KLS

Vishwanathrao Deshpande Institute of Technology

Haliyal – 581329

Electronics and Communication Engineering Department

LAB Manual

2022 Scheme

Semester: 4th			
Sl.No.	**Lab Details**	**Lab Code**	**Pg. No.**
1	Principles of Communication Systems	BEC402	4
2	Control Systems Laboratory	BEC403	38
3	Communication Laboratory	BECL404	81
4	Microcontroller Laboratory	BECL456A	123

As prescribed by

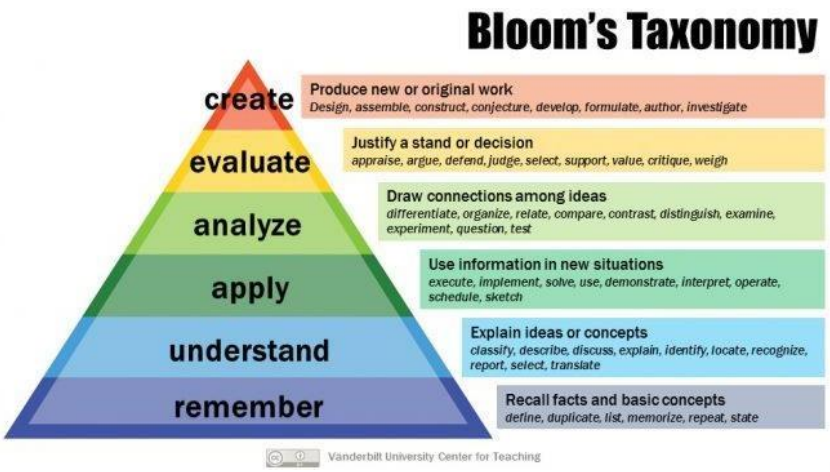
VISVESVARAYA TECHNOLOGICAL UNIVERSITY,

BELAGAVI – 590014

PROGRAM OUTCOMES(POs)

Program Outcomes as defined by NBA (PO) Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



Vision (College)

To nurture talent & enrich society through excellence in technical education, research & innovation.

Mission (College)

1. To augment innovative. Pedagogy & kindle quest for interdisciplinary learning & to enhance conceptual understanding.
2. To build competence, professional ethics & develop entrepreneurial thinking.
3. To strengthen Industry Institute Partnership & explore global collaborations.
4. To inculcate culture of socially responsible citizenship.
5. To focus on Holistic & Sustainable development

Vision (Dept)

To bring out talented, skilled, and sustainable Electronics and Communication Engineering Graduates through strong domain expertise to serve the Society with greater Professional Ethics.

Mission

1. **To create and impart an active learning ambience to accomplish a high degree of Professional competencies**
2. **To inculcate innovative research and developmental thinking in effective Teaching and Learning processes for solving Societal challenges**
3. **To deliver the needs and requirements of the latest state of art of the Industry through quality multidisciplinary internship and training programs**

Program Educational Objectives (PEO)

PEO1	To be successful in professional career in electronics, communication and allied industries by acquiring the knowledge in the fundamentals of Electronics and Communication Engineering principles and professional skills.
PEO2	To be in a position to analyze real life problems and design socially accepted and economically feasible solutions in the respective fields.
PEO3	To exhibit good communication skills in their professional career, lead a team with good leadership traits and good interpersonal relationship with the members related to other engineering streams.
PEO4	To involve themselves in lifelong learning and professional development by pursuing higher education and participation in research and development activities.
PEO5	To demonstrate professional and ethical responsibilities towards their profession, society and the environment.

Program Specific Outcomes (PSO)

PSO 1	An ability to use appropriate modern techniques for analysis, design and development of VLSI and Embedded Systems.
PSO 2	Understand the architectural specifications of a communication system and determine their performance.

Lab Manual 1

Principles of Communication Systems (IPCC) (BEC402)

CO's And PO's Mapping Chart

Sl. No.	Description	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	P0 12	PSO 1	PSO 2
1	Understand the principles of analog communication systems and noise modeling.	1	1			3					2				2
2	Identify the schemes for analog modulation and demodulation and compare their performance.	1	1			3					2				2
3	Design of PCM systems through the processes sampling, quantization and encoding.	1	2			3					2				2
4	Describe the ideal condition, practical considerations of the signal representation for baseband transmission of digital signals.	1	1			3					2				2
5	Identify and associate the random variables and random process in Communication system design.	1	1			3					2				2

Evaluation:

Integrated Lab(IPCC)		
CIE		
Particulars	**Marks**	**Total**
Performance	05	15 (Reduced)
Journal	03	
Viva Voce	02	
Lab IA		
Particulars	**Marks**	**Total**
IA	50	10 (Reduced)
Total (CIE +Lab IA)		25

Mapping of Experiments with CO, PO and PSO

S.No.	Experiment Details	CO	PO	PSO
1	Basic Signals and Signal Graphing: a) unit Step, b) Rectangular, c) standard triangle d) sinusoidal and e) Exponential signal.	4	1, 2, 5, 10	2
2	Illustration of signal representation in time and frequency domains for a rectangular pulse	4	1, 2, 5, 10	2
3	Amplitude Modulation and demodulation: Generation and display the relevant signals and its spectrums.	2	1, 2, 5, 10	2
4	Frequency Modulation and demodulation: Generation and display the relevant signals and its spectrums	2	1, 2, 5, 10	2
5	Sampling and reconstruction of low pass signals. Display the signals and its spectrum.	4	1, 2, 5, 10	2
6	Time Division Multiplexing and demultiplexing.	1	1, 2, 5, 10	2
7	PCM Illustration: Sampling, Quantization and Encoding	3	1, 2, 5, 10	2
8	Generate a)NRZ, RZ and Raised cosine pulse, b) Generate and plot eye diagram	4	1, 2, 5, 10	2
9	Generate the Probability density function of Gaussian distribution function.	5	1, 2, 5, 10	2
10	Display the signal and its spectrum of an audio signal.	4	1, 2, 5, 10	2
11	Virtual Lab: Study Of Sampling Theorem, Effect Of Under sampling	5	1, 2, 5, 10	2

EXPERIMENT WISE LESSON PLAN

Experiment No.1	
Name	Basic Signals and Signal Graphing: a) unit Step, b) Rectangular, c) standard triangle d) sinusoidal and e) Exponential signal.
Objectives	Student will be able to understand different basic signals and their properties
Experiment No.2	
Name	Illustration of signal representation in time and frequency domains for a rectangular pulse
Objectives	Students will able to understand signal representation in time and frequency domains
Experiment No.3	
Name	Amplitude Modulation and demodulation: Generation and display the relevant signals and its spectrums.
Objectives	Students will able to study and understand Amplitude Modulation and demodulation using its spectrum
Experiment No.4	
Name	Frequency Modulation and demodulation: Generation and display the relevant signals and its spectrums
Objectives	Students will able to study and understand Frequency Modulation and demodulation
Experiment No. 5	
Name	Sampling and reconstruction of low pass signals. Display the signals and its spectrum.
Objectives	Students will able to study the concept of Sampling and reconstruction of low pass signals
Experiment No. 6	
Name	Time Division Multiplexing and demultiplexing.
Objectives	Students will able to understand Time Division Multiplexing and demultiplexing of Signals
Experiment No. 7	
Name	PCM Illustration: Sampling, Quantization and Encoding
Objectives	Students will able to understand PCM Illustration: Sampling, Quantization and Encoding
Experiment No. 8	

Name	Generate a)NRZ, RZ and Raised cosine pulse, b) Generate and plot eye diagram
Objectives	Students will able to understand Generation of NRZ, RZ and Raised cosine pulse and plot eye diagram
Experiment No.9	
Name	Generate the Probability density function of Gaussian distribution function.
Objectives	Students will able to understand Generation of Probability density function of Gaussian distribution function.
Experiment No.10	
Name	Display the signal and its spectrum of an audio signal.
Objectives	Students will able to understand spectrum of an audio signal
	Experiment No.11
Name	Virtual Lab: Study Of Sampling Theorem, Effect Of Under sampling
Objectives	Students will able to understand the principle of sampling of continuous time analog signal

LIST OF EXPERIMENTS

LAB SAFETY & USAGE INSTRUCTIONS

It is necessary follow safety precautions while using the electric supply to avoid the serious problems like shocks and fire hazards.

- Insulation of the conductor used must be proper and in good condition. If it is not so the current carried by the conductors may leak out. The person coming in contact with such faculty insulated conductors may receive a shock
- Earth connection should be always maintained in proper condition.
- Make the mains supply switch off and remove the fuses before starting work with any installation.
- Fuses must have correct ratings
- Use rubber soled shoes while working. Use some wooden slipper under the feet. This removes the contact with earth.
- Use rubber gloves while touching any terminals or removing insulation layer from conductor.
- Use a line tester to check whether a live terminal carries any current still better method is to use a test lamp.
- Always use insulated screw drivers, cutting players, line tester etc.
- Never touch two different terminals at the same time.
- Never remove the plug by pulling the wires connected to it.

1. Eating, drinking, or smoking is strictly prohibited inside the computer laboratory.
2. Do not bring liquids near computers, electrical connections, or peripherals.
3. Ensure proper handling of computers, keyboards, mouse, and UPS systems.
4. Students should maintain discipline and follow instructions given by the lab instructor.
5. Report any hardware, software, or network issues immediately to the instructor or lab staff.
6. Do not install, uninstall, or modify software settings without permission.
7. Use only authorized datasets and software provided for the laboratory exercises.
8. Save work frequently and maintain backup copies of files to avoid data loss.
9. Switch off systems properly after use and log out of user accounts.
10. Bags and personal belongings should be kept in designated areas.
11. Maintain cleanliness and ensure proper cable management around workstations.

EXPT. NO: 1

AIM: Basic Signals and Signal Graphing: a) unit Step, b) Rectangular, c) standard triangle d) sinusoidal and e) Exponential signal.

OBJECTIVE: To understand different basic signals and their properties

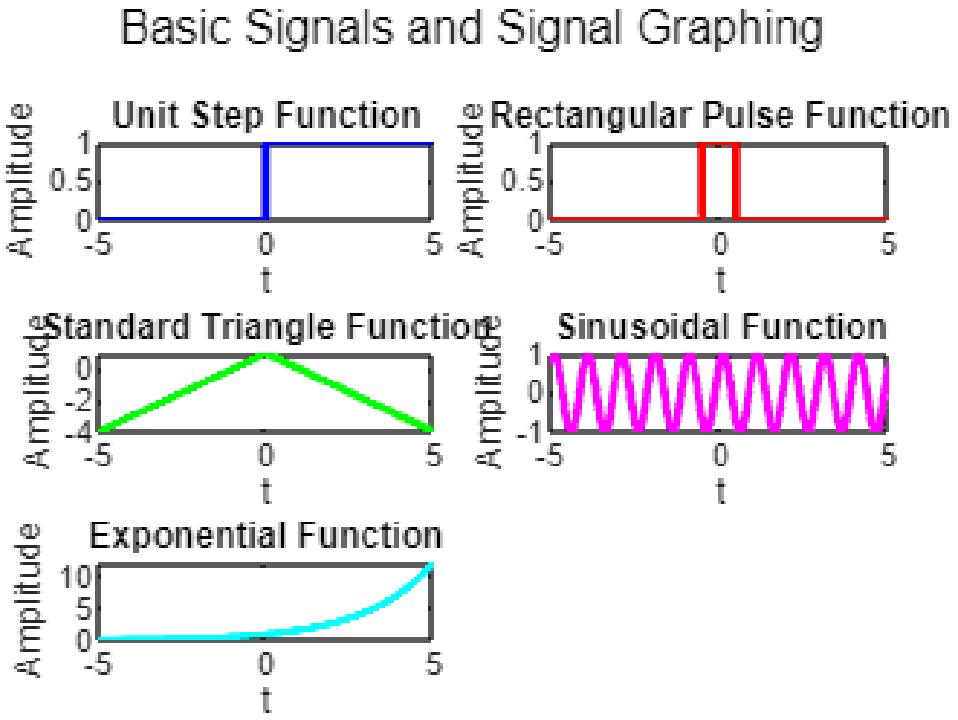
TOOLS REQUIRED: MATLAB

Program:

```
% Define the time vector
t = -5:0.01:5;

% a) Unit Step Function
u = t >= 0;

% b) Rectangular Pulse Function
rect = abs(t) <= 0.5;

% c) Standard Triangle Function
triangle = 1 - abs(t);

% d) Sinusoidal Function
A = 1; % Amplitude
omega = 2*pi; % Angular frequency
phi = pi/4; % Phase angle
sinusoid = A*sin(omega*t + phi);

% e) Exponential Function
A_exp = 1; % Amplitude
k = 0.5; % Exponential rate
exponential = A_exp * exp(k*t);

% Plotting
subplot(3,2,1);
plot(t, u, 'b', 'LineWidth', 2);
title('Unit Step Function');
xlabel('t');
ylabel('Amplitude');

subplot(3,2,2);
plot(t, rect, 'r', 'LineWidth', 2);
title('Rectangular Pulse Function');
xlabel('t');
ylabel('Amplitude');

subplot(3,2,3);
plot(t, triangle, 'g', 'LineWidth', 2);
title('Standard Triangle Function');
xlabel('t');
ylabel('Amplitude');

subplot(3,2,4);
plot(t, sinusoid, 'm', 'LineWidth', 2);
title('Sinusoidal Function');
xlabel('t');
```

```
ylabel('Amplitude');

subplot(3,2,5);
plot(t, exponential, 'c', 'LineWidth', 2);
title('Exponential Function');
xlabel('t');
ylabel('Amplitude');

% Adjusting subplot spacing
sgtitle('Basic Signals and Signal Graphing');
```

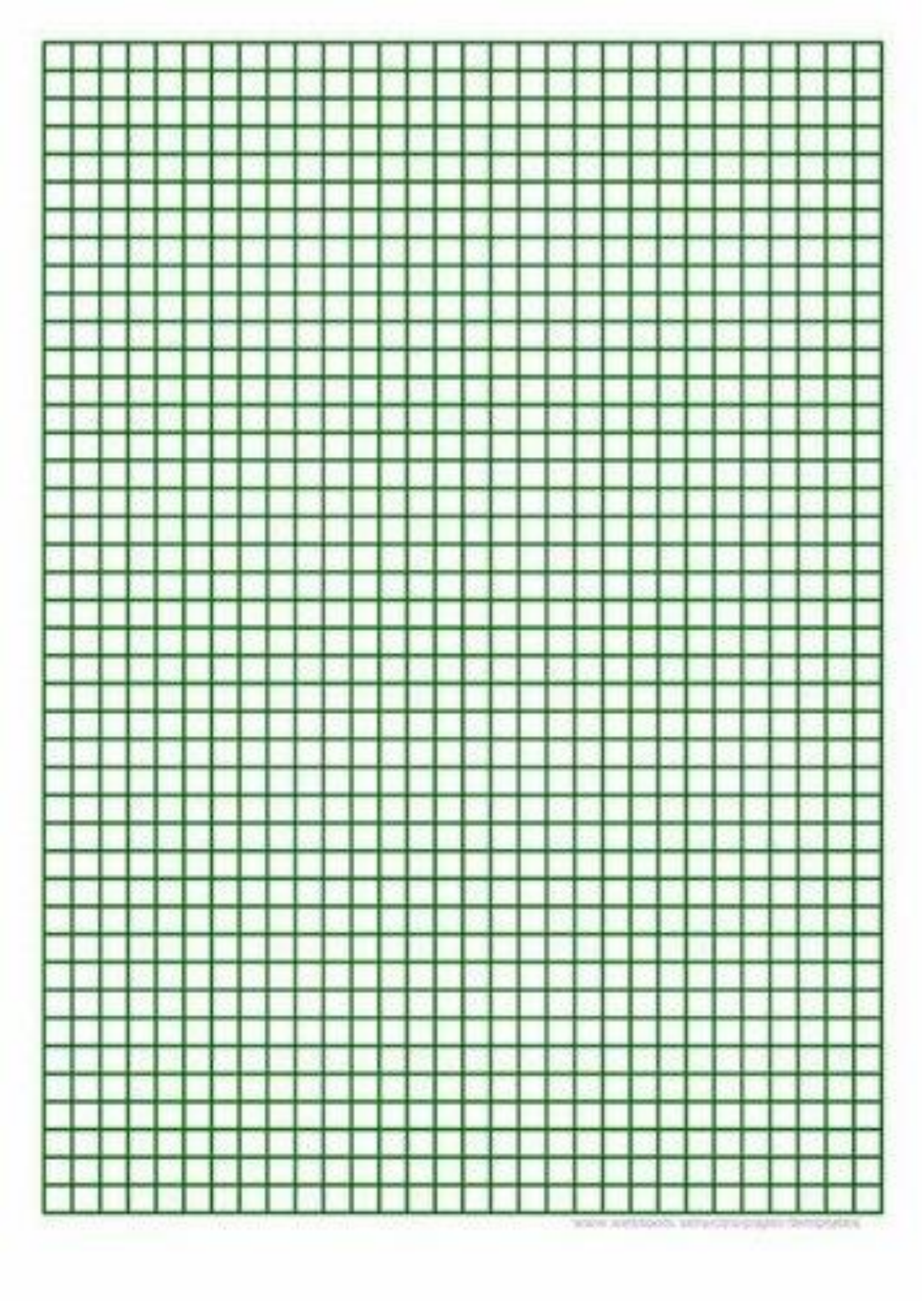
RESULT:



Observation Graph:

Scale: X- Axis:

Y-Axis:



CONCLUSION: Different signals are generated and Observed

EXPT. NO: 2

AIM: Illustration of signal representation in time and frequency domains for a rectangular pulse.

