EXPERIMENT NO: 1 **DATE:**

AIM: INTRODUCTION TO THE MATLAB SOFTWARE.

EQUIPMENT: PC system containing MATLAB SOFTWARE.

THEORY:

Overview of the MATLAB Environment:

The MATLAB high-performance language for technical computing integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. It allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. You can add on toolboxes for signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many other areas.

The MATLAB system consists of these main parts:

- **Desktop Tools and Development Environment**

This is the set of tools and facilities that help you use and become more productive with MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, a code analyzer and other reports, and browsers for viewing help, the workspace, files, and the search path.

- **Mathematical Function Library**

This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

- **The Language**

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create large and complex application programs.

- **Graphics**

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

- **External Interfaces**

This is a library that allows you to write C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), for calling MATLAB as a computational engine, and for reading and writing MAT-files.

- **Entering Matrices:**

The best way for you to get started with MATLAB is to learn how to handle matrices. Start MATLAB and follow along with each example.

Matrices Operations in MATLAB

You can enter matrices in MTLAB in following different ways
- Enter an explicit list of elements.
- Load matrices from external data files.
- Generate matrices using built-in functions.
- Create matrices with your own functions in M-files.

Start by entering Dürer's matrix as a list of its elements. You only have to follow a few basic conventions:
- Separate the elements of a row with blanks or commas.
- Use a semicolon, ; , to indicate the end of each row.
- Surround the entire list of elements with square brackets, [ ].

❖ **To Enter Dürer's Matrix, Simply Type In The Command Window**

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

MATLAB displays the matrix you just entered:

```
A =
   16    3    2   13
    5   10   11    8
    9    6    7   12
    4   15   14    1
```

❖ **Sum, Transpose, Diagonal And Magic**

```
>> sum(A)
ans =
    34   34   34   34
>> A'
ans =

   16    5    9    4
    3   10    6   15
    2   11    7   14
   13    8   12    1

>>diag(A)
ans =
   16
   10
    7
    1
```

A MAGIC MATRIX: A Magic Square Is A Matrix In Which The Sum Of The Elements In Each Column, Or Each Row, Or Each Main Diagonal Is The Same. To Create A 5-By-5 Magic Square Matrix, Use The Magic Function As Shown.

```
A = Magic (5)
A =
     17   24    1    8   15
     23    5    7   14   16
      4    6   13   20   22
     10   12   19   21    3
     11   18   25    2    9
```

Note That The Elements Of Each Row, Each Column, And Each Main Diagonal Add Up To TheSame Value: 65.

❖ **Subscripts :**

The element in row i and column j of A is denoted by A(i,j). For example, A(4,2) is the number in the fourth row and second column. For our magic square, A(4,2) is 15. So to compute the sum of the elements in the fourth column of A, type

```
A(1,4) + A(2,4) + A(3,4) + A(4,4)
```

This produces

```
ans =
    34
```

❖ **The Colon Operator:**

The colon, :, is one of the most important MATLAB operators. It occurs in several different forms. The expression **1:10** is a row vector containing the integers from 1 to 10:

```
1    2    3    4    5    6    7    8    9   10
```

To obtain nonunit spacing, specify an increment. For example,
100:-7:50

Produces following output
```
100    93    86    79    72    65    58    51
```

❖ **Operators:**

Expressions use familiar arithmetic operators and precedence rules.

+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Left division (described in Linear Algebra in the MATLAB documentation)
^	Power
'	Complex conjugate transpose
( )	Specify evaluation order

Several special functions provide values of useful constants.

Pi	- 3.14159265...
i	- Imaginary unit,
j	- Same as i
eps	- Floating-point relative precision,
realmin	- Smallest floating-point number,
realmax	- Largest floating-point number,
Inf	- Infinity
NaN	- Not-a-number

 Examples of Expressions:

You have already seen several examples of MATLAB expressions. Here are a few more examples, and the resulting values:

```
>>rho = (1+sqrt(5))/2
rho =
   1.6180
>>a = abs(3+4i)
a =
    5
>>z = sqrt(besselk(4/3,rho-i))
z =
  0.3730+ 0.3214i
>>huge = exp(log(realmax))
huge =
  1.7977e+308
>>toobig = pi*huge
toobig=  Inf
```

MATLAB- software provides four functions that generate basic matrices:

- Zeros - All zeros
- Ones - All ones
- Rand - Uniformly distributed random elements
- Randn -Normally distributed random elements

Here are some examples:

```
Z = zeros(2,4)
Z =
   0    0    0    0
   0    0    0    0
F = 5*ones(3,3)
F =
   5    5    5
   5    5    5
   5    5    5

N = fix(10*rand(1,10))
N =
   9    2    6    4    8    7    4    0    8    4

R = randn(4,4)
R =
  0.6353   0.0860  -0.3210  -1.2316
 -0.6014  -2.0046   1.2366   1.0556
  0.5512  -0.4931  -0.6313  -0.1132
 -1.0998   0.4620  -2.3252   0.3792
```

M-Files:

You can create your own matrices using M-files, which are text files containing MATLAB code. Use the MATLAB Editor or another text editor to create a file containing the same statements you would type at the MATLAB command line. Save the file under a name that ends in .m.

For example, create a file in the current directory named magik.m containing these five lines:

```
A = [16.0    3.0    2.0    13.0
      5.0   10.0   11.0   8.0
      9.0    6.0    7.0    12.0
      4.0   15.0   14.0   1.0 ];
```

The statement **magik** reads the file and creates a variable, A, containing the example matrix.

Conditional Control – if, else, switch

The if statement evaluates a logical expression and executes a group of statements when the expression is true. The optional elseif and else keywords provide for the execution of alternate groups of statements. An end keyword, which matches the if, terminates the last group of

statements. The groups of statements are delineated by the four keywords—no braces or brackets are involved.
The MATLAB algorithm for generating a magic square of order n involves three different cases: when n is odd, when n is even but not divisible by 4, or when n is divisible by 4. This is described by

```
if rem(n,2) ~= 0
  M = odd_magic(n)
elseif rem(n,4) ~= 0
  M = single_even_magic(n)
else
  M = double_even_magic(n)
end
```

The proper way to check for equality between two variables is to use the isequal function:

```
>> if isequal(A, B), ...
```

Here is another example to emphasize this point. If A and B are scalars, the following program will never reach the "unexpected situation". But for most pairs of matrices, including our magic squares with interchanged columns, none of the matrix conditions A > B, A < B, or A == B is true for all elements and so the else clause is executed:

```
if A > B
  'greater'
elseif A < B
  'less'
elseif A == B
  'equal'
else
error('Unexpected situation')
end
```

Loop Control – for, while, continue, break
For:
The for loop repeats a group of statements a fixed, predetermined number of times. A matching end delineates the statements:

```
for n = 3:32
  r(n) = rank(magic(n));
end
r
```

The semicolon terminating the inner statement suppresses repeated printing, and the r after the loop displays the final result.
It is a good idea to indent the loops for readability, especially when they are nested:

```
for i = 1:m
  for j = 1:n
    H(i,j) = 1/(i+j);
  end
end
```

☐ Two Dimensional Plots (Graphs)

