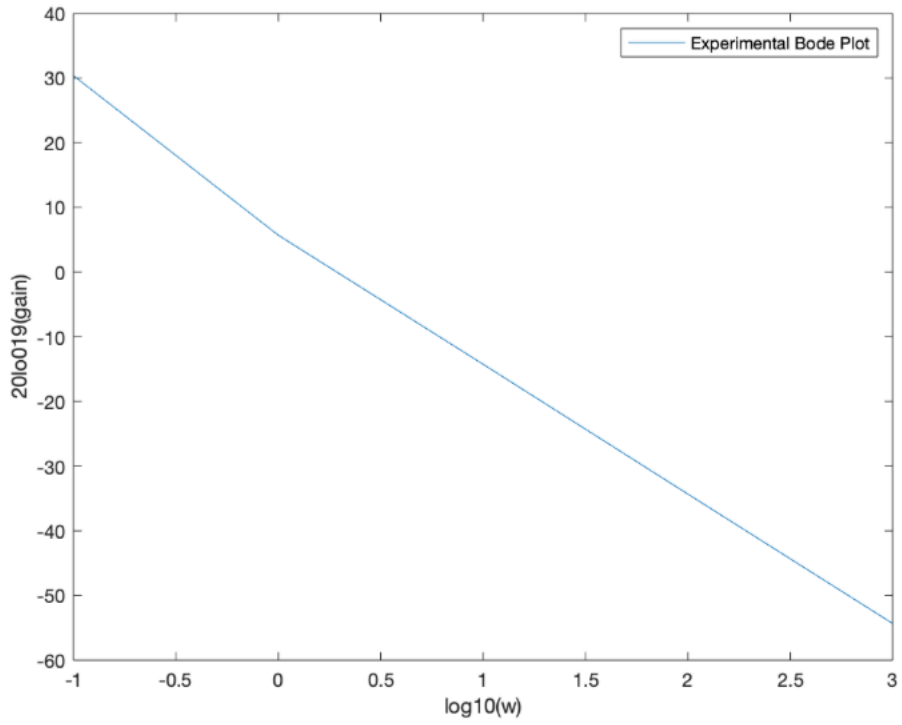
- There is no system steady state or a finite rise time.

3. Using Signal Generator or Sine Wave blocks, measure response of the system to sine waves of amplitude 1 and frequencies w = [0.1, 1, 10, 100, 1000].
Using these data, plot (approximate) Bode diagram(gain diagram) of the system.
Using the transfer function and bode command in the Matlab environment, plot the theoretical Bode diagram and then compare it with the diagram you found by experiments. [10 marks]

w	Response
0.1	

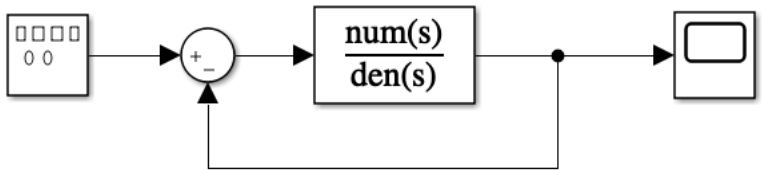
1	 2 1.5 1 0.5 0 0 20 40 60 80 100 120 140 160 180 200
10	 0.2 0.15 0.1 0.05 0 0 20 40 60 80 100 120 140 160 180 200

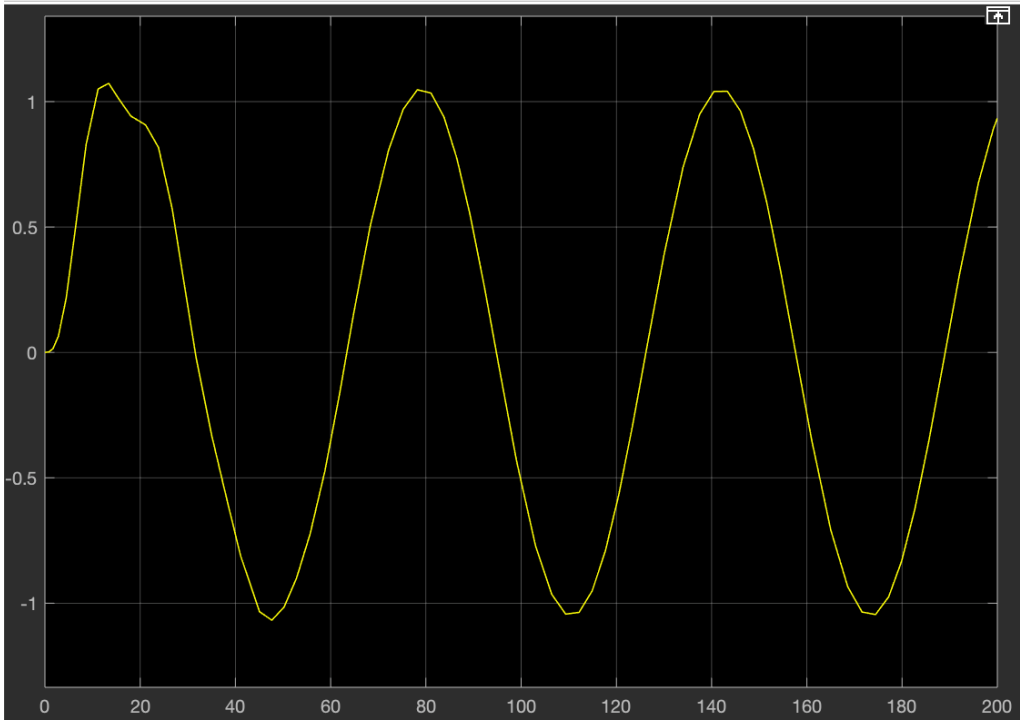
100	 ×10^-3 20 15 10 5 0 0 20 40 60 80 100 120 140 160 180 200
1000	 ×10^-4 20 15 10 5 0 0 20 40 60 80 100 120 140 160 180 200

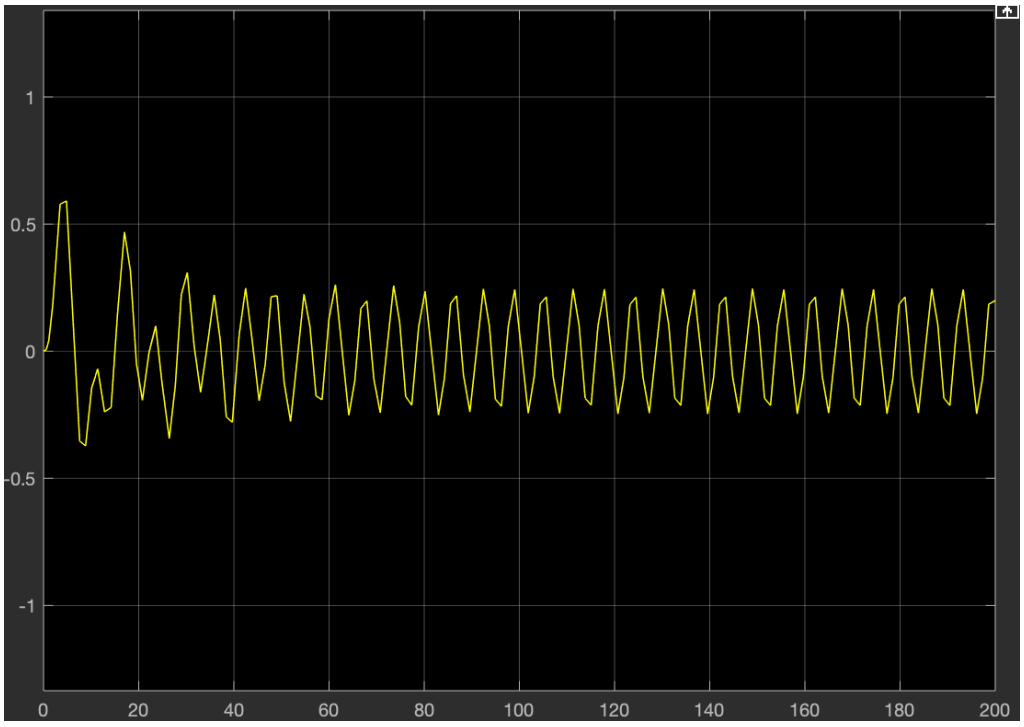
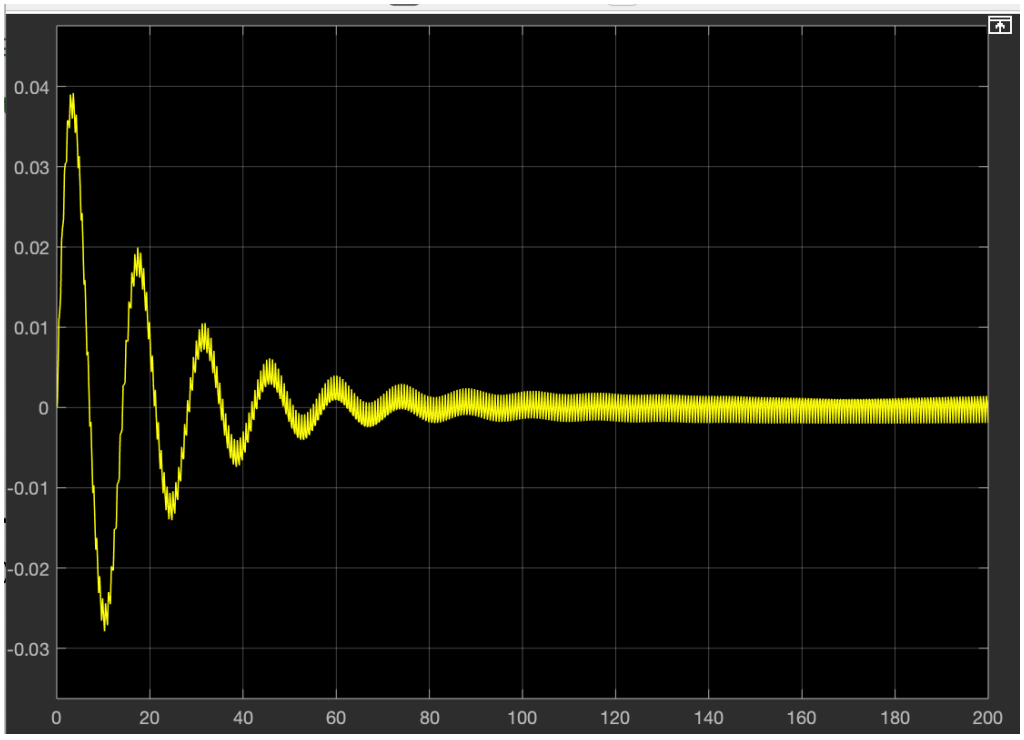
Theoretical	
Experimental	

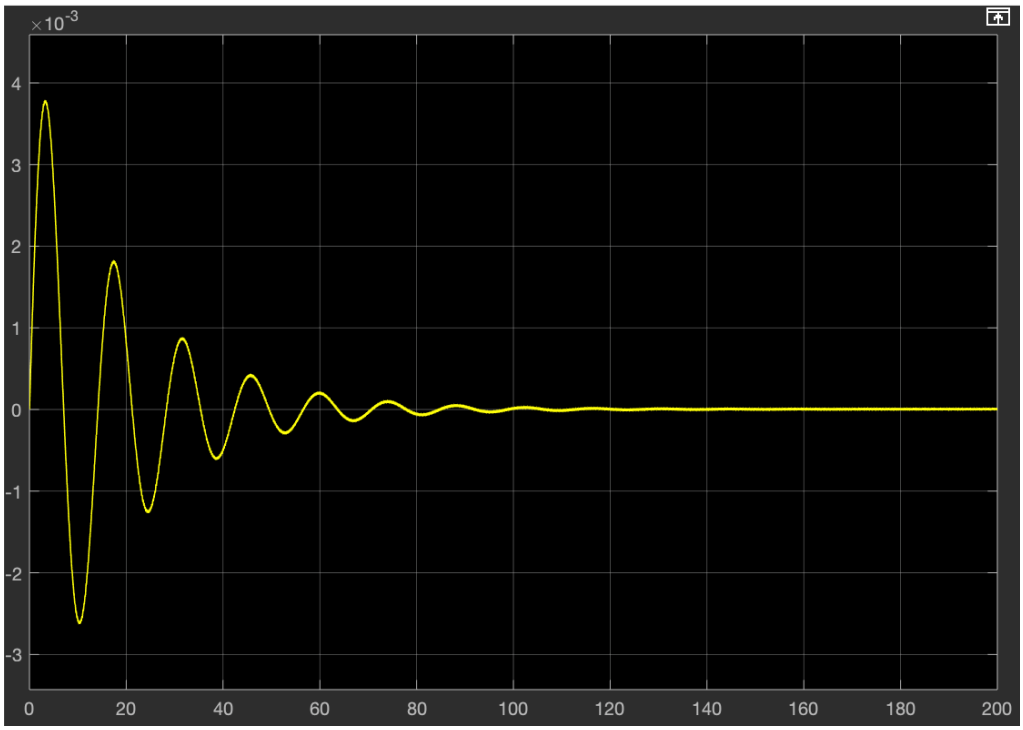
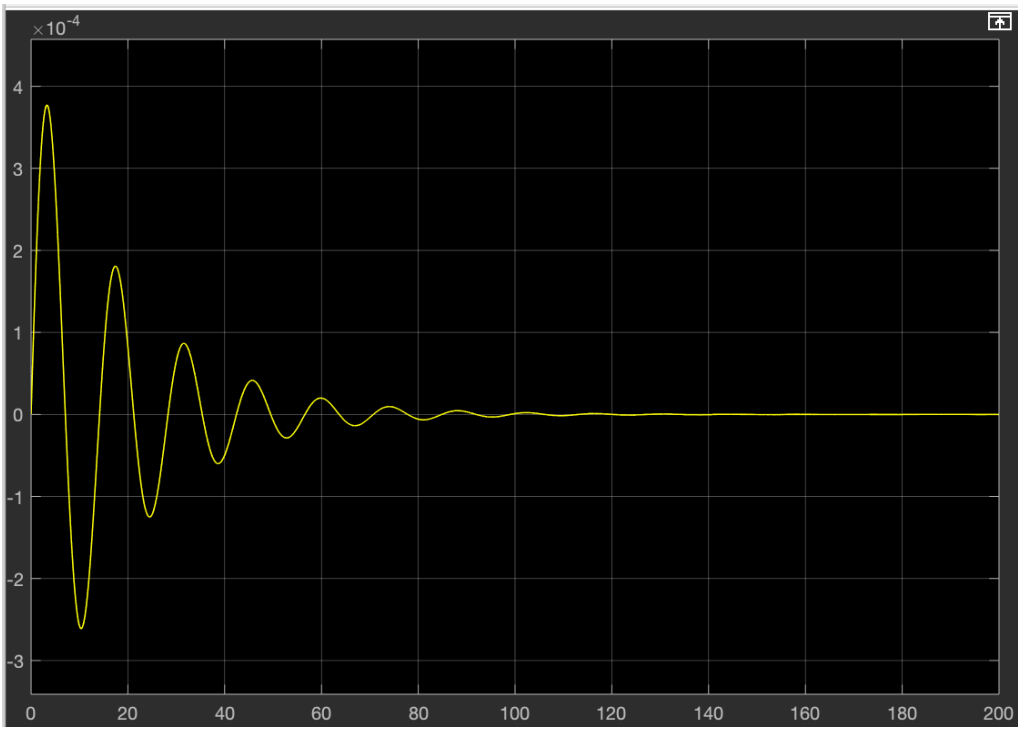
The experimental bode plot decays by approximately -20dB / decade, whereas the theoretical blade plot days by twice as much at around -40 dB/ decade. The theoretical plot has an initial magnitude of around 22dB at w=0.1, whereas the experimental plot has an initial value of 30dB. The final value of the theoretical plot at w=1000 is around -130dB, whereas that of the experimental plot is around -54dB.

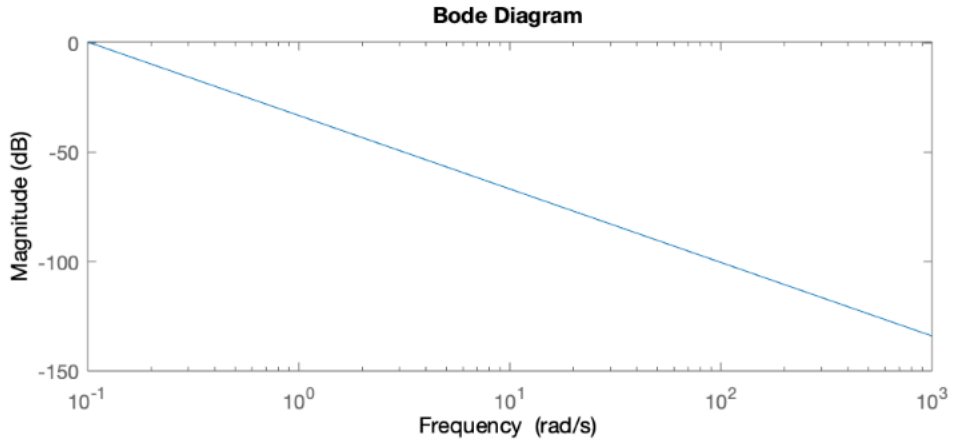
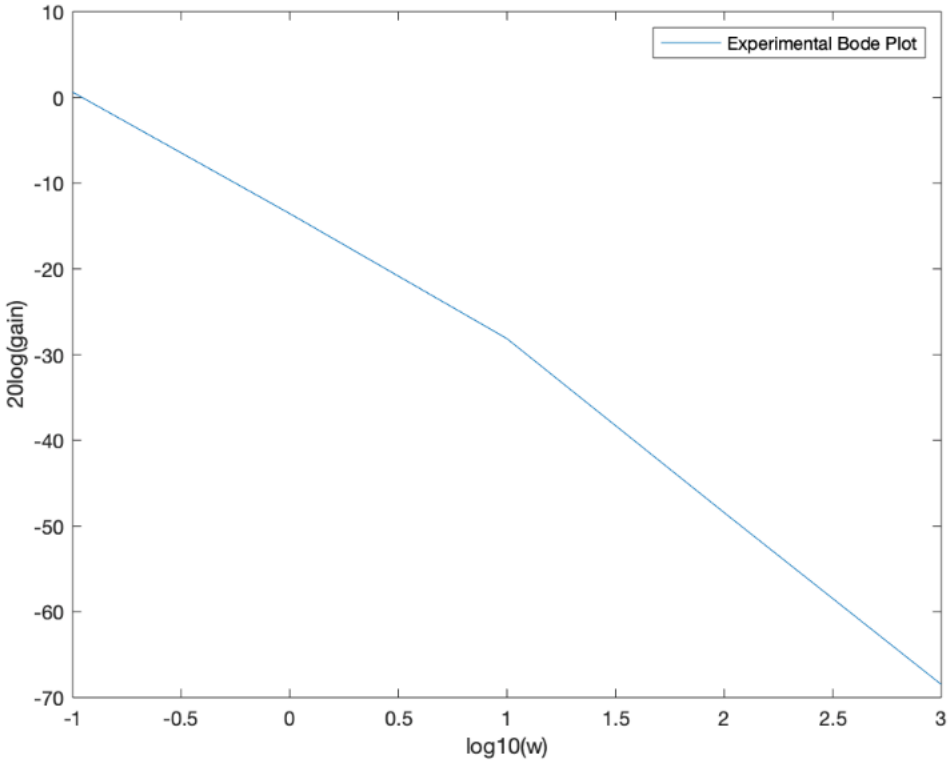
4. Implement the unity Feedback loop in Simulink. Repeat the steps of Q3 to derive the experimental and theoretical Bode diagram of the closed loop system and compare the diagrams in the MATLAB. [10 marks]



W	Response
0.1	

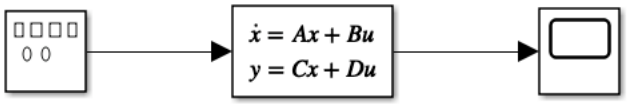
1	 1 0.5 0 -0.5 -1 0 20 40 60 80 100 120 140 160 180 200
10	 0.04 0.03 0.02 0.01 0 -0.01 -0.02 -0.03 0 20 40 60 80 100 120 140 160 180 200

100	 ×10^-3 4 3 2 1 0 -1 -2 -3 0 20 40 60 80 100 120 140 160 180 200
1000	 ×10^-4 4 3 2 1 0 -1 -2 -3 0 20 40 60 80 100 120 140 160 180 200

Theoretical	
Experimental	

The experimental bode plot decays by approximately -20dB / decade, whereas the theoretical blade plot days by twice as much at around -40 dB/ decade. Both the theoretical plot and the experimental plot have initial magnitudes of 0 dB at w=0.1. The theoretical plot has a final value of around -130 dB, whereas the experimental plot's final value is around half that.

