UNIVERSITY OF ENGINEERING AND MANAGEMENT, KOLKATA

INDEX

S.NO	DATE OF EXPERIMENT	EXPERIMENT NAME	DATE OF SUBMISSION	SIGNATURE
01	25/01/23	Introduction to control systems: Transfer Function	09/02/23	
02	02/02/23	Time Response of First Order System	09/02/23	09/02/23
03	09/02/23	Familiarization with Matlab Simulink	11/04/23	
04	16/02/23	Time Response of Second Order System	30/03/23	
05	23/03/23	Effect of P, PI & PID controller.	30/03/23	30/03/23
06	16/03/23	Root locus plot using Matlab	~~30/03~~ 11/04/23	
07	23/03/23	Nyquist plot using Matlab	~~30/03~~ 11/04/23	
08	06/04/23	State Space Model for classic T(S)	06/04/23	6/04/23
09	23/03/23	Bode plot using Matlab	11/04/23	
10	~~23~~ 06/04/23	Study of working Principle of servo motor	11/04/23	

EXPERIMENT 1

INTRODUCTION TO CONTROL SYSTEMS : TRANSFER FUNCTION

AIM: To obtain a transfer function from given poles and zeroes using MATLAB

SOFTWARE: MATLAB

THEORY: A transfer function is also known as the network function is a mathematical representation, in terms of spatial or temporal frequency, of the relation between the input and output of a (linear time invariant) system. The transfer function is the ratio of the output Laplace Transform to the input Laplace Transform assuming zero initial conditions. Many important characteristics of dynamic or control systems can be determined from the transfer function.
The transfer function is commonly used in the analysis of single-input single-output electronic system, for instance. It is mainly used in signal processing, communication theory, and control theory. The term is often used exclusively to refer to linear time-invariant systems (LTI). In its simplest form for continuous time input signal x(t) and output y(t), the transfer function is the linear mapping of the Laplace transform of the input, X(s), to the output Y(s).
Zeros are the value(s) for z where the numerator of the transfer function equals zero. The complex frequencies that make the overall gain of the filter transfer function zero. Poles are the value(s) for z where the denominator of the transfer function equals zero. The complex frequencies that make the overall gain of the filter transfer function infinite.
The general procedure to find the transfer function of a linear differential equation from input to output is to take the Laplace Transforms of both sides assuming zero conditions, and to solve for the ratio of the output Laplace over the input Laplace.

MATLAB PROGRAM TO OBTAIN ZEROS AND POLES OF TRANSFER FUNCTION:

1) $G(s) = \dfrac{4s^2+5s+2}{2s^3+s^2+1}$

```
=> num = [4 5 2]
den = [2 1 0 1]
Sys = tf(num,den)
[z p k] = tf2zp(num,den)
```

2) $G(s) = \dfrac{(s^2+9)(2+5s)}{(s^2+16)(2+3s)}$

~~=> numf~~

```
=> num1=[1 0 9]
num 2=[5 2]
num 3 =[conv[num1,num2];
den 1 = [1 0 16]
den 2 = [3 2]
den 3 =[conv(den1,den2)];
Sys=tf(num3,den3)
[Z,P,K] = tf2zp(num3,den3)
Sys1 = zpk[(Z,P,K)
```

(5) $G(s) = \dfrac{79s^2+916s+1000}{s(s+10)^3}$

```
=> num = [79 916 1000]
   den1 = [1 0]
   r = [-10 -10 -10]
   P2 = poly(r)
   den = conv(den1, P2)
   sys = tf(num, den)
   [z,p,k] = tf2zp(num, den)
   sys1 = zpk(z,p,k)
```

RESULT:

(1) Continuous-Time Transfer Function

Z = -0.6250+0.3307i
-0.6250-0.3307i

P = -1.0000+0.0000i
0.2500+0.6614i
0.2500-0.6614i

K = 2

(2) Continuous-Time Transfer Function

Z = -0.0000 + 3.0000i
-0.0000 - 3.0000i
-0.4000 + 0.0000i

P = 0.0000 + 4.0000i
0.0000 - 4.0000i
-0.6667 + 0.0000i

K = 1.6667

(3) Continuous-Time Transfer Function

Z = -10.3748
-1.2201

P = 0.0000+0.0000i
-10.0000+0.0000i
-10.0000+0.0000i
-10.0000-0.0000i

K = 79

EXPERIMENT 2

TIME RESPONSE OF FIRST ORDER SYSTEM

AIM: To determine the time response of first order system for step, impulse and ramp inputs.

THEORY:
A step signal is a signal whose value changes from one level to another level in zero time.
Mathematically, the step signal is represented as given below:
R(t)=u(t)

U(t)=1 t>0

=0 t<0

In laplace transform form r(s)=1/s

Step response of a given transfer function

T(s)=c(s)/r(s); r(s) is the input and c(s) is the output.

C(s)=t(s)/s

C(t)=laplace^-1(c(s))

MATLAB PROGRAM:

(1) $G(s) = \frac{5}{2S+1}$

```
=> t=0:0.01:10;
num=[5]
den=[2 1]
[y,x,t]=step(num,den)
sys=tf(num,den)
plot (t,y)
```

For ~~step~~ impulse function

```
t=0:0.01:10;
num=[5]
den=[2 1]
y=impulse(num,den,t)
plot (t,y)
```

(2) $G(s) = \dfrac{2}{3s+4}$

<u>for impulse</u>.

```
t = 0: 0.01 : 10;
num = [2]
den = [3 4]
y = impulse (num, den, t)
plot (t, y)
```

<u>for step</u>

```
t = 0: 0.01 : 10
num = [2]
den = [3 4]
[y, x, t] = step (num, den)
sys = tf (num, den)
plot (t, y).
```

(3) $5\ddot{y}(t) + y(t) = 2x(t)$

i.e, $\dfrac{Y(s)}{X(s)} = \dfrac{2}{5s^2+1}$

$G(s) = \dfrac{2}{5s^2+1}$

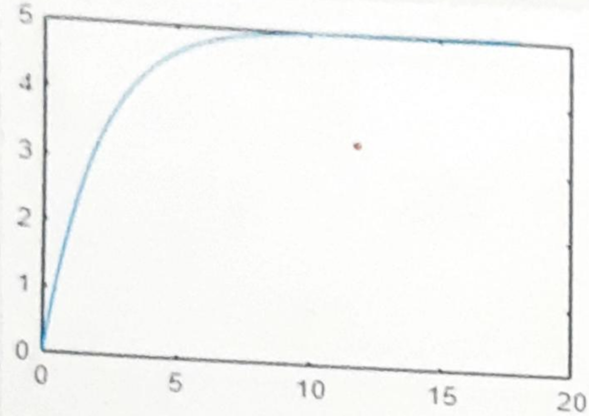
=>
```
t = 0: 0.01 : 10
num = [2]
den = [5 0 1]
y = impulse (num, den, t)
plot (t, y)
```

<u>for step</u>

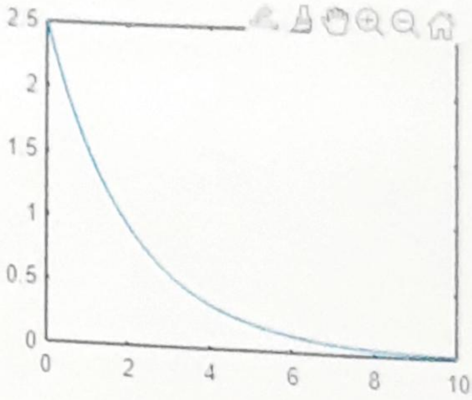
```
t = 0: 0.01 : 10;
num = [2]
den = [5 0 1]
[y, x, t] = step (num, den)
sys = tf (num, den)
plot (t, y)
```

