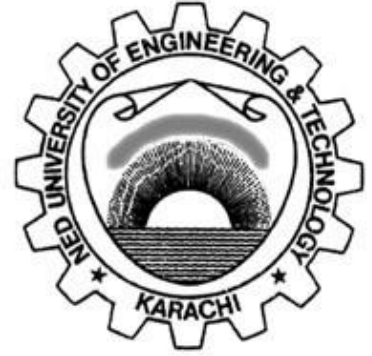




NED University of Engineering & Technology
Department of Electrical Engineering

LAB MANUAL
For the course

FEEDBACK CONTROL SYSTEMS
(EE-374) For T.E.(EL), T.E.(TC)

Instructor name: _____

Student name: _____

Roll no: _____ **Batch:** _____

Semester: _____ **Year:** _____

LAB MANUAL

For the course

FEEDBACK CONTROL SYSTEMS **(EE-374) For T.E.(EL), T.E.(TC)**

Content Revision Team:

Mr. Iqbal Azeem, Mr. Hafiz M. Furqan, Ms. Uzma Parveen

Last Revision Date:

Approved By

The Board of Studies of Department of Electrical Engineering

CONTENTS

S.No.	Date	Title of Experiment	Page No.	Remarks
1		Introduction to MATLAB briefly including tutorial of polynomials, script writing and programming aspect of MATLAB from control systems view point.		
2		Mathematical models of physical systems in the design and analysis of control systems.		
3		Mathematical modeling of Multiple-Element Mechanical Translation System		
4		Mathematical modeling of Electrical System		
5		Mathematical modeling of Electrical System by Simulink		
6		Develop a linear model for a DC motor		
7		Performance of First order and Second order systems and development of Time response specifications function		
8		Response of Control System to Ramp and Arbitrary Inputs.		
9		To learn commands in MATLAB that would be used to reduce linear systems block diagram using series, parallel and feedback configuration.		
10		Study the three term (PID) controller and its effects on the feedback loop response. Also investigate the characteristics of the each of proportional (P), the integral (I), and the derivative (D) controls and obtaining a desired response by using them.		
11		Open Ended Lab – I		
12		Open Ended Lab – II		

LAB SESSION 01

OBJECT:

Introduction to MATLAB briefly including tutorial of polynomials, script writing and programming aspect of MATLAB from control systems view point.

THEORY:

MATLAB Stands for **MA**Tri**x** **LAB**oratory.

MATLAB is a computer program that combines computation and visualization power that makes it particularly useful tool for engineers. It is an executive program, and a script can be made with a list of MATLAB commands like other programming language. The windows in MATLAB are:

Command window: Commands can be entered ,data and results are displayed

Workspace: list all the variables you are using

command history window: it displays a log of the command used.

Graphic (Figure) Window: Displays plots and graphs, created in response to graphics commands.

M-file editor/debugger window: Create and edit scripts of commands called M-files.

Variable declaration:

The variables are declared as:

Must start with a letter

May contain only letters, digits, and the underscore “_”

Matlab is case sensitive, i.e. one & OnE are different variables.

For assigning statement:

Variable = number;

Special variables:

ans : default variable name for the result

pi: $\pi = 3.1415926$

NaN or **nan**: not-a-number

Commands involving variables:

who: lists the names of defined variables

whos: lists the names and sizes of defined variables

clear: clears all variables, reset the default values of special variables.

clear name: clears the variable name

clc: clears the command window

clf: clears the current figure and the graph window

A **Matrix array** is two-dimensional, having both multiple rows and multiple columns, similar to vector arrays:

It begins with [, and end with]

spaces or commas are used to separate elements in a row.

semicolon or enter is used to separate rows.

Example:

```
>> f = [ 1 2 3; 4 5 6]
```

```
f =
```

```
1 2 3
```

```
4 5 6
```

Transpose	$B = A'$
Identity Matrix	$\text{eye}(n) \rightarrow$ returns an $n \times n$ identity matrix $\text{eye}(m,n) \rightarrow$ returns an $m \times n$ matrix with ones on the main diagonal and zeros elsewhere.
Addition and subtraction	$C = A + B$ $C = A - B$
Scalar Multiplication	$B = \alpha A$, where α is a scalar.
Matrix Multiplication	$C = A*B$
Matrix Inverse	$B = \text{inv}(A)$, A must be a square matrix in this case. $\text{rank}(A) \rightarrow$ returns the rank of the matrix A .
Matrix Powers	$B = A.^2 \rightarrow$ squares each element in the matrix $C = A * A \rightarrow$ computes $A*A$, and A must be a square matrix.
Determinant	$\det(A)$, and A must be a square matrix.

A, B, C are matrices, and m, n, α are scalars.

A system of 3 linear equations with 3 unknowns (x_1, x_2, x_3):

$$3x_1 + 2x_2 - x_3 = 10$$

$$-x_1 + 3x_2 + 2x_3 = 5$$

$$x_1 - x_2 - x_3 = -1$$

Let

$$A = \begin{bmatrix} 3 & 2 & 1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$b = \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

■ **Solution by Matrix Inverse:**

$$Ax = b$$

$$A^{-1}Ax = A^{-1}b$$

$$x = A^{-1}b$$

■ **MATLAB:**

```
>> A = [ 3 2 -1; -1 3 2; 1 -1 -1];
```

```
>> b = [ 10; 5; -1];
```

```
>> x = inv(A)*b
```

```
x =
```

```
-2.0000
```

```
5.0000
```

```
-6.0000
```

Answer:

$x_1 = -2, x_2 = 5, x_3 = -6$

■ **Solution by Matrix Division:**

The solution to the equation

$$Ax = b$$

can be computed using **left division**.

■ **MATLAB:**

```
>> A = [ 3 2 -1; -1 3 2; 1 -1 -1];
```

```
>> b = [ 10; 5; -1];
```

```
>> x = A\b
```

```
x =
```

```
-2.0000
```

```
5.0000
```

```
-6.0000
```

Answer:

$x_1 = -2, x_2 = 5, x_3 = -6$

Some useful commands

<u>command</u>	<u>description</u>
axis ([xmin xmax ymin ymax])	Define minimum and maximum values of the axes
axis square	Produce a square plot
axis equal	equal scaling factors for both axes
axis normal	turn off axis square, equal
axis (auto)	return the axis to defaults

Plotting Curves:

plot (x,y) – generates a linear plot of the values of x (horizontal axis) and y (vertical axis).

semilogx (x,y) – generate a plot of the values of x and y using a logarithmic scale for x and a linear scale for y

semilogy (x,y) – generate a plot of the values of x and y using a linear scale for x and a logarithmic scale for y.

loglog(x,y) – generate a plot of the values of x and y using logarithmic scales for both x and y

Example: (polynomial function)

plot the polynomial using linear/linear scale, log/linear scale, linear/log scale, & log/log scale: $y = 2x^2 + 7x + 9$

```
% Generate the polynomial:
x = linspace (0, 10, 100);
y = 2*x.^2 + 7*x + 9;

% plotting the polynomial:
figure (1);
subplot (2,2,1), plot {x,y};
title ('Polynomial, linear/linear scale');
ylabel ('y'), grid;
subplot (2,2,2), semilogx {x,y};
title ('Polynomial, log/linear scale');
ylabel ('y'), grid;
subplot (2,2,3), semilogy {x,y};
title ('Polynomial, linear/log scale');
xlabel('x'), ylabel ('y'), grid;
subplot (2,2,4), loglog {x,y};
title ('Polynomial, log/log scale');
xlabel('x'), ylabel ('y'), grid;
```

Adding new curves to the existing graph:

Use the hold command to add lines/points to an existing plot.

hold on – retain existing axes, add new curves to current axes. Axes are rescaled when necessary.

hold off – release the current figure window for new plots

Grids and Labels:

<u>Command</u>	<u>Description</u>
grid on	Adds dashed grids lines at the tick marks
grid off	removes grid lines (default)
grid	toggles grid status (off to on, or on to off)
title ('text')	labels top of plot with text in quotes
xlabel ('text')	labels horizontal (x) axis with text in quotes
ylabel ('text')	labels vertical (y) axis with text in quotes
text (x,y,'text')	Adds text in quotes to location (x,y) on the current axes, where (x,y) is in units from the current plot.

Additional commands for Plotting

Color of the point or curve

Marker of the data point

Plot line styles

<u>Symbol</u>	<u>Color</u>
y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

<u>Symbol</u>	<u>Marker</u>
.	•
o	◦
x	×
+	+
*	*
s	□
d	◇
v	▽
^	△
h	hexagram

<u>Symbol</u>	<u>Line Style</u>
—	solid line
:	dotted line
-.	dash-dot line
--	dashed line

Polynomial evaluation:

Function	Description
Conv	Multiply polynomials
Deconv	Divide polynomials
Poly	Polynomial with specified roots
Polyder	Polynomial derivative
Polyfit	Polynomial curve fitting
Polyval	Polynomial evaluation
Polyvalm	Matrix polynomial evaluation
Residue	Partial-fraction expansion (residues)
Roots	Find polynomial roots

Polynomial Roots

The roots function calculates the roots of a polynomial:

>>p = [1 0 -2 -5];r =

2.0946

-1.0473 + 1.1359i -1.0473 - 1.1359i

Polynomial Evaluation

The polyval function evaluates a polynomial at a specified value. To evaluate p at s = 5, use

```
>> polyval(p,5)
```

```
ans = 110
```

To evaluate the polynomial p at X: >>X = [2 4 5; -1 0 3; 7 1 5];

```
>> Y = polyvalm(p,X)
```

```
Y =
```

```
377 179 439
```

```
111 81 136
```

```
490 253 639
```

Convolution and Deconvolution

Polynomial multiplication and division correspond to the operations convolution and deconvolution. The functions conv and deconv implement these operations. Consider the

```
>>a = [1 2 3]; b = [4 5 6]; >>c = conv(a,b)
```

```
c = 4 13 28 27 18
```

Use deconvolution to divide back out of the product:

```
>>[q,r] = deconv(c,a)
```

```
q = 4 5 6
```

```
r =
```

```
0 0 0 0
```

Polynomial Derivatives

The polyder function computes the derivative of any polynomial. To obtain the derivative of the polynomial

```
>>p = [1 0 -2 -5] >>q = polyder(p)
```

```
q = 3 0 -2
```

polyder also computes the derivative of the product or quotient of two polynomials. For example, create two polynomials a and b:

```
>>a = [1 3 5]; >>b = [2 4 6];
```

Calculate the derivative of the product a*b by calling polyder with a single output argument: >>c = polyder(a,b)

```
c =
```

```
8 30 56 38
```

Calculate the derivative of the quotient a/b by calling polyder with two output arguments:

```
>>[q,d] = polyder(a,b)
```

```
q =
```

```
-2 -8 -2
```

```
d =
```

```
4 16 40 48 36
```

q/d is the result of the operation.

Partial Fraction Expansion

'residue' finds the partial fraction expansion of the ratio of two polynomials. This is particularly useful for applications that represent systems in transfer function form. If there are no multiple roots, where r is a column vector of residues, p is a column vector of pole locations, and k is a row vector of direct terms.

Consider the transfer function `>>b = [-4 8];`

`>>a = [1 6 8]; >>[r,p,k] = residue(b,a)`

`r = -12 8`

`p = -4 -2`

`k = []`

Given three input arguments (r , p , and k), residue converts back to polynomial form:

`>>[b2,a2] = residue(r,p,k)`

`b2 = -4 8`

`a2 = 1 6 8`

MATLAB is a powerful programming language as well as an interactive computational environment. Files that contain code in the MATLAB language are called M-files. You create M-files using a text editor, then use them as you would any other MATLAB function or command. There are two kinds of M-files:

Scripts, which do not accept input arguments or return output arguments. They operate on data in the workspace. MATLAB provides a full programming language that enables you to write a series of MATLAB statements into a file and then execute them with a single command. You write your program in an ordinary text file, giving the file a name of 'filename.m'. The term you use for 'filename' becomes the new command that MATLAB associates with the program. The file extension of .m makes this a MATLAB M-file.

Functions, which can accept input arguments and return output arguments. Internal variables are local to the function.

If you're a new MATLAB programmer, just create the M-files that you want to try out in the current directory. As you develop more of your own M-files, you will want to organize them into other directories and personal toolboxes that you can add to your MATLAB search path. If you duplicate function names, MATLAB executes the one that occurs first in the search path.

Scripts:

When you invoke a script, MATLAB simply executes the commands found in the file. Scripts can operate on existing data in the workspace, or they can create new data on which to operate. Although scripts do not return output arguments, any variables that they create remain in the workspace, to be used in subsequent computations. In addition, scripts can produce graphical output using functions like plot. For example, create a file called 'myprogram.m' that contains these MATLAB commands:

`% Create random numbers and plot these numbers clc`

`clear`

`r = rand(1,50) plot(r)`

Typing the statement 'myprogram' at command prompt causes MATLAB to execute the

commands, creating fifty random numbers and plots the result in a new window. After execution of the file is complete, the variable 'r' remains in the workspace.

Functions:

Functions are M-files that can accept input arguments and return output arguments. The names of the M-file and of the function should be the same. Functions operate on variables within their own workspace, separate from the workspace you access at the MATLAB command prompt.

M-File Element	Description
Function definition line (functions only)	Defines the function name, and the number and order of input and output arguments.
H1 line	A one line summary description of the program, displayed when you request help on an entire directory, or when you use 'lookfor'.
Help text	A more detailed description of the program, displayed together with the H1 line when you request help on a specific function
Function or script body	Program code that performs the actual computations and assigns values to any output arguments.
Comments	Text in the body of the program that explains the internal workings of the program.

The first line of a function M-file starts with the keyword 'function'. It gives the function name and order of arguments. In this case, there is one input arguments and one output argument. The next several lines, up to the first blank or executable line, are comment lines that provide the help text. These lines are printed when you type 'help fact'. The first line of the help text is the H1 line, which MATLAB displays when you use the 'lookfor' command or request help on a directory. The rest of the file is the executable MATLAB code defining the function.

The variable n & f introduced in the body of the function as well as the variables on the first line are all local to the function; they are separate from any variables in the MATLAB workspace. This example illustrates one aspect of MATLAB functions that is not ordinarily found in other programming languages—a variable number of arguments. Many M-files work this way. If no output argument is supplied, the result is stored in ans. If the second input argument is not supplied, the function computes a default value.

