

PES UNIVERSITY

**(EstablishedunderKarnatakaActNo.16of2013)**

100-ftRingRoad,Bengaluru–560085,Karnataka,India

**UE21EC241B**

**IV Semester**

**Control Systems**

**PROJECT**

*Submitted by*

*AKASH RAVI BHAT*PES1UG21EC025

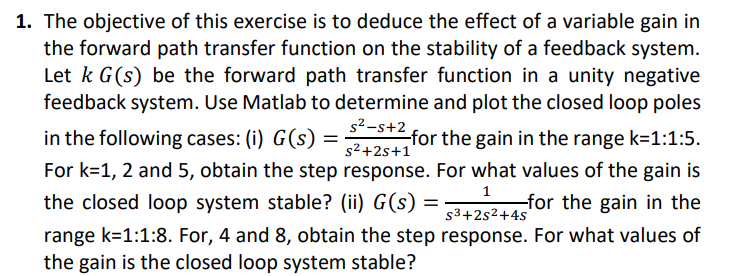
**Jan-May2023**

**PROF. THIPPESWAMY E.**

Department of Electronics and Communication Engineering PESUniversity

Bengaluru-560085

Questions

1. 

Ans:

CODE:

(i).

%%For 𝐺(𝑠) = (𝑠^2−𝑠+2) / (𝑠^2+2𝑠+1),

%Defining transfer function

num = [1 -1 2];

den = [1 2 1];

G = tf(num, den);

%to obtain closed loop transfer function

for k = 1:5

sys = feedback(k\*G, 1);

poles = pole(sys);

disp(['k = ' num2str(k) ', poles = ' num2str(poles.')]);

end

%Checking stability

for k = 1:5

sys = feedback(k\*G, 1);

poles = pole(sys);

if all(real(poles) < 0)

disp(['k = ' num2str(k) ', system is stable']);

else

disp(['k = ' num2str(k) ', system is unstable']);

end

end

for k = [1 2 5]

sys = feedback(k\*G, 1);

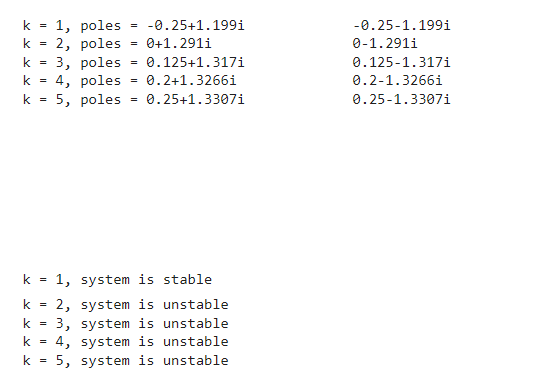
step(sys);

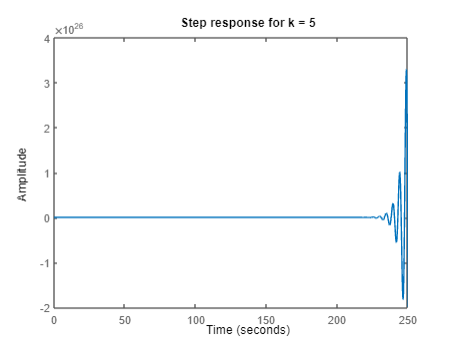
title(['Step response for k = ' num2str(k)]);

%pause;

end

OUTPUT:





(ii).

num = 1;

den = [1 2 4 0];

G = tf(num, den);

%To obtain closed loop tf

for k = 1:8

sys = feedback(k\*G, 1);

poles = pole(sys);

disp(['k = ' num2str(k) ', poles = ' num2str(poles.')]);

end

for k = 1:8

sys = feedback(k\*G, 1);

poles = pole(sys);

if all(real(poles) < 0)

disp(['k = ' num2str(k) ', system is stable']);

else

disp(['k = ' num2str(k) ', system is unstable']);

end

end

%check stability

for k = [4 8]

sys = feedback(k\*G, 1);

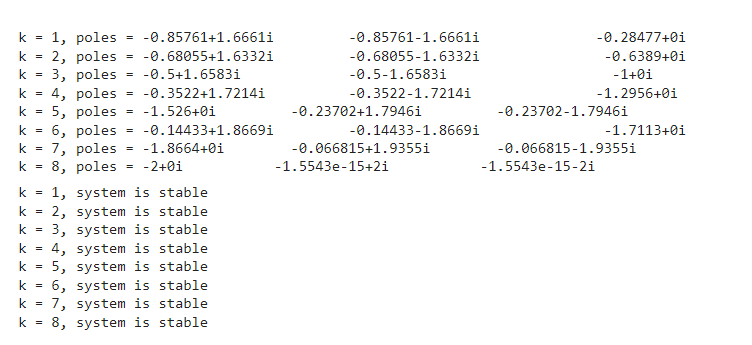
step(sys);

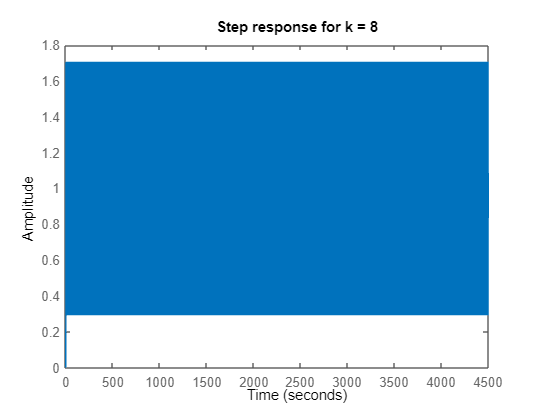
title(['Step response for k = ' num2str(k)]);