- To draw a 2-D plot in MATLAB, the **plot** command is used. The syntax is given as follows…..

  plot (x, y)
  Here, x is a vector containing the x-coordinates
  y is a vector containing the y-coordinates

- To make the plot more understable, the two axes are labelled and a title is provided above the top of the plot using following commands:

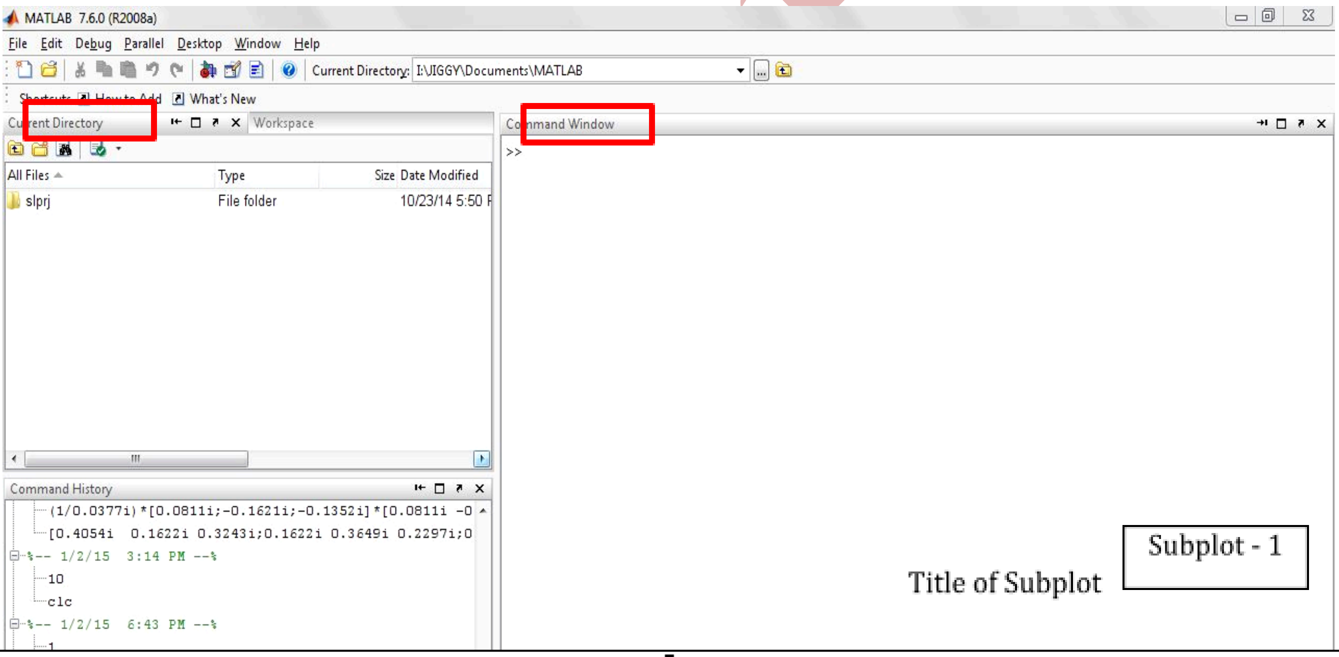
  xlabel ('Title X-Axis')
  ylabel ('Title Y-Axis')
  title ('Title of Graph')

- To give better illustration of the co-ordinates on the plot, a grid is used. The command **grid** is used to show grid lines on the plot. The grid can be added or removed by using **on** and **off** with the **grid** command.

  To make grid visible use **grid on**
  To make grid removed use **grid off**

☐ Matlab default screen:

It has mainly three windows shown in figure 1:
Command Window: Where programme can and command can be implemented directly
Command History : Past used command can be seen and used again
Current Dictionary : You can see your currant dictinory where you are working



```
>>
```

```
(1/0.0377i)*[0.0811i;-0.1621i;-0.1352i]*[0.0811i -0
[0.4054i  0.1622i 0.3243i;0.1622i 0.3649i 0.2297i;0
%-- 1/2/15  3:14 PM --%
10
clc
%-- 1/2/15  6:43 PM --%
1
```

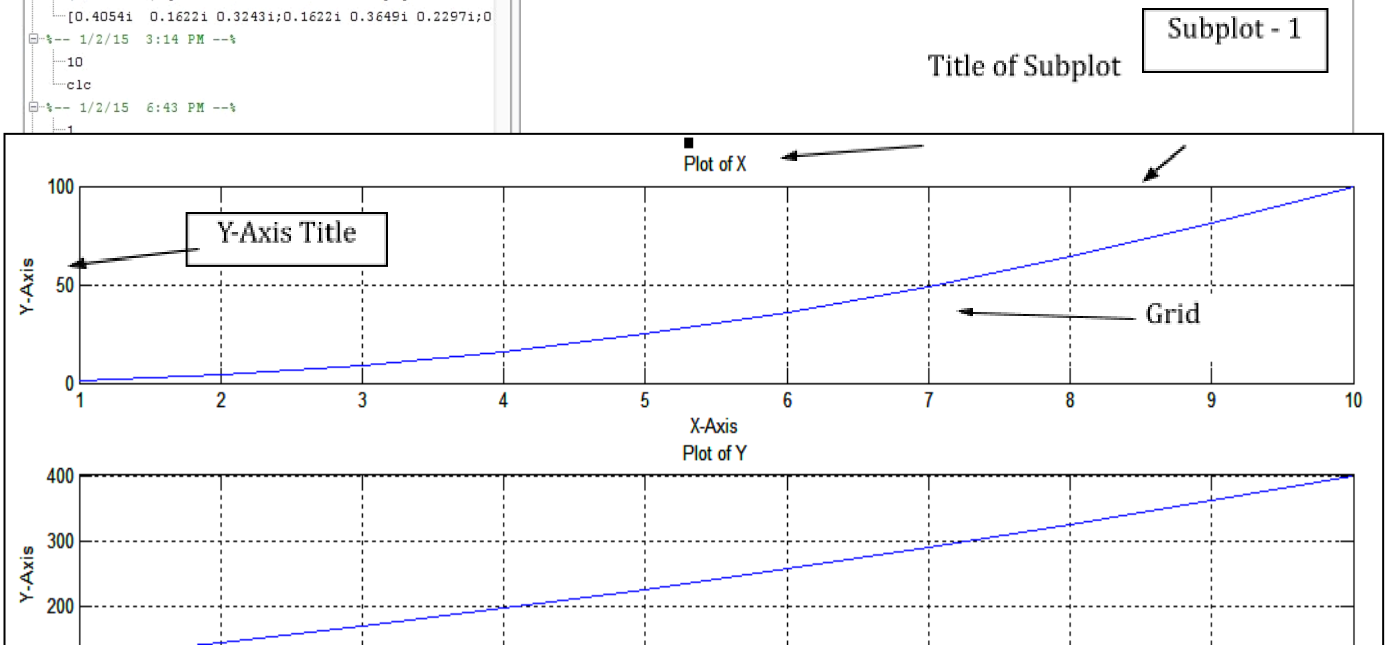


Figure 2 : Two Dimensional Plot Using Subplot

EXERCISE

(1) For

$$A = \begin{matrix} 16 & 3 & 2 \\ 5 & 10 & 11 \\ 9 & 6 & 7 \end{matrix} \quad \& \quad B = \begin{matrix} 5 & 2 & 9 \\ 6 & 7 & 2 \\ 1 & 3 & 54 \end{matrix}$$

Find Following using Matrices A and B

(i) Determinant of A and B
(ii) Inverse of A and B
(iii) Transposed of A and B
(iv) Sum of A and B
(v) Diagonal of A and B
(vi) A*B and A.*B
(vii) A^2-2AB+B^2
(viii) A/B and A./B
(ix) $(A+B)^3$

(2) (i) Make a 3x3 Magic Matrix
(ii) Find square root of 2020.
(iii) Find absolute value and angle of 5-6i.
(iv) Program for $(a+b+c)^2=a^2+b^2+c^2+2ab+2bc+2ca$.

