DEPARTMENT OF ELECTRONICS & COMMUNICATION

Control System

(BEC403)

IV SEMESTER

Student name: .....

USN: .....

Section: .....

At the end of the course students should be able to:

CO1	Construct the mathematical model of mechanical and electrical systems and solve for the transfer functionAnalysis
CO2	Develop the transfer function for a given control system by applying the knowledge of block diagram reduction techniques and signal flow graphs
CO3	Determine the time domain specifications of first order and second order systems and discuss the operations of different controllers
CO4	Analyze the stability of a given control system in time domain and frequency domain using relevant techniques
CO5	Develop the state model of the given electrical/mechanical system using state variable analysis
CO6	Analyze the stability of a system from the transfer function

CO-PO Mapping:

CO/PO'S	PO1	PO2	PO3	PO4	PO5	PO6	P07	PO8	PO9	P010	PO11	PO12
CO1	3	2			1	-	-	-	1	2	-	2
CO2	2	3	3	1	1	-	-	-	1	2	-	2
CO3	3	3	2	1	1	-	-	-	1	1	-	2
CO4	3	3	2	1	1	1	1	-	1	1	-	2
CO5	3	3	2		1	1	1	-	1	1	-	2
CO6	3	3	2		1	1	1		1	1		1
AVG	2.83	2.83	2.2	1	1	1	1	-	1	1.33	-	1.83

CO-PSO Mapping:

CO/PSO'S	PSO1	PSO2
CO1	3	2
CO2	2	2
CO3	3	2
CO4	3	1
CO5	3	0
CO6	3	0
AVG	2.833	1.16

PRACTICAL COMPONENT OF IPCC Using suitable simulation software (P-Spice/ MATLAB / Python / Scilab / OCTAVE / LabVIEW) demonstrate the operation of the following circuits:	
Sl.No	**Experiments**
1	Implement Block diagram reduction technique to obtain transfer function a control system.
2	Implement Signal Flow graph to obtain transfer function a control system.
3	Simulation of poles and zeros of a transfer function.
4	Implement time response specification of a second order Under damped System, for different damping factors.
5	Implement frequency response of a second order System.
6	Implement frequency response of a lead lag compensator.
7	Analyze the stability of the given system using Routh stability criterion.
8	Analyze the stability of the given system using Root locus.
9	Analyze the stability of the given system using Bode plots.
10	Analyze the stability of the given system using Nyquist plot.
11	Obtain the time response from state model of a system.
12	Implement PI and PD Controllers.

CIE for the practical component of IPCC

- On completion of every experiment/program in the laboratory, the students shall be evaluated and marks shall be awarded on the same day. The **15 marks** are for conducting the experiment and preparation of the laboratory record, the other **05 marks shall be for the test** conducted at the end of the semester.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to 15 marks.
- The laboratory test **(duration 03 hours)** at the end of the 15th week of the semester /after completion of all the experiments (whichever is early) shall be conducted for 50 marks and scaled down to 05 marks**.**
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for **20 marks**.

SEE for IPCC

Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the course (duration 03 hours)

- The question paper will have ten questions. Each question is set for 20 marks.
- There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that module.
- The students have to answer 5 full questions, selecting one full question from each module.

The theory portion of the IPCC shall be for both CIE and SEE, whereas the practical portion will have a CIE component only. Questions mentioned in the SEE paper shall include questions from the practical component.

- The minimum marks to be secured in CIE to appear for SEE shall be the 12 (40% of maximum marks-30) in the theory component and 08 (40% of maximum marks -20) in the practical component. The laboratory component of the IPCC shall be for CIE only. However, in SEE, the questions from the laboratory component shall be included. The maximum of 04/05 questions to be set from the practical component of IPCC, the total marks of all questions should not be more than the 20 marks.

SEE will be conducted for 100 marks and students shall secure 35% of the maximum marks to qualify inthe SEE. Marks secured out of 100 shall be reduced proportionally to 50.

Suggested Learning Resources:

Text Books

1. Control Systems Engineering, I J Nagrath, M. Gopal, New age international Publishers, Fifth edition.

Web links and Video Lectures (e-Resources):

- https://nptel.ac.in/courses/108106098

Activity Based Learning (Suggested Activities in Class)/ Practical Based learning

Programming Assignments / Mini Projects can be given to improve programming skills

What is Octave?

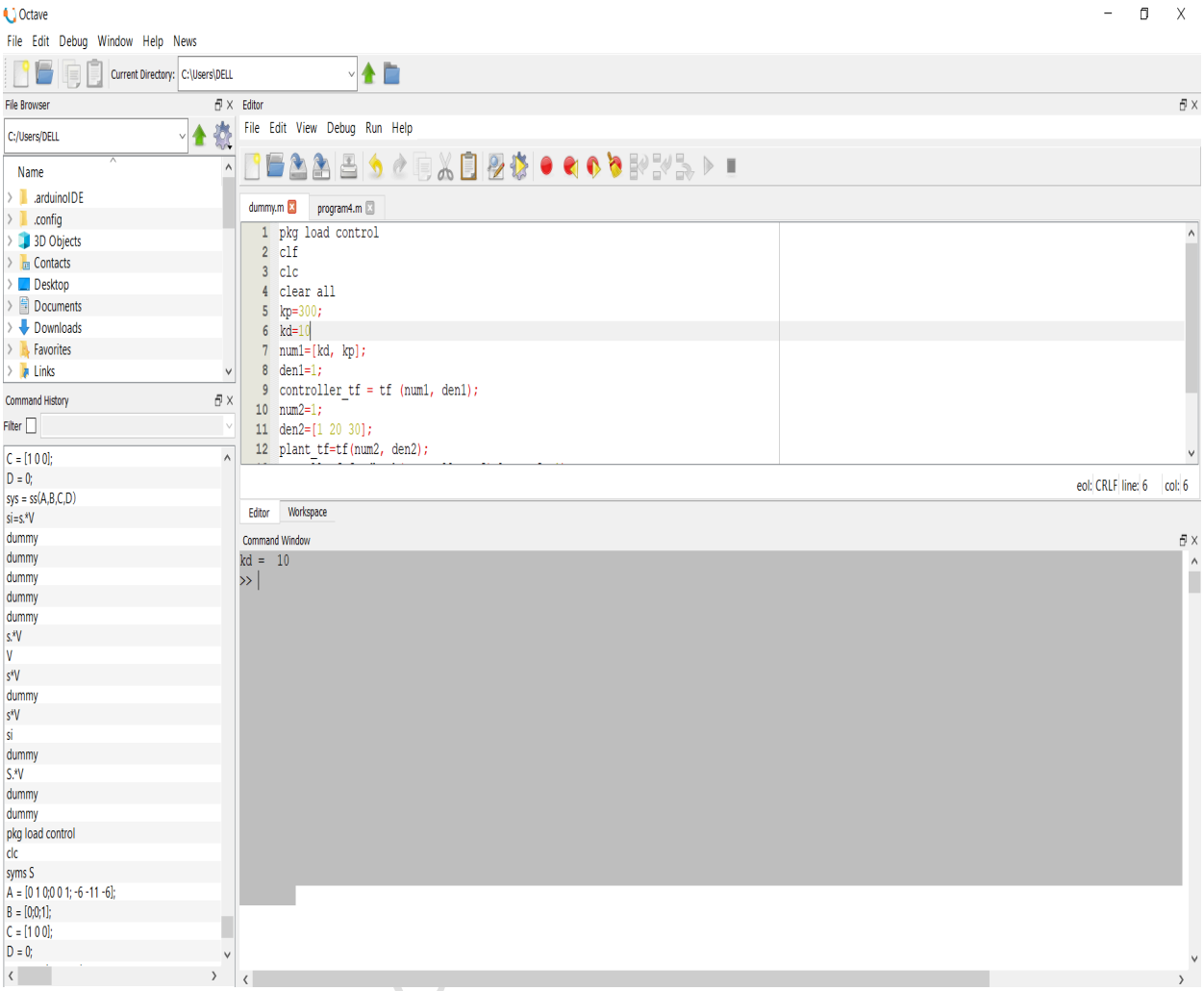
Octave is an open-source interactive software system for numerical computations and graphics. It is particularly designed for matrix computations: solving simultaneous equations, computing eigenvectors and eigenvalues and so on. In many real-world engineering problems the data can be expressed as matrices and vectors, and boil down to these forms of solution. In addition, Octave can display data in a variety of different ways, and it also has its own programming language which allows the system to be extended. It can be thought of as a very powerful, programmable, graphical calculator. Octave makes it easy to solve a wide range of numerical problems, allowing you to spend more time experimenting and thinking about the wider problem.

Octave was originally developed as a companion software to a undergraduate course book on chemical reactor design. It is currently being developed under the leadership of Dr. J.W. Eaton and released under the GNU General Public License. Octave's usefulness is enhanced in that it is mostly syntax compatible with MATLAB which is commonly used in industry and academia.

Who uses Octave?

Octave and MATLAB are widely used by engineers and scientists, in both industry and academia for performing numerical computations, and for developing and testing mathematical algorithms. For example, NASA use it to develop spacecraft docking systems; Jaguar Racing use it to display and analyse data transmitted from their Formula 1 cars; Sheffield University use it to develop software to recognise cancerous cells. It makes it very easy to write mathematical programs quickly, and display data in a wide range of different ways.

Starting Octave



Octave as a calculator

The simplest way to use Octave is just to type mathematical commands at the prompt, like a normal calculator. All of the usual arithmetic expressions are recognised. For example, type

octave:##> 2+2

at the prompt and press return, and you should see

ans = 4

The basic arithmetic operators are + - * /, and ^ is used to mean `to the power of' (e.g. 2^3=8). Brackets ( ) can also be used. The order *precedence* is the same usual i.e. brackets are evaluated first, then powers, then multiplication and division, and finally addition and subtraction. Try a few examples.