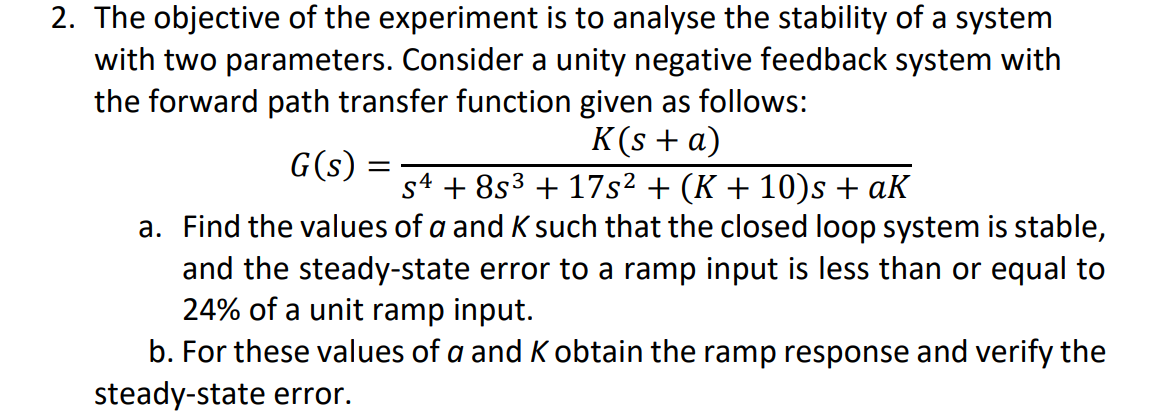
%pause;

end

OUTPUT:





1. 

Ans:

CODE: K = 50;

a = 1;

G = tf([K\*a, K], [1, 8+a, 17+10\*a, K+10, a\*K]);

% Ramp input

t = 0:0.01:10;

ramp = t;

% Closed-loop system

H = feedback(G, 1);

% Step response

[y, t] = lsim(H, ramp, t);

% Steady-state error

ess = abs(1 - y(end));

% Plot

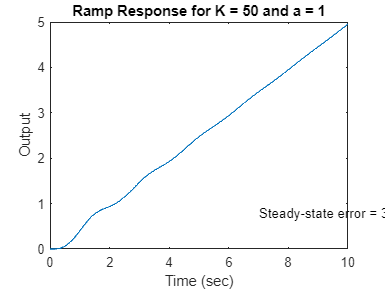
plot(t, y);

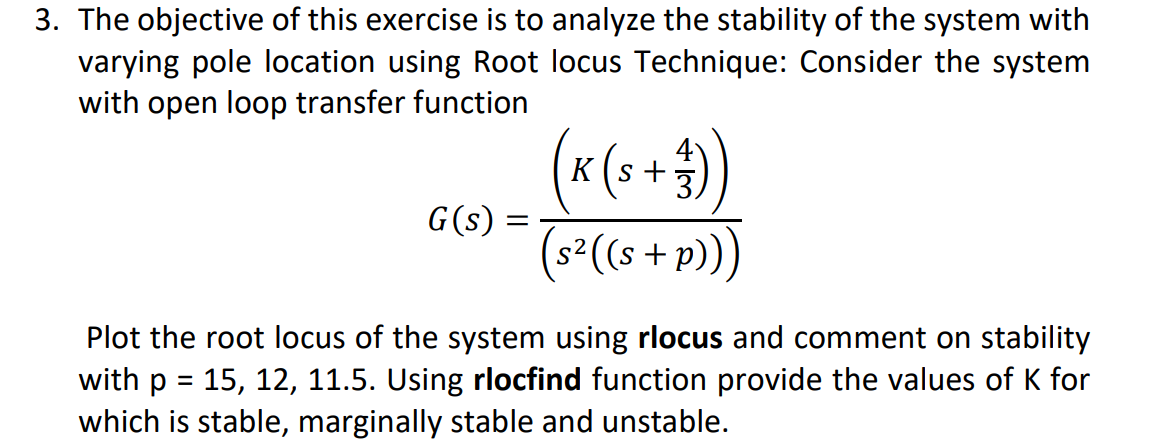
title(['Ramp Response for K = ', num2str(K), ' and a = ', num2str(a)]);

xlabel('Time (sec)');

ylabel('Output');

text(7, 0.8, ['Steady-state error = ', num2str(ess)]);



1. 

Ans:  
CODE:

p = [15 12 11.5];

for i = 1:length(p)

G = tf([1.3333\*K K], [1 0 p(i)]);

figure;

rlocus(G);

if p(i) >= 11.5

disp('System is stable');

else

disp('System is unstable');

end

K\_stable = rlocfind(G, -5);

K\_marginally\_stable = rlocfind(G, -4.8);

K\_unstable = rlocfind(G, -4.5);

disp(['K for stability: ' num2str(K\_stable)]);

disp(['K for marginal stability: ' num2str(K\_marginally\_stable)]);

disp(['K for instability: ' num2str(K\_unstable)]);

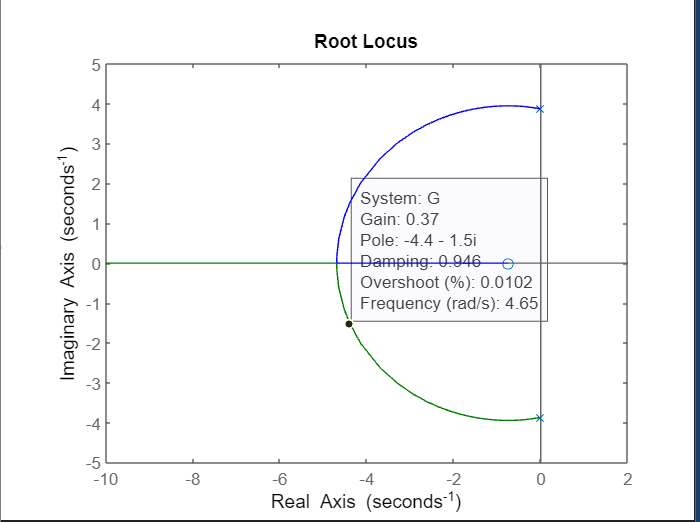
end

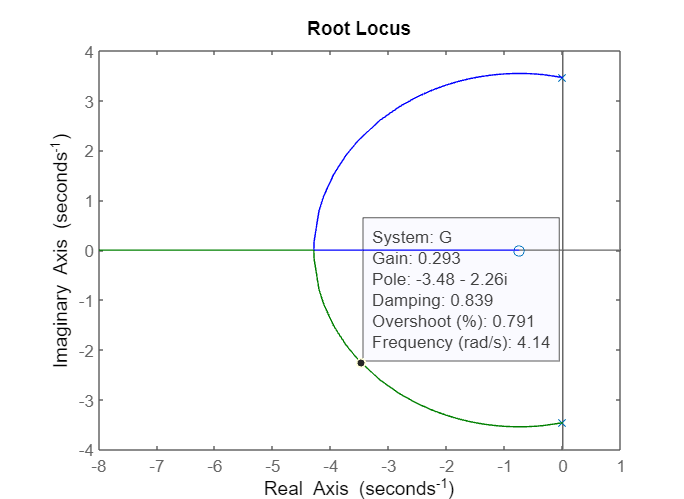
OUTPUT:

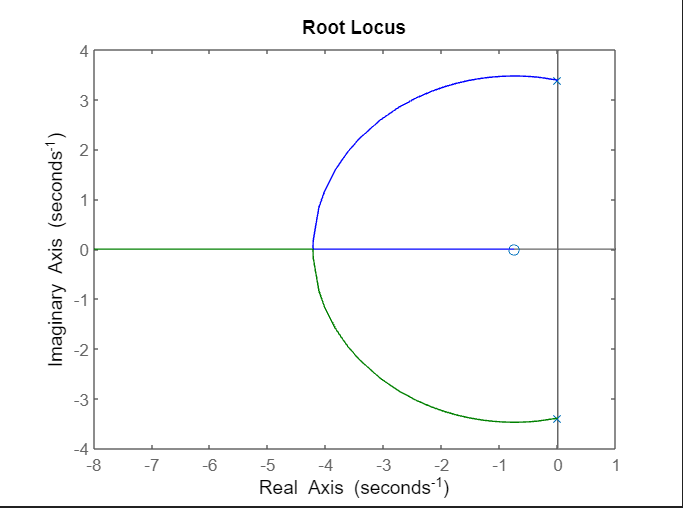
System is stable  
K for stability: 0.39559  
K for marginal stability: 0.39478  
K for instability: 0.39509

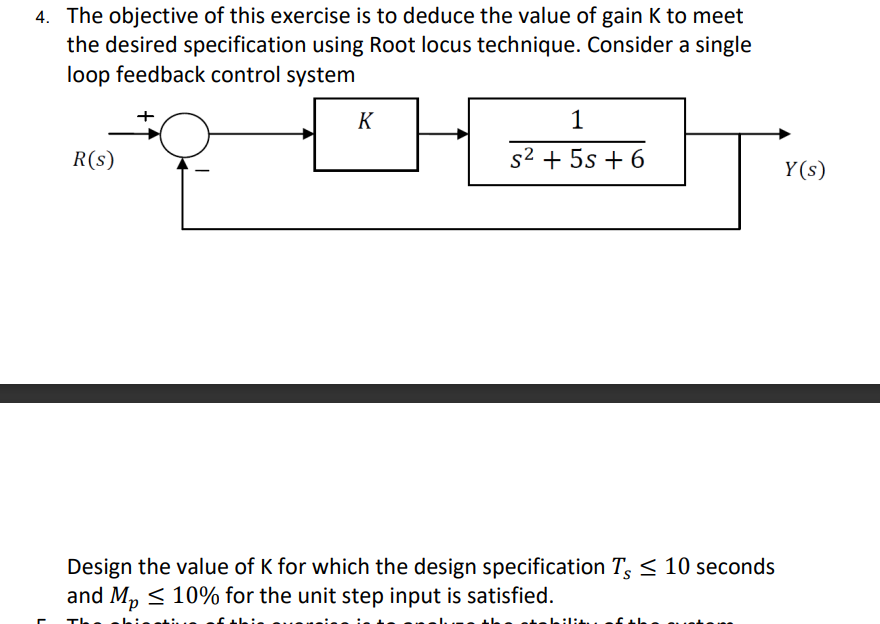
System is stable  
K for stability: 0.36592  
K for marginal stability: 0.36365  
K for instability: 0.36147

System is stable  
K for stability: 0.36097  
K for marginal stability: 0.35846  
K for instability: 0.35586







1. 

Ans:

CODE:

% Define the transfer function of the system

num = K;

den = [1 5 6];

G = tf(num, den);

% Plot the root locus of the system

rlocus(G);

% Define the desired specifications

T\_desired = 10;

Mp\_desired = 0.1;

% Find the value of K from the root locus that meets the specifications

s = -5 + 5i; % point on the root locus

z = -log(Mp\_desired)/sqrt(pi^2 + log(Mp\_desired)^2); % damping ratio

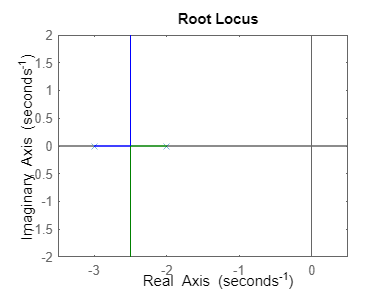
wn = pi/(T\_desired\*sqrt(1-z^2)); % natural frequency

sigma = real(s); % real part of the point on the root locus

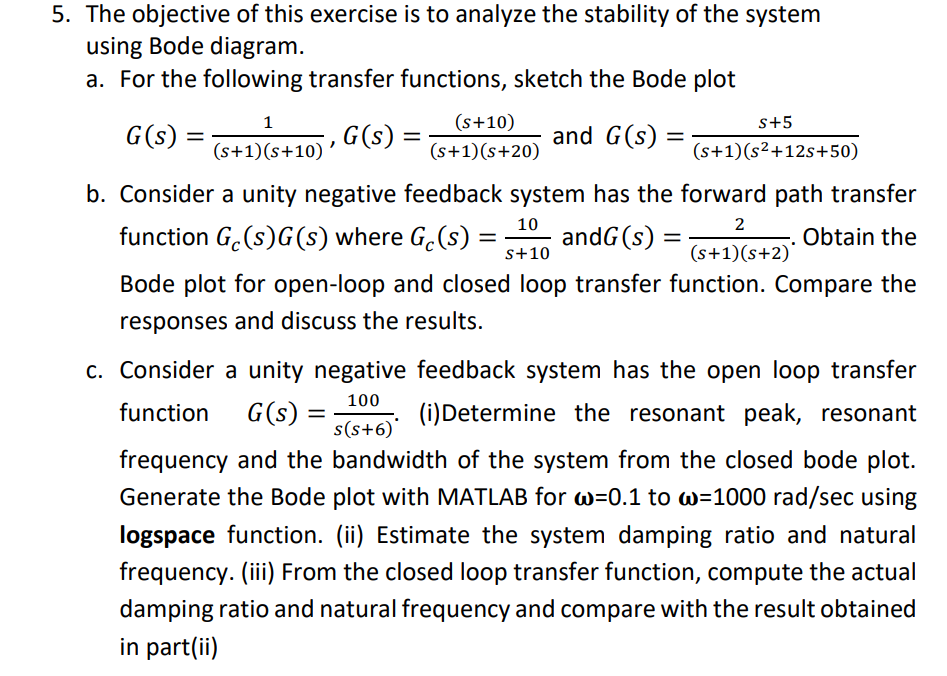
K = (sigma^2 + (2\*z\*wn - 5)\*sigma + wn^2)/1; % solve for K