RESULT :

1)



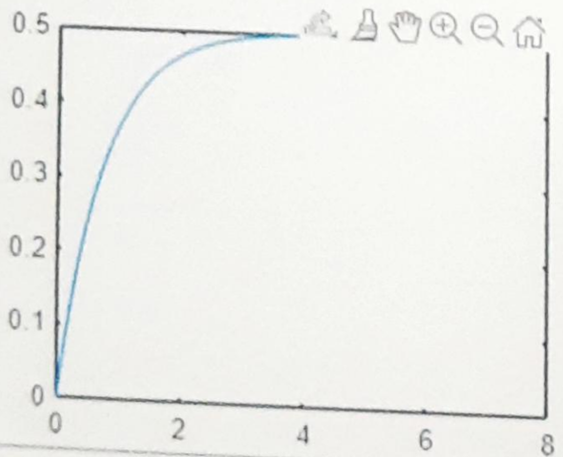
STEP SIGNAL



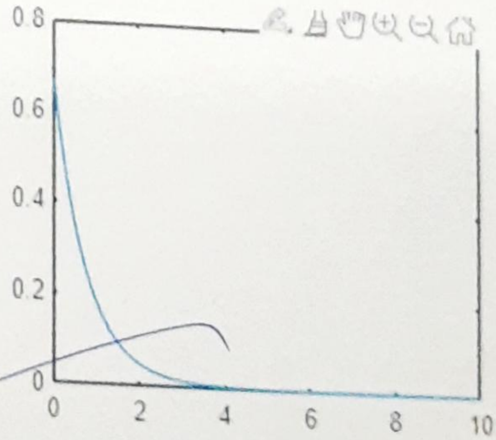
IMPULSE SIGNAL

2)

2.



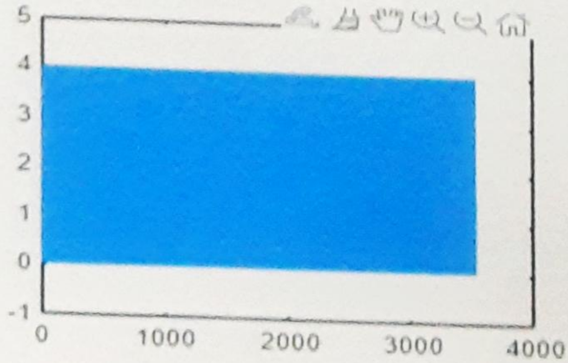
STEP SIGNAL



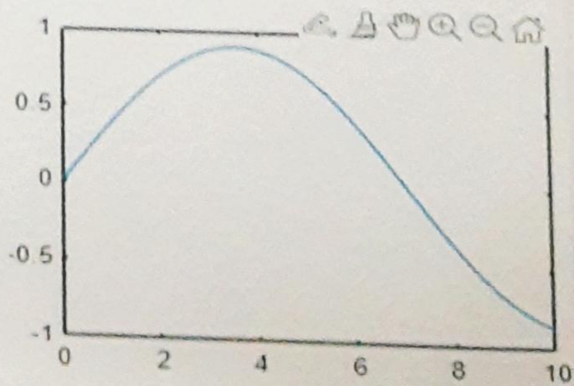
IMPULSE SIGNAL

3)

3.



STEP SIGNAL



IMPULSE SIGNAL

EXPERIMENT

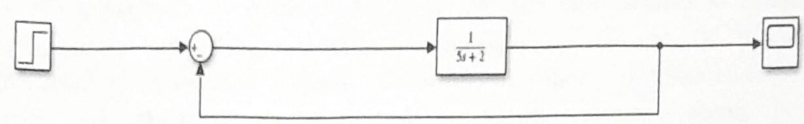
FAMILIARISATION WITH MATLAB SIMULINK

AIM: To become familiar with utilization of simulink.

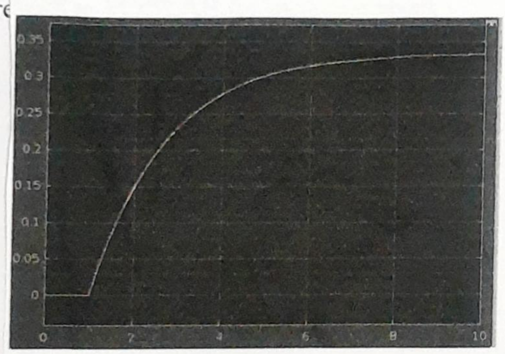
SOFTWARE: MATLAB

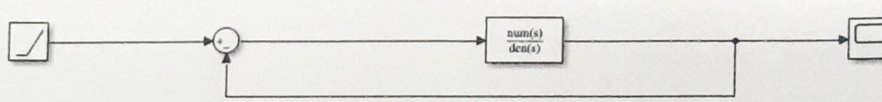
THEORY: Simuli
Based Design. It
continuous test an
customizable block
integrated with MA
export simulation re

CODE:

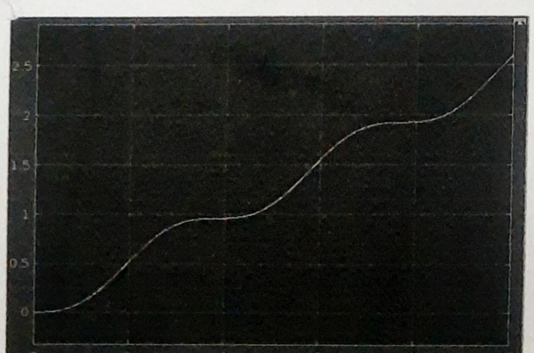


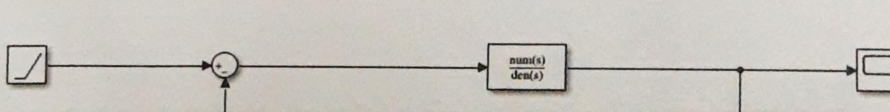
1)



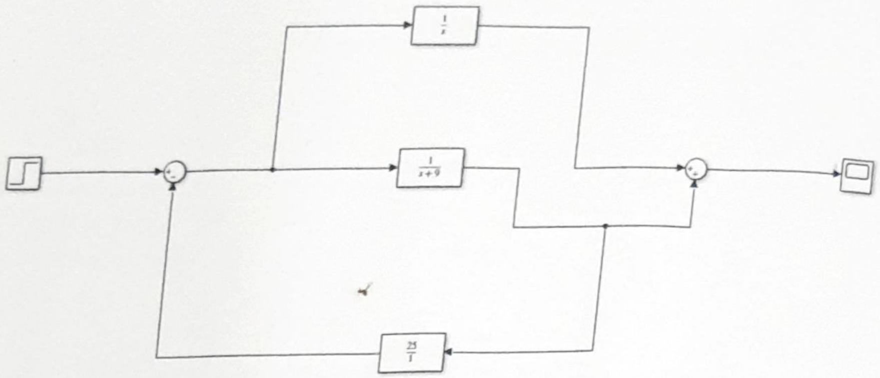


2)