(3) Plot Sinusoidal and Cosine waves for x = 0 to 4*pi values and perform following operations.
- Give Individual titles for Both Graphs
- Give Labels to Axes.
- Make Grid on

❖ **Attach Programme and Result Sheets.**

CONCLUSION:

QUESTIONS:

(1) Write different functions of matlab.
(2) Write all different oprator used in matlab.

SIGNATURE: **MARKS:**

EXPERIMENT NO:2 **DATE:**

AIM: INTRODUCTION TO CONTROL THEORY.

Q -1 Define following terms with respect to Control System.

- System
- Output
- Input
- Control
- Control System
- Transfer Function
- Step Input
- Ramp Input
- Parabolic Input
- Impulse Input
- Pole & Zero
- Pole-Zero Plot
- Characteristic Equation
- Block Diagram
- Signal Flow Graph
- Path Gain
- Forward Path
- Feedback Path
- Path
- Touching Path
- Non-touching Path
- Loop Gain
- Dummy Node
- Stable System
- Unstable System
- Absolutely Stable System
- Critically/Marginally Stable System
- Conditionally Stable System
- Root Locus

Q-2 Define Open Loop & Closed Loop Systems with appropriate Figures& list out examples of these two systems & explain any one example in detail.

Q -3 Define following with respect to Time Response with proper diagram.

- Time Response
- Delay Time

- Transient Response
- Steady State Response
- Damping Ratio
- Rise Time
- Peak Overshoot
- Peak Time

Q -4 Define following with respect to Frequency Response with proper diagram.

- Frequency Response
- Resonance Peak
- Resonance Frequency
- Cut-Off Rate
- Phase Margin
- Cut-Off Frequency
- Gain Crossover
- Phase Margin Angle
- Phase Crossover
- Gain Margin

Signature: **Marks:**

EXPERIMENT NO:3 **DATE:**

AIM: - TO REPRESENT GRAPHICALLY TIME RESPONSE OF 1ST ORDER CONTROL SYSTEM SUBJECTED TO STEP, RAMP, PARABOLIC, IMPULSE INPUT FUNCTION FOR DIFFERENT TIME PERIOD USING MATLAB.

THEORY:

STEP FUNCTION

Let us take an independent voltage source or a battery which is connected across a voltmeter via a switch, s. It is clear from the figure below, whenever the switch s is open, the voltage appears between the voltmeter terminals is zero. If the voltage between the voltmeter terminals is represented as v (t), the situation can be mathematically represented as

$$v(t) = 0 \;\; when \quad -\infty < t < 0$$

Now let us consider at t = 0, the switch is closed and instantly the battery voltage V volt appears across the voltmeter and that situation can be represented as,

$$v(t) = k \;\; when \;\; 0 < t < \infty$$

Combining the above two equations we get

$$\begin{aligned} v(t) &= 0 \;\; when \quad -\infty < t < 0 \\ \& \qquad &= k \;\; when \qquad 0 < t < \infty \end{aligned}$$

In the above equations if we put 1 in place of V, we will get a unit step function which can be defined as

$$\begin{aligned} u(t) &= 0 \ \ when \ \ t \leq 0 \\ \& \qquad &= 1 \ \ when \ \ t \geq 0 \end{aligned}$$

Now let us examine the Laplace transform of unit step function. Laplace transform of any function can be obtained by multiplying this function by e^{-st} and integrating multiplied from 0 to infinity.

$$\pounds u(t) = \int_0^\infty u(t)e^{-st}dt = \int_0^\infty 1.e^{-st}dt = \left[\frac{e^{-st}}{-s}\right]_0^\infty = \frac{1}{s}$$

If input is R(s), then

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

RAMP FUNCTION

The function which is represented by an inclined straight line intersecting the origin is known as ramp function. That means this function starts from zero and increases or decreases linearly with time. A ramp function can be represented as,

$$\begin{aligned} r(t) &= 0 \quad when \ t < 0 \\ \& \qquad &= kt \quad when \ t > 0 \end{aligned}$$

Here in this above equation, k is the slope of the line.Now let us examine the Laplace transform of ramp function. As we told earlier Laplace transform of any function can be obtained by multiplying this function by e^{-st} and integrating multiplied from 0 to infinity.

$$\pounds r(t) = \int_0^\infty r(t)e^{-st}dt = \int_0^\infty kt.e^{-st}dt = \frac{k}{s^2}$$

$$R(S) = \frac{k}{s^2}$$

PARABOLIC FUNCTION

Here the value of function is zero when time t<0 and is quadratic when time t>0. A parabolic function can be defined as,

$$\begin{aligned} & p(t) = 0 \quad when\ t < 0 \\ \& \quad & = \frac{kt^2}{2} \quad when\ t > 0 \end{aligned}$$

Now let us examine the Laplace transform of parabolic function. As we told earlier Laplace transform of any function can be obtained by multiplying this function by e^{-st} and integrating multiplied from 0 to infinity.

$$\pounds p(t) = \int_0^\infty p(t)e^{-st}\,dt = \int_0^\infty \frac{kt^2}{2}.e^{-st}dt = \frac{k}{s^3}$$

$$R(S) = \frac{k}{s^3}$$

IMPULSE FUNCTION

Impulse signal is produced when input is suddenly applied to the system for infinitesimal duration of time. The waveform of such signal is represented as impulse function. If the magnitude of such function is unity, then the function is called unit impulse function. The first time derivative of step function is impulse function. Hence Laplace transform of unit impulse function is nothing but Laplace transform of first-time derivative of unit step function.

$$\pounds\left(Unit\ impulse\ function\right) = \pounds\frac{d}{dt}\left(Unit\ step\ function\right)$$
$$= s\pounds\left(Unit\ step\ function\right) = s.\frac{1}{s} = 1$$

EXAMPLES:

1) FOR STEP INPUT:

1. $\frac{1}{s+1}$

2. $\frac{1}{10s+1}$

2) FOR RAMP INPUT:

1. $\frac{1}{s^2+s}$

2. $\frac{1}{10s^2+s}$

3) FOR PARABOLIC INPUT:

1. $\frac{1}{s^3+s^2}$

2. $\frac{1}{10s^3+s^2}$

4) FOR IMPULSE INPUT:

1. impulse response with t=1

2. impulse response with t=10

3. impulse response with t=100

❖ **ATTACH GRAPH**

CONCLUSION:

Signature: **Marks:**

EXPERIMENT NO:4 **DATE:**

AIM: FIND TRANSFER FUNCTION AND IT'S POLE AND ZERO BY USING MATLAB.