5. Using state-space from continuous library, implement A_2, B_2, C_2 model from Lab2. Repeat steps of question 2 to compare experimental Bode diagram with theoretical one of the accu- rate model of the system. Are the experimental and theoretical diagrams similar? [10 marks]



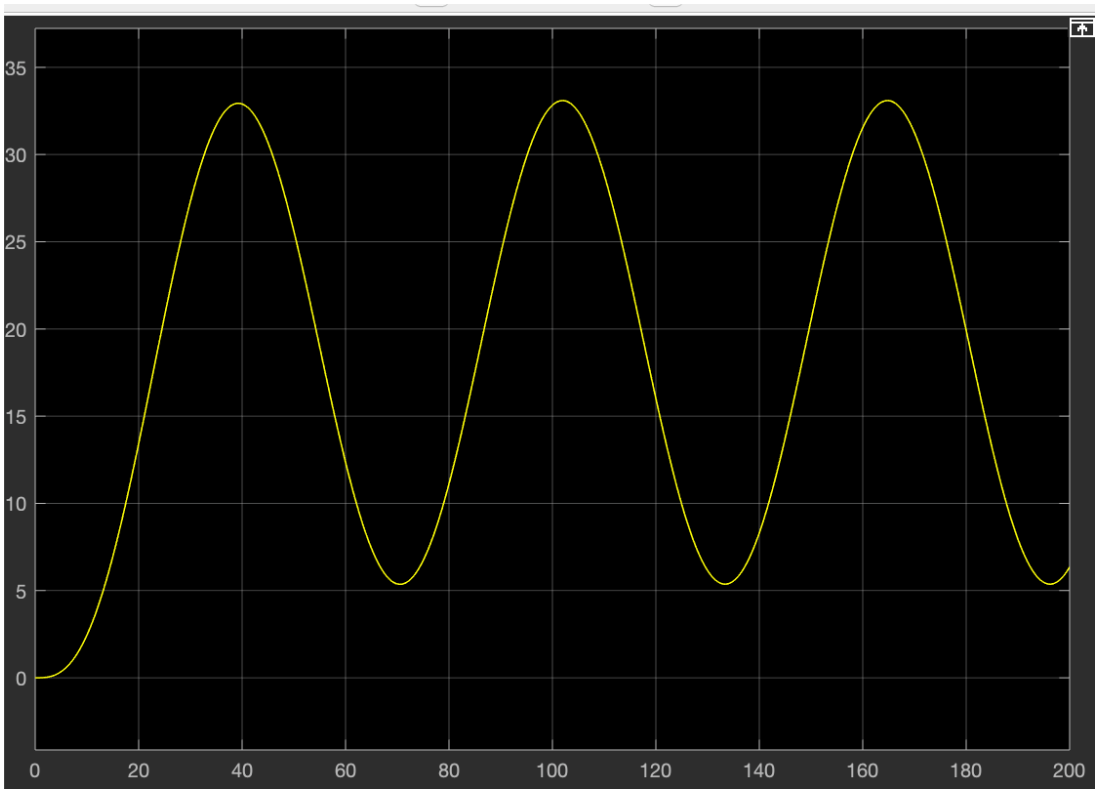
Parameters

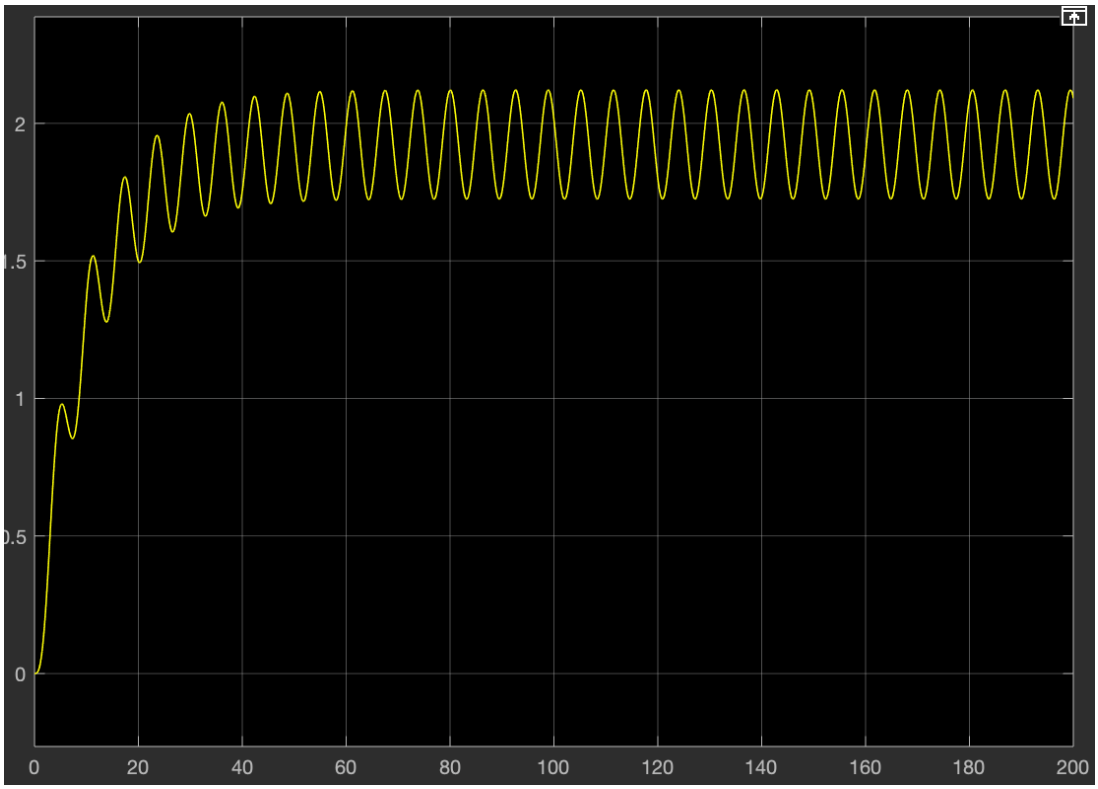
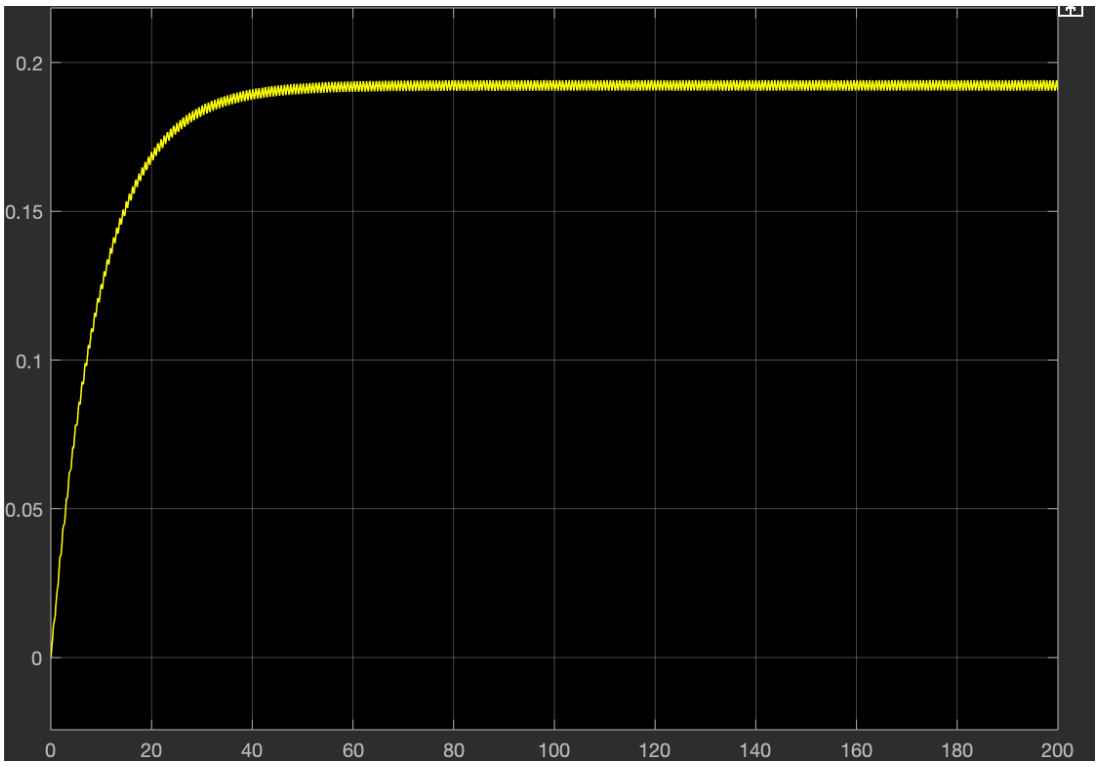
A:
[0 1 0 ; 0 -0.1 2 ; 0 -0.04 -20]

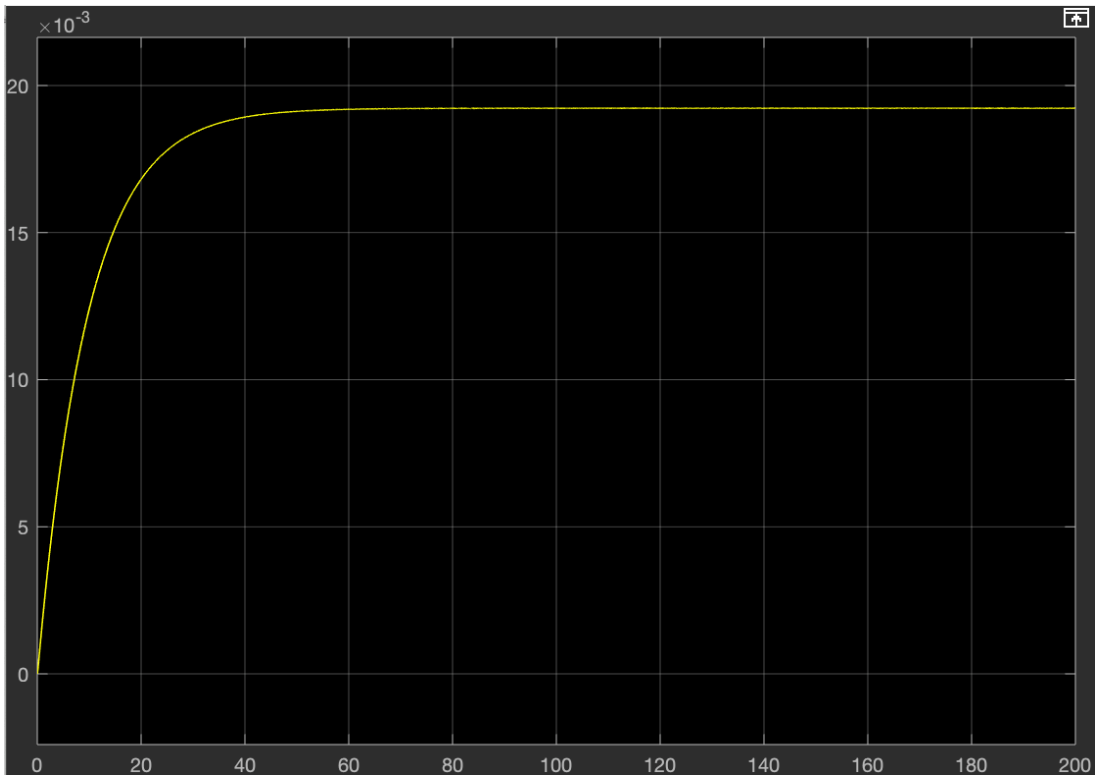
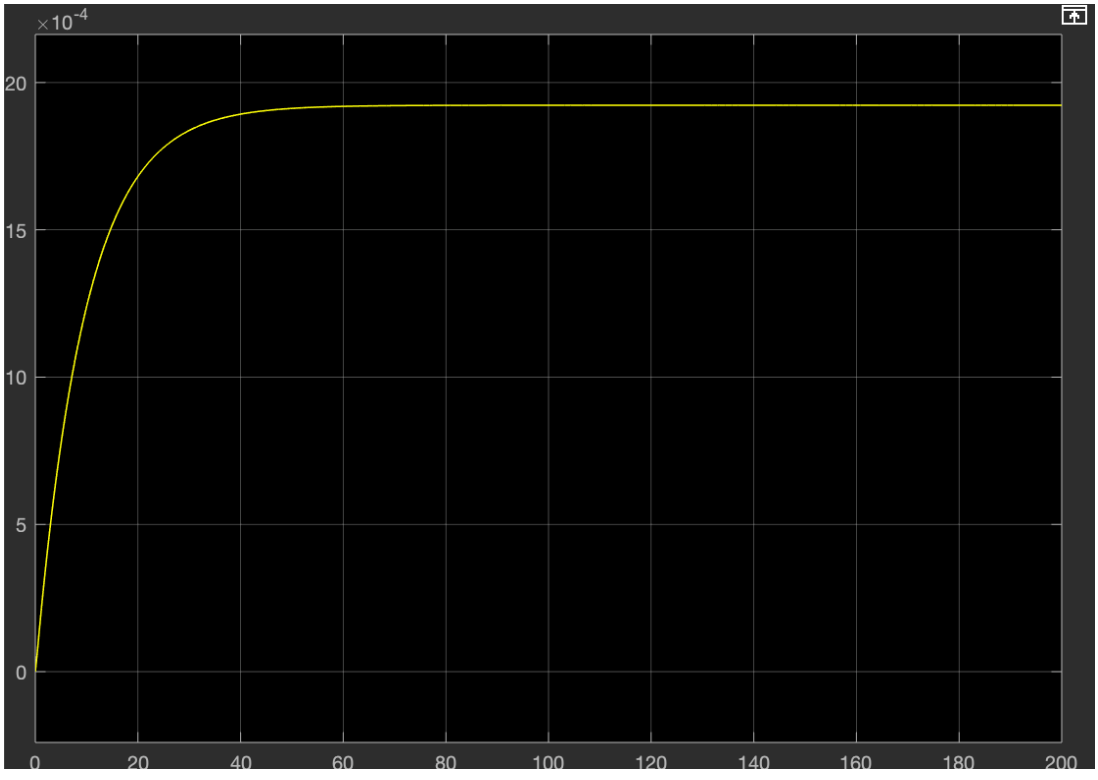
B:
[0 ; 0 ; 2]

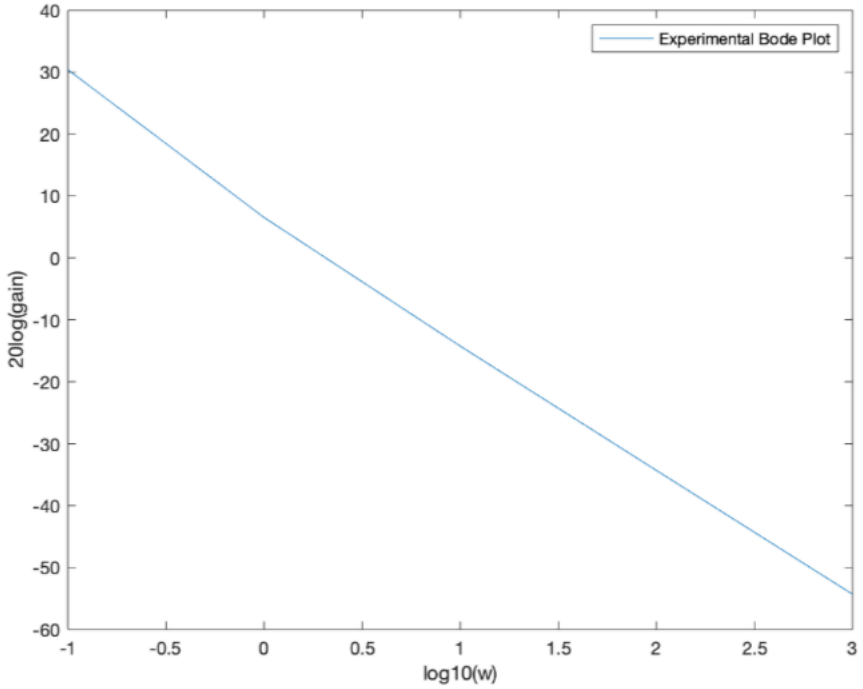
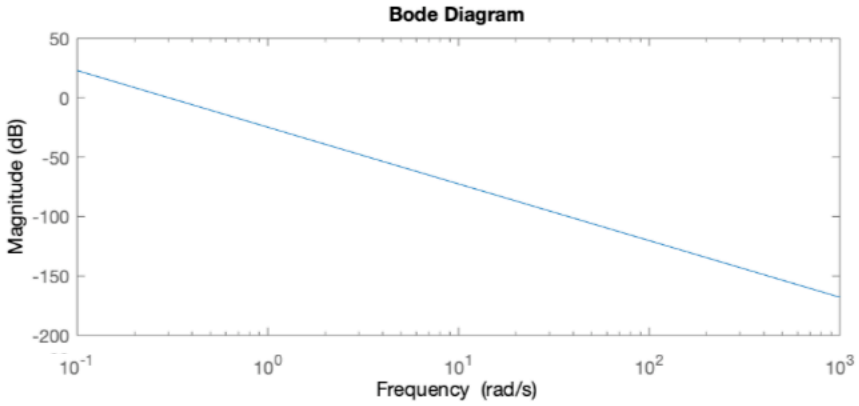
C:
[1 0 0]