Flow Control:

Conditional Control – if, else, switch

This section covers those MATLAB functions that provide conditional program control. if, else, and elseif. 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 as given below.

```
if <condition> <statements>;
elseif <condition> <statements>;
else
<statements>;
End
```

It is important to understand how relational operators and if statements work with

matrices. When you want to check for equality between two variables, you might use
if A == B, ...

This is valid MATLAB code, and does what you expect when A and B are scalars. But when A and B are matrices, A == B does not test if they are equal, it tests where they are equal; the result is another matrix of 0's and 1's showing element-by-element equality. (In fact, if A and B are not the same size, then A == B is an error.)

if isequal(A, B),

isequal returns a scalar logical value of 1 (representing true) or 0 (false), instead of a matrix, as the expression to be evaluated by the if function.

Using the A and B matrices from above, you get

>>isequal(A, B) ans =0. 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

```
if A > B 'greater'
```

```
elseif A < B 'less'
```

```
elseif A == B 'equal'
```

```
else
```

```
error('Unexpected situation')
```

end 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:

Several functions are helpful for reducing the results of matrix comparisons to scalar conditions for use with if, including 'isequal', 'isempty', 'all', 'any'.

Switch and Case:

The switch statement executes groups of statements based on the value of a variable or expression. The keywords case and otherwise delineate the groups. Only the first matching case is executed. The syntax is as follows

```
switch <condition or expression> case <condition>
```

```
<statements>;
```

```
...
```

```
case <condition>
```

```
...
```

```
otherwise <statements>;
```

```
end
```

There must always be an end to match the switch. An example is shown below.

```
n=5
```

```
switch rem(n,2) % to find remainder of any number 'n' case 0
```

```
disp('Even Number') % if remainder is zero
```

```
case 1
```

```
disp('Odd Number') % if remainder is one
```

```
end
```

Lab exercise:**Exercise: 1**

Consider the two polynomials $p(s)=s^2+2s+1$ and $q(s)=s+1$.

Use MATLAB to compute

- $p(s)*q(s)$
- Roots of $p(s)$ and $q(s)$
- $p(-1)$ and $q(6)$

Exercise: 2

Use MATLAB command to find the partial fraction of the following

a.
$$\frac{B(s)}{A(s)} = \frac{2s^3 + 5s^2 + 3s + 6}{s^3 + 6s^2 + 11s + 6}$$

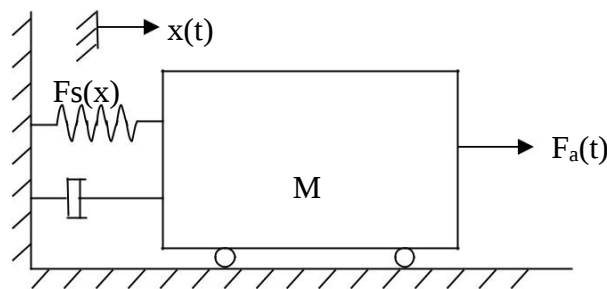
LAB SESSION 02

OBJECT: Mathematical models of physical systems in the design and analysis of control systems.

Theory

Mass-Spring System Model

Consider the following Mass-Spring system shown in the figure. Where $F_s(x)$ is the spring force, $F_f()$ is the friction coefficient, $x(t)$ is the displacement and $F_a(t)$ is the applied force:



$a = dv(t)/dt = d^2x(t)/dt^2$ is the acceleration

$dx(t)$ is the displacement

According to the laws of physics

$$Ma + F_f(v) + F_s(x) = F_a(t)$$

The differential equation for the above Mass-Spring system can then be written as follows

$$M(dx^2/dt^2) + B(dx(t)/dt) + Kx(t) = F_a(t) \dots\dots\dots (1)$$

B is called the friction coefficient and K is called the spring constant.

- **B** is called the friction coefficient and
- **K** is called the spring constant.

The linear differential equation of second order (2) describes the relationship between the displacement and the applied force. The differential equation can then be used to study the time behavior of $x(t)$ under various changes of the applied force. In reality, the spring force and/or the friction force can have a more complicated expression or could be represented by a graph or data table. For instance, a nonlinear spring can be designed (see figure 2.2) such that $r > 1$.



Solving the differential equation using MATLAB:

The objectives behind modeling the mass-damper system can be many and may include

- Understanding the dynamics of such system
- Studying the effect of each parameter on the system such as mass M , the friction coefficient B , and the elastic characteristic $F_s(x)$.
- Designing a new component such as damper or spring.
- Reproducing a problem in order to suggest a solution.

MATLAB can help solve linear or nonlinear ordinary differential equations (ODE). To show how you can solve ODE using MATLAB we will proceed in two steps. We first see how can we solve first order ODE and second

PROCEDURE:**Speed Cruise Control example:**

When $F_s(x)=0$, which means that $K=0$, Equation (1) becomes

$$M(d^2x(t)/dt^2) + B(dx(t)/dt) = F_a(t)$$

or

$$M(dv(t)/dt) + Bv = F_a(t)$$

Using MATLAB solver ode45 we can write do the following:

1 create a MATLAB-function cruise_speed.m

```
function dvdt=cruise_speed(t, v)
```

```
%flow rate
```

```
M=750; %(Kg)
```

```
B=30; %( Nsec/m)
```

```
Fa=300; %N
```

```
% dv/dt=Fa/M-B/M v
```

```
dvdt=Fa/M-B/M*v;
```

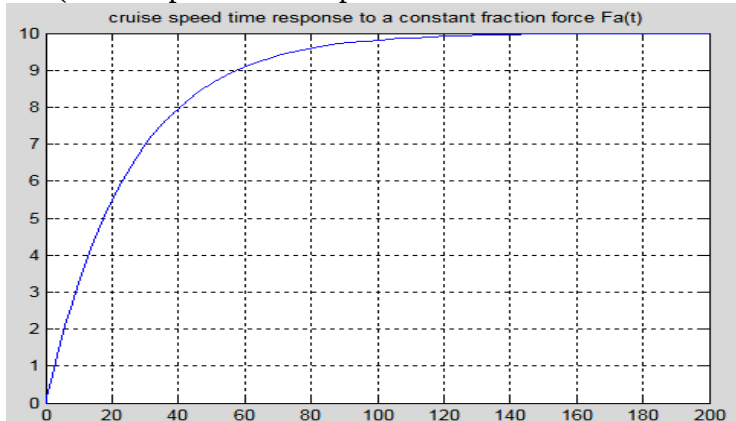
2 create a new MATLAB m-file and write

```
v0= 0; %(initial speed)
```

```
[t,v]=ode45('cruise_speed', [0 200],v0);
```

```
plot(t,v); grid on;
```

```
title('cruise speed time response to a constant traction force Fa(t) ')
```



In the above program the behavior of a car speed is shown in which the car starts with rest position, after that it attains its maximum speed so that it reaches its maximum limit then after that its speed becomes constant throughout the time.

Mass-Spring System Example:

$$M(d^2x(t)/dt^2) + B(dx(t)/dt) + Kx(t) = F_a(t)$$

Variables	New variable	Differential equation
$x(t)$	X_1	$\frac{dX_1}{dt} = X_2$
$dx(t)/dt$	X_2	$\frac{dX_2}{dt} = -\frac{B}{M}X_2 - \frac{K}{M}X_1 + \frac{F_a(t)}{M}$

In vector form

$$\dot{X} = \begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \end{bmatrix} \quad \frac{dX}{dt} = \begin{bmatrix} \frac{dX_1}{dt} \\ \frac{dX_2}{dt} \end{bmatrix}$$

The system equations can be written as:

$$\frac{dX}{dt} = -\frac{B}{M}X_2 - \frac{K}{M}X_1 + \frac{F_a(t)}{M}$$

1. create a MATLAB-function mass_spring.m

```
function dXdt=mass_spring(t,X)
```

```
M=705; % (Kg)
```

```
B=30; % (Nsec/m)
```

```
Fa=300; % (N)
```

```
K=15; % (N/m)
```

```
r=1; % dX/dt
```

```
dXdt(1,1)=X(2);
```

```
dXdt(2,1)=-B/M*X(2)-K/M*X(1)+Fa/M;
```

2. Program of mass spring system with r=1

```
clear all
```

```
close all
```

```
clc
```

```
X0=[0;0];% (Initial speed and position)
```

```
[t,X]=ode45('mass_spring',[0 200],X0);
```

```
figure;
```

```
plot(t,X(:,1));
```

```
xlabel('Time(t)');
```

```
ylabel('Position');
```

```
title('Mass spring system');
```

```
legend('Position ');
```

```
grid;
```

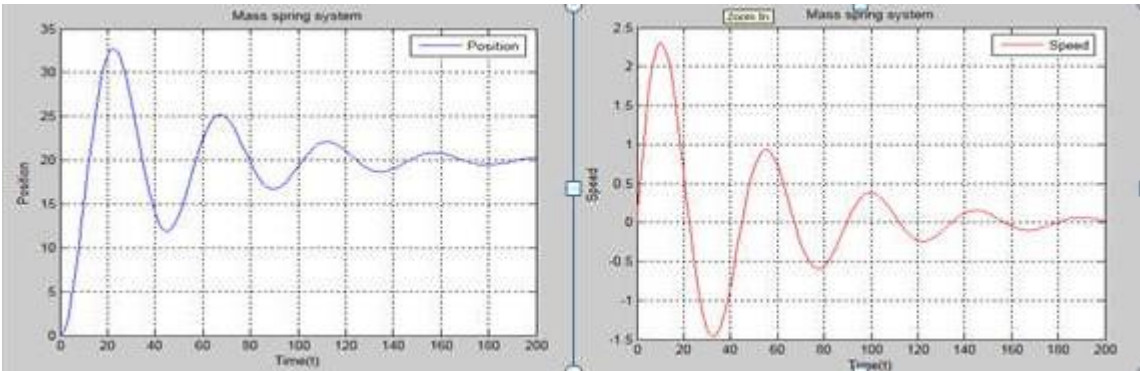
```
figure;
```

```
plot(t,X(:,2),'r');
```

```
xlabel('Time(t)');
```

```
label('Speed');
```

```
--title('Mass spring system');
legend('Speed ');
grid;
```



OBSERVATIONS:

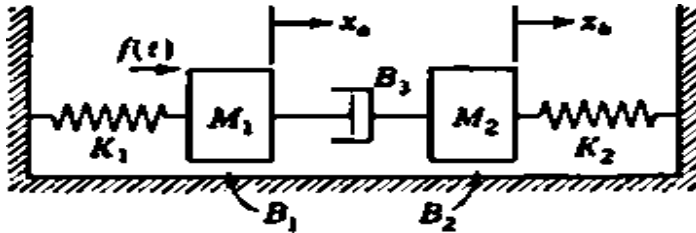
Parameter	Behavior of system
Mass	
Friction Coefficient	
Stiffness	
Applied Force	

CONCLUSION:

LAB SESSION 03

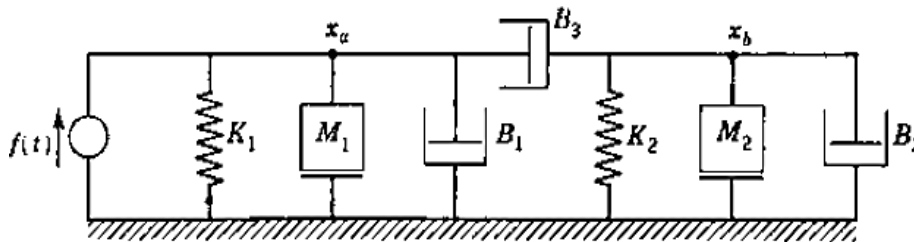
OBJECT: Mathematical modeling of Multiple-Element Mechanical Translation System

Theory:



Where;

- $f(t)$ is applied force to the mass M_1 .
- B_1 and B_2 are the viscous friction coefficients indicating the sliding friction between the masses M_1 and M_2 and the surface.



According to the rules for node equations:

For node a: $(M_1 D^2 + B_1 D + B_3 D + K_1)x_a - (B_3 D)x_b = f$

For node b: $-(B_3 D)x_a + (M_2 D^2 + B_2 D + B_3 D + K_2)x_b = 0$

$X_1 = X_b$ for spring K_2

$X_2 = X'_1 = V_b$

$X_3 = X_a$ for spring K_1

$X_4 = X'_3 = V_a$

The system equations are:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{K_2}{M_2} & -\frac{B_2 + B_3}{M_2} & 0 & \frac{B_3}{M_2} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{B_3}{M_1} & -\frac{K_1}{M_1} & -\frac{B_1 + B_3}{M_1} \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{M_1} \end{bmatrix}$$

PROCEDURE:

1. create a MATLAB-function multiple_element_sys.m

function dXdt=multiple_element_sys (t,X)

Fa=300; %(N)

M1=750; %(Kg)

M2=750; %(Kg)

B1=20; %(Nsec/m)

B2=20; %(Nsec/m)

B3=30; %(Nsec/m)

K1=15; %(N/m)

```

K2=15;                %(N/m)
dXdt(1,1)=X(2);
dXdt(2,1)=-K2/M2*X(1)-((B1+B2)/M2)*X(2)+B3*X(4)/M2;
dXdt(3,1)=X(4);
dXdt(4,1)=B3/M1*X(2)-K1/M1*X(3)-((B1+B3)/M1)*X(4)+Fa/M1;

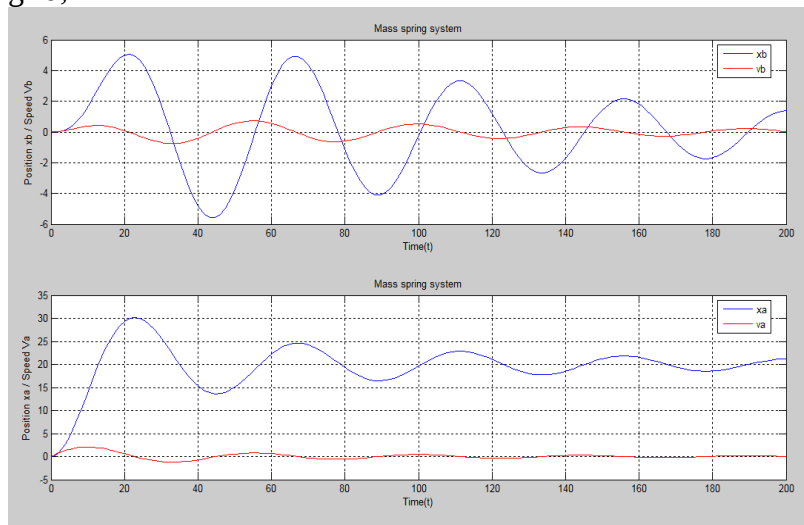
```