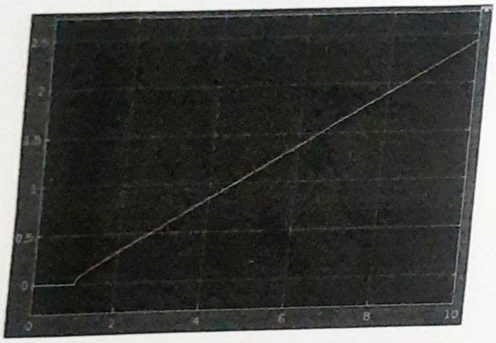




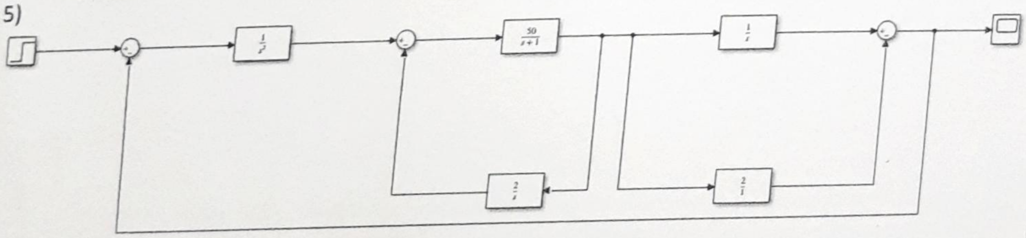
3)

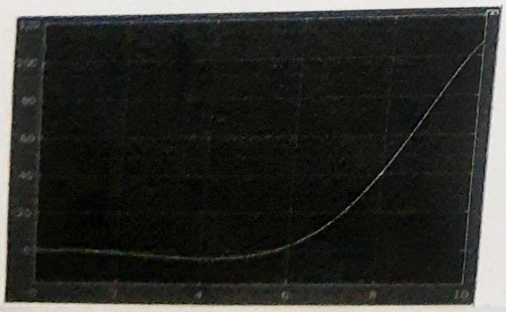


4)

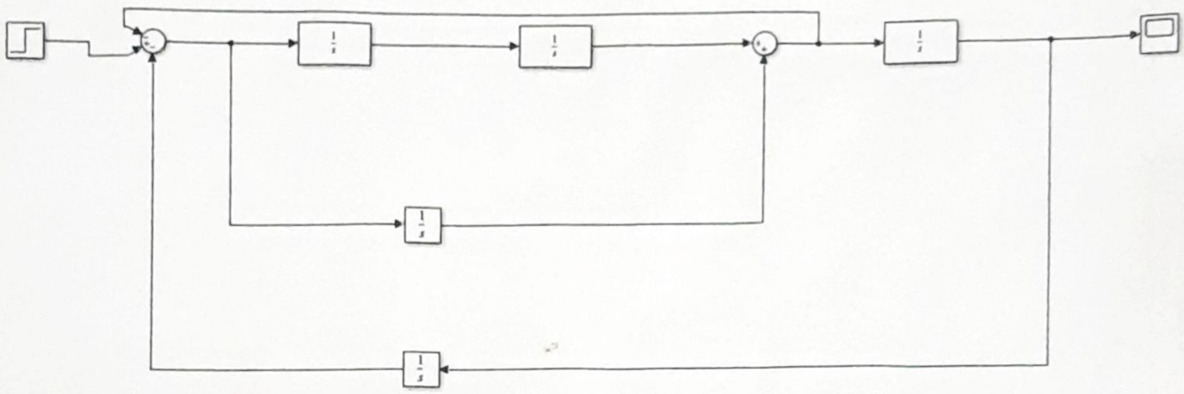


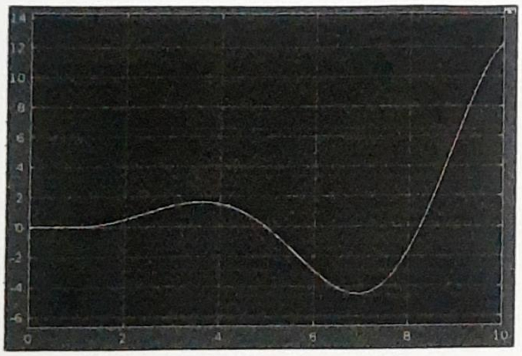
5)



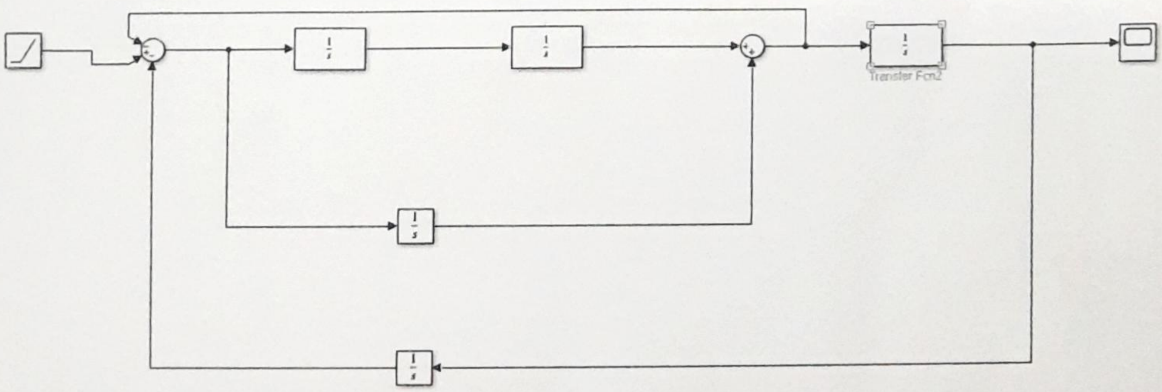


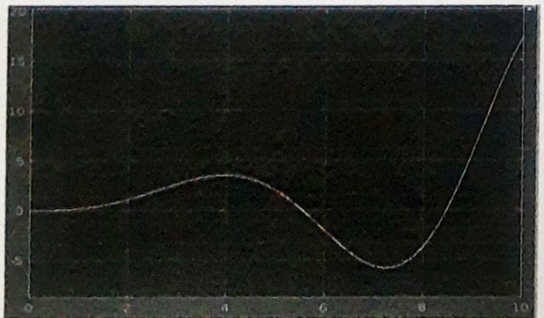
6)



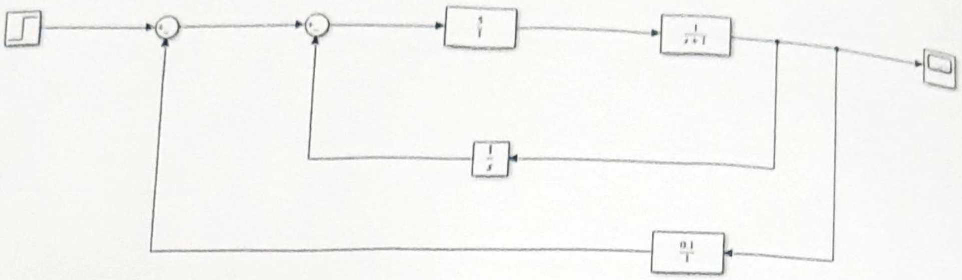


7)

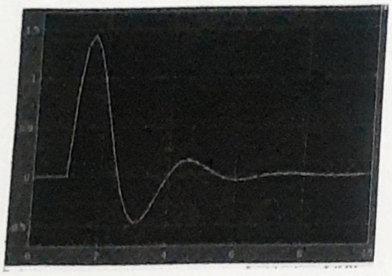




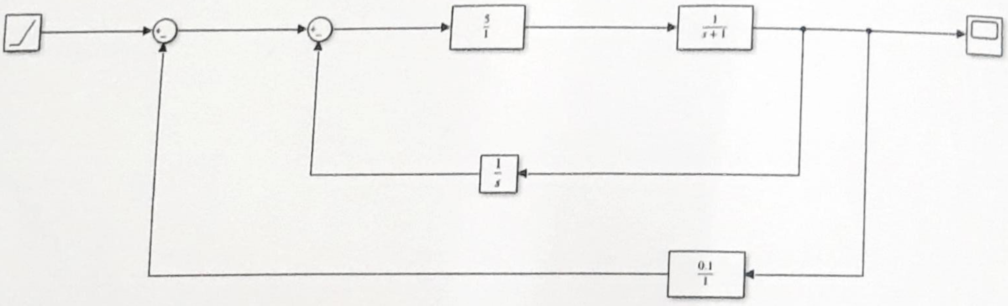
8)