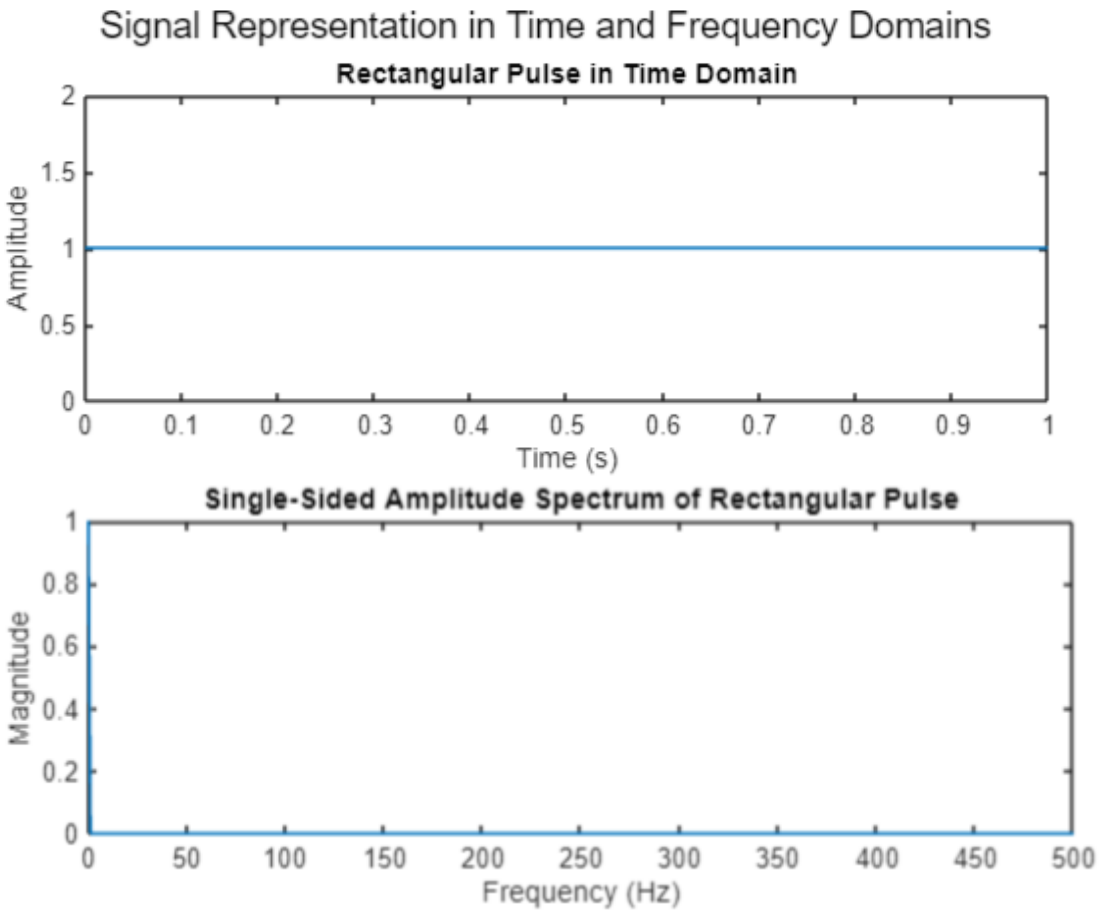
OBJECTIVE: To understand signal representation in time and frequency domains

TOOLS REQUIRED: MATLAB

Program:

```
% Define the parameters
Fs = 1000; % Sampling frequency (Hz)
T = 1/Fs; % Sampling period
L = 1000; % Length of signal
t = (0:L-1)*T; % Time vector

% Generate rectangular pulse
rect_pulse = rectpuls(t - 0.5, 1);

% Compute the Fourier transform
Y = fft(rect_pulse);

% Compute the two-sided spectrum
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);

% Define the frequency domain
f = Fs*(0:(L/2))/L;

% Plotting time domain
subplot(2,1,1);
plot(t, rect_pulse);
title('Rectangular Pulse in Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plotting frequency domain
subplot(2,1,2);
plot(f, P1);
title('Single-Sided Amplitude Spectrum of Rectangular Pulse');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Adjusting subplot spacing
sgtitle('Signal Representation in Time and Frequency Domains');
```

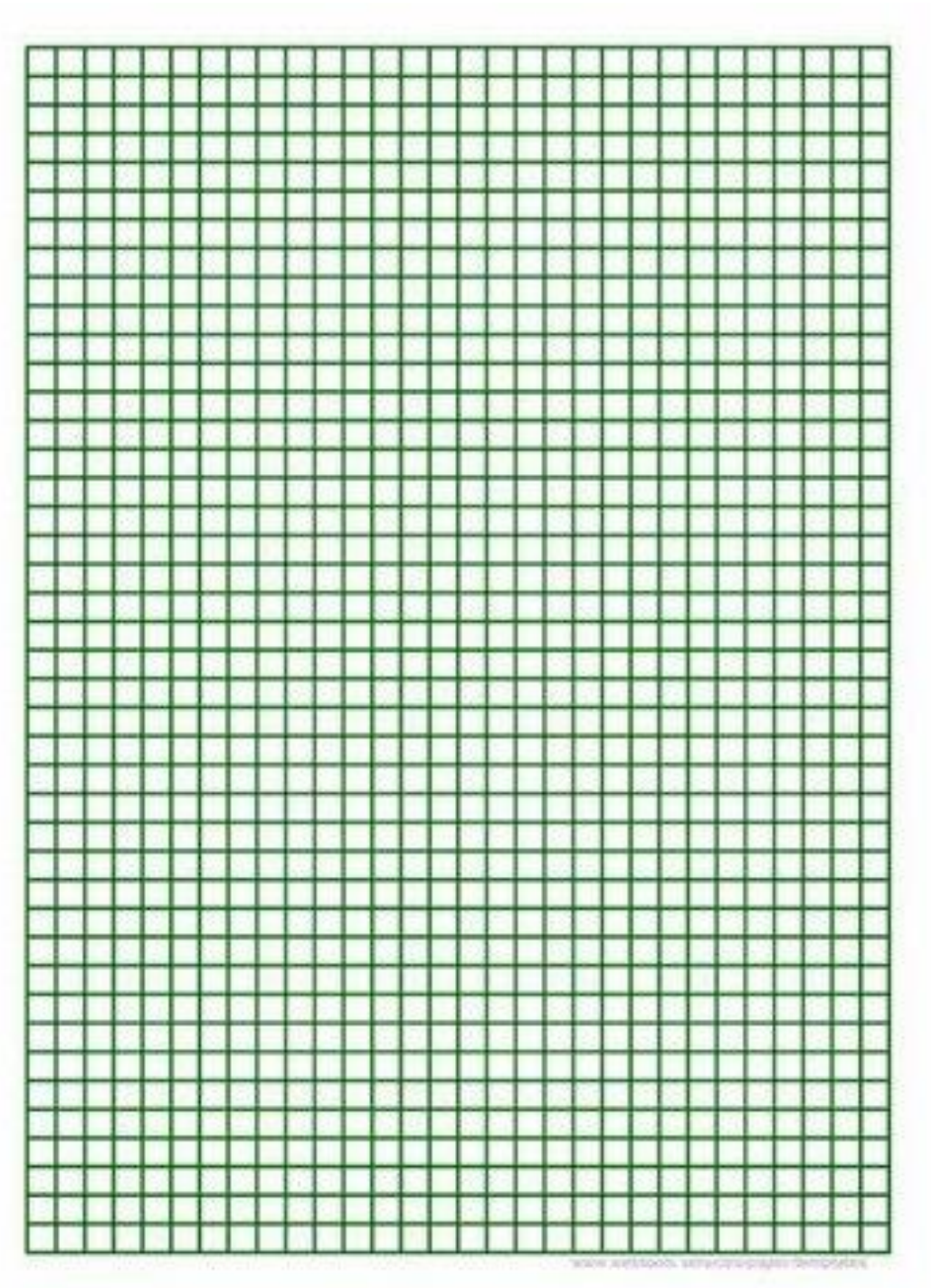
RESULT:


Signal Representation in Time and Frequency Domains
Rectangular Pulse in Time Domain
Amplitude
Time (s)
Single-Sided Amplitude Spectrum of Rectangular Pulse
Magnitude
Frequency (Hz)

Observation Graph:

Scale: X- Axis:

Y-Axis:



CONCLUSION: Signal representation in time and frequency domains for a rectangular pulse is verified

EXPT. NO: 3

AIM: Amplitude Modulation and demodulation: Generation and display the relevant signals and its spectrums.

OBJECTIVE: To study and understand Amplitude Modulation and demodulation using its spectrum

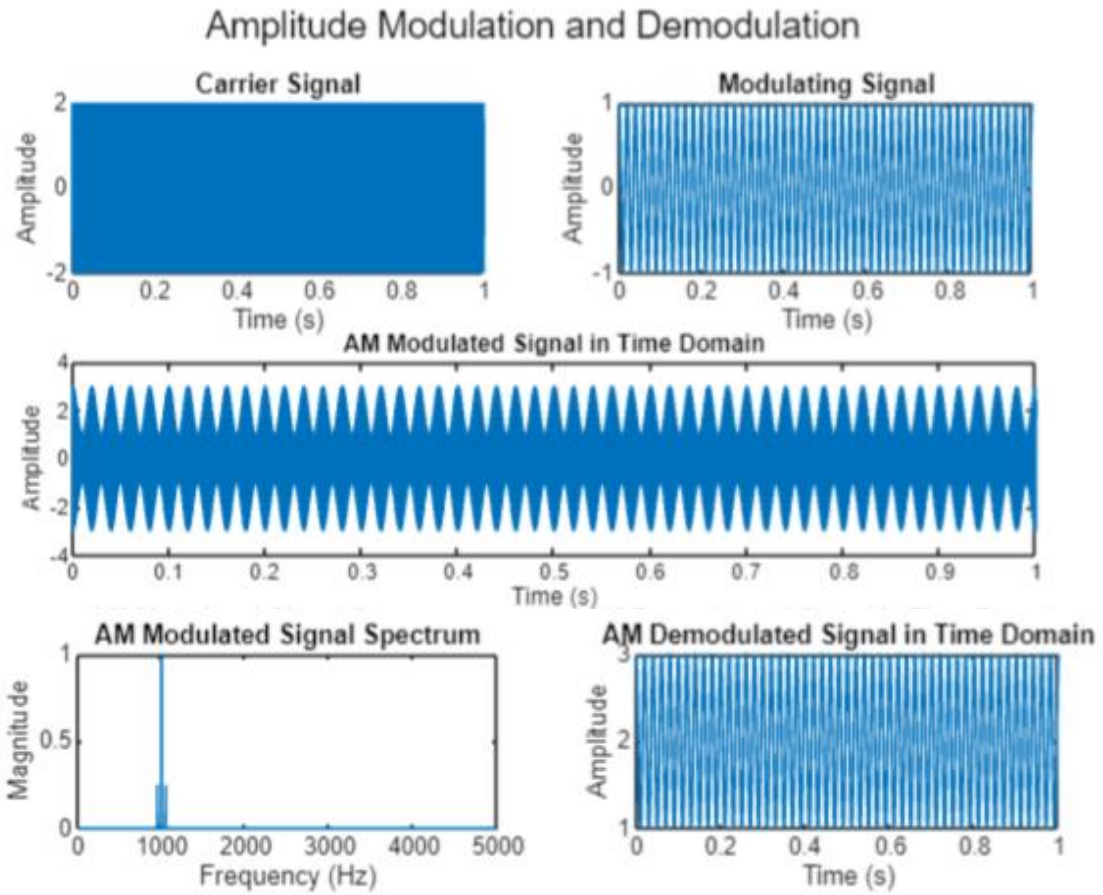
TOOLS REQUIRED: MATLAB

PROGRAM:

```
% Parameters
Fc = 1000; % Carrier frequency (Hz)
Fm = 50;   % Modulating frequency (Hz)
Am = 1;    % Modulating signal amplitude
Ac = 2;    % Carrier signal amplitude

% Time vector
Fs = 10*Fc; % Sampling frequency (must be higher than Nyquist frequency)
t = 0:1/Fs:1-(1/Fs); % Time vector

% Generate carrier and modulating signals
carrier = Ac * cos(2*pi*Fc*t);
modulating = Am * cos(2*pi*Fm*t);

% Amplitude modulation
modulated_signal = (1 + (Am/Ac) * modulating) .* carrier;

% Plotting original signals
subplot(3,2,1);
plot(t, carrier);
title('Carrier Signal');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(3,2,2);
plot(t, modulating);
title('Modulating Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Plotting modulated signal in time domain
subplot(3,2,[3,4]);
plot(t, modulated_signal);
title('AM Modulated Signal in Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute FFT of modulated signal
L = length(t);
f = Fs*(0:(L/2))/L;
modulated_signal_fft = fft(modulated_signal);
P2 = abs(modulated_signal_fft/L);
P1 = P2(1:L/2+1);

% Plotting modulated signal spectrum
subplot(3,2,5);
```

```
plot(f, P1);
title('AM Modulated Signal Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Amplitude demodulation
demodulated_signal = abs(hilbert(modulated_signal));

% Plotting demodulated signal in time domain
subplot(3,2,6);
plot(t, demodulated_signal);
title('AM Demodulated Signal in Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Adjusting subplot spacing
sgtitle('Amplitude Modulation and Demodulation');
```