Built-in functions

As well as the basic operators, Octave provides all of the usual mathematical functions, and a selection of these can be seen in the Table. These functions are invoked as in C++ with the name of the function and then the function *argument* (or arguments) in ordinary brackets (), for example (A function's arguments are the values which are passed to the function which it uses to calculate its response. In this example the argument is the value `1', so the exponent function calculates the exponential of 1 and returns the value (i.e. e^1) = 2.7183).)

octave:##> exp(1)

ans = 2.7183

Here is a longer expression: to calculate $1.2*\sin(40^o + \ln(2.4^2))$, type

octave:##> 1.2 * sin(40*pi/180 + log(2.4^2))

ans = 0.76618

There are several things to note here:

An explicit multiplication sign is always needed in equations, for example between the 1.2 and sin.

The trigonometric functions (for example sin) work in *radians*. The factor pi/180 can be used to convert degrees to radians. pi is an example of a named *variable*, discussed in the next section.

The function for a natural logarithm is called `log', not `ln'.

Table 1: Basic maths functions

cos	Cosine of an angle (in radians)
sin	Sine of an angle (in radians)
tan	Tangent of an angle (in radians)
exp	Exponential function (e^x)
log	Natural logarithm (NB this is $\log_e$, not $\log_{10}$)
log10	Logarithm to base 10
sinh	Hyperbolic sine
cosh	Hyperbolic cosine
tanh	Hyperbolic tangent
acos	Inverse cosine
acosh	Inverse hyperbolic cosine
asin	Inverse sine
asinh	Inverse hyperbolic sine
atan	Inverse tangent
atan2	Two-argument form of inverse tangent
atanh	Inverse hyperbolic tangent
abs	Absolute value
sign	Sign of the number (-1 or +1)
round	Round to the nearest integer
floor	Round down (towards minus infinity)
ceil	Round up (towards plus infinity)
fix	Round towards zero
rem	Remainder after integer division

Using these functions, and the usual mathematical constructions, Octave can do all of the things that your normal calculator can do.

Named variables

In any significant calculation you are going to want to store your answers, or reuse values, just like using the memory on a calculator. Octave allows you to define and use named variables. For example, consider the degrees example in the previous section. We can define a variable deg to hold the conversion factor, writing

```
octave:##> deg = pi/180

deg =0.017453
```

Note that the *type* of the variable does not need to be defined, unlike most high level languages e.g. in C++. All variables in Octave are floating point numbers. (Or strings, but those are obvious from the context. However, even strings are stored as a vector of character ID numbers.) Using this variable, we can rewrite the earlier expression as

```
octave:##> 1.2 * sin(40*deg + log(2.4^2))
```

ans =0.76618

You will have already have seen another example of a variable in Octave. Every time you type in an expression which is *not* assigned to a variable, such as in the most recent example, Octave assigns the answer to a variable called ans. This can then be used in exactly the same way:

octave:##> new = 3*ans

new =2.2985

Getting help

If you are not sure what a particular Octave command does, or want to find a particular function, Octave contains an integrated help system. The basic form of using help is to type

help *commandname*

For example:

octave:1> help sqrt

sqrt is a built-in function

- Mapping Function: sqrt (X)

Compute the square root of X. If X is negative, a complex result is returned. To compute the matrix square root, see *Note Linear Algebra::.Additional help for built-in functions, operators, and variables is available in the on-line version of the manual. Use the command `doc ' to search the manual index.

Help and information about Octave is also available on the WWW at http://www.octave.org and via the help@octave.org mailing list.

Arrays and vectors

There are lots of ways of defining vectors and matrices. Usually the easiest thing to do is to type the vector inside *square* brackets [], for example

```
octave:##> a=[1 4 5]

  a =

    1     4     5

octave:##> b=[2,1,0]

  b =

     2     1     0
```

```
octave:##> c=[4;7;10]

c =

   4

   7

  10
```

A list of numbers separated by spaces or commas, inside square brackets, defines a row vector. Numbers separated by semicolons, or carriage returns, define a column vector.

You can also construct a vector from an existing vector by including it in the definition, for example

```
octave:##> a=[1 4 5]

a =

    1    4    5

octave:##> d=[a 6]

d =

    1    4    5    6
```

Plotting graphs

Octave has powerful facilities for plotting graphs via a second open-source program GNUPLOT, however some of the range of plotting options are restricted compared with MATLAB. The basic command is plot(x,y), where x and y are the co-ordinates. If given just one pair of numbers it plots a point, but usually you pass *vectors*, and it plots all the points given by the two vectors, joining them up with straight lines.(The two vectors must, naturally, both be the same length.) The sine curve defined in the previous section can be plotted by typing

octave:##> plot(angles,y)

A new window should open up, displaying the graph, shown below. Note that it automatically selects a sensible scale, and plots the axes.

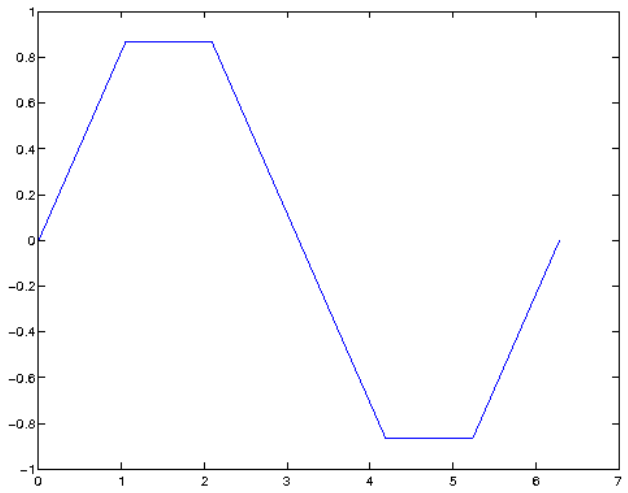


Figure 1: Graph of y=sin(x), sampled every 60°

At the moment it does not look particularly like a sine wave, because we have only taken values one every 60 degrees. To plot a more accurate graph, we need to calculate y at a higher resolution:

```
octave:##> angles=linspace(0,2*pi,100);
octave:##> y=sin(angles);
octave:##> plot(angles, y);
```

The linspace command creates a vector with 100 values evenly spaced between 0 and 2π (the value 100 is picked by trial and error). Try using these commands to re-plot the graph at this higher resolution. Remember that you can use the arrow keys to go back and reuse your previous commands.

Improving the presentation

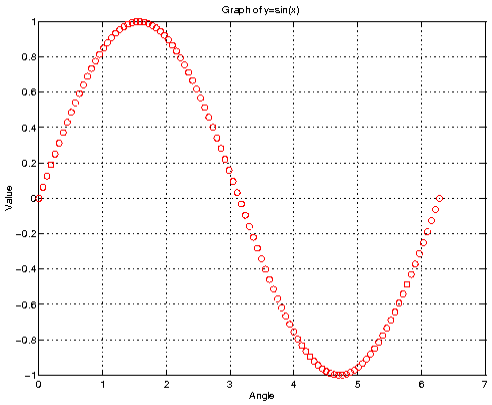
You can select the colour and the line style for the graph by using a third argument in the plot command. For example, to plot the graph instead with red circles, type

```
octave:##> plot(angles, y, 'ro')
```

The last argument is a string which describes the desired styles. Table 3 shows the possible values (also available by typing help plot in Octave).

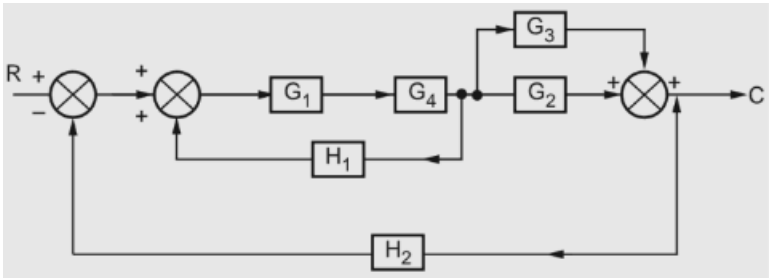
Table 3: Colours and styles for symbols and lines in the plot command

```
octave:##>  title('Graph of y=sin(x)')
octave:##>  xlabel('Angle')
octave:##>  ylabel('Value')
```



Aim: **Implement Block diagram reduction technique to obtain transfer function a control system**

Problem 1:



G1=1/s, G4=1/(s+1) , G2=s , G3=1, H1=s, H2=1/s

Code:

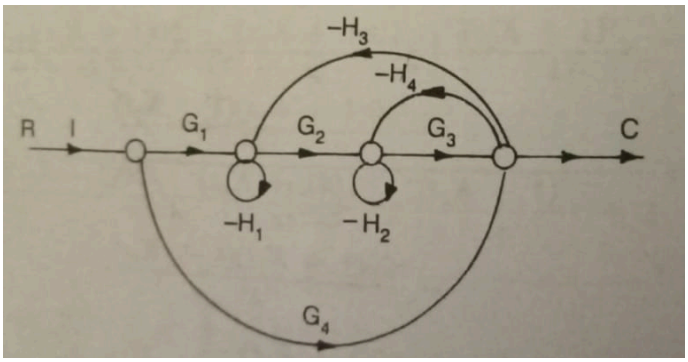
```
% Implement Block diagram reduction technique -Program1
pkg load control
clc
clf
num1=[1]
den1=[1 0];
tf1=tf(num1, den1) % transfer function 1/s
num2=[1]
den2=[1 1]
tf2=tf(num2, den2) % transfer function 1/s+1
num3=[1 0]
den3=[1]
tf3=tf(num3, den3)% transfer function s
tf4=series(tf1, tf2)
tf5=feedback(tf4,tf3)
num3=[1 0]
den3=[1]
tf6=tf(num3, den3)% transfer function s
tf7=parallel(tf6,1)
tf8=series(tf5, tf7)
num3=[1]
den3=[1 0]
tf9=tf(num3, den3)% transfer function 1/s
tfinal=feedback(tf8,tf9)
```

Output: Transfer function 'tfinal' from input 'u1' to output ...

```
Transfer function 'tfinal' from input 'u1' to output ...

           s^2 + s
 y1:  -------------------
      s^3 + 2 s^2 + s + 1
```