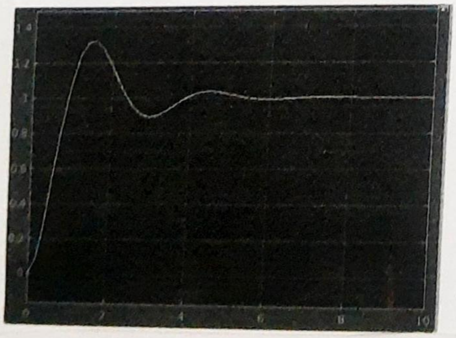
$\frac{5}{1}$
$\frac{1}{s+1}$
$\frac{1}{s}$
$\frac{0.1}{1}$



9)

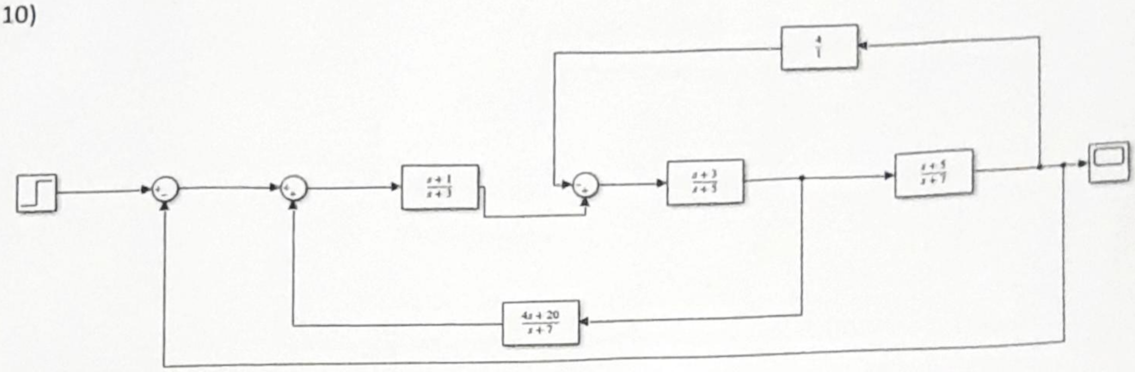


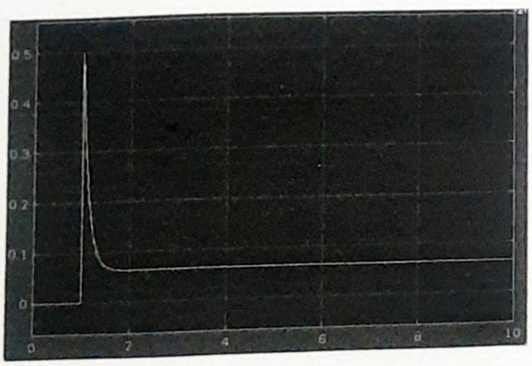
$\frac{5}{1}$
$\frac{1}{s+1}$
$\frac{1}{s}$
$\frac{0.1}{1}$



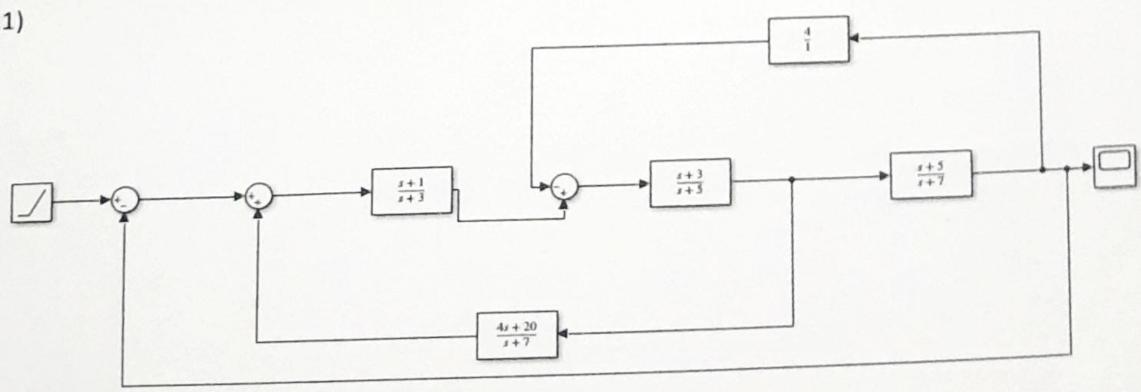
0
2
4
6
8
10

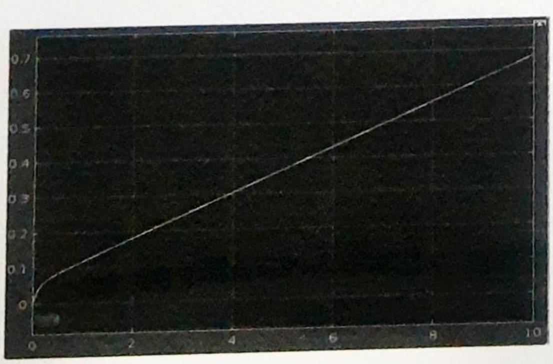
10)


4
1
s+1
s+3
s+3
s+5
s+5
s+7
4s + 20
s+7


0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10

11)


4
1
s+1
s+3
s+3
s+5
s+5
s+7
4s + 20
s+7


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10

EXPERIMENT 4

TIME RESPONSE OF SECOND ORDER SYSTEM

AIM – To determine the time response of a first order and second order system

THEORY-: The time response has utmost importance for the design and analysis of control systems because these are inherently time domain systems where time is independent variable. During the analysis of response, the variation of output with respect to time can be studied and it is known as time response. To obtain satisfactory performance of the system with respect to time the must be within the specified limits. From time response analysis and corresponding results, the stability of system, accuracy of system and complete evaluation can be studied easily.
Due to the application of an excitation to a system, the response of the system is known as time response and it is a function of time. The two parts of response of any system:
(i) Transient response
(ii) Steady-state response.
Transient response: The part of the time response which goes to zero after large interval of time is known as transient response.
Steady state response: The part of response that means even after the transients have died out is said to be steady state response.
The total response of a system is sum of transient response and steady state response:
$C(t)=C_{tr}(t)+C_{ss}(t)$

STEP RESPONSE OF DIFFERENT TYPES OF SYSTEMS

MATLAB PROGRAM TO CALCULATE THE TIME DOMAIN PARAMETERS OF A SECOND ORDER SYSTEM:

$$F(s) = \frac{16}{s^2+1.6s+16}$$

```
1) t=0:0.01:20
num [16]
den [1 1.6 16]
sys = tf(num,den)
[y,x,t] = step(num,den,t)
plot(t,y)
[ymax, tp] = max(y)
Peakovershoot = [ymax-1]*100
r1=1;
while y(r1)>=0.1
    r1=r1+1;
end
r2=1;
while y(r2)<=0.9
    r2=r2+1;
end
risetime = (r2-r1)*0.01
r3=1
while y(r3)<=0.5
    r3=r3+1
end
DelayTime = (r3-1)*0.01
S=200;
while y(s)>=0.98 && y(s)<=1.02
    s=s-1
end
SettlingTime = (s-1)*0.01
```