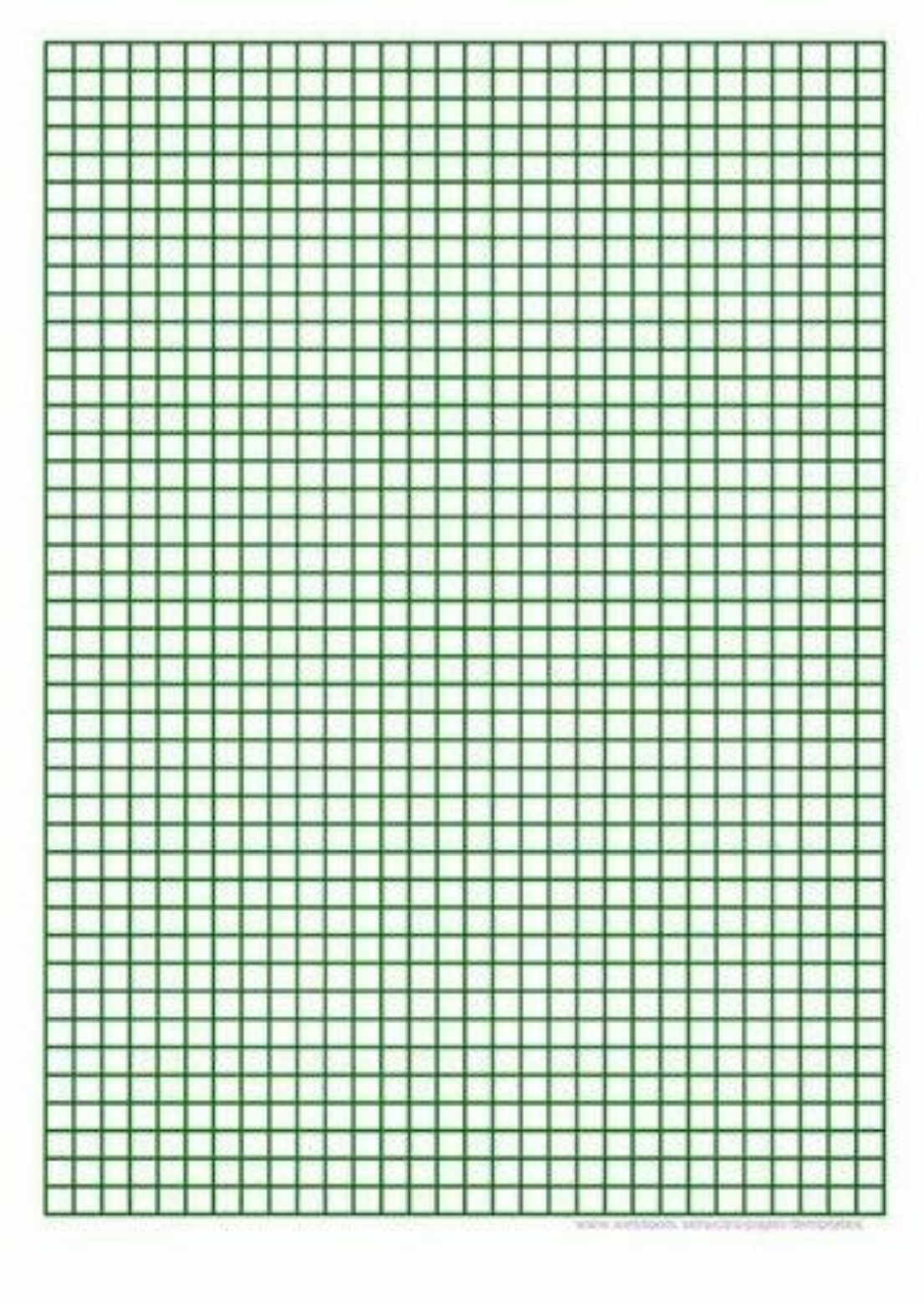
RESULT:



Observation Graph:

Scale: X- Axis:

Y-Axis:



CONCLUSION: Amplitude Modulation and Demodulation of a signal are verified

EXPT. NO: 4

AIM: Frequency Modulation and demodulation: Generation and display the relevant signals and its spectrums

OBJECTIVE: To study and understand Frequency Modulation and demodulation

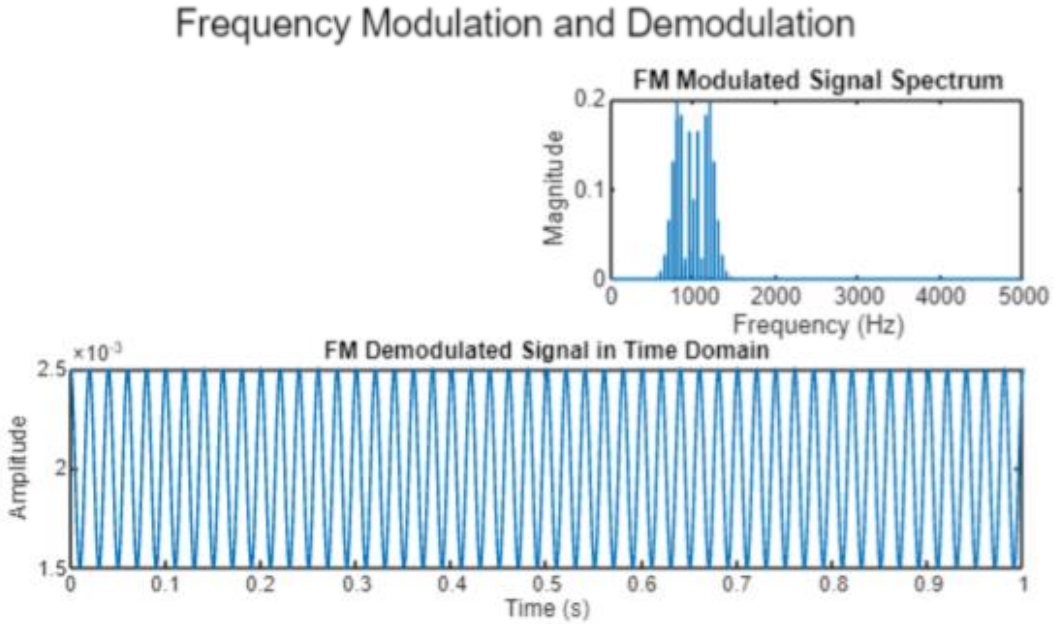
TOOLS REQUIRED: MATLAB

PROGRAM:

```
% Parameters
Fc = 1000; % Carrier frequency (Hz)
Fm = 50;   % Modulating frequency (Hz)
Am = 1;    % Modulating signal amplitude
Ac = 1;    % Carrier signal amplitude
beta = 5;  % Modulation index

% Time vector
Fs = 10*Fc; % Sampling frequency (must be higher than Nyquist frequency)
t = 0:1/Fs:1-(1/Fs); % Time vector

% Generate modulating signal
modulating = Am * cos(2*pi*Fm*t);

% Frequency modulation
modulated_signal = Ac * cos(2*pi*Fc*t + beta*sin(2*pi*Fm*t));

% Plotting modulating signal
subplot(3,2,1);
plot(t, modulating);
title('Modulating Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Plotting modulated signal in time domain
subplot(3,2,[2,3]);
plot(t, modulated_signal);
title('FM Modulated Signal in Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute FFT of modulated signal
L = length(t);
f = Fs*(0:(L/2))/L;
modulated_signal_fft = fft(modulated_signal);
P2 = abs(modulated_signal_fft/L);
P1 = P2(1:L/2+1);

% Plotting modulated signal spectrum
subplot(3,2,4);
plot(f, P1);
title('FM Modulated Signal Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Frequency demodulation using FM discriminator
delta_phi = diff(unwrap(angle(hilbert(modulated_signal))));
```

```
demodulated_signal = delta_phi/(2*pi*Fm);

% Plotting demodulated signal in time domain
subplot(3,2,[5,6]);
plot(t(1:end-1), demodulated_signal);
title('FM Demodulated Signal in Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Adjusting subplot spacing
sgtitle('Frequency Modulation and Demodulation');
```

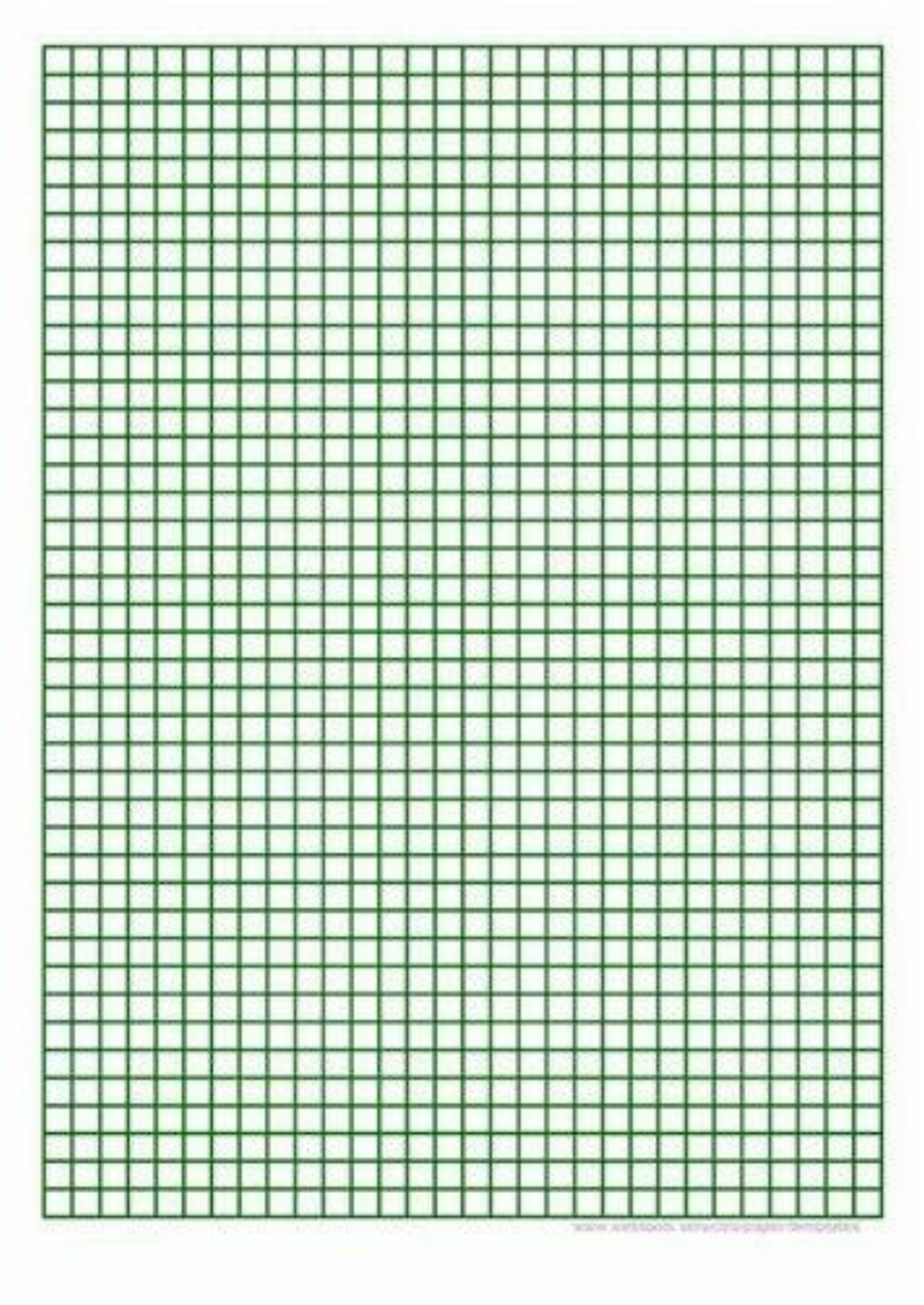
RESULT:



Observation Graph:

Scale: X- Axis:

Y-Axis:



CONCLUSION: Frequency Modulation and Demodulation of a signal are verified

EXPT. NO: 5

AIM: Sampling and reconstruction of low pass signals. Display the signals and its spectrum.

