1. PREAMBLE:

Control Systems simulation Lab consists of multiple workstations, each equipped with an oscilloscope, digital multi-meter, PID trainers, control system trainers and stand alone inverted-pendulum, ball and beam control, magnetic-levitation trainers. This lab also covers the industrial implementation of advanced control systems via different computer tools such as MATLAB and Simulink.

2 OBJECTIVE & RELEVANCE

The aim of this Control system laboratory is to provide sound knowledge in the basic concepts of linear control theory and design of control system, to understand the methods of representation of systems and getting their transfer function models, to provide adequate knowledge in the time response of systems and steady state error analysis, to give basic knowledge is obtaining the open loop and closed–loop frequency responses of systems and to understand the concept of stability of control system and methods of stability analysis. It helps the students to study the compensation design for a control system. This lab consist of DC,AC servomotor, synchros, DC position control, PID controller kit with temperature control, lead lag compensator kit, PLC kit, Stepper ,process control simulator

OUTCOME

- After the completion of this course student able solve the control system problems by using the programs through MATLAB.
- Determination of transfer function useful to design the systems.

Introducing of MATLAB in control systems solutions

3 List of Experiments:

1. Pspice simulation of op-amp based integrator & differentiator circuits
2. Simulation of saw tooth wave and sine wave using matlab
3. Simulation of triangular wave and ramp wave using matlab
4. Unity and non unity feedback system using matlab
5. Block diagram reduction technique using matlab
6. Simulation of p, pd, pi, pid controller
7. Simulation of dc motor characteristics using matlab
8. Simulation of poles and zeros of a transfer function
9. State model for classical transfer function &vice versa using matlab
10. Transfer function analysis of 3rd order using simulink
11. Stability analysis using bode plote using matlab
12. Stability analysis using root locus using matlab
13. Stability analysis using nyquist plot using matlab

4. Text and Reference Books

TEXT BOOKS :

T1 : B. C. Kuo ***"Automatic Control Systems"*** 8th edition– by 2003– John wiley and son's.,

T2 : I. J. Nagrath and M. Gopal, ***"Control Systems Engineering"*** New Age International (P) Limited, Publishers, 2nd edition.

REFERENCE BOOKS *:*

R1 : Katsuhiko Ogata ***"Modern Control Engineering"*** Prentice Hall of India Pvt. Ltd., 3rd edition, 1998.

R2 : N.K.Sinha, ***"Control Systems"*** New Age International (P) Limited Publishers, 3rd Edition,1998.

R3 : NISE ***"Control Systems Engg."*** 5th Edition – John wiley

R4 : Narciso F. Macia George J. Thaler, ***" Modeling & Control Of Dynamic Systems"*** Thomson Publishers

5. SESSION PLAN

S.no	Name of the experiment	Week of Experiment
1.	Pspice simulation of op-amp based integrator & differentiator circuits	Week #1
2.	Simulation of saw tooth wave and sine wave using matlab	Week #2
3.	Simulation of triangular wave and ramp wave using matlab	Week #3
4.	Unity and non unity feedback system using matlab	Week #4
5.	Block diagram reduction technique using matlab	Week #5
6.	Simulation of p, pd, pi, pid controller	Week #6
7.	Simulation of dc motor characteristics using matlab	Week #7
8.	Simulation of poles and zeros of a transfer function	Week #8
9.	State model for classical transfer function &vice versa using matlab	Week #9
10.	Transfer function analysis of 3rd order using simulink	Week #10
11.	Stability analysis using bode plote using matlab	Week #11
12.	Stability analysis using root locus using matlab	Week #12
13.	Stability analysis using nyquist plot using matlab	Week #13

6 Experiment write up

6.1.PSPICE SIMULATION OF OP-AMP BASED INTEGRATOR & DIFFERENTIATOR CIRCUITS

AIM: To simulate the integrator & differentiator circuits by using PSPICE.

SOFTWARE REQUIRED: **PSPICE – P**ersonal Computer **S**imulated **P**rogram with **I**ntegrated **C**ircuit **E**mphasis.

a) Simulation of INTEGRATOR CIRCUIT Using PSPICE:

SYNTAX USED:

S.NO	TYPE OF SOURCE	REPRESENTATION OF SOURCE	DECLARATION FORMAT
1.	STEP RESPONSE	PWL	STEP (Time at a Point) (Voltage at a Point)
2.	TRANSIENT ANALYSIS	.TRAN	.TRAN TStep Tstop [TStart TMax] [UIC]
3.	PROBE STATEMENT	.PROBE	It is a wave form analyzer
4.	PLOT STATEMENT	.PLOT	.PLOT (Output Variables) {(Lower limit Value), (Upper Limit Value)}

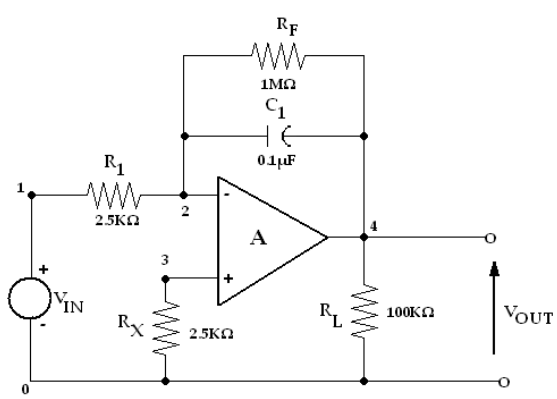
DATA REQUIRED FOR DRAWING CIRCUIT DIAGRAM:

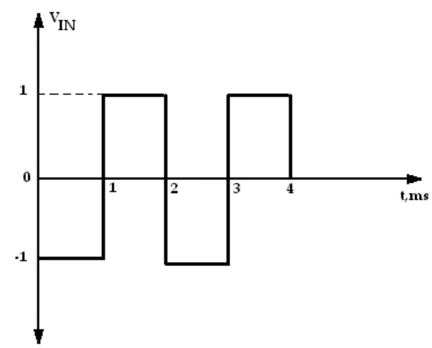
Draw an Integrator Circuit Using **OP-AMP**. For this circuit apply the **Step Response** whose values are **T_1=0, V_1=0V, T_2=1NS, V_2=-1V, T_3=1mS, V_3=-1V,**

T_4=1.0001mS, V_4=1V, T_5=2mS, V_5=1V, T_6=2.0001mS, V_6=-1V, T_7=3mS, V_7=-1V,

T_8=3.0001mS, V_8=1V, T_9=4mS, V_9=1V respectively. It consists of resistances and Capacitors whose Values are **R_1=2.5KΩ, R_f=1MΩ, R_x=2.5KΩ** and **R_L=100KΩ** and the capacitance value as **0.1μF.** Plot the transient response from **0** to **4mseconds** with an increment of **50μsecond**.

CIRCUIT DIAGRAM:





PROGRAM:

*

VIN1 1 0 PWL(0 0 1NS -1V 1MS -1V 1.0001MS 1V 2MS 1V +2.0001MS -1V 3MS -1V 3.0001MS 1V 4MS 1V)

R1 1 2 2.5K

RF 2 4 1MEG

RX 3 0 2.5K

RL 4 0 100K

C1 2 4 0.1UF

XA1 2 3 4 0 OPAMP

```
.SUBCKT OPAMP 1 2 7 4

RI 1 2 2MEG

GB 4 3 1 2 0.1M

R1 3 4 10K

C1 3 4 1.5619UF

EA 4 5 3 4 2E+5

RO 5 7 75

.ENDS OPAMP

.TRAN 50US 4MS

.PLOT TRAN V(4) V(1)

.PROBE

.END
```

OUTPUT:

RESULT: Analysis of Integrator circuit has been successfully completed.

Simulation of DIFFERENTIATOR CIRCUIT Using PSPICE:

SYNTAX USED:

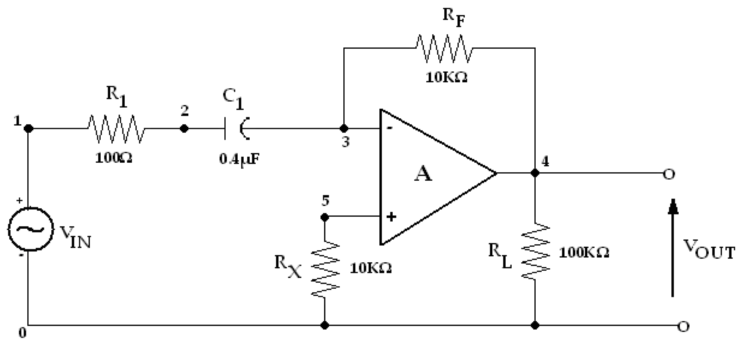
S.NO	TYPE OF SOURCE	REPRESENTATION OF SOURCE	DECLARATION FORMAT
1.	STEP RESPONSE	PWL	STEP (Time at a Point) (Voltage at a Point)
2.	TRANSIENT ANALYSIS	.TRAN	.TRAN TStep Tstop [TStart TMax] [UIC]
3.	PROBE STATEMENT	.PROBE	It is a wave form analyzer
4.	PLOT STATEMENT	.PLOT	.PLOT (Output Variables) {(Lower limit Value), (Upper Limit Value)}

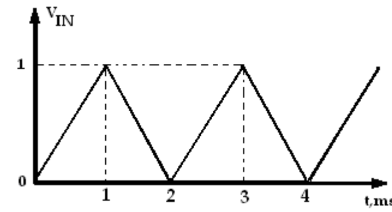
DATA REQUIRED FOR DRAWING CIRCUIT DIAGRAM:

Draw a Differentiator Circuit Using **OP-AMP**. For this circuit apply the **Step Response** whose values are **T_1=0, V_1=0V, T_2=1MS, V_2=1V, T_3=2mS, V_3=0V,**

T_4=3mS, V_4=1V, T_5=4mS, V_5=0V respectively. It consists of resistances and Capacitors whose Values are **R_1=100Ω, R_f=10KΩ, R_x=10KΩ,R_L=100KΩ** and the capacitance value as **0.4µF.** Plot the transient response from **0** to **4mseconds** with an increment of **50µsecond**.