2. Write another M. file to call the function:

```

clear all;
close all;
clc;
X0=[0;0;0;0]; % (Initial xb, Vb, xa, Va)
[t,X]=ode45('multiple_element_sys',[0 200],X0);
figure;
subplot(2,1,1);
plot(t,X(:,1));
plot(t,X(:,2),'r');
xlabel('Time(t)');
ylabel('Position xb / Speed Vb');
title('Mass spring system');
legend('xb', 'Vb');
grid;
subplot(2,1,2);
plot(t,X(:,3));
hold;
plot(t,X(:,4),'r');
xlabel('Time(t)');
ylabel('Position xa / Speed Va');
title('Mass spring system');
legend('xa', 'Va');
grid;

```



OBSERVATIONS:

Parameter		Behavior of system
Mass	M1	
	M2	
Friction Coefficient	B1	
	B2	
	B3	
Stifness	K1	
	K2	
Applied Force	F _a	

CONCLUSION:

NED University of Engineering & Technology
Department of _____ Engineering



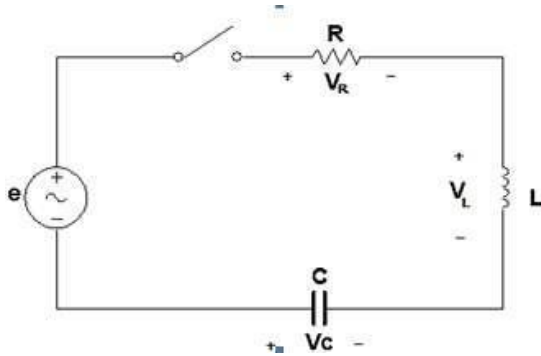
Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
Software Menu Identification and Usage: Ability to initialise, configure and <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc.	Unable to understand and use software menu	Little ability and understanding of software menu operation, makes many mistake	Moderate ability and understanding of software menu operation, makes lesser mistakes	Reasonable understanding of software menu operation, makes no major mistakes	Demonstrates command over software menu usage with frequent use of advance menu options
Procedural Programming of given model: <u>Practice</u> procedural programming techniques, in order to code specific model	Little to no understanding of procedural programming techniques	Slight ability to use procedural programming techniques for coding given algorithm	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given model	Correctly recognises and uses procedural programming techniques with no errors but unable to run model successfully	Correctly recognises and uses procedural programming techniques with no errors and runs model successfully
Relating Theoretical Concepts, Equations and Transforms to Code: <u>Recognise</u> relation between model concepts and written code and <u>manipulate</u> the code in accordance of requirements	Completely unable to relate between model concepts and written code, unable to do manipulations	Able to recognise some relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, able to do some manipulations	Able to recognise relation between model concepts and written code, able to completely manipulate code in line with theoretical concepts
Detecting and Removing Errors: <u>Detect</u> Errors/Exceptions and in simulation and <u>manipulate</u> code to rectify the simulation	Unable to check and detect error messages and indications in software	Able to find error messages and indications in software but no understanding of detecting those errors and their types	Able to find error messages and indications in software as well as understanding of detecting some of those errors and their types	Able to find error messages in software as well as understanding of detecting all of those errors and their types	Able to find error messages in software along with the understanding to detect and rectify them

Graphical Visualisation and Comparison of model Parameters: <u>Manipulate</u> given simulation under supervision, in order to produce graphs/plots for measuring and comparing model parameters	Unable to understand and utilise visualisation or plotting features	Ability to understand and utilise visualisation and plotting features with frequent errors	Ability to understand and utilise visualisation and plotting features successfully but unable to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, partially able to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, also able to compare and analyse them
Following step-by-step procedure to complete lab work: <u>Observe, imitate and operate</u> software to complete the provided sequence of steps	Inability to recognise and perform given lab procedures	Able to recognise given lab procedures and perform them but could not follow the prescribed order of steps	Able to recognise given lab procedures and perform them by following prescribed order of steps, with frequent mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with occasional mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with no mistakes
Recording Simulation Observations: <u>Observe and copy</u> prescribed or required simulation results in accordance with lab manual instructions	Inability to recognise prescribed or required simulation measurements	Able to recognise prescribed or required simulation measurements but does not record according to given instructions	—	Able to recognise prescribed or required simulation measurements but records them incompletely	Able to recognise prescribed or required simulation measurements and records them completely, in tabular form
Discussion and Conclusion: <u>Demonstrate</u> discussion capacity on the recorded observations and draw conclusions from it, relating them to theoretical principles/concepts	Complete inability to discuss recorded observations and draw conclusions	Slight ability to discuss recorded observations and draw conclusions	Moderate ability to discuss recorded observations and draw conclusions	Reasonable ability to discuss recorded observations and draw conclusions	Full ability to discuss recorded observations and draw conclusions

LAB SESSION 04

OBJECT: Mathematical modeling of Electrical System

Theory:



- e is applied Potential.
- i is the mesh current.

The differential equations for the given figure.

According to Mesh Analysis:

$$e(t) = V_L + V_C + V_R$$

$$e(t) = LD \dot{i} + \frac{1}{CD} i + iR$$

The state equations for the given figure.

This circuit contains two energy-storage elements, Inductor and capacitor.

Let state variables are

$X_1 = V_C$ the voltage across the capacitor and

$X_2 = i$ the current in the inductor.

STATE EQUATION:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{C} \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} [u]$$

PROCEDURE:

1. create a MATLAB-function RLC.m

```
function dXdt=RLC(t,X)
```

```
e=60; % (V)
```

```
R=10; % (Ohm)
```

```
L=1; % (H)
```

```
C=10; % (F)
```

```
%dX/dt
```

```
dXdt(1,1)=(1/C)*X(2);
```

$$dXdt(2,1)=(-1/L)*X(1)-(R/L)*X(2)+(1/L)*e;$$

2. Write an other M. file to call the function:

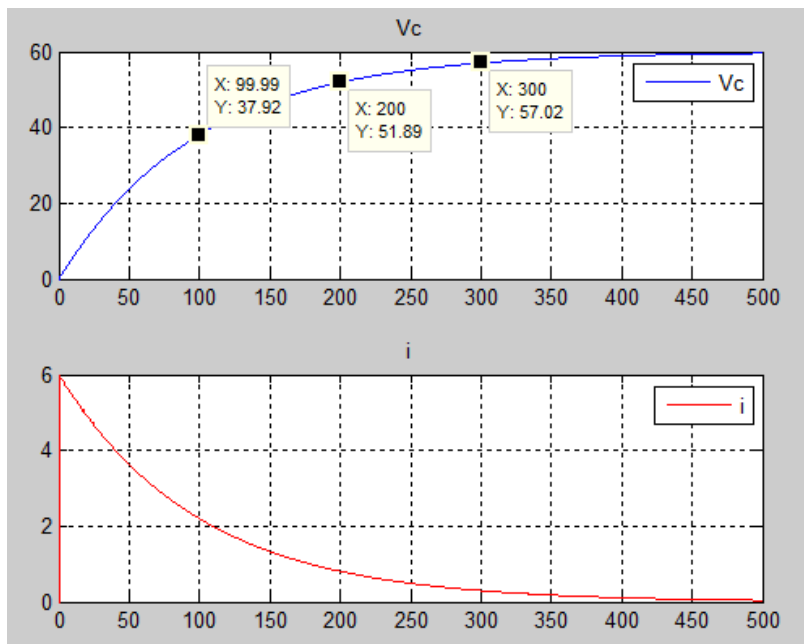
```
clear
close
Clc
X0=[0 0];
[t,X]=ode45('RLC',[0 500],X0);
subplot(2,1,1);
plot(t,X(:,1));
legend('Vc');
grid on;
title('Vc');
subplot(2,1,2);
plot(t,X(:,2),'r');
legend('i');
grid on;
title('i');
```

Graph:

➤ Time constant = $RC = 10*10 = 100$ sec

For first time constant :

➤ $V_c = 63.2\% * e = 0.632*60 = 37.92$ V

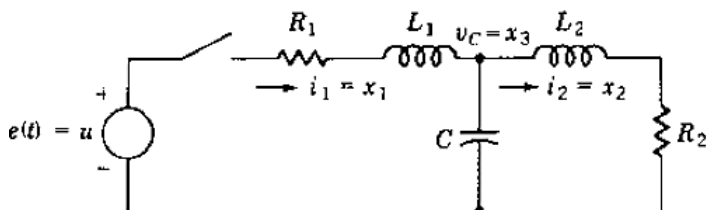


OBSERVATIONS:

Parameter	Behavior of system
Voltage source(e)	
Resistance(R)	
Inductance(L)	
Capacitance(C)	

CONCLUSION:**EXERCISE:**

Write the function and program of the following circuit diagram. Also explain the plots of the respective state variables.



NED University of Engineering & Technology
Department of _____ Engineering



Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
Software Menu Identification and Usage: Ability to initialise, configure and <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc.	Unable to understand and use software menu	Little ability and understanding of software menu operation, makes many mistake	Moderate ability and understanding of software menu operation, makes lesser mistakes	Reasonable understanding of software menu operation, makes no major mistakes	Demonstrates command over software menu usage with frequent use of advance menu options
Procedural Programming of given model: <u>Practice</u> procedural programming techniques, in order to code specific model	Little to no understanding of procedural programming techniques	Slight ability to use procedural programming techniques for coding given algorithm	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given model	Correctly recognises and uses procedural programming techniques with no errors but unable to run model successfully	Correctly recognises and uses procedural programming techniques with no errors and runs model successfully
Relating Theoretical Concepts, Equations and Transforms to Code: <u>Recognise</u> relation between model concepts and written code and <u>manipulate</u> the code in accordance of requirements	Completely unable to relate between model concepts and written code, unable to do manipulations	Able to recognise some relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, able to do some manipulations	Able to recognise relation between model concepts and written code, able to completely manipulate code in line with theoretical concepts
Detecting and Removing Errors: <u>Detect</u> Errors/Exceptions and in simulation and <u>manipulate</u> code to rectify the simulation	Unable to check and detect error messages and indications in software	Able to find error messages and indications in software but no understanding of detecting those errors and their types	Able to find error messages and indications in software as well as understanding of detecting some of those errors and their types	Able to find error messages in software as well as understanding of detecting all of those errors and their types	Able to find error messages in software along with the understanding to detect and rectify them

Graphical Visualisation and Comparison of model Parameters: <u>Manipulate</u> given simulation under supervision, in order to produce graphs/plots for measuring and comparing model parameters	Unable to understand and utilise visualisation or plotting features	Ability to understand and utilise visualisation and plotting features with frequent errors	Ability to understand and utilise visualisation and plotting features successfully but unable to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, partially able to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, also able to compare and analyse them
Following step-by-step procedure to complete lab work: <u>Observe, imitate and operate</u> software to complete the provided sequence of steps	Inability to recognise and perform given lab procedures	Able to recognise given lab procedures and perform them but could not follow the prescribed order of steps	Able to recognise given lab procedures and perform them by following prescribed order of steps, with frequent mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with occasional mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with no mistakes
Recording Simulation Observations: <u>Observe and copy</u> prescribed or required simulation results in accordance with lab manual instructions	Inability to recognise prescribed or required simulation measurements	Able to recognise prescribed or required simulation measurements but does not record according to given instructions	—	Able to recognise prescribed or required simulation measurements but records them incompletely	Able to recognise prescribed or required simulation measurements and records them completely, in tabular form
Discussion and Conclusion: <u>Demonstrate</u> discussion capacity on the recorded observations and draw conclusions from it, relating them to theoretical principles/concepts	Complete inability to discuss recorded observations and draw conclusions	Slight ability to discuss recorded observations and draw conclusions	Moderate ability to discuss recorded observations and draw conclusions	Reasonable ability to discuss recorded observations and draw conclusions	Full ability to discuss recorded observations and draw conclusions

LAB SESSION 05

OBJECT: Mathematical modeling of Electrical System by Simulink

THEORY:

To use graphical user interface diagrams to model the physical systems for the purpose of design and analysis of control systems. We will learn how **MATLAB/SIMULINK** helps in solving such models. This lab introduces powerful graphical user interface (GUI), **Simulink** of MATLAB.

This software is used for solving the modeling equations and obtaining the response of a system to different inputs. Both linear and nonlinear differential equations can be solved numerically with high precision and speed, allowing system responses to be calculated and displayed for many input functions.

A block diagram is an interconnection of blocks representing basic mathematical operations in such a way that the overall diagram is equivalent to the system's mathematical model.

The lines interconnecting the blocks represent the variables describing the system behavior. Block diagrams can represent modeling equations in both input-output and state variable form.

SIMULINK

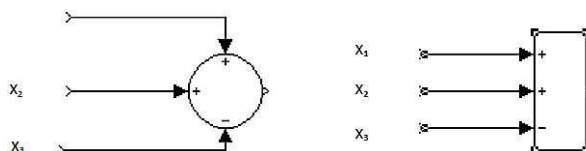
Simulink provides access to an extensive set of blocks that accomplish a wide range of functions useful for the simulation and analysis of dynamic systems. The blocks are grouped into libraries, by general classes of functions.

- Mathematical functions such as summers and gains are in the Math library.
- Integrators are in the Continuous library.
- Constants, common input functions, and clock can all be found in the Sources library.
- Scope, To Workspace blocks can be found in the Sinks library.
- Simulink uses blocks to write a program.
- Blocks are arranged in various libraries according to their functions.
- Properties of the blocks and the values can be changed in the associated dialog boxes.

Some of the blocks are given below.

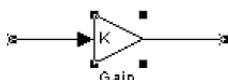
SUM (Math library)

The sum block can be represented in two ways in Simulink, by a circle or by a rectangle.



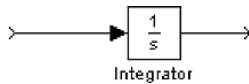
GAIN (Math library)

A gain block is shown by a triangular symbol, with the gain expression written inside



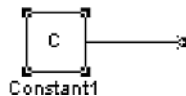
INTEGRATOR (Continuous library)

- The block for an integrator as shown below looks unusual. The quantity $1/s$ comes from the Laplace transform expression for integration.