OBJECTIVE: To study concept of Sampling and reconstruction of low pass signals

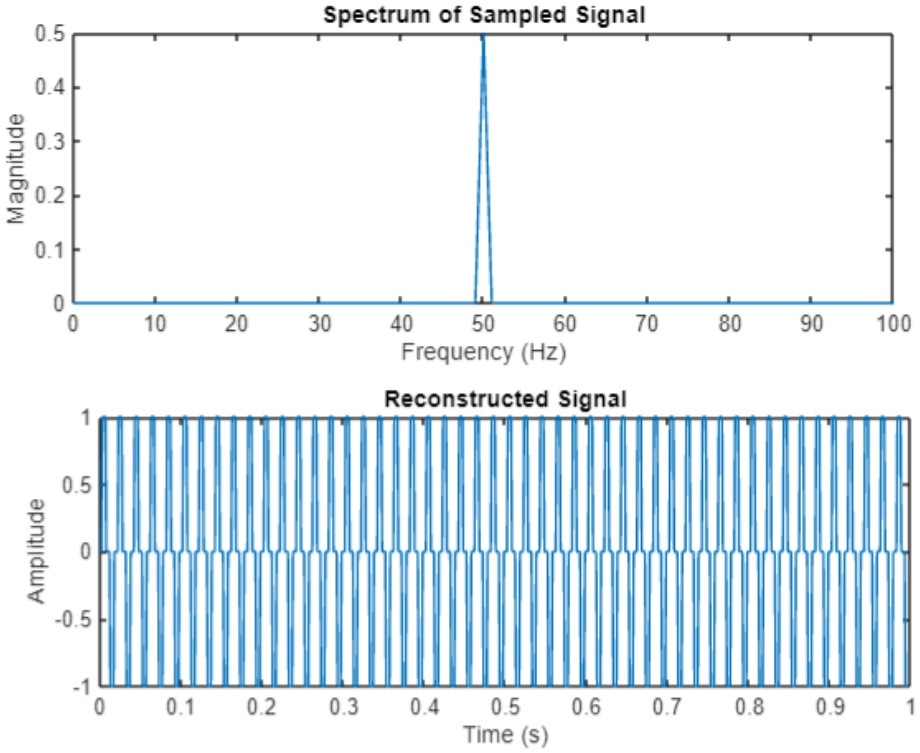
TOOLS REQUIRED: MATLAB

PROGRAM:

```
% Parameters
Fs = 1000;          % Sampling frequency (Hz)
T = 1/Fs;           % Sampling period
duration = 1;       % Duration of the signal (s)
t = 0:T:duration-T; % Time vector

% Define a low-pass signal (e.g., a sinusoid)
f_signal = 50;      % Signal frequency (Hz)
signal = sin(2*pi*f_signal*t);

% Plot original signal
subplot(3,1,1);
plot(t, signal);
title('Original Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute the Fourier transform of the original signal
L = length(signal);
f = Fs*(0:(L/2))/L;
signal_fft = fft(signal);
P2 = abs(signal_fft/L);
P1 = P2(1:L/2+1);

% Plot the spectrum of the original signal
subplot(3,1,2);
plot(f, P1);
title('Spectrum of Original Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Sampling the signal
Fs_new = 200;               % New sampling frequency (Hz)
T_new = 1/Fs_new;           % New sampling period
t_new = 0:T_new:duration-T_new; % New time vector
signal_sampled = interp1(t, signal, t_new, 'nearest');

% Plot sampled signal
subplot(3,1,3);
stem(t_new, signal_sampled, 'r');
hold on;
plot(t, signal, 'b');
hold off;
title('Sampled Signal');
xlabel('Time (s)');
ylabel('Amplitude');
legend('Sampled Signal', 'Original Signal');
```

```
% Compute the Fourier transform of the sampled signal
L = length(signal_sampled);
f_new = Fs_new*(0:(L/2))/L;
signal_sampled_fft = fft(signal_sampled);
P2_new = abs(signal_sampled_fft/L);
P1_new = P2_new(1:L/2+1);

% Plot the spectrum of the sampled signal
figure;
subplot(2,1,1);
plot(f_new, P1_new);
title('Spectrum of Sampled Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Reconstruction of the signal using zero-order hold
reconstructed_signal = interp1(t_new, signal_sampled, t, 'nearest');

% Plot reconstructed signal
subplot(2,1,2);
plot(t, reconstructed_signal);
title('Reconstructed Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```

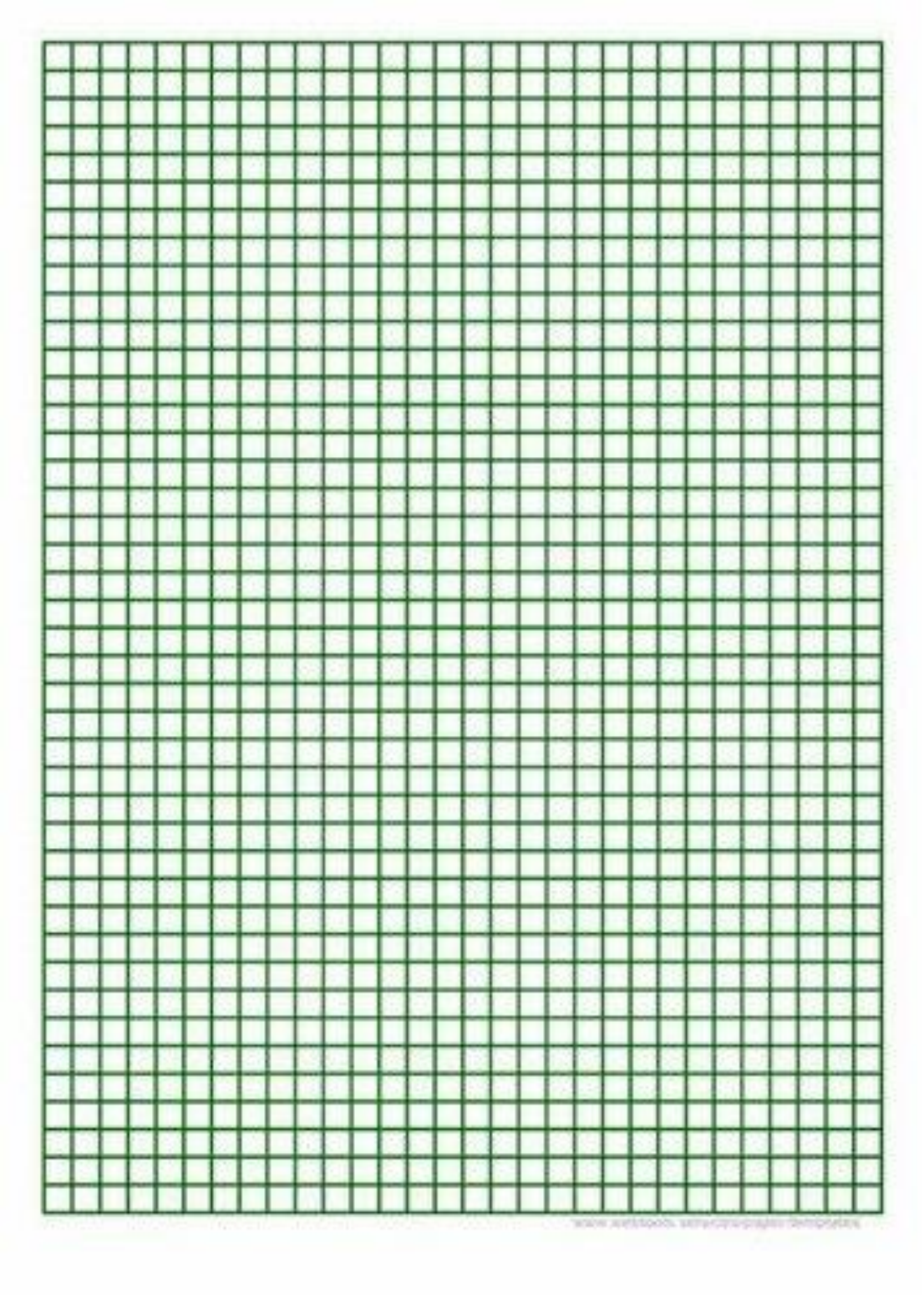
RESULT:



Observation Graph:

Scale: X- Axis:

Y-Axis:



CONCLUSION: Sampling and reconstruction of low pass signals is observed

EXPT. NO: 6

AIM: Time Division Multiplexing and demultiplexing

OBJECTIVE: To understand Time Division Multiplexing and demultiplexing of Signals.

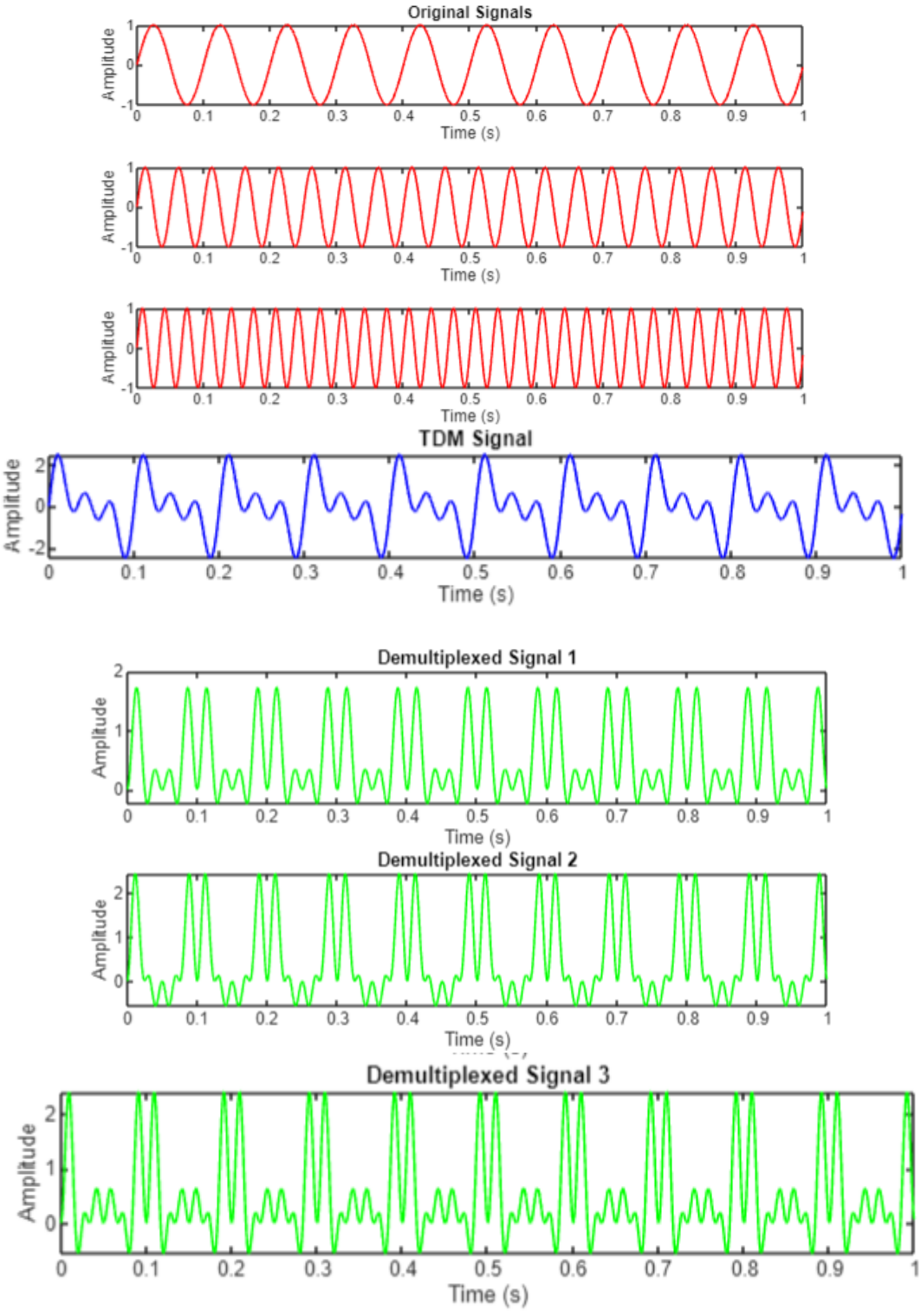
TOOLS REQUIRED: MATLAB

PROGRAM:

```
% Parameters
Fs = 1000;          % Sampling frequency (Hz)
T = 1/Fs;           % Sampling period
duration = 1;       % Duration of each signal (s)
t = 0:T:duration-T; % Time vector

% Define multiple signals (for illustration purposes)
frequencies = [10, 20, 30]; % Frequencies of the signals (Hz)
signals = zeros(length(frequencies), length(t));

% Generate signals
for i = 1:length(frequencies)
    signals(i, :) = sin(2*pi*frequencies(i)*t);
end

% Plot original signals
figure;
subplot(length(frequencies)+1,1,1);
plot(t, signals(1,:), 'r');
title('Original Signals');
xlabel('Time (s)');
ylabel('Amplitude');
hold on;

for i = 2:length(frequencies)
    subplot(length(frequencies)+1,1,i);
    plot(t, signals(i,:), 'r');
    xlabel('Time (s)');
    ylabel('Amplitude');
    hold on;
end

% Time Division Multiplexing (TDM)
tdm_signal = sum(signals);

% Plot TDM signal
subplot(length(frequencies)+1,1,length(frequencies)+1);
plot(t, tdm_signal, 'b');
xlabel('Time (s)');
ylabel('Amplitude');
title('TDM Signal');

% Demultiplexing
demux_signals = zeros(length(frequencies), length(t));
for i = 1:length(frequencies)
    demux_signals(i, :) = tdm_signal .* signals(i,:);
end
```

```
% Plot demultiplexed signals
figure;
for i = 1:length(frequencies)
    subplot(length(frequencies),1,i);
    plot(t, demux_signals(i,:), 'g');
    xlabel('Time (s)');
    ylabel('Amplitude');
    title(['Demultiplexed Signal ', num2str(i)]);
end
```

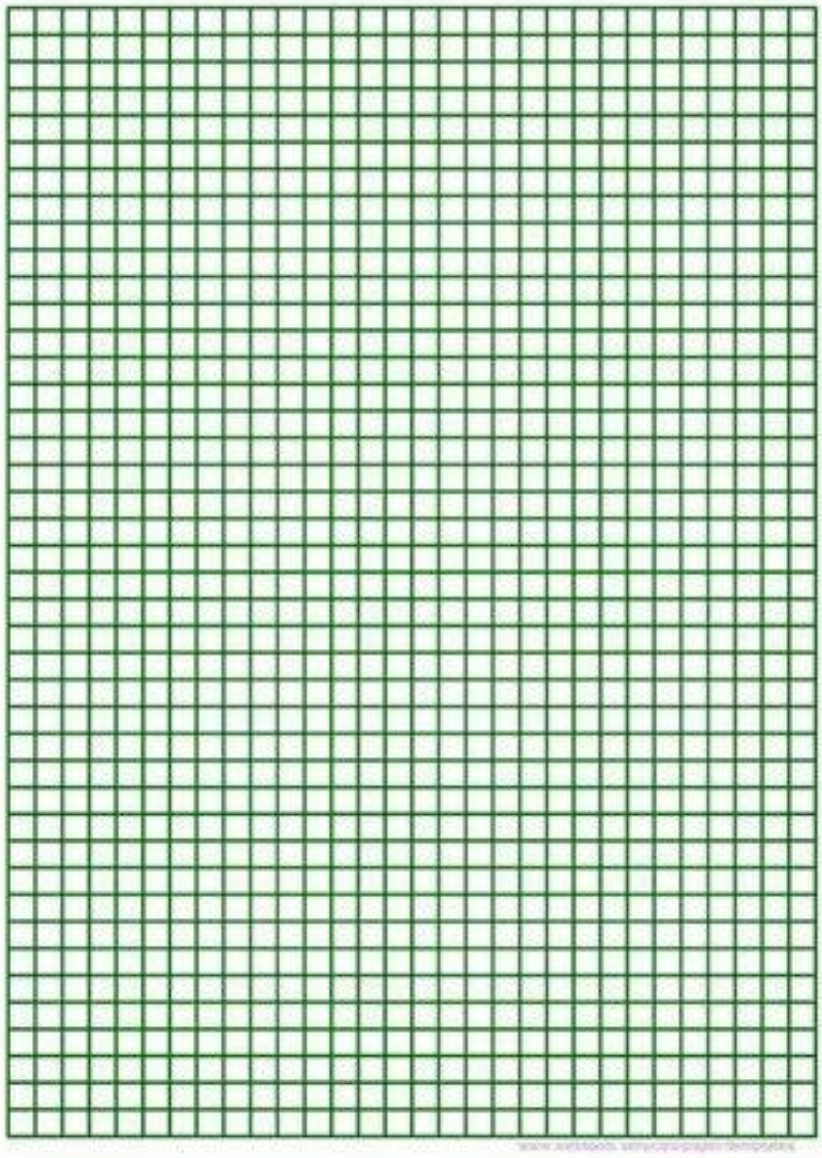
Result :



Observation Graph:

Scale: X- Axis:

Y-Axis:



Conclusion: Division Multiplexing and demultiplexing of Signals is verified

EXPT. NO: 7

AIM: PCM Illustration: Sampling, Quantization and Encoding

OBJECTIVE: To understand PCM Illustration: Sampling, Quantization and Encoding

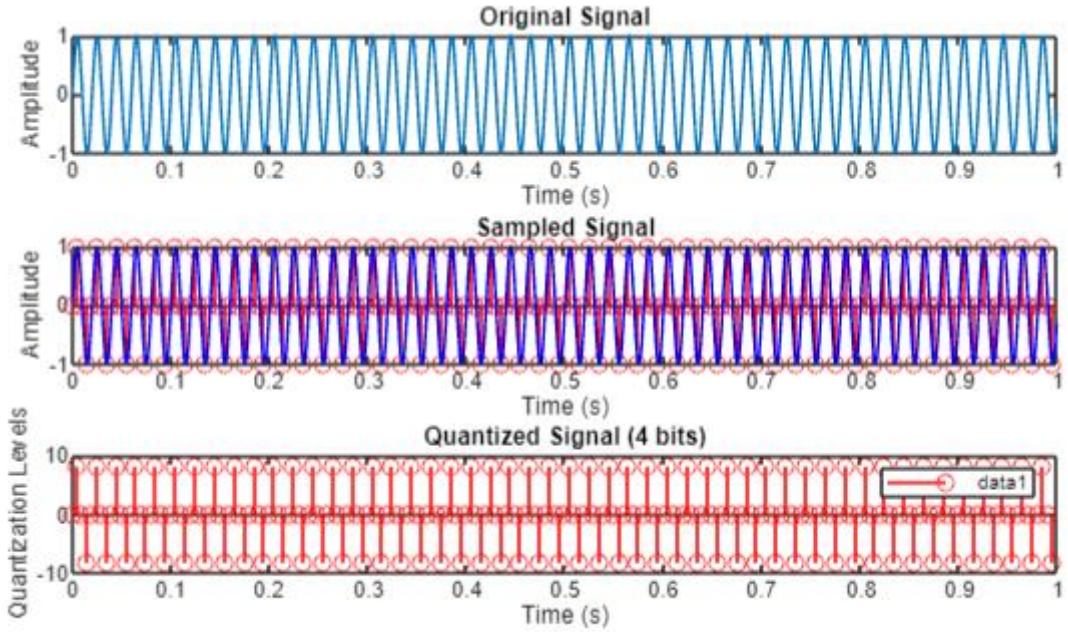
TOOLS REQUIRED: MATLAB

PROGRAM:

```
% Parameters
Fs = 1000;          % Sampling frequency (Hz)
T = 1/Fs;           % Sampling period
duration = 1;       % Duration of the signal (s)
t = 0:T:duration-T; % Time vector

% Define a low-pass signal (e.g., a sinusoid)
f_signal = 50;      % Signal frequency (Hz)
signal = sin(2*pi*f_signal*t);

% Plot original signal
subplot(4,1,1);
plot(t, signal);
title('Original Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Sampling the signal
Fs_new = 200;                % New sampling frequency (Hz)
T_new = 1/Fs_new;            % New sampling period
t_new = 0:T_new:duration-T_new; % New time vector
signal_sampled = interp1(t, signal, t_new, 'nearest');

% Plot sampled signal
subplot(4,1,2);
stem(t_new, signal_sampled, 'r');
hold on;
plot(t, signal, 'b');
hold off;
title('Sampled Signal');
xlabel('Time (s)');
ylabel('Amplitude');
legend('Sampled Signal', 'Original Signal');

% Quantization
n_bits = 4; % Number of bits for quantization
quantized_signal = round(signal_sampled * (2^(n_bits-1)));