% Display the value of K

disp(['The value of K is: ' num2str(K)]);



The value of K is: 47.8491

1. 

Ans:

CODE:

a.

clc;

clear all;

close all;

n1 = [1];

d1 = [1 11 10];

disp('Transfer function 1:-')

sys1 = tf(n1,d1)

bode(sys1)

n2 = [1 10];

d2 = [1 21 20];

disp('Transfer function 1:-')

sys2 = tf(n2,d2)

bode(sys2)

n3 = [1 5];

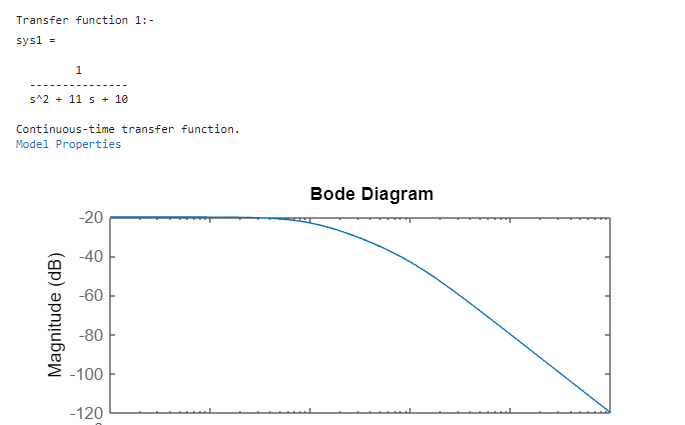
d3 = [1 13 62 50];

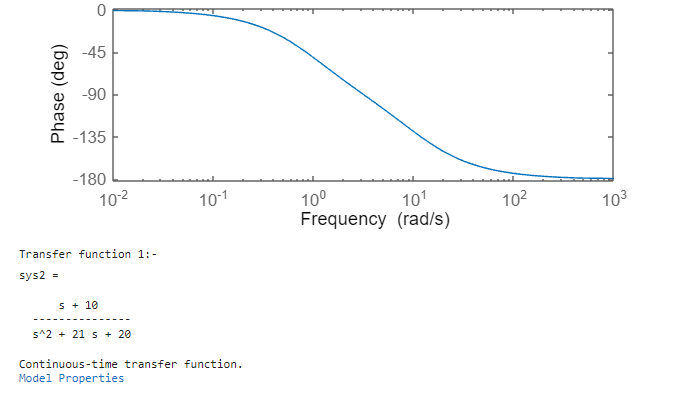
disp('Transfer function 3:-')

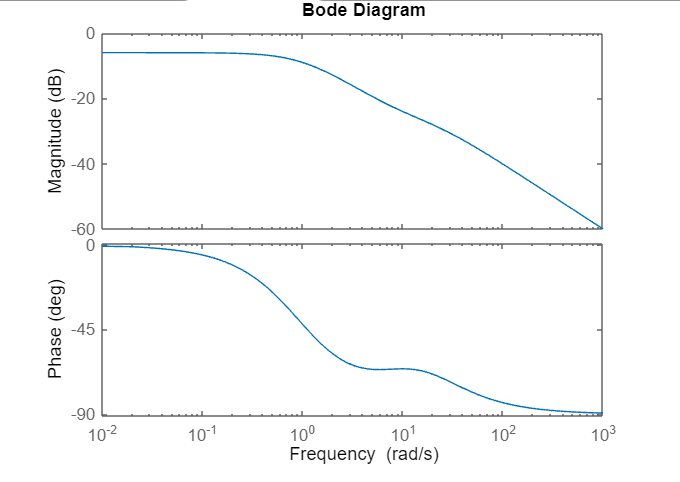
sys3 = tf(n3,d3)

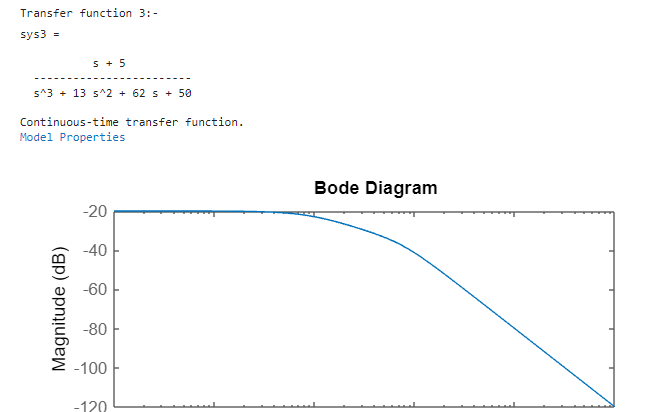
bode(sys3)

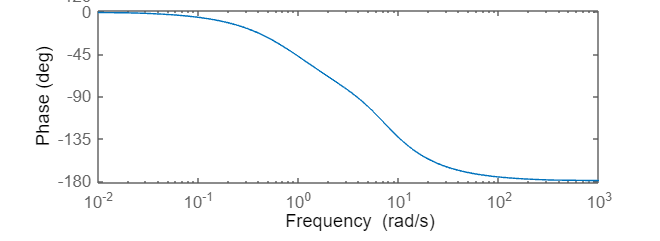
OUTPUT:











b.

clc;

clear all;

close all;

n1 = [2];

d1 = [1 3 2];

disp('Open loop Transfer Function :-')

sys1 = tf(n1,d1)

bode(sys1)

n2 = [2];