D:
0

w	Response
0.1	

1	 2 1.5 1 0.5 0 0 20 40 60 80 100 120 140 160 180 200
10	 0.2 0.15 0.1 0.05 0 0 20 40 60 80 100 120 140 160 180 200

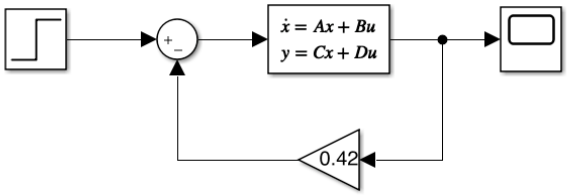
100	 ×10-3 20 15 10 5 0 0 20 40 60 80 100 120 140 160 180 200
1000	 ×10-4 20 15 10 5 0 0 20 40 60 80 100 120 140 160 180 200

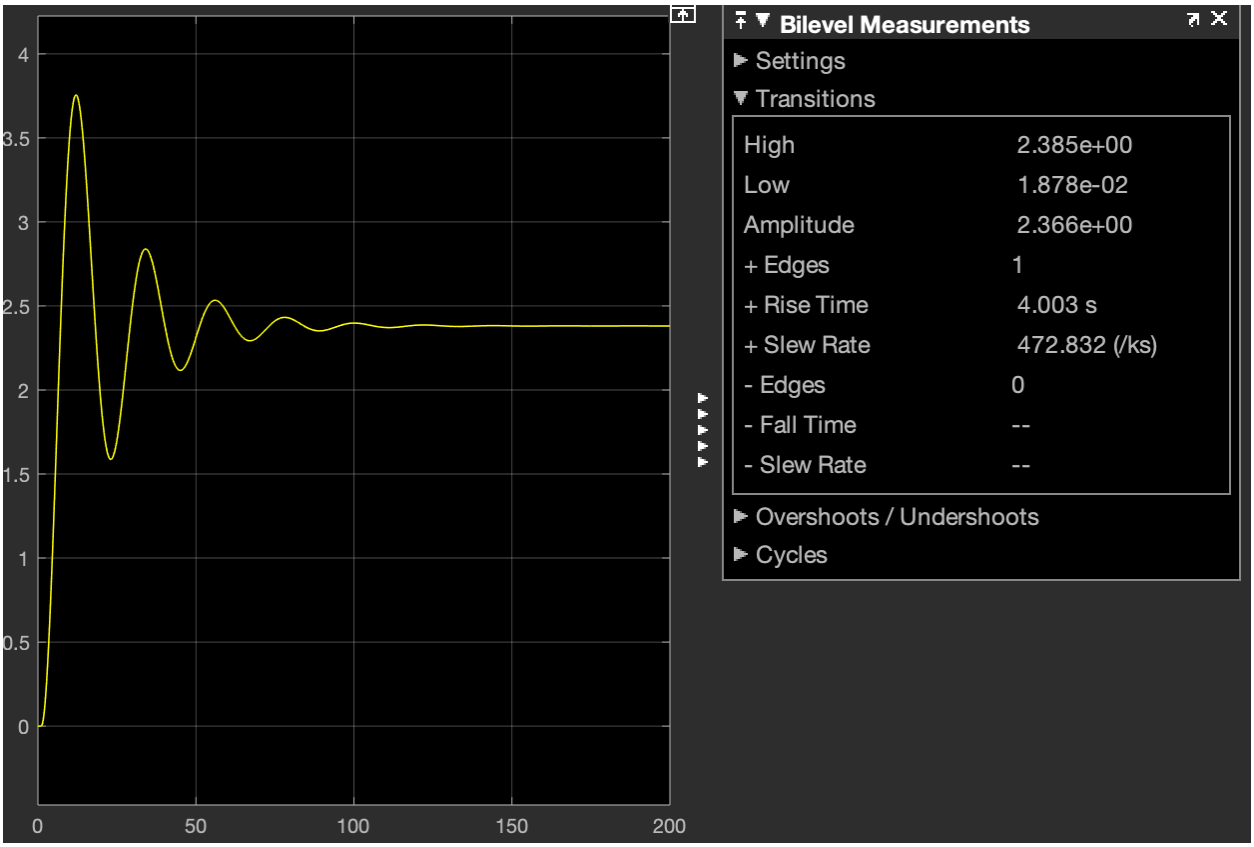
Experimental	
Theoretical	

The experimental bode plot decays by approximately -20dB / decade, whereas the theoretical blade plot days by twice as much at around -40 dB/ decade. The theoretical plot has an initial magnitude of around 22dB at w=0.1, whereas the experimental plot has an initial value of 30dB. The final value of the theoretical plot at w=1000 is around -130dB, whereas that of the experimental plot is around -54dB.

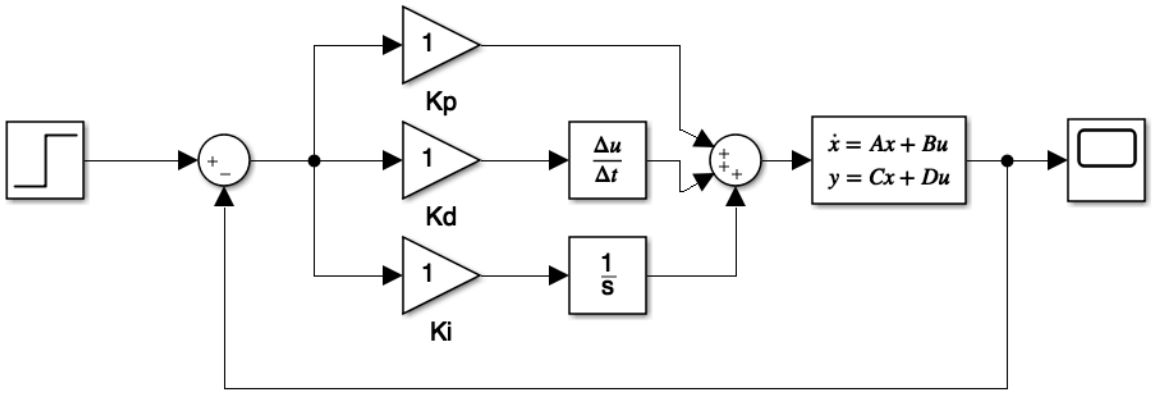
See Appendix A for Matlab code for bode plots

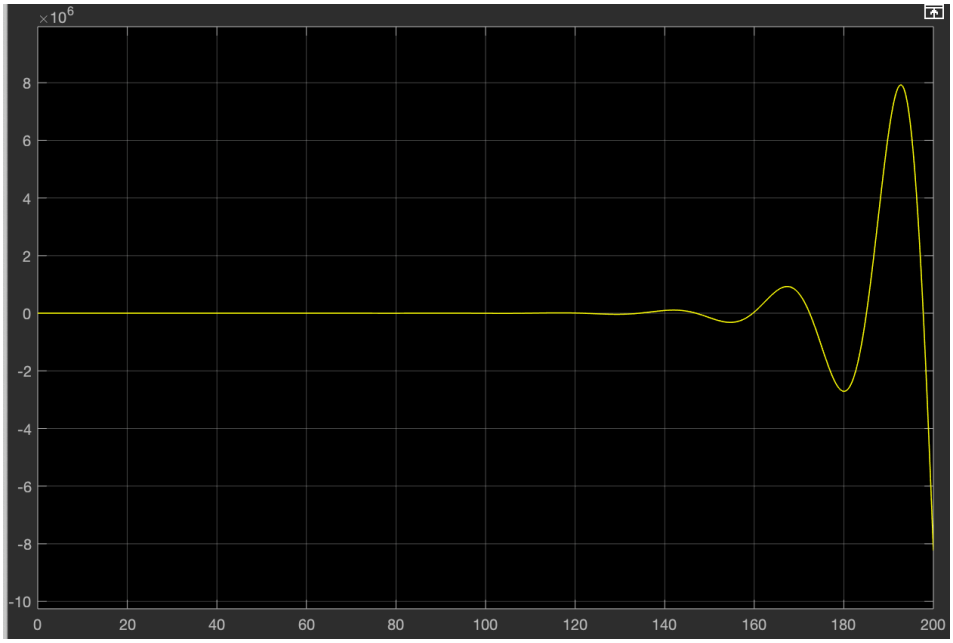
6. *Implement a proportional controller in Simulink for the system in previous question (the open loop system is a state space model with A_2, B_2, C_2). By changing the amount of proportional gain K, find the corresponding K for which we get rise-time of 4 seconds. [10 marks.*

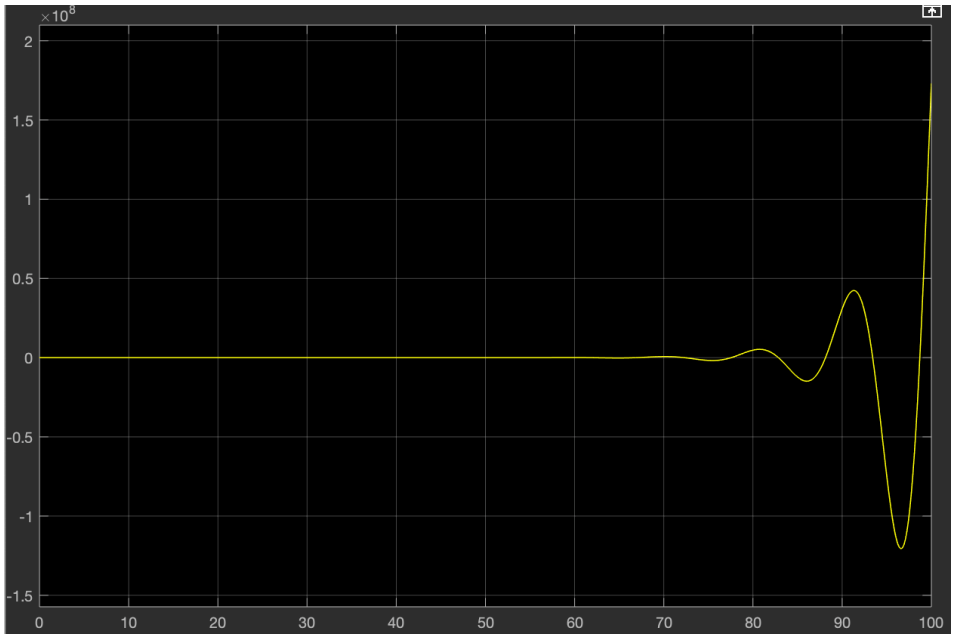
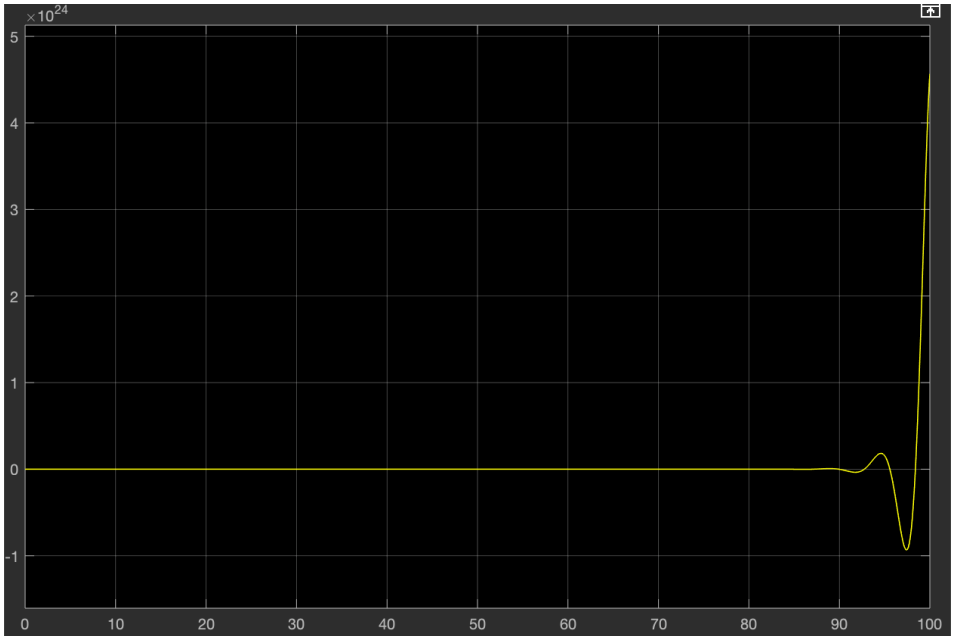


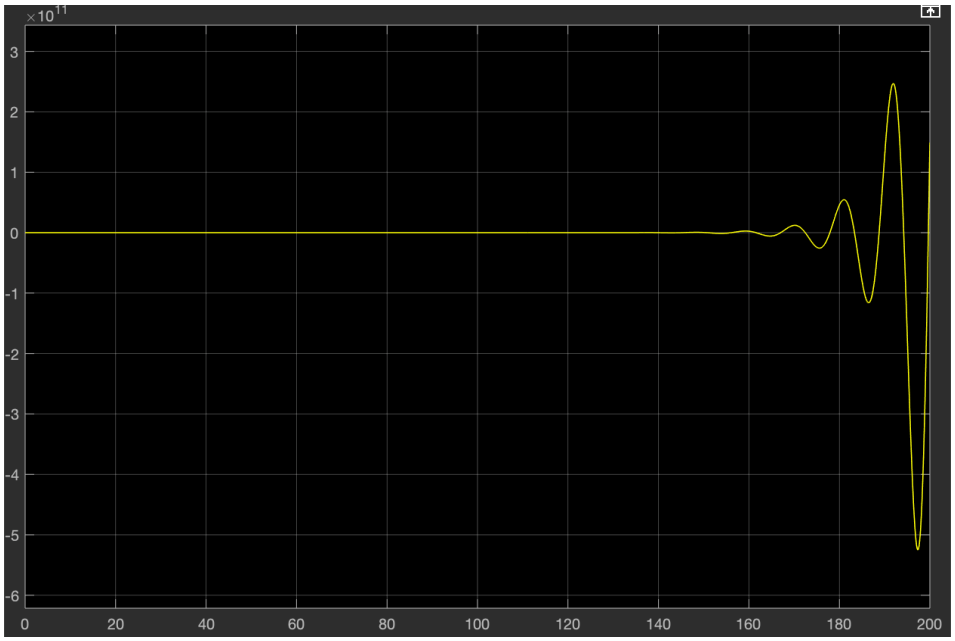
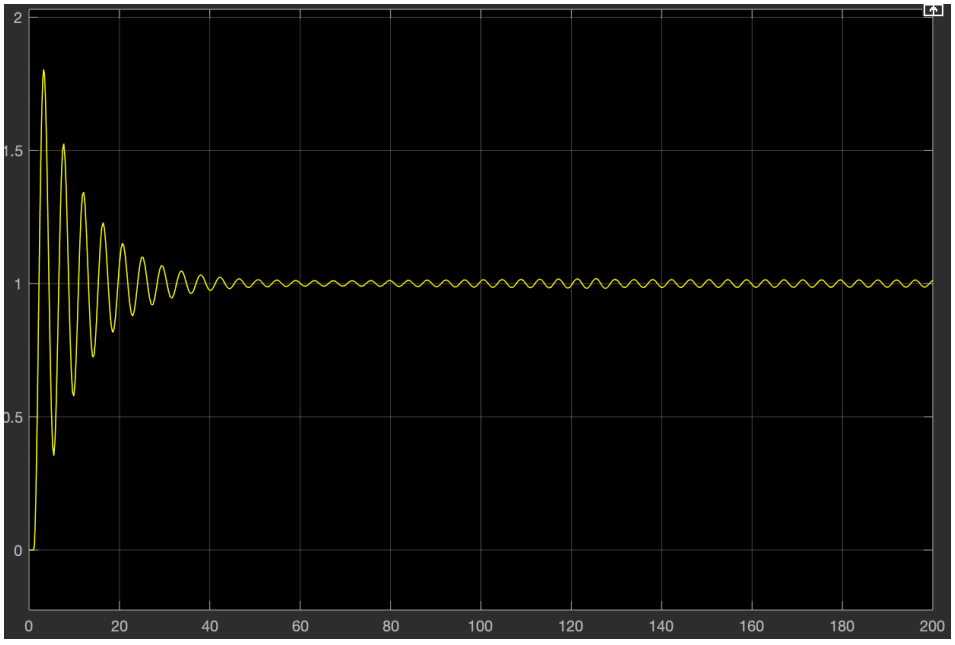