THEORY:

DEFINITION OF TRANSFER FUNCTION

The transfer function of a control system is defined as the ration of the Laplace transform of the output variable to Laplace transform of the input variable assuming all initial conditions to be zero.

Procedure for determining the transfer function of a control system are as follows

1. We form the equations for the system

2. Now we take Laplace transform of the system equations, assuming initial conditions as zero.
3. Specify system output and input
4. Lastly we take the ratio of the Laplace transform of the output and the Laplace transform of the input which is the required transfer function

POLES AND ZEROS OF TRANSFER FUNCTION

Generally a function can be represented to its polynomial form. For example, $F(s) = f_0 s^n + f_1 s^{n-1} + f_2 s^{n-2} + f_3 s^{n-3} + \cdots\cdots + f_{n-1} s^1 + f_n$

Now similarly transfer function of a control system can also be represented as

$$G(s) = \frac{C(s)}{R(s)} = \frac{C_0 s^n + C_1 s^{n-1} + C_2 s^{n-2} + \cdots\cdots + C_{n-1} s + C_n}{R_0 s^m + R_1 s^{m-1} + R_2 s^{m-2} + \cdots + R_{m-1} s + R_m}$$
$$= K \frac{(s-z_1)(s-z_2)(s-z_3)\cdots(s-z_n)}{(s-p_1)(s-p_2)(s-p_3)\cdots(s-p_m)}$$

Where, K is known as gain factor of the transfer function. Now in the above function if s = z_1, or s = z_2, or s = z_3,…, s = z_n,the value of transfer function becomes zero. These z_1, z_2, z_3,…,z_n, are roots of the numerator polynomial. As for these roots the numerator polynomial, the transfer function becomes zero, these roots are called zeros of the transfer function. Now, if s = p_1, or s = p_2, or s = p_3,....s = p_m, the value of transfer function becomes infinite. Thus the roots of denominator are called the poles of the function.

Now let us rewrite the transfer function in its polynomial form.

$$G(s) = K \frac{(s-z_1)(s-z_2)(s-z_3)\cdots(s-z_n)}{(s-p_1)(s-p_2)(s-p_3)\cdots(s-p_m)}$$

EXAMPLES:

1) $\frac{(s+1)}{s\,(s+2)}$ **2)** $\frac{s}{(s-2)}$

3) $\frac{12\,(s+3)}{(s+1)\,(s-1)}$ **4)** $\frac{(s+2)\,(s+3)}{s\,(s-2)}$

5) $\frac{(s-3)}{(s-1)^2}$ **6)** $\frac{12\,(s+2.653)}{s\,(s+1)\,(s+7.813)}$

7) $\frac{25\,(s+1)}{(s+5)}$ **8)** $\frac{(s+0.1)}{(s+10)}$

- ❖ **ATTACH PROGRAM SHEET**
- ❖ **ATTACH RESULT SHEET**

CONCLUSION:

Signature: **Marks:**

EXPERIMENT NO:5 **DATE:**

AIM: TO OBTAIN THE POLES AND ZEROS OF A SYSTEM, GIVEN THE TRANSFER FUNCTION, AND PLOT THEM ON REAL & IMAGINARY AXIS USING MATLAB.

THEORY:

DEFINITION OF TRANSFER FUNCTION

The transfer function of a control system is defined as the ration of the Laplace transform of the output variable to Laplace transform of the input variable assuming all initial conditions to be zero.

Procedure for determining the transfer function of a control system are as follows

1. We form the equations for the system
2. Now we take Laplace transform of the system equations, assuming initial conditions as zero.
3. Specify system output and input
4. Lastly we take the ratio of the Laplace transform of the output and the Laplace transform of the input which is the required transfer function

POLES AND ZEROS OF TRANSFER FUNCTION

Generally a function can be represented to its polynomial form. For example, $F(s) = f_0 s^n + f_1 s^{n-1} + f_2 s^{n-2} + f_3 s^{n-3} + \cdots\cdots + f_{n-1} s^1 + f_n$

Now similarly transfer function of a control system can also be represented as

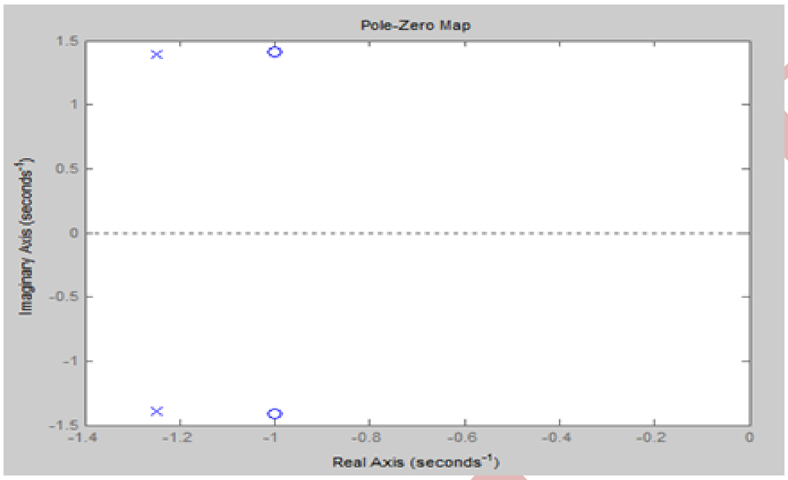
$$G(s) = \frac{C(s)}{R(s)} = \frac{C_0 s^n + C_1 s^{n-1} + C_2 s^{n-2} + \cdots\cdots + C_{n-1} s + C_n}{R_0 s^m + R_1 s^{m-1} + R_2 s^{m-2} + \cdots + R_{m-1} s + R_m}$$
$$= K \frac{(s - z_1)(s - z_2)(s - z_3) \cdots (s - z_n)}{(s - p_1)(s - p_2)(s - p_3) \cdots (s - p_m)}$$

Where, K is known as gain factor of the transfer function. Now in the above function if s = z_1, or s = z_2, or s = z_3,…, s = z_n,the value of transfer function becomes zero. These z_1, z_2, z_3,…,z_n, are roots of the numerator polynomial. As for these roots the numerator polynomial, the transfer function becomes zero, these roots are called zeros of the transfer function. Now, if s = p_1, or s

= p_2, or s = p_3,....s = p_m, the value of transfer function becomes infinite. Thus the roots of denominator are called the poles of the function.