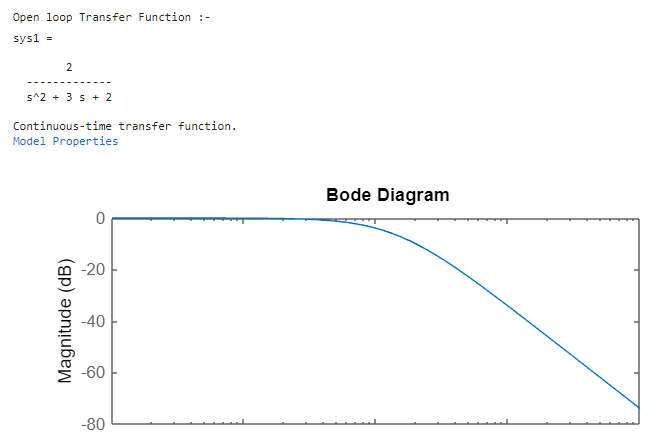
d2 = [1 3 22];

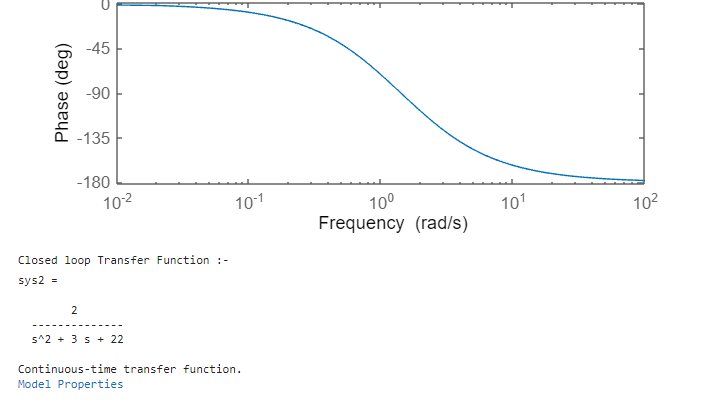
disp('Closed loop Transfer Function :-')

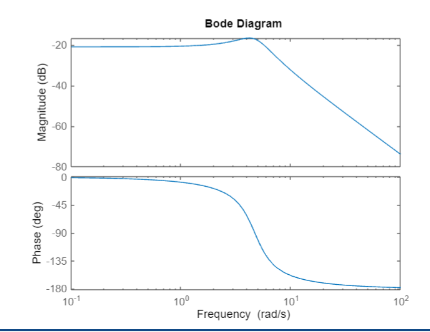
sys2 = tf(n2,d2)

bode(sys2)

OUTPUT:







c.

n1 = [100];

d1 = [1 7 106];

disp('Closed loop Transfer Function :-')

sys1 = tf(n1,d1)

bode(sys1)

disp('Bandwidth : ')

bw = bandwidth(sys1)

disp('Resonant peak : ')

[mag,freq] = bode(sys1);

magdb = 20\*log10(squeeze(mag));

[pks,locs] = findpeaks(magdb);

[peak\_mag,idx] = max(pks)

peak\_freq = freq(locs(idx));

[Wn,z]=damp(sys1);

wn=Wn(2)./(2\*pi);

disp('Resonant frequency in rad/s : ')

Wr = Wn(2).\*sqrt(1-2\*(z(2).^2)) %Resonance frequency (rad/s)

disp('Resonant frequency in Hz : ')

wr = Wr/(2\*pi) % Resonance frequency (Hz

n2 = [100];

d2 = [1 7 106];

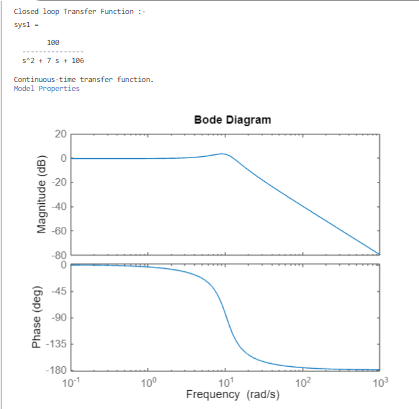
disp('Modified Bode Plot :-')

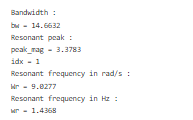
sys2 = tf(n2,d2)

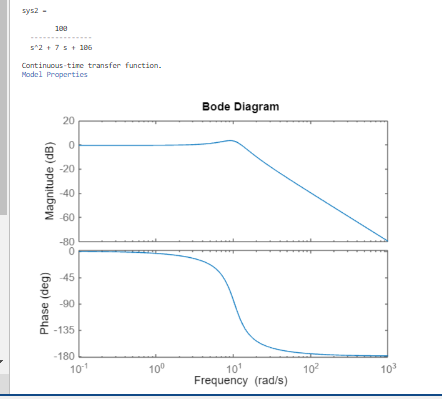
w = logspace(-1,3,1000);

bode(sys1, w)

OUTPUT :







n2 = [100];

d2 = [1 7 106];

disp('Closed loop transfer function :-')

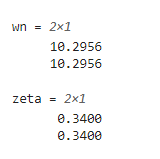
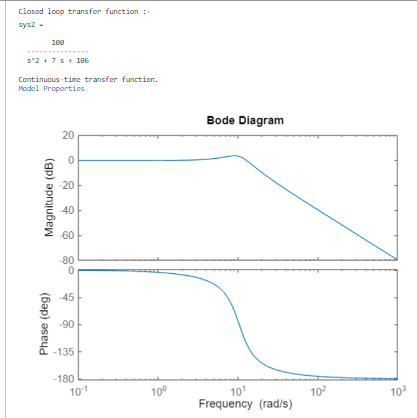
sys2 = tf(n2,d2)

%w = logspace(-1,3,1000);

