INDEX

Sno	TOPIC	EXPT. DATE	SUB. DATE	SIGN.
1.	Different Toolboxes in MATLAB, Introduction to Control Systems Toolbox.	21/01/19	04/02/19	04/02/19
2.	Determine transpose, inverse values of given matrix.	28/01/19	04/02/19	04/02/19
3.	Plot the pole-zero configuration in s-plane for the given transfer function.	04/02/19	18/2/19	18/02/19
4.	Determine the transfer function for given closed loop system in block diagram representation.	04/02/19	18/2/19	18/02/19
5.	Plot unit step response of given transfer function and find delay time, rise time, peak time and peak overshoot.	18/2/19	11/3/19	11/03/19
6.	Determine the time response of the given system subjected to any arbitrary input.	25/2/19	11/3/19	11/03/19
7.	Plot root locus of given transfer function, locate closed loop poles for different values of k. Also find out Wd and Wnat for a given root.	11/3/19	15/4/19	
8.	Create the state space model of a linear continuous system.	15/4/19	22/4/19	
9.	Determine the State Space representation of the given transfer function.	15/4/19	22/4/19	
10.	Plot bode plot of given transfer function. Also determine the relative stability by measuring gain and phase margins.	22/4/19	22/4/19	

11.	Determine the steady state errors of a given transfer function.	21/3/19	15/04/19	
12.	Plot Nyquist plot for given transfer function and to discuss closed loop stability. Also determine the relative stability by measuring gain and phase margin	22/4/19	22/4/19	22/04/19

EXPERIMENT : 1

EXPERIMENT :. Different Toolboxes in MATLAB, Introduction to Control Systems Toolboxor its equivalent open source freeware software like Scilabusing Spoken Tutorial MOOCs.

SOFTWARE REQUIRED:- **MATLAB (R2015a)**

THEORY:-

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python.

TOOLBOXES OF MATLAB (R2015a)

1) Control System Toolbox

2)Curve Fitting Toolbox

3)Database Toolbox

4)Signal Processing Toolbox

5)Econometrics Toolbox

CONTROL SYSTEM TOOLBOX-

Control System Toolbox provides industry-standard algorithms and apps for systematically analyzing, designing, and tuning linear control systems. You can specify your system as a transfer function, state-space, zero-pole-gain or frequency-response model. Apps and functions, such as step response plot and Bode plot, let you visualize system behavior in time domain and frequency domain. You can tune compensator parameters using automatic PID controller tuning, Bode loop shaping, root locus method, LQR/LQG design, and other interactive and automated techniques. You can validate your design by verifying rise time, overshoot, settling time, gain and phase margins, and other requirements.

GENERAL COMMANDS OF MATLAB-

clc - Clears Command window.

clear - Removes variables from memory.

det - Computes determinant of an array.

inv - Computes inverse of a matrix.

grid - Displays gridlines.

plot - Generates xy plot.

print - Prints plot or saves plot to a file.

title - Puts text at top of plot.

xlabel - Adds text label to x-axis.

ylabel - Adds text label to y-axis.

subplot - Creates plots in subwindows.

semilogx - Creates semilog plot (logarithmic abscissa).

semilogy - Creates semilog plot (logarithmic ordinate).

loglog - Creates log-log plot.

roots - Computes polynomial roots.

COMMANDS OF CONTROL SYSTEM TOOLBOX-

abs - Absolute value

axis - Set the scale of the current plot

c2d - Continuous system to discrete system

conv - Convolution (useful for multiplying polynomials)

deconv - Deconvolution and polynomial division

eig - Compute the eigenvalues of a matrix

feedback - Connect linear systems in a feedback loop

for - For loop

grid - Draw the grid lines on the current plot

help - Matlab help documentation

hold - Hold the current graph

if - Conditionally execute statements

inv -Find the inverse of a matrix

log - Natural logarithm

pzmap - Pole-zero map of linear systems

roots - Find the roots of a polynomial

sqrt - Square root

step - Plot the step response

tf - Creation of transfer functions or conversion to transfer function

zeros - Returns a vector or matrix of zeros

zgrid - Generates grid lines of constant damping ratio (zeta) and natural frequency (Wn)

04/02/19

EXPERIMENT 02(a)

AIM: To plot basic signals

SOFTWARE: MATLAB R216a

THEORY:

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python.

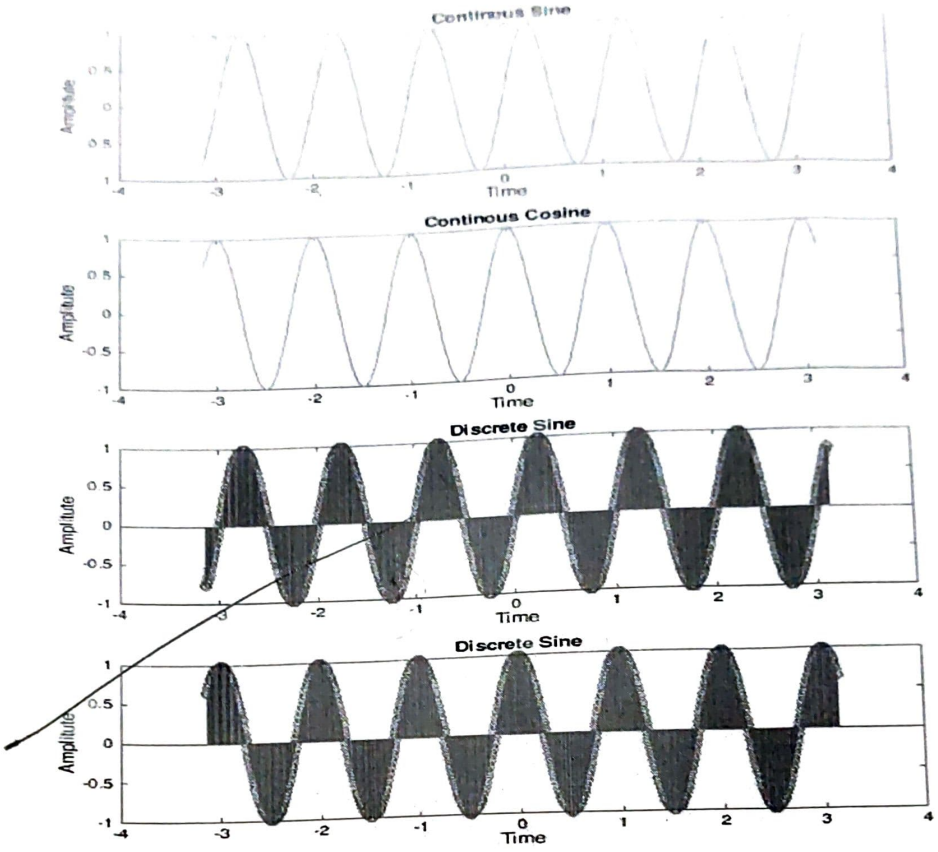
Basic signal functions: Y = sin(X) returns the sine of the elements of X. The sin function operates element-wise on arrays. The function accepts both real and complex inputs Y = cos(X) returns the cosine for each element of X. The cos function operates element-wise on arrays. The function accepts both real and complex inputs.