Experiment – 2

Aim: **Implement Signal Flow graph to obtain transfer function a control system.**




```
% Signal Flow graph (Masons Gain Formula)
pkg load control
clc
G1=1;   G2=2;   G3=3;   G4=4;
H1=-5;   H2=-6;    H3=-7;   H4=-8;

fp1=G1*G2*G3;
fp2=G4;

d1=1;
d2=1+H1+H2+H1*H2;

l1=H1;
l2=H2;
l3=G3*H4;
l4=G2*G3*H3;

ntl1=H1*H2;
ntl2=H1*(G3*H4);

num=( fp1*d1 + fp2*d2 )
d =1-(l1+l2+l3+l4)+(ntl1+ntl2)
tf = num/d
display(tf)
```

Output

```
num =  86

d =  228

tf =  0.37719
```

Solution

Forward Paths

(1) $G_1G_2G_3$

(2) G_4

Loops

(1) $- H_1$

(2) $- H_2$

(3) $- G_3H_4$

(4) $- G_2G_3H_3$

Non Touching Loops

(1) $- H_1$ and $- H_2$

(2) $- H_1$ and $- G_3H_4$

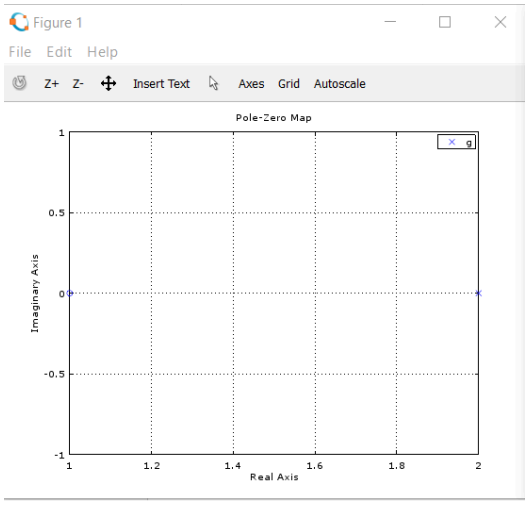
$$T = \frac{G_1G_2G_3 + G_4(1 + H_1 + H_2 + H_1H_2)}{1 + H_1 + H_2 + G_3H_4 + G_2G_3H_3 + H_1H_2 + G_1H_1H_4}$$ Ans.

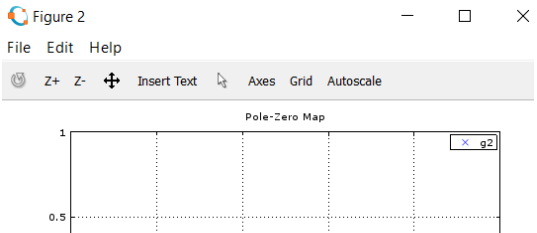
Experiment – 3

Aim: **Simulation of poles and zeros of a transfer function.**

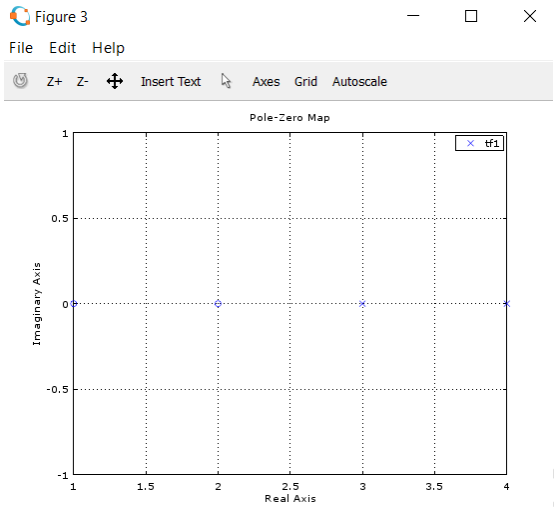
Code:

```
pkg load control
clf
clc
clear all
% Plot poles and zeros


 s = tf('s');
 g =(s-1)/(s-2);
pzmap(g);
figure

g2=(s+1)/(s-2)/(s+3);
 pzmap(g2);

num1=[1 -3 2]
```





```
den1=[1 -7 12];
tf1=tf(num1, den1)

figure
 pzmap(tf1);
```

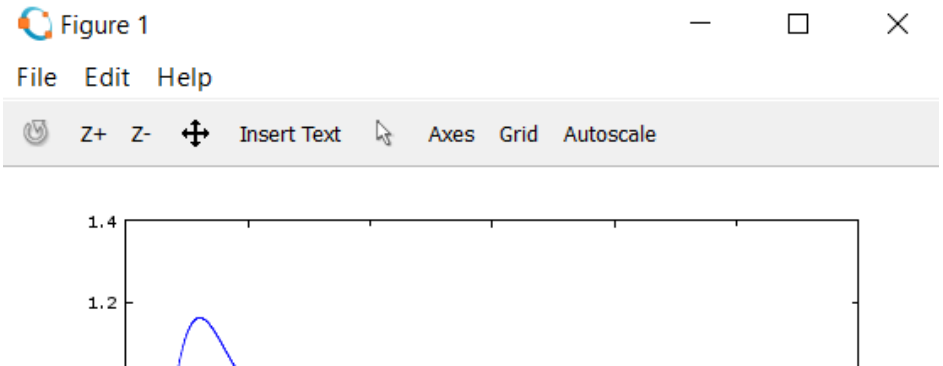


Experiment – 4

Aim: **Implement time response specification of a second order Under damped System, for different damping factors.**

Code:

```
pkg load control
clc

 s = tf('s');
sys =36/(s^2+6*s+36);

t = 0:0.001:6;
[y, t] = step(sys, t);
plot(t, y);
```



Maximum overshoot:

$$M_P = e^{-\frac{\pi \times 0.5}{\sqrt{1-(0.5)^2}}} \times 100$$

$$M_P = 0.163 \times 100 = 16.3\%$$

Peak Time:

$$t_p = \frac{\pi}{\omega_n \sqrt{1-\xi^2}}$$

$$t_p = \frac{\pi}{6\sqrt{1-0.5^2}} = 0.605\ sec.$$

Rise Time:

$$t_r = \frac{\pi - \tan^{-1}\frac{\sqrt{1-\xi^2}}{\xi}}{\omega_n\sqrt{1-\xi^2}} \qquad t_r = \frac{\pi - \tan^{-1}\frac{\sqrt{1-0.5^2}}{0.5}}{\omega_n\sqrt{1-0.5^2}}$$

$$= \frac{3.14 - 1.047}{5.19} = 0.403\text{sec.}$$

Settling Time: $t_s = \frac{4}{\xi\omega_n}$ $\qquad t_s = \frac{4}{0.5 \times 6} = 1.33\ sec.$

Experiment – 5

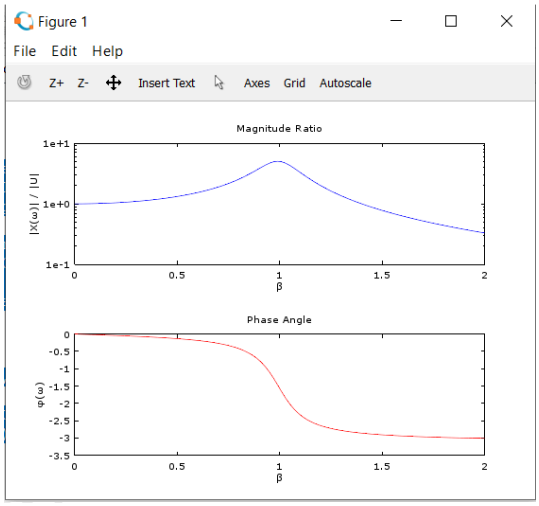
Aim: **Implement frequency response of a second order System.**

Code

```
pkg load control
clc

% Define system parameters
omega_n = 10;  % Natural frequency
zeta = 0.1;    % Damping ratio

% Excitation frequency ratio
beta = linspace(0, 2, 100);  % Vary beta from 0 to 2
X_over_U = 1 ./ sqrt((1 - beta.^2).^2 + (2*zeta*beta).^2);
phi = atan2(-2*zeta*beta, 1 - beta.^2);
```

```
% Plot magnitude ratio
subplot(2, 1, 1);
semilogy(beta, X_over_U, 'b');
xlabel('\beta');
ylabel('|X(\omega)| / |U|');
title('Magnitude Ratio');

% Plot phase angle
subplot(2, 1, 2);
plot(beta, phi, 'r');
xlabel('\beta');
ylabel('\phi(\omega)');
title('Phase Angle');

% Adjust plot appearance
sgtitle('Frequency Response of Second-Order System');
```

Output:



Experiment – 6

Aim: Analyze the stability of the given system using Routh stability criterion.