% Plot quantized signal
subplot(4,1,3);
stem(t_new, quantized_signal, 'r');
title(['Quantized Signal (', num2str(n_bits), ' bits)']);
xlabel('Time (s)');
ylabel('Quantization Levels');

% Encoding
encoded_signal = de2bi(quantized_signal, n_bits, 'left-msb');
```

```
% Reshape encoded signal for plotting
encoded_signal = reshape(encoded_signal', 1, []);

% Plot encoded signal
subplot(4,1,4);
stairs(t_new, encoded_signal);
title('Encoded Signal');
xlabel('Time (s)');
ylabel('Encoded Bits');
```

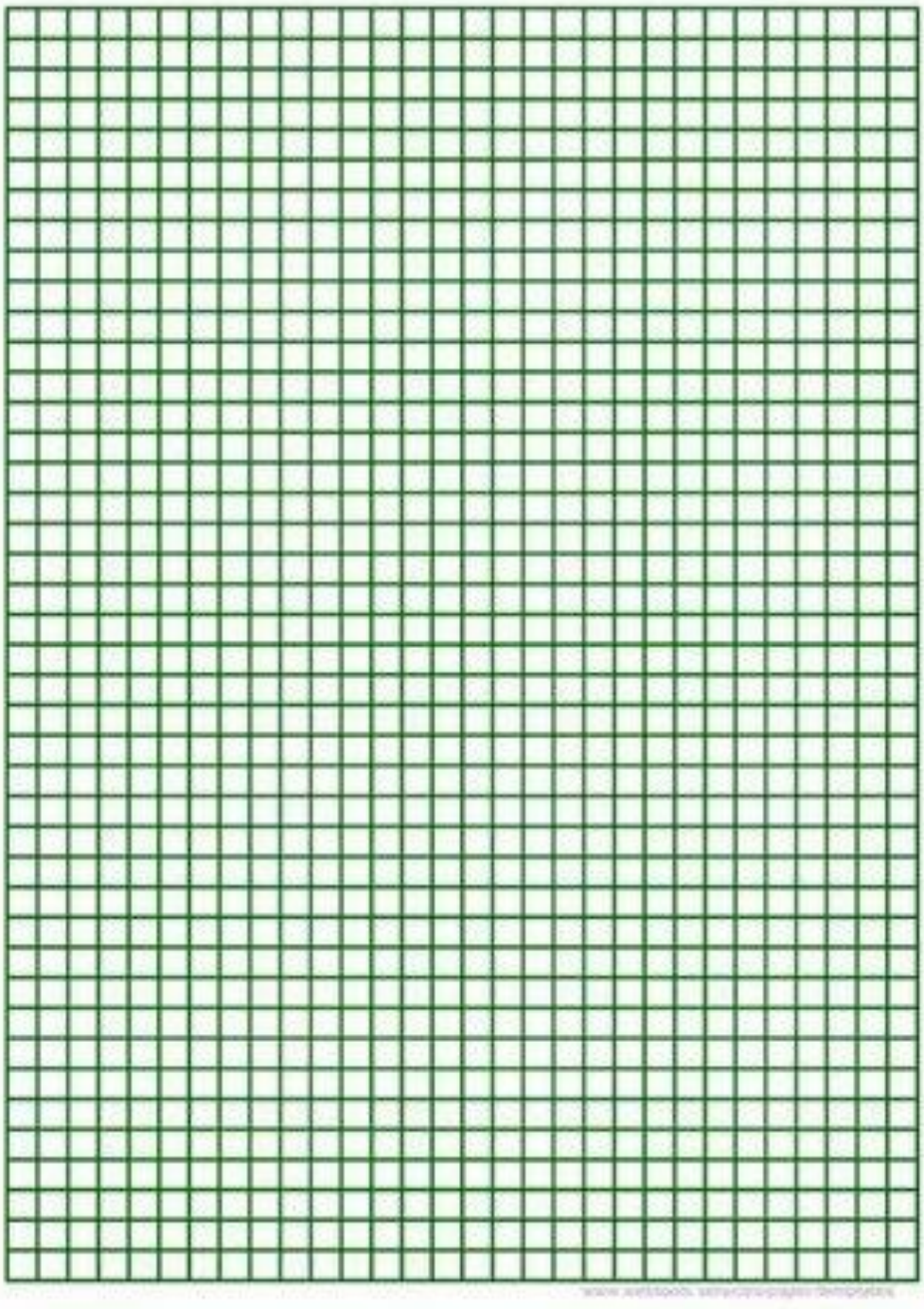
RESULT:



Observation Graph:

Scale: X- Axis:

Y-Axis:



Conclusion : PCM Illustration of a signal is verified

EXPT. NO: 8

AIM: Generate a) NRZ, RZ and Raised cosine pulse, b) Generate and plot eye diagram

OBJECTIVE: To understand Generation of NRZ, RZ and Raised cosine pulse and plot eye diagram

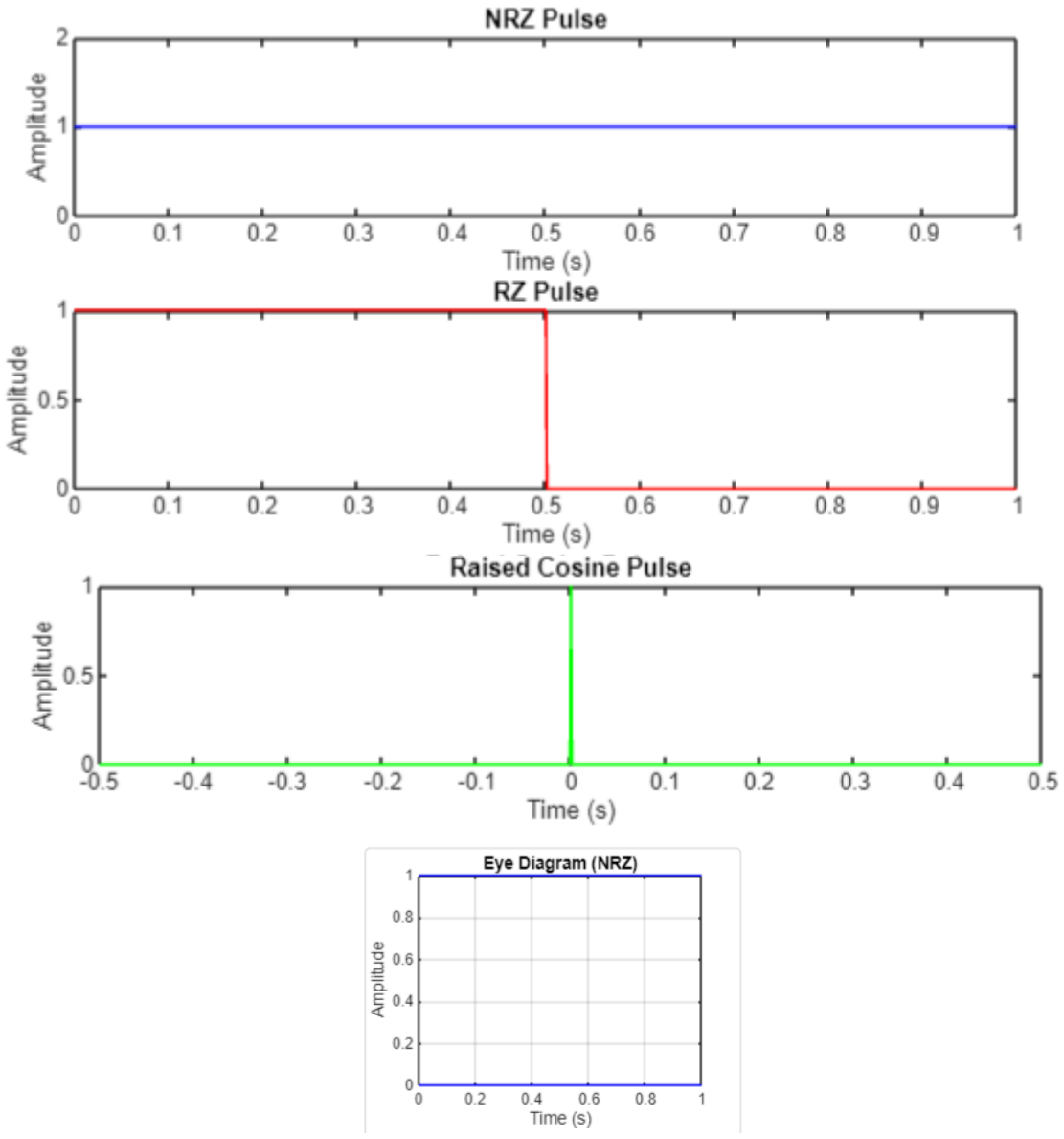
TOOLS REQUIRED: MATLAB

PROGRAM:

```
% Parameters
Fs = 1000;          % Sampling frequency (Hz)
T = 1/Fs;           % Sampling period
duration = 1;       % Duration of the signal (s)
t = 0:T:duration-T; % Time vector

% a) Generate NRZ, RZ, and Raised Cosine pulses

% Generate NRZ pulse
nrz_pulse = ones(1, length(t));

% Generate RZ pulse
rz_pulse = zeros(1, length(t));
rz_pulse(t <= duration/2) = 1;

% Generate Raised Cosine pulse
alpha = 0.5; % Roll-off factor
t_rc = -duration/2:T:duration/2-T;
raised_cosine_pulse = sinc(t_rc/T) .* cos(pi * alpha * t_rc / T) ./ (1 - (2 * alpha * t_rc / T).^2);

% Normalize the pulses
nrz_pulse = nrz_pulse / max(nrz_pulse);
rz_pulse = rz_pulse / max(rz_pulse);
raised_cosine_pulse = raised_cosine_pulse / max(raised_cosine_pulse);

% Plot pulses
figure;
subplot(3,1,1);
plot(t, nrz_pulse, 'b');
title('NRZ Pulse');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(3,1,2);
plot(t, rz_pulse, 'r');
title('RZ Pulse');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(3,1,3);
plot(t_rc, raised_cosine_pulse, 'g');
title('Raised Cosine Pulse');
xlabel('Time (s)');
ylabel('Amplitude');

% b) Generate and plot eye diagram
```

```
% Generate random data
data = randi([0, 1], 1, 1000);

% Generate a sequence of pulses based on NRZ pulse
nrz_pulses = repmat(data, length(t), 1);

% Plot eye diagram
figure;
plot(t, nrz_pulses, 'b');
title('Eye Diagram (NRZ)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

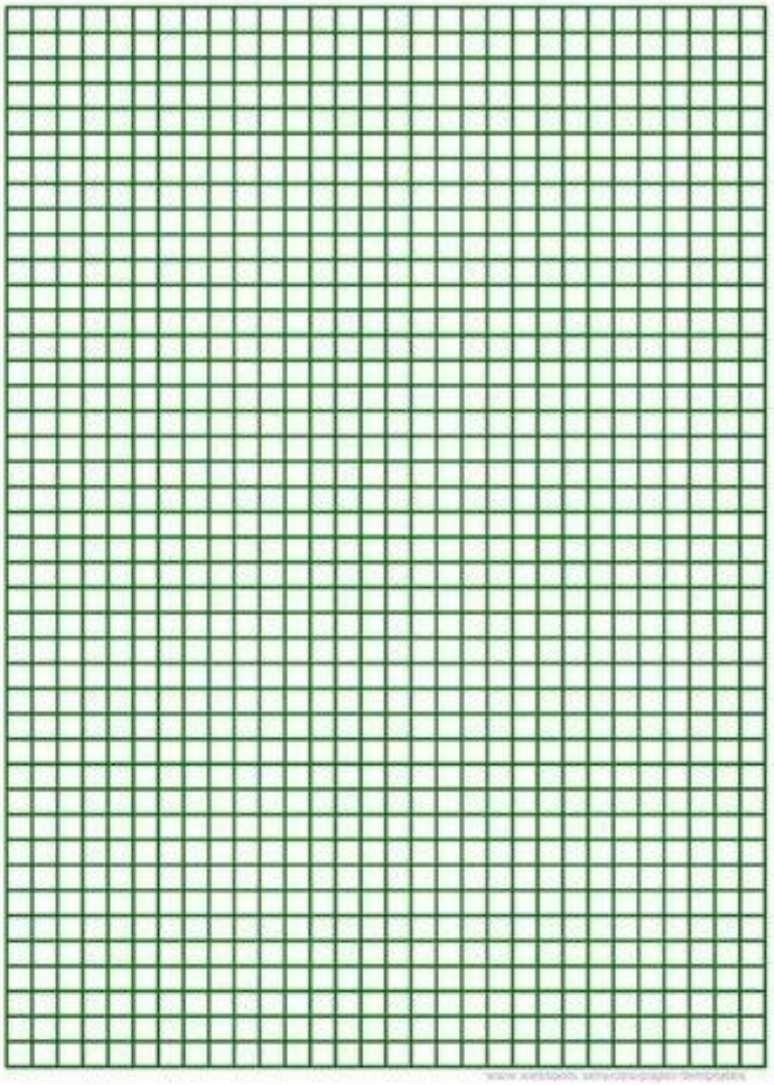
RESULT:



Observation Graph:

Scale: X- Axis:

Y-Axis:



Conclusion : Generation of NRZ, RZ and Raised cosine pulse and plotting eye diagram is verified

EXPT. NO: 9

AIM: Generate the Probability density function of Gaussian distribution function.

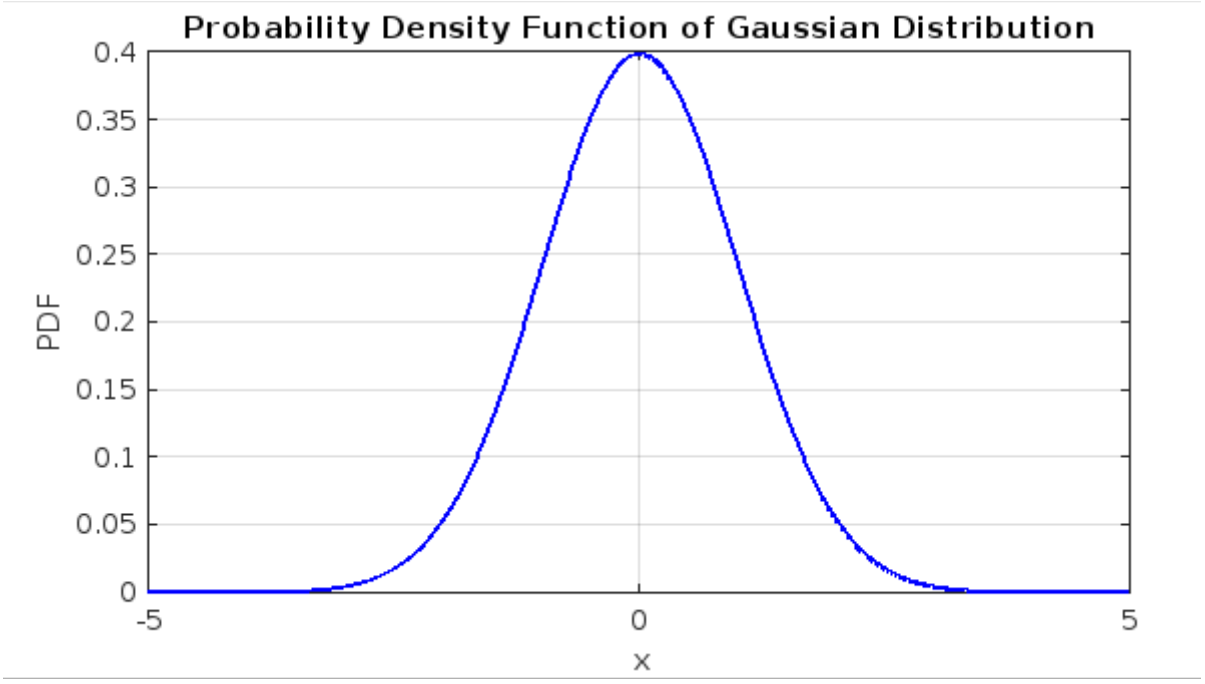
OBJECTIVE: To understand Generation of Probability density function of Gaussian distribution function.