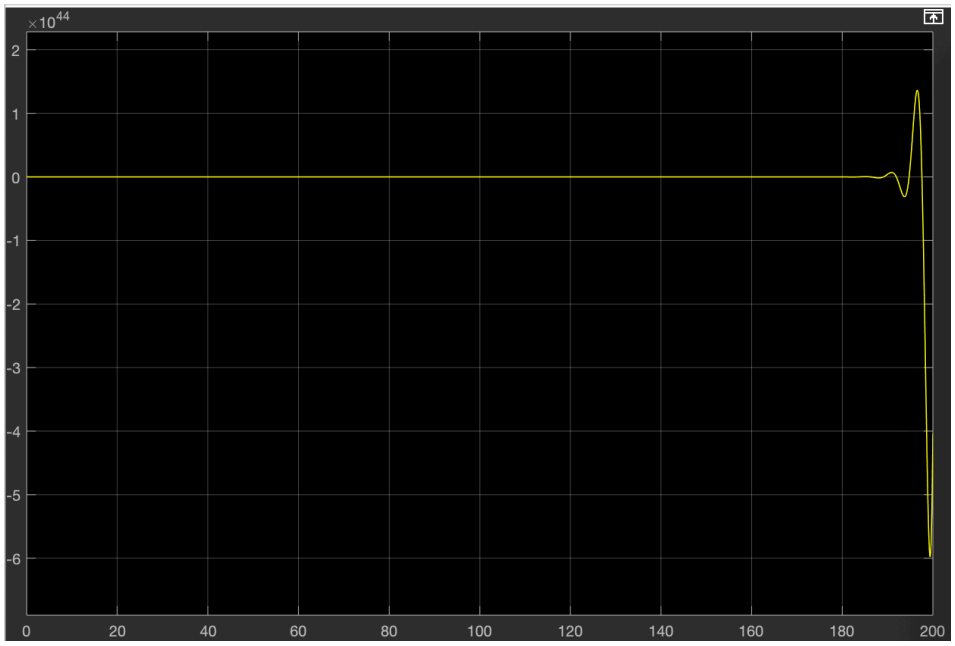
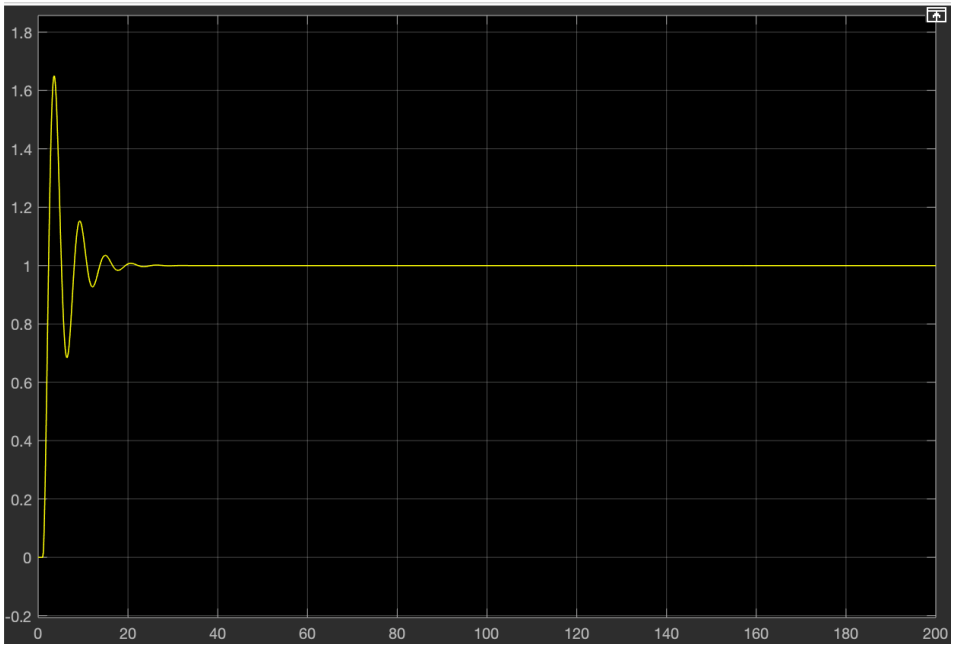
7. *Using a derivative block, and an integrator block, implement a PID controller for the sys- tem(these blocks are installed in parallel before the plant).*
Suppose we name the gain block before the derivative block K_d, and the gain block before the integrator block K_i, and proportional gain K_p, observe the output of the system to unit step function for different combinations of K_i, K_d, $K_p \in \{0.1, 1, 10\}$(by different combinations we mean fixing two of the gains and iterating over the last one). [15 marks]
Using the above experiment describe intuitively the impact of increment in K_d, K_i(for K_p = 1), on the step response of the system. [15 marks]

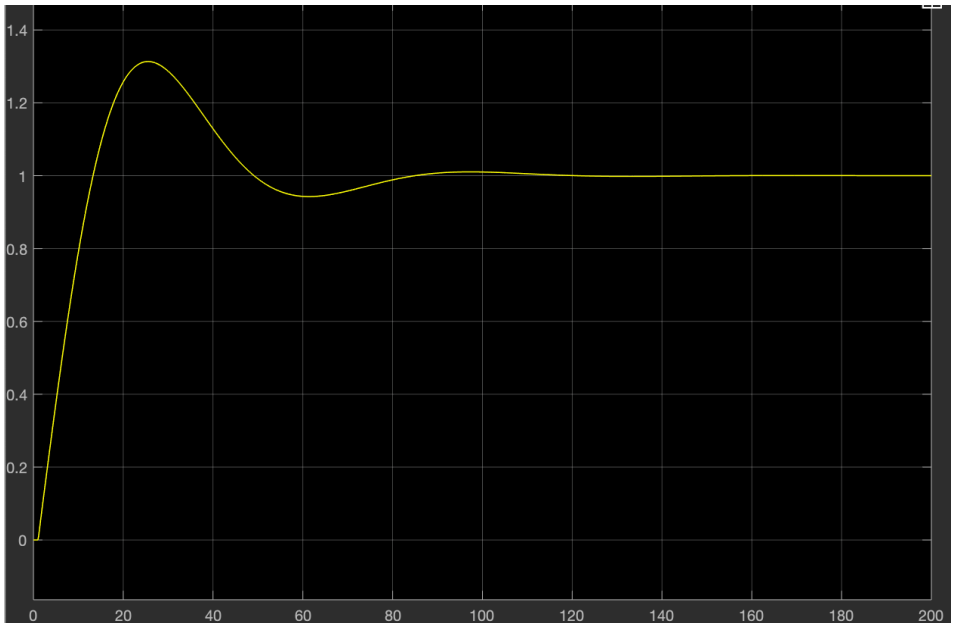
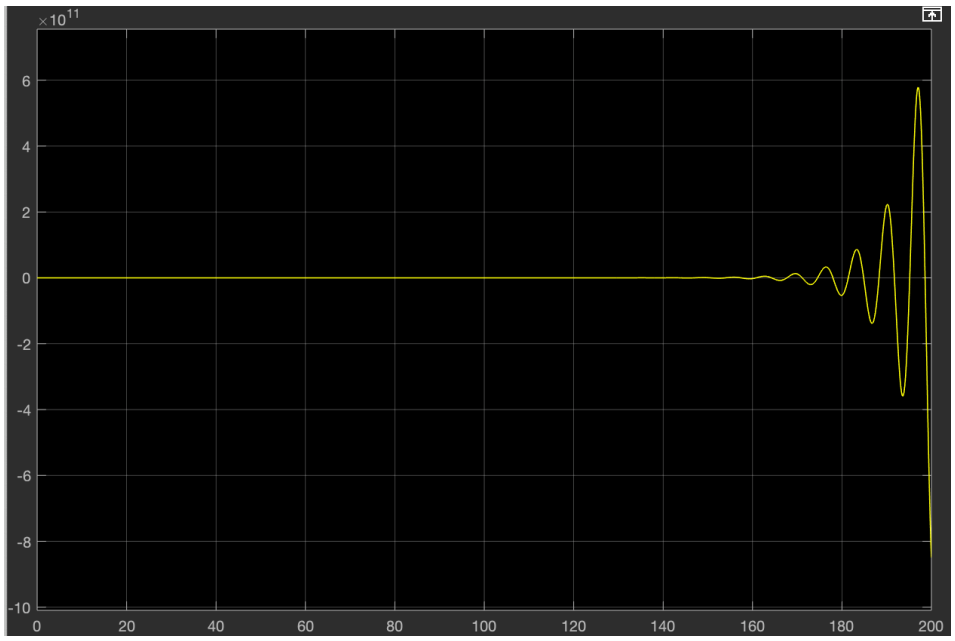


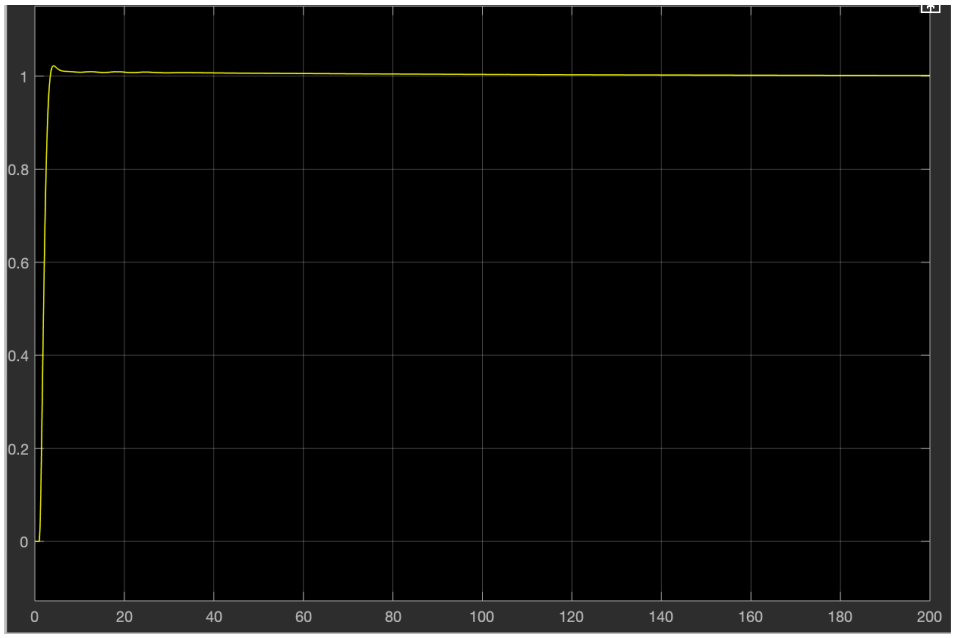
Ki	Kd	Kp	Diagram
0.1	0.1	0.1	

1	0.1	1	 ×10^8 2 1.5 1 0.5 0 -0.5 -1 -1.5 0 10 20 30 40 50 60 70 80 90 100
10	0.1	1	 ×10^24 5 4 3 2 1 0 -1 0 10 20 30 40 50 60 70 80 90 100

1	1	1	 ×10^11 3 2 1 0 -1 -2 -3 -4 -5 -6 0 20 40 60 80 100 120 140 160 180 200
0.1	1	10	 2 1.5 1 0.5 0 0 20 40 60 80 100 120 140 160 180 200

10	1	1	 ×10^44 2 1 0 -1 -2 -3 -4 -5 -6 0 20 40 60 80 100 120 140 160 180 200
10	10	10	 1.8 1.6 1.4 1.2 1 0.8 0.6 0.4 0.2 0 -0.2 0 20 40 60 80 100 120 140 160 180 200

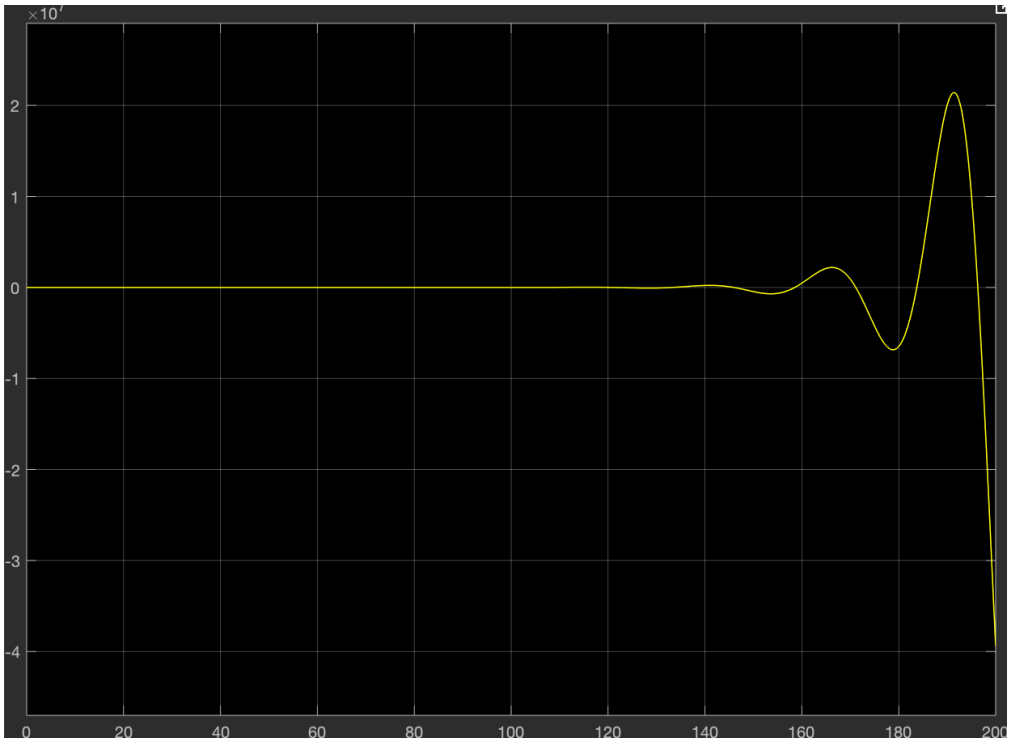
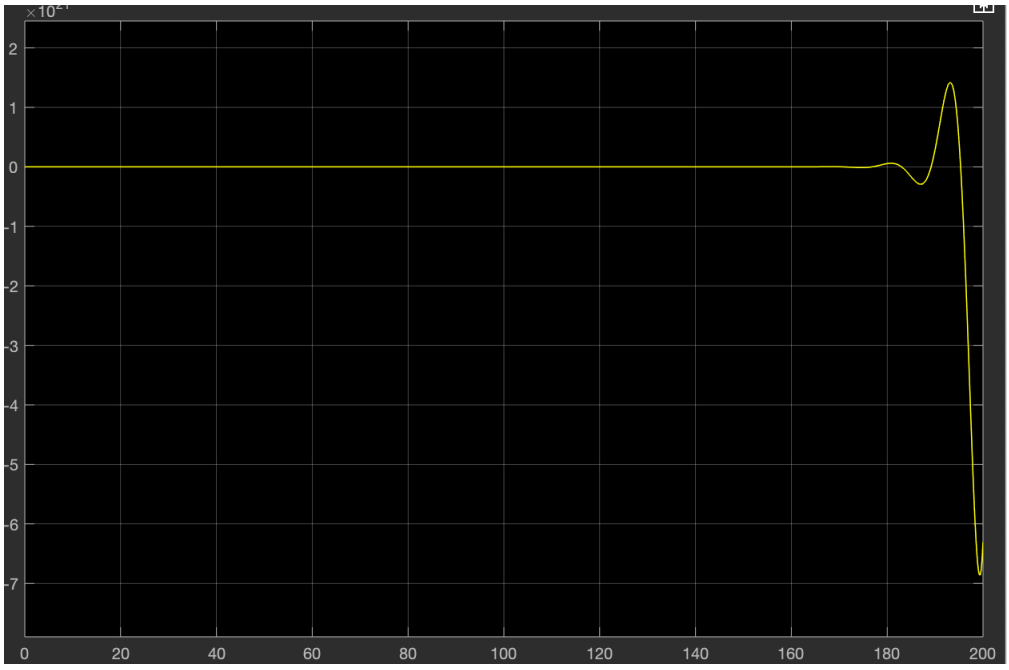
0.1	10	1	 1.4 1.2 1 0.8 0.6 0.4 0.2 0 0 20 40 60 80 100 120 140 160 180 200
10	10	1	 ×10^11 6 4 2 0 -2 -4 -6 -8 -10 0 20 40 60 80 100 120 140 160 180 200

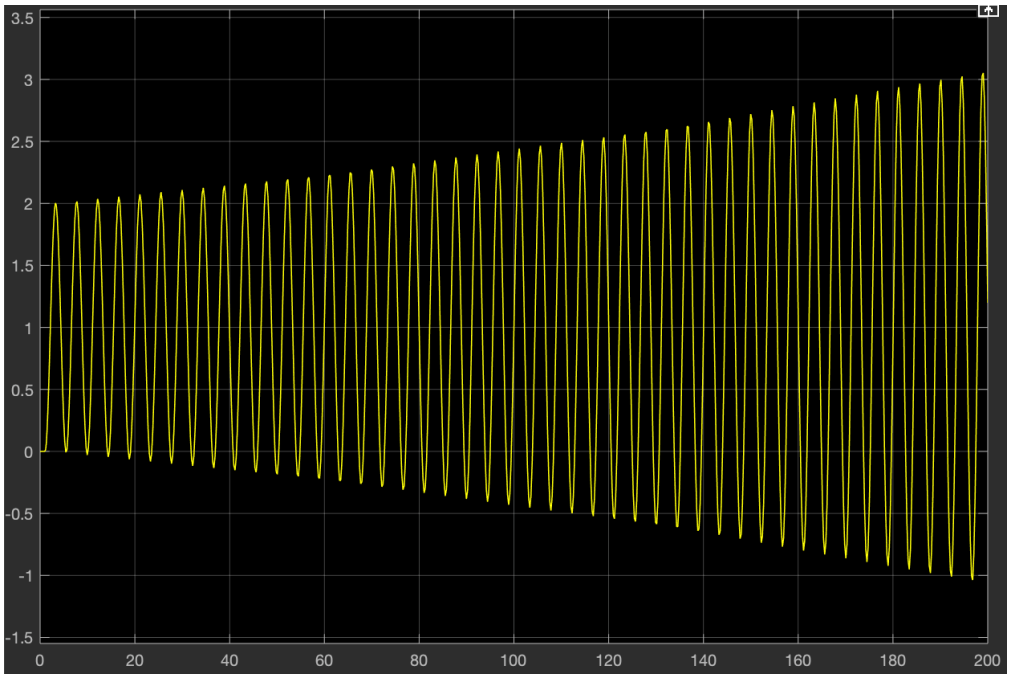
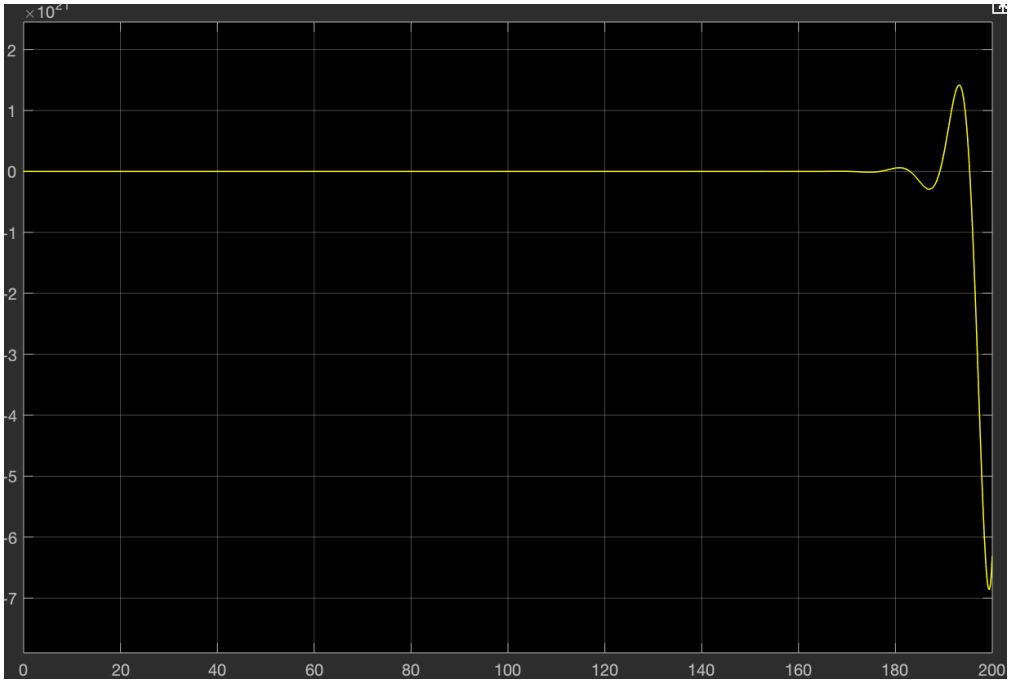
0.1	10	10	