Code:

```
pkg load control
clc
% Taking coefficients vector and organizing the first two rows
coeffVector = input('input vector of your system coefficients: \n i.e. [an an-1 an-2 ... a0] = ');
ceoffLength = length(coeffVector);
rhTableColumn = round(ceoffLength/2);
%  Initialize Routh-Hurwitz table with empty zero array
rhTable = zeros(ceoffLength,rhTableColumn);
%  Compute first row of the table
rhTable(1,:) = coeffVector(1,1:2:ceoffLength);
%  Check if length of coefficients vector is even or odd
```

```
if (rem(ceoffLength,2) ~= 0)
   % if odd, second row of table will be
   rhTable(2,1:rhTableColumn - 1) = coeffVector(1,2:2:ceoffLength);
else
   % if even, second row of table will be
   rhTable(2,:) = coeffVector(1,2:2:ceoffLength);
end
%% Calculate Routh-Hurwitz table's rows
%  Set epss as a small value
epss = 0.01;
%  Calculate other elements of the table
for i = 3:ceoffLength

   %  special case: row of all zeros
   if rhTable(i-1,:) == 0
      order = (ceoffLength - i);
      cnt1 = 0;
      cnt2 = 1;
      for j = 1:rhTableColumn - 1
         rhTable(i-1,j) = (order - cnt1) * rhTable(i-2,cnt2);
         cnt2 = cnt2 + 1;
         cnt1 = cnt1 + 2;
      end
   end


   for j = 1:rhTableColumn - 1
      %  first element of upper row
      firstElemUpperRow = rhTable(i-1,1);

      %  compute each element of the table
      rhTable(i,j) = ((rhTable(i-1,1) * rhTable(i-2,j+1)) - ....
         (rhTable(i-2,1) * rhTable(i-1,j+1))) / firstElemUpperRow;
   end


   %  special case: zero in the first column
   if rhTable(i,1) == 0
      rhTable(i,1) = epss;
   end
end
%%  Compute number of right hand side poles(unstable poles)
%   Initialize unstable poles with zero
unstablePoles = 0;
%   Check change in signs
for i = 1:ceoffLength - 1
   if sign(rhTable(i,1)) * sign(rhTable(i+1,1)) == -1
      unstablePoles = unstablePoles + 1;
   end
end
%   Print calculated data on screen
```

```
fprintf('\n Routh-Hurwitz Table:\n')
rhTable
%   Print the stability result on screen
if unstablePoles == 0
   fprintf('~~~~~> it is a stable system! <~~~~~\n')
else
   fprintf('~~~~~> it is an unstable system! <~~~~~\n')
end
   fprintf('\n Number of right hand side poles =%2.0f\n',unstablePoles)
```

Output:

```
input vector of your system coefficients: \n i.e. [an an-1 an-2 ... a0] = [1 2 3 4 5 6]

 Routh-Hurwitz Table:
rhTable =

    1.00000    3.00000    5.00000
    2.00000    4.00000    6.00000
    1.00000    2.00000    0.00000
    0.01000    6.00000    0.00000
 -598.00000    0.00000    0.00000
    6.00000    0.00000    0.00000

~~~~~> it is an unstable system! <~~~~~
Number of right hand side poles = 2
```

Example 8.8.1 *Comment on stability using Routh criteria if characteristic equation is :*

$Q(s) = s^5 + 2s^4 + 3s^3 + 4s^2 + 5s + 6 = 0$

How many poles lie in right half of s-plane ?

Solution : The Routh's array is,

s^5	1	3	5	
s^4	2	4	6	
s^3	1	2	0	
s^2	$0 + \varepsilon$	6		⇐ Special case 1
s^1	$\frac{2\varepsilon - 6}{\varepsilon}$	0		$\lim_{\varepsilon \to 0} \frac{2\varepsilon - 6}{\varepsilon} \to -Ve$
s^0	6			

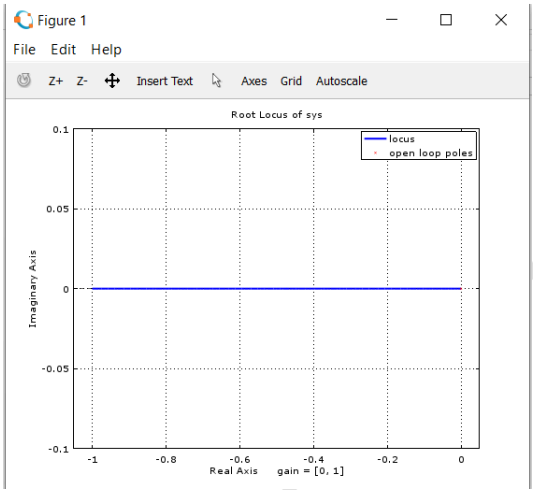
Thus there are two sign changes in the first column hence two poles lie in right half of s-plane.

System unstable.

Aim: **Analyze the stability of the given system using Root locus**

Code 1:

```
pkg load control
clc
%sys = tf([2 5 1],[1 2 3]);
 s = tf('s');
 g =1/(s-0);
 sys = tf(g);
rlocus(sys)
```



root locus.

Example 9.2.1 *Consider unity feedback system with* $G(s) = \frac{K}{s}$. *Obtain its roots locus.*

Solution : The characteristic equation becomes,

$$1 + G(s)H(s) = 0, \quad H(s) = 1$$

$$\therefore \quad 1 + \frac{K}{s} = 0$$

$$s + K = 0$$

The root of this equation is located at $s = -K$.

Now if gain 'K' is varied from 0 to + ∞, the location of this root is going to change.

The locus obtained by joining all such locations when K is varied from 0 to + ∞ is alled root locus.

K	s = – K Root location
0	0
1	– 1
10	– 10
:	:
+ ∞	– ∞

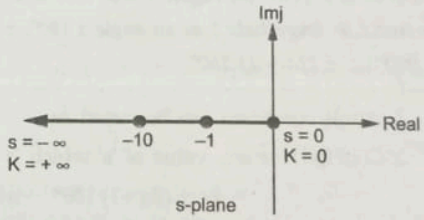
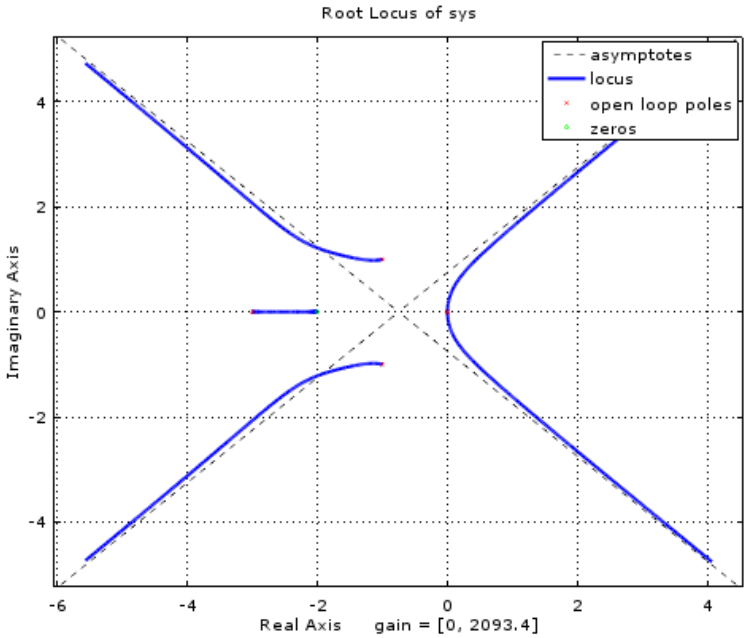


Fig. 9.2.2

Code 2:

```
pkg load control
clc
s = tf('s');
g =(s+2)/(s^2+2*s+2)/(s^3+3*s^2);
sys = tf(g);
rlocus(sys)
```



Code 3:

Example 9.5.2 $G(s)H(s) = \dfrac{K(s+2)}{s^2(s^2+2s+2)(s+3)}$. *Find the sections of real axis which belongs to the root locus.*

Solution : Poles are at 0, 0, $-1 \pm j$, -3. [Zero is at -2.]

Pole - zero plot is as shown in the Fig. 9.5.3.

For positive real axis, there is no pole and zero to right hand side so sum is zero and hence there is no root locus.

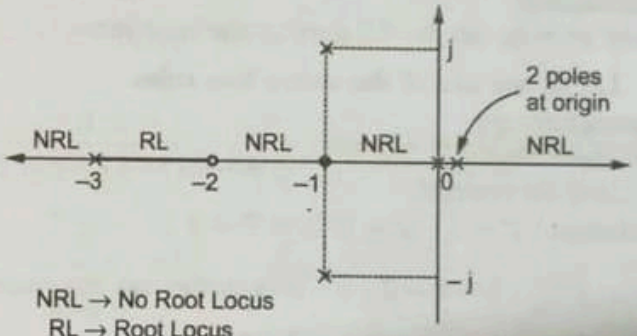
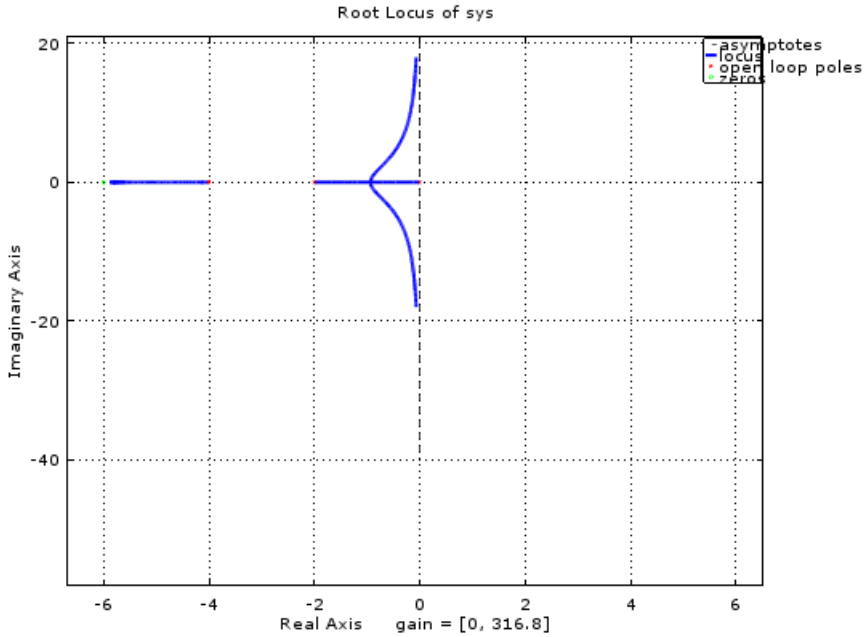


Fig. 9.5.3

For next section between s = 0 to s = –2, to the right hand side sum is 2 which is even hence there is no root locus. Complex conjugate roots should not be considered while using this rule. For next section between s = –2 and s = –3, to right hand side sum is 3, odd hence full section is part of the root locus. While section to left of s = – 3 there is no root locus.