Now let us rewrite the transfer function in its polynomial form.

$$G(s) = K\frac{(s-z_1)(s-z_2)(s-z_3)\cdots(s-z_n)}{(s-p_1)(s-p_2)(s-p_3)\cdots(s-p_m)}$$



EXAMPLES:

1) $\frac{s2+2s+3}{2s2+5s+7}$ **2)** $\frac{7s2-7}{4s2+4s+1}$

- **ATTACH PROGRAM SHEET**
- **ATTACH RESULT SHEET**

CONCLUSION:

Signature: **Marks:**

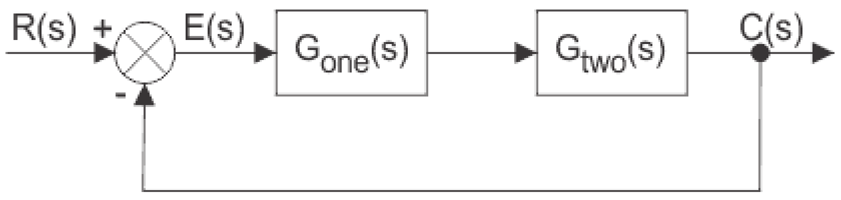
EXPERIMENT NO:6 **DATE:**

AIM: FIND OUT OVERALL TRANSFER FUNCTION USING BLOCK DIAGRAM REDUCTION TECHNIQUE IN MATLAB.

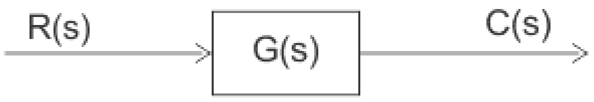
THEORY:

BLOCK DIAGRAMS OF CONTROL SYSTEM

Under Control Systemthe block diagram is to represent a control system in diagram form. In other words practical representation of a control system is its block diagram. It is not always convenient to derive the entire transfer function of a complex control system in a single function. It is easier and better to derive transfer function of control element connected to the system, separately. The transfer function of each element is then represented by a block and they are then connected together with the path of signal flow. For simplifying a complex control system, block diagrams are used. Each element of the control system is represented with a block and the block is the symbolic representation of transfer function of that element. A complete control system can be represented with a required number of interconnected such blocks. In the figure below, there are two elements with transfer function $G_{one}(s)$ and $G_{two}(s)$. Where $G_{one}(s)$ is the transfer function of first element and $G_{two}(s)$ is the transfer function of second element of the system. In addition to that, the diagram also shows there is a feedback path through which output signal C(s) is fed back and compared with the input R(s) and the difference between input and output E(s) = R(s) – C(s) is acting as actuating signal or error signal.

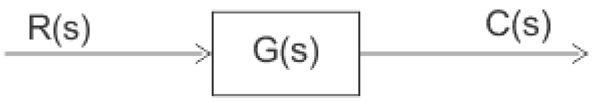


In each block of diagram, the output and input are related together by transfer function. Where, transfer function $G(s) = \frac{C(s)}{R(s)}$ where, C(s) is the output and R(s) is the input of that particular block.



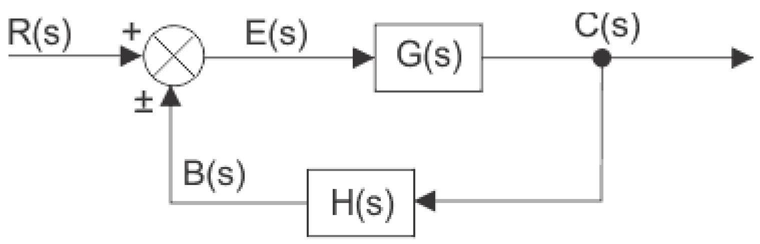
A complex control system consists of several blocks. Each of them has its own transfer function. But overall transfer function of the system is the ratio of transfer function of final output to transfer function of initial input of the system. This overall transfer function of the system can be obtained by simplifying the control system by combining this individual blocks, one by one. Technique of combining of these blocks is referred as **block diagram reduction technique**. For successful implementation of this technique, some rules for block diagram reduction to be followed. Let us discuss these rules, one by one for reduction of block diagram of control system.

If the transfer function of input of control system is R(s) and corresponding output is C(s), and the overall transfer function of the control system is G(s), then the control system can be represented as



$$Where,\ G(s) = \frac{C(s)}{R(s)}$$

BLOCK DIAGRAM OF CLOSED LOOP CONTROL SYSTEM



In a closed loop control system, a fraction of output is fed-back and added to input of the system. If H (s) is the transfer function of feedback path, then the transfer function of feedback signal will be B(s) = C(s)H(s). At summing point, the input signal R(s) will be added to B(s) and produces actual input signal or error signal of the system and it is denoted by E(s).

$$E(s) = R(s) \pm B(s) = R(s) \pm C(s)H(s)$$

$$Now,\ G(s) = \frac{C(s)}{E(s)} = \frac{C(s)}{R(s) \pm C(s)H(s)} [\because E(s) = R(s) \pm C(s)H(s)]$$

$$Now,\ overall\ transfer\ function\ of\ the\ system\ is$$

$$G'(s) = \frac{C(s)}{R(s)} = \frac{G(s)E(s)}{R(s)} [\because\ from\ above\ equation\ C(s) = G(s)E(s)]$$

$$\Rightarrow G'(s) = \frac{G(s)\,[R(s) \pm C(s)H(s)]}{R(s)} [\because\ from\ above\ equation\ E(s) = R(s) \pm C(s)H(s)]$$

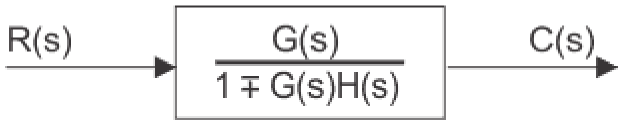
$$\Rightarrow G'(s) = G(s)\left[1 \pm \frac{C(s)H(s)}{R(s)}\right]$$

$$\Rightarrow G'(s) = G(s)\left[1 \pm G'(s)H(s)\right] \left[\because G'(s) = \frac{C(s)}{R(s)}\right]$$

$$\Rightarrow G'(s) = G(s) \pm G(s)G'(s)H(s)$$

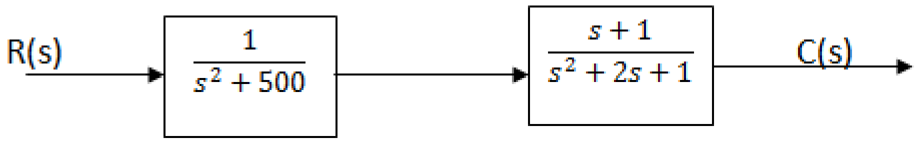
$$\Rightarrow G'(s)\left[1 \mp G(s)H(s)\right] = G(s)$$

$$\Rightarrow G'(s) = \frac{G(s)}{1 \mp G(s)H(s)}$$

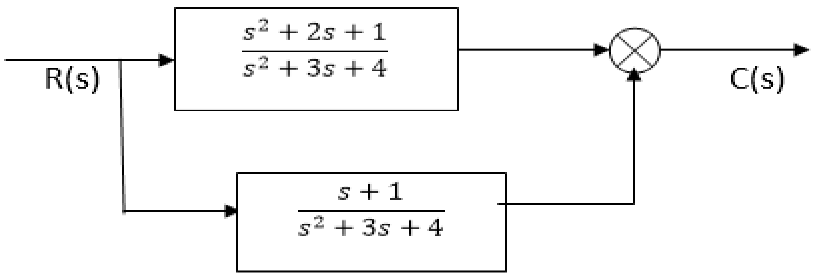


EXAMPLES:

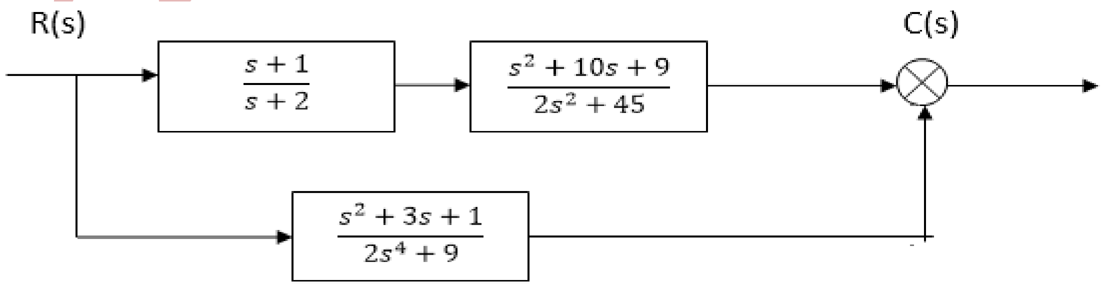
1.