Output values
1) Peak-Overshoot
=> 0.5266
2) Risetime
=> 0.4200
3) Delay time
=> 0.2900
4) Settling time
=> 4.9000
5) ξ = 0.2

2) $G(S) = \frac{16}{S(S+5)}$ [Open-Loop] $\rightarrow \frac{16}{S^2+5S+16}$ [Closed loop]

```
⇒ t=0:0.01:20
num=[16]
den=[1 5 16]
sys = tf(num,den)
[y,x,t] = step(num,den,t)
plot(t,y)
[ymax,tp]=max(y)
Peakovershoot=[ymax-1]
r1 = 1;
while y(r1)>=0.1
    r1=r1+1
end
r2=1;
while y(r2)<=0.9
    r2=r2+1;
end
risetime=(r2-r1)*0.01
r3=1;
while y(r3)<=0.5
    r3=r3+1;
end
DelayTime=(r3-1)*0.01
S=2001;
while y(S)>=0.98 && y(S)<=1.02
    S=S-1;
end
SettlingTime=(S-1)*0.01
```

Output Values

1) Peak Overshoot = 0.0808
2) Rise Time = 0.6100
3) Delay Time = 0.3500
4) Settling Time = 1.4900
5) ζ = 0.625

3) $G(S) = \frac{10}{S^2+4S+15}$ [Open-Loop]

```
⇒ t=0:0.01:20
num=[10]
den=[1 4 15]
sys=tf(num,den)
[y,x,t]=step(num,den,t)
plot(t,y)
peakovershoot=((ymax-0.67/0.67))*100
Peaktime=(tp-1)*0.01
r1=1;
while y(r1)>=0.07
    r1=r1+1;
end
r2=1;
while y(r2)<=0.96
    r2=r2+1;
end
risetime=(r2-r1)*0.01
r3=1;
while y(r3)<=0.5
    r3=r3+1;
end
DelayTime=(r3-1)*0.01
S=2001;
while y(S)>=(0.98*0.67) && y(S)<=(1.02*0.67)
    S=S-1;
end
SettlingTime=(S-1)*0.01
sys1=tf(num,den)
stepinfo(sys1)
```

Output Values

RiseTime = 0.4316
SettlingTime = 2.0314
SettlingMin = 0.6167
SettlingMax = 0.7669
Overshoot = 15.0389
Undershoot = 0
Peak = 0.7669
PeakTime = 0.9441

ζ = 0.257

30/03/23

Q) Find the response of PID controller whose expression is $\frac{1}{s^2+20s+30}$. Determine the value of Kp, Ki & Kd.
Kp = 200, KI = 300 & Kd = 10.

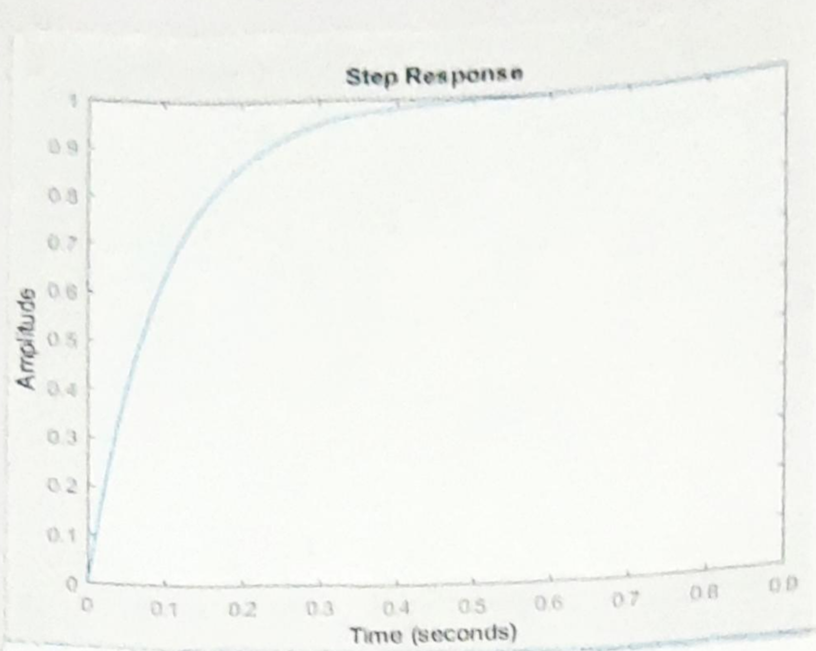
=>
```
Kp = 200;
KI = 300;
Kd = 10;
c = pid(Kp, Ki, Kd).
num = [1]
den = [1 20 30]
plant = tf(num,den)
sys1 = series(c,plant).
sys2 = feedback(sys1,1)
step(sys2).
```

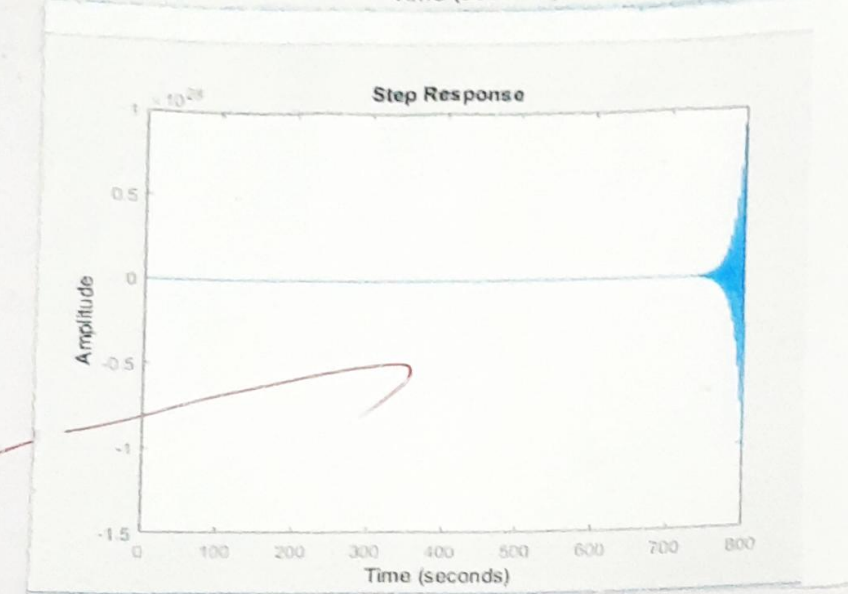
Q) Find the response of PID controller, where system is a closed loop equation for plant is $F(s) = \frac{s}{s(s^2-3)}$ of controller loop is $C(s) = \frac{s+6}{s(s^2+4s+3)}$.

=>
```
num1 = [1 0]
den1 = [1 0]
den2 = [1 0 3]
den3 = [conv(den1, den2)]
plant = tf(num1,den3).
num2 = [1 6]
den4 = [1 0]
den5 = [1 4 3]
den6 = [conv(den4,den5)]
cont = tf(num2, den6)
```

```
sys1 = series(cont, plant)
sys2 = feedback(sys1, 1)
step(sys2).
```

RESULT:





Conclusion:- We have successfully conducted the experiment showing the effect of P, PI & PID controller of second order system. We have plotted graphs of different ~~systems~~ respectively.

30/03/23

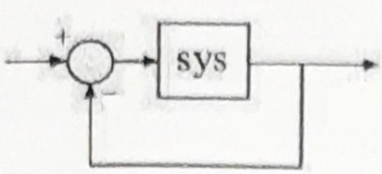
EXPERIMENT 6

ROOT LOCUS PLOT USING MATLAB

AIM – Root locus plot using MATLAB.