```
pkg load control
clc
s = tf('s');
 g =(s+6)/(s+2)/(s^2+4*s);
 sys = tf(g);
rlocus(sys)
```



Example 9.5.4 *For* $G(s)H(s) = \dfrac{K(s+6)}{s(s+2)(s+4)}$ *how many minimum breakaway points exist?*

Solution : In this case there are two pairs of adjacently placed poles on real axis.

$s = 0$ and $s = -2$, section between them is a part of root locus hence minimum one breakaway point exists in between them.

Now $s = -2$ and $s = -4$ also forms a pair of adjacently placed poles but section between them is not the part of root locus and hence there cannot be a breakaway point in between them. So minimum one breakaway point exists between $s = 0$ and $s = -2$. In such case, **branches are approaching towards breakaway point from the poles.**

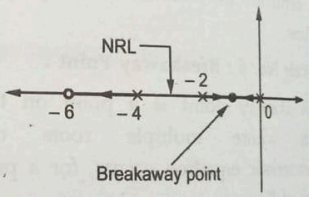
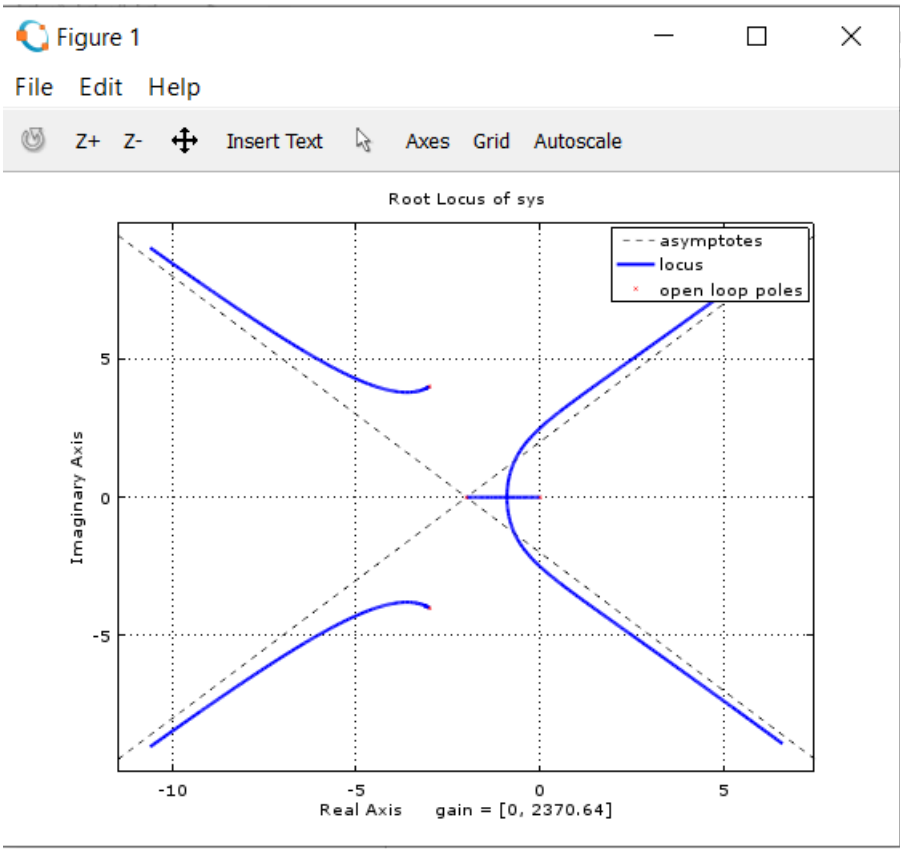
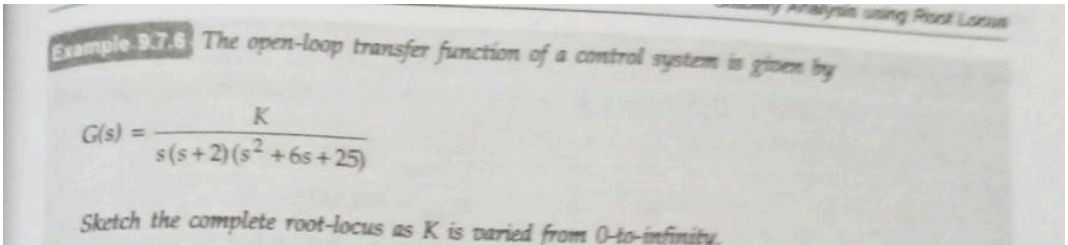


Fig. 9.5.7

Code 4:

```
pkg load control
clc
%sys = tf([2 5 1],[1 2 3]);
 s = tf('s');
g =10/(s^2+2*s)/(s^2+6*s+25);
 sys = tf(g);
rlocus(sys)
```





Example 9.7.6 The open-loop transfer function of a control system is given by

$$G(s) = \frac{K}{s(s+2)(s^2+6s+25)}$$

Sketch the complete root-locus as K is varied from 0-to-infinity.

Step 6 : Intersection with imaginary axis

$s^4 + 8s^3 + 37s^2 + 50s + K = 0$ ... Characteristic equation

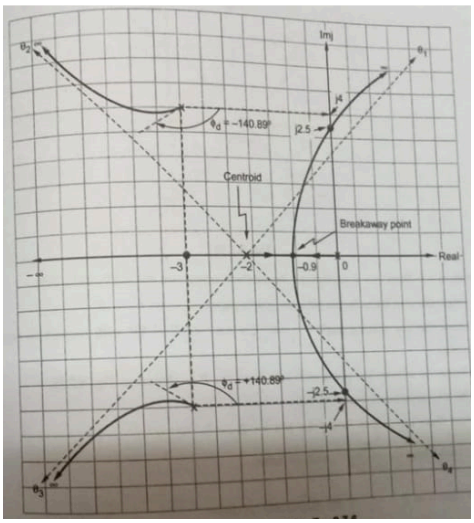
s^4	1	37	K
s^3	8	50	0
s^2	30.75	K	
	$1537.5 - 8K$		

For K_{mar}, $1537.5 - 8K = 0$

$\therefore K_{mar} = \mathbf{192.1875}$

$A(s) = 30.75\, s^2 + K = 0$

$\therefore \quad s^2 = -\frac{K}{30.75}$

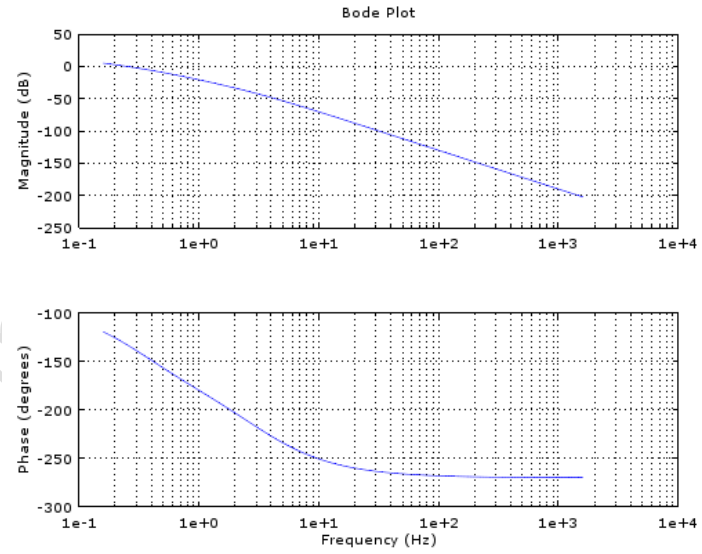


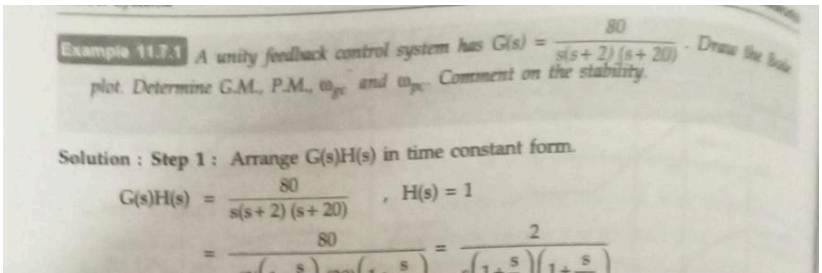
Experiment – 8

Aim: **Analyze the stability of the given system using Bode plots.**

Code 1:

```
pkg load control
clc

% Define the transfer function
s = tf('s');
g = 80/(s^2+2*s)/(s+20);
% Specify the frequency range (1 Hz to 10^4 Hz)
freq_range = {1, 10^4};

% Compute the Bode plot
[bmag, bphase, bfreq] = bode(g, freq_range);

% Convert frequency from rad/s to Hz
bfreq_hz = bfreq / (2 * pi);

% Plot the magnitude response
subplot(2, 1, 1);
semilogx(bfreq_hz, 20 * log10(bmag));
grid on;
title('Bode Plot');
ylabel('Magnitude (dB)');