TOOLS REQUIRED: MATLAB

PROGRAM:

```
% Parameters
mu = 0;         % Mean
sigma = 1;      % Standard deviation

% Define range of x values
x = linspace(-5, 5, 1000);

% Compute PDF using the normal distribution formula
pdf = (1/(sqrt(2*pi)*sigma)) * exp(-(x - mu).^2 / (2*sigma^2));

% Plot the PDF
plot(x, pdf, 'b', 'LineWidth', 2);
title('Probability Density Function of Gaussian Distribution');
xlabel('x');
ylabel('PDF');
grid on;
```

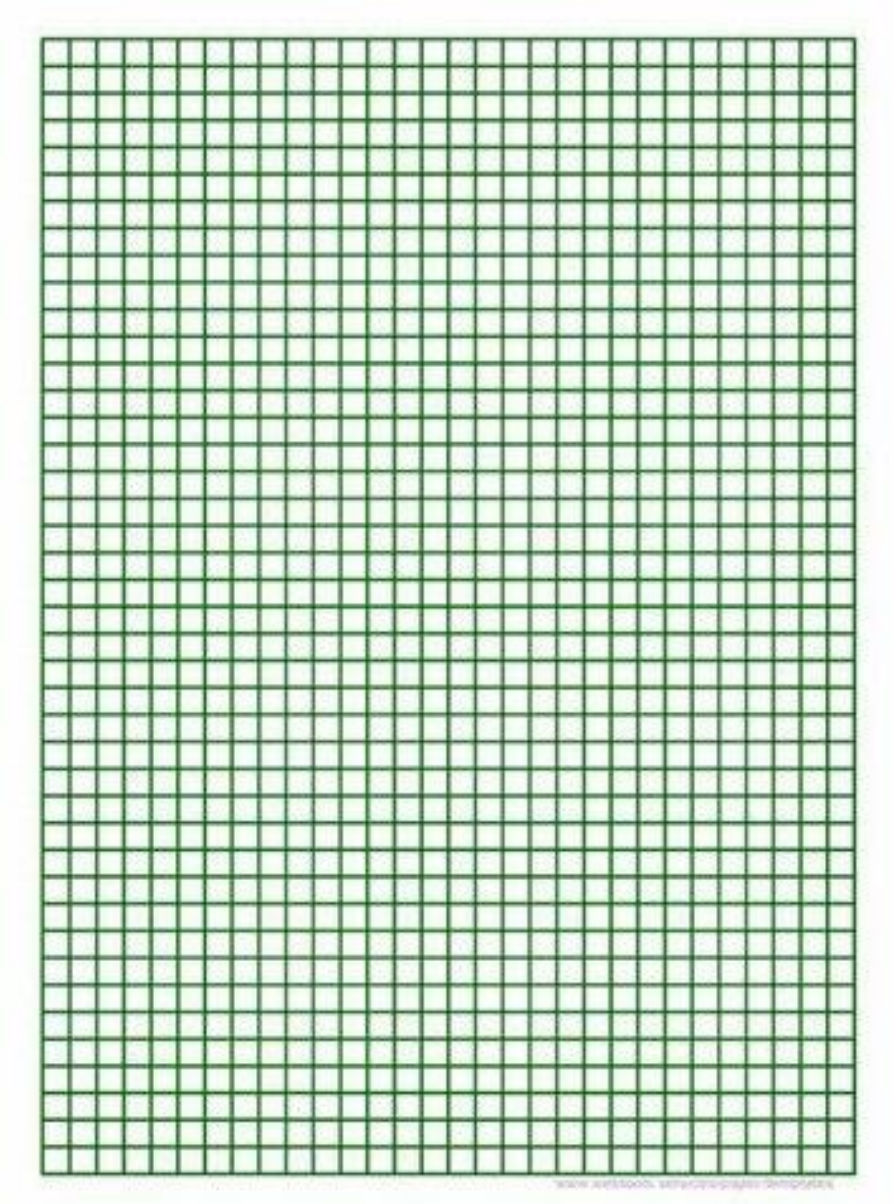
RESULT:



Observation Graph:

Scale: X- Axis:

Y-Axis:



Conclusion: Generation of Probability density function of Gaussian distribution function is verified.

EXPT. NO: 10

AIM: Display the signal and its spectrum of an audio signal.

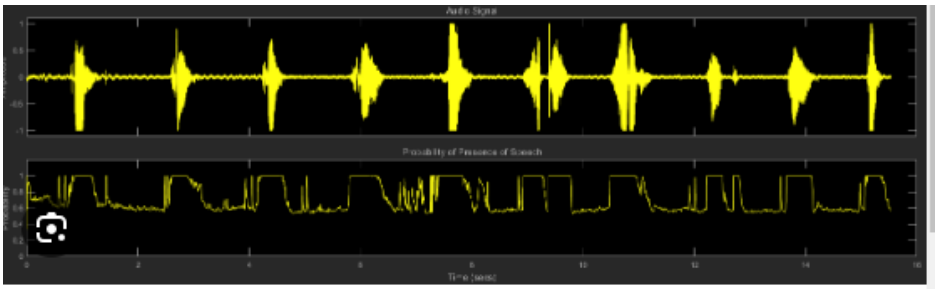
OBJECTIVE: To understand spectrum of an audio signal

TOOLS REQUIRED: MATLAB

PROGRAM:

```
% Load an audio file (replace 'audio_file.wav' with the path to your audio file)
[y, Fs] = audioread('audio_file.wav');

% Plot the audio signal in the time domain
t = (0:length(y)-1) / Fs; % Time vector
subplot(2, 1, 1);
plot(t, y);
title('Audio Signal in Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Compute the Fourier transform of the audio signal
L = length(y);
f = Fs * (0:(L/2)) / L; % Frequency vector
Y = fft(y);
P2 = abs(Y / L);
P1 = P2(1:L/2 + 1);
% Plot the spectrum of the audio signal
subplot(2, 1, 2);
plot(f, P1);
title('Spectrum of Audio Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
% Limit the frequency range for better visualization (optional)
xlim([0, Fs/2]);
```

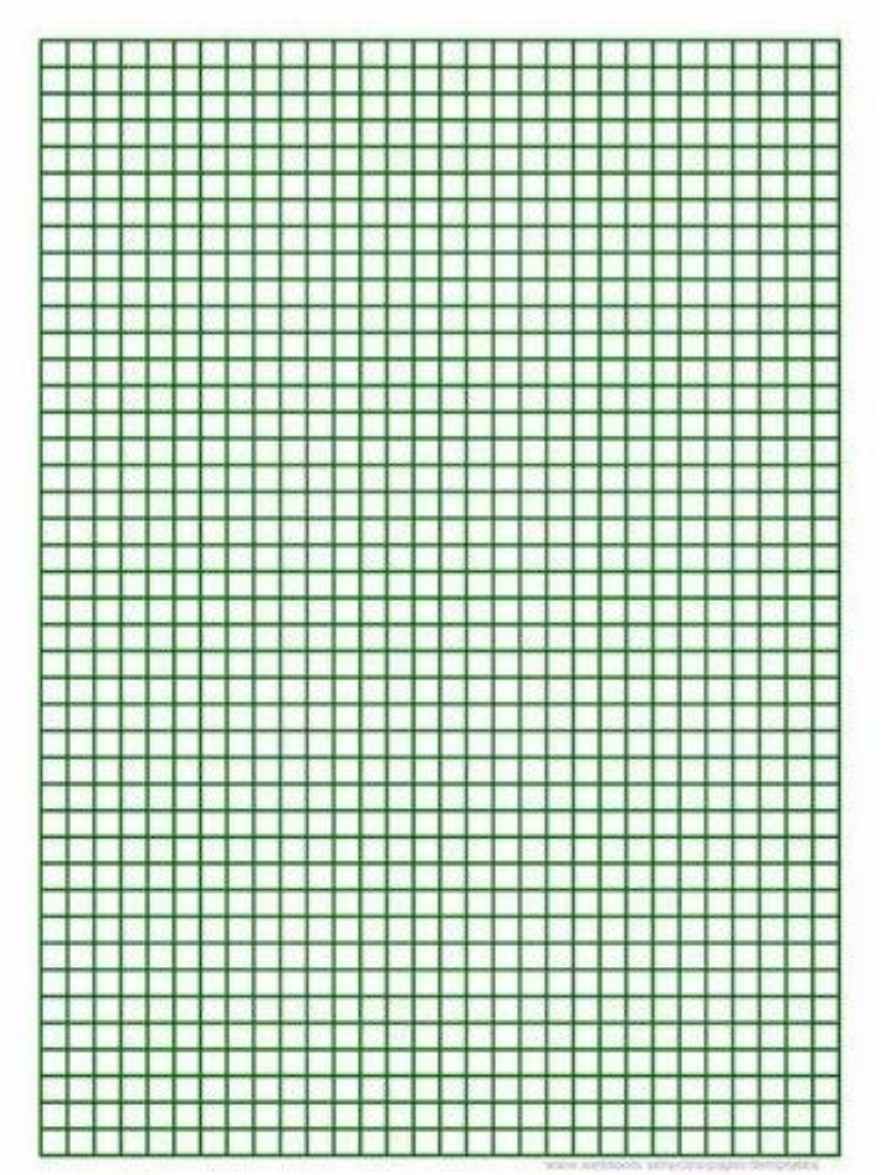
RESULT:



Observation Graph:

Scale: X- Axis:

Y-Axis:



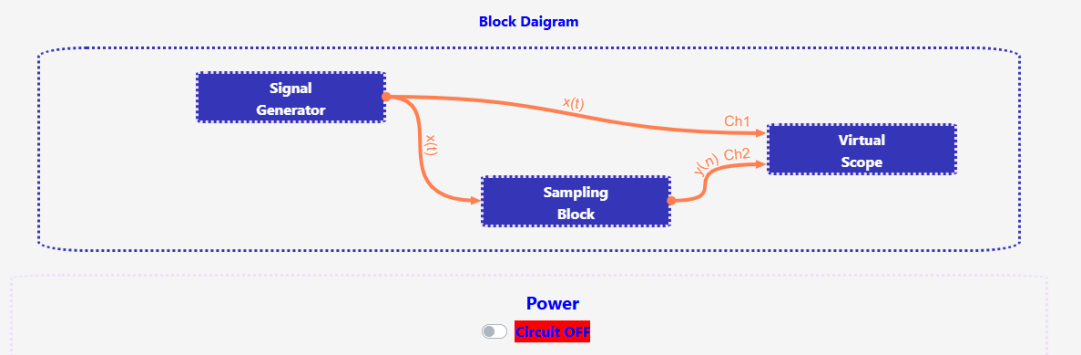
Conclusion: Audio signal and its spectrum are verified.

EXPT. NO: 11

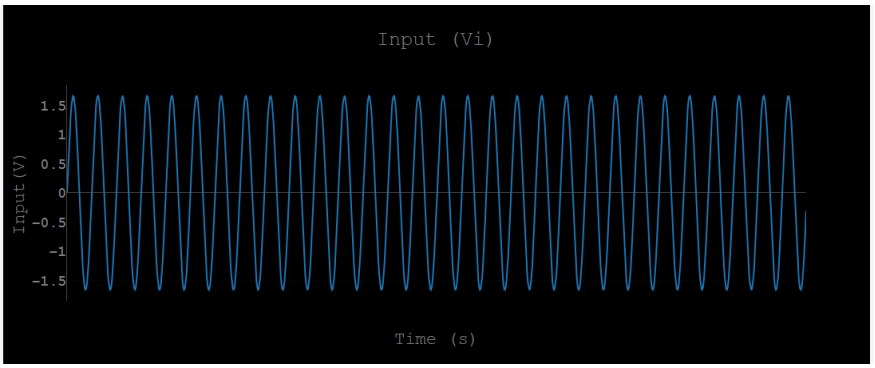
AIM: To study the Sampling Theorem and Effect of Undersampling

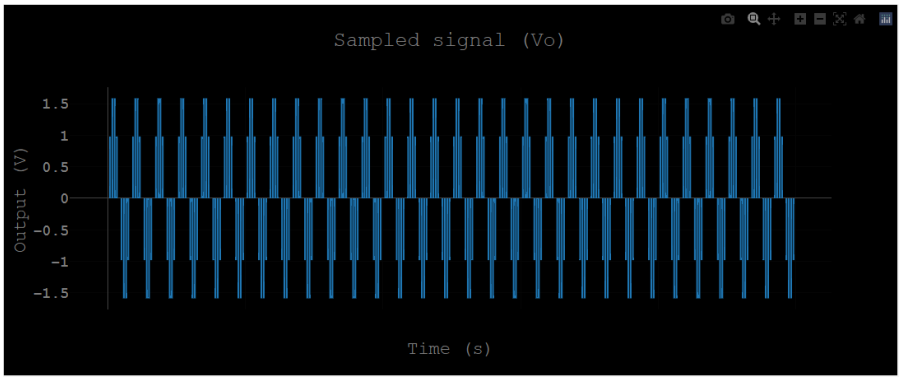
OBJECTIVE: To understand the principle of sampling of continuous time analog signal

BLOCK DIAGRAM:



VIRTUAL LAB LINK: https://dsp-iitkgp.vlabs.ac.in/exp/sampling-theorem/simulation.html





OBSERVATION:

Sl.No	V_{in}	V_{out}

CONCLUSION: The experiment verifies the Sampling Theorem, showing that accurate signal reconstruction is possible when the sampling frequency is at least twice the highest signal frequency.

Lab Manual 2

Control Systems Laboratory (BEC403)

CO's And PO's Mapping Chart

Sl. No.	Description	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
1	Deduce transfer function of a given physical system, from differential equation representation or Block Diagram representation and SFG Representation.	3	2			2								1	
2	Calculate time response specifications and analyse the stability of the System.	3	2			2								1	
3	Draw and analyse the effect of gain on system behavior using root loci.	3	2			2								1	
4	Perform frequency response Analysis and find the stability of the system.	3	2			2								1	
5	Represent State model of the system and find the time response of the system.	3	2			2								1	

Evaluation:

Integrated Lab(IPCC)		
CIE		
Particulars	**Marks**	**Total**
Performance	05	15(Reduced)
Journal	03	
Viva Voce	02	
Lab IA		
Particulars	**Marks**	**Total**
IA	50	10 (Reduced)
Total (CIE +Lab IA)		25

Mapping of Experiments with CO, PO and PSO

S. No.	Experiment Details	CO	PO	PSO
1	Implement Block diagram reduction technique to obtain transfer function a control system	1	1	1
2	Implement Signal Flow graph to obtain transfer function a control system	1	1	1
3	Simulation of poles and zeros of a transfer function	2	1	1
4	Implement time response specification of a second order Under damped System, for different damping factors.	2	2	1
5	Implement frequency response of a second order System.	2	2	1
6	Implement frequency response of a lead lag compensator.	2	2	1
7	Analyze the stability of the given system using Routh stability criterion.	3	2	1
8	Analyze the stability of the given system using Root locus.	3	2	1
9	Analyze the stability of the given system using Bode plots.	3	2	1
10	Analyze the stability of the given system using Nyquist plot.	4	2	1
11	Obtain the time response from state model of a system.	4	2	1
12	Implement PI and PD Controllers	5	5	1
13	Implement a PID Controller and hence realize an Error Detector.	5	5	1
14	Demonstrate the effect of PI, PD and PID controller on the system response.	5	5	1
15	**Virtual Lab Experiment-1** - To plot the response of a unity feedback system for different values of damping ratio (ζ) and calculate its rise time (tr), settling time (ts), & prcent; maximum overshoot. **http://vlabs.iitkgp.ac.in/gps/ctrl/Exp12/index.html**	4	1	1
16	**Virtual Lab Experiment-2** - To plot the root locus diagram of a unity feedback system and plot the output response of the system for different values of amplifier gain(k). **(http://vlabs.iitkgp.ac.in/gps/ctrl/Exp12/index.html)**	4	1	1

EXPERIMENT WISE LESSON PLAN

Experiment No.1	
Name	Implement Block diagram reduction technique to obtain transfer function a control system
Objectives	Students will able to understand how to reduce transfer function a control system
Experiment No.2	
Name	Implement Signal Flow graph to obtain transfer function a control system
Objectives	Students will able to understand how to solve given signal flow graph

Experiment No.3	
Name	Simulation of poles and zeros of a transfer function
Objectives	Students will able to understand how to locate poles and zeros of a transfer function
Experiment No. 4	
Name	Implement time response specification of a second order Under damped System, for different damping factors.
Objectives	Students will able to understand different damping factors.
Experiment No. 5	
Name	Implement frequency response of a second order System.
Objectives	Students will able to understand the implementation of frequency response of a second order System.
Experiment No. 6	
Name	Implement frequency response of a lead lag compensator.
Objectives	Students will able to understand the implementation of frequency response of a lead lag compensator.
Experiment No. 7	
Name	Analyze the stability of the given system using Routh stability criterion.
Objectives	Students will able to Analyze the stability of the given system using Routh stability criterion.
Experiment No.8	
Name	Analyze the stability of the given system using Root locus.
Objectives	Students will able to Analyze the stability of the given system using Root locus.
Experiment No.9	

Name	Analyze the stability of the given system using Bode plots.
Objectives	Students will able to Analyze the stability of the given system using Bode plots.