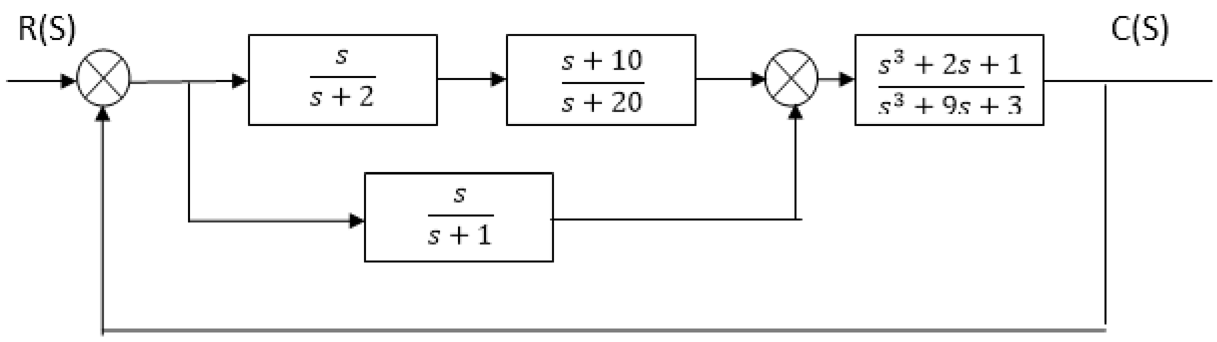


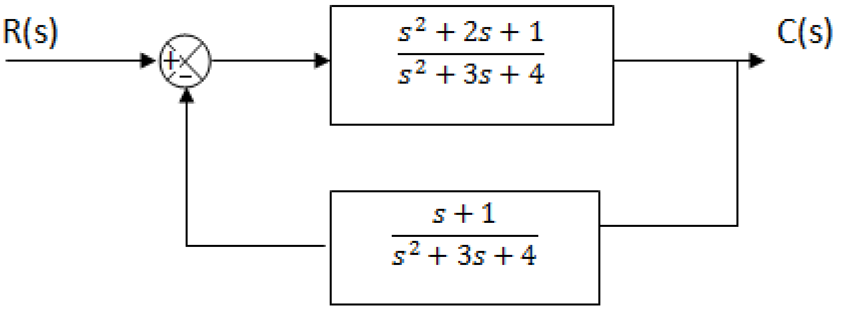
2.



3.



4. 

5. 

❖ **ATTACH RESULT SHEET**

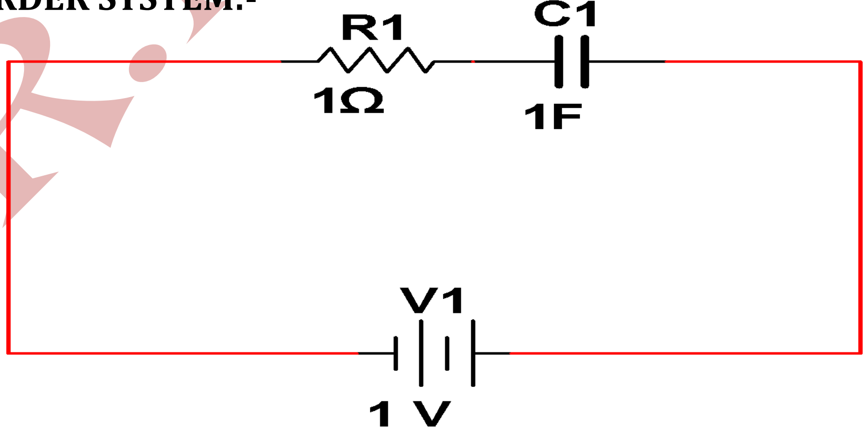
CONCLUSION:

Signature: **Marks:**

EXPERIMENT NO:7 **DATE:**

AIM: PLOTTING THE RESPONSE OF FIRST AND SECOND ORDER CIRCUIT WITH THE HELP OF SIMULINK.

FIRST ORDER SYSTEM:-



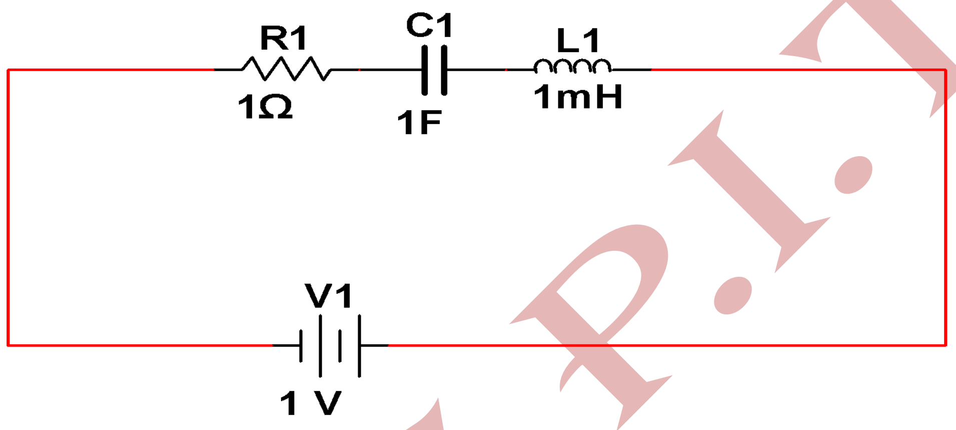
Applying Kirchhoff's law of voltage we get the equation

Equation:

$$V= Ir + \frac{1}{c}\int idt$$

$$V=\frac{1}{r}\int (v - vc)dt$$

SECOND ORDER SYSTEM:-



Applying Kirchhoff's law of voltage

Equation:

$$\frac{dvc}{dt} = \frac{iR}{c} - i\frac{vc}{Rc} - \frac{1}{Lc}\int vc\, dt$$

$$vc = \int\frac{iR}{c} - i\int\frac{vc}{r} - \frac{1}{Lc}\int vc\, dt$$

- ❖ **ATTACH SIMULATION SHEET**
- ❖ **ATTACH GRAPH**

CONCLUSION:

Signature: **Marks:**

EXPERIMENT NO:8 **DATE:**

AIM: - TO PLOT THE ROOT LOCUS FOR THE GIVEN SET OF SYSTEMS USING MATLAB.