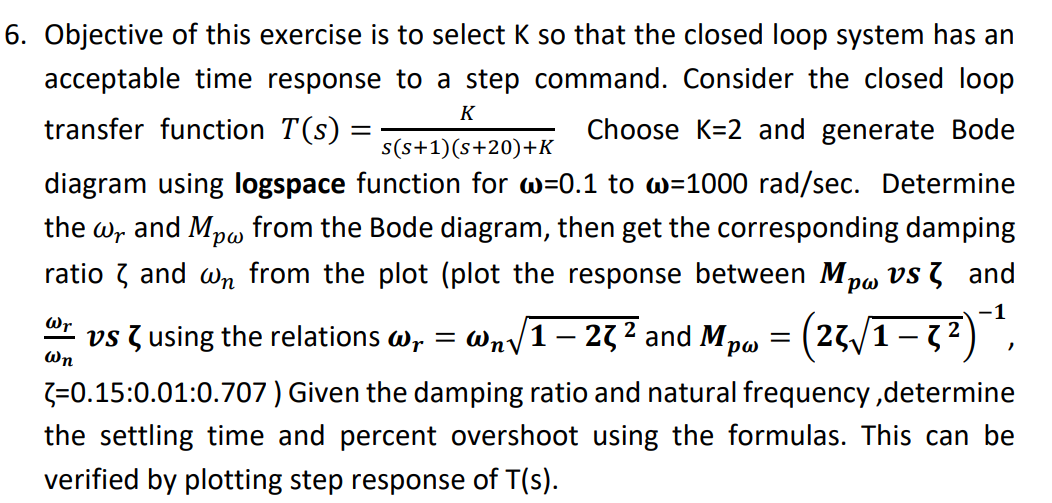
%get poles and zeroes

bode(sys2)

[wn, zeta] = damp(sys2)



Using the poles and zeroes, we can compute the actual damping ratio and natural frequency, which come very close to the approximated values

1. 

Ans:

clc;

clear;

close all;

n1 = [2];

d1 = [1 21 20 2];

disp('Closed Loop Transfer Function')

sys1 = tf(n1,d1)

%using logspace

w = logspace(-1,3,1000);

%Getting Bode plot

bode(sys1)

disp('Rise time,Fall time and other features')

stepinfo(sys1)

%To get step response

t = 0:0.1:100; % time vector

[y, t] = step(sys1, t); % step response of the transfer function

disp('Step response ')

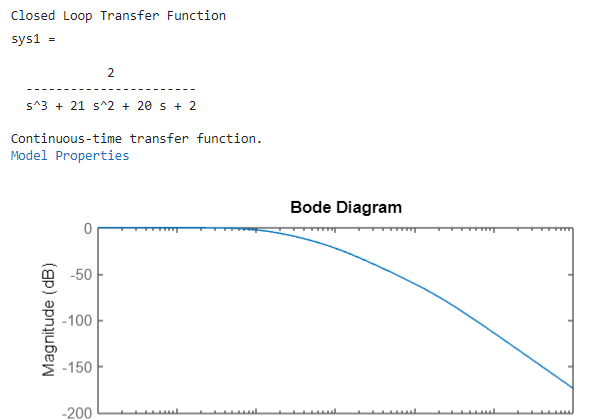
plot(t, y);

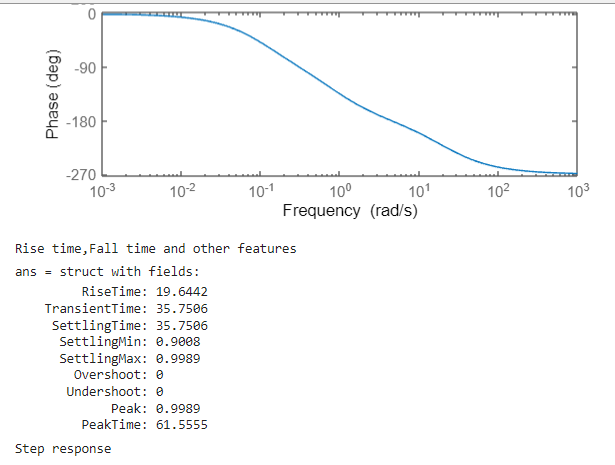
xlabel('Time');

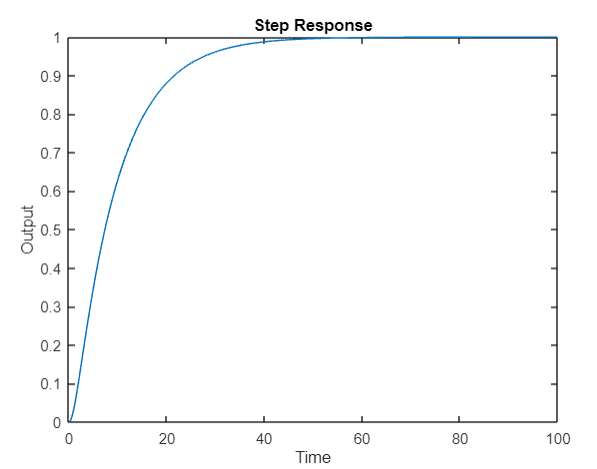
ylabel('Output');