SOFTWARE: MATLAB

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.



margin calculates the minimum gain margin, Gm, phase margin, Pm, and associated frequencies Wgm and Wpm of SISO open-loop models. The gain and phase margin of a system sys indicates the relative stability of the closed-loop system formed by applying unit negative feedback to sys.

The gain margin is the amount of gain increase or decrease required to make the loop gain unity at the frequency Wgm where the phase angle is –180° (modulo 360°). In other words, the gain margin is $1/g$ if g is the gain at the –180° phase frequency. Similarly, the phase margin is the difference between the phase of the response and –180° when the loop gain is 1.0. The frequency Wpm at which the magnitude is 1.0 is called the *unity-gain frequency* or *gain crossover frequency*. It is generally found that gain margins of three or more combined with phase margins between 30 and 60 degrees result in reasonable trade-offs between bandwidth and stability.

MATLAB CODE:

1. **Draw the root locus for the function** $\frac{k}{s(s+2)(s+3)}$.

a)
```
K=[1]
z=[0]
P=[0 -2 -3]
Sys=zpk(z,P,K)
rlocus(sys)
K=rlocfind(Sys)
```

b)
```
K=[1]
z=[-1]
P=[-2 -3 -4]
Sys=zpk(z,P,K)
rlocus(sys)
K=rlocfind(sys)
```

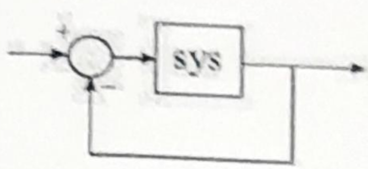
EXPERIMENT 6

ROOT LOCUS PLOT USING MATLAB

AIM – Root locus plot using MATLAB.

SOFTWARE: MATLAB

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.



margin calculates the minimum gain margin, Gm, phase margin, Pm, and associated frequencies Wgm and Wpm of SISO open-loop models. The gain and phase margin of a system sys indicates the relative stability of the closed-loop system formed by applying unit negative feedback to sys.

The gain margin is the amount of gain increase or decrease required to make the loop gain unity at the frequency Wgm where the phase angle is –180° (modulo 360°). In other words, the gain margin is 1/g if g is the gain at the –180° phase frequency. Similarly, the phase margin is the difference between the phase of the response and –180° when the loop gain is 1.0. The frequency Wpm at which the magnitude is 1.0 is called the *unity-gain frequency* or *gain crossover frequency*. It is generally found that gain margins of three or more combined with phase margins between 30 and 60 degrees result in reasonable trade-offs between bandwidth and stability.

MATLAB CODE:

1. **Draw the root locus for the function** $\frac{k}{s(s+2)(s+3)}$.

a)
```
K=[1]
z=[0]
P=[0 -2 -3]
Sys= zpk(z,P,K)
rlocus(sys)
K =rlocfind(Sys)
```

b)
```
K=[1]
z=[-1]
P=[-2 -3 -4]
Sys = zpk (z,P,K)
rlocus (sys)
K = rlocfind (sys)
```

c)
```
K=[1]
num=[1 1];
den=[1 4 13];
sys=tf(num,den,K)
rlocus(sys)
K=rlocfind(sys)
```

d)
```
K=[1]
num1=[1 1];
num2=[1 5];
num3=conv(num1,num2);
den1=[1 3];
den2=[1 4 13];
den3=conv(den1,den2);
Sys=tf(num3,den3,K)
rlocus(sys)
K=rlocfind(sys)
```

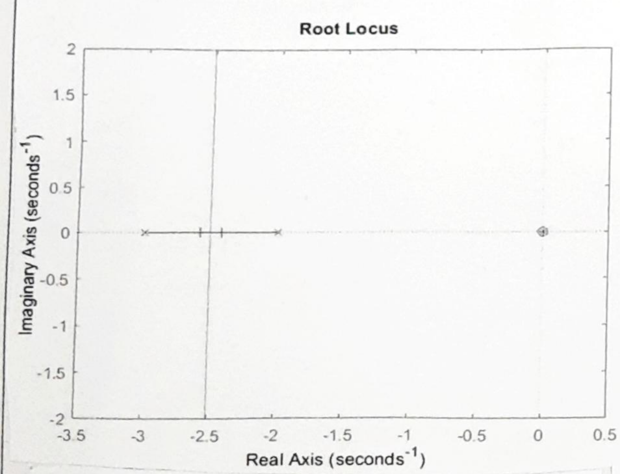
Output

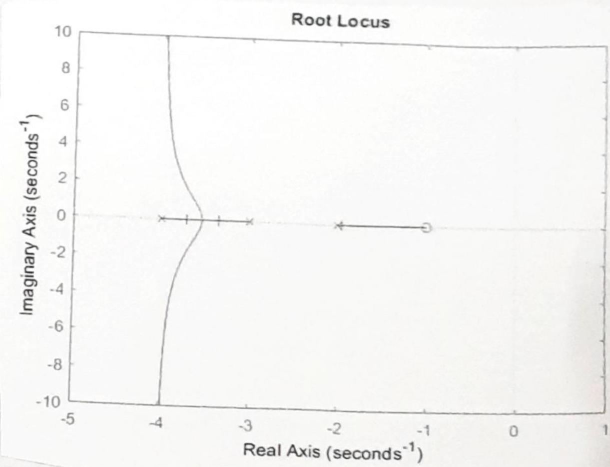
a) K = 0.2209

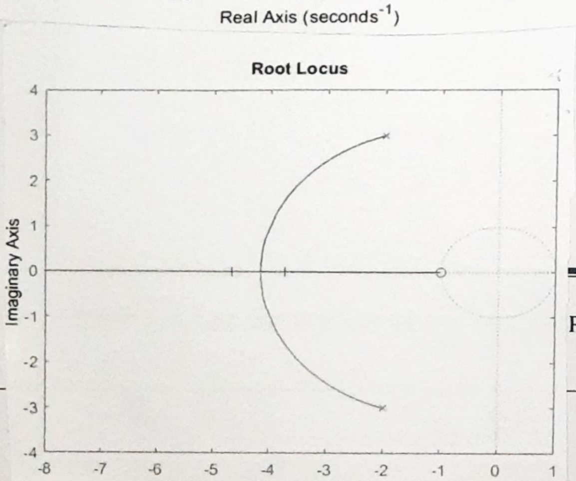
b) K = 0.1274

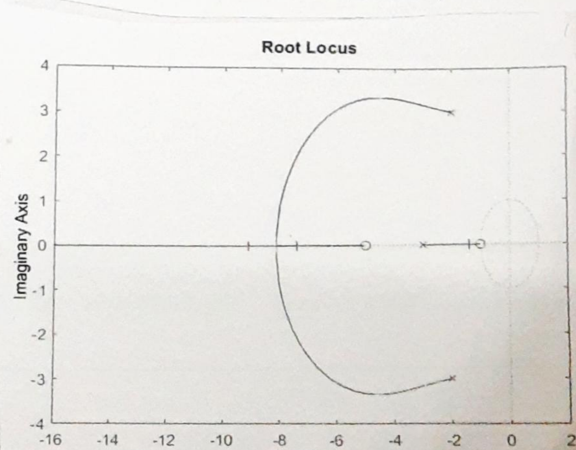
c) K = 4.4322

d) K = 10.7634









1) $G(s)H(s) = \frac{50(s+1)}{s(s+3)(s+5)}$

=> K = [50];
z = [-1];
P = [0 -3 -5]
[num, den] = zp2tf(z,P,K)
sys = tf(num, den)
bode(sys)
[gm, Pm] = margin(sys)