CIRCUIT DIAGRAM:





PROGRAM:

```
*
VIN1 1 0 PWL(0 0 1MS 1V 2MS 0V 3MS 1V 4MS 0V)
R1 1 2 100
RF 3 4 10K
RX 5 0 10K
RL 4 0 100K
C1 2 3 0.4UF
XA1 3 5 4 0 OPAMP
.SUBCKT OPAMP 1 2 7 4
RI 1 2 2MEG
GB 4 3 1 2 0.1M
R1 3 4 10K
C1 3 4 1.5619UF
EA 4 5 3 4 2E+5
RO 5 7 75
.ENDS OPAMP
```

.TRAN 10US 4MS

.PLOT TRAN V(4) V(1)

.PROBE

.END

OUTPUT:

RESULT: Analysis of Integrator circuit has been successfully completed.

6.2. Simulation of saw tooth wave and sine wave using MATLAB

AIM: To simulate saw tooth wave and sine wave by using MATLAB.

SOFTWARE REQUIRED: **MATLAB – P**ersonal Computer with MATLAB

a) Simulation of saw tooth wave by using MATLAB.

PROGRAM:

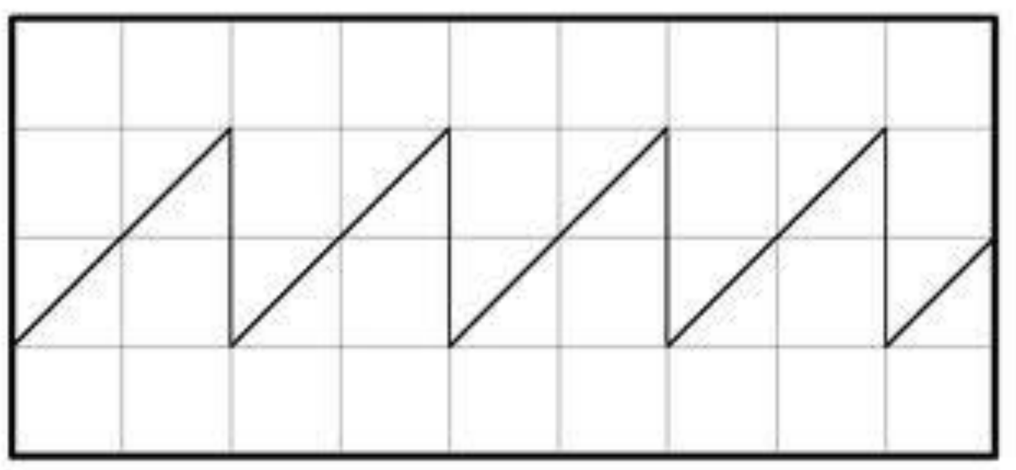
```
clear all;
close all;
clc;
n=input('enter the number of cycles');
t1=0:25;
t=[];
for i=1:n,t=[t,t1];
end;
subplot(2,1,1);
plot(t);
subplot(2,1,2);
```

```
stem(t);
```

OUTPUT:

Enter the no of cycles

MODEL GRAPH:



RESULT: Analysis saw tooth wave has been successfully completed.

B) Simulation of SINE wave by using MATLAB.

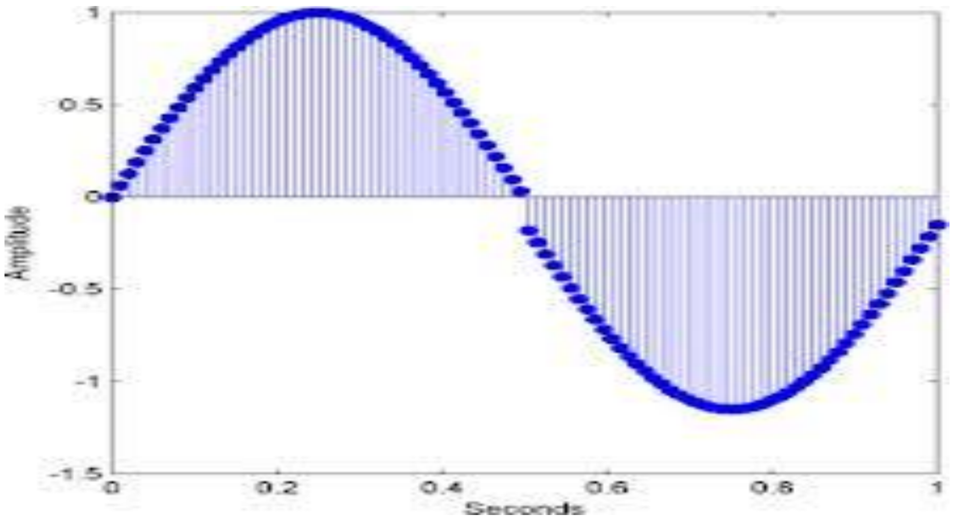
PROGRAM:

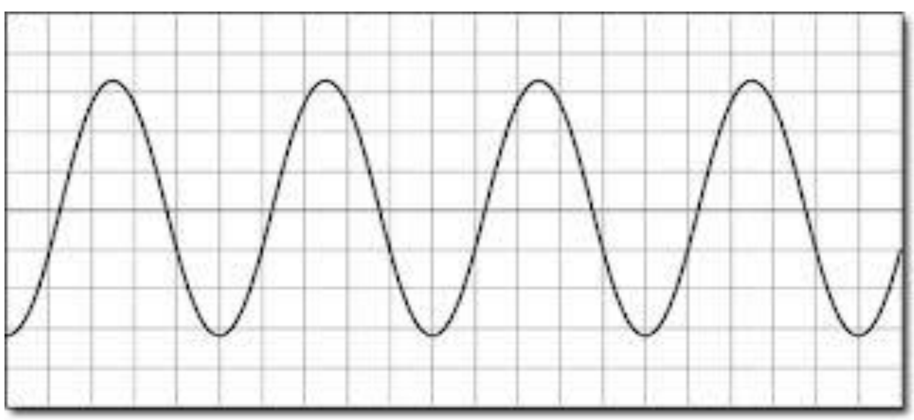
```
clear all;

close all;

clc;

n=input('enter the number of cycles');

t1=0:0.05:n;

x=sin(2*pi*t1);

subplot(1,2,1);

plot(t1,x);

subplot(1,2,2);

stem(t1);
```

OUTPUT:

Enter the no of cycles

MODEL GRAPH:





RESULT: Analysis of sine wave has been successfully completed.

6.3. Simulation of Triangular wave and Ramp wave using MATLAB

AIM: To simulate triangular wave and ramp wave by using MATLAB.

SOFTWARE REQUIRED: **MATLAB – P**ersonal Computer with MATLAB

a) Simulation of triangular wave by using MATLAB.

PROGRAM:

```
clear all;
close all;
clc;
n=input('enter the number of cycles');
m=input('enter the number of period');
t1=0:0.1:m/2;
t2=m/2:-0.01:0;
t=[];
for i=1:n;
   t=[t,t1,t2];
end;
subplot(1,2,1);
plot(t);
```