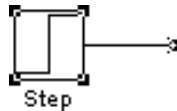
CONSTANTS (Source library)

- Constants are created by the Constant block .
- Double- clicking on the symbol opens a dialog box to establish the constant's value.
- It can be a number or an algebraic expression using constants whose values are defined in the workspace .



STEP (Source library)

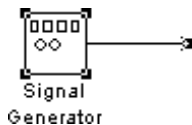
- A Simulink block is provided for a Step input, a signal that changes (usually from zero) to a specified new, constant level at a specified time. These levels and time can be specified through the dialog box, obtained by double-clicking on the Step block.



SIGNAL GENERATOR (Source library)

- One source of repetitive signals in Simulink is called the Signal Generator.
- Double-clicking on the Signal Generator block opens a dialog box, where a sine wave, a square wave, a ramp (saw tooth), or a random waveform can be chosen.
- In addition, the amplitude and frequency of the signal may be specified.
- The signals produced have a mean value of zero.

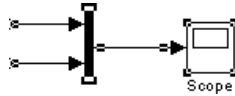
The repetition frequency can be given in Hertz (Hz).



SCOPE (Sinks library)

- The system response can be examined graphically, as the simulation runs, using the Scope block in the sinks library .
- This name is derived from the electronic instrument, oscilloscope which performs a similar function with electronic signals.
- Any of the variables in a Simulink diagram can be connected to the Scope block, and when the simulation is started, that variable is displayed.
- It is possible to include several Scope blocks.
- Also it is possible to display several signals in the same scope block using a MUX block in the signals & systems library.

- The Scope normally chooses its scales automatically to best display the data.



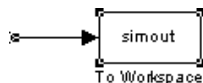
CLOCK (Sources library)

- The clock produces the variable “time” that is associated with the integrators as MATLAB calculates a numerical (digital) solution to a model of a continuous system.
- The result is a string of sample values of each of the output variables.
- These samples are not necessarily at uniform time increments, so it is necessary to have the variable “time” that contains the time corresponding to each sample point.
- Then MATLAB can make plots versus “time.”
- The clock output could be given any arbitrary name; we use “t” in most of the cases.



To Workspace (Sinks library)

- The To Workspace block is used to return the results of a simulation to the MATLAB workspace, where they can be analyzed and/or plotted.
- Any variable in a Simulink diagram can be connected to a To Workspace block.
- In our exercises, all of the state variables and the input variables are usually returned to the workspace.



In the Simulink diagram, the appearance of a block can be changed by changing the foreground or background colors, or by drop shadow or other options available in the format drop down menu. The available options can be reached in the Simulink window by highlighting the block, then clicking the right mouse button. The Show Drop Shadow option is on the format drop-down menu.

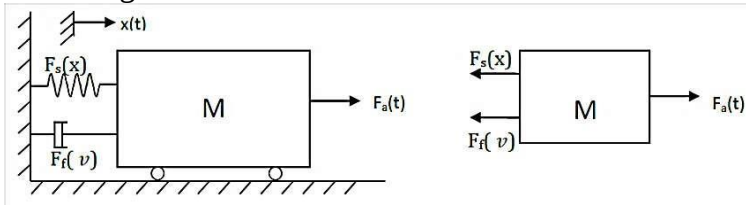
GENERAL INSTRUCTIONS FOR WRITING A SIMULINK PROGRAM

- To create a simulation in Simulink, follow the steps:
- Start MATLAB.
- Start Simulink.
- Open the libraries that contain the blocks you will need. These usually will include the Sources, Sinks, Math and Continuous libraries, and possibly others.
- Open a new Simulink window.
- Drag the needed blocks from their library folders to that window. The Math library, for example, contains the Gain and Sum blocks.
- Arrange these blocks in an orderly way corresponding to the equations to be solved.

- Interconnect the blocks by dragging the cursor from the output of one block to the input of another block. Interconnecting branches can be made by right-clicking on an existing branch.
- Double-click on any block having parameters that must be established, and set these parameters. For example, the gain of all Gain blocks must be set. The number and signs of the inputs to a Sum block must be established. The parameters of any source blocks should also be set in this way.
- It is necessary to specify a stop time for the solution. This is done by clicking on the Simulation > Parameters entry on the Simulink toolbar.

Mass-Spring System Model

Consider the Mass-Spring system used in the previous exercise as shown in the figure.



Where;

- $F_s(x)$ is the spring force.
- $F_f(t)$ is the friction coefficient
- $x(t)$ is the displacement
- $F_a(t)$ is the applied force

The differential equation for the above Mass-Spring system can then be written as follows

$$M \frac{d^2 x(t)}{dt^2} + B \frac{dx(t)}{dt} + Kx(t) = F_a(t)$$

PROCEDURE:

Modeling of a second order system

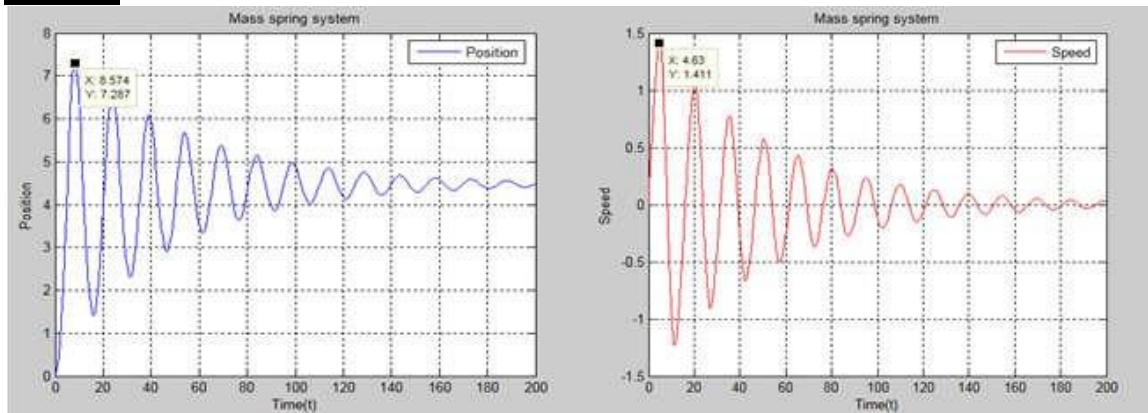
- Construct a Simulink diagram to calculate the response of the Mass-Spring system.
- The input force increases from 0 to 8 N at $t = 1$ s.
- The parameter values are $M = 2$ kg, $K = 16$ N/m, and $B = 4$ N.s/m.
- Draw the free body diagram.
- Write the modeling equation from the free body diagram
- Solve the equations for the highest derivative of the output.
- Draw a block diagram to represent this equation.
- Draw the corresponding Simulink diagram.
- Use Step block to provide the input $f_a(t)$.
- In the Step block, set the initial and final values and the time at which the step occurs.
- Use the “To Workspace” blocks for t , $f_a(t)$, x , and v in order to allow MATLAB to plot the desired responses. Set the save format to array in block parameters.
- Select the duration of the simulation to be 10 seconds from the Simulation > Parameters entry on the toolbar

M-file for parameter values

```
Fa = 300;    % N  
M = 750;    %kg  
K = 15;     %N/m  
B = 30;     %Ns/m
```

Plotting the outputs in MATLAB:

```
sim('mass_spring')    % Has the same effect as clicking on  
                        % Start on the toolbar.  
plot(t,x);             % Plots the initial run with B=4  
hold on;               % Plots later results on the same axes  
                        % as the first.  
B = 25;                % New value of B; other parameter values  
                        % stay the same.  
sim('mass_spring')    % Rerun the simulation with new B value.  
plot(t,x);             % Plots new x on original axes.  
B = 50;  
sim('mass_spring');  
plot(t,x);
```

GRAPH:**EXERCISE:**

1. Use SIMULINK to model the same systems which you have already done in the previous Lab 3 and Lab 4.
2. Plot the same outputs and compare it with the previous plots.

CONCLUSION:

NED University of Engineering & Technology
Department of _____ Engineering



Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
Software Menu Identification and Usage: Ability to initialise, configure and <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc.	Unable to understand and use software menu	Little ability and understanding of software menu operation, makes many mistake	Moderate ability and understanding of software menu operation, makes lesser mistakes	Reasonable understanding of software menu operation, makes no major mistakes	Demonstrates command over software menu usage with frequent use of advance menu options
Procedural Programming of given model: <u>Practice</u> procedural programming techniques, in order to code specific model	Little to no understanding of procedural programming techniques	Slight ability to use procedural programming techniques for coding given algorithm	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given model	Correctly recognises and uses procedural programming techniques with no errors but unable to run model successfully	Correctly recognises and uses procedural programming techniques with no errors and runs model successfully
Relating Theoretical Concepts, Equations and Transforms to Code: <u>Recognise</u> relation between model concepts and written code and <u>manipulate</u> the code in accordance of requirements	Completely unable to relate between model concepts and written code, unable to do manipulations	Able to recognise some relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, able to do some manipulations	Able to recognise relation between model concepts and written code, able to completely manipulate code in line with theoretical concepts
Detecting and Removing Errors: <u>Detect</u> Errors/Exceptions and in simulation and <u>manipulate</u> code to rectify the simulation	Unable to check and detect error messages and indications in software	Able to find error messages and indications in software but no understanding of detecting those errors and their types	Able to find error messages and indications in software as well as understanding of detecting some of those errors and their types	Able to find error messages in software as well as understanding of detecting all of those errors and their types	Able to find error messages in software along with the understanding to detect and rectify them

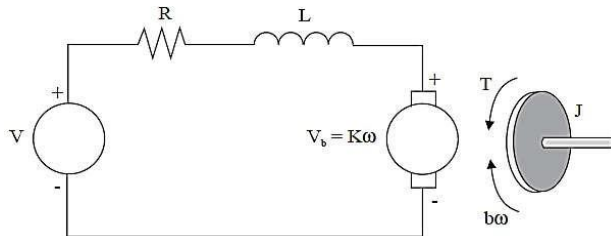
Graphical Visualisation and Comparison of model Parameters: <u>Manipulate</u> given simulation under supervision, in order to produce graphs/plots for measuring and comparing model parameters	Unable to understand and utilise visualisation or plotting features	Ability to understand and utilise visualisation and plotting features with frequent errors	Ability to understand and utilise visualisation and plotting features successfully but unable to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, partially able to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, also able to compare and analyse them
Following step-by-step procedure to complete lab work: <u>Observe, imitate and operate</u> software to complete the provided sequence of steps	Inability to recognise and perform given lab procedures	Able to recognise given lab procedures and perform them but could not follow the prescribed order of steps	Able to recognise given lab procedures and perform them by following prescribed order of steps, with frequent mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with occasional mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with no mistakes
Recording Simulation Observations: <u>Observe and copy</u> prescribed or required simulation results in accordance with lab manual instructions	Inability to recognise prescribed or required simulation measurements	Able to recognise prescribed or required simulation measurements but does not record according to given instructions	—	Able to recognise prescribed or required simulation measurements but records them incompletely	Able to recognise prescribed or required simulation measurements and records them completely, in tabular form
Discussion and Conclusion: <u>Demonstrate</u> discussion capacity on the recorded observations and draw conclusions from it, relating them to theoretical principles/concepts	Complete inability to discuss recorded observations and draw conclusions	Slight ability to discuss recorded observations and draw conclusions	Moderate ability to discuss recorded observations and draw conclusions	Reasonable ability to discuss recorded observations and draw conclusions	Full ability to discuss recorded observations and draw conclusions

LAB SESSION 06

OBJECT: Develop a linear model for a DC motor.

THEORY:

Consider a DC motor, whose electric circuit of the armature and the free body diagram of the rotor are shown in Figure.



Consider the following values for the physical parameters:

moment of inertia of the rotor	$J = 0.01 \text{ kg} \cdot \text{m}^2$
damping (friction) of the mechanical system	$b = 0.1 \text{ Nms}$
(back-)electromotive force constant	$K = 0.01 \text{ Nm/A}$
electric resistance	$R = 1 \Omega$
electric inductance	$L = 0.5 \text{ H}$

The input is the armature voltage V (e_a) in Volts (driven by a voltage source).

Measured variables are the angular

velocity of the shaft ω in radians per second, and

the shaft angle Q in radians.