% Plot the phase response
subplot(2, 1, 2);
semilogx(bfreq_hz, bphase);
grid on;
xlabel('Frequency (Hz)');
ylabel('Phase (degrees)');
```



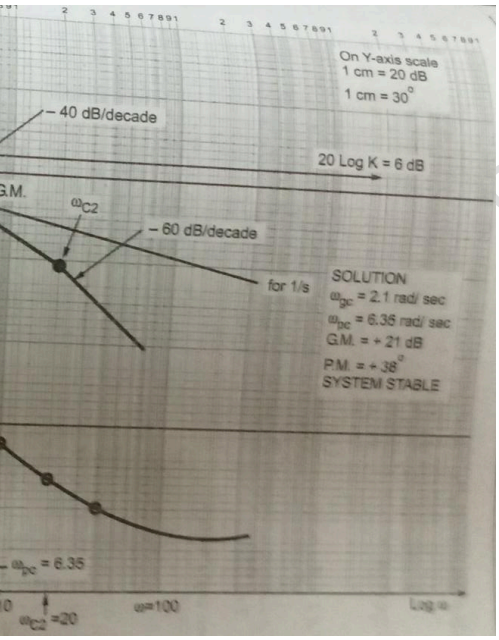


Example 11.7.1 A unity feedback control system has $G(s) = \frac{80}{s(s+2)(s+20)}$. Draw the Bode plot. Determine G.M., P.M., ω_{gc} and ω_{pc}. Comment on the stability.

Solution : Step 1 : Arrange G(s)H(s) in time constant form.

$$G(s)H(s) = \frac{80}{s(s+2)(s+20)}, \quad H(s) = 1$$

$$= \frac{80}{\left(\quad s\right)\left(\quad s\right)} = \frac{2}{\left(1+\frac{s}{\quad}\right)\left(1+\frac{s}{\quad}\right)}$$



Experiment – 9

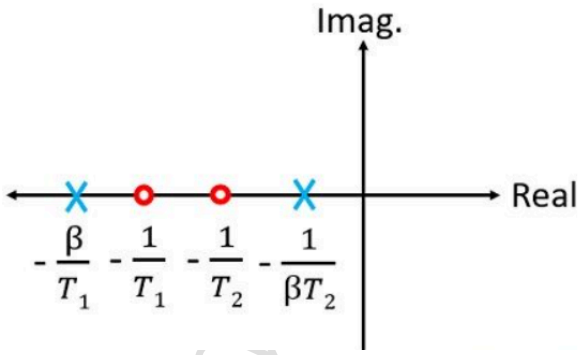
Aim: **Implement frequency response of a lead lag compensator.**

Theory: A lead-lag compensator is a component in a control system that combines a lead compensator and a lag compensator to improve a system's frequency response. It's a fundamental building block in classical control theory and is used in a variety of disciplines, including robotics, satellite control, and automobile diagnostics.

A lag lead compensator is a type of electrical network that generates phase lag as well as phase lead in the output signal at different frequencies when a steady-state sinusoidal input is provided to it. It is also known as **lag lead network**.

In case of a lag lead compensator, the phase lead and phase lag occur at different frequency regions. Generally, at low-frequency phase lag characteristics is noticed in the output of the circuit. While at high frequency, we have phase lead characteristics.

The figure below shows the pole-zero plot of the lag lead compensator:



A lead-lag compensator combines the effects of a lead compensator with those of a lag compensator. The result is a system with improved transient response, stability, and steady-state error. To implement a lead-lag compensator, first design the lead compensator to achieve the desired transient response and stability, and then design a lag compensator to improve the steady-state response of the lead-compensated system.

Lead or phase-lead compensator using frequency response

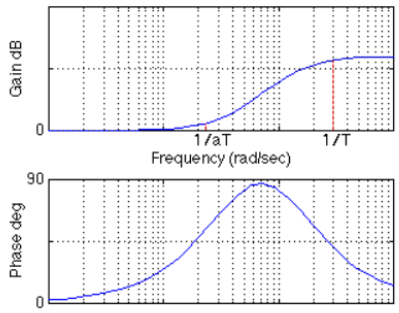
A first-order phase-lead compensator can also be designed using a frequency reponse approach. A lead compensator in frequency response form is given by the following.

$$C(s) = \frac{1 + aTs}{1 + Ts} \qquad [a > 1] \tag{3}$$

Note that this is equivalent to the root locus form repeated below

$$C(s) = K_c \frac{(s - z_0)}{(s - p_0)} \tag{4}$$

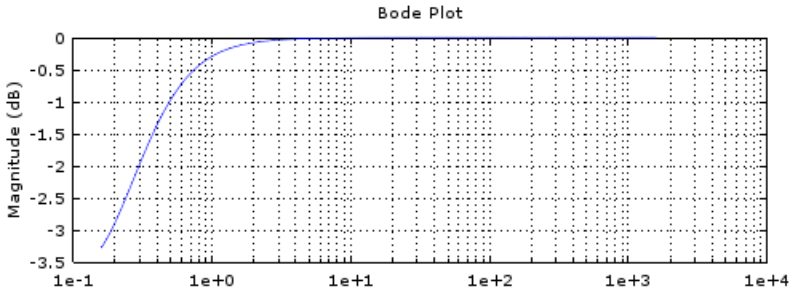
with p =1 / T, z = 1 / aT, and Kc = a. In frequency response design, the phase-lead compensator adds positive phase to the system over the frequency range 1 / aT to 1 / T. A Bode plot of a phase-lead compensator $C(s)$ has the following form.

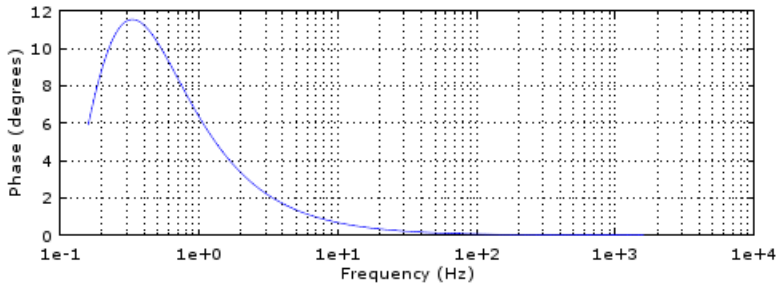


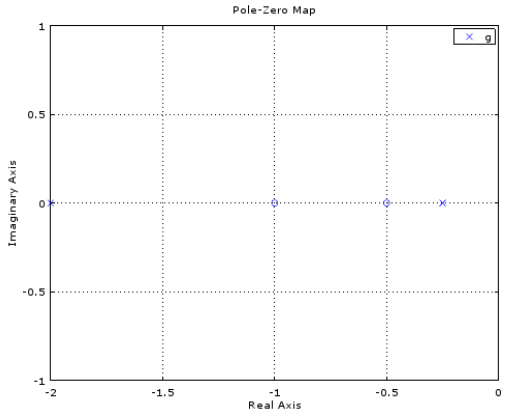
Code :

```
pkg load control
clc

% Define the transfer function
num1=[2 3 1]
den1=[2 4.5 1];
g=tf(num1, den1)
% Specify the frequency range (1 Hz to 10^4 Hz)
freq_range = {1, 10^4};

% Compute the Bode plot
[bmag, bphase, bfreq] = bode(g, freq_range);

% Convert frequency from rad/s to Hz
bfreq_hz = bfreq / (2 * pi);

% Plot the magnitude response
subplot(2, 1, 1);
semilogx(bfreq_hz, 20 * log10(bmag));
grid on;
title('Bode Plot');
ylabel('Magnitude (dB)');

% Plot the phase response
subplot(2, 1, 2);
```

```
semilogx(bfreq_hz, bphase);
grid on;
xlabel('Frequency (Hz)');
ylabel('Phase (degrees)');
```





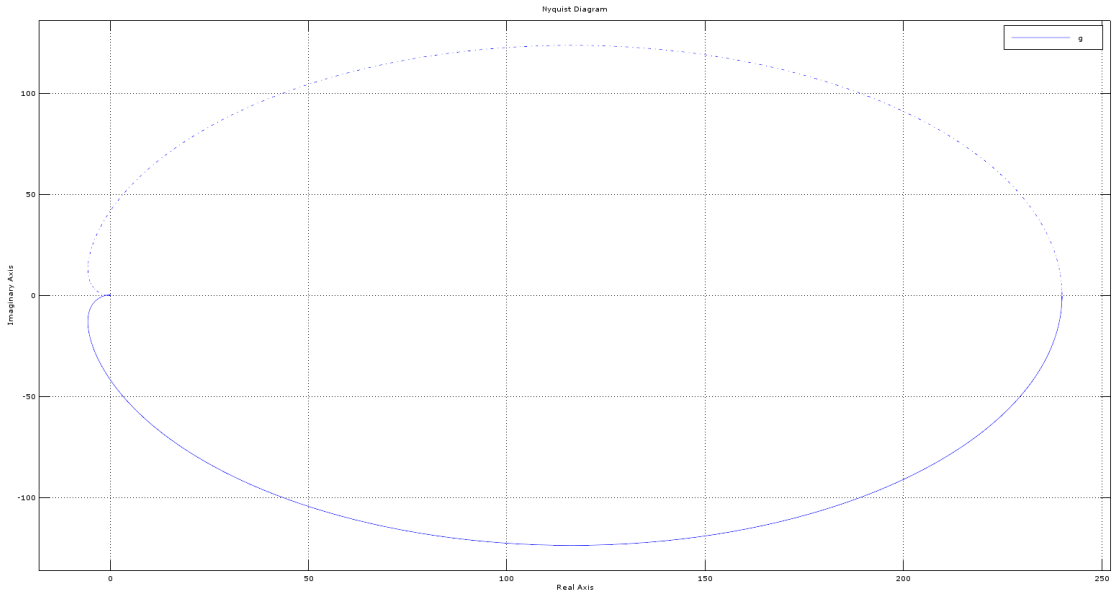


Experiment – 10

Aim: **Analyze the stability of the given system using Nyquist plot.**

Code:

```
pkg load control
clc

% Define the transfer function
num1=[240]
den1=[1 12 20 1];
g=tf(num1, den1)
nyquist(g)
```



Example 12.13.1 *For a certain control system*

$$G(s)H(s) = \frac{K}{s(s+2)(s+10)}$$

Sketch the Nyquist plot and hence calculate the range of values of K for stability.

Solution : Step 1 : $P = 0$

Step 2 : $N = -P = 0$, the critical point $-1 + j0$ should not get encircled by Nyquist plot.

Step 3 : Pole at origin hence Nyquist path is as shown in the Fig. 12.13.1.

Step 4 :

$$G(j\omega)H(j\omega) = \frac{K}{j\omega(2+j\omega)(10+j\omega)}$$

$$M = |G(j\omega)H(j\omega)|$$

$$= \frac{K}{\omega \times \sqrt{4+\omega^2} \times \sqrt{100+\omega^2}}$$

$$\phi = \frac{\tan^{-1}\left(\frac{0}{K}\right)}{\tan^{-1}\left(\frac{\omega}{0}\right)\tan^{-1}\left(\frac{\omega}{2}\right)\tan^{-1}\left(\frac{\omega}{10}\right)}$$

$$= -90° - \tan^{-1}\frac{\omega}{2} - \tan^{-1}\frac{\omega}{10}$$

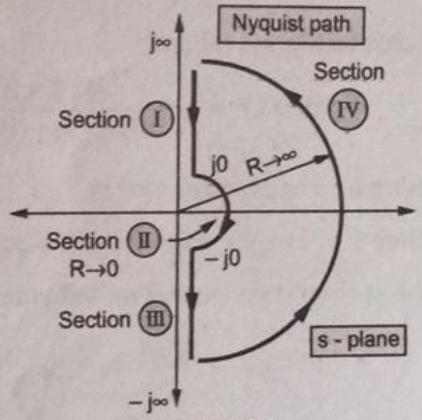


Fig. 12.13.1

Section I : $s = +j\infty$ to $s = +j0$ i.e. $\omega \to \infty$ to $\omega \to +0$

Starting point	$\omega \to \infty$	$0 \angle -270°$	$-90° - (-270)° = +180°$ Anticlockwise rotation
Terminating point	$\omega \to +0$	$0 \angle +90°$	

Section II : $s = +j0$ to $s = -j0$ i.e. $\omega \to +0$ to $\omega \to -0$

Starting point	$\omega \to +0$	$\infty \angle -90°$	$90° - (-90)° = +180°$ Anticlockwise rotation
Terminating point	$\omega \to -0$	$\infty \angle +90°$	

Section III is mirror image of section about real axis.

Section IV is an origin.

Step 5 :

$$G(j\omega)H(j\omega) = \frac{K}{j\omega(10+j\omega)(2+j\omega)}$$

Rationalizing ,

$$G(j\omega)H(j\omega) = \frac{K(-j\omega)(10-j\omega)(2-j\omega)}{(j\omega)(-j\omega)(10+j\omega)(10-j\omega)(2+j\omega)(2-j\omega)}$$

$$\therefore \quad G(j\omega)H(j\omega) = \frac{-Kj\omega[20-12j\omega-\omega^2]}{\omega^2(4+\omega^2)(100+\omega^2)} = \frac{-12K\omega^2}{D} - \frac{Kj\omega(20-\omega^2)}{D}$$

where $D = \omega^2(4+\omega^2)(100+\omega^2)$

Equating imaginary part to zero,

$$\omega(20-\omega^2) = 0 \quad \text{i.e.} \quad \omega^2 = 20 \quad \text{i.e.} \quad \omega_{pc} = \sqrt{20}$$

Substituting in real part,

$$\text{Point Q} = \frac{-12K\times 20}{20\times(20+4)\times(100+20)} = -\frac{K}{240}$$

Step 6 : The Nyquist plot is,

Step 7 : Now for absolute stability, N = 0

i.e. it should be located on left side of point Q i.e. $|OQ| < 1$

$$\therefore \quad \left|-\frac{K}{240}\right| < 1$$

$$\therefore \quad K < 240$$

So range of values of K for stability is

$$0 < K < 240$$

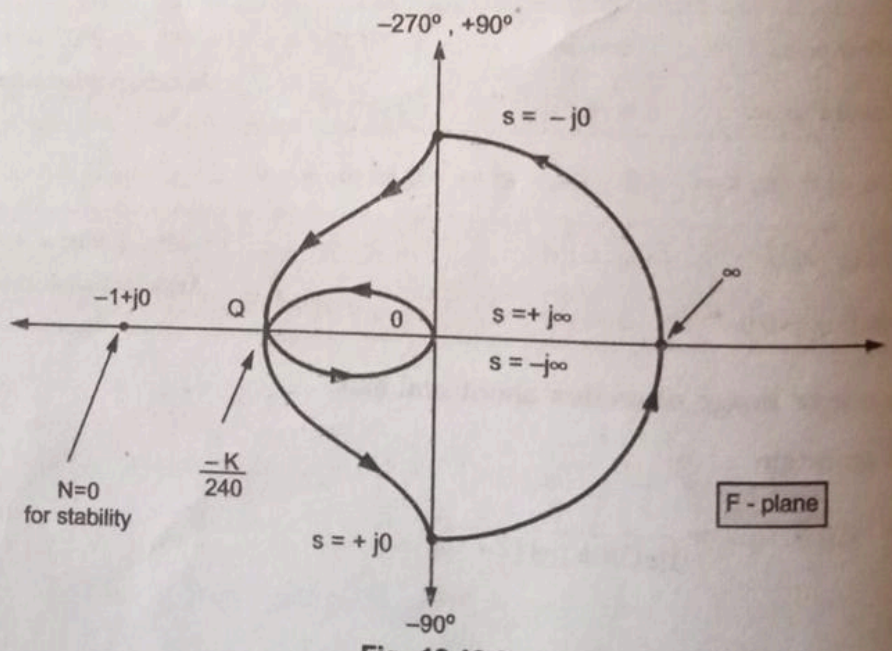


Fig. 12.13.2

Experiment – 11

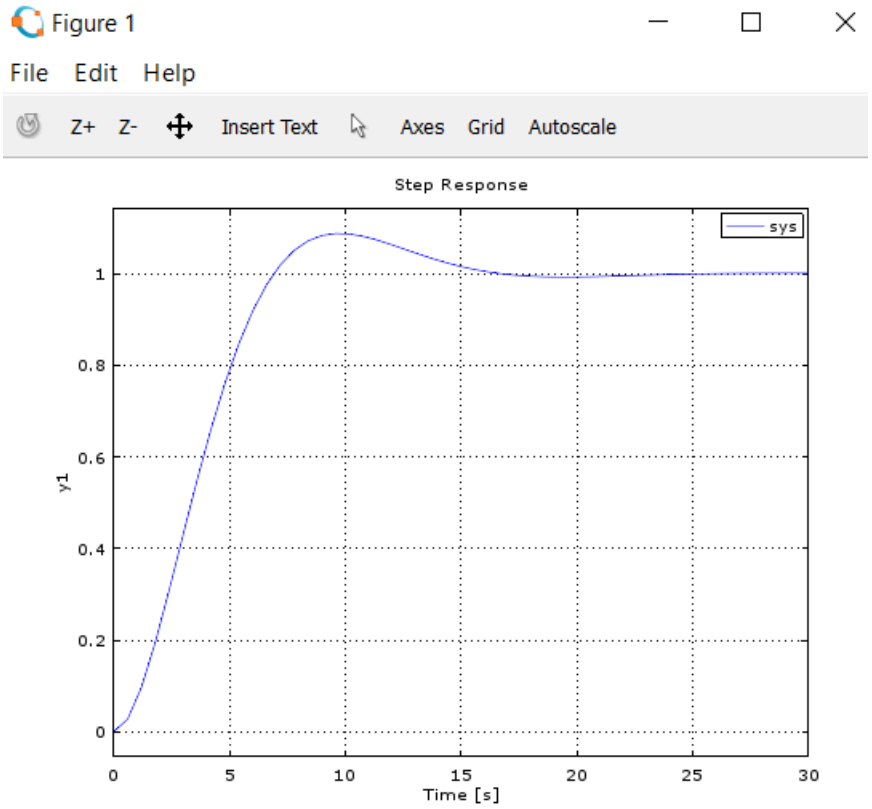
Aim: **Obtain the time response from state model of a system**

Code:

```
pkg load control
clc
A = [-0.5 -0.5;0.333 0];
B = [0.5;0];
C = [0 1];
D = 0;
sys = ss(A,B,C,D);
step(sys)
```

Output:

```
sys.a =
       x1    x2
  x1   -0.5   -0.5
  x2  0.333     0

sys.b =
       u1
  x1  0.5
  x2    0

sys.c =
      x1  x2
  y1   0   1

sys.d =
      u1
  y1   0
```



Example 13.4.1 *Obtain the state model of the given electrical system.*

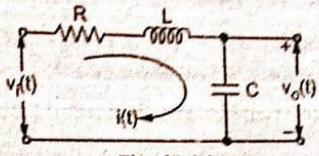


Fig. 13.4.1

lution : There are two energy storing elements L and C. So the two state variables are rrent through inductor i(t) and voltage across capacitor i.e. $v_o(t)$.

$$X_1(t) = i(t) \quad \text{and} \quad X_2(t) = v_o(t)$$

nd $U(t) = v_i(t)$ = Input variable

Applying KVL to the loop,

$$v_i(t) = i(t)\,R + L\frac{di(t)}{dt} + v_o(t)$$

Arrange it for di(t)/dt,

$$\frac{di(t)}{dt} = \frac{1}{L}v_i(t) - \frac{R}{L}i(t) - \frac{1}{L}v_o(t) \quad \text{but} \quad \frac{di(t)}{dt} = \dot{X}_1(t)$$

e. $$\dot{X}_1(t) = -\frac{R}{L}X_1(t) - \frac{1}{L}X_2(t) + \frac{1}{L}U(t) \qquad \ldots (1)$$

While $v_o(t)$ = Voltage across capacitor $= \frac{1}{C}\int i(t)\,dt$

$$\therefore \quad \frac{dv_o(t)}{dt} = \frac{1}{C}i(t) \quad \text{but} \quad \frac{dv_o(t)}{dt} = \dot{X}_2(t)$$

i.e. $$\dot{X}_2(t) = \frac{1}{C}X_1(t) \qquad \ldots (2)$$

The equations (1) and (2) give required state equations.

$$\begin{bmatrix}\dot{X}_1(t)\\ \dot{X}_2(t)\end{bmatrix} = \begin{bmatrix}-\frac{R}{L} & -\frac{1}{L}\\ \frac{1}{C} & 0\end{bmatrix}\begin{bmatrix}X_1(t)\\ X_2(t)\end{bmatrix} + \begin{bmatrix}\frac{1}{L}\\ 0\end{bmatrix}U(t)$$

i.e. $$\dot{X}(t) = A\,X(t) + B\,U(t)$$

While the output variable $Y(t) = v_o(t) = X_2(t)$

$\therefore$ $$Y(t) = [0\ \ 1]\begin{bmatrix}X_1\\ X_2\end{bmatrix} + [0]\,U(t)$$

i.e. $$Y(t) = C\,X(t) \quad \text{and} \quad D = [0]$$

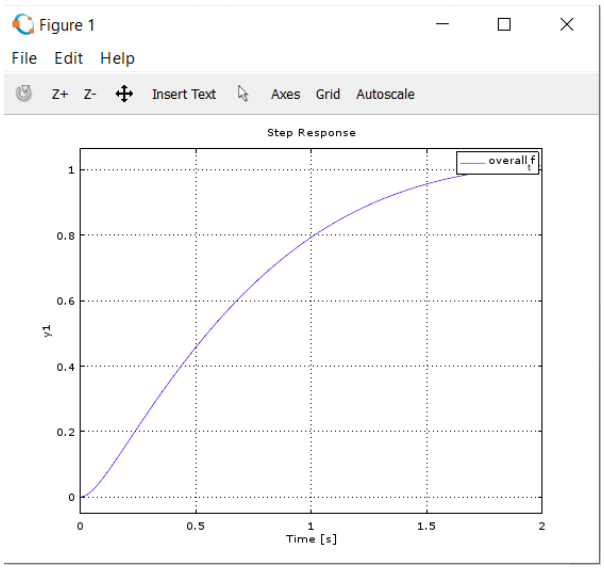
This is the required state model. As n = 2, it is a second order system.

Experiment – 12

Aim: **Implement PI and PD Controllers**

Code:

```
pkg load control
clf
clc
clear all
kp=20;
ki=50
num1=[kp, ki];
den1=[1 0];
controller_tf = tf (num1, den1);
num2=1;
den2=[1 20 30];
plant_tf=tf(num2, den2);
overall_tf=feedback(controller_tf*plant_tf, 1);
t=0:0.01:2;
step (overall_tf, t);
```



Given, transfer function of a plant is $P(s) = \frac{1}{s^2+20s+30}$

Solution:

Block Diagram

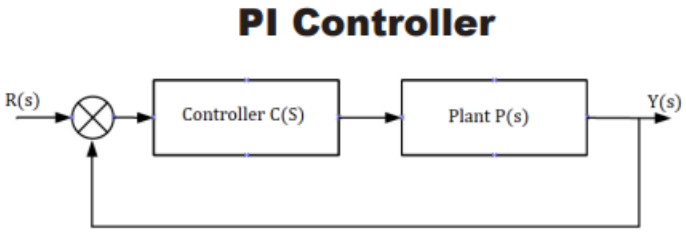


Figure 7.1 : Block diagram of cliosed loop control system.

We Know that, the tranfer function of PI controller

$$C(s) = K_P + \frac{K_i}{s}$$

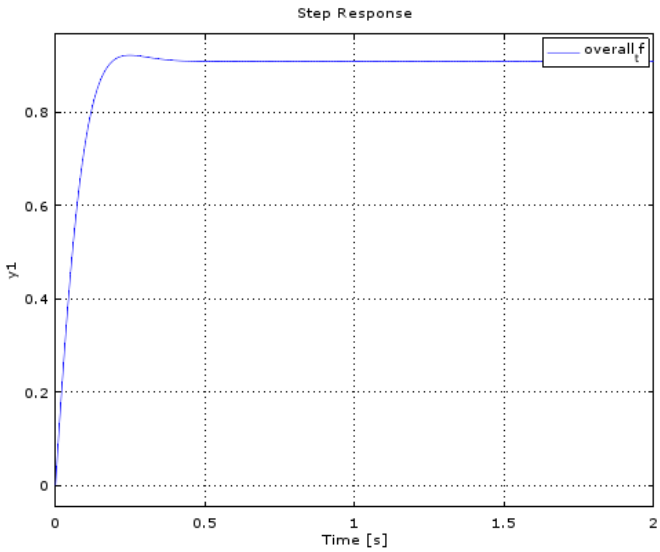
Tranfer function after introducing PI controller

$$T(s) = \frac{(K_P s + K_I)}{s^3 + 10s^2 + (20 + K_P)s + K_I}$$

Let, K_P=20 and K_I=50

Code:

```
pkg load control
clf
clc
clear all
kp=300;
kd=10
num1=[kd, kp];
den1=1;
controller_tf = tf (num1, den1);
num2=1;
den2=[1 20 30];
plant_tf=tf(num2, den2);
overall_tf=feedback(controller_tf*plant_tf,
1);
t=0:0.01:2;
step (overall_tf, t);
```



Given, transfer function of a plant is $P(s) = \frac{1}{s^2+20s+30}$

Solution:

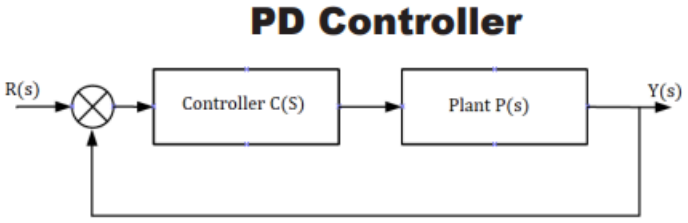


Figure 8.1 : Block diagram of cliosed loop control system

We Know that, the tranfer function of PI controller

$$C(s) = K_P + K_d s$$

Tranfer function after introducing PI controller

$$T(s) = \frac{(K_D s + K_P)}{s^2 + (10 + K_D)s + (20 + K_P)}$$

Let, K_P=200 and K_D=10

Sample Viva Questions

Q-1 **What is frequency response?**

Ans: A frequency response is the steady state response of a system when the input to the system is a sinusoidal signal.

Q-2 **List out the different frequency domain specifications?**

Ans: The frequency domain specification are 1. Resonant peak. 2. Resonant frequency.

Q-3 **Define –resonant Peak?**

Ans: The maximum value of the magnitude of closed loop transfer function is called resonant peak.

Q-4 **Define –Resonant frequency?**

Ans: The frequency at which resonant peak occurs is called resonant frequency.

Q-5 **What is bandwidth?**

Ans: The bandwidth is the range of frequencies for which the system gain Is more than 3 dbB. The bandwidth is a measure of the ability of a feedback system to reproduce the input signal, noise rejection characteristics and rise time.

Q-6 **Define Cut-off rate?**

Ans: The slope of the log-magnitude curve near the cut-off is called cut-off rate. The cutoff rate indicates the ability to distinguish the signal from noise.

Q-7 **Define – Gain Margin?**

Ans: The gain margin is defined as the reciprocal of the magnitude of the open loop transfer function at phase cross over frequency.

Q-8 **Define Phase cross over?**

Ans :The frequency at which, the phase of open loop transfer functions is called phase cross over frequency ωpc.

Q-9 **What is phase margin?**

Ans : The phase margin is the amount of phase lag at the gain cross over frequency required to bring system to the verge of instability.

Q-10 **What are the main significances of root locus?**

Ans:

1. The main root locus technique is used for stability analysis.

2. Using root locus technique the range of values of K, for as table system can be determined

Q-11 **Define Gain cross over?**

Ans :The gain cross over frequency ωgc is the frequency at which the magnitude of the open loop transfer function is unity.

Q-12 **What is Bode plot?**

Ans: The Bode plot is the frequency response plot of the transfer function of a system. A Bode plot consists of two graphs. One is the plot of magnitude of sinusoidal transfer function versus log ω. The other is a plot of the phase angle of a sinusoidal function versus log ω.

Q-13 **What are the main advantages of Bode plot?**

Ans: The main advantages are:

1. Multiplication of magnitude can be in to addition.

2. A simple method for sketching an approximate log curve is available.

3. It is based on asymptotic approximation. Such approximation is sufficient if rough information on the frequency response characteristic is needed.

Q-14 **Define Corner frequency?**

Ans The frequency at which the two asymptotic meet in a magnitude plot is called corner frequency. Q-

15 **Define Phase lag and phase lead?**

Ans A negative phase angle is called phase lag. A positive phase angle is called phase lead.

Q-16 **What are M circles?**

Ans The magnitude of closed loop transfer function with unit feedback can be shown to be in the for every value if M. These circles are called M circles.

Q-17 **What is Nichols chart?**

Ans The chart consisting if M & N loci in the log magnitude versus phase diagram is called Nichols chart.

Q-18 **What are two contours of Nichols chart?**

Ans :Nichols chart of M and N contours, superimposed on ordinary graph. The M contours are the magnitude of closed loop system in decibels and the N contours are the phase angle locus of closed loop system.

Q-19 **How is the Resonant Peak(Mr), resonant frequency(ωr) , and band width determined from Nichols chart?**

Ans :1. The resonant peak is given by the value of contour which is tangent to G(jω) locus.

2. The resonant frequency is given by the frequency of G(jω) at the tangency point. 3. iii) The bandwidth is given by frequency corresponding to the intersection point of G(jω) and –3dB M-contour.

Q-20 **What are the advantages of Nichols chart?**

Ans: The advantages are: 1. It is used to find the closed loop frequency response from open loop frequency response. 2. Frequency domain specifications can be determined from Nichols chart. 3. The gain of the system can be adjusted to satisfy the given specification.

Q-21:**What are the two types of compensation?**

Ans :1. Cascade or series compensation 2. Feedback compensation or parallel compensation

Q-22 **What are the three types of compensators?**

Ans: 1. Lag compensator 2. Lead compensator 3. Lag-Lead compensator

Q-23 **What are the uses of lead compensator?**

Ans: 1. speeds up the transient response 2. increases the margin of stability of a system 3. Increases the system error constant to a limited extent.

Q-24 **What is the use of lag compensator?**

Ans: Improve the steady state behaviour of a system, while nearly preserving its transient response.

Q-25 **What are the basic elements used for modelling mechanical rotational system?**

Ans: Moment of inertia J, dashpot with rotational frictional coefficient B and torsional spring with stiffness K