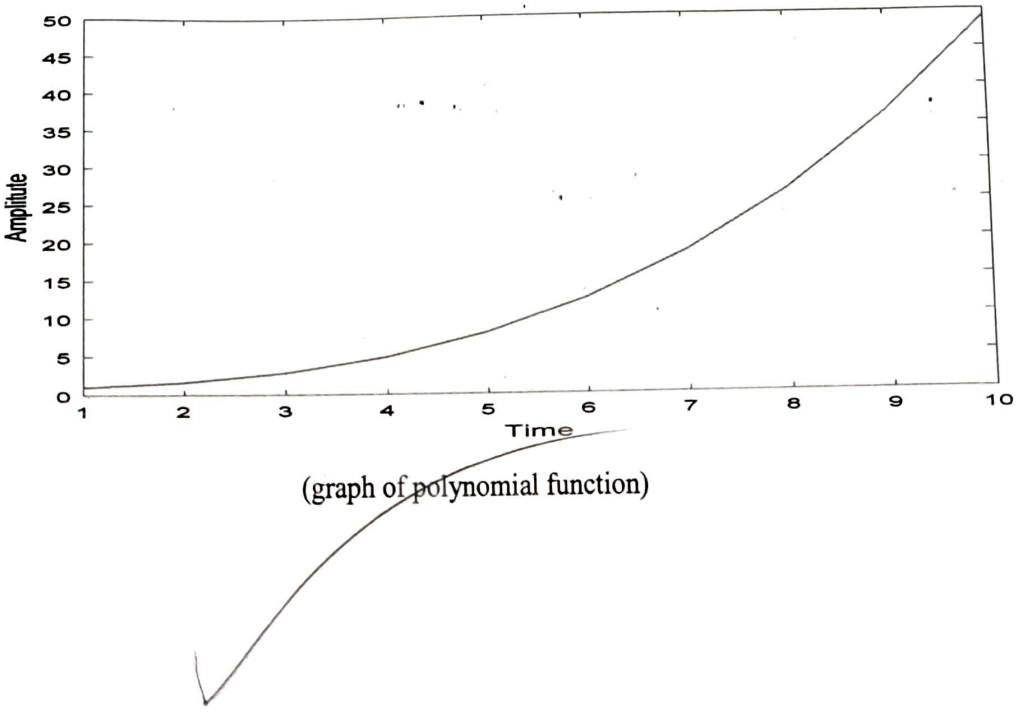
PROGRAM CODES:

a) To plot basic signals

```
clc;
close all;
t=0:0.1:2*pi;
a=sin(2*pi*t);
subplot(4,2,1)
plot(t,a)
title('Continous Sine')
xlabel('Time')
ylabel('Amplitute')
subplot(4,2,2)
stem(t,a)
title('Discrete Sine')
xlabel('Time')
ylabel('Amplitute')
subplot(4,2,3)
b=cos(2*pi*t);
subplot(4,2,4)
plot(t,b)
title('Continous Cosine')
xlabel('Time')
ylabel('Amplitute')
stem(t,b)
title('Discrete cosine')
xlabel('Time')
ylabel('Amplitute')
x=linspace(0,2,10);
y1=(4.*x.*x.*x)+(3.*x.*x)+(2.*x)+1;
subplot(4,2,5)
plot(y1)
xlabel('Time')
ylabel('Amplitute')
```

OUTPUT





(graph of polynomial function)

EXPERIMENT 02 (b)

AIM: Basic matrix operation

i) choosing an element and deletion & selection of a row of a matrix,

ii) matrix multiplication,

iii) Inverse of a matrix,

iv) transpose of a matrix.

SOFTWARE: MATLAB R216a

THEORY:

Basic matrix operations:

C = A*B is the matrix product of A and B. If A is an m-by-p and B is a p-by-n matrix, then C is an m-by-n matrix. This definition says that C (i, j) is the inner product of the ith row of A with the jth column of B.

C = A.*B is the element by element product of two matrix.

d = det(A) returns the determinant of square matrix A.

i) choosing an element and deletion & selection of a row of a matrix,

```
clc;
close all;
clear all;
A=[1 2 3 4; 5 6 7 8; 9 3 4 5; 8 3 5 7];
B=A(3,3);
C=A(3:4,1:4)
A(3:4,:)=[]
```

OUTPUT

```
C =

     9     3     4     5
     8     3     5     7


A =

     1     2     3     4
     5     6     7     8


B =

     4
```

ii) matrix multiplication,

```
clc;
close all;
clear all;
A=[1 2 3; 4 5 6; 7 8 9];
B=[9 8 7; 6 5 4; 3 2 1];
c=A.*B;
d=A*B;
```

OUTPUT

```
c =

     9    16    21
    24    25    24
    21    16     9

d =

    30    24    18
    84    69    54
   138   114    90
```

iii) transpose of matrix

```
clc;
close all;
clear all;
A=[1 2 3;4 5 6; 7 8 9];
for i=1:3
    for j=1:3
        B(i,j)=A(j,i);
    end
end
```

OUTPUT

```
B =

     1     4     7
     2     5     8
     3     6     9
```

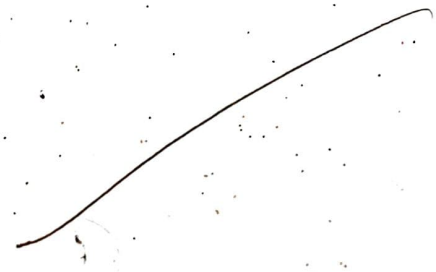
iv) Inverse of a matrix,

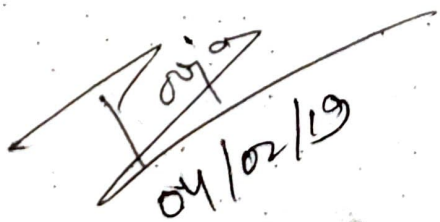
```
clc;
close all;
clear all;
A=[1 2; 3 4];
B=det(A);
for i=1:2
    for j=1:2
        D(i,j)=((-1)^(i+j))*(A(3-i,3-j));
```

```
            I(i,j)=(D(i,j)/B);
            C=I.';
        end
    end
```

OUTPUT

```
I =

   -2.0000    1.5000
    1.0000   -0.5000

C =

   -2.0000    1.0000
    1.5000   -0.5000
```





EXPERIMENT No 3

04/02/19

- **AIM**:-Plot the pole-zero configuration in splane for the given transfer function.
- **SOFTWARE USED** :- MATLAB R2015a
- **THEORY** :-

$$H(s)=P(s)/Q(s)$$

The two polynomials, $P(s)$ and $Q(s)$, allow us to find the poles and zeros of the Laplace-Transform.

Definition 1: zeros

The value(s) for s where $P(s)=0$.
The complex frequencies that make the overall gain of the filter transfer function zero.