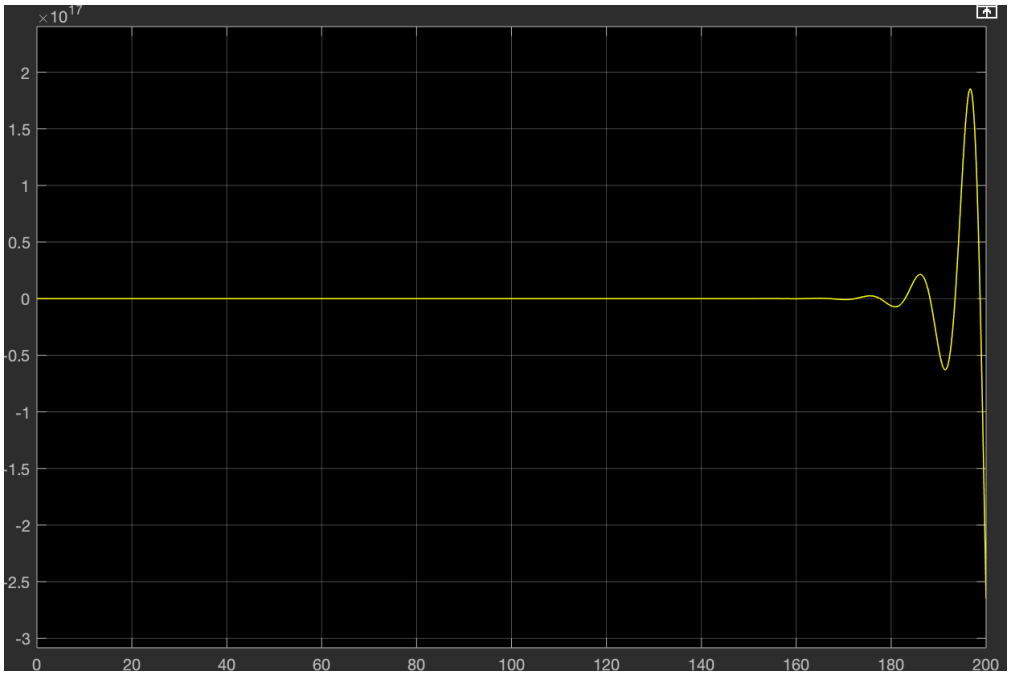
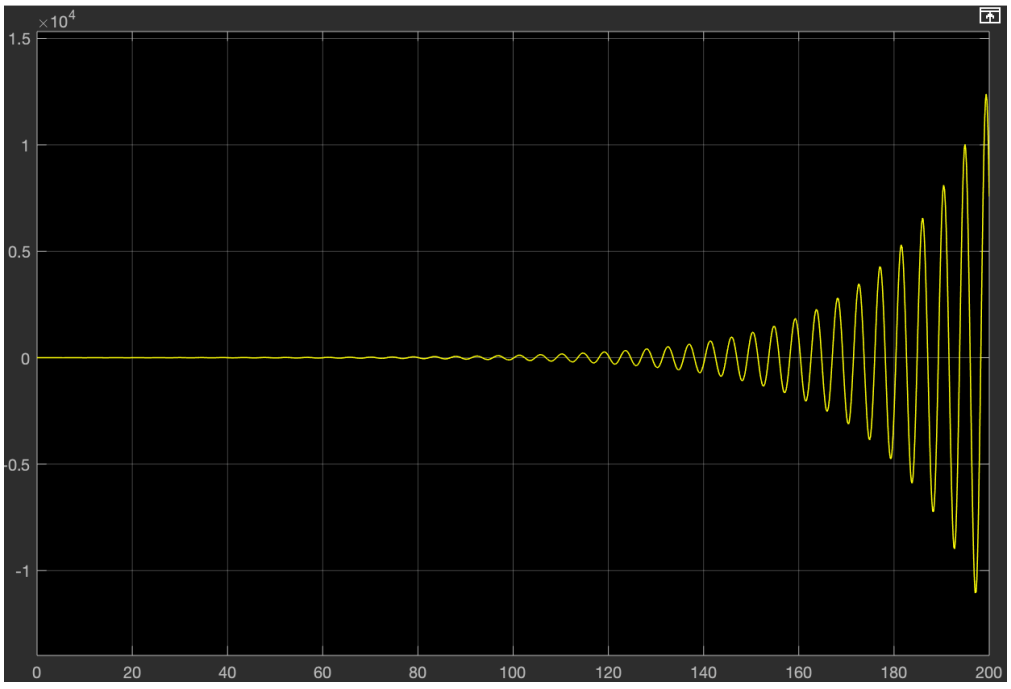
As the integral gain Ki increase, so does the amplitude of the frequency response. As the proportional gain constant Kp increases, so does the frequency of the oscillations of the frequency response. Finally, when the derivative gain constant Kd decreases the frequency of oscillations. If the integral constant is greater than the derivative constant, the system is unstable.

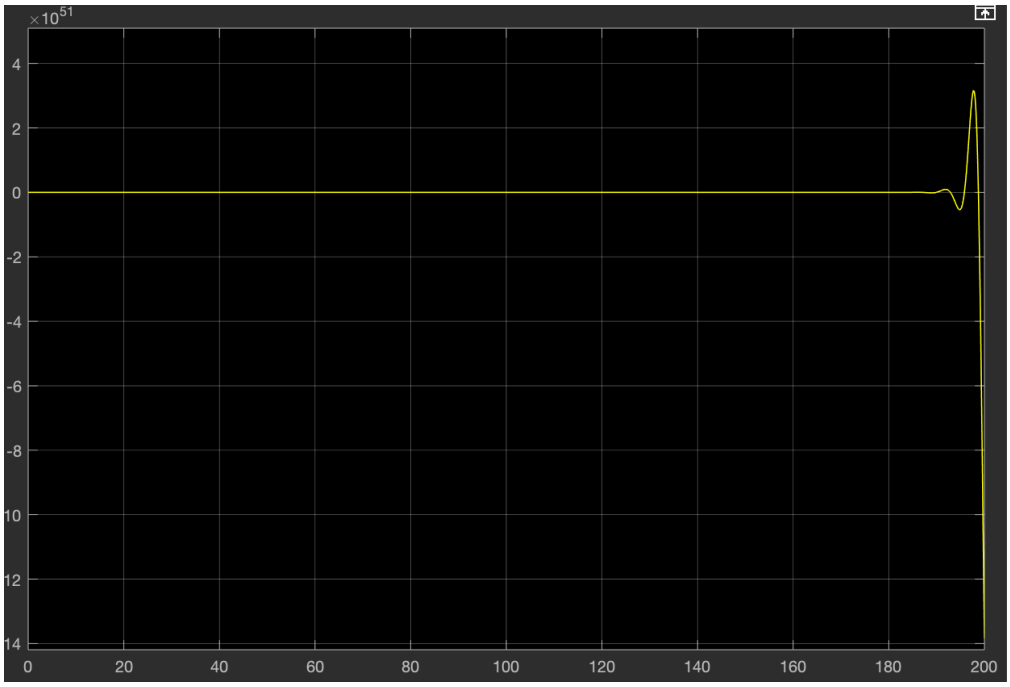
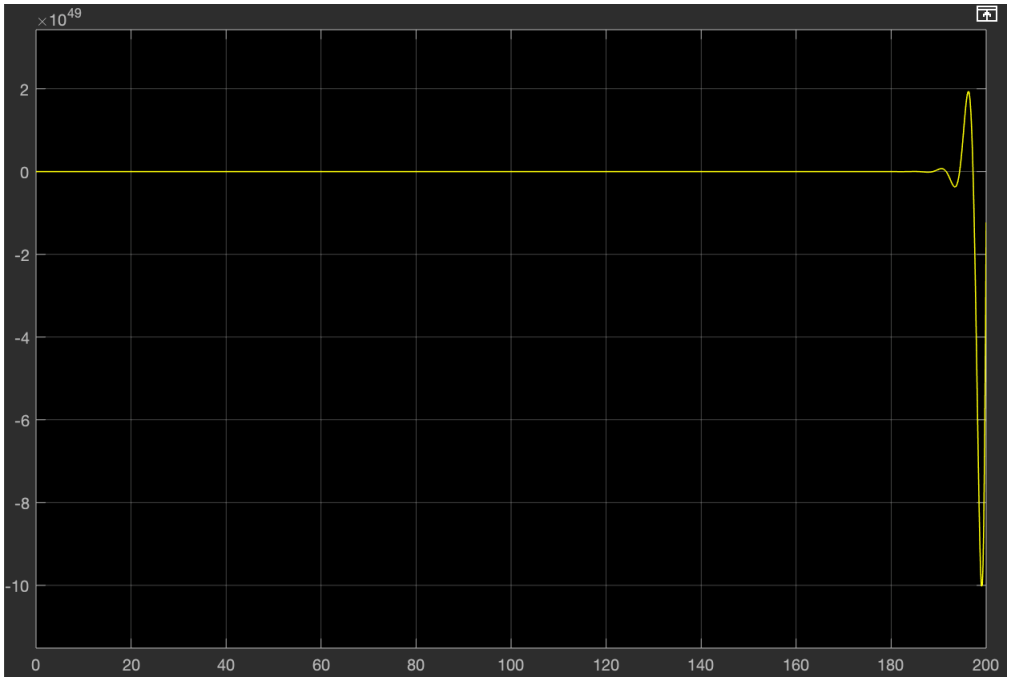
8. Setting $K_d = 0$, Find the step response of the system for different combinations of K_i, $K_p \in \{0.1, 1, 10\}$. This setup corresponds to PI controller. [5 marks]

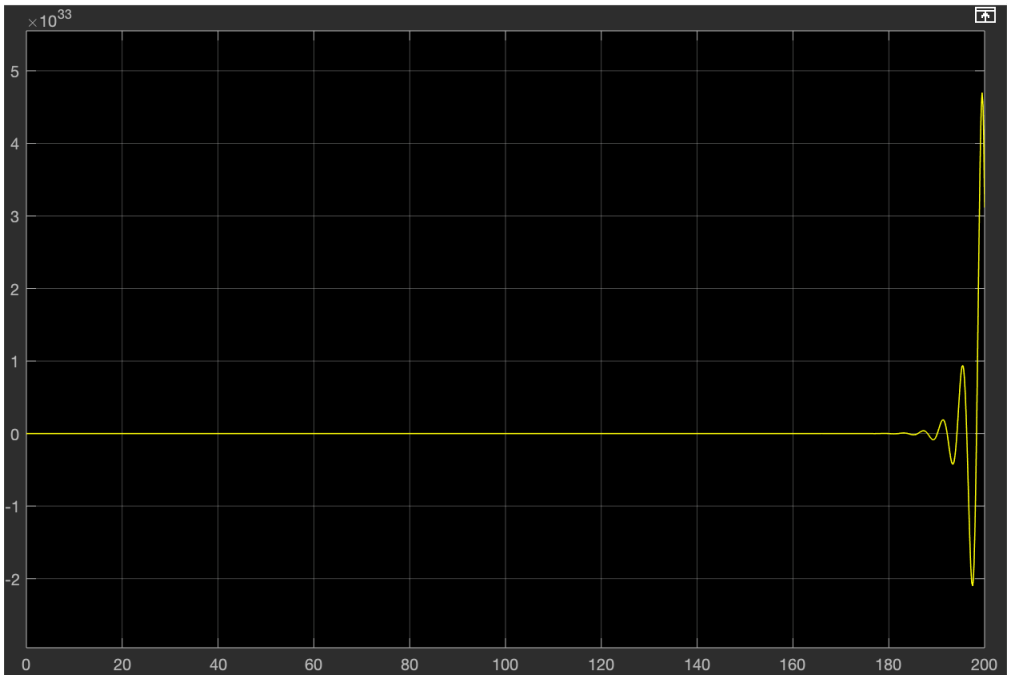
Ki	Kp	Frequency Response

0.1	0.1	
0.1	1	

0.1	10	 3.5 3 2.5 2 1.5 1 0.5 0 -0.5 -1 -1.5 0 20 40 60 80 100 120 140 160 180 200
1	0.1	 ×10^21 2 1 0 -1 -2 -3 -4 -5 -6 -7 0 20 40 60 80 100 120 140 160 180 200

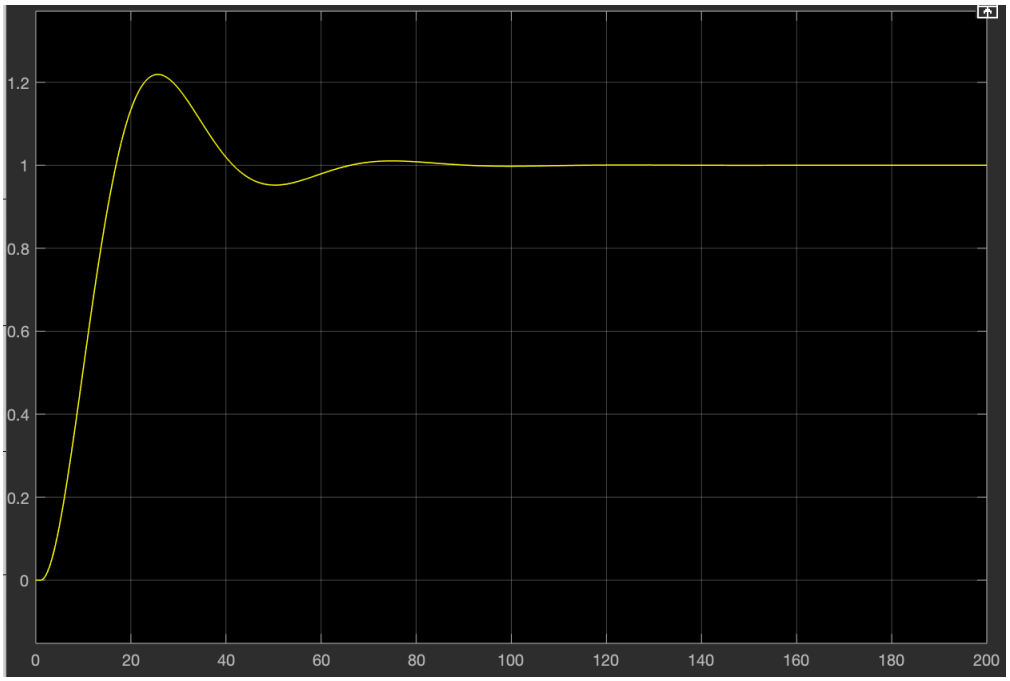
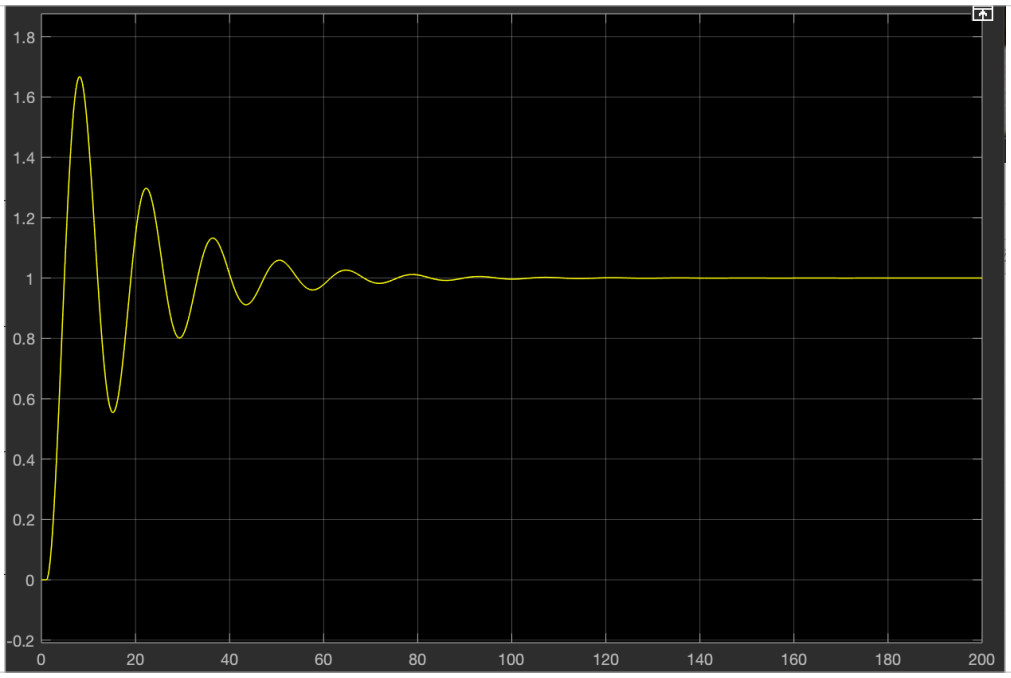
1	1	 ×10^17 2 1.5 1 0.5 0 -0.5 -1 -1.5 -2 -2.5 -3 0 20 40 60 80 100 120 140 160 180 200
1	10	 ×10^4 1.5 1 0.5 0 -0.5 -1 0 20 40 60 80 100 120 140 160 180 200