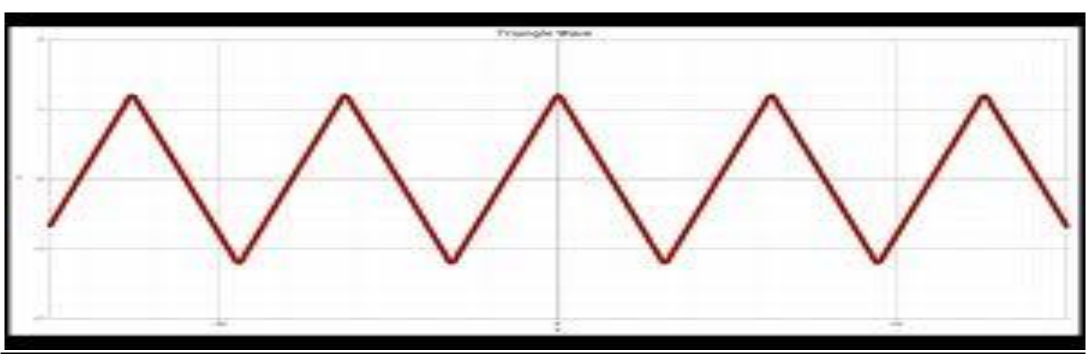
```
subplot(1,2,2);

stem(t);
```

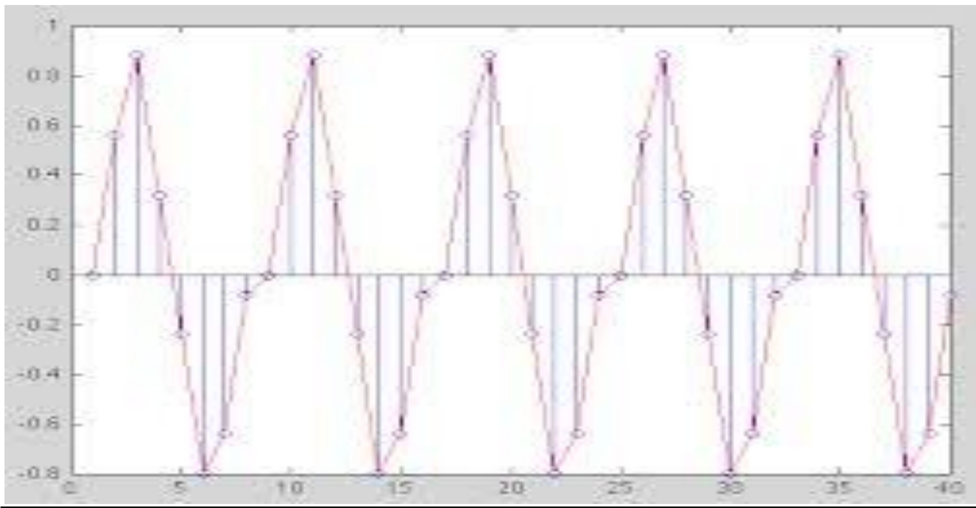
OUTPUT:

Enter the no of cycles

Enter the no of periods

MODEL GRAPH:





RESULT: Analysis triangular wave has been successfully completed.

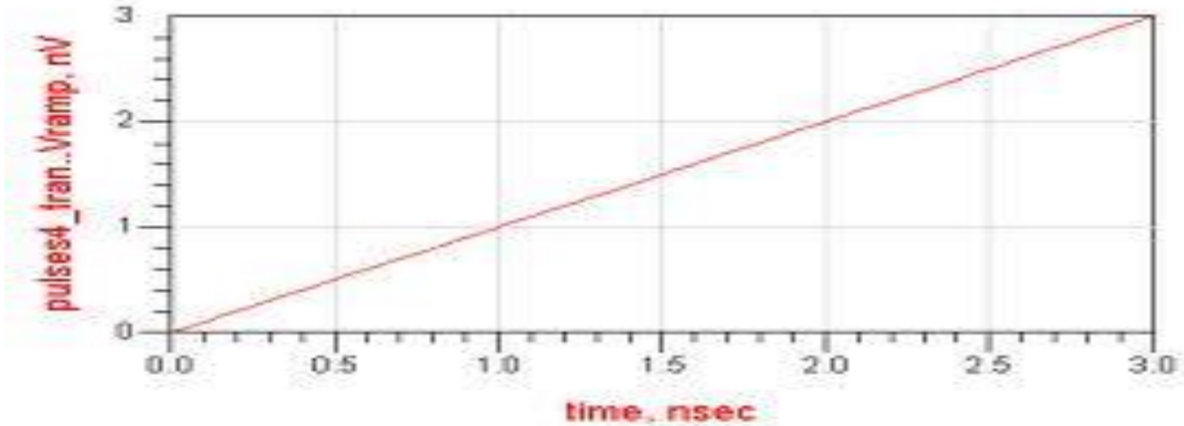
B) Simulation of ramp wave by using MATLAB.

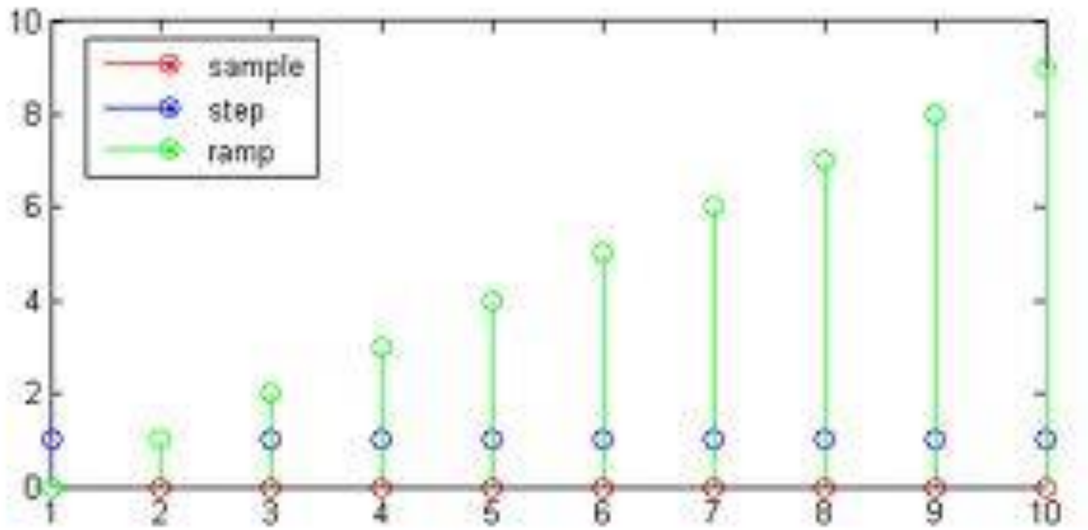
PROGRAM:

```
clear all;
close all;
clc;
t=0:1:210;
y=t;
subplot(1,2,1);
plot(t,y);
subplot(1,2,2);
stem(t,y)
```

OUTPUT:

MODEL GRAPH:





RESULT: Analysis of ramp wave has been successfully completed

6.4. Unity and Non Unity Feedback System using MATLAB

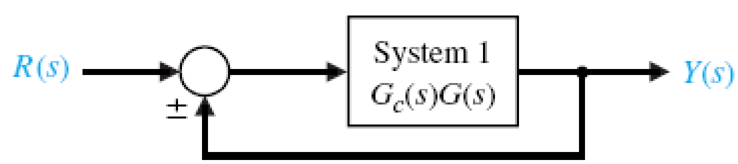
Aim: To simulation of unity and non unity feedback transfer function using MATLAB.

SOFTWARE REQUIRED: **MATLAB – P**ersonal Computer with MATLAB

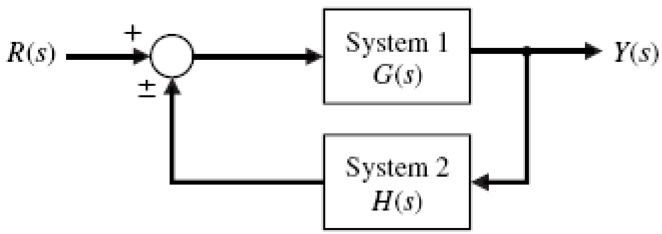
a) **simulation of unity feedback system.**

Theory:

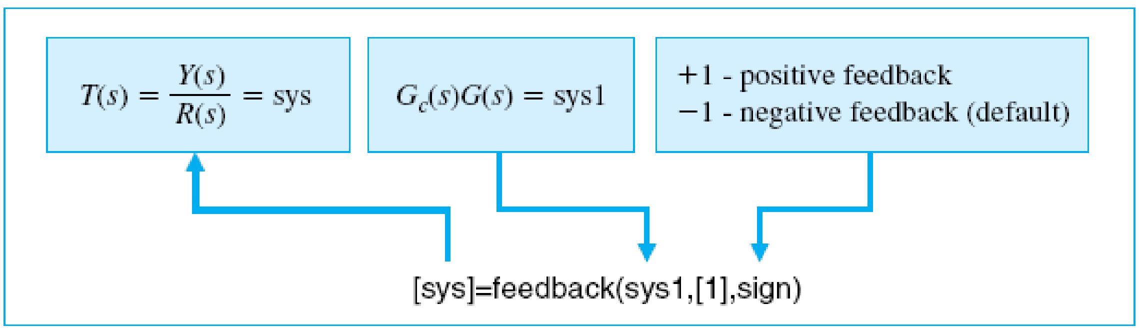
Feedback configuration: If the blocks are connected as shown below then the blocks are said to be in feedback. Notice that in the feedback there is no transfer function H(s) defined. When not specified, H(s) is unity. Such a system is said to be a unity feedback system.