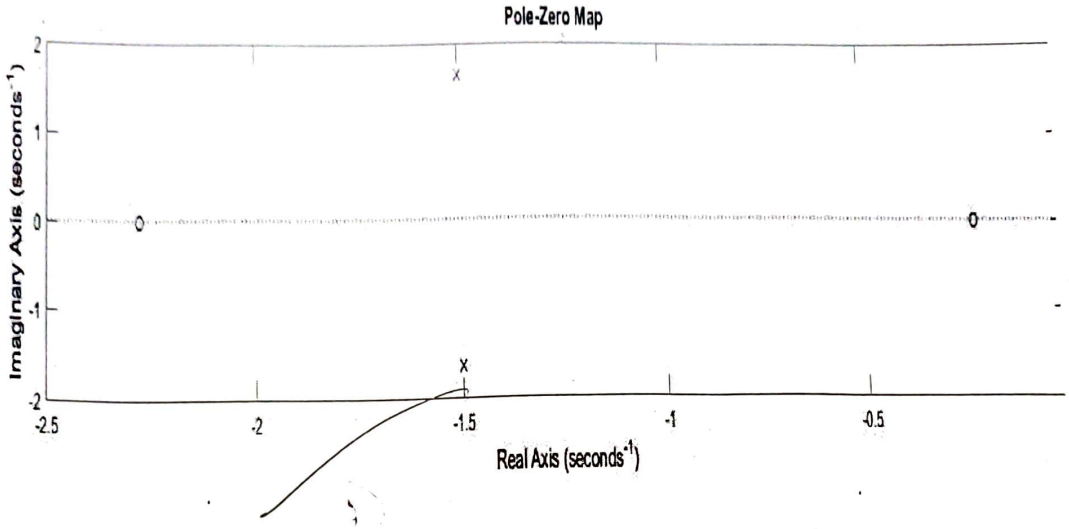
Definition 2: poles

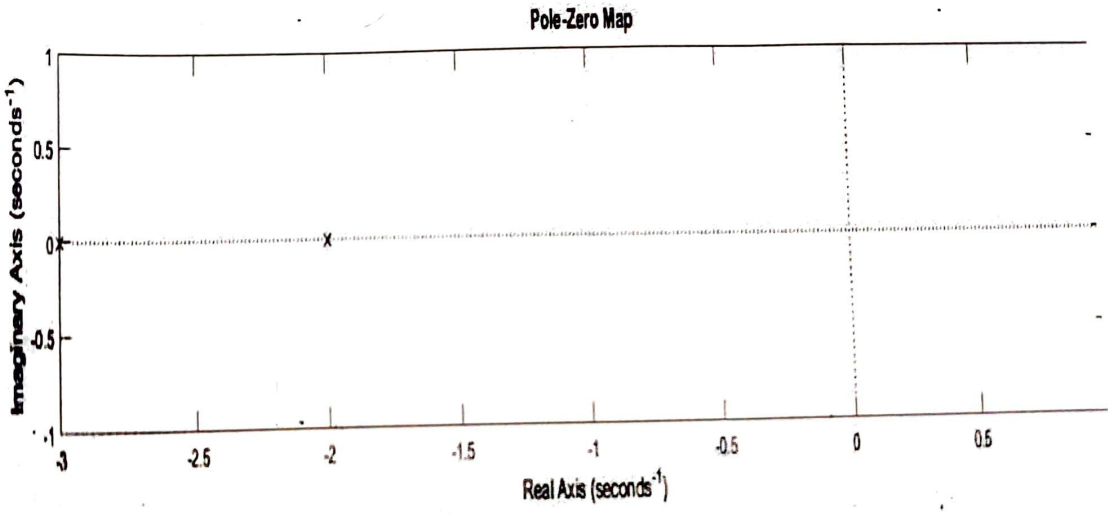
The value(s) for s where $Q(s)=0$.
The complex frequencies that make the overall gain of the filter transfer function infinite.

PROGRAM :-

```
clear all;
close all;
clc;
H= tf([2 5 1], [1 3 5]);
Y= zpk(1,[-2 -3],6);
subplot(2,1,1)
pzmap(H)
subplot(2,1,2)
pzmap(Y)
```

RESULT: -






```
a=[1 2 3]
b=[1 2 4]
[z,p,k]=tf2zpk(a,b)
```

RESULT: -

```
a =   1   2   3

b =   1   2   4
```

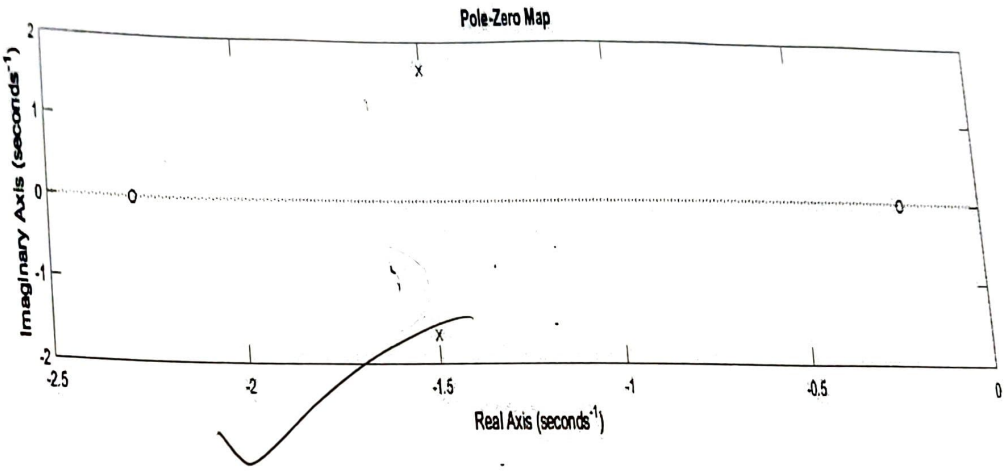
z = -1.0000 + 1.4142i

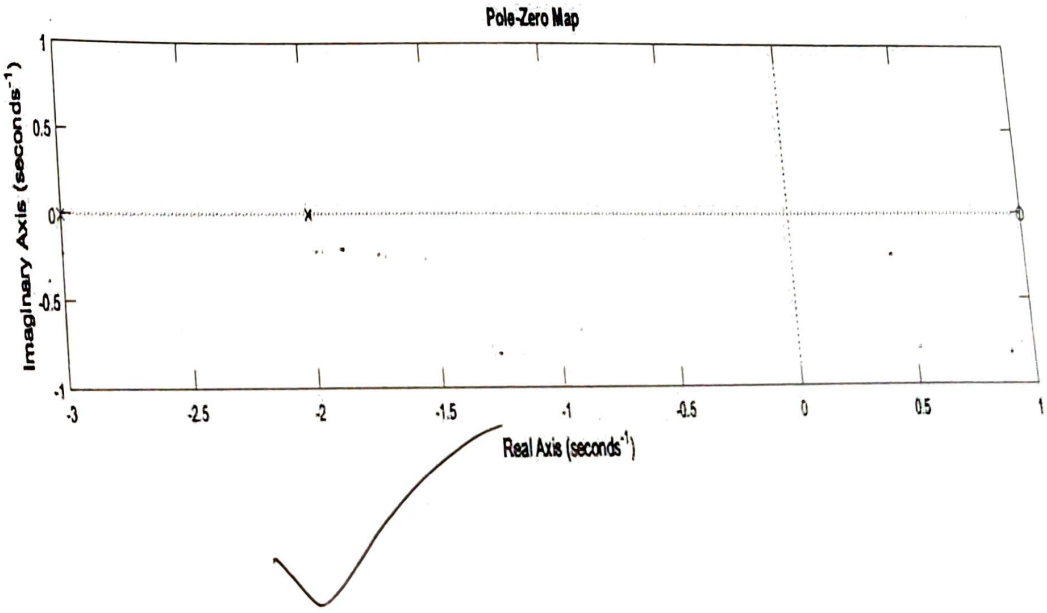
-1.0000 - 1.4142i

p = -1.0000 + 1.7321i

-1.0000 - 1.7321i

k = 1





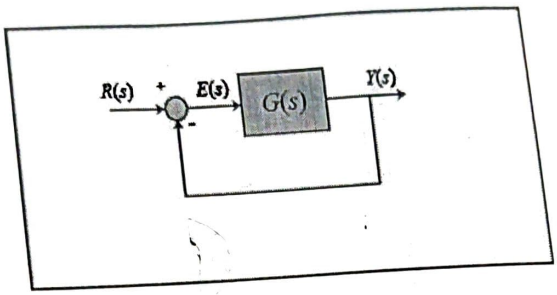
18/02/19

EXPERIMENT No#4

04/02/19

- **AIM**:-Determine the transfer function for given closed loop system in block diagram representation.
- **SOFTWARE USED** :- MATLAB R2015a
- **THEORY** :-

Block diagram: It is used to represent all types of systems. It can be used, together with transfer functions, to describe the cause and effect relationships throughout the system.