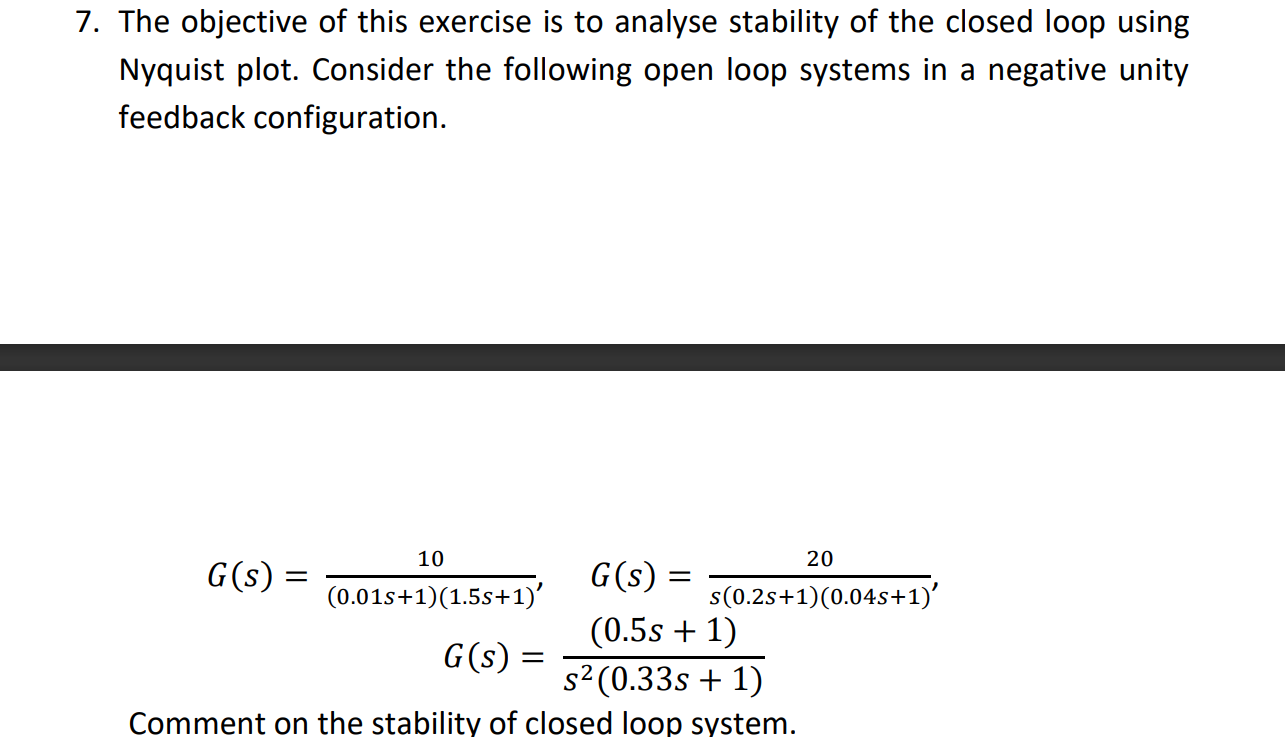
title('Step Response');

OUTPUT:







1. 

Ans:

i.

clc;

clear;

close all;

disp('Transfer function 1')

s = tf('s');

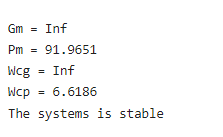
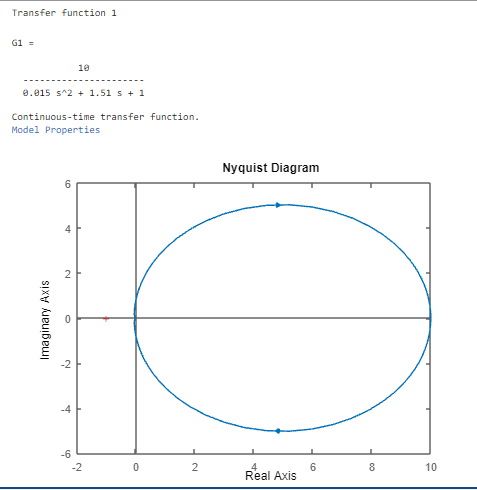
G1 = (10)/(0.015\*s^2 + 1.51\*s + 1)

nyquist(G1);

[Gm, Pm, Wcg, Wcp] = margin(G1)

disp('The systems is stable')

OUTPUT:



ii.

clc;

clear;

close all;

disp('Transfer function 2')

%n2 = [20];

%d2 = [0.008 0.24 1];

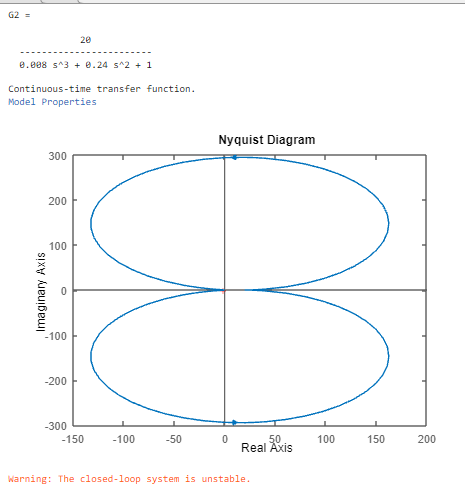
s = tf('s');

G2 = (20)/(0.008\*s^3 + 0.24\*s^2 + 1)

nyquist(G2);

[Gm, Pm, Wcg, Wcp] = margin(G2);

disp('The closed loop system is unstable')



iii.

clc;

clear;

close all;

disp('Transfer function 3')

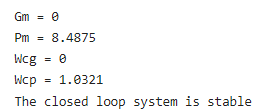
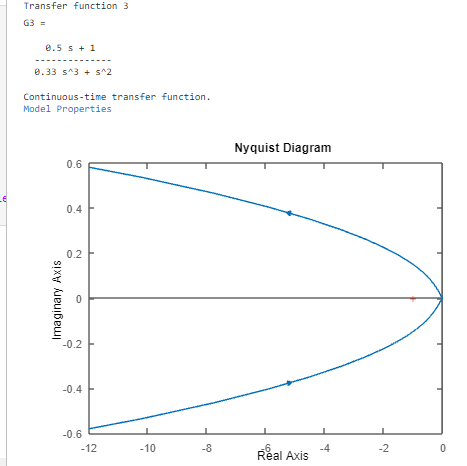
s = tf('s');

G3 = (0.5\*s + 1)/(0.33\*s^3 + s^2)