THEORY:

The **root locus technique in control system** was first introduced in the year 1948 by Evans. Any physical system is represented by a transfer function in the form of $G(s) = k \times \frac{numerator\ of\ s}{denomerator\ of\ s}$ We can find poles and zeros from G(s). The location of poles and zeros are crucial keeping view stability, relative stability, transient response and error analysis. When the system put to service stray inductance and capacitance get into the system, thus changes the location of poles and zeros. In **root locus technique in control system** we will evaluate the position of the roots, their locus of movement and associated information. These information will be used to comment upon the system performance.

Now before I introduce what is a root locus technique, it is very essential here to discuss a few of the advantages of this technique over other stability criteria. Some of the advantages of root locus technique are written below.

ADVANTAGES OF ROOT LOCUS TECHNIQUE

1. Root locus technique in control system is easy to implement as compared to other methods.
2. With the help of root locus we can easily predict the performance of the whole system.
3. Root locus provides the better way to indicate the parameters.

Now there are various terms related to root locus technique that we will use frequently in this article.

1. Characteristic Equation Related to Root Locus Technique: 1 + G(s) H(s) = 0 is known as characteristic equation. Now on differentiating the characteristic equation and on equating dk/ds equals to zero, we can get break away points.

2. Break away Points: Suppose two root loci which start from pole and moves in opposite direction collide with each other such that after collision they start moving in different directions in the symmetrical way. Or the breakaway points at which multiple roots of the characteristic equation 1 + G(s) H(s) = 0 occur. The value of K is maximum at the points where the branches of root loci break away. Break away points may be real, imaginary or complex.
3. Break in Point: Condition of break in to be there on the plot is written below: Root locus must be present between two adjacent zeros on the real axis.
4. Centre of Gravity: It is also known centroid and is defined as the point on the plot from where all the asymptotes start. Mathematically, it is calculated by the difference of summation of poles and zeros in the transfer function when divided by the difference of total number of poles and total number of zeros. Centre of gravity is always real & it is denoted by σ_A.

$$\sigma_A = \frac{(Sum\ of\ real\ parts\ of\ poles) - (Sum\ of\ real\ parts\ of\ zeros)}{N - M}$$

   Where N is number of poles & M is number of zeros.

5. Asymptotes of Root Loci: Asymptote originates from the centre of gravity or centroid and goes to infinity at definite some angle. Asymptotes provide direction to the root locus when they depart break away points.
6. Angle of Asymptotes : Asymptotes makes some angle with the real axis and this angle can be calculated from the given formula,

$$Angle\ of\ asymptotes = \frac{(2p+1) \times 180}{N-M}$$

Where p = 0, 1, 2....... (N-M-1) N is the total number of poles M is the total number of zeros.

7. Angle of Arrival or Departure: We calculate angle of departure when there exists complex poles in the system. Angle of departure can be calculated as 180-{(sum of angles to a complex pole from the other poles)-(sum of angle to a complex pole from the zeros)}.
8. Intersection of Root Locus with the Imaginary Axis: In order to find out the point of intersection root locus with imaginary axis, we have to use Routh Hurwitz criterion. First, we find the auxiliary equation then the corresponding value of K will give the value of the point of intersection.
9. Gain Margin: We define gain margin as a by which the design value of the gain factor can be multiplied before the system becomes unstable. Mathematically it is given by the formula

$$Gain\ margin = \frac{Value\ of\ K\ at\ the\ imaginary\ axes\ cross\ over}{Design\ value\ of\ K}$$

10. Phase Margin : Phase margin can be calculated from the given formula:

$$Phase\ margin = 180 + \angle(G(jw)H(jw))$$

11. Symmetry of Root Locus: Root locus is symmetric about the x axis or the real axis.

How to determine the value of K at any point on the root loci? Now there are two ways of determining the value of K, each way is described below.

1. Magnitude Criteria : At any points on the root locus we can apply magnitude criteria as,

   $$|G(s)H(s)| = 1$$

   Using this formula we can calculate the value of K at any desired point.

2. Using Root Locus Plot : The value of K at any s on the root locus is given by

$$K = \frac{product\ of\ all\ of\ the\ vector\ lengths\ drawn\ from\ the\ poles\ of\ G(s)H(s)\ to\ s}{product\ of\ all\ of\ the\ vector\ lengths\ drawn\ from\ the\ zeros\ of\ G(s)H(s)\ to\ s}$$

EXAMPLES:

(1) To plot the root locus for

$$G(s).H(s) = \frac{k(s+2)}{3s2+5s+7}$$

(2) To plot the root locus for

$$G(s).H(s) = \frac{k(s+3)}{s(s+2)(s+4)(s+5)}$$

(3) To plot the root locus for

$$G(s).H(s) = \frac{(2s+9)\,k}{s\,(s2+6s+12)}$$

And gain k=10 to 50

(4) To plot the root locus for

$$G(s).H(s) = \frac{k(s+1)}{s(s2+5s+3)}$$

For ɀ=0.5 ,ω_n=1.5

❖ **ATTACH GRAPH**

CONCLUSION:

Signature: **Marks:**

EXPERIMENT NO:9 **DATE:**

AIM: - TO OBTAIN THE BODE-PLOTS FOR THE GIVEN SYSTEMS BY USING MATLAB.

THEORY:

BODE PLOT:

These are also known as logarithmic plot (because we draw these plots on semi-log papers) and are used for determining the relative stabilities of the given system. Now in order to determine the stability of the system using bode plot we draw two curves, one is for magnitude called magnitude curve another for phase called **Bode phase plot**.

Now there are some results that one should remember in order to plot the Bode curve. These results are written below:

- **Constant term K:** This factor has a slope of zero dB per decade. There is no corner frequency corresponding to this constant term. The phase angle associated with this constant term is also zero.
- **Integral factor $1/(j\omega)^n$:** This factor has a slope of -20 × n (where n is any integer)dB per decade. There is no corner frequency corresponding to this integral factor. The phase angle associated with this integral factor is -90 × n here n is also an integer.
- **First order factor 1/ (1+jωT):** This factor has a slope of -20 dB per decade. The corner frequency corresponding to this factor is 1/T radian per second. The phase angle associated with this first factor is $-\tan^{-1}(\omega T)$.
- **First order factor (1+jωT):** This factor has a slope of 20 dB per decade. The corner frequency corresponding to this factor is 1/T

radian per second. The phase angle associated with this first factor is $\tan^{-1}(\omega T)$.

- ❖ **Second order or quadratic factor : $[\{1/(1+(2\zeta/\omega)\} \times (j\omega) + \{(1/\omega^2)\} \times (j\omega)^2)]$:** This factor has a slope of -40 dB per decade. The corner frequency corresponding to this factor is ω^n radian per second. The phase angle associated with this first factor is $-\tan^{-1}\{(2\zeta\omega / \omega_n) / (1-(\omega / \omega_n)^2)\}$.