- **PROGRAM** :-

```
clc
clear all
close all
sys1=tf([0 0 1],[0 1 0])
sys2=tf([0 0 1],[0 1 1])
sys3=tf([0 0 1],[1 0 0])
sys4=tf([0 0 1],[0 0 1])
sys5=tf([0 0 1],[0 0 1])
sys6=series(sys1, sys2)
sys7=parallel(sys3,sys4)
sys8=series(sys6,sys7)
sys9=feedback(sys8,sys5)
pzmap(sys9)
```

RESULT:-

```
sys1 =

  1
  -
  s

Continuous-time transfer function.
```

sys2 =

1

---

s + 1

Continuous-time transfer function.

sys3 =

1

---

s^2

Continuous-time transfer function.

sys4 =

1

Static gain.

sys5 =

1

Static gain.

sys6 =

1

---

s^2 + s

Continuous-time transfer function.

sys7 =

s^2 + 1

---

s^2

Continuous-time transfer function.

sys8 =

s^2 + 1

---

s^4 + s^3

Continuous-time transfer function.

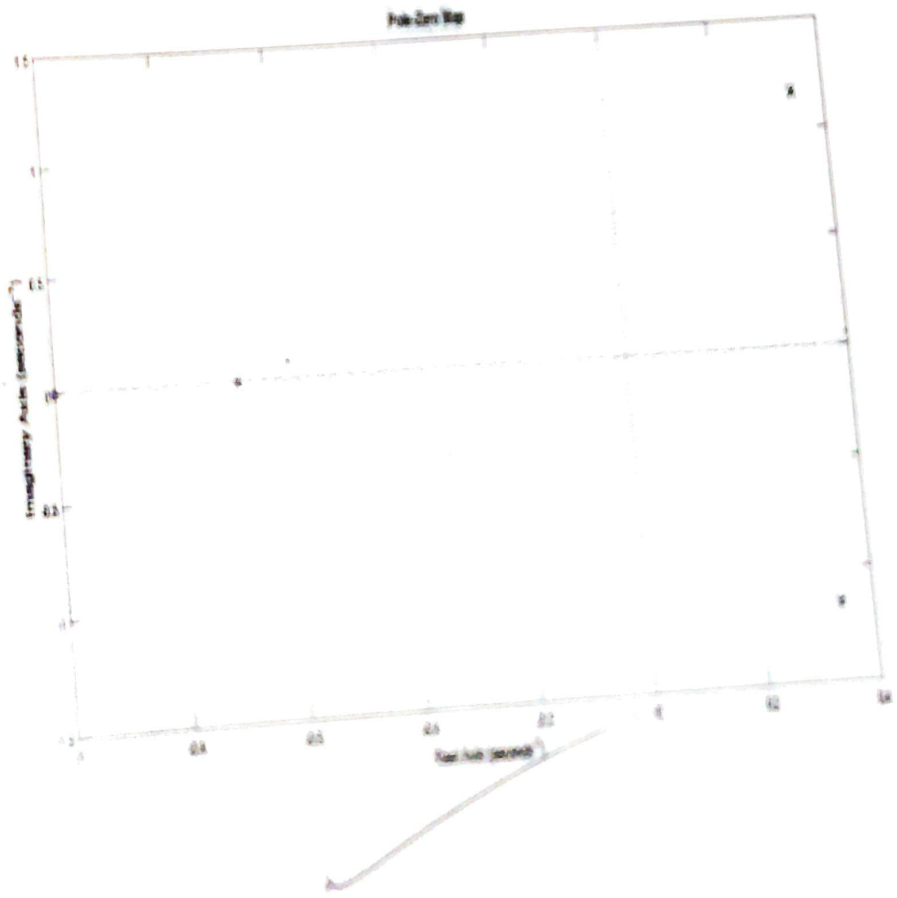
sys9 =

s^2 + 1

---

s^4 + s^3 + s^2 + 1

Continuous-time transfer function.



```
clc
clear all
close all
sys1=tf([1 0 1],[0 1 0])
sys2=tf([0 0 1],[0 1 0])
sys3=tf([0 0 1],[1 0 1])
sys4=tf([1 0 1],[0 1 0])
sys5=tf([0 0 1],[0 0 1])
sys6=series(sys1, sys2)
sys7=parallel(sys3,sys4)
sys8=parallel(sys6,sys7)
sys9=feedback(sys8,sys5)
pzmap(sys9)
```

RESULT:-

sys1 =

s^2 + 1

------

s

Continuous-time transfer function.

sys2 =

1

-

s

Continuous-time transfer function.

sys3 =

1

------

s^2 + 1

Continuous-time transfer function.

sys4 =

s^2 + 1

---

s

Continuous-time transfer function.

sys5 =

1

Static gain.

sys6 =

s^2 + 1

---

s^2

Continuous-time transfer function.

sys7 =

s^4 + 2 s^2 + s + 1

---

s^3 + s

Continuous-time transfer function.

sys8 =

s^6 + s^5 + 2 s^4 + 3 s^3 + s^2 + s

---

s^5 + s^3

Continuous-time transfer function.

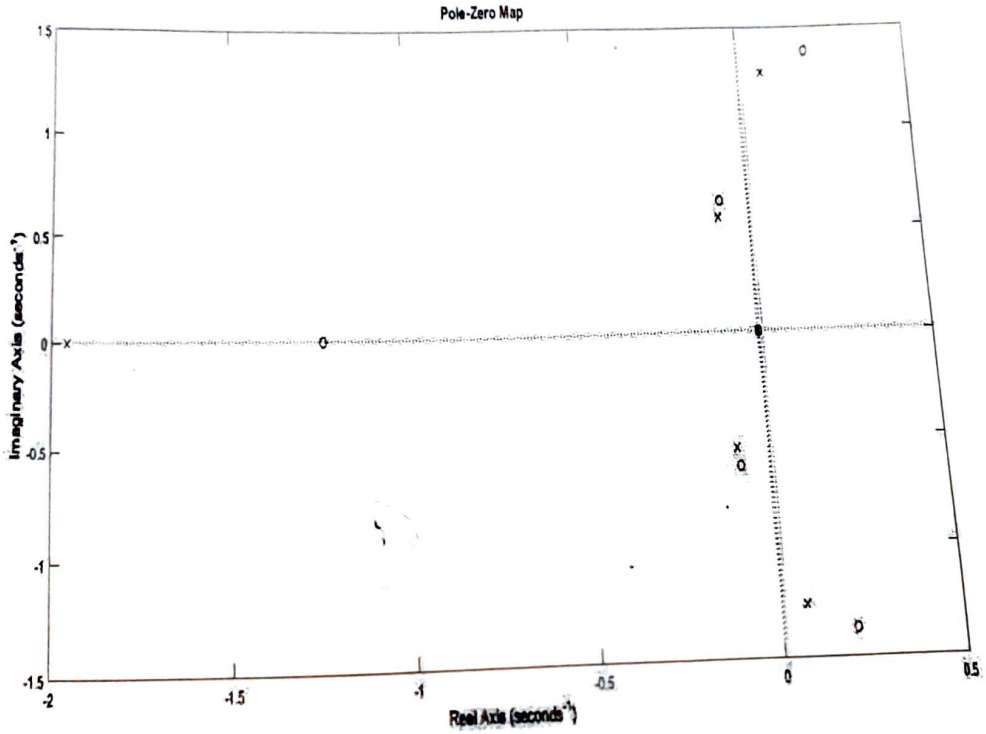
sys9 =

s^6 + s^5 + 2 s^4 + 3 s^3 + s^2 + s

---

s^6 + 2 s^5 + 2 s^4 + 4 s^3 + s^2 + s

Continuous-time transfer function.


Pole-Zero Map
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1.5
-1
-0.5
0
0.5

18/02/19

Experiment No-05

AIM: Write a Program to plot unit step response of a given transfer function and find delay time, rise time, peak time and peak overshoot.