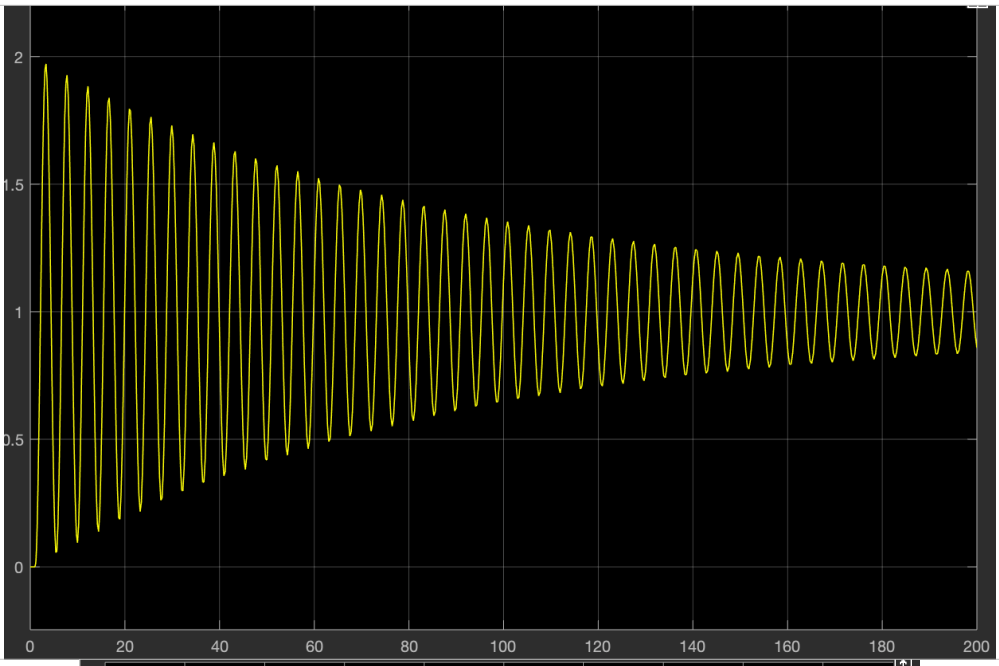
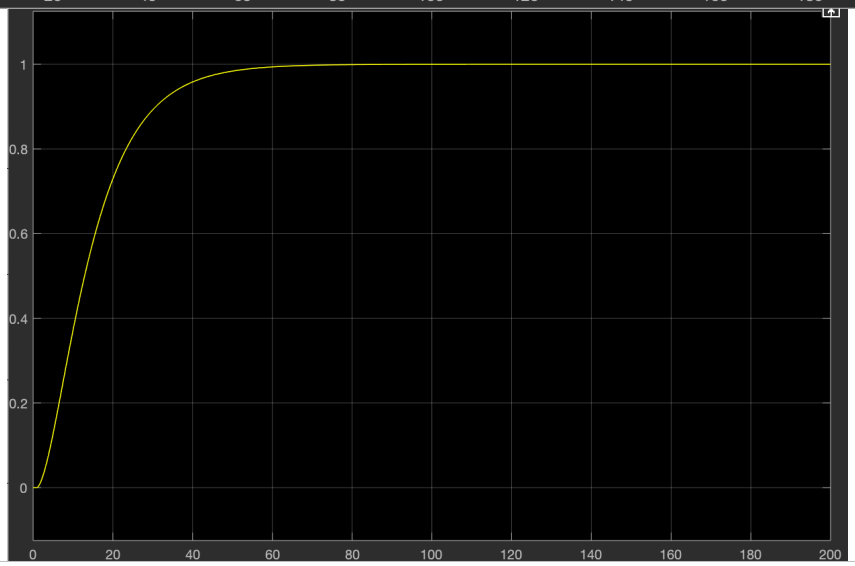
10	0.1	 ×10^51 4 2 0 -2 -4 -6 -8 10 12 14 0 20 40 60 80 100 120 140 160 180 200
10	1	 ×10^49 2 0 -2 -4 -6 -8 -10 0 20 40 60 80 100 120 140 160 180 200

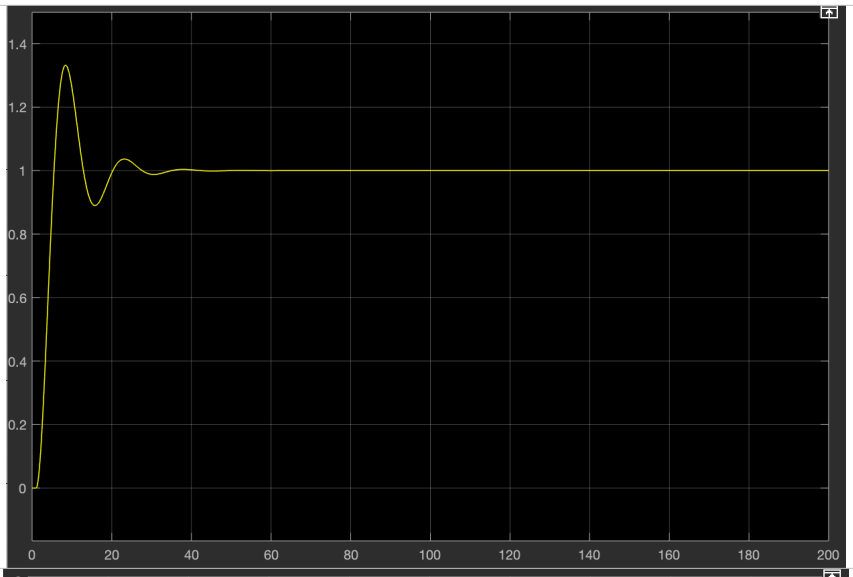
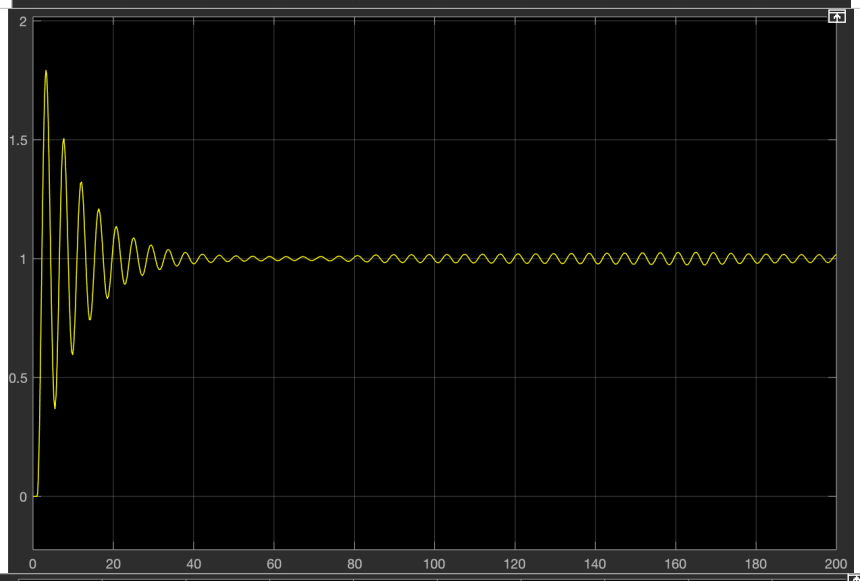
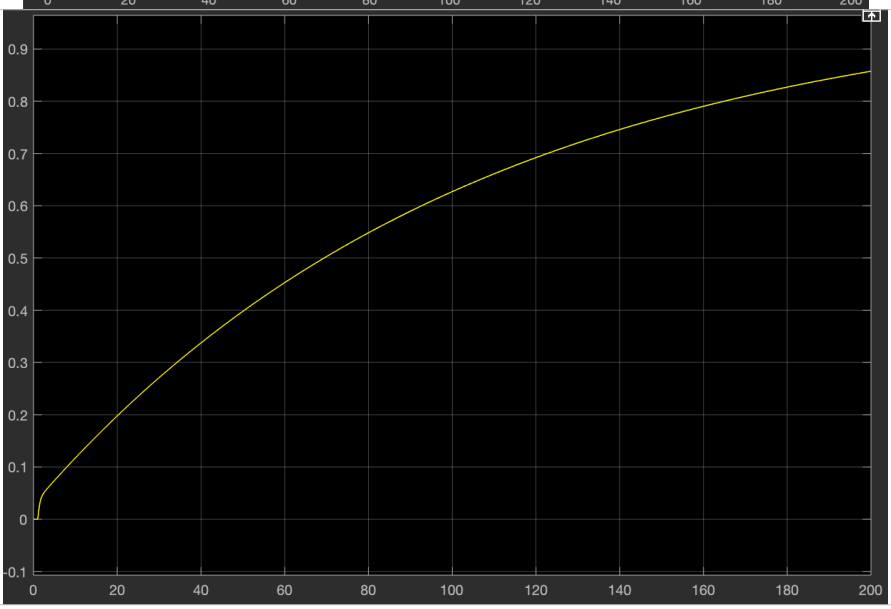
10	10	

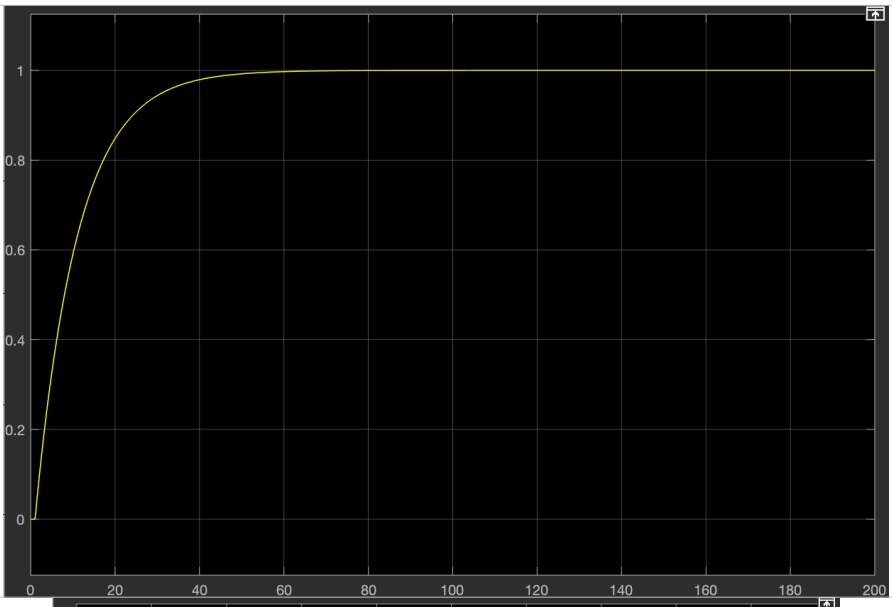
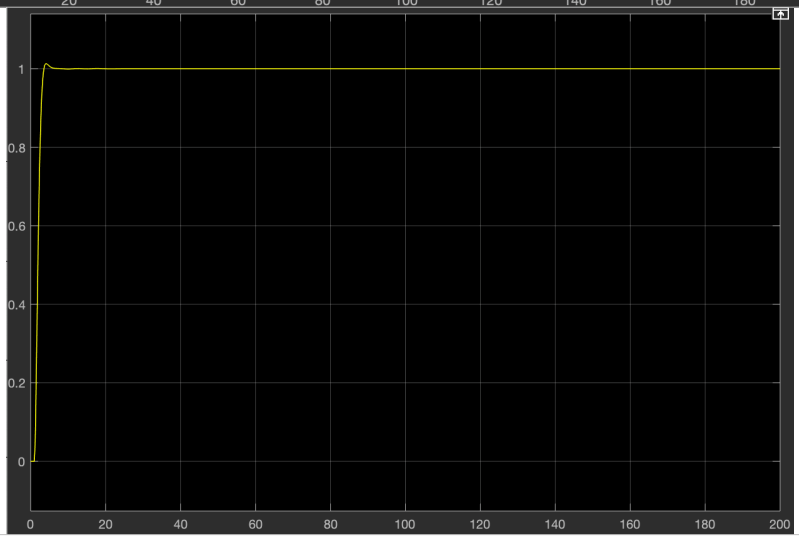
9. *Setting K_i = 0, Find the step response of the system for different combinations of K_d, $K_p \in$ {0.1, 1, 10}.This setup corresponds to PD controller. [5 marks]*

Kd	Kp	

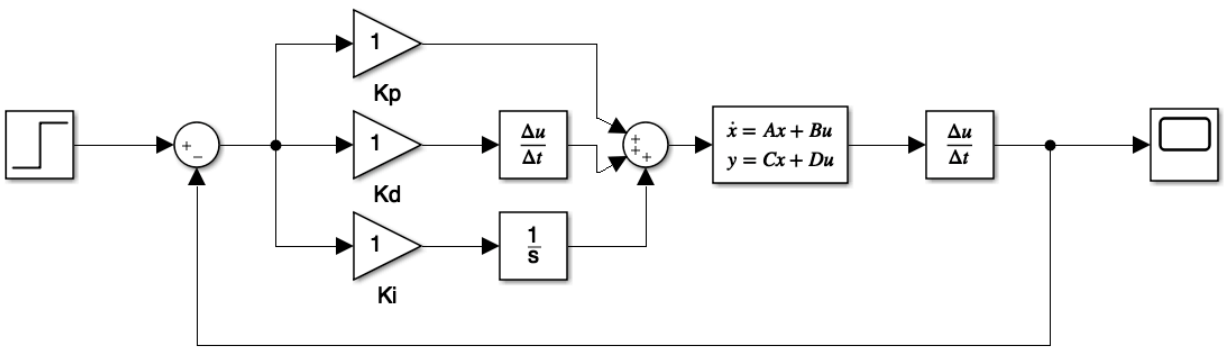
0.1	0.1	 1.2 1 0.8 0.6 0.4 0.2 0 0 20 40 60 80 100 120 140 160 180 200
0.1	1	 1.8 1.6 1.4 1.2 1 0.8 0.6 0.4 0.2 0 -0.2 0 20 40 60 80 100 120 140 160 180 200

0.1	10	 2 1.5 1 0.5 0 0 20 40 60 80 100 120 140 160 180 200
1	0.1	 1 0.8 0.6 0.4 0.2 0 0 20 40 60 80 100 120 140 160 180 200

1	1	 1.4 1.2 1 0.8 0.6 0.4 0.2 0 0 20 40 60 80 100 120 140 160 180 200
1	10	 2 1.5 1 0.5 0 0 20 40 60 80 100 120 140 160 180 200
10	0.1	 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0 -0.1 0 20 40 60 80 100 120 140 160 180 200

10	1	
10	10	

10. Without changing the transfer function block, change (add a block) to the Simulink model such that you can observe angular velocity as the output of the system, and then plot the step-response.



11. Suppose in the real system, we can directly measure the velocity or we can measure the position and implement a derivative block after that to derive velocity. Theoretically these two should give same responses. However, suppose there exists additive noise in the system. Which of these approaches you would chose to measure velocity? Why? [10 marks]

Measuring the the velocity directly is a lot more effective that measuring the velocity has the output of a derivative block applied to position in the presence of noise. This is because the noise will corrupt the velocity, as the derivative of high frequency noise will be very large (positively and negatively) and cannot reflect the true velocity.

12. Using the pid command in Matlab, implement pid controller. By iterating over $K_d \in \{0.1, 1, 10, 100\}$ (and setting K_i, K_p = 1) plot the pole-zero map of the system and explain the effect of derivative block on poles. Repeat these steps for K_p,K_i. which of these plots corresponds to classical root locus diagram? [10 marks]

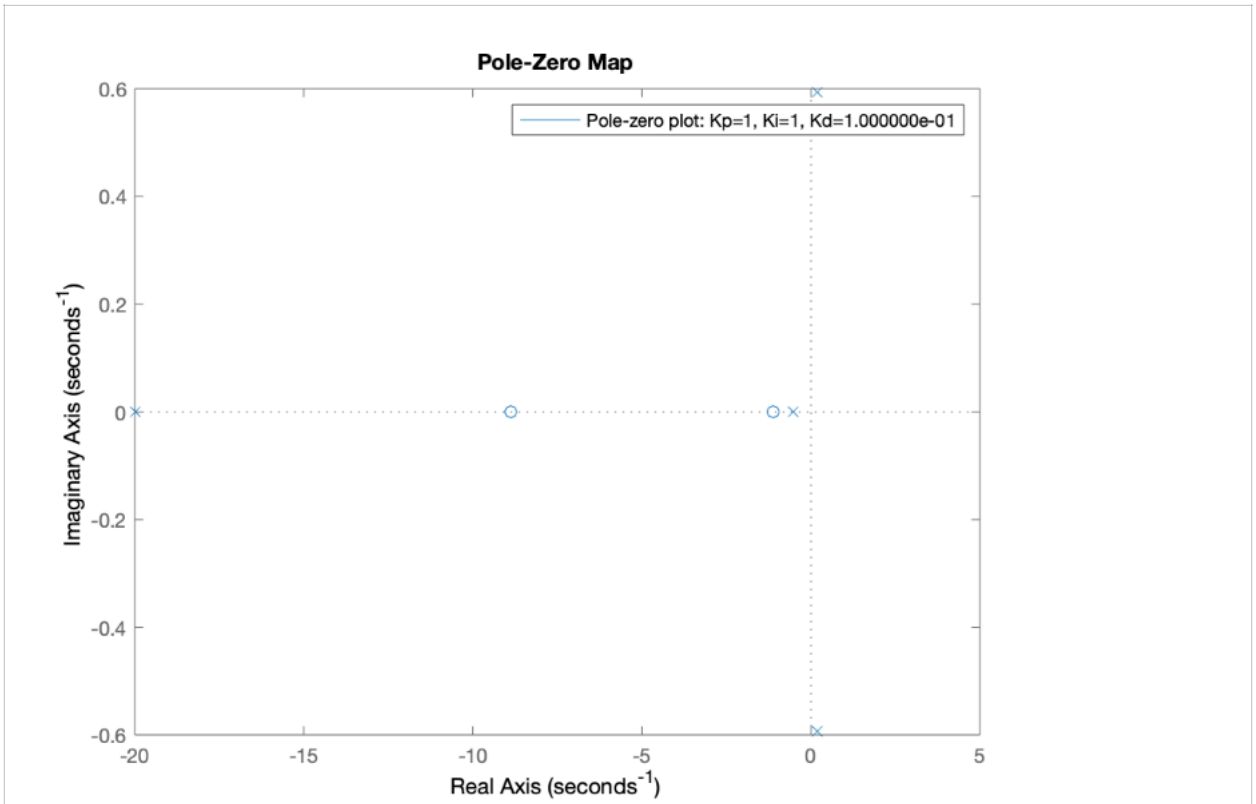
Hint: The pid block gives the transfer function of the controller, in order to find the closed loop pole zero plot you have to put the controller and plant in the open loop and then apply a unity feedback.