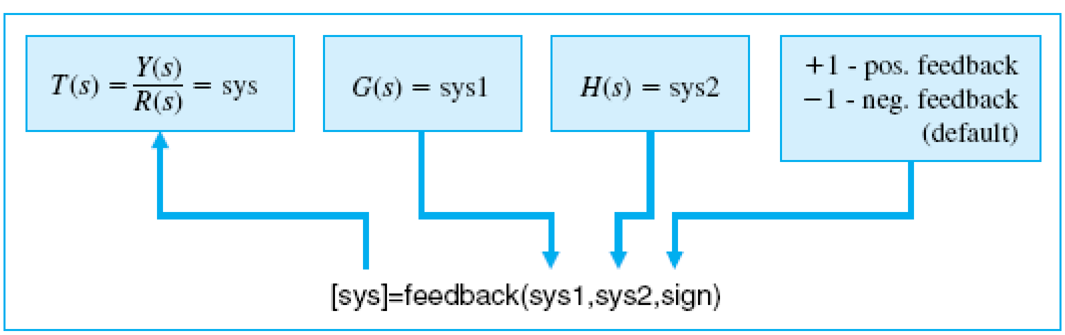
When H(s) is non-unity or specified, such a system is said to be a non-unity feedback system as shown below:



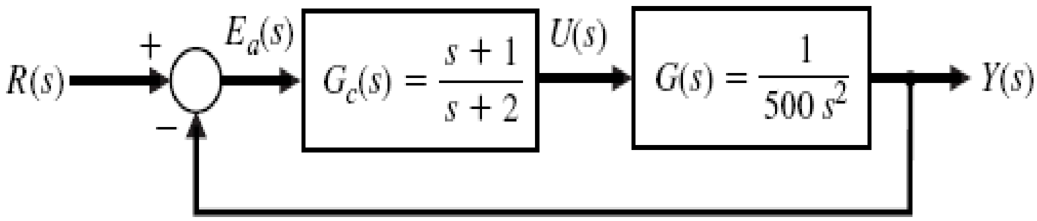
The MATLAB command for implementing a feedback system is "feedback" as shown below:



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



Block diagram:



Program:

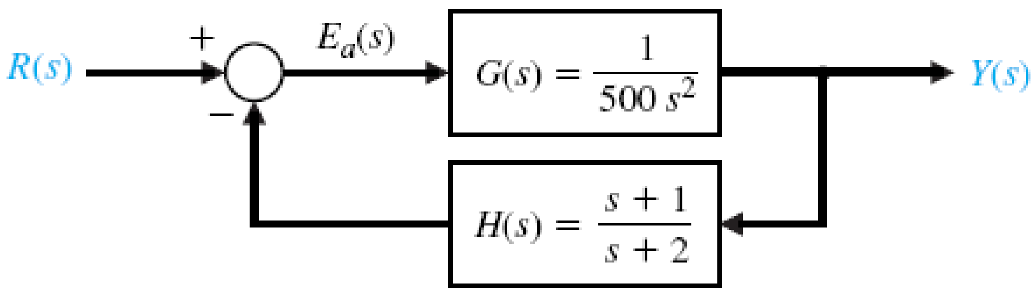
```
>>numg=[1]; deng=[500 0 0]; sys1=tf(numg,deng);
>>numc=[1 1]; denc=[1 2]; sys2=tf(numc,denc);
>>sys3=series(sys1,sys2);
>>sys=feedback(sys3,[1])

Transfer function:
           s + 1
---------------------------
500 s^3 + 1000 s^2 + s + 1
```

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

Result:

b) simulation of non unity feedback system

Block diagram



:

```
>>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:

```
          s + 2
--------------------------
500 s^3 + 1000 s^2 + s + 1
```

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

Result:

6.5. Block diagram reduction technique using MATLAB

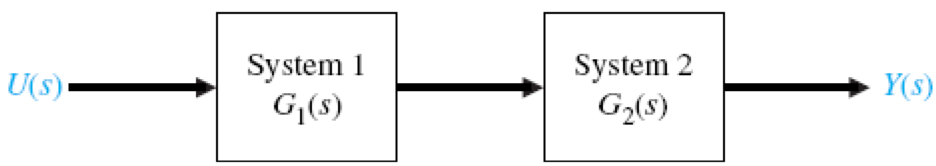
Aim: simulation of multi feedback system

SOFTWARE REQUIRED: MATLAB – Personal Computer with MATLAB

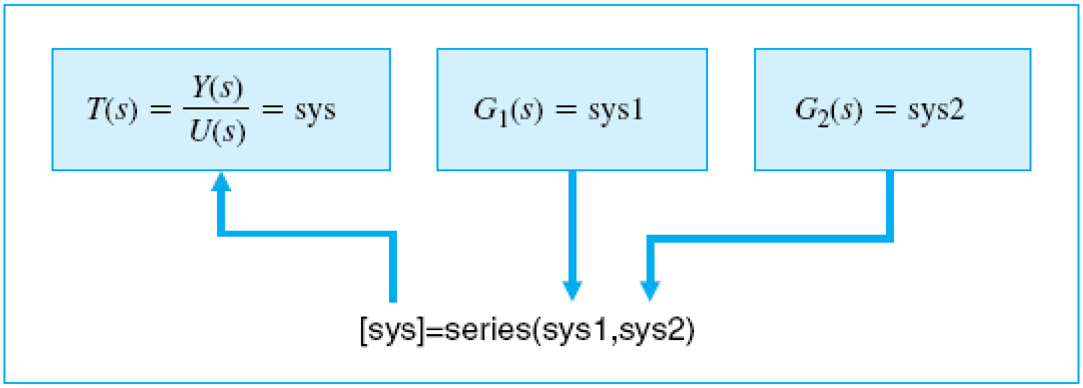
Theory:

Series configuration: If the two blocks are connected as shown below then the blocks are said

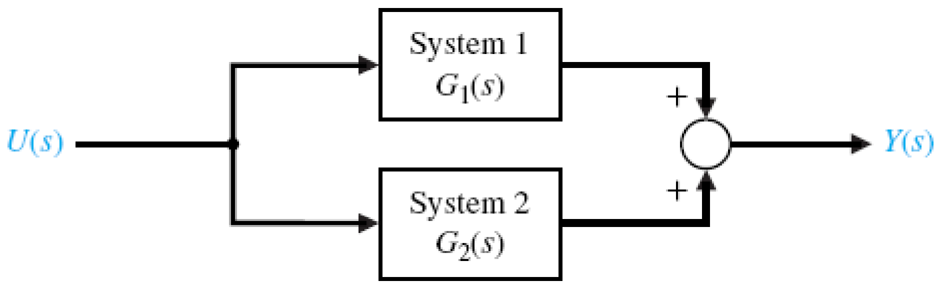
to be in series. It would like multiplying two transfer functions. The MATLAB command for the such configuration is "series".



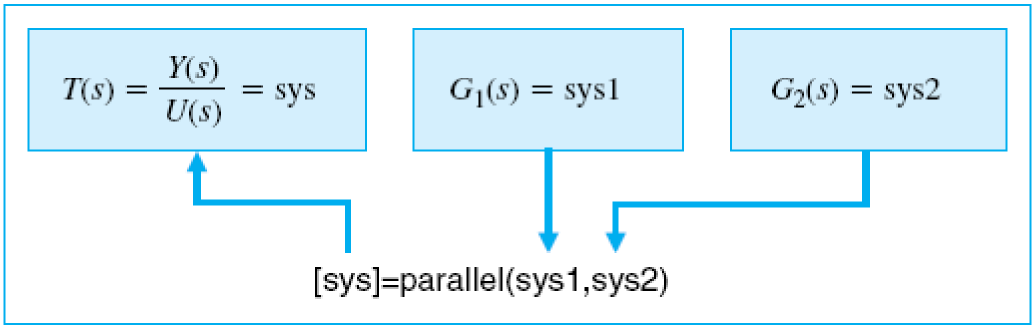
The series command is implemented as shown below



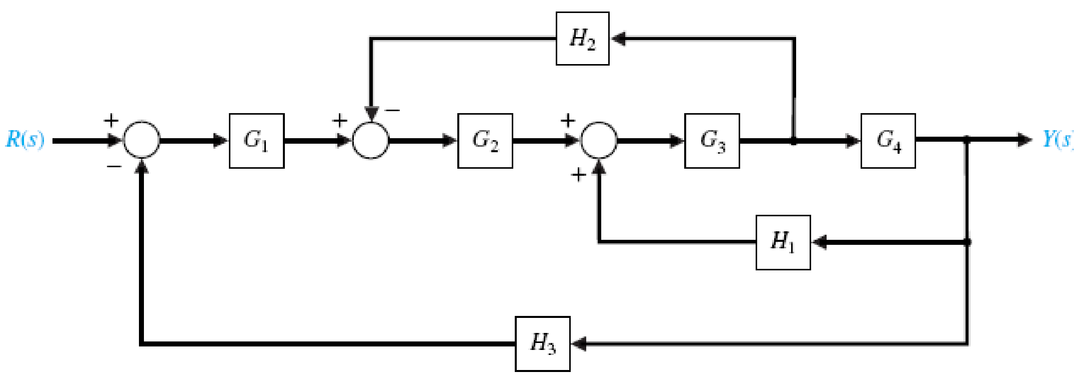
Parallel configuration: If the two blocks are connected as shown below then the blocks are said to be in parallel. It would like adding two transfer functions



The MATLAB command for implementing a parallel configuration is "parallel" as shown below:



Blockdiagram:



Program:

```
>>ng1=[1]; dg1=[1 10]; sysg1=tf(ng1,dg1);
>>ng2=[1]; dg2=[1 1]; sysg2=tf(ng2,dg2);
>>ng3=[1 0 1]; dg3=[1 4 4]; sysg3=tf(ng3,dg3);
>>ng4=[1 1]; dg4=[1 6]; sysg4=tf(ng4,dg4);        Step 1
>>nh1=[1 1]; dh1=[1 2]; sysh1=tf(nh1,dh1);
>>nh2=[2]; dh2=[1]; sysh2=tf(nh2,dh2);
>>nh3=[1]; dh3=[1]; sysh3=tf(nh3,dh3);
>>sys1=sys2/sys4;                                 Step 2
>>sys2=series(sysg3,sysg4);
>>sys3=feedback(sys2,sysh1,+1);                   Step 3
>>sys4=series(sysg2,sys3);
>>sys5=feedback(sys4,sys1);                       Step 4
>>sys6=series(sysg1,sys5);
>>sys=feedback(sys6,[1]);                         Step 5
Transfer function:

                  s^5 + 4 s^4 + 6 s^3 + 6 s^2 + 5 s + 2
  -----------------------------------------------------------------------
  12 s^6 + 205 s^5 + 1066 s^4 + 2517 s^3 + 3128 s^2 + 2196 s + 712
```

Result:

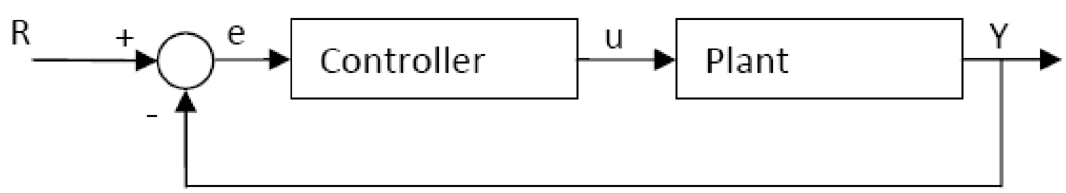
6.6. Simulation of P, PD, PI, PID controller

Aim: To simulate the P, PD, PI, PID controller for unit step input.

SOFTWARE REQUIRED: **MATLAB – P**ersonal Computer with MATLAB.

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 above. 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 PIDcontroller, 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 themagnitude of the error plus the integral gain (KI) times the integral of the error plus thederivative gain (KD) times the derivative of the error.

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

Plug these values into the above transfer function

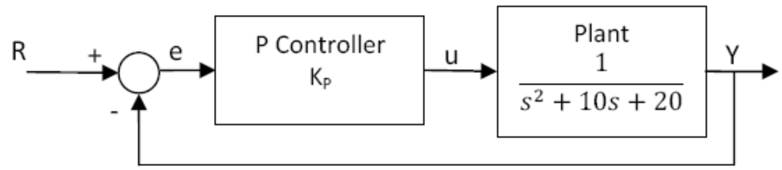
$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 10s + 20}$$

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

Proportional control:

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

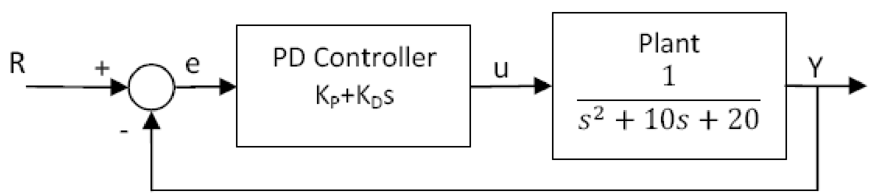


$$\frac{X(s)}{F(s)} = \frac{K_P}{s^2 + 10s + (20 + K_P)}$$

Let the proportional gain (K_P) equal 300:

Proportional-Derivative control:

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

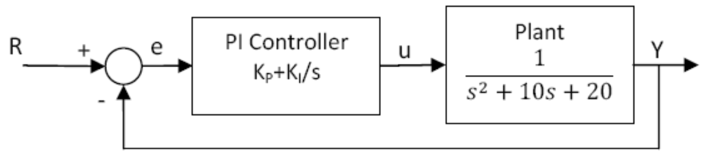


$$\frac{X(s)}{F(s)} = \frac{K_D s + K_P}{s^2 + (10 + K_D)s + (20 + K_P)}$$

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

Proportional-Integral control:

Before going into a PID control, let's take a look at a PI control. For the given system, the closed-loop transfer function with a PI control is:

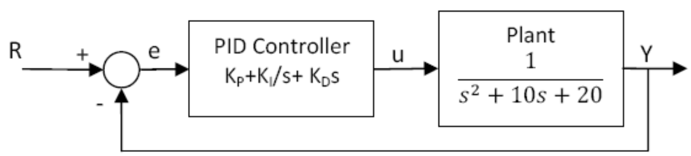


$$\frac{X(s)}{F(s)} = \frac{K_P s + K_I}{s^3 + 10s^2 + (20 + K_P)s + K_I}$$

Let's reduce the K_P to 30, and let K_I equal 70.

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 Kp=350, Ki=300, and Kd=50 provided the desired response. To confirm, enter the following commands to an m-file and run it in the command window. You should get the following step response.

The characteristics of P, I, and D controllers:

The proportional controller (K_P) will have the effect of reducing the rise time and will reduce, but never eliminate, the steady state error. An integral controller (K_I) will have the effect of eliminating the steady state error, but it may make the transient response worse. A derivative control (K_D) will have the effect of increasing the stability of the system, reducing the overshoot and improving the transient response.

Effect of each controller K_P, K_I and K_D on the closed-loop system are summarized below

CL Response	Rise Time	Overshoot	Settling Time	S-S Error
K_P	Decrease	Increase	Small Change	Decrease
K_I	Decrease	Increase	Increases	Eliminate
K_D	Small Change	Decreases	Decreases	Small Change

Program:

Open-loop step response:

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

Proportional control:

```
Kp=300;
contr=Kp;
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```

Proportional-Derivative control:

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

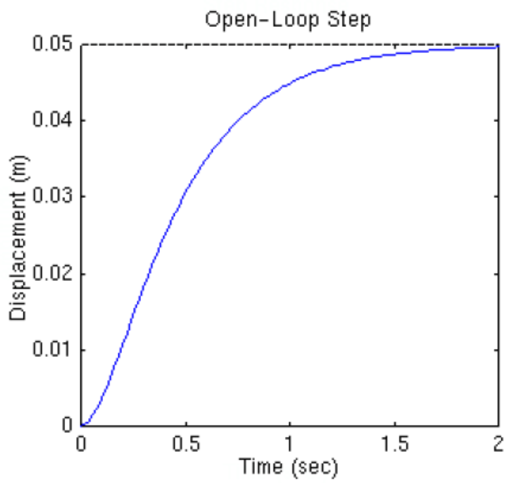
Proportional-Integral control:

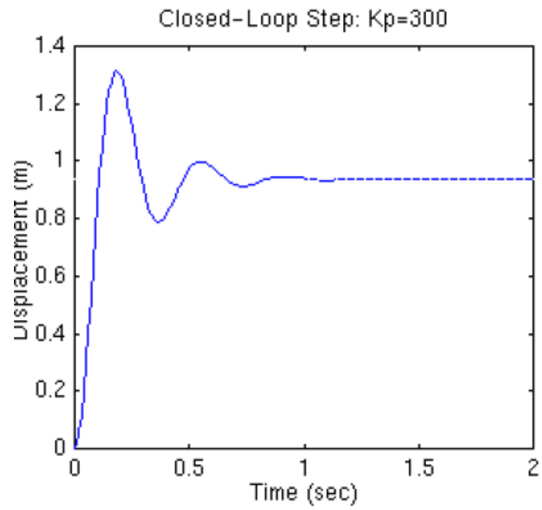
```
Kp=30;
Ki=70;
contr=tf([Kp Ki],[1 0]);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```

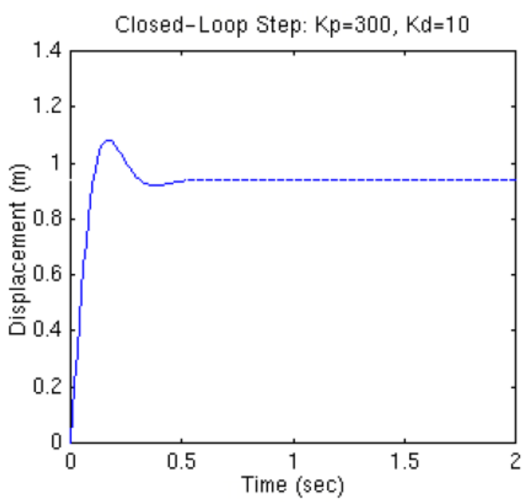
Proportional-Integral-Derivative control:

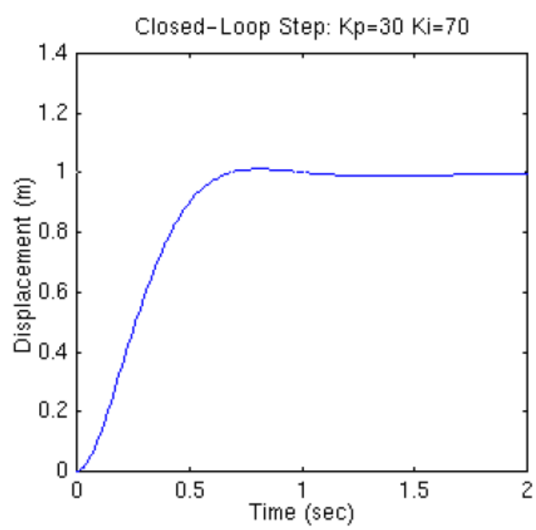
```
Kp=350;
Ki=300;
Kd=50;
contr=tf([Kd Kp Ki],[1 0]);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```