Software Used: MATLAB R2018b

Theory: One of the most common test inputs used is the unit step function,

$$\gamma(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases}$$

$$\Gamma(s) = \frac{1}{s}$$

The response of a system (with all initial conditions equal to zero at t=0, i.e., a zero response) to the unit step input is called the unit step response.

Delay Time: Delay Time is defined as a delay that separates the occurrence of two events.

Rise Time: The time required for a pulse to rise from 10 per cent to 90 per cent of its steady value.

Peak Time: The peak time is the time required for the system response to reach the "first peak" of overshoot.

Program:

```
clc
clear all
close all
a1=tf([4],[1 4 4])
s1=stepinfo(a1)
subplot(2,2,1)
step(a1)
a2=tf([4],[1 6 4])
s2=stepinfo(a2)
subplot(2,2,2)
step(a2)
a3=tf([36],[1 0 36])
s3=stepinfo(a3)
subplot(2,2,3)
step(a3)
a4=tf([4],[1 2 4])
s4=stepinfo(a4)
subplot(2,2,4)
step(a4)
```

Output:

```
a1 =

     4
  ------------
  s^2 + 4 s + 4
Continuous-time transfer function.
s1 =
  struct with fields:
      RiseTime: 1.6790
    SettlingTime: 2.9170
     SettlingMin: 0.9008
     SettlingMax: 0.9991
      Overshoot: 0
     Undershoot: 0
          Peak: 0.9991
      PeakTime: 4.6900

a2 =

     4
  -------------
  s^2 + 6 s + 4

Continuous-time transfer function.
s2 =
  struct with fields:
      RiseTime: 2.9292
    SettlingTime: 5.3274
     SettlingMin: 0.9012
     SettlingMax: 0.9999
      Overshoot: 0
     Undershoot: 0
          Peak: 0.9999
      PeakTime: 12.9891
a3 =

    36
  --------
  s^2 + 36

Continuous-time transfer function.
s3 =
  struct with fields:
      RiseTime: NaN
    SettlingTime: NaN
```

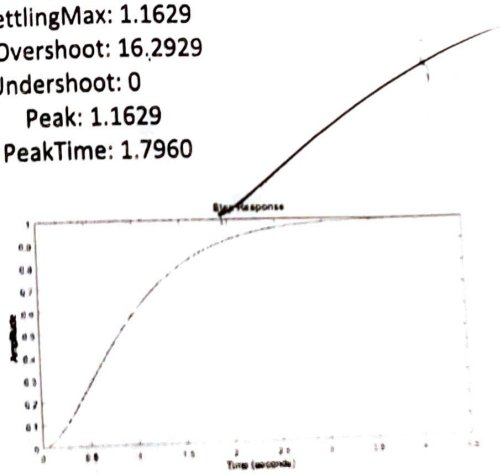
SettlingMin: NaN
SettlingMax: NaN
Overshoot: NaN
Undershoot: NaN
Peak: Inf
PeakTime: Inf

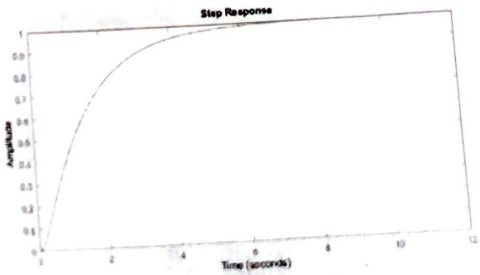
a4 =

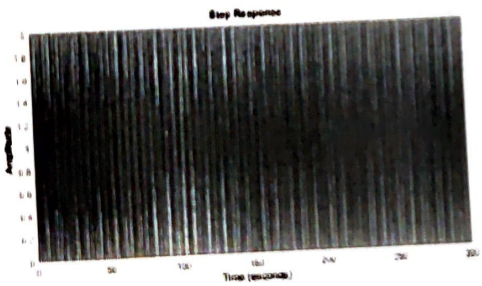
4
---
s^2 + 2 s + 4

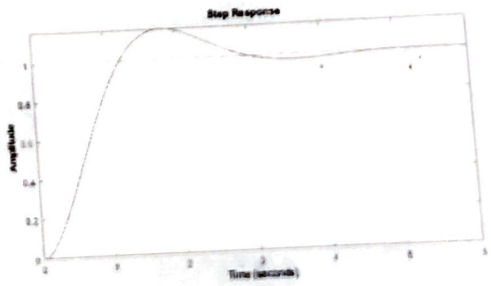
Continuous-time transfer function.

s4 =
struct with fields:
RiseTime: 0.8195
SettlingTime: 4.0379
SettlingMin: 0.9315
SettlingMax: 1.1629
Overshoot: 16.2929
Undershoot: 0
Peak: 1.1629
PeakTime: 1.7960









Result: We have plotted unit step response and calculated the values of delay time, rise time , peak time and peak overshoot.

11/03/19

EXPERIMENT 06

AIM:

Determine the time response of the given system subjected to any arbitrary input.

SOFTWARE: MATLAB R2016

THEORY:

Basic operations:

tf- Use tf to create real- or complex-valued transfer function models (tfobjects) or to convert state-space or zero-pole-gain models to transfer function form.

step- step calculates the step response of a dynamic system. For the state-space case, zero initial state is assumed. When it is invoked with no output arguments, this function plots the step response on the screen.

Impulse(sys)-impulse(sys) plots the impulse response of the dynamic system model sys. This model can be continuous or discrete, and SISO or MIMO.

lsim(sys,u,t)- lsim(sys,u,t) produces a plot of the time response of the dynamic system model sys to the input history, t,u. The vector t specifies the time samples for the simulation (in system time units, specified In the TimeUnit property of sys), and consists of regularly spaced time samples:

PROGRAM CODE:

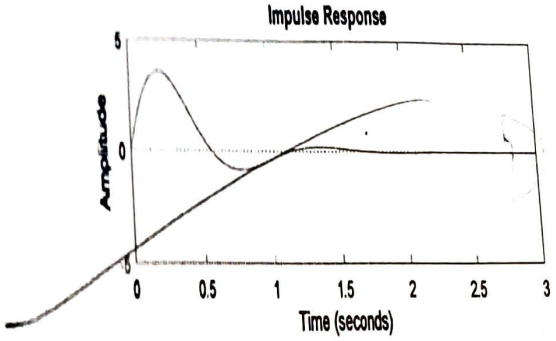
```
clc
close all;
clear all;
t=0:0.01:10;
sys=tf(36,[1 5 36])
%impulse response
subplot(3,2,1)
impulse(sys);
%step response
subplot(3,2,2)
step(sys)
subplot(3,2,3)
%linear time response
u1=2*t
lsim(sys,u1,t);
subplot(3,2,4)
%parabolic response
u=5*t.*t;
lsim(sys,u,t);
%sinosuidal response
subplot(3,2,5)
a=sin(t)
lsim(sys,a,t);
%polynomial input response
x=5*t.*t+2*t+1
subplot(3,2,6)
```

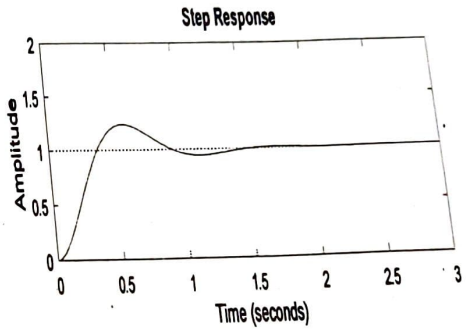
```
lsim(sys3,x,t);
```

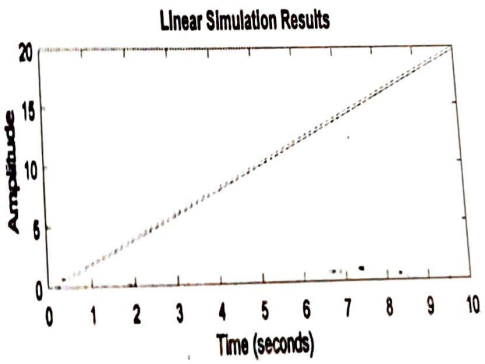
OUTPUT:

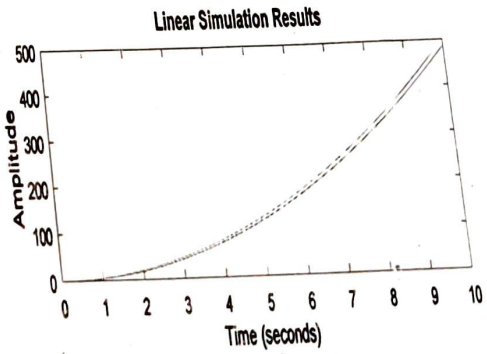
```
  36
  ---------------
  s^2 + 5 s + 36

Continuous-time transfer function.
```

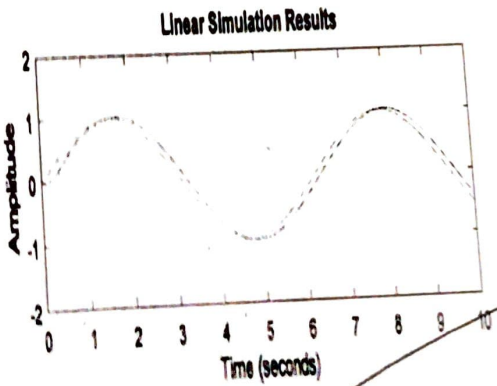
FIGURE:

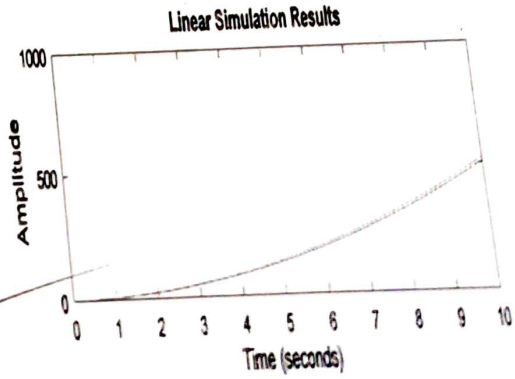












EXPERIMENT NO#7

01/04/19

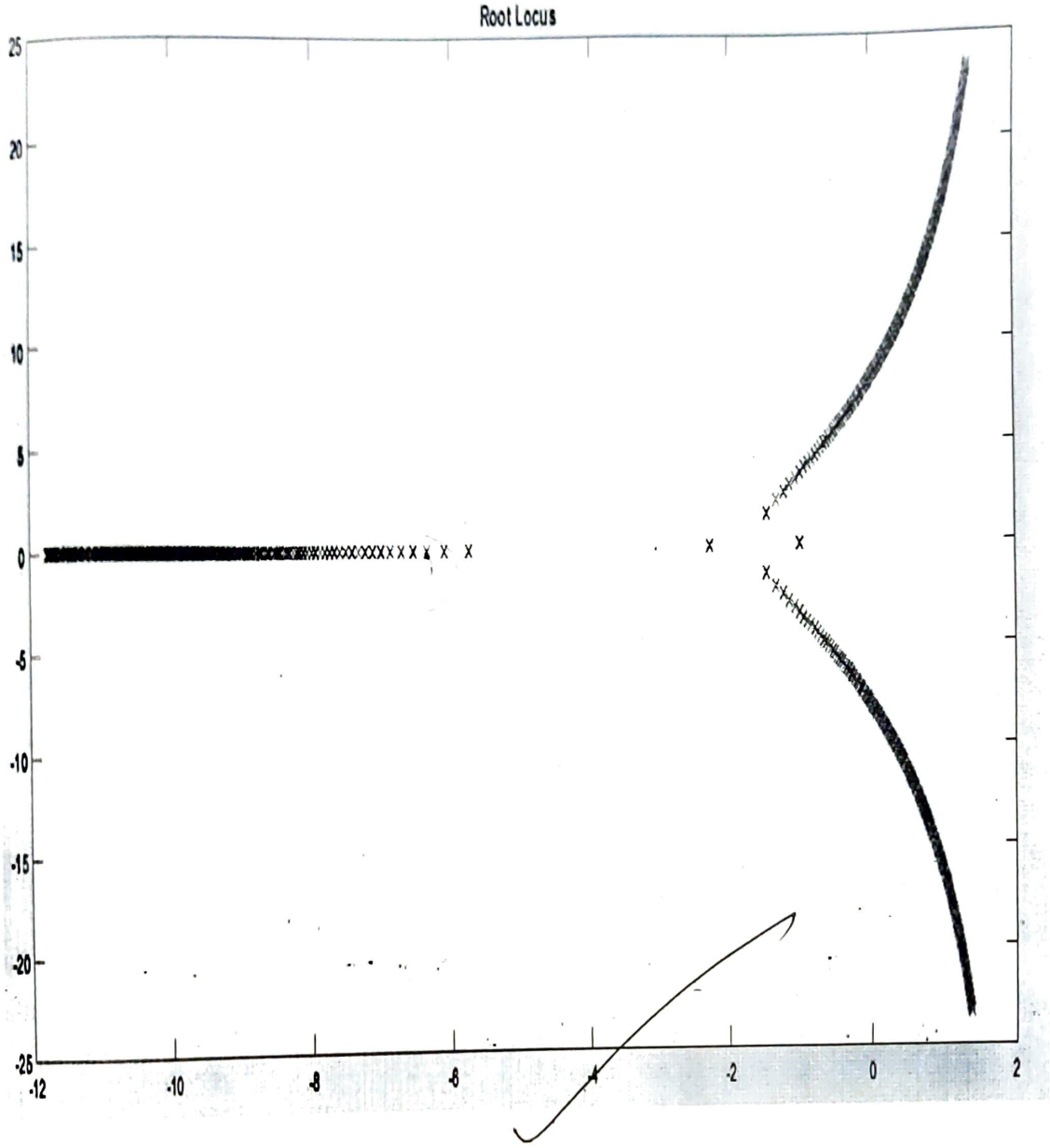
- **AIM**:-Plot root locus of given transfer function, locate closed loop poles for different values of k. Also find out Wd and Wnat for a given root.
- **SOFTWARE USED** :- MATLAB R2015a.
- **THEORY** :- In control theory and stability theory, **root locus analysis** is a graphical method for examining how the roots of a system change with variation of a certain system parameter, commonly a gain within a feedback system. This is a technique used as a stability criterion in the field of classical control theory developed by Walter R. Evans which can determine stability of the system. The root locus plots the poles of the closed loop transfer function in the complex *s*-plane as a function of a gain parameter (see pole–zero plot).

  - The root locus of a feedback system is the graphical representation in the complex *s*-plane of the possible locations of its closed-loop polesfor varying values of a certain system parameter. The points that are part of the root locus satisfy the angle condition. The value of the parameter for a certain point of the root locus can be obtained using the magnitude condition.