We can write the following equations based on the Newton's law combined with the Kirchhoff's law:

$$L(di/dt) + Ri = V - K(d\theta/dt)$$

$$J(d^2\theta/dt^2) + b(d\theta/dt) = Ki$$

Or

$$L_m Di_m + R_m i_m + e_m = e_a$$

$$J D \omega_n + B \omega_n = T$$

Transfer Function:

The transfer function from the input voltage, $V(s)$, to the output angle, Q , directly follows:

$$\frac{\theta(s)}{V(s)} = \frac{K}{s[(R + Ls)(Js + b) + K^2]}$$

And the transfer function from the input voltage, $V(s)$, to the output velocity of the shaft

$$\frac{\omega(s)}{V(s)} = \frac{K}{(R + Ls)(Js + b) + K^2}$$

ω in radians per second, is:

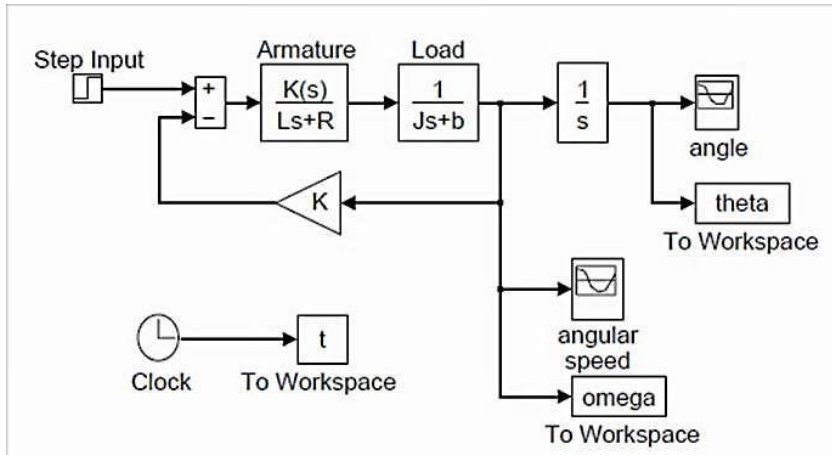
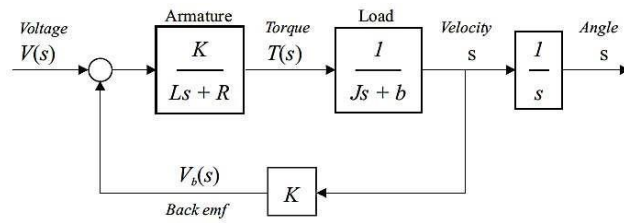
PROCEDURE:

SIMULINK Model

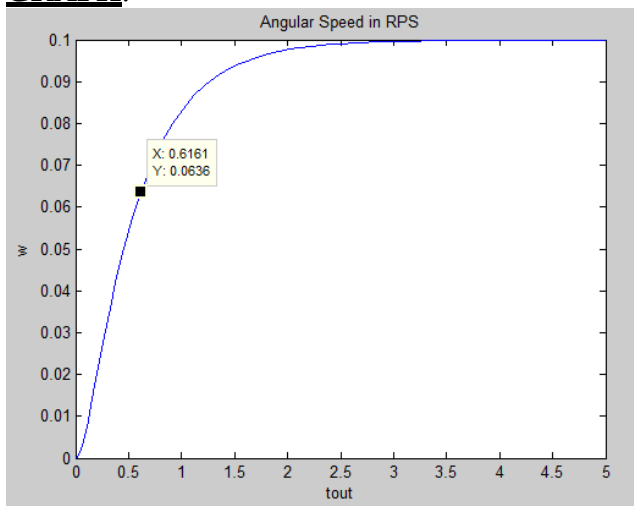
$$J = 0.01 \quad \% \text{kg m}^2$$

$$b = 0.1 \quad \% \text{Nms}$$

$$\begin{aligned} K &= 0.01 \quad \% \text{Nm/A} \\ R &= 1 \quad \% \text{ohm} \\ L &= 0.5 \quad \% \text{H} \end{aligned}$$



GRAPH:



OBSERVATIONS:

Parameters	Effects on system
Moment of inertia	
Resistance	
Inductance	
Friction Coefficient	

CONCLUSION:**EXERCISE:**

1. Change the electrical parameters such as 'R' or 'L' to reduce the Time Constant of motor.
2. Also change the mechanical parameters such as 'J' or 'B' to reduce the Time Constant to zero.
3. What are the parameters which are responsible to change the speed of the rotor? Explain with graph.

NED University of Engineering & Technology
Department of _____ Engineering



Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
Software Menu Identification and Usage: Ability to initialise, configure and <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc.	Unable to understand and use software menu	Little ability and understanding of software menu operation, makes many mistake	Moderate ability and understanding of software menu operation, makes lesser mistakes	Reasonable understanding of software menu operation, makes no major mistakes	Demonstrates command over software menu usage with frequent use of advance menu options
Procedural Programming of given model: <u>Practice</u> procedural programming techniques, in order to code specific model	Little to no understanding of procedural programming techniques	Slight ability to use procedural programming techniques for coding given algorithm	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given model	Correctly recognises and uses procedural programming techniques with no errors but unable to run model successfully	Correctly recognises and uses procedural programming techniques with no errors and runs model successfully
Relating Theoretical Concepts, Equations and Transforms to Code: <u>Recognise</u> relation between model concepts and written code and <u>manipulate</u> the code in accordance of requirements	Completely unable to relate between model concepts and written code, unable to do manipulations	Able to recognise some relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, able to do some manipulations	Able to recognise relation between model concepts and written code, able to completely manipulate code in line with theoretical concepts
Detecting and Removing Errors: <u>Detect</u> Errors/Exceptions and in simulation and <u>manipulate</u> code to rectify the simulation	Unable to check and detect error messages and indications in software	Able to find error messages and indications in software but no understanding of detecting those errors and their types	Able to find error messages and indications in software as well as understanding of detecting some of those errors and their types	Able to find error messages in software as well as understanding of detecting all of those errors and their types	Able to find error messages in software along with the understanding to detect and rectify them

Graphical Visualisation and Comparison of model Parameters: <u>Manipulate</u> given simulation under supervision, in order to produce graphs/plots for measuring and comparing model parameters	Unable to understand and utilise visualisation or plotting features	Ability to understand and utilise visualisation and plotting features with frequent errors	Ability to understand and utilise visualisation and plotting features successfully but unable to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, partially able to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, also able to compare and analyse them
Following step-by-step procedure to complete lab work: <u>Observe, imitate and operate</u> software to complete the provided sequence of steps	Inability to recognise and perform given lab procedures	Able to recognise given lab procedures and perform them but could not follow the prescribed order of steps	Able to recognise given lab procedures and perform them by following prescribed order of steps, with frequent mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with occasional mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with no mistakes
Recording Simulation Observations: <u>Observe and copy</u> prescribed or required simulation results in accordance with lab manual instructions	Inability to recognise prescribed or required simulation measurements	Able to recognise prescribed or required simulation measurements but does not record according to given instructions	—	Able to recognise prescribed or required simulation measurements but records them incompletely	Able to recognise prescribed or required simulation measurements and records them completely, in tabular form
Discussion and Conclusion: <u>Demonstrate</u> discussion capacity on the recorded observations and draw conclusions from it, relating them to theoretical principles/concepts	Complete inability to discuss recorded observations and draw conclusions	Slight ability to discuss recorded observations and draw conclusions	Moderate ability to discuss recorded observations and draw conclusions	Reasonable ability to discuss recorded observations and draw conclusions	Full ability to discuss recorded observations and draw conclusions

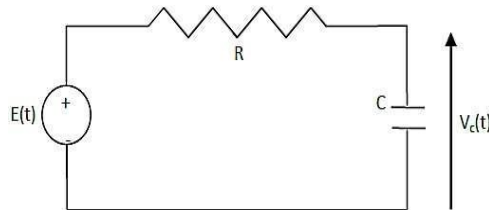
LAB SESSION 07

OBJECT: Performance of First order and Second order systems and development of Time response specifications function

THEORY:

First order system:

An electrical RC-circuit is the simplest example of a first order system. It comprises of a resistor and capacitor connected in series to a voltage supply as shown below on Figure 1



Where ;

- **$V_c(t)$** is the voltage across the capacitor,
- **R** is the resistance and
- **C** is the capacitance.

Obtain the transfer function of the above electrical circuit. (Take V_c as output and $V_c(0) = V_0$)

For the RC-circuit as shown in Fig. 1, the equation governing its behavior is given by :

$$\frac{dv_c(t)}{dt} + \frac{1}{RC} v_c(t) = \frac{1}{RC} E \quad \text{where } v_c(0) = v_0$$

The constant $\tau = RC$ is the time constant of the system and is defined as the time required by the system output i.e. $V_c(t)$ to rise to 63% of its final value (which is E). Hence the above equation can be expressed in terms of the time constant as:

$$\tau \frac{dv_c(t)}{dt} + v_c(t) = E$$

Transfer Function

Obtaining the transfer function of the above differential equation, we get

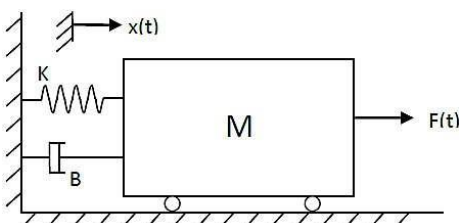
$$\frac{V_c(s)}{E(s)} = \frac{1}{\tau s + 1}$$

The above system is known as the first order system.

The performance measures of a first order system are its **time constant** and its **steady state**.

Second Order System:

Consider the following Mass-Spring system shown



Where ;

- K is the spring constant,
- B is the friction coefficient,
- $x(t)$ is the displacement and
- $F(t)$ is the applied force:

The differential equation for the above Mass-Spring system can be derived as follows:

$$M \frac{d^2 x(t)}{dt^2} + B \frac{dx(t)}{dt} + Kx(t) = F(t)$$

Transfer Function

Applying the Laplace transformation we get

$$(Ms^2 + Bs + K) * X(s) = F(s)$$

Provided that, all the initial conditions are zero. Then the transfer function representation of the system is given by

$$TF = \frac{\text{Output}}{\text{Input}} = \frac{F(s)}{X(s)} = \frac{1}{(Ms^2 + Bs + K)}$$

The above system is known as a **second order system**.

The generalized notation for a second order system described above can be written as

$$Y(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} R(s)$$

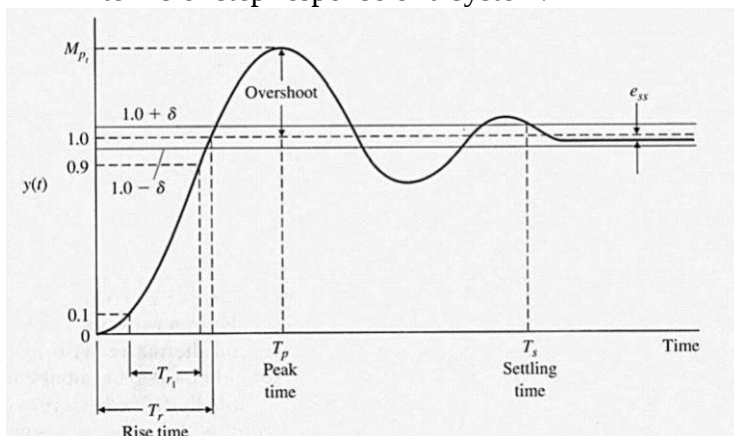
With the step input applied to the system, we obtain

$$Y(s) = \frac{\omega_n^2}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)}$$

For which the transient output, as obtained from the Laplace transform table

$$y(t) = 1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \sin(\omega_n \sqrt{1-\zeta^2} t + \cos^{-1}(\zeta))$$

- where $0 < \zeta < 1$.
- The transient response of the system changes for different values of damping ratio, ζ .
- Standard performance measures for a second order feedback system are defined in terms of step response of a system.



The performance measures could be described as follows

❑ **Rise Time 'T_r':**

measures the time from 10% to 90% of the response to the step input.

❑ **Peak Time 'T_p':**

The time for a system to respond to a step input and rise to peak response.

Overshoot

The amount by which the system output response proceeds beyond the desired response.

It is calculated as

$$P.O. = \frac{M_{pt} - f_v}{f_v} \times 100\%$$

where **M_{pt}** is the peak value of the time response, and **f_v** is the final value of the response.

Settling Time 'T_s':

The time required for the system's output to settle within a certain percentage of the input amplitude (which is usually taken as 2%). Then, settling time, T_s, is calculated as

$$T_s = \frac{4}{\zeta \omega_n}$$

Delay Time 'T_d':

It is the time required for the response to reach 50% of the final value the very first time.

OBSERVATIONS:

1. Effect of damping ratio 'ζ' on performance measures of the second order system. Find the step response of the system for values of ω_n = 1 and ζ = 0.1, 0.4, 0.7, 1.0 and 2.0.

Plot all the results in the same figure window and fill the following table.

ζ	Rise time	Peak Time	% Overshoot	Settling time	Steady state value
0.1					
0.4					
0.7					
1.0					
2.0					

CONCLUSION:

Exercise

1. Given the values of R and C, obtain the unit step response of the first order system.

ii. $R=2K\Omega$ and $C=0.01F$

iii. $R=2.5K\Omega$ and $C=0.003F$

Verify in each case that the calculated time constant ($\tau=RC$) and the one measured from the figure as 63% of the final value are same. Obtain the steady state value of the system.

2. Understand the below codes for the time specification of second order.

```
function steptimespec % find time specification for step
response of a second order system and calculates the rise
time,
%delay time, maximum overshoot, peak time and settling time
of a system
%whose damping ratio and natural frequency are known.
clc;
zeta=input('Enter the value of damping ratio ');
wn=input('Enter the value of Natural frequency ');
n=wn*wn;
d=[1 2*zeta*wn wn*wn];
disp('The transfer function is: ')
printsys(n,d);
t=0:0.02:6.0;
[y,x,t]=step(n,d,t);
plot(t,y);
grid on;
title('step response');
%to find rise time i.e. time taken for output to rise from
10% to 90%
k=1;
while y(k)<=0.1;
    k=k+1;
end
tenpercent=t(k);
while y(k)<=0.9;
    k=k+1;
end
nintypercent=t(k);
rtime=nintypercent-tenpercent;
fprintf('The rise time is: %f sec \n',rtime);
format short
% to find delay time i.e. time taken to rise to 50% of step
k=1;
while y(k)<=0.5;
    k=k+1;
end
dtime=t(k);
fprintf('The delay time is: %f sec\n', dtime);
```

```
% to find maximum overshoot
for k=1:1:300;
    if y(k+1)<=y(k);
        % to find value of k till response keeps rising
        break;
    end;
end;
Oshoot=y(k)-1;
fprintf('The overshoot is: %f sec\n', Oshoot);
% to find the peak time
tp=t(k);
fprintf('the peak time is :%f sec\n',tp)
% to find the settling time
%maximum tolerance for comnsidering output to be in steady
state taken as
%2%
tol=0.02;
for k=300:-1:2;
    if(abs(y(k)-y(300))>tol)
        break;
    end;
end;
stime=t(k);
fprintf('the settling time is :%f sec\n',stime)
```

NED University of Engineering & Technology
Department of _____ Engineering



Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
Software Menu Identification and Usage: Ability to initialise, configure and <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc.	Unable to understand and use software menu	Little ability and understanding of software menu operation, makes many mistake	Moderate ability and understanding of software menu operation, makes lesser mistakes	Reasonable understanding of software menu operation, makes no major mistakes	Demonstrates command over software menu usage with frequent use of advance menu options
Procedural Programming of given model: <u>Practice</u> procedural programming techniques, in order to code specific model	Little to no understanding of procedural programming techniques	Slight ability to use procedural programming techniques for coding given algorithm	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given model	Correctly recognises and uses procedural programming techniques with no errors but unable to run model successfully	Correctly recognises and uses procedural programming techniques with no errors and runs model successfully
Relating Theoretical Concepts, Equations and Transforms to Code: <u>Recognise</u> relation between model concepts and written code and <u>manipulate</u> the code in accordance of requirements	Completely unable to relate between model concepts and written code, unable to do manipulations	Able to recognise some relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, able to do some manipulations	Able to recognise relation between model concepts and written code, able to completely manipulate code in line with theoretical concepts
Detecting and Removing Errors: <u>Detect</u> Errors/Exceptions and in simulation and <u>manipulate</u> code to rectify the simulation	Unable to check and detect error messages and indications in software	Able to find error messages and indications in software but no understanding of detecting those errors and their types	Able to find error messages and indications in software as well as understanding of detecting some of those errors and their types	Able to find error messages in software as well as understanding of detecting all of those errors and their types	Able to find error messages in software along with the understanding to detect and rectify them

Graphical Visualisation and Comparison of model Parameters: <u>Manipulate</u> given simulation under supervision, in order to produce graphs/plots for measuring and comparing model parameters	Unable to understand and utilise visualisation or plotting features	Ability to understand and utilise visualisation and plotting features with frequent errors	Ability to understand and utilise visualisation and plotting features successfully but unable to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, partially able to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, also able to compare and analyse them
Following step-by-step procedure to complete lab work: <u>Observe, imitate and operate</u> software to complete the provided sequence of steps	Inability to recognise and perform given lab procedures	Able to recognise given lab procedures and perform them but could not follow the prescribed order of steps	Able to recognise given lab procedures and perform them by following prescribed order of steps, with frequent mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with occasional mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with no mistakes
Recording Simulation Observations: <u>Observe and copy</u> prescribed or required simulation results in accordance with lab manual instructions	Inability to recognise prescribed or required simulation measurements	Able to recognise prescribed or required simulation measurements but does not record according to given instructions	—	Able to recognise prescribed or required simulation measurements but records them incompletely	Able to recognise prescribed or required simulation measurements and records them completely, in tabular form
Discussion and Conclusion: <u>Demonstrate</u> discussion capacity on the recorded observations and draw conclusions from it, relating them to theoretical principles/concepts	Complete inability to discuss recorded observations and draw conclusions	Slight ability to discuss recorded observations and draw conclusions	Moderate ability to discuss recorded observations and draw conclusions	Reasonable ability to discuss recorded observations and draw conclusions	Full ability to discuss recorded observations and draw conclusions

LAB SESSION 08

OBJECT: Response of Control System to Ramp and Arbitrary Inputs

PROCEDURE:

TASK1:

The ramp response of the following transfer function.