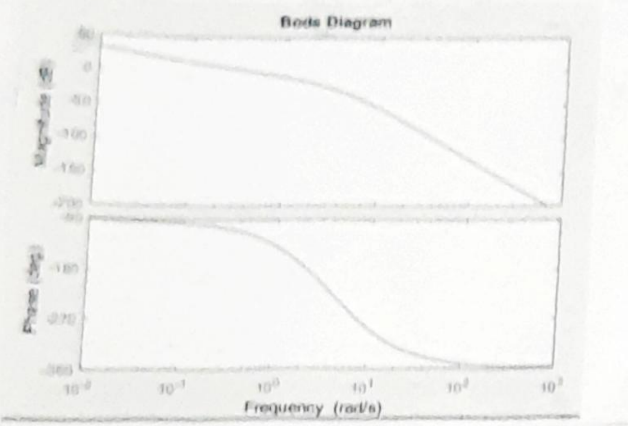
2) $G(s)H(s) = \frac{25(s+1)(s+7)}{s(s+2)(s+4)(s+8)}$

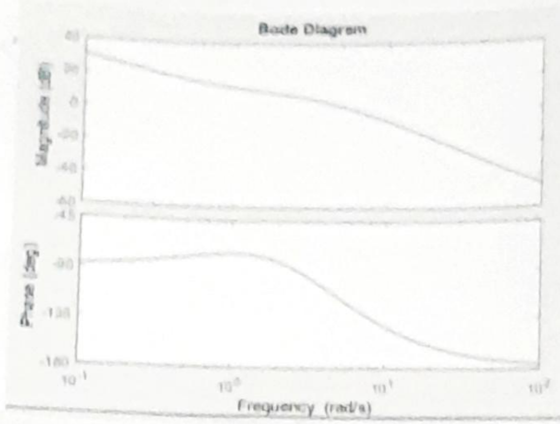
=> K = [25];
z = [-1 -7]
P = [0 -2 -4 -8]
[num, den] = zp2tf(z,P,K)
sys = tf(num, den)
bode(sys)
[gm, pm] = margin(sys)

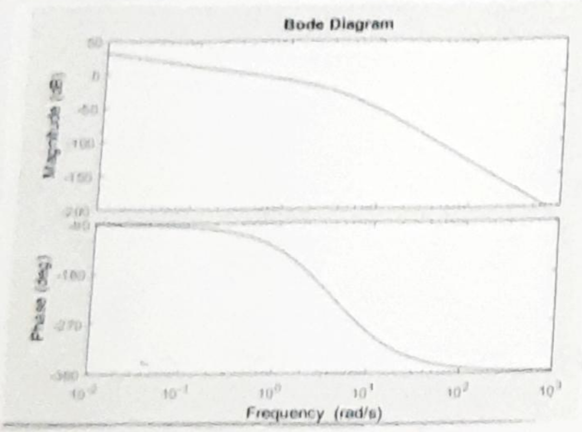
3) $G(s)H(s) = \frac{K}{s(s+3)(s+12)}$

=> P = [0 -3 -12]
z = [0]
[num, den] = zp2tf(z,P,K)
sys = tf(num, den)
bode(sys)
[gm, Pm] = margin(sys)

OUTPUT:


Bode Diagram
Magnitude (dB)
Phase (deg)
Frequency (rad/s)


Bode Diagram
Magnitude (dB)
Phase (deg)
Frequency (rad/s)


Bode Diagram
Magnitude (dB)
Phase (deg)
Frequency (rad/s)

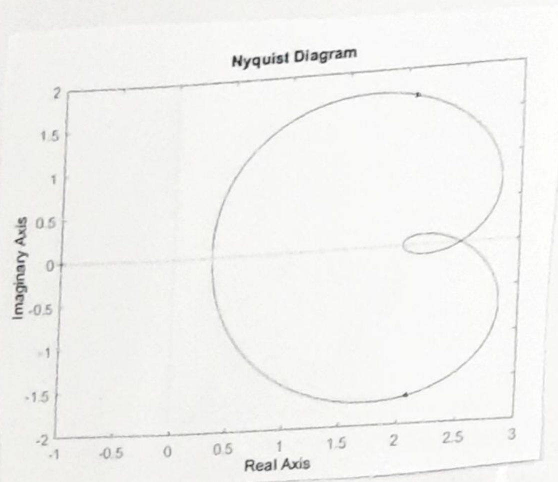
loop transfer function in the right half of the 's' plane. The shift in origin to (1+j0) gives the characteristic equation plane.

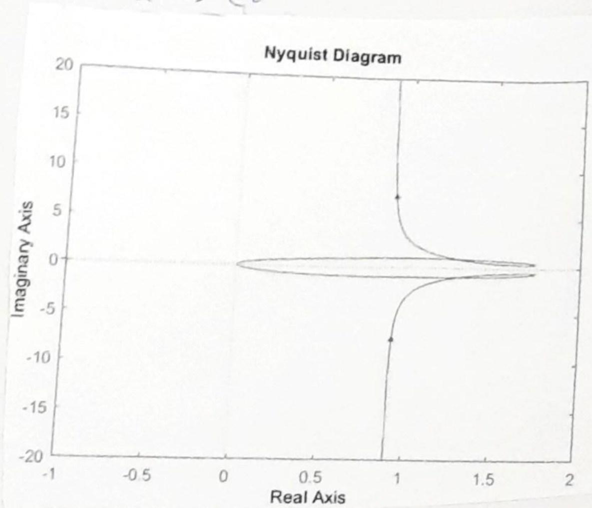
MATLAB CODE:

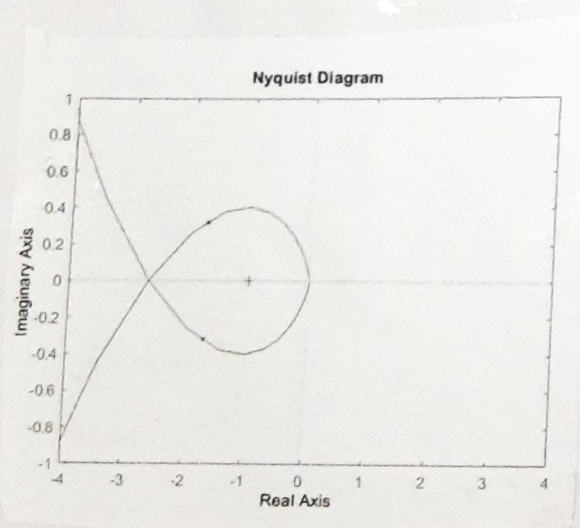
```
1) num = [2 5 1]
   den = [1 2 3]
   sys = tf(num,den)
   nyquist (sys)
   [re, im, w] = nyquist (sys)
```

```
2) num = [20 20 10]
   den = poly ([0 -1 -10])
   sys = tf(num,den)
   nyquist (sys)
```

```
3) num = [2]
   den = poly ([0 -1 -0.5])
   sys = tf (num,den)
   nyquist (sys)
   axis ([-4 4 -1 1]).
```