- **PROGRAM** :-

```
clc
clear all
close all
for k=1:500
   zeros=[-13]
   poles=[0 -4 -5]
   G=zpk(zeros,poles,k)
   sys=feedback(G,1)
   p=pole(sys)
p_real=real(p)
p_img=imag(p)
   plot(p_real,p_img,'X')
   hold on;
end
title('Root Locus')
```

RESULT:-



EXPERIMENT NO#11

01/04/19

- **AIM**:-Determine the steady state errors of a given transfer function.
- **SOFTWARE USED** :- MATLAB R2015a.
- **THEORY** :-

The deviation of the output of control system from desired response during steady state is known as **steady state error**. It is represented as essess. We can find steady state error using the final value theorem as follows.

$$e_{ss} = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s)$$

Where,E(s) is the Laplace transform of the error signal, e(t)

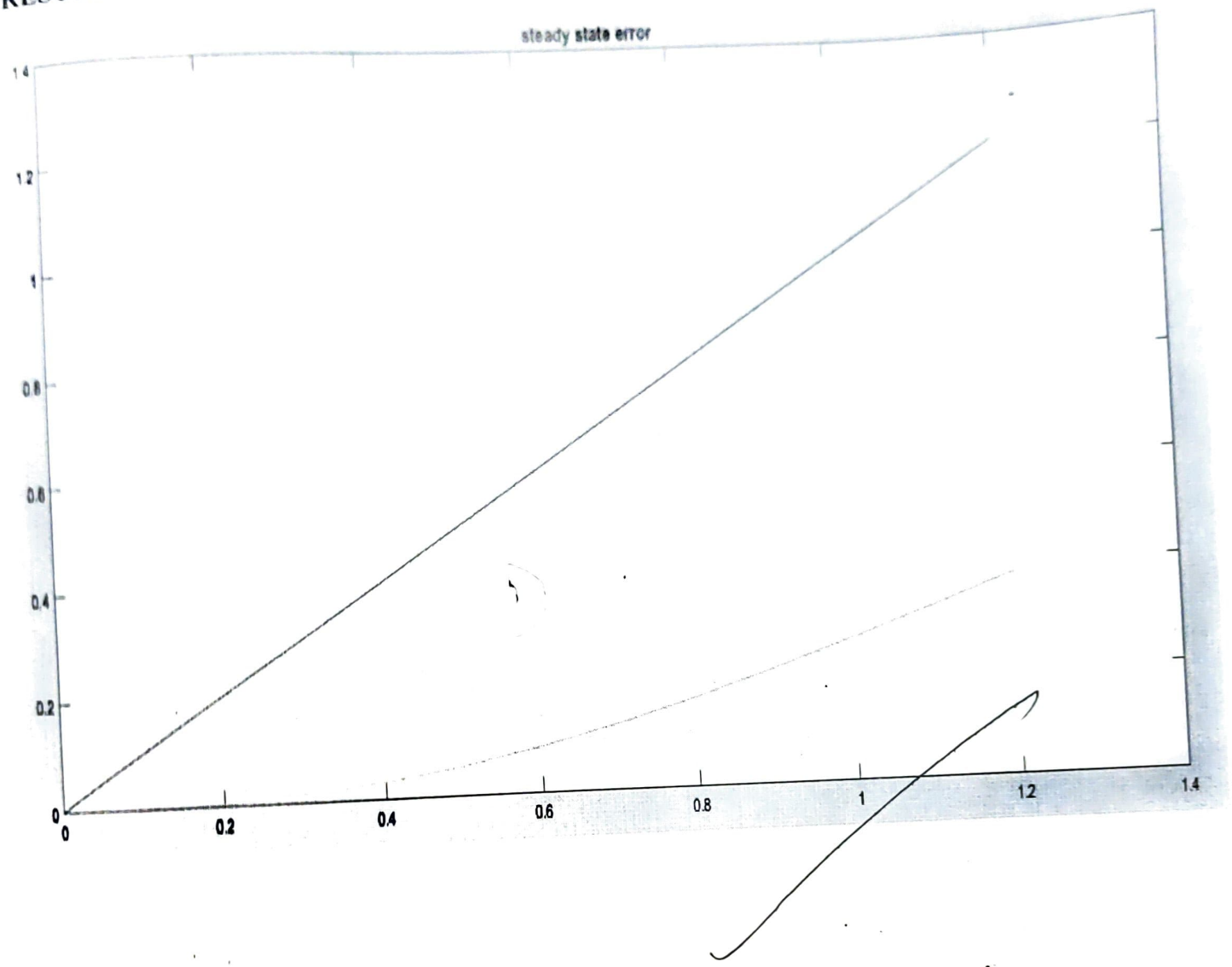
Input signal	Steady state error e_{ss}	Error constant
unit step signal	$\frac{1}{1+k_p}$	$K_p = \lim_{s\to 0} G(s)$
unit ramp signal	$\frac{1}{K_v}$	$K_v = \lim_{s\to 0} sG(s)$
unit parabolic signal	$\frac{1}{K_a}$	$K_a = \lim_{s\to 0} s^2G(s)$

Where, KpKp, KvKv and KaKa are position error constant, velocity error constant and acceleration error constant respectively.

- **PROGRAM** :-

```
clc
clear all
close all
sys=tf([4],[1 3 4])
T=feedback(sys,1)
t=0:0.1:1.25;
u=t
[y,t,x]=lsim(T,u,t);
plot(t,y,'y',t,u,'m')
title('steady state error')
xlable('time')
ylable('amplitude')
```

RESULT:-



EXPERIMENT: 8

AIM: Create the state space model of a linear continuous system

SOFTWARE REQUIRED: Matlab(2015)

THEORY: In control engineering, a **state-space representation** is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations or difference equations. State variables are variables whose values evolve through time in a way that depends on the values they have at any given time and also depends on the externally imposed values of input variables. Output variables' values depend on the values of the state variables.

The "state space" is the Euclidean space in which the variables on the axes are the state variables. The state of the system can be represented as a vector within that space.

PROGRAM CODE:

```
clear all
close all
clc
b=[0 2 3]
a=[1 0 .4 1]
[A B C D]=tf2ss(b,a)
```

0UTPUT:

```
A=      0   -.4   -1
        1     0    0
        0     1    0

B=      1
        0
        0

C=      0     2     3

D=      0
```

22/04/19

EXPERIMENT: 9

AIM: Determine the state space representation of the given transfer function

SOFTWARE REQUIRED: Matlab(2015)

THEORY: ; In control engineering, a **state-space representation** is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations or difference equations. State variables are variables whose values evolve through time in a way that depends on the values they have at any given time and also depends on the externally imposed values of input variables. Output variables' values depend on the values of the state variables.

The "state space" is the Euclidean space in which the variables on the axes are the state variables. The state of the system can be represented as a vector within that space.

PROGRAM CODE:

```
clear all
close all
clc
A=[0 -0.4 -1; 1 0 0;0 1 0]
B=[1;0;0]
C=[0 2 3]
D=[0]
[num,den]=ss2tf(A,B,C,D)
```

OUTPUT:

```
A =

         0   -0.4000   -1.0000
    1.0000         0         0
         0    1.0000         0


B =

     1
     0
     0


C =

     0     2     3
```

D =

0

num =

0 2 3

den =

1.0000 0 0.4000 1.0000

22/04/19

EXPRERIMENT-10

AIM:- Plot bode plot of given transfer function. Also determine the relative stability by measuring gain and phase margin.

SOFTWARE USED:- Matlab(R2016a)

THEORY:- A Bode Plot is a useful tool that shows the gain and phase response of a given LTI system for different frequencies. Bode Plots are generally used with the Fourier Transform of a given system. An example of a Bode magnitude and phase plot setThe Bode plot for a linear, time-invariant system with transfer function {displaystyle H(s)} H(s) ({displaystyle s} s being the complex frequency in the Laplace domain) consists of a magnitude plot and a phase plot.