Experiment No.10	
Name	Analyze the stability of the given system using Nyquist plot.
Objectives	Students will able to Analyze the stability of the given system using Nyquist plot.
Experiment No. 11	
Name	Obtain the time response from state model of a system.
Objectives	Students will able to Obtain the time response from state model of a system.
Experiment No. 12	
Name	Implement PI and PD Controllers
Objectives	Students will able to Implement PI and PD Controllers
Experiment No. 13	
Name	Implement a PID Controller and hence realize an Error Detector.
Objectives	Students will able to Implement a PID Controller and Error Detector.
Experiment No. 14	
Name	Demonstrate the effect of PI, PD and PID controller on the system response.
Objectives	Students will able to understand the effect of PI, PD and PID controller on the system response
Experiment No.15	
Name	**Virtual Lab Experiment-1** - To plot the response of a unity feedback system for different values of damping ratio (ζ) and calculate its rise time (tr), settling time (ts), & prcent; maximum overshoot.
Objectives	Students will able to plot the response of a unity feedback system for different conditions
Experiment No.16	
Name	**Virtual Lab Experiment-2** - To plot the root locus diagram of a unity feedback system and plot the output response of the system for different values of amplifier gain (k).
Objectives	Students will able to plot the root locus diagram of a unity feedback system & output response of the system

LIST OF EXPERIMENTS

Experiment No. 1

Aim:

To write an octave program to implement Block diagram reduction technique to obtain transfer function a control system.

Theory:

A transfer function of a system, sub-system, or component is a mathematical function that models the system's output for each possible input.

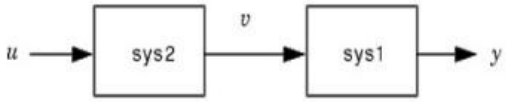
Connection of System Models:

In general, a system has different structures, such as series connection, parallel connection, and feedbacks. Octave also provides powerful tools to connect different parts of the system.

- The series connection of two sub-systems can be created by the following syntax:

  sys = series(sys2, sys1)

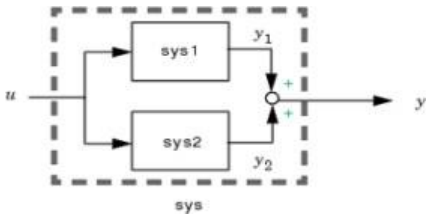
  The function "series" implements the following operation:



- The parallel connection of two sub-systems can be created by the following syntax:

  sys = parallel(sys1, sys2)

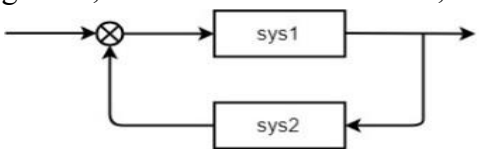
  The function "parallel" implements the following operation:



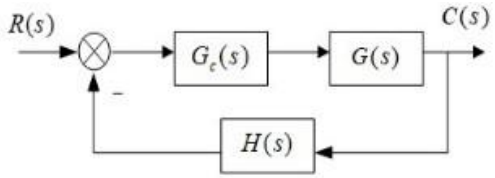
- The feedbacks can be created by the following syntax:

  sys = feedback(sys1, sys2, sign)

  The function "feedback" implements the following operation. If "sign=1", it gives a positive feedback. If "sign=-1", which is the default value, it gives a negative feedback.



Example: Compute the transfer function of the following system.



G(s) = $\frac{4}{s^3+2s^2+3s+4}$, $G_c(s) = \frac{s-3}{s+3}$, H(s) = $\frac{1}{0.01s+1}$

Octave Program:

```
clc; clear all; close all;
pkg load control;
s = tf('s');
G = 4/(s^3+2*s^2+3*s+4);
G_c = (s-3)/(s+3);
H = 1/(0.01*s+1);
sys1 = series (G, G_c);
sys = feedback(sys1, H)
```

Results:

Transfer function 'sys' from input 'u1' to output...

```
                0.04 s^2 + 3.88 s - 12
  y1:  --------------------------------
     0.01 s^5 + 1.05 s^4 + 5.09 s^3 + 9.13 s^2 + 17.12 s
```

Continuous-time model.

Experiment No. 2

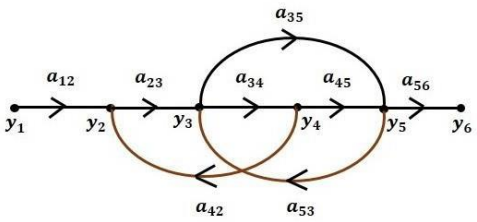
Implement Signal Flow graph to obtain transfer function a control system

Aim:

To write an octave program to implement Signal Flow graph to obtain transfer function a control system.

Theory:

Signal Flow Graphs are most commonly used to represent signal flow in a physical system and its controller(s), forming a cyber-physical system. Among their other uses are the representation of signal flow in various electronic networks and amplifiers, digital filters, state-variable filters and some other types of analog filters. In nearly all literature, a signal-flow graph is associated with a set of linear equations.



Octave Program:

```
clc; clear all; close all;
pkg load control;
s = tf('s');
G = 4/(s^3+2*s^2+3*s+4);
G_c = (s-3)/(s+3);
H = 1/(0.01*s+1);
sys1 = series(G, G_c);
sys = feedback(sys1, H)
```

Results:

Transfer function 'sys' from input 'u1' to output ...

```
                0.04 s^2 + 3.88 s - 12
  y1:  --------------------------------
     0.01 s^5 + 1.05 s^4 + 5.09 s^3 + 9.13 s^2 + 17.12 s
Continuous-time model.
```

Experiment No. 3

Simulation of poles and zeros of a transfer function.

Aim:

To write an octave program to simulate poles and zeros of a transfer function.

Theory:

Poles and Zeros of a transfer function are the frequencies for which the value of the denominator and numerator of a transfer function becomes infinite and zero respectively. The values of the poles and the zeros of a system determine whether the system is stable, and how well the system performs. Control systems, in the simplest sense, can be designed simply by assigning specific values to the poles and zeros of the system.

Let's assume that we have a transfer function in which the variable z appears in both the numerator and the denominator. In this situation, at least one value of z will cause the numerator to be zero, and at least one value of z will cause the denominator to be zero. A value that causes the numerator to be zero is a transfer-function zero, and a value that causes the denominator to be zero is a transfer-function pole.

Example:

$$T(z) = \frac{3z+2}{z^2+0.707z+0.25}$$

In this system, we have a zero at z = -0.6667, complex conjugate poles at z = –0.3536 + 0.3536i and z = -0.3536 - 0.3536i, and with gain value of k = 3.

Octave Program:

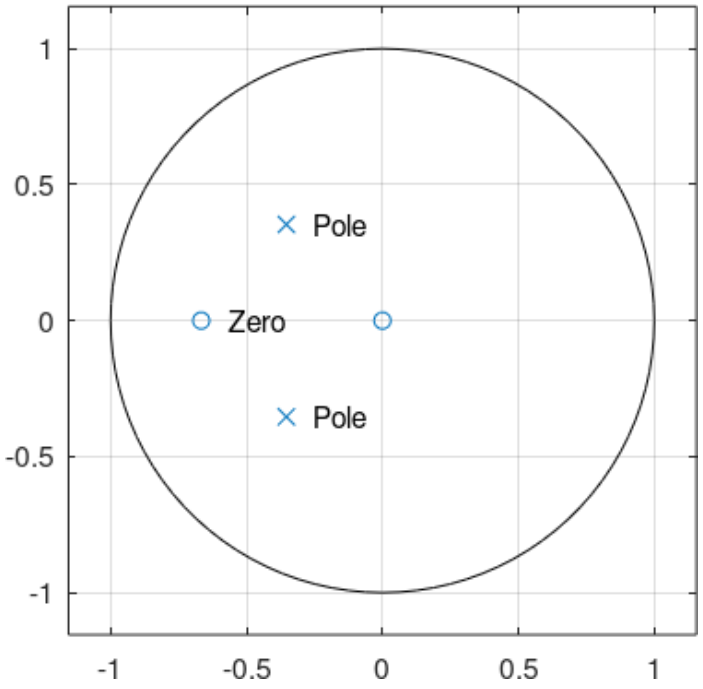
```
clc;  clear
all; close
all;
% Numerator and Denominator Coefficients b
= input('Enter Numerator Coefficients:');
a = input('Enter Numerator Coefficients:');
% Function to convert transfer function to poles and zeros
[z,p,k] = tf2zp(b,a)
% z-plane plot
zplane(b,a)
text(real(z)+0.1,imag(z),"Zero")
text(real(p)+0.1,imag(p),"Pole")
```

Results:

Enter Numerator Coefficients: [3 2]
Enter Numerator Coefficients: [1 0.707 0.25]
z = -0.6667
p =

-0.3535 + 0.3536i
-0.3535 - 0.3536i

k = 3



Experiment No. 4

Implement time response specification of a second order under damped System, for different damping factors

Aim:

To write an octave program to implement time response specification of a second order Under damped System, for different damping factors.

Theory:

An underdamped second-order system is a type of dynamic system that exhibits oscillatory behavior in its response. It is characterized by a second-order transfer function with complex conjugate poles.

In this system, the response initially overshoots the desired value and then settles down to the steady-state value. The amount of overshoot and the speed of settling are determined by the damping ratio (ζ) and the natural frequency (ωn) of the system. For an underdamped system, the damping ratio is less than 1 ($0 < \zeta < 1$), resulting in oscillatory behavior. To analyze and design underdamped second-order systems, techniques such as pole-zero analysis, time-domain analysis, frequency-domain analysis, and controller design methods like PID (Proportional-Integral-Derivative) control are often employed. By adjusting the damping ratio and natural frequency, the response of the system can be tailored to meet specific requirements, such as minimizing overshoot or achieving a faster settling time.

Octave Program:

```
clc; clear all; close all;
pkg load control;
zeta = [0.1 0.3 0.5 0.7 0.9];
wn = 10;
t = 0:0.01:10;
for i = 1:length(zeta)
    wd = wn*sqrt(1-zeta(i)^2);
    tr = pi/(wd*sqrt(1-zeta(i)^2));
    ts = 4/(zeta(i)*wd);
    Mp = exp(-zeta(i)*pi/sqrt(1-zeta(i)^2));

    fprintf('Damping Ratio = %.1f\n', zeta(i));
    fprintf('Damped Natural Frequency = %.2f rad/s\n', wd);
    fprintf('Rise Time = %.2f s\n', tr);
```

```
    fprintf('Settling Time = %.2f s\n', ts);
    fprintf('Percent Overshoot = %.2f %%\n\n', Mp*100);

    num = wn^2;
    den = [1 2*zeta(i)*wn wn^2];
    sys = tf(num, den);
    y = step(sys, t);
    plot(t, y)
  hold on
end

legend('zeta = 0.1', 'zeta = 0.3', 'zeta = 0.5.', 'zeta = 0.7', 'zeta = 0.9')
xlabel('Time (s)')
ylabel('Amplitude')
title('Step Response of Second-Order Underdamped System for Different Damping Factors')
```

Results:

Damping Ratio = 0.1
Damped Natural Frequency = 9.95 rad/s
Rise Time = 0.32 s
Settling Time = 4.02 s Percent
Overshoot = 72.92 %

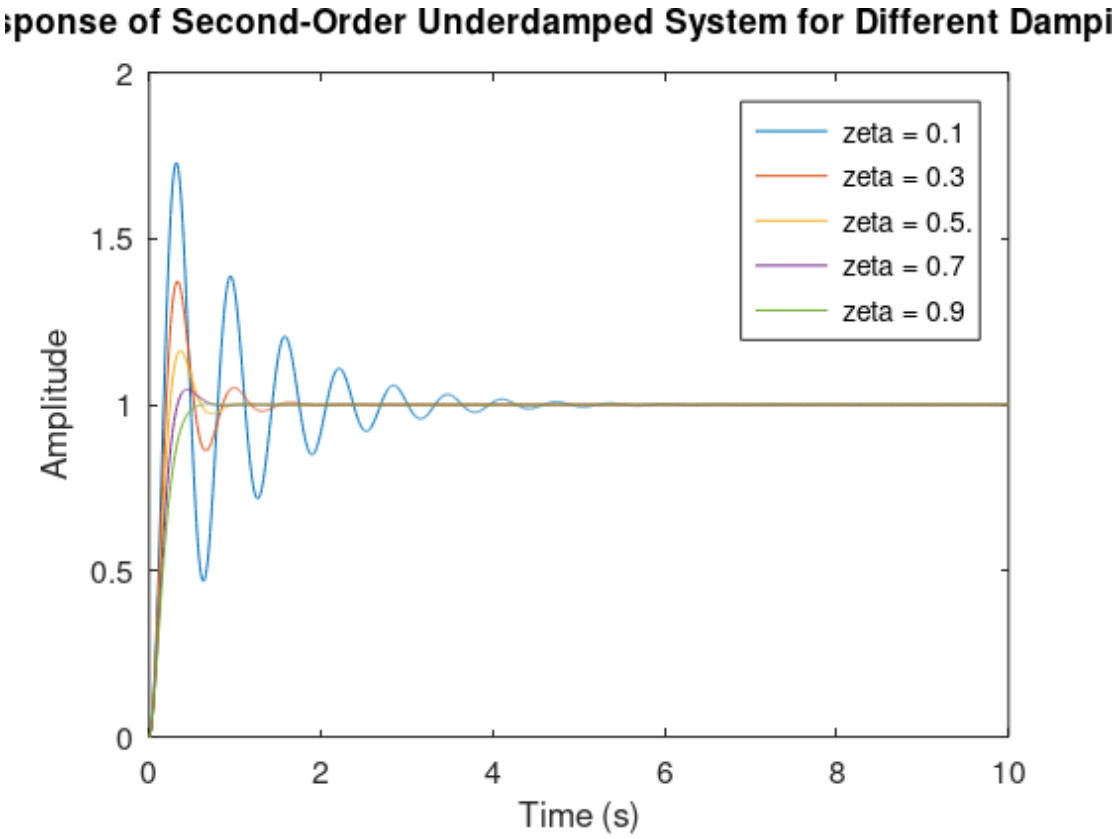
Damping Ratio = 0.3
Damped Natural Frequency = 9.54 rad/s
Rise Time = 0.35 s
Settling Time = 1.40 s Percent
Overshoot = 37.23 %

Damping Ratio = 0.5
Damped Natural Frequency = 8.66 rad/s
Rise Time = 0.42 s
Settling Time = 0.92 s Percent
Overshoot = 16.30 %

Damping Ratio = 0.7
Damped Natural Frequency = 7.14 rad/s
Rise Time = 0.62 s

Settling Time = 0.80 s Percent
Overshoot = 4.60 %

Damping Ratio = 0.9
Damped Natural Frequency = 4.36 rad/s
Rise Time = 1.65 s
Settling Time = 1.02 s Percent
Overshoot = 0.15 %



Experiment No. 5

Implement frequency response of a second order System

Aim:

To write an octave program to implement frequency response of a second order System.