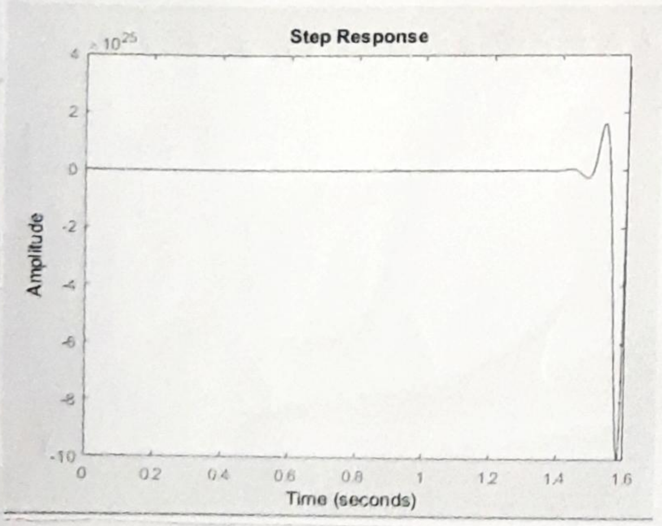


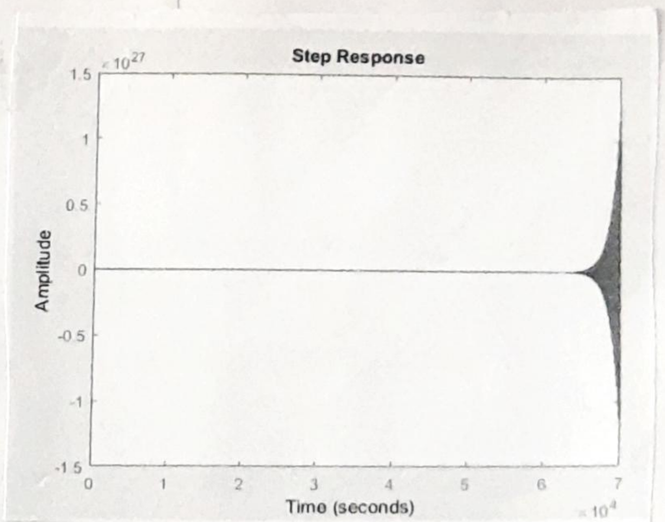


1)
```
num1 = [0.2]
den1 = [2.58*10^(-6) 0 0]
sys1 = tf(num1,den1)
den2 = [1]
num2 = [0 0.199 0]
sys2 = tf(num2,den2)
sys = feedback(sys1,sys2,-1)
step(sys)
```

2)
```
num1 = [1]
den1 = [50*10^(-3) 0.75]
sys1 = tf(num1,den1)
num2 = [1]
den2 = [50 0 0]
sys2 = tf(num2,den2)
sys3 = series(sys1,sys2)
num3 = [1 0]
den3 = [1]
sys4 = tf(num3,den3)
sys5 = feedback(sys3,sys4,-1)
step(sys5)
```

RESULT:





EXPERIMENT 10

STATE SPACE MODEL FOR CLASSICAL TRANSFER FUNCTION USING MATLAB

AIM: To Transform a given Transfer Function to State Space Model and from State Space Model to Transfer Function using MATLAB.

SOFTWARE: MATLAB

THEORY: To abstract from the number of inputs, outputs and states, these variables are expressed as vectors. Additionally, if the dynamical system is linear, time-invariant, and finite-dimensional, then the differential and algebraic equations may be written in matrix form. The state-space method is characterized by significant algebraization of general system theory, which makes it possible to use vector-matrix structures. The capacity of these structures can be efficiently applied to research systems with modulation or without it.

MATLAB CODE:

(1) $T(s) = \dfrac{s^2+s+2}{s^3+9s^2+26s+24}$

```
num = [1 1 2]
den = [1 9 26 29]
sys = tf(num, den)
[A,B,C,D] = tf2ss(num,den)
  AP1 = fliplr(A)
  Ap = flipud(AP1)
  Bp = flipud(B)
  Cp = fliplr(C)
```

Output:-

$A = \begin{bmatrix} -9 & -26 & -29 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ $B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$C = [1 \ 1 \ 2]$

$D = 0$

$AP_1 = \begin{bmatrix} -29 & -26 & -9 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

$Ap = \begin{bmatrix} -9 & -26 & -29 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

$Bp = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

$Cp = [2 \ 1 \ 1]$

② num = [8 1]
den = [9 1 1 2]
sys = tf(num, den)
[A, B, C, D] = tf2ss(num, den)

O/P

$$A = \begin{bmatrix} -0.1111 & -0.1111 & -0.2222 \\ 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad C = [0 \ \ 0.8889 \ \ 0.1111]$$

D = 0

③ num = [1 1 1 1]
den = [9 1 1 2]
sys = tf(num, den)
[A, B, C, D] = tf2ss(num, den)

O/P

$$A = \begin{bmatrix} -0.1111 & -0.1111 & -0.2222 \\ 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad C = [0.0988 \ \ 0.0988 \ \ 0.0864]$$

D = 0.1111

④ num = [1 0]
den = [9 1 1 2]
sys = tf(num, den)
[A, B, C, D] = tf2ss(num, den)

O/P

$$A = \begin{bmatrix} -0.1111 & -0.1111 & -0.2222 \\ 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad C = [0 \ \ 0.1111 \ \ 0]$$

D = 0

⑤ ~~m = obsv (A, C)~~
A = [1 -1; 1 -1]
B = [0 ; 1]
C = [1 0]
m = obsv (A, C)
r = rank(m)
n = ctrb(A, B)
r1 = rank(n)
if (r == 2 && r1 == 2)
display("Observable & Controllable");
else
display("Not Observable & Controllable");
end.

O/P

$$A = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C = [1 \ 0]$$

$$m = \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix} \quad r = 2$$

$$n = \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix} \quad r_1 = 2$$

"System is observable & Controllable"

⑥ A = [-1 0 ; 1 -2]
B = [0 ; 1]
[m d] = eig(A)
if (r == 2)
display ("System is Controllable")
else
display ("System is not Controlable")
end

O/P

$$A = \begin{bmatrix} -1 & 0 \\ 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$md = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$$

"System is controlable"

6/04/23

(2) num = [8 1]
den = [9 1 1 2]
sys = tf(num,den)
[A,B,C,D] = tf2ss(num,den)

O/P

$$A = \begin{bmatrix} -0.1111 & -0.1111 & -0.2222 \\ 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad C = [0 \quad 0.8889 \quad 0.1111]$$

D = 0

(3) num = [1 1 1 1]
den = [9 1 1 2]
sys = tf(num,den)
[A,B,C,D] = tf2ss(num,den)

O/P

$$A = \begin{bmatrix} -0.1111 & -0.1111 & -0.2222 \\ 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad C = [0.0988 \quad 0.0988 \quad 0.0864]$$

D = 0.1111

(4) num = [1 0]
den = [9 1 1 2]
sys = tf(num, den)
[A,B,C,D] = tf2ss(num,den)

O/P

$$A = \begin{bmatrix} -0.1111 & -0.1111 & -0.2222 \\ 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad C = [0 \quad 0.1111 \quad 0]$$

D = 0

(5) ~~m = obsv (A,C)~~
A = [1 -1; 1 -1]
B = [0 ; 1]
C = [1 0]
m = obsv (A,C)
r = rank(m)
n = ctrb(A,B)
r1 = rank(n)
if (r == 2 && r1 == 2)
display ("Observable & Controllable");
else
display ("Not Observable & Controllable");
end .

O/P

$$A = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C = [1 \quad 0]$$

$$m = \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix} \quad r = 2$$

$$n = \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix} \quad r_1 = 2$$

"System is observable & Controllable"

(6) A = [-1 0 ; 1 -2]
B = [0 ; 1]
[m,d] = eig(A)
if (r == 2)
display ("System is Controllable")
else
display ("System is not Controlable")
end

O/P

$$A = \begin{bmatrix} -1 & 0 \\ 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$m,d = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$$

"System is controlable"

6/04/23