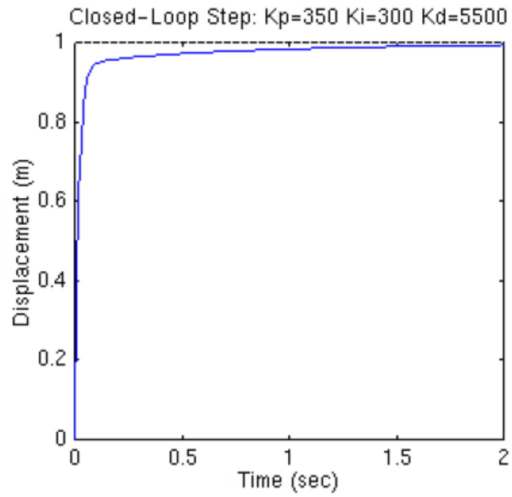
Model graph;


Open-Loop Step
Displacement (m)
Time (sec)
0.05
0.04
0.03
0.02
0.01
0
0
0.5
1
1.5
2


Closed-Loop Step: Kp=300
Displacement (m)
Time (sec)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2


Closed-Loop Step: Kp=300, Kd=10
Displacement (m)
Time (sec)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2


Closed-Loop Step: Kp=30 Ki=70
Displacement (m)
Time (sec)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2



Observation:

Effect of each controller K_P, K_I and K_D on the closed-loop system are summarized below

CL Response	Rise Time	Overshoot	Settling Time	S-S Error
K_P	Decrease	Increase	Small Change	Decrease
K_I	Decrease	Increase	Increases	Eliminate
K_D	Small Change	Decreases	Decreases	Small Change

Result:

6.7.Simulation of DC Motor characteristics using MATLAB

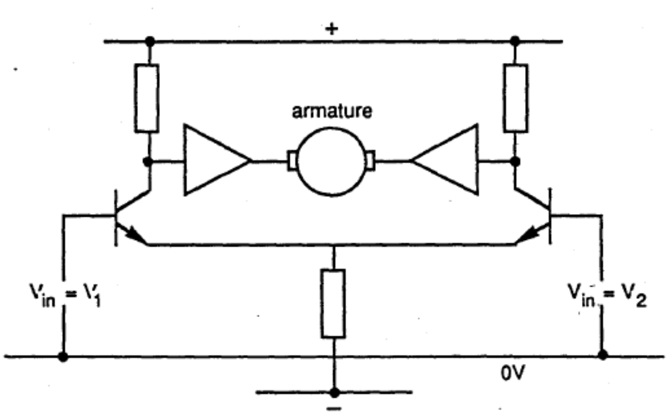
AIM: To Simulate the Dc Motor Characteristics Using MATLAB.

SOFTWARE REQUIRED: **MATLAB – P**ersonal Computer with MATLAB.

THEORY:

This experiment will illustrate the characteristics of the D.C. motor used in the Modular Servo and show how it can be controlled by the Servo Amplifier.

The motor is a permanent magnet type and has a single armature winding. Current flow through the armature is controlled by power amplifiers as in figure so that rotation in both directions is possible by using one, or both of the inputs. In most of the later assignments the necessary input signals are provided by a specialized Pre-Amplifier Unit PA150C, which connected to Inputs 1 and 2 on SA150D



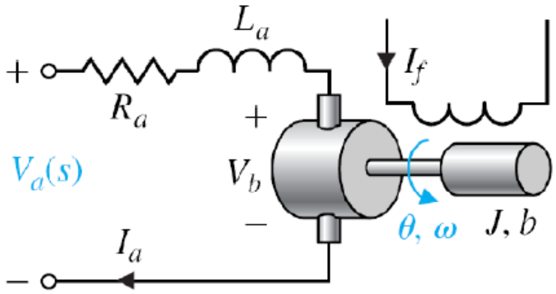
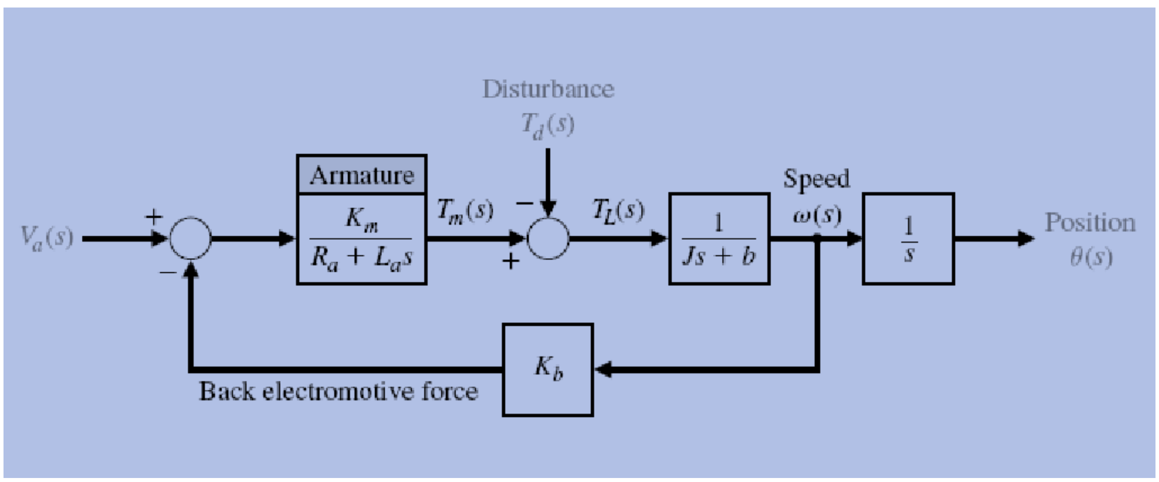


Figure: DC motor armature-controlled rotational actuator

As the motor accelerates the armature generates an increasing 'back-emf' V_a tending to oppose the driving voltage V_{in}. The armature current is thus roughly proportional to (V_{in} - V_a). If the speed drops (due to loading) V_a reduces, the current increases and thus so does the motor torque. This tends to oppose the speed drop. This mode of control is called 'armature-control' and gives a speed proportional to V_{in} as in figure.

The final block diagram is as follows:

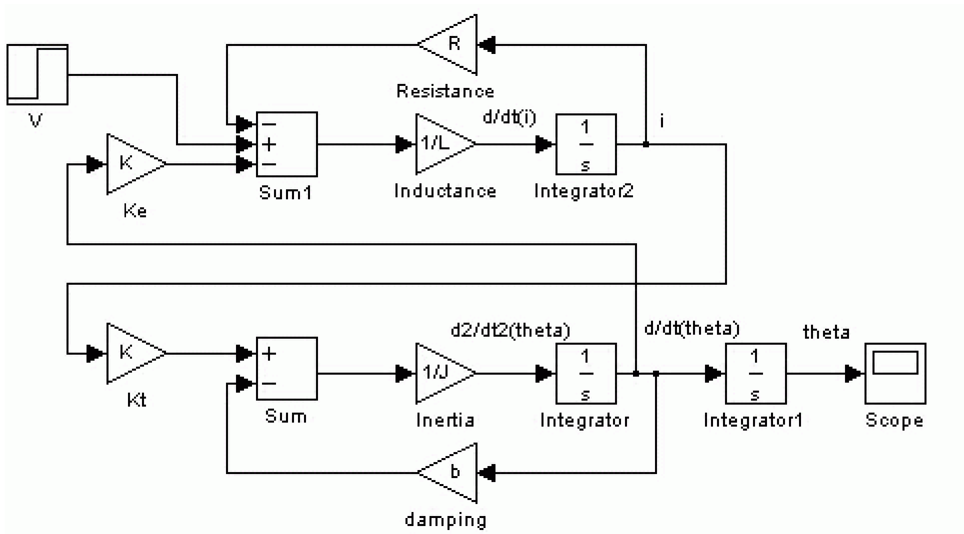


2. DC motor nominal values

- moment of inertia of the rotor (J) = 3.2284E-6 kg.m^2/s^2
- damping ratio of the mechanical system (b) = 3.5077E-6 Nms
- electromotive force constant (K=Ke=Kt) = 0.0274 Nm/Amp

- electric resistance (R) = 4 ohm
- electric inductance (L) = 2.75E-6 H
- input (V): Source Voltage
- output (theta): position of shaft

SIMULINK DIAGRAM:



OUTPUT WAVE FORMS:

RESULT:

6.8.Simulation of poles and zeros of a transfer function

Aim: To simulate the Poles and zeros of the given transfer function.

SOFTWARE REQUIRED: **MATLAB – P**ersonal Computer with MATLAB.

Theory:

To obtain the poles and zeros of the system use the MATLABcommand “pole” and“zero” respectively as shown in example 5. You can also use MATLABcommand “pzmap” to obtain the same.

Transfer function:

$$\text{Given } G_1 = \frac{1}{(s+10)};\ G_2 = \frac{1}{(s+1)};\ G_3 = \frac{s^2+1}{(s^2+4s+4)};$$

H — 1

Program:

```
ng1=[1];

dg1=[1 10];
```

```
sysg1=tf(ng1,dg1);

ng2=[1];

dg2=[1 1];

sysg2=tf(ng2,dg2);

ng3=[1 0 1];

dg3=[1 4 4];

sysg3=tf(ng3,dg3);

p1=pole(sysg1)

z1=zero(sysg1)

p2=pole(sysg2)

z2=zero(sysg2)

p3=pole(sysg3)

z3=zero(sysg3)
```

Output:

6.9.STATE MODEL FOR CLASSICAL TRANSFER FUNCTION &

VICE VERSA USING MATLAB

AIM:

To find state model for classical transfer function and transfer function from state model using MATLAB.