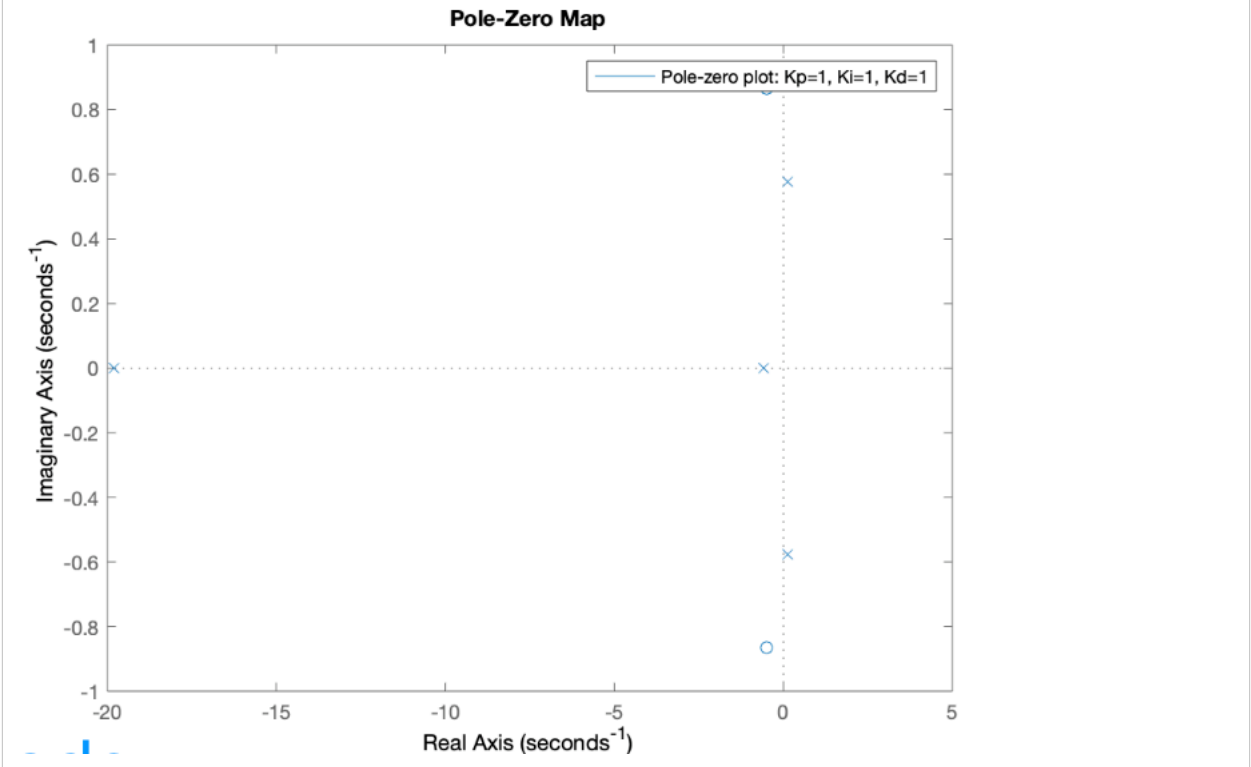
```
Jm = 0.01; %kgm^2 , inertia of the rotor and shaft
b = 0.001; %Nmsec , viscous friction coefficient
Ke = 0.02; %Vsec  , back emf constant
Kt = 0.02; %Nm/A  , motor torque constant
Ra = 10;   %Ω     , armature resitance
La = 0.5;

a0 = 0;
a1 = (b+(Kt*Ke)/Ra)/Jm; %divide equation Jm to get unity coefficient for θ"
b0 = Kt / (Ra*Jm);

A2 = [0 1 0 ; 0 -b/Jm Kt/Jm ; 0 -Ke/La -Ra/La];
B2 = [0 ; 0 ; 1/La];
C2 = [1 0 0 ];
sys = ss(A2,B2,C2,0);
Kd = [0.1 1 10 100];
Kp = [0.1 1 10 100];
Ki = [0.1 1 10 100];

%kd varies
Gd = pid(Kp(1,2), Ki(1,2), Kd(1,1));
```

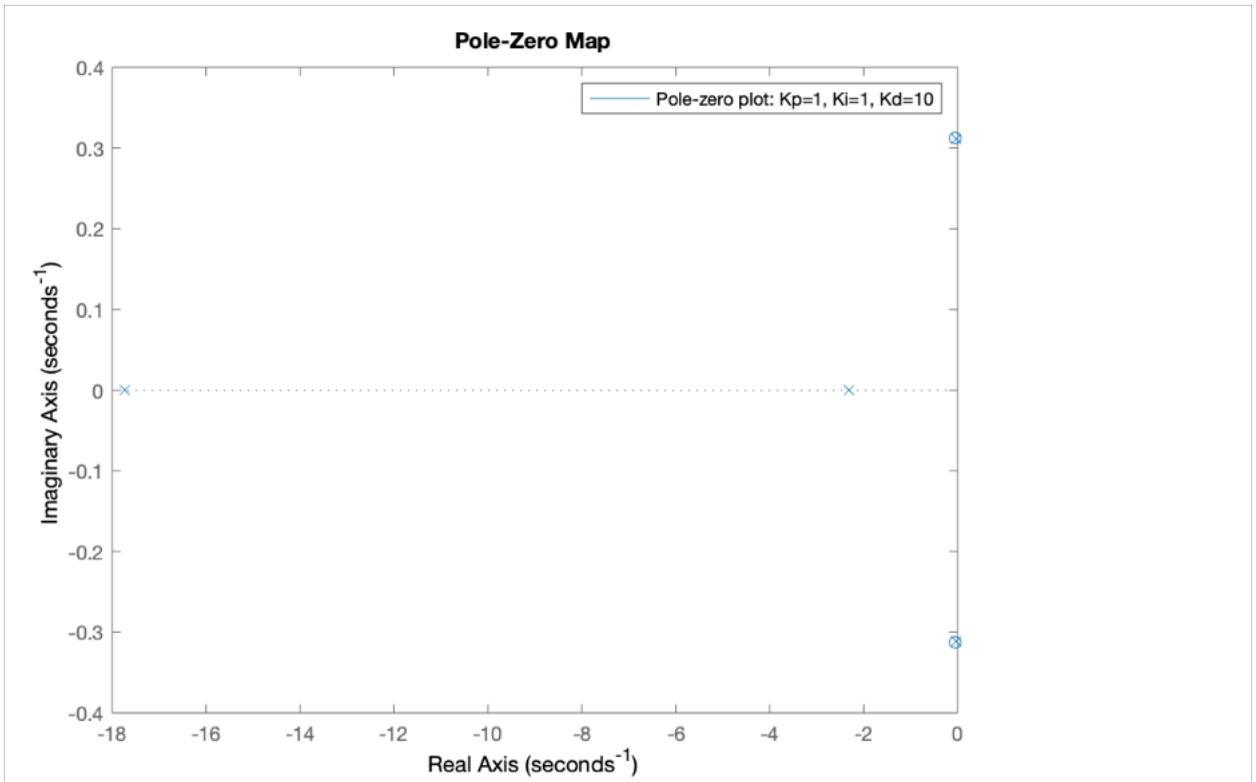
```
Gd = feedback(Gd*sys, 1);
pzmap(ss(Gd))
title = 'Pole-zero plot: Kp=1, Ki=1, Kd=%d';
legend(sprintf(title,Kd(1,1)))

Gd = pid(Kp(1,2), Ki(1,2), Kd(1,2));
Gd = feedback(Gd*sys, 1);
pzmap(ss(Gd))
title = 'Pole-zero plot: Kp=1, Ki=1, Kd=%d';
legend(sprintf(title,Kd(1,2)))

Gd = pid(Kp(1,2), Ki(1,2), Kd(1,3));
Gd = feedback(Gd*sys, 1);
pzmap(ss(Gd))
title = 'Pole-zero plot: Kp=1, Ki=1, Kd=%d';
legend(sprintf(title,Kd(1,3)))

Gd = pid(Kp(1,2), Ki(1,2), Kd(1,4));
Gd = feedback(Gd*sys, 1);
pzmap(ss(Gd))
title = 'Pole-zero plot: Kp=1, Ki=1, Kd=%d';
legend(sprintf(title,Kd(1,4)))

%kp varies
Gp = pid(Kp(1,1), Ki(1,2), Kd(1,2));
Gp = feedback(Gp*sys, 1);
pzmap(ss(Gp))
title = 'Pole-zero plot: Kd=1, Ki=1, Kp=%d';
legend(sprintf(title, Kp(1,1)))

Gp = pid(Kp(1,2), Ki(1,2), Kd(1,2));
Gp = feedback(Gp*sys, 1);
pzmap(ss(Gp))
title = 'Pole-zero plot: Kd=1, Ki=1, Kp=%d';
legend(sprintf(title, Kp(1,2)))

Gp = pid(Kp(1,3), Ki(1,2), Kd(1,2));
Gp = feedback(Gp*sys, 1);
pzmap(ss(Gp))
title = 'Pole-zero plot: Kd=1, Ki=1, Kp=%d';
legend(sprintf(title, Kp(1,3)))

Gp = pid(Kp(1,4), Ki(1,2), Kd(1,2));
```

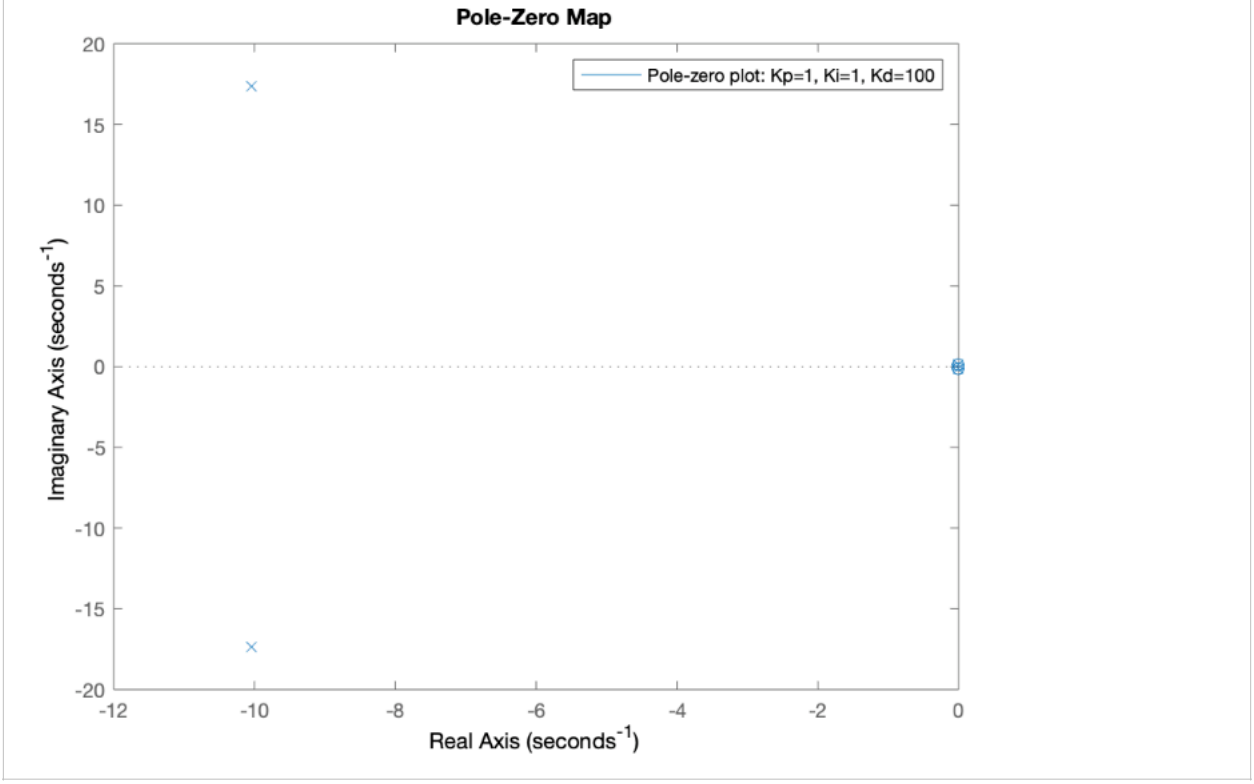
```
Gp = feedback(Gp*sys, 1);
pzmap(ss(Gp))
title = 'Pole-zero plot: Kd=1, Ki=1, Kp=%d';
legend(sprintf(title, Kp(1,4)))

%ki varies
Gi = pid(Kp(1,2), Ki(1,1), Kd(1,2));
Gi = feedback(Gi*sys, 1);
pzmap(ss(Gi))
title = 'Pole-zero plot: Kp=1, Kd=1, Ki=%d';
legend(sprintf(title,Ki(1,1)))

Gi = pid(Kp(1,2), Ki(1,2), Kd(1,2));
Gi = feedback(Gi*sys, 1);
pzmap(ss(Gi))
title = 'Pole-zero plot: Kp=1, Kd=1, Ki=%d';
legend(sprintf(title,Ki(1,2)))

Gi = pid(Kp(1,2), Ki(1,3), Kd(1,2));
Gi = feedback(Gi*sys, 1);
pzmap(ss(Gi))
title = 'Pole-zero plot: Kp=1, Kd=1, Ki=%d';
legend(sprintf(title,Ki(1,3)))

Gi = pid(Kp(1,2), Ki(1,4), Kd(1,2));
Gi = feedback(Gi*sys, 1);
pzmap(ss(Gi))
title = 'Pole-zero plot: Kp=1, Kd=1, Ki=%d';
legend(sprintf(title,Ki(1,4)))
```

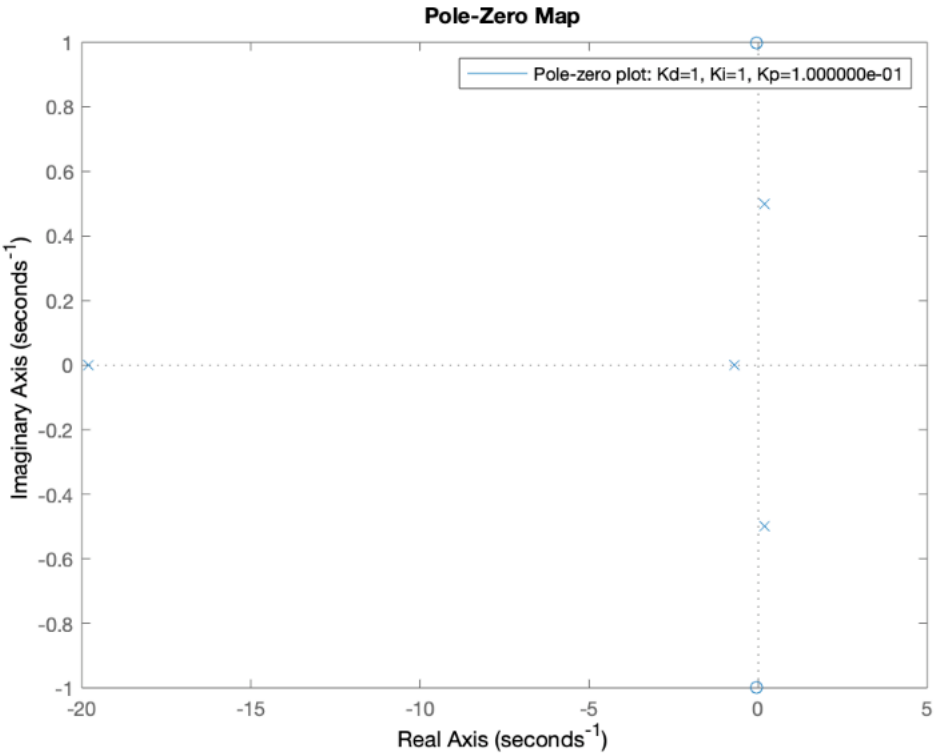
Results:

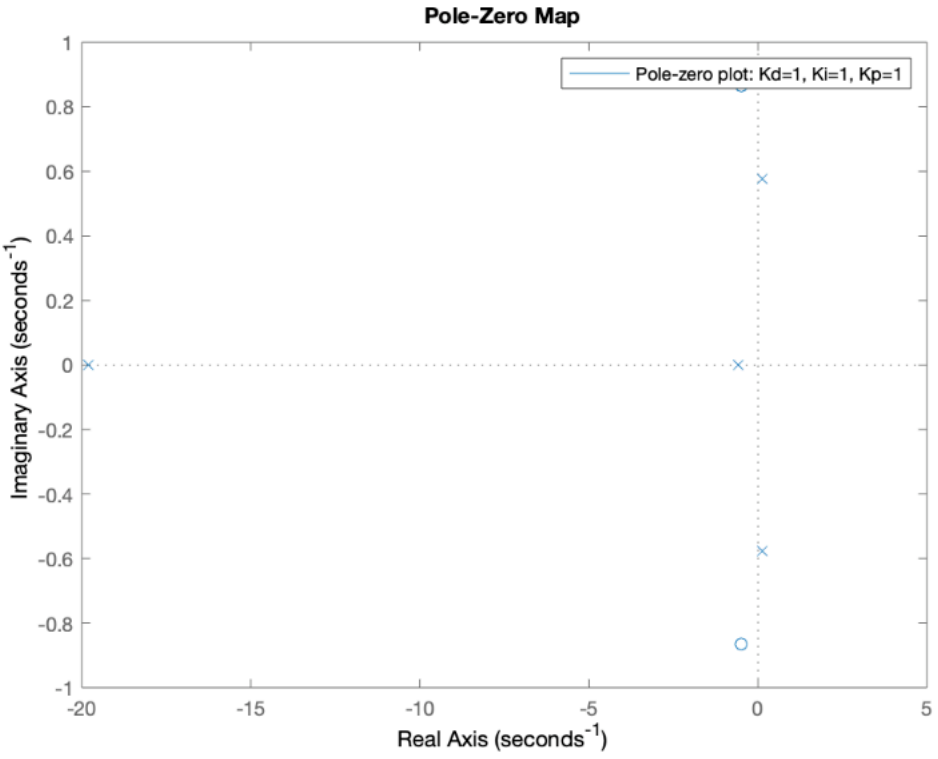

Pole-Zero Map
Pole-zero plot: Kp=1, Ki=1, Kd=1.000000e-01
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-20
-15
-10
-5
0
5

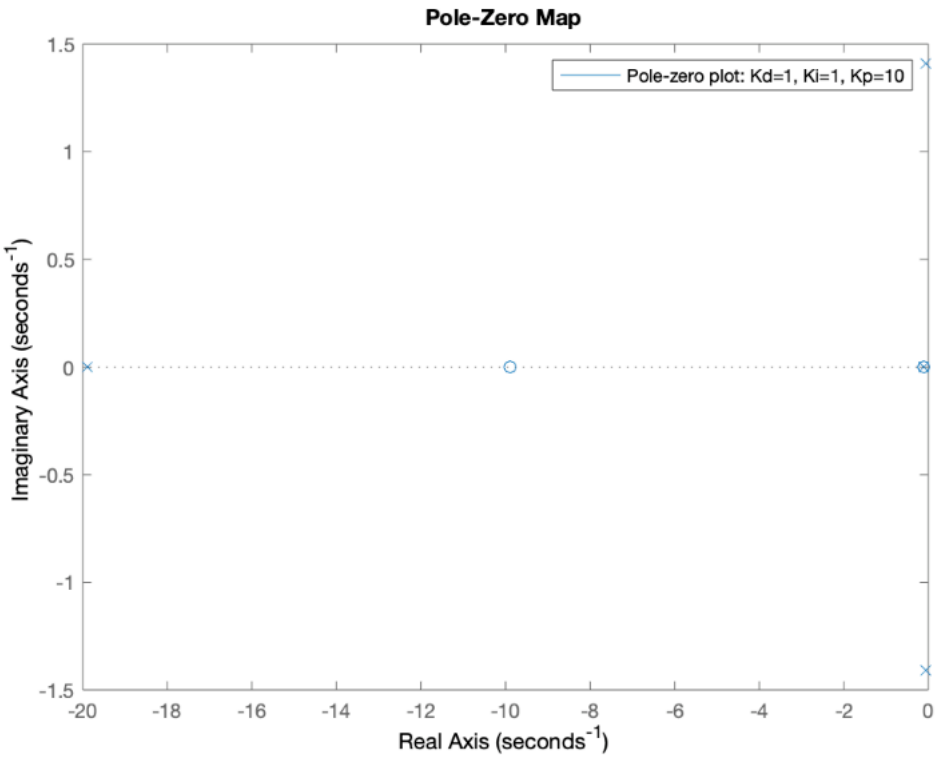

Pole-Zero Map
Pole-zero plot: Kp=1, Ki=1, Kd=1
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-20
-15
-10
-5
0
5