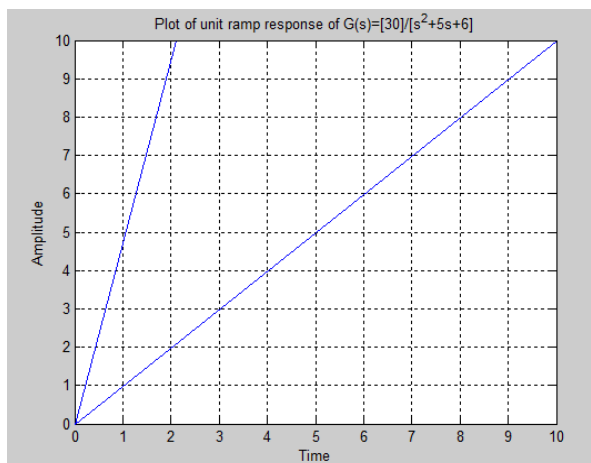
30

$$s^2 + 5s + 6$$

There is no ramp command in MATLAB. However, ramp signal is one order higher than step signal. The step input signal can be used to obtain the ramp response by dividing the transfer function by s and then evaluating it using the step command.

The following program can be used:

```
close all;
clear all;
clc;
n=[0 0 30];
d=[1 5 6];
% the ramp response can be obtained by using step command for transfer
% function divided by s. The transfer function G1(s)=G(s)/s.
n1=[0 0 0 30];
d1=[1 5 6 0];
[y,x,t]=step(n1,d1);
% To plot output y vs time t and t vs t i.e ramp signal on same graph window.
v=[0 10 0 10];
plot(t,y);
axis(v);
hold on;
plot(t,t);
grid;
title('Plot of unit ramp response of G(s)=[30]/[s^2+5s+6]');
xlabel('Time');
ylabel('Amplitude');
```



TASK2:

A closed-loop control system has a transfer function .

$$\frac{C(s)}{R(s)} = \frac{s + 5}{s^3 + 2s^2 + 3s + 5}$$

Obtain the response of the system for an input $r(t) = e^{-0.2t}$, for $t=0$ to 9 sec, in steps of 0.15 sec.

In case the input signal is not a standard signal, MATLAB command **lsim** can be used to obtain the response of the system.

The syntax of the command is

`lsim(n,d,u,t)`

Transfer function.

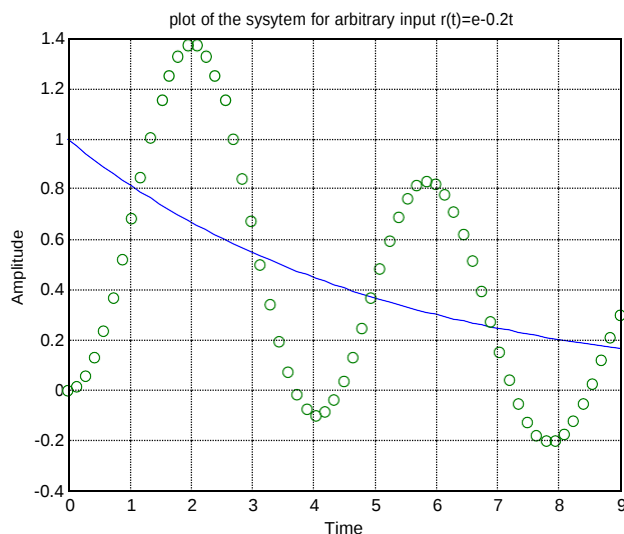
u is the arbitrary **input signal** and **t** defines time for which response of the system is required.

Another form of this command, which gives the output **y** in vector form without response plot, is;

`[y,t]=lsim(n,d,u,t)`

The following program can be used:

```
n=[1 5];
d=[1 2 3 5];
t=0:0.15:9;
r=exp(-0.2*t);
y=lsim(n,d,r,t);
plot(t,r,'-t,y','o');
grid;
title('plot of the sysytem for arbitrary input r(t)=e-0.2t');
xlabel('Time');
ylabel('Amplitude');
```



CONCLUSION:**Exercise**

1. Obtain the ramp response of the following transfer function.

$$\frac{30}{s^2 + 5s + 30}$$

2. Obtain the response of the system for an input $r(t) = \sin t + e^{-0.2t}$, for $t=0$ to 15 sec, in steps of 0.001 sec and comment on the result.

LAB SESSION 09

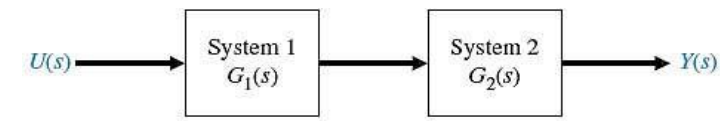
OBJECT:

To learn commands in MATLAB that would be used to reduce linear systems block diagram using series, parallel and feedback configuration.

THEORY:

▪ **Series Configuration:**

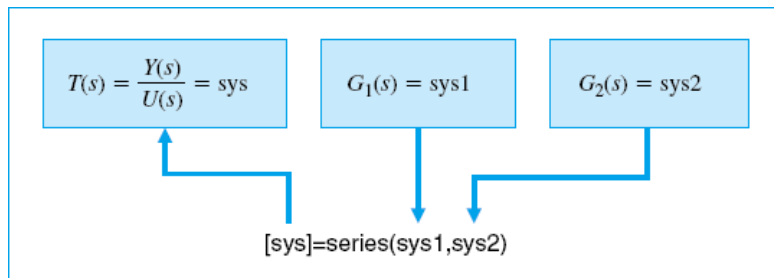
- **If the two blocks are connected as shown below then the blocks are said to be in series.**



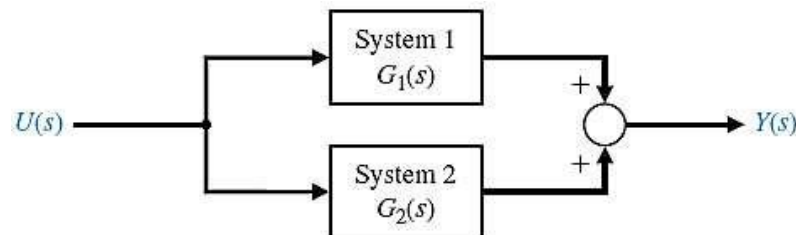
It would like **multiplying** two transfer functions.

The MATLAB command for the such configuration is “**series**”.

The series command is implemented as shown below:



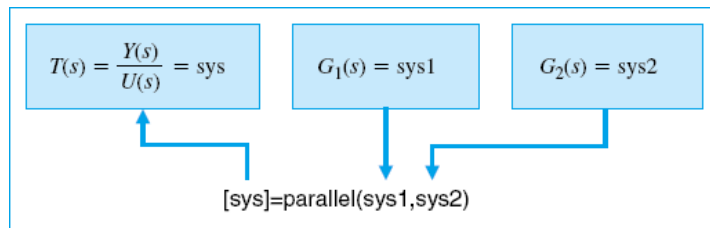
Parallel configuration: If the two blocks are connected as shown below then the blocks are said to be in parallel. It would like adding two transfer functions.



It would like **adding** two transfer functions.

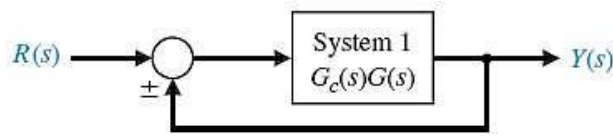
The MATLAB command for the such configuration is “**parallel**”.

The parallel command is implemented as shown below:



Feedback Configuration:

If the blocks are connected as shown below then the blocks are said to be in **feedback**.

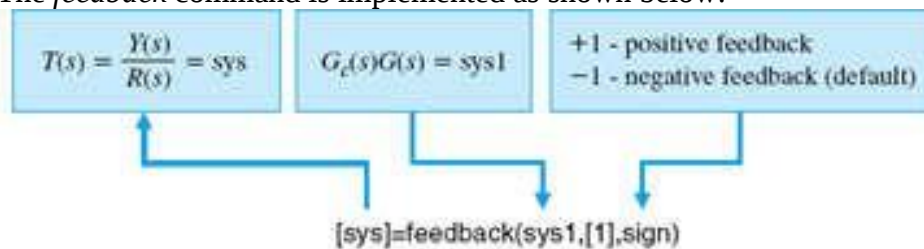


Notice that in the feedback there is *no transfer function $H(s)$* defined.

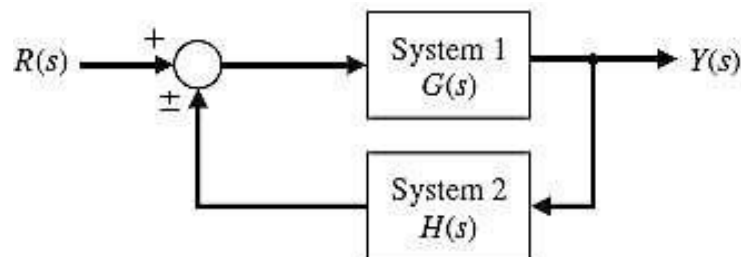
Such a system is said to be a *unity feedback system*.

The MATLAB command for implementing a feedback system is “**feedback**”.

The **feedback** command is implemented as shown below:



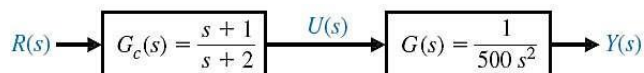
When $H(s)$ is non-unity or specified, such a system is said to be a *non-unity feedback system* as shown below:



PROCEDURE:

Task 1:

Given the transfer functions of individual blocks generate the system transfer function of the block combination.



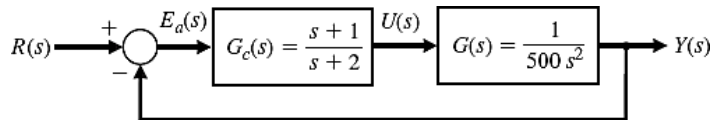
```
>> numg=[1]; deng=[500 0 0]; sysg=tf(numg,deng);
>> numh=[1 1]; denh=[1 2]; sysh=tf(numh,denh);
>> sys=series(sysg,sysh);
>> sys
```

Transfer function:

$$\frac{s+1}{500s^3+1000s^2} \leftarrow G_c(s)G(s)$$

Task 2:

Given a unity feedback system as shown in the figure, obtain the overall transfer function using MATLAB:



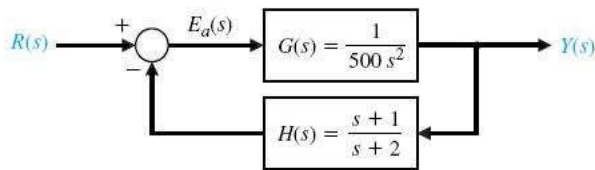
```
>> numg=[1]; deng=[500 0 0]; sys1=tf(numg,deng);
>> numc=[1 1]; denc=[1 2]; sys2=tf(numc,denc);
>> sys3=series(sys1,sys2);
>> sys=feedback(sys3,1)
```

Transfer function:

$$\frac{s+1}{500s^3+1000s^2+s+1} \leftarrow \frac{Y(s)}{R(s)} = \frac{G_c(s)G(s)}{1+G_c(s)G(s)}$$

Task 3:

Given a non-unity feedback system as shown in the figure, obtain the overall transfer function using MATLAB:

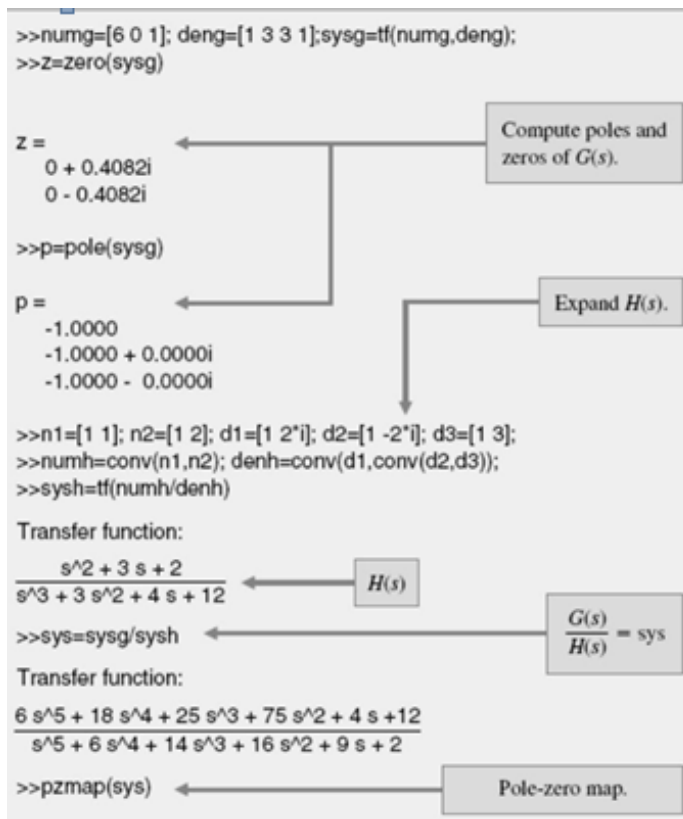


```
>> numg=[1]; deng=[500 0 0]; sys1=tf(numg,deng);
>> numh=[1 1]; denh=[1 2]; sys2=tf(numh,denh);
>> sys=feedback(sys1,sys2);
>> sys
```

Transfer function:

$$\frac{s+2}{500s^3+1000s^2+s+1} \leftarrow \frac{Y(s)}{R(s)} = \frac{G(s)}{1+G(s)H(s)}$$

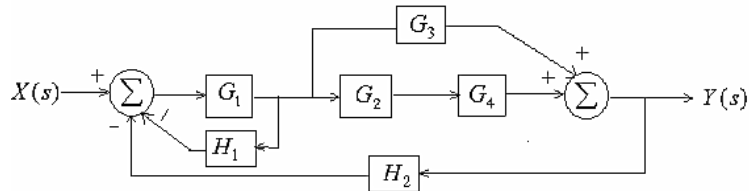
Task4: Given a system transfer function plot the location of the system zeros and poles using the MATLAB pole-zero map command



CONCLUSION

EXERCISE:

For the following multi-loop feedback system, get closed loop transfer function and the corresponding pole-zero map of the system



$$G_1 = \frac{1}{(s+10)}$$

$$G_2 = \frac{1}{(s+1)(s^2 + 1)}$$

$$G_3 = \frac{(s+2)^2}{(s+1)}$$

$$G_4 = \frac{(s+6)}{(s+1)}$$

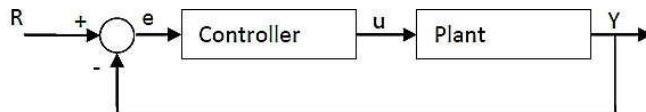
$$H_1 = \frac{(s+2)}{(s+1)}$$

$$H_2 = 1$$

LAB SESSION 10

OBJECT: Study the three term (PID) controller and its effects on the feedback loop response. Also investigate the characteristics of the each of proportional (P), the integral (I), and the derivative (D) controls and obtaining a desired response by using them.

THEORY: Consider the following unity feedback system:



Plant: A system to be controlled.

Controller: Provides excitation for the plant; Designed to control the overall system behavior.

The three-term controller: The transfer function of the PID controller looks like the following:

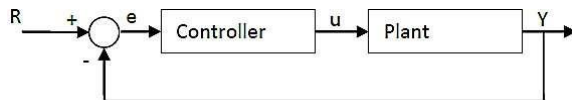
$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

KP = Proportional gain

KI = Integral gain

KD = Derivative gain

First, let's take a look at how the PID controller works in a closed-loop system using the schematic shown.



The variable (**e**) represents the **tracking error**, the difference between the desired **input value (R)** and the **actual output (Y)**.

This error signal (**e**) will be **sent** to the **PID controller**, and

The controller computes both the **derivative** and the **integral** of this error signal.

The signal (**u**) just past the controller is now equal to the **proportional gain (KP) times the magnitude of the error plus the integral gain (KI) times the integral of the error plus the derivative gain (KD) times the derivative of the error**.

$$u = K_P e(t) + K_I \int e(t) dt + K_D \frac{de(t)}{dt}$$

This **signal (u)** will be sent to the **plant**, and the new output (**Y**) will be obtained.

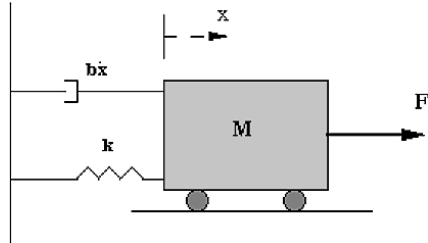
This new **output (Y)** will be **sent back** to the sensor again to find the **new error signal (e)**.

The controller takes this new error signal and computes its derivatives and its internal again.

The process goes on and on.

PROCEDURE:

For a simple mass, spring, and damper problem .



The transfer function between the displacement $X(s)$ and the input $F(s)$ then becomes:

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k}$$

Let

- $M = 1\text{kg}$
- $b = 10\text{ N.s/m}$
- $k = 20\text{ N/m}$
- $F(s) = 1$
- Plug these values into the above transfer function

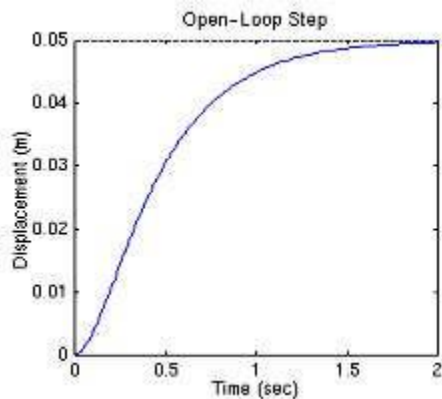
$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 10s + 20}$$

The goal of this problem is to show you how each of K_p , K_i and K_d contributes to obtain

- **Fast rise time**
- **Minimum overshoot**
- **No steady-state error**
- **Open-loop step response:**
 - Let's first view the open-loop step response.

MATLAB command window should give you the plot shown below.

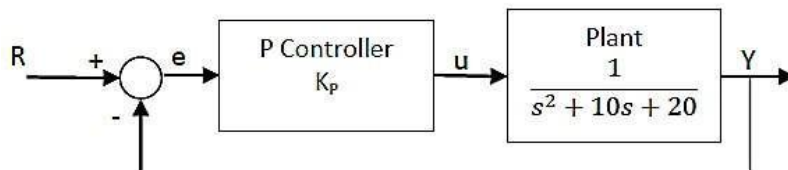
```
num=1;
den=[1 10 20];
plant=tf(num,den);
step(plant)
```



- **0.05** is the final value of the output to a **unit step input**.
- This corresponds to the **steady-state error** of **0.95**, quite large indeed.
- Furthermore, the **rise time** is about **one second**, and the **settling time** is about **1.5 seconds**.
- Let's **design a controller** that will **reduce the rise time, reduce the settling time, and eliminates the steady-state error**.

Proportional control:

- The closed-loop transfer function of the above system with a proportional controller is:



$$\frac{X(s)}{F(s)} = \frac{K_p}{s^2 + 10s + (20 + K_p)}$$

- Let the proportional gain (Kp) equal **300**:

MATLAB PROGRAM:

Kp=300;

contr=Kp;

sys_cl=feedback(contr*plant,1);

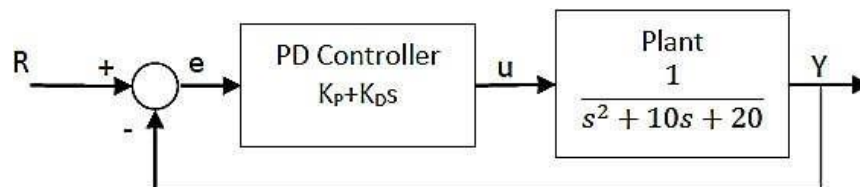
t=0:0.01:2;

step(sys_cl,t)

%by default -ve feedback

Proportional-Derivative control:

- The closed-loop transfer function of the given system with a PD controller is:



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

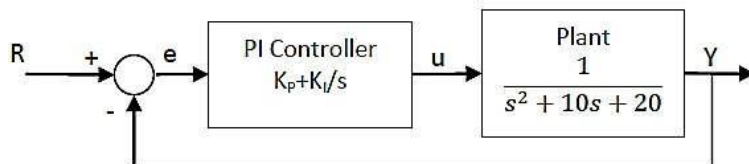
- Let K_P equal 300 as before and let K_D equal 10.

Proportional-Derivative control:

- $K_p=300$;
- $K_d=10$;
- `contr=tf([Kd Kp],1);`
- `sys_cl=feedback(contr*plant,1);`
- `t=0:0.01:2;`
- `step(sys_cl,t)`

Proportional-Integral control:

- The closed-loop transfer function of the given system with a PI controller is:



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

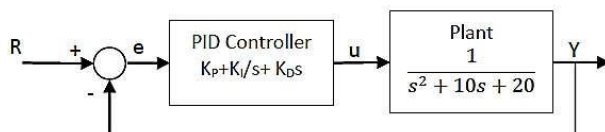
- Let K_P equal 30 and let K_I equal 70.

Proportional-Integral control:

- $K_p=30$;
- $K_i=70$;
- `contr=tf([Kp Ki],[1 0]);`
- `sys_cl=feedback(contr*plant,1);`
- `t=0:0.01:2;`
- `step(sys_cl,t)`

Proportional-Integral-Derivative control:

- Now, let's take a look at a PID controller. The closed-loop transfer function of the given system with a PID controller is:



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

After several trial and error runs, the gains $K_p=350$, $K_i=300$, and $K_d=50$ provided the desired response.

Proportional-Integral-Derivative control:

- $K_p=350$;
- $K_i=300$;
- $K_d=50$;
- $\text{contr}=\text{tf}([K_d \ K_p \ K_i],[1 \ 0])$;
- $\text{sys_cl}=\text{feedback}(\text{contr}*\text{plant},1)$;
- $t=0:0.01:2$;
- $\text{step}(\text{sys_cl},t)$

OBSERVATIONS:

<u>CL Response</u>	<u>Rise time</u>	<u>Overshoot time</u>	<u>Settling time</u>	<u>S-S error</u>
K_p				
K_i				
K_D				

RESULTS:

Plots of all the simulated systems with their rise time, settling time and final value

NED University of Engineering & Technology
Department of _____ Engineering



Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
Software Menu Identification and Usage: Ability to initialise, configure and <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc.	Unable to understand and use software menu	Little ability and understanding of software menu operation, makes many mistake	Moderate ability and understanding of software menu operation, makes lesser mistakes	Reasonable understanding of software menu operation, makes no major mistakes	Demonstrates command over software menu usage with frequent use of advance menu options
Procedural Programming of given model: <u>Practice</u> procedural programming techniques, in order to code specific model	Little to no understanding of procedural programming techniques	Slight ability to use procedural programming techniques for coding given algorithm	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given model	Correctly recognises and uses procedural programming techniques with no errors but unable to run model successfully	Correctly recognises and uses procedural programming techniques with no errors and runs model successfully
Relating Theoretical Concepts, Equations and Transforms to Code: <u>Recognise</u> relation between model concepts and written code and <u>manipulate</u> the code in accordance of requirements	Completely unable to relate between model concepts and written code, unable to do manipulations	Able to recognise some relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, able to do some manipulations	Able to recognise relation between model concepts and written code, able to completely manipulate code in line with theoretical concepts
Detecting and Removing Errors: <u>Detect</u> Errors/Exceptions and in simulation and <u>manipulate</u> code to rectify the simulation	Unable to check and detect error messages and indications in software	Able to find error messages and indications in software but no understanding of detecting those errors and their types	Able to find error messages and indications in software as well as understanding of detecting some of those errors and their types	Able to find error messages in software as well as understanding of detecting all of those errors and their types	Able to find error messages in software along with the understanding to detect and rectify them

Graphical Visualisation and Comparison of model Parameters: <u>Manipulate</u> given simulation under supervision, in order to produce graphs/plots for measuring and comparing model parameters	Unable to understand and utilise visualisation or plotting features	Ability to understand and utilise visualisation and plotting features with frequent errors	Ability to understand and utilise visualisation and plotting features successfully but unable to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, partially able to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, also able to compare and analyse them
Following step-by-step procedure to complete lab work: <u>Observe, imitate and operate</u> software to complete the provided sequence of steps	Inability to recognise and perform given lab procedures	Able to recognise given lab procedures and perform them but could not follow the prescribed order of steps	Able to recognise given lab procedures and perform them by following prescribed order of steps, with frequent mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with occasional mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with no mistakes
Recording Simulation Observations: <u>Observe and copy</u> prescribed or required simulation results in accordance with lab manual instructions	Inability to recognise prescribed or required simulation measurements	Able to recognise prescribed or required simulation measurements but does not record according to given instructions	—	Able to recognise prescribed or required simulation measurements but records them incompletely	Able to recognise prescribed or required simulation measurements and records them completely, in tabular form
Discussion and Conclusion: <u>Demonstrate</u> discussion capacity on the recorded observations and draw conclusions from it, relating them to theoretical principles/concepts	Complete inability to discuss recorded observations and draw conclusions	Slight ability to discuss recorded observations and draw conclusions	Moderate ability to discuss recorded observations and draw conclusions	Reasonable ability to discuss recorded observations and draw conclusions	Full ability to discuss recorded observations and draw conclusions

Lab Session 11 (Open Ended Lab – I)

Objective:

To develop a mathematical model of a DC buck converter using first principle approach based on underlying physics of the circuit and verify it using MATLAB Simulink.

Task:

Evaluate the transfer function, $\frac{V_o(s)}{d(s)}$ using first principle approach considering continuous conduction mode of inductor current. Also, verify the transfer function using MATLAB Simulink, considering the following system parameters.

[Attach the screenshot of Simulink Model along with the plot of V_o . Using the plot find ζ and peak time and use these parameters to estimate transfer function.]

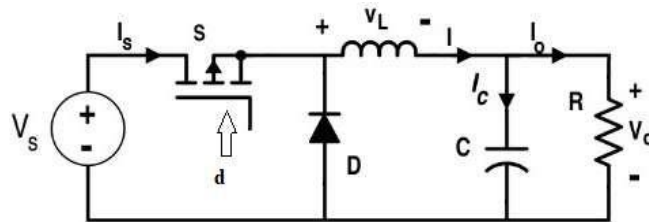
$$R = 1 \Omega$$

$$L = 33 \mu H$$

$$C = 47 \mu F$$

$$d = 0.5 \text{ (50\% duty cycle), Switching frequency} = 30 \text{ KHz}$$

$$V_s = 12 \text{ V}$$



Cover Page for Each PBL/OEL

Course Code:	EE-374
Course Name:	Feedback Control Systems for EL and TC only
Semester:	Spring / Fall
Year:	20__
Section:	
Batch:	
Lab Instructor name:	
Submission deadline:	

PBL or OEL Statement: To develop a mathematical model of a DC buck converter using first principle approach based on underlying physics of the circuit and verify it using MATLAB Simulink.