APPARATUS: *Computer with MATLAB software*

THEORY:

COMMAND 1:	CLC: It clears the MATLAB command window
COMMAND 2:	CLEAR: it clears the MATLAB work shop variables.
COMMAND 3:	DISP: Syntax – disp (variable): It displays the variable specified on command window.
COMMAND 4:	PAUSE: With this command the execution will be stopped and it waits for the enter key.
COMMAND 5:	INPUT: Syntax: Variable = Input ('Comment');
COMMAND 6:	PERCENTAGE: It is used at the beginning of any statement to make it as a comment in the program.
COMMAND 7:	R-LOCUS: Syntax: r locus (Variable): With this we can plot the root locus of any transfer function. That means in the above syntax the variable is nothing but a transfer function.
COMMAND 8:	BODE: Syntax: Bode (Variable): With this command we can get bode plot of the given transfer function.
COMMAND 9:	MARGIN: Syntax: Margin (Variable): With this command we can get gain and phase margin of a bode plot of the given transfer function.

COMMAND10: SS: Syntax: Variable1= SS(Variable2): With this command we can get state space model for the given transfer function. Variable 2 is a transfer function and variable 1 holds the SS model.

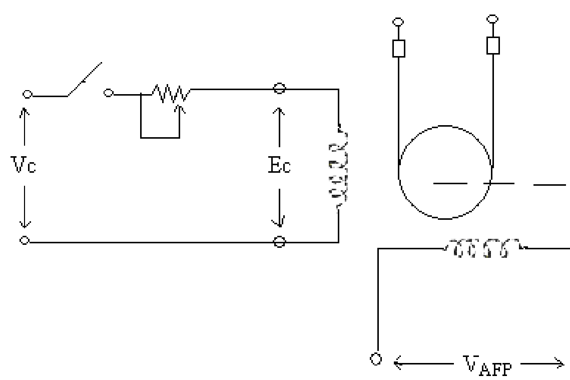
COMMAND11: SS DATA:Syntax: [a,b,c,d]=SSdata (Variable):

With this command we can retrieve the a,b,c,d matrices of a state space model. Variable holds the state space model.

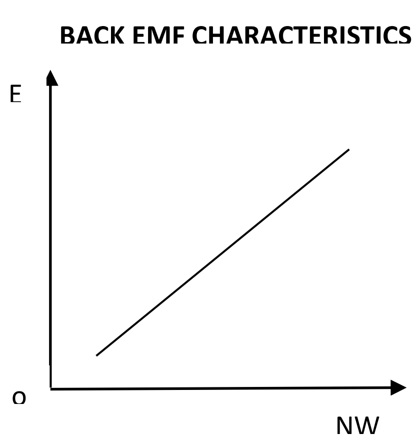
PROCEDURE:

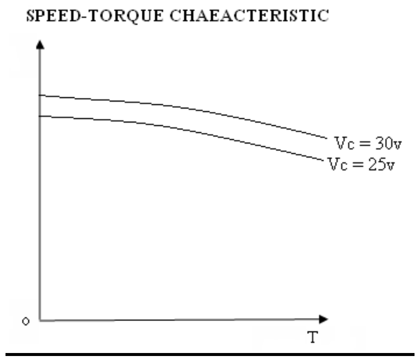
1. Write the programme in MATLAB text editor using mat lab instructions for state model of classical transfer function and for transfer function from state model.
2. Run the programs.
3. Note down the outputs.

CIRCUIT DIAGRAM:



GRAPHS





PROGRAM 1:

```
a= input ( "Enter the values of a matrix" );
b= input ( "Enter the values of b matrix" );
c= input ( "Enter the values of c matrix" );
d= input ( "Enter the values of d matrix" );

[num , den] = SS2 tf  (a,b,c,d,1)
S1=tf  (num(1, : ) , den );
S2=tf  (num(2, : ) , den );
 [num1 , den1 ] = SS2 tf  (a,b,c,d,2);
S3=tf  (num1 (1, : ) , den1 );
```

```
S4=tf  (num1 (2, : ) , den1 );

DISP [S1,S2,S3,S4];
```

PROGRAM 2:

```
Num = input ("Enter numerator polynomial values in the form of matrix array" );

den1 = input ("Enter denominator 1 values" );

den2 = input ("Enter denominator 2 values" );

den = conv (den1,den2);

H = tf (num,den);

P = SS(H);

[a,b,c,d] = SS data(P);
```

RESULT:

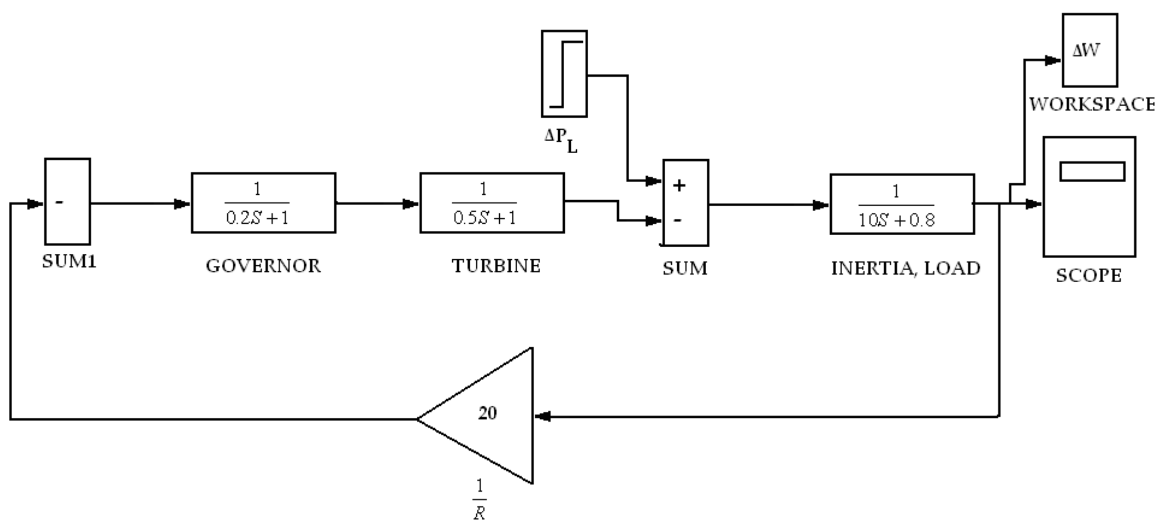
The state model for classical transfer function and transfer function from state model are obtained using MATLAB software.

6.10. TRANSFER FUNCTION ANALYSIS OF 3rd ORDER USING SIMULINK

AIM: To Simulate the transfer function of 3rd order system using SIMULINK.

SOFTWARE REQUIRED: **MATLAB – MAT**rix **LAB**oratory.

CIRCUIT DIAGRAM:



OUTPUT:

RESULT: Determination of Transfer function analysis of 3RD order system with SIMULINK has been successfully completed

6.11.STABILITY ANALYSIS USING BODE PLOTE USING MATLAB

AIM: To find out the Simulation of stability analysis of Linear Time Invariant using MATLAB.

SOFTWARE REQUIRED: **MATLAB – Mat**rix **Lab**oratory.

BODE PLOT for 2^{nd}, 3^{rd}, 4^{th} & 5^{th} order systems:

2^{nd} order system:

The transfer function is G(s) = $\frac{25}{s^2 + 4s + 25}$. Plot the bode plot.(p.517)

Program:

```
num=[0 0 25];
den=[1 4 25];
bode(num,den)
title('Bode diagram of G(s)= 25/s^2+4s+25)')
```

3^{rd} order system:

The transfer function is G(s) = $\frac{9\left(s^2+0.2s+1\right)}{s(s^2+4s+25)}$. Plot the bode plot.(p.518)

Program:

```
num=[0 9 1.8 9];

den=[1 1.2 9 0];

bode(num,den)

title('Bode diagram of G(s)= 9(s^2+0.2s+1)/s(s^2+1.2s+9)')
```

4th order system:

The transfer function is G(s)= $\frac{4(2s+1)}{s^2(s^2+0.4s+4)}$. obtain the Bode plot(p.611)

Program:

```
num=[0 0 0 8 4];

den=[1 0.4 4 0 0];

bode(num,den)

title('Bode diagram of G(s)= 4(2s+1)/(s^2(s^2+0.4s+4)')
```

5th order system:

The transfer function is G(s) = $\frac{40(s+0.4)(s+0.2)}{s(s+4)(s+0.02)(s+1)(s+2)}$

Plot the bode plot.(p.670)

Program:

```
num=[0 0 0 40 24 3.2];

den=[1 9.02 24.18 16.48 0.32 0];

bode(num,den)

title('Bode diagram of G(s)')
```

Result: Bode plot is plotted for the given transfer function is plotted using MATLAB has been successfully completed.

6.12. STABILITY ANALYSIS USING ROOT LOCUS USING MATLAB

ROOT LOCUS for 2nd , 3rd ,4th & 5th order systems:

2nd order system:

The transfer function is G(s) = $\frac{K(s+2)}{(s+3)(s+4)}$. Obtain the Root Locus plot.