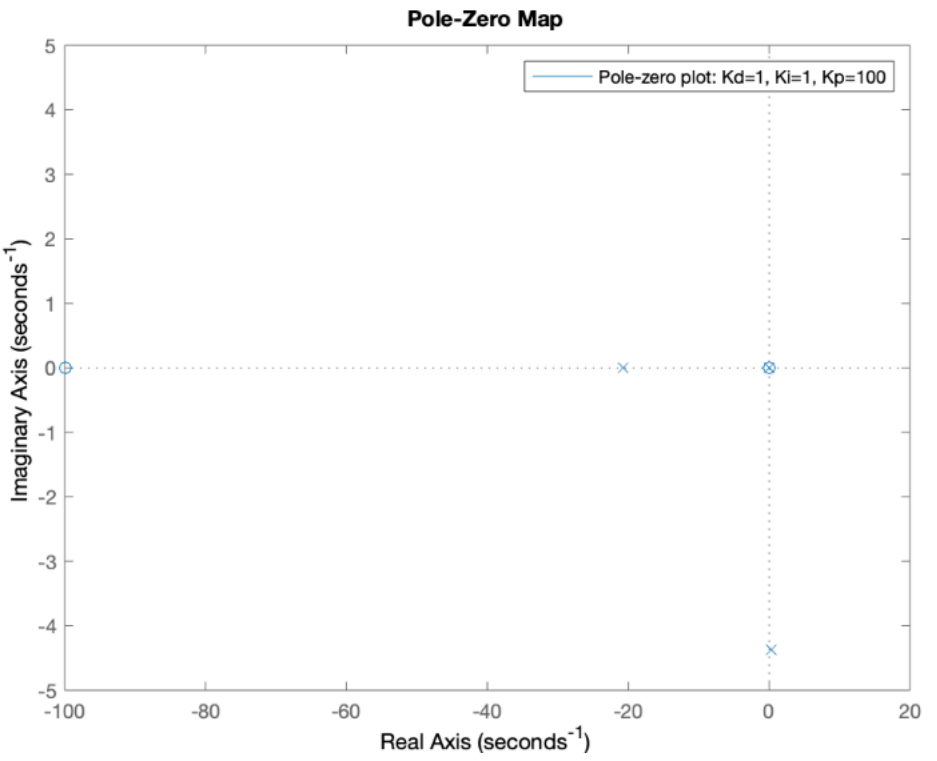

Pole-Zero Map
Pole-zero plot: Kp=1, Ki=1, Kd=10
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-18
-16
-14
-12
-10
-8
-6
-4
-2
0

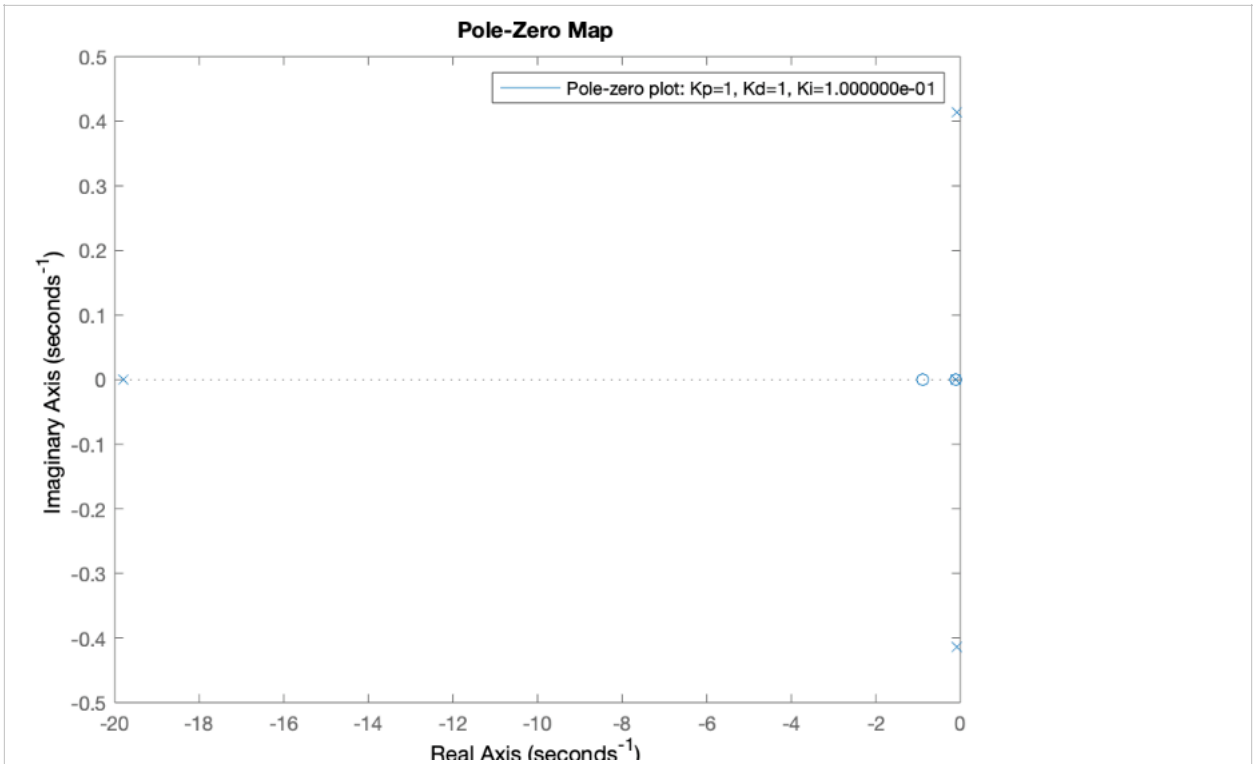

Pole-Zero Map
Pole-zero plot: Kp=1, Ki=1, Kd=100
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
20
15
10
5
0
-5
-10
-15
-20
-12
-10
-8
-6
-4
-2
0

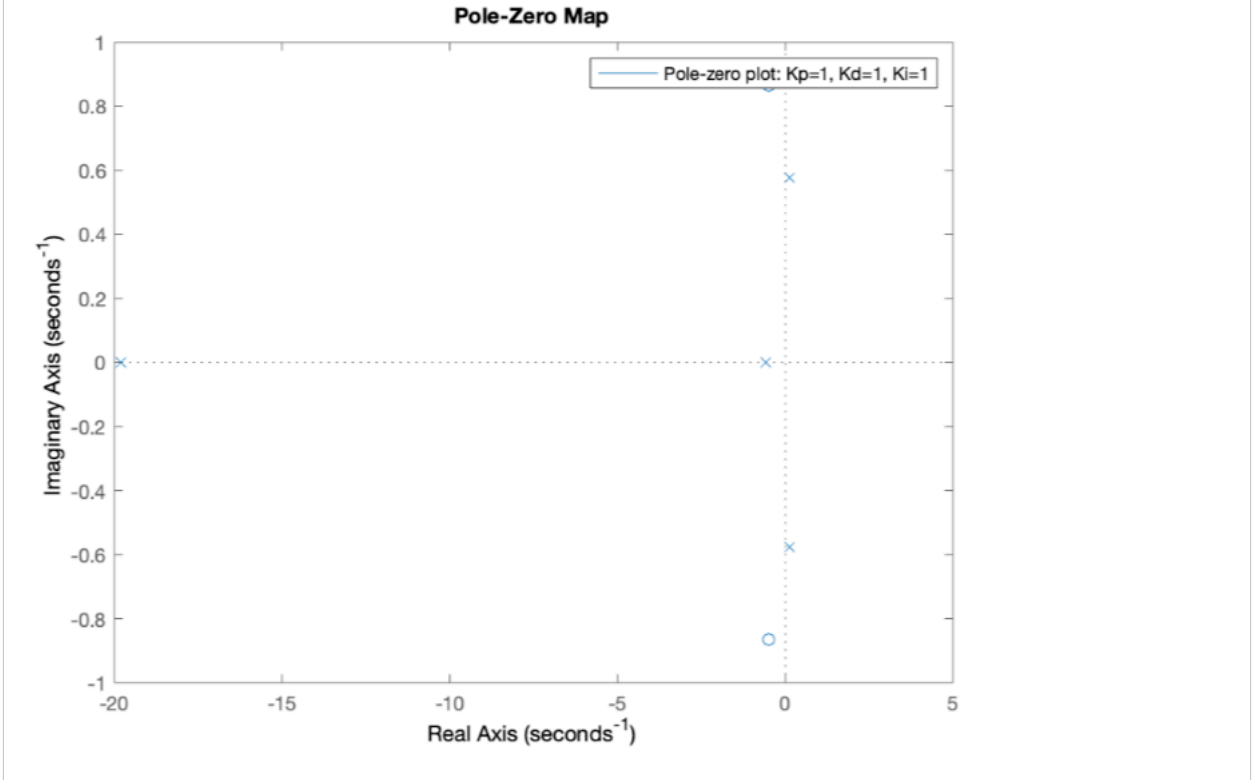

Pole-Zero Map
Pole-zero plot: Kd=1, Ki=1, Kp=1.000000e-01
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-20
-15
-10
-5
0
5

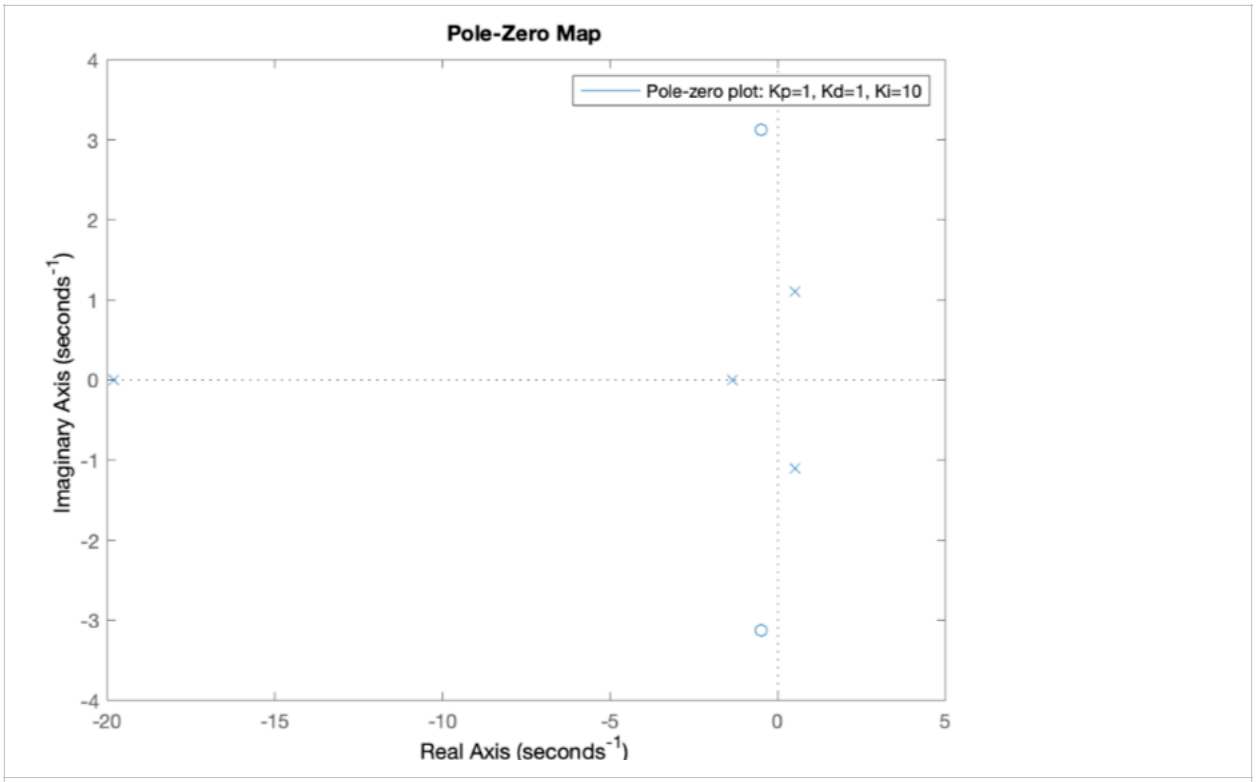

Pole-Zero Map
Pole-zero plot: Kd=1, Ki=1, Kp=1
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-20
-15
-10
-5
0
5

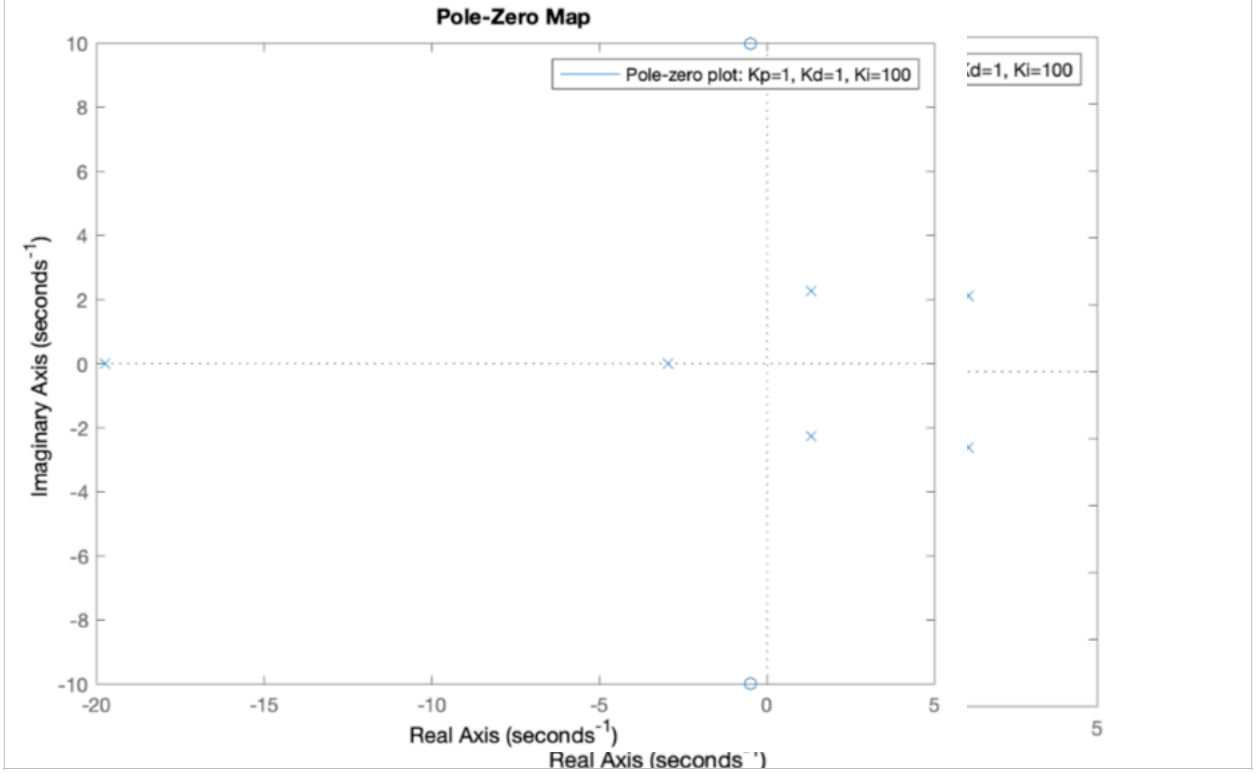

Pole-Zero Map
Pole-zero plot: Kd=1, Ki=1, Kp=10
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
1.5
1
0.5
0
-0.5
-1
-1.5
-20
-18
-16
-14
-12
-10
-8
-6
-4
-2
0


Pole-Zero Map
Pole-zero plot: Kd=1, Ki=1, Kp=100
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
5
4
3
2
1
0
-1
-2
-3
-4
-5
-100
-80
-60
-40
-20
0
20


Pole-Zero Map
Pole-zero plot: Kp=1, Kd=1, Ki=1.000000e-01
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-20
-18
-16
-14
-12
-10
-8
-6
-4
-2
0


Pole-Zero Map
Pole-zero plot: Kp=1, Kd=1, Ki=1
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-20
-15
-10
-5
0
5


Pole-Zero Map
Pole-zero plot: Kp=1, Kd=1, Ki=10
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
4
3
2
1
0
-1
-2
-3
-4
-20
-15
-10
-5
0
5


Pole-Zero Map
Pole-zero plot: Kp=1, Kd=1, Ki=100
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
10
8
6
4
2
0
-2
-4
-6
-8
-10
-20
-15
-10
-5
0
5

Appendices:

Appendix A: Matlab code for bode plots

```
Jm = 0.01; %kgm^2 , inertia of the rotor and shaft
b = 0.001; %Nmsec , viscous friction coefficient
Ke = 0.02; %Vsec  , back emf constant
Kt = 0.02; %Nm/A  , motor torque constant
Ra = 10;   %Ω     , armature resitance
La = 0.5;

a0 = 0;
a1 = (b+(Kt*Ke)/Ra)/Jm; %divide equation Jm to get unity coefficient for θ"
b0 = Kt / (Ra*Jm);

% %Theoretical
G1 = tf([b0], [1 a1 a0]);
w = [0.1, 1000];
figure(1);
bode(G1, w);
%
% %Expeimental
x = [-1 0 1 2 3];
y = [20*log10(33) 20*log10(1.93) 20*log10(0.1942) 20*log10(0.01925) 20*log10(0.001923)];
plot(x, y)
legend('Experimental Bode Plot')
xlabel('log10(w)')
ylabel('20log(gain)')


% %Theoretical
G2 = tf([b0], [1 a1 b0]);
figure(1);
bode(G2, w);
%
% %Expeimental
x2 = [-1 0 1 2 3];
y2 = [20*log10(1.073) 20*log10(0.21) 20*log10(0.03918) 20*log10(0.00379)
20*log10(0.000377)];
plot(x2, y2)
legend('Experimental Bode Plot')
xlabel('log10(w)')
```

```
ylabel('20log(gain)')


%Theoretical
A2 = [0 1 0 ; 0 -b/Jm Kt/Jm ; 0 -Ke/La -Ra/La];
B2 = [0 ; 0 ; 1/La];
C2 = [1 0 0 ];
[b5,a5] = ss2tf(A2,B2,C2,0);
G5 = tf([b5],[a5])
bode(G5, wmin_max);


% %Expeimental
x3 = [-1 0 1 2 3];
y3 = [20*log10(33.1) 20*log10(2.122) 20*log10(0.1941) 20*log10(0.01923)
20*log10(0.001923)];
plot(x3, y3)
legend('Experimental Bode Plot')
xlabel('log10(w)')
ylabel('20log(gain)')
```