Deliverables:

Write the report containing all calculations by hand and simulation on Matlab/Simulink. Include all code and waveforms

Methodology:

[Attach the screenshot of Simulink Model along with the plot of V_o . Using the plot find ζ and peak time and use these parameters to estimate transfer function.]

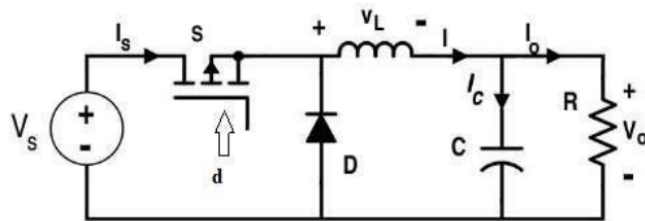
$$R = 1 \, \Omega$$

$$L = 33 \, \mu H$$

$$C = 47 \, \mu F$$

$$d = 0.5 \text{ (50\% duty cycle), Switching frequency} = 30 \text{ KHz}$$

$$V_s = 12 \text{ V}$$



Guidelines: The report should be maximum 5 pages long which should include figures, calculations, simulation results and waveforms Attach these two pages on top of the report.

Rubrics: Standard software rubrics as defined for EE-374

NED University of Engineering & Technology
Department of _____ Engineering



Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
Software Menu Identification and Usage: Ability to initialise, configure and <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc.	Unable to understand and use software menu	Little ability and understanding of software menu operation, makes many mistake	Moderate ability and understanding of software menu operation, makes lesser mistakes	Reasonable understanding of software menu operation, makes no major mistakes	Demonstrates command over software menu usage with frequent use of advance menu options
Procedural Programming of given model: <u>Practice</u> procedural programming techniques, in order to code specific model	Little to no understanding of procedural programming techniques	Slight ability to use procedural programming techniques for coding given algorithm	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given model	Correctly recognises and uses procedural programming techniques with no errors but unable to run model successfully	Correctly recognises and uses procedural programming techniques with no errors and runs model successfully
Relating Theoretical Concepts, Equations and Transforms to Code: <u>Recognise</u> relation between model concepts and written code and <u>manipulate</u> the code in accordance of requirements	Completely unable to relate between model concepts and written code, unable to do manipulations	Able to recognise some relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, able to do some manipulations	Able to recognise relation between model concepts and written code, able to completely manipulate code in line with theoretical concepts
Detecting and Removing Errors: <u>Detect</u> Errors/Exceptions and in simulation and <u>manipulate</u> code to rectify the simulation	Unable to check and detect error messages and indications in software	Able to find error messages and indications in software but no understanding of detecting those errors and their types	Able to find error messages and indications in software as well as understanding of detecting some of those errors and their types	Able to find error messages in software as well as understanding of detecting all of those errors and their types	Able to find error messages in software along with the understanding to detect and rectify them

Graphical Visualisation and Comparison of model Parameters: <u>Manipulate</u> given simulation under supervision, in order to produce graphs/plots for measuring and comparing model parameters	Unable to understand and utilise visualisation or plotting features	Ability to understand and utilise visualisation and plotting features with frequent errors	Ability to understand and utilise visualisation and plotting features successfully but unable to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, partially able to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, also able to compare and analyse them
Following step-by-step procedure to complete lab work: <u>Observe, imitate and operate</u> software to complete the provided sequence of steps	Inability to recognise and perform given lab procedures	Able to recognise given lab procedures and perform them but could not follow the prescribed order of steps	Able to recognise given lab procedures and perform them by following prescribed order of steps, with frequent mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with occasional mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with no mistakes
Recording Simulation Observations: <u>Observe and copy</u> prescribed or required simulation results in accordance with lab manual instructions	Inability to recognise prescribed or required simulation measurements	Able to recognise prescribed or required simulation measurements but does not record according to given instructions	—	Able to recognise prescribed or required simulation measurements but records them incompletely	Able to recognise prescribed or required simulation measurements and records them completely, in tabular form
Discussion and Conclusion: <u>Demonstrate</u> discussion capacity on the recorded observations and draw conclusions from it, relating them to theoretical principles/concepts	Complete inability to discuss recorded observations and draw conclusions	Slight ability to discuss recorded observations and draw conclusions	Moderate ability to discuss recorded observations and draw conclusions	Reasonable ability to discuss recorded observations and draw conclusions	Full ability to discuss recorded observations and draw conclusions

Lab Session 12 (Open Ended Lab – II)

Objective:

To identify experimentally the Transfer Function of a Buck Converter using the step response.

Task:

Identify the transfer function, $\frac{V_o(s)}{d(s)}$ experimentally by recording the step response of the output voltage V_o of the following buck converter either by using Oscilloscope or by using a Microcontroller (For, e.g. Arduino DUE) with following system parameters.

[Attach the screenshot of the plot of V_o . Using plot find ζ and peak time and use these parameters to estimate transfer function.]

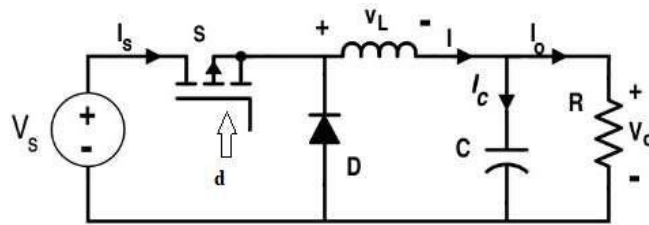
$$R = 1 \Omega$$

$$L = 0.5 \text{ mH}$$

$$C = 940 \mu\text{F}$$

$$d = 0.5 \text{ (50\% duty cycle), Switching frequency} = 10 \text{ KHz}$$

$$V_s = 5 \text{ V}$$



Cover Page for Each PBL/OEL

Course Code:	EE-374
Course Name:	Feedback Control Systems for EL and TC only
Semester:	Spring / Fall
Year:	20__
Section:	
Batch:	
Lab Instructor name:	
Submission deadline:	

PBL or OEL Statement: To identify experimentally the Transfer Function of a Buck Converter using the step response.

Deliverables:

Write the report containing all calculations by hand and simulation on Matlab/Simulink/Arduino and pictures of oscilloscope output. Include all code and waveforms

Methodology:

Identify the transfer function, $\frac{V_o(s)}{d(s)}$ experimentally by recording the step response of the output voltage V_o of the following buck converter either by using Oscilloscope or by using a Microcontroller (For, e.g. Arduino DUE) with following system parameters.

[Attach the screenshot of the plot of V_o . Using plot find ζ and peak time and use these parameters to estimate transfer function.]

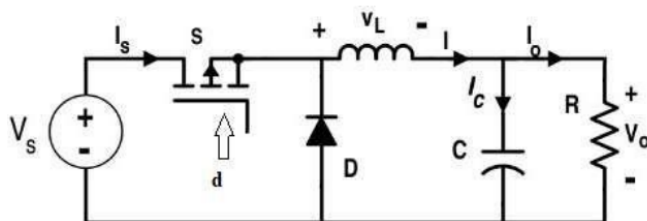
$$R = 1 \Omega$$

$$L = 0.5 \text{ mH}$$

$$C = 940 \mu\text{F}$$

$$d = 0.5 \text{ (50\% duty cycle), Switching frequency} = 10 \text{ KHz}$$

$$V_s = 5 \text{ V}$$



Guidelines: The report should be maximum 5 pages long which should include figures, calculations, simulation results and waveforms. Attach these two pages on top of the report.

Rubrics: Standard software rubrics as defined for EE-374

NED University of Engineering & Technology
Department of _____ Engineering



Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
Software Menu Identification and Usage: Ability to initialise, configure and <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc.	Unable to understand and use software menu	Little ability and understanding of software menu operation, makes many mistake	Moderate ability and understanding of software menu operation, makes lesser mistakes	Reasonable understanding of software menu operation, makes no major mistakes	Demonstrates command over software menu usage with frequent use of advance menu options
Procedural Programming of given model: <u>Practice</u> procedural programming techniques, in order to code specific model	Little to no understanding of procedural programming techniques	Slight ability to use procedural programming techniques for coding given algorithm	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given model	Correctly recognises and uses procedural programming techniques with no errors but unable to run model successfully	Correctly recognises and uses procedural programming techniques with no errors and runs model successfully
Relating Theoretical Concepts, Equations and Transforms to Code: <u>Recognise</u> relation between model concepts and written code and <u>manipulate</u> the code in accordance of requirements	Completely unable to relate between model concepts and written code, unable to do manipulations	Able to recognise some relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, able to do some manipulations	Able to recognise relation between model concepts and written code, able to completely manipulate code in line with theoretical concepts
Detecting and Removing Errors: <u>Detect</u> Errors/Exceptions and in simulation and <u>manipulate</u> code to rectify the simulation	Unable to check and detect error messages and indications in software	Able to find error messages and indications in software but no understanding of detecting those errors and their types	Able to find error messages and indications in software as well as understanding of detecting some of those errors and their types	Able to find error messages in software as well as understanding of detecting all of those errors and their types	Able to find error messages in software along with the understanding to detect and rectify them

Graphical Visualisation and Comparison of model Parameters: <u>Manipulate</u> given simulation under supervision, in order to produce graphs/plots for measuring and comparing model parameters	Unable to understand and utilise visualisation or plotting features	Ability to understand and utilise visualisation and plotting features with frequent errors	Ability to understand and utilise visualisation and plotting features successfully but unable to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, partially able to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, also able to compare and analyse them
Following step-by-step procedure to complete lab work: <u>Observe, imitate and operate</u> software to complete the provided sequence of steps	Inability to recognise and perform given lab procedures	Able to recognise given lab procedures and perform them but could not follow the prescribed order of steps	Able to recognise given lab procedures and perform them by following prescribed order of steps, with frequent mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with occasional mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with no mistakes
Recording Simulation Observations: <u>Observe and copy</u> prescribed or required simulation results in accordance with lab manual instructions	Inability to recognise prescribed or required simulation measurements	Able to recognise prescribed or required simulation measurements but does not record according to given instructions	—	Able to recognise prescribed or required simulation measurements but records them incompletely	Able to recognise prescribed or required simulation measurements and records them completely, in tabular form
Discussion and Conclusion: <u>Demonstrate</u> discussion capacity on the recorded observations and draw conclusions from it, relating them to theoretical principles/concepts	Complete inability to discuss recorded observations and draw conclusions	Slight ability to discuss recorded observations and draw conclusions	Moderate ability to discuss recorded observations and draw conclusions	Reasonable ability to discuss recorded observations and draw conclusions	Full ability to discuss recorded observations and draw conclusions

NED University of Engineering & Technology
Department of _____ Engineering



Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
Software Menu Identification and Usage: Ability to initialise, configure and <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc.	Unable to understand and use software menu	Little ability and understanding of software menu operation, makes many mistake	Moderate ability and understanding of software menu operation, makes lesser mistakes	Reasonable understanding of software menu operation, makes no major mistakes	Demonstrates command over software menu usage with frequent use of advance menu options
Procedural Programming of given model: <u>Practice</u> procedural programming techniques, in order to code specific model	Little to no understanding of procedural programming techniques	Slight ability to use procedural programming techniques for coding given algorithm	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given model	Correctly recognises and uses procedural programming techniques with no errors but unable to run model successfully	Correctly recognises and uses procedural programming techniques with no errors and runs model successfully
Relating Theoretical Concepts, Equations and Transforms to Code: <u>Recognise</u> relation between model concepts and written code and <u>manipulate</u> the code in accordance of requirements	Completely unable to relate between model concepts and written code, unable to do manipulations	Able to recognise some relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, unable to do manipulations	Able to recognise relation between model concepts and written code, able to do some manipulations	Able to recognise relation between model concepts and written code, able to completely manipulate code in line with theoretical concepts
Detecting and Removing Errors: <u>Detect</u> Errors/Exceptions and in simulation and <u>manipulate</u> code to rectify the simulation	Unable to check and detect error messages and indications in software	Able to find error messages and indications in software but no understanding of detecting those errors and their types	Able to find error messages and indications in software as well as understanding of detecting some of those errors and their types	Able to find error messages in software as well as understanding of detecting all of those errors and their types	Able to find error messages in software along with the understanding to detect and rectify them

Graphical Visualisation and Comparison of model Parameters: <u>Manipulate</u> given simulation under supervision, in order to produce graphs/plots for measuring and comparing model parameters	Unable to understand and utilise visualisation or plotting features	Ability to understand and utilise visualisation and plotting features with frequent errors	Ability to understand and utilise visualisation and plotting features successfully but unable to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, partially able to compare and analyse them	Ability to understand and utilise visualisation and plotting features successfully, also able to compare and analyse them
Following step-by-step procedure to complete lab work: <u>Observe, imitate and operate</u> software to complete the provided sequence of steps	Inability to recognise and perform given lab procedures	Able to recognise given lab procedures and perform them but could not follow the prescribed order of steps	Able to recognise given lab procedures and perform them by following prescribed order of steps, with frequent mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with occasional mistakes	Able to recognise given lab procedures and perform them by following prescribed order of steps, with no mistakes
Recording Simulation Observations: <u>Observe and copy</u> prescribed or required simulation results in accordance with lab manual instructions	Inability to recognise prescribed or required simulation measurements	Able to recognise prescribed or required simulation measurements but does not record according to given instructions	—	Able to recognise prescribed or required simulation measurements but records them incompletely	Able to recognise prescribed or required simulation measurements and records them completely, in tabular form
Discussion and Conclusion: <u>Demonstrate</u> discussion capacity on the recorded observations and draw conclusions from it, relating them to theoretical principles/concepts	Complete inability to discuss recorded observations and draw conclusions	Slight ability to discuss recorded observations and draw conclusions	Moderate ability to discuss recorded observations and draw conclusions	Reasonable ability to discuss recorded observations and draw conclusions	Full ability to discuss recorded observations and draw conclusions