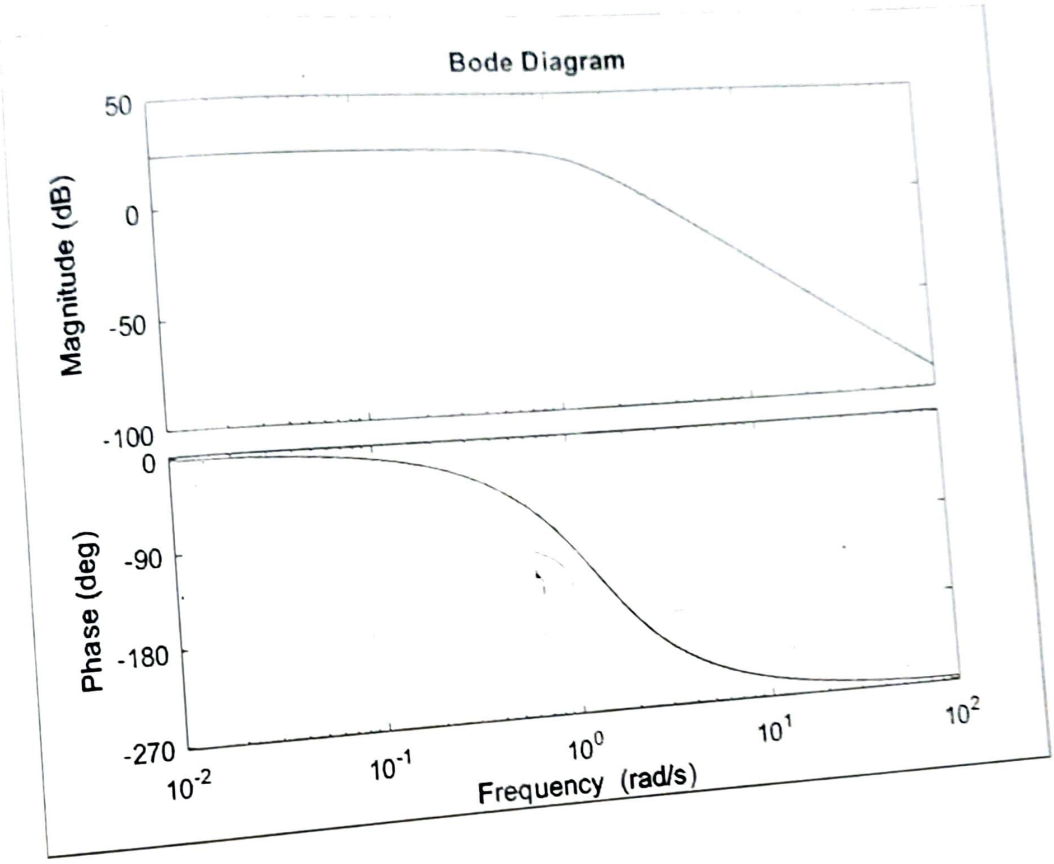
PROGRAM:-

```
clc;

clear all;

close all;

s=tf([32],[1 3 4 2])

subplot(1,1,1)

bode(s)

grid on
```

TRANSFER FUNCTION:-

$$S = \frac{32}{s^3 + 3s^2 + 4s + 2}$$

OUTPUT:-



RESULT:- Bode plot of given transfer function has been plotted.

22/04/19

EXPRERIMENT-12

AIM:- Plot Nyquist plot for given transfer function and to discuss closed loop stability. Also determine the relative stability by measuring gain and phase margin.

SOFTWARE USED:- Matlab(R2016a)

THEORY:- The Nyquist criterion is widely used in electronics and control system engineering, as well as other fields, for designing and analyzing systems with feedback. While Nyquist is one of the most general stability tests, it is still restricted to linear, time-invariant (LTI) systems. Non-linear systems must use more complex stability criteria, such as Lyapunov or the circle criterion. While Nyquist is a graphical technique, it only provides a limited amount of intuition for why a system is stable or unstable, or how to modify an unstable system to be stable. Techniques like Bode plots, while less general, are sometimes a more useful design tool.

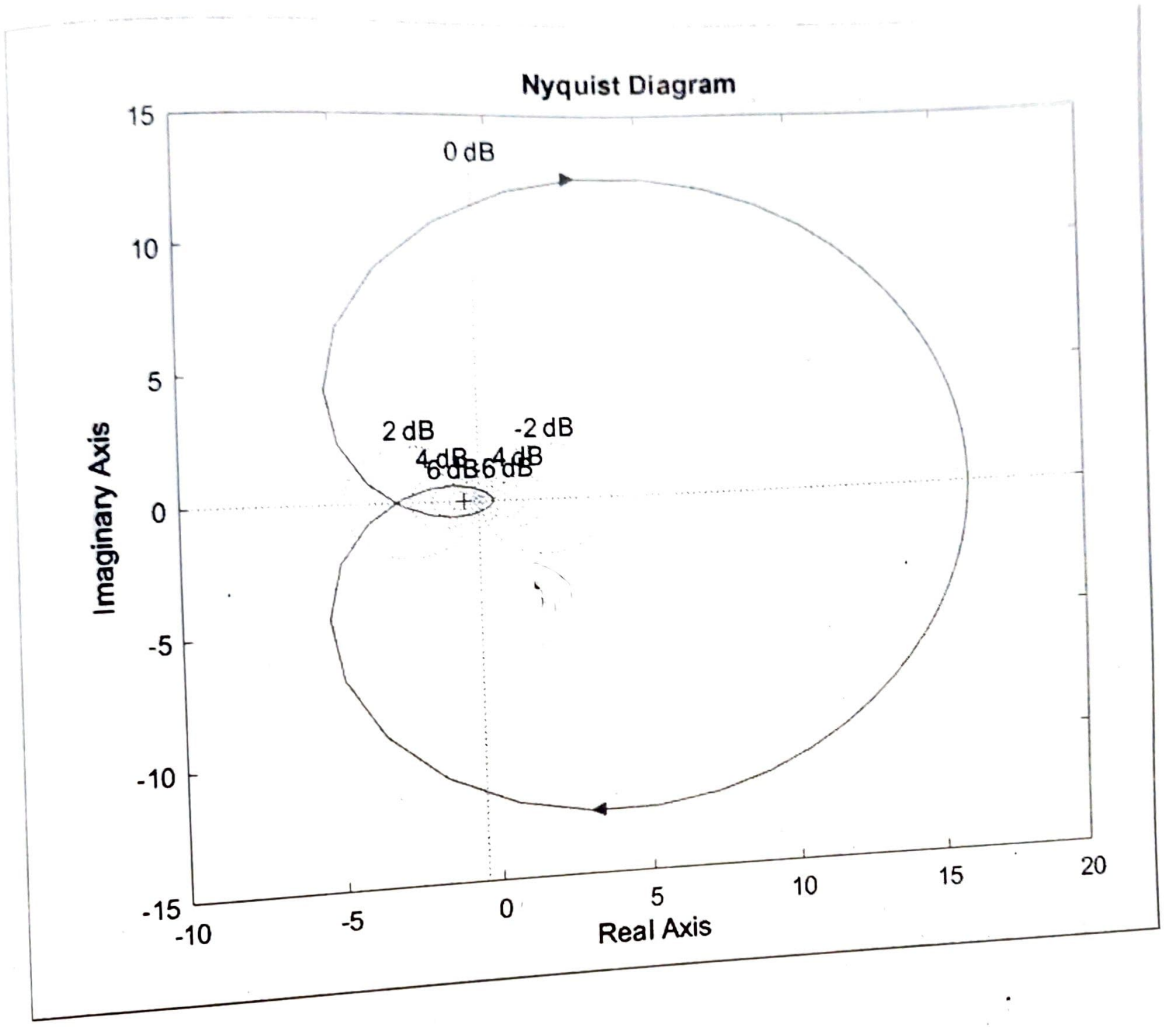
PROGRAM:-

```
clc;
clear all;
close all;
s=tf([32],[1 3 4 2])
subplot(1,1,1)
nyquist(s)
grid on
```

TRANSFER FUNCTION:-

$$S = \frac{32}{s^3 + 3s^2 + 4s + 2}$$

OUTPUT:-



RESULT:- Nyquist plot of the given transfer function has been plotted.

21/04/19