Keeping all these points in mind we are able to draw the plot for any kind of system. Now let us discuss the procedure of making a bode plot:

1. Substitute the s = jω in the open loop transfer function G(s) × H(s).
2. Find the corresponding corner frequencies and tabulate them.
3. Now we are required one semi-log graph chooses a frequency range such that the plot should start with the frequency which is lower than the lowest corner frequency. Mark angular frequencies on the x-axis, mark slopes on the left hand side of the y-axis by marking a zero slope in the middle and on the right hand side mark phase angle by taking -180 degrees in the middle.
4. Calculate the gain factor and the type or order of the system.
5. Now calculate slope corresponding to each factor.

FOR DRAWING THE MAGNITUDE CURVE:

(a) Mark the corner frequency on the semi log graph paper.

(b)Tabulate these factors moving from top to bottom in the given sequence.

1. Constant term K.
2. Integral factor $1/(j\omega)^n$.

3. First order factor $1/(1+j\omega T)$.
4. First order factor $(1+j\omega T)$.
5. Second order or quadratic factor : $[\{1/(1+(2\zeta/\omega)\} \times (j\omega) + \{(1/\omega^2)\} \times (j\omega)^2)]$

(c) Now sketch the line with the help of corresponding slope of the given factor. Change the slope at every corner frequency by adding the slope of the next factor. You will get magnitude plot.

(d) Calculate the gain margin.

FOR DRAWING THEBODE PHASE PLOT:

1. Calculate the phase function adding all the phases of factors.
2. Substitute various values to above function in order to find out the phase at different points and plot a curve. You will get a phase curve.
3. Calculate the phase margin.

STABILITY CONDITIONS OF BODE PLOTS:

Stability conditions are given below:

1. **For Stable System:** Both the margins should be positive. Or phase margin should be greater than the gain margin.
2. **For Marginal Stable System:** Both the margins should be zero. Or phase margin should be equal to the gain margin.
3. **For Unstable System:** If any of them is negative. Or phase margin should be less than the gain margin.

Now there are various terms related to this plot that we will use frequently in this article.

1. **Gain Margin:** Greater will the **gain margin** greater will be the stability of the system. It refers to the amount of gain, which can be increased or decreased without making the system unstable. It is usually expressed in dB.
2. **Phase Margin:** Greater will the **phase margin** greater will be the stability of the system. It refers to the phase which can be increased or decreased without making the system unstable. It is usually expressed in phase.
3. **Gain Cross Over Frequency:** It refers to the frequency at which magnitude curve cuts the zero dB axis in the bode plot.
4. **Phase Cross Over Frequency:** It refers to the frequency at which phase curve cuts the negative times the 180 degree axis in this plot.
5. **Corner Frequency:** The frequency at which the two asymptotes cuts or meet each other is known as break frequency or corner frequency.
6. **Resonant Frequency:** The value of frequency at which the modulus of G (jω) has a peak value is known as resonant frequency.
7. **Factors:** Every loop transfer function (i.e. G(s) × H(s)) product of various factors like constant term K, Integral factors (jω), first order factors $(1 + j\omega T)^{(\pm n)}$ where n is an integer, second order or quadratic factors.
8. **Slope:** There is a slope corresponding to each factor and slope for each factor is expressed in the dB per decade.
9. **Angle:** There is an angle corresponding to each factor and angle for each factor is expressed in the degrees.

EXAMPLES:

1. $G(s).H(s) = \frac{1}{s\,(s+3)\,(s+1)}$

2. $G(s).H(s) = \frac{10}{s\,(s+1)\,(s+10)}$

3. $G(s).H(s) = \frac{10}{s\,(s+1)\,(s+5)}$

4. $G(s).H(s) = \frac{1000}{(\,0.1s+1)\,(0.001s+1)}$

5. $G(s).H(s) = \frac{100\,(0.1s+10)}{s\,(0.001s+1)}$

❖ **ATTACH GRAPH**

CONCLUSION:

Signature: **Marks:**

EXPERIMENT NO:10 **DATE:**

AIM: - TO PLOT THE NYQUIST PLOT FOR THE GIVEN SET OF SYSTEMS USING MATLAB.

THEORY:

STEPS OF DRAWING THE NYQUIST PATH

• Step 1 - Check for the poles of G(s) H(s) of jω axis including that at origin.

• Step 2 – Select the proper **Nyquist contour** – a) Include the entire right half of s-plane by drawing a semicircle of radius R with R tends to infinity.

• Step 3 – Identify the various segments on the contour with reference to **Nyquist path**

• Step 4 – Perform the mapping segment by segment substituting the equation for respective segment in the mapping function. Basically we have to sketch the polar plots of the respective segment.

• Step 5 - Mapping of the segments are usually mirror images of mapping of respective path of +ve imaginary axis.

• Step 6 - The semi-circular path which covers the right half of s plane generally maps into a point in G(s) H(s) plane.

• Step 7- Interconnect all the mapping of different segments to yield the required **Nyquist diagram**

• Step 8 – Note the number of clockwise encirclement about (-1, 0) and decide stability by N = Z – P

$$G(s)H(s) = \frac{N(s)}{D(s)}$$

$$\frac{G(s)}{1 + G(s)H(s)}$$

N(s) = 0 is the open loop zero and D(s) is the open loop pole

From stability point of view no closed loop poles should lie in the RH side of s-plane. Characteristics equation 1 + G(s) H(s) = 0 denotes closed loop poles.

$$Let,\ q(s) = 1 + G(s)H(s) = \frac{N_1(s)}{D_1(s)}$$

$$If,\ q(s) = 0,\ \frac{N_1(s)}{D_1(s)} = 0\ i.e.\ N_1(s) = 0$$

Therefore, from the stability point of view zeroes of q(s) should not lie in RHP of s-plane. To define the stability entire RHP (Right Hand Plane) is considered. We assume a semicircle which encloses all points in the RHP by considering the radius of the semicircle R tends to infinity. [R → ∞]. The first step to understand the application of **Nyquist criterion** in relation to determination of stability of control systems is mapping from s-plane to G(s) H(s) - plane. s is considered as independent complex variable and corresponding value of G(s) H(s) being the dependent variable plotted in another complex plane called G(s) H(s) - plane. Thus for every point in s-plane there exists a corresponding point in G(s) H(s) - plane. During the process of mapping the independent variable s is varied along a specified path in s - plane and the corresponding points in G(s)H(s) plane are joined. This completes the process of mapping from s-plane to G(s)H(s) - plane.

Nyquist stability criterion says that N = Z - P where N is the total no. of encirclement about the origin, P is the total no. of poles and Z is the total no. of zeroes.

Case 1:- N = 0 (no encirclement), so Z = P = 0 & Z = P If N = 0, P must be zero therefore system is stable.

Case 2:- N > 0 (clockwise encirclement), so P = 0, Z ≠0 & Z > P For both cases system is unstable.

Case 3 :- N < 0 (counterclockwise encirclement), so Z = 0, P ≠0 & P > Z System is stable.

EXAMPLES:

1) Plot the Nyquist response for the given system

$$H(s) = \frac{2S^2+5s+1}{S^2+2s+3}$$

2) Plot the Nyquist response for the given system

$$G(s) = \frac{1}{S^2+0.8s+1}$$

3) Plot the Nyquist response for the given system

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

- **ATTACH GRAPH**

CONCLUSION:

Signature: **Marks:**