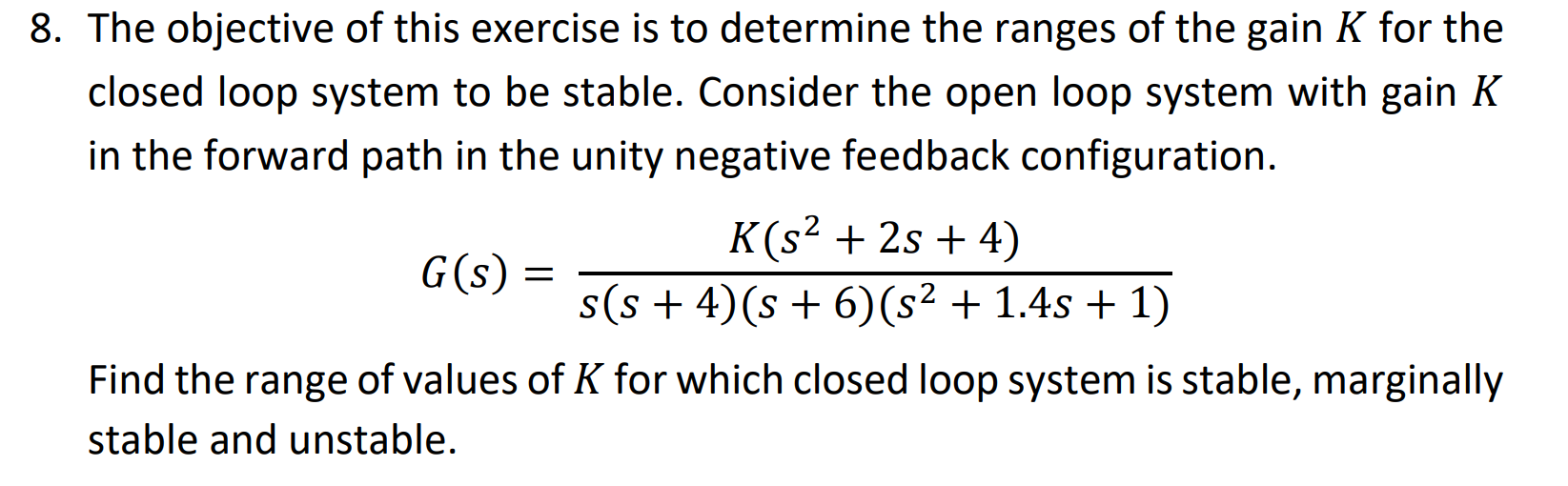
nyquist(G3);

[Gm, Pm, Wcg, Wcp] = margin(G3)

disp('The closed loop system is stable')



8.



Ans:

clc;

clear;

close all;

s = tf('s');

K = input('Enter the value of K: ');

G = K\*(s^2 + 2\*s + 4)/(s\*(s+4)\*(s+6)\*(s^2 + 1.4\*s + 1));

H = 1 + G;

T = G/H;

p = pole(T);

if real(p) < 0

fprintf('The closed-loop system is stable for all values of K.\n');

elseif real(p) == 0

fprintf('The closed-loop system is marginally stable for some values of K.\n');

else

fprintf('The closed-loop system is unstable for some values of K.\n');

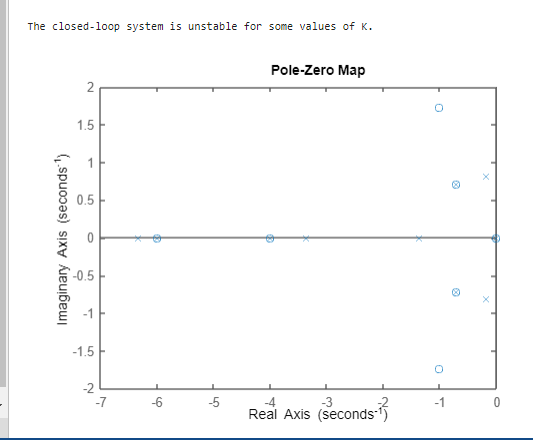
end

%plotting the poles

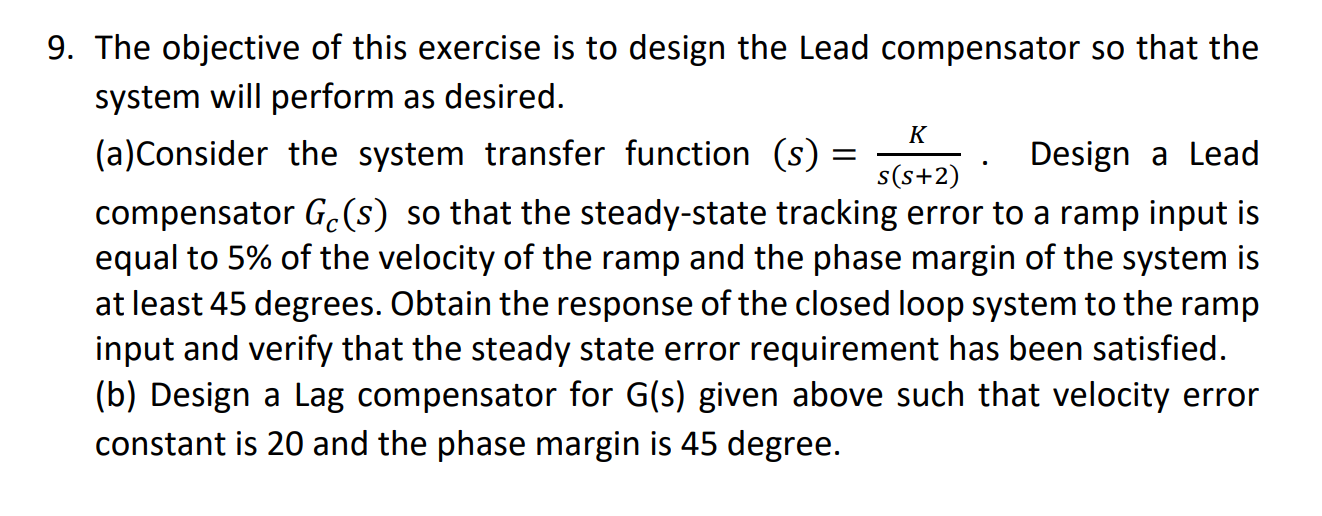
pzmap(T);

OUTPUT:





9.



Ans:

K = 40;

T = 1;

a = 2.357;

s = tf('s');

G = K\*s\*(s+2)/(s^3+2\*s^2+K\*s\*(s+2));

Ge = (1+T\*s)/(1+a\*T\*s);

Gc = Ge\*G;

Kv = limit(s\*Gc,s,0);

ess = 1/Kv;

ramp = 1/s^2;

sys\_cl = feedback(Gc\*ramp,1);

t = 0:0.01:20;

y = t;

[~,y] = step(ramp\*t);

y = y(:);

t = t(:);

error = abs(y-5\*t);

max\_error = max(error);

disp(['Max tracking error: ' num2str(max\_error)]);

if max\_error <= ess\*5

disp('Steady state error requirement satisfied.');

else

disp('Steady state error requirement not satisfied.');

end

figure;

plot(t,y);

title('Step response of the closed-loop system with Lead compensator');

xlabel('Time (s)');

ylabel('Output');