Program:

```
num=[0 1 2 ];
den=[1 7 12];
rlocus(num,den)
v=[-6 6 6 -6];
axis(v);
grid;
title('Root Locus Plot of G(s)=K(s+2)/[(s+3)(s+4)]')
```

3rd order system:

The transfer function is G(s) = $\frac{K(s+0.4)}{s^2(s+3.6)}$. Obtain the Root Locus plot. (p.400)

Program:

```
num=[0 0 1 0.4];

den=[1 3.6 0 0];

rlocus(num,den)

v=[-5 1 –3  3]; axis(v);

grid;

title('Root Locus Plot of G(s)=K(s+0.4)/[s^2(s+3.6)]')
```

4th order system:

The transfer function is G(s) = $\frac{K(s+3)}{s(s+1)(s^2+4s+16)}$.. Obtain Root locus plot. (p.360)

Program:

```
num=[0 0 0 1 3];

den=[1 5 20 16 0];

rlocus(num,den)

v=[-6 6 -6 6];

axis(v); axis('square');

grid;

title('Root-Locus Plot of G(s)=K(s+3)/[s(s+1)(s^2+4s+16)]')
```

5th order system:

The transfer function is G(s) = $\frac{K(s^2+2s+4)}{s(s+4)(s+6)(s^2+1.4s+1)}$. Obtain the Root locus Plot. (p.378)

Program:

```
num=[0 0 0 1 2 4];

den=[1 11.4 39 43.6 24 0];

rlocus(num,den)

v=[-7 3 -5 5];

axis(v); axis('square');

grid;

title('Root Locus Plot of G(s) = K(s^2+2s+4)/[s(s+4)(s+6)(s^2+1.4s+1)]')
```

Result: Root Locus Plot is plotted for the given transfer function is plotted using MATLAB has been successfully completed.

6.13.STABILITY ANALYSIS USING NYQUIST PLOT USING MATLAB

NYQUIST PLOT for 2nd , 3rd ,4th & 5th order systems:

2nd order system:

The transfer function is G(s) = $\frac{1}{s^2 + 0.8s + 1}$. Obtain the Root Locus plot. (p.532)

Program:

```
num=[0 0 1];
den=[1 0.8 1];
nyquist(num,den)
v=[-2 2 -2 2];
axis(v);
grid;
title('Nyquist Plot of G(s)=1/s^2+0.8s+1)')
```

3rd order system:

The transfer function is G(s) = $\frac{20(s^2 + s + 0.5)}{s(s+1)(s+10)}$. Draw a Nyquist plot using MATLAB. (p.600)

Program:

```
num=[0 20 20 10];

den=[1 11 10 0];

nyquist(num,den)

v=[-2 3 –3  3]; axis(v);

grid;

title('Nyquist Plot of G(s)=20(s^2+s+0.5)/[s(s+1)(s+10)]')
```

4^{TH} order system:

The transfer function is G(s) = $\frac{100(s+0.1)}{(s+10)(s^3+3S^2+2S+10)}$. Draw a Nyquist plot using MATLAB. (p.600)

Program:

```
num=[0 0 0 100 10];

den=[1 13 32 26 60];

nyquist(num,den)

v=[-2 3 –3  3]; axis(v);

grid;

title('Nyquist Plot of G(s)=100(s+0.1)/[(s+10)(s^3+3S^2+2S+10)]')
```

Result: Nyquist Plot is plotted for the given transfer function is plotted using MATLAB has been successfully completed.

7 Content beyond syllabus:

8 Sample Viva Voce Questions

Exp 1:

1. What is delay time?
2. What is rise time?
3. What is peak time?
4. What is peak overshoot?
5. What is settling time?

Exp 2:

1. What is a synchro?
2. What is the use of synchro?
3. What is the constructional difference between synchro transmitter & synchro receiver?
4. What is the relation between a synchro & a transformer?
5. Where do we get maximum e m f in a synchro?
6. When we will get maximum e m f in a synchro?
7. What is the phase different between three voltages induced in the stator of synchro and why ?
8. How do you determine zero position of synchro
9. what is the error voltage induced ?

Exp 3:

1. What is a servomotor?
2. What are the applications of servomotor?
3. How do you load the D.C Servomotor?
4. Why a servomotor should not be switched on load?
5. What are the elements used as feedback
6. What are the general input and o/p parameters of D.C. servomotor

7. What is the element used as error detector in the given circuit.

Exp 4:

What is a servomotor?

2. What are the applications of servomotor?
3. How can we get the feed back characteristics of D.C Servomotor?
4. How do you load the D.C Servomotor?
5. Why a servomotor should not be switched on load?
6. What is a mathematical model ? What is its importance?
7. How do you define transfer function? What is its significance?
8. What are K_b, K_T

Exp 5:

1. What is the use of a controller in control system?
2. What is the use of proportionality controller?
3. Why is integral controller used?
4. Why is differential controller used?
5. How can you rectify an error using controller?
6. What is meant by sampling network?
7. How do you sense the errors in a control system?
8. What do you mean by tuning of controller?
9. Which controller is most commonly used?

Exp 6:

1. What is the instruction used for plotting bode plot
2. do we get transfer function of a control system using MATLAB?
3. How do you load the A.C Servomotor?

Exp 7:

1. What is the formula for calculating phase angle?
2. What is the formula magnitude of phase lead circuit lag network & loss?
3. What is the difference between lag network & low pass filter?
4. What is meant by compensation?
5. How a lag network can be compensated?

Exp 8:

1. What is a magnetic amplifier?
2. What is the difference between magnetic amplifier & electronic amplifier?
3. Which amplifier (series parallel) gives maximum amplification?
4. What is the need of control winding?
5. What is the need of bias winding?

Exp 9:

1. What is MATLAB?
2. What are the applications of MATLAB?
3. What is the instruction used for plotting root locus?
4. What is the instruction used for plotting bode plot?
5. How do we get transfer function of a control system using MATLAB?
6. How many windows does it has?
7. How do you differentiate C language programming with MATLAB?

Exp 10:

1. What is MATLAB?
2. What are the applications of MATLAB?
3. What is the instruction used for plotting root locus?
4. What is the instruction used for plotting bode plot?
5. How do we get transfer function of a control system using MATLAB?

Exp 11:

1. What is a servomotor?
2. What are the applications of servomotor?
3. How can we get the feed back characteristics of A.C Servomotor?
4. How do you load the A.C Servomotor?
5. Why a servomotor should not be switched on load?
6. How can a A.C servomotor be controlled?

9. **Sample Question paper of the lab external**

1 Simulate op-amp based integrator &differentiat circuits

2 write a program for saw tooth wave and sine wave using matlab

3 write a program for triangular wave and ramp wave using matlab

4 write a program for Unity and non unity feedback system using matlab

5 Simulate Block diagram reduction technique using matlab

6 write a program for p, pd, pi, pid controller

7 Simulate dc motor characteristics using matlab

8 write a program for poles and zeros of a transfer function

9 write a program for State model for classical transfer function &vice versa using matlab

10 write a program for Transfer function analysis of 3rd order using simulink

11 write a program for Stability analysis using bode plote using matlab

12 write a program for Stability analysis using root locus using matlab

13 write a program for Stability analysis using nyquist plot using matlab

10. Applications of the laboratory

1) To find the state space model of control system using simulation

2) To find the second order system output using simulation

3) to know the closed loop system using simulation

4) to find the stability of control systems using simulation

5) plot the locus diagram for second and higher order system using simulation

11. Precautions to be taken while conducting the lab

SAFETY – 1

Power must be switched-OFF while making any connections.

- Do not come in contact with live supply.
- Power should always be in switch-OFF condition, EXCEPT while you are taking readings.
- The Circuit diagram should be approved by the faculty before making connections.
- Circuit connections should be checked & approved by the faculty before switching on the power.
- Keep your Experimental Set-up neat and tidy.
- Check the polarities of meters and supplies while making connections.
- Always connect the voltmeter after making all other connections.
- Check the Fuse and it's ratify.
- Use right color and gauge of the fuse.
- All terminations should be firm and no exposed wire.
- Do not use joints for connection wire.

SAFETY – II

1. The voltage employed in electrical lab are sufficiently high to endanger human life.
2. Compulsorily wear shoes.
3. Don't use metal jewelers on hands.
4. Do not wear loose dress

Don't switch on main power unless the faculty gives the permission

12. Code of Conduct

1. Students should report to the labs concerned as per the timetable.
2. Students who turn up late to the labs will in no case be permitted to perform the experiment scheduled for the day.
3. After completion of the experiment, certification of the staff in-charge concerned in the observation book is necessary.
4. Students should bring a notebook of about 100 pages and should enter the readings/observations/results into the notebook while performing the experiment.
5. The record of observations along with the detailed experimental procedure of the experiment performed in the immediate previous session should be submitted and certified by the staff member in-charge.
6. Not more than three students in a group are permitted to perform the experiment on a set up.
7. The group-wise division made in the beginning should be adhered to, and no mix up of student among different groups will be permitted later.
8. The components required pertaining to the experiment should be collected from Lab-in-charge after duly filling in the requisition form.
9. When the experiment is completed, students should disconnect the setup made by them, and should return all the components/instruments taken for the purpose.
10. Any damage of the equipment or burnout of components will be viewed seriously either by putting penalty or by dismissing the total group of students from the lab for the semester/year.
11. Students should be present in the labs for the total scheduled duration.
12. Students are expected to prepare thoroughly to perform the experiment before coming to Laboratory.
13. Procedure sheets/data sheets provided to the students' groups should be maintained neatly and are to be returned after the experiment.

13. Graphs if any.