Theory:

The frequency response of a second-order system refers to how the system responds to different frequencies of input signals. It describes how the output of the system changes in amplitude and phase in response to varying input frequencies.

A second-order system is characterized by a transfer function that can be represented in the form:

$$H(s) = \frac{K}{S^2 + 2\varsigma w_n S + w_n^2}$$

To analyze the frequency response of a second-order system, we typically consider the response to sinusoidal inputs of varying frequencies. By substituting s with jω (where j represents the imaginary unit and ω is the angular frequency), The frequency response is often expressed in terms of magnitude and phase. The magnitude response indicates how the system amplifies or attenuates the input signal at different frequencies, while the phase response describes the phase shift introduced by the system.

The magnitude response can be calculated as:|H(jω)| = |K / [(-ω^2) + (2ζω_nω) + (ω_n^2)]|

The phase response can be calculated as: ∠H(jω) = atan2(2ζω_nω, (ω_n^2) - ω^2)

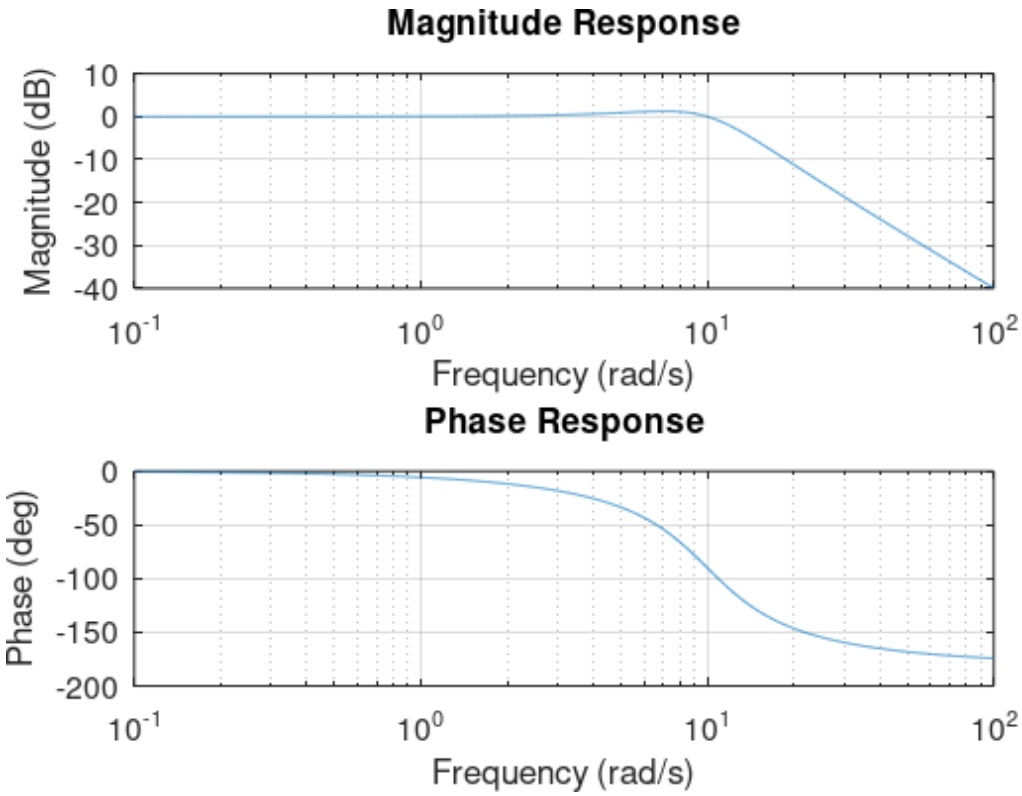
By evaluating these expressions for different values of ω, we can plot the frequency response of the second-order system. The plot typically shows the magnitude response on a logarithmic scale (in decibels) and the phase response in degrees. The frequency response of a second-order system can reveal important characteristics such as resonant frequencies, bandwidth, and the system's ability to amplify or attenuate specific frequencies. It is widely used in fields such as control systems, signal processing, and audio engineering to analyze and design systems that respond appropriately to different frequencies

Octave Program:

```
clc; clear all; close all;
pkg load control;
% Second-Order System Frequency Response Example

% Given System
wn = 10; % Natural Frequency
zeta = 0.5; % Damping Ratio
```

```
% Define Transfer Function
num = wn^2;
den = [1 2*zeta*wn wn^2]; sys
= tf(num, den);

% Define Frequency Range
w = logspace(-1, 2, 1000);

% Calculate Frequency Response
[mag, phase] = bode(sys, w);

% Reshape data to 2D matrix
mag2d = reshape(mag, [], size(mag, 3));
phase2d = reshape(phase, [], size(phase, 3));

% Plot Frequency Response
subplot(211);
semilogx(w, 20*log10(mag2d));
grid on;
title('Magnitude Response');
xlabel('Frequency (rad/s)');
ylabel('Magnitude (dB)');

subplot(212);
semilogx(w, phase2d);
grid on;
title('Phase Response');
xlabel('Frequency (rad/s)');
ylabel('Phase (deg)');
```

Results:


Magnitude Response
Magnitude (dB)
10
0
-10
-20
-30
-40
10^{-1}
10^{0}
10^{1}
10^{2}
Frequency (rad/s)
Phase Response
Phase (deg)
0
-50
-100
-150
-200
10^{-1}
10^{0}
10^{1}
10^{2}
Frequency (rad/s)

Experiment No. 6

Implement frequency response of a lead lag compensator

Aim:

To write an octave program to implement frequency response of a lead lag compensator.

Theory:

A lead compensator is a type of controller used in control systems engineering to improve system stability and transient response. It introduces a phase lead in the frequency response of the system, which increases the system's phase margin and reduces settling time and overshoot. The lead compensator achieves the phase lead by introducing a zero in the transfer function of the system. This zero causes a positive phase shift as the frequency increases, helping the system respond faster to disturbances. The lead compensator is commonly used when fast response and stability are required, such as in motion control applications or systems with time delays.

The transfer function of lead Compensator is

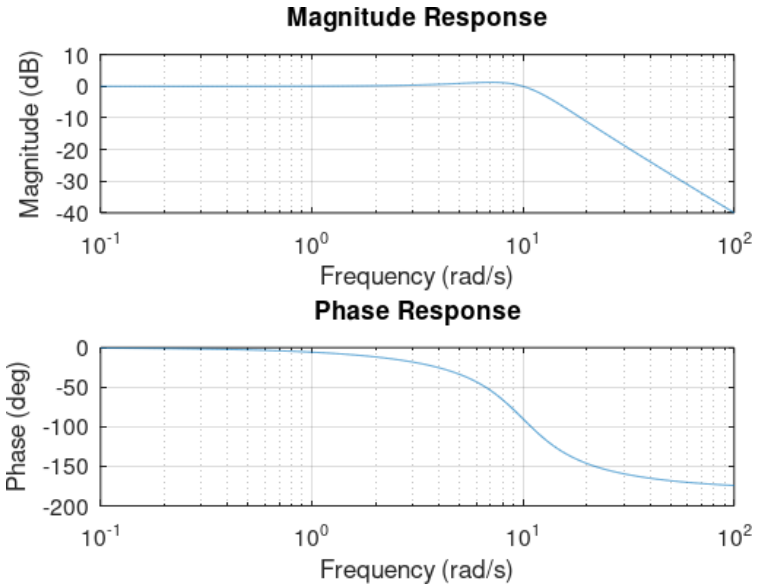
$$\frac{V_o(s)}{V_i(s)} = \frac{(s+\frac{1}{T})}{[s+\frac{1}{\alpha T}]}$$ Where, $T = R_1C$ $\alpha = \frac{R_2}{R_1+R_2}$

Octave Program:

```
clc; clear all; close all;
pkg load control;
% Second-Order System Frequency Response Example

% Given System
wn = 10; % Natural Frequency
zeta = 0.5; % Damping Ratio

% Define Transfer Function
num = wn^2;
den = [1 2*zeta*wn wn^2]; sys
= tf(num, den);

% Define Frequency Range
w = logspace(-1, 2, 1000);

% Calculate Frequency Response
```

```
[mag, phase] = bode(sys, w);

% Reshape data to 2D matrix
mag2d = reshape(mag, [], size(mag, 3));
phase2d = reshape(phase, [], size(phase, 3));

% Plot Frequency Response
subplot(2,1,1);
semilogx(w, 20*log10(mag2d));
grid on;
title('Magnitude Response');
xlabel('Frequency (rad/s)');
ylabel('Magnitude (dB)');

subplot(2,1,2);
semilogx(w, phase2d);
grid on;
title('Phase Response');
xlabel('Frequency (rad/s)');
ylabel('Phase (deg)');
```

Results:



Experiment No. 7

Analyze the stability of the given system using Routh stability criterion

Aim:

To write an octave program to analyze the stability of the given system using Routh stability criterion.

Theory:

In the control system theory, the Routh–Hurwitz stability criterion is a mathematical test that is a necessary and sufficient condition for the stability of a linear time-invariant (LTI) dynamical system or control system. A stable system is one whose output signal is bounded; the position, velocity or energy do not increase to infinity as time goes on. The importance of the criterion is that the roots p of the characteristic equation of a linear system with negative real parts represent solutions e^{pt} of the system that are stable (bounded). Thus the criterion provides a way to determine if the equations of motion of a linear system have only stable solutions, without solving the system directly.

$s^4+19s^3+111s^2+189s+ s+5=0$

Octave Program:

```
%%routh hurwitz criteria
clc; clear all; close all; pkg
load control;
%%firstly it is required to get first two row of routh matrix
e=input('enter the coefficients of characteristic equation: ');
disp('  ------------------------------------------------')
l=length(e);
m=mod(l,2); if
m==0
      a=rand(1,(l/2));
      b=rand(1,(l/2));
      for i=1:(l/2)
         a(i)=e((2*i)-1);
         b(i)=e(2*i);
    end
else
      e1=[e 0];
      a=rand(1,((l+1)/2));
      b=[rand(1,((l-1)/2)),0];
      for i=1:((l+1)/2)
```

```
        a(i)=e1((2*i)-1);
        b(i)=e1(2*i);
    end
end
%%now we genrate the remaining rows of routh matrix
l1=length(a);
c=zeros(l,l1);
c(1,:)=a;
c(2,:)=b;
for m=3:l
    for n=1:l1-1
        c(m,n)=-(1/c(m-1,1))*det([c((m-2),1) c((m-2),(n+1));c((m-1),1) c((m-1),(n+1))]);
  end
end
disp('the routh matrix:')
disp(c)
%%now we check the stablity of system if
c(:,1)>0
  disp('System is Stable')
else
  disp('System is Unstable');
end
```

Results:

```
enter the coefficients of characteristic equation: [1 19 111 189 1 5]
--------------------------------------------------------------------
the routh matrix:
    1.0000  111.0000    1.0000
   19.0000  189.0000    5.0000
  101.0526    0.7368         0
  188.8615    5.0000         0
   -1.9385         0         0
    5.0000         0         0
System is Unstable
```

Experiment No. 8

Analyze the stability of the given system using Root Locus

Aim:

To write an octave program to analyze the stability of the given system using Root Locus.

Theory:

The stability of a closed loop system is determined from the location of roots of the closed loop characteristic equation 1+G(s)H(s) = 0. For system to be stable the roots of the characteristic equation should be located in the L.H.S of the s plane. Any roots of the characteristic equation satisfy the following two condition:

1) |G(s)H(S)| = 1
2) /_G(S)H(s) = (2k +1)1800 where k = 0,1,2….

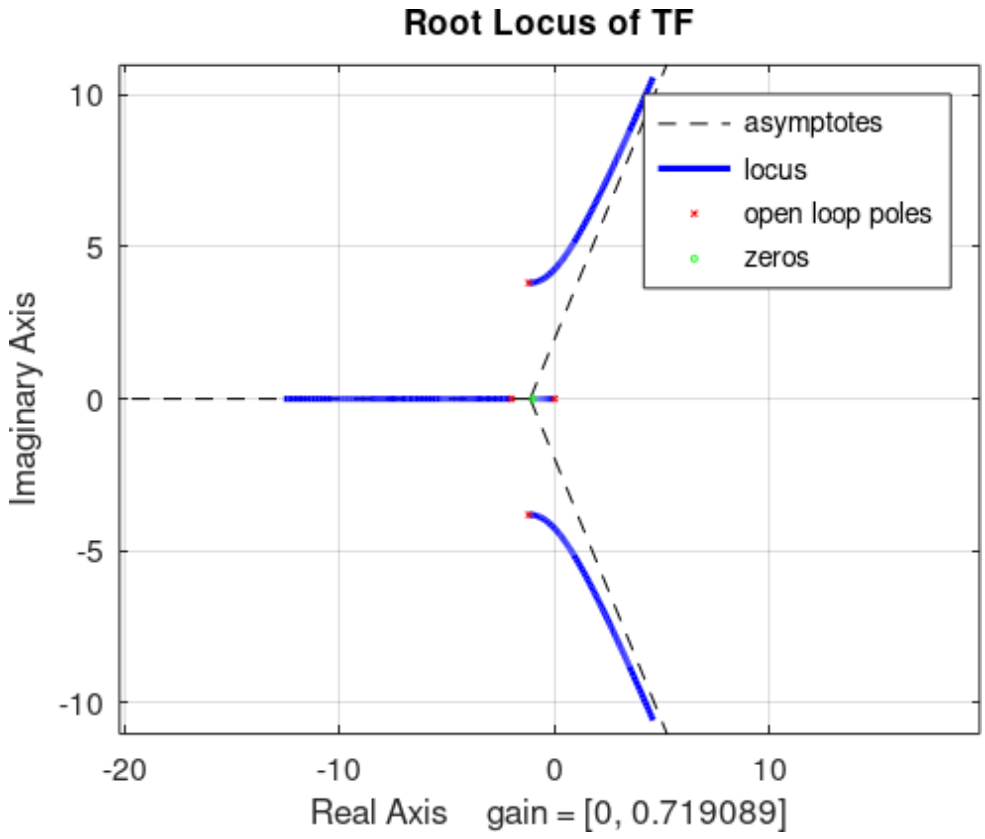
The root locus method of analysis is a process of of determining the points in the s plane satisfying the above two condition. Usually, the forward path gain factor K is considered as an independent variable and the roots of 1+G(s)H(S) = 0 as dependent variables, the roots are plotted in s plane with K as a variable parameter.

$$G(s)H(s) = \frac{48(s+1)}{(s+20)(s^2+2.4s+16)}$$

Octave Program:

```
clc; clear all; close all; pkg
load control;
s = tf('s'); G1
= 48/s; G2 =
s+1;
G3 = 1/(s + 2);
G4 = 48/((s^2) + (2.4*s) + 16);

% Given Transfer Function TF
= G1*G2*G3*G4;

% Root locus of the Transfer Function
rlocus(TF);
axis([-30 10 -20 2])
```

Results:


Root Locus of TF
asymptotes
locus
open loop poles
zeros
Imaginary Axis
Real Axis
gain = [0, 0.719089]
10
5
0
-5
-10
-20
-10
0
10

Experiment No. 9

Analyze the stability of the given system using Bode Plots

Aim:

To write an octave program to analyze the stability of the given system using Bode Plots.

Theory:

The Bode plot method gives a graphical procedure for determining the stability of a control system based on sinusoidal frequency response. The transfer function for sinusoidal input response can be obtained by substituting jw in place of Laplace operator s.

Therefore the open loop transfer function G(s)H(s) becomes G(jw)H(jw) which can be expressed in the form of magnitude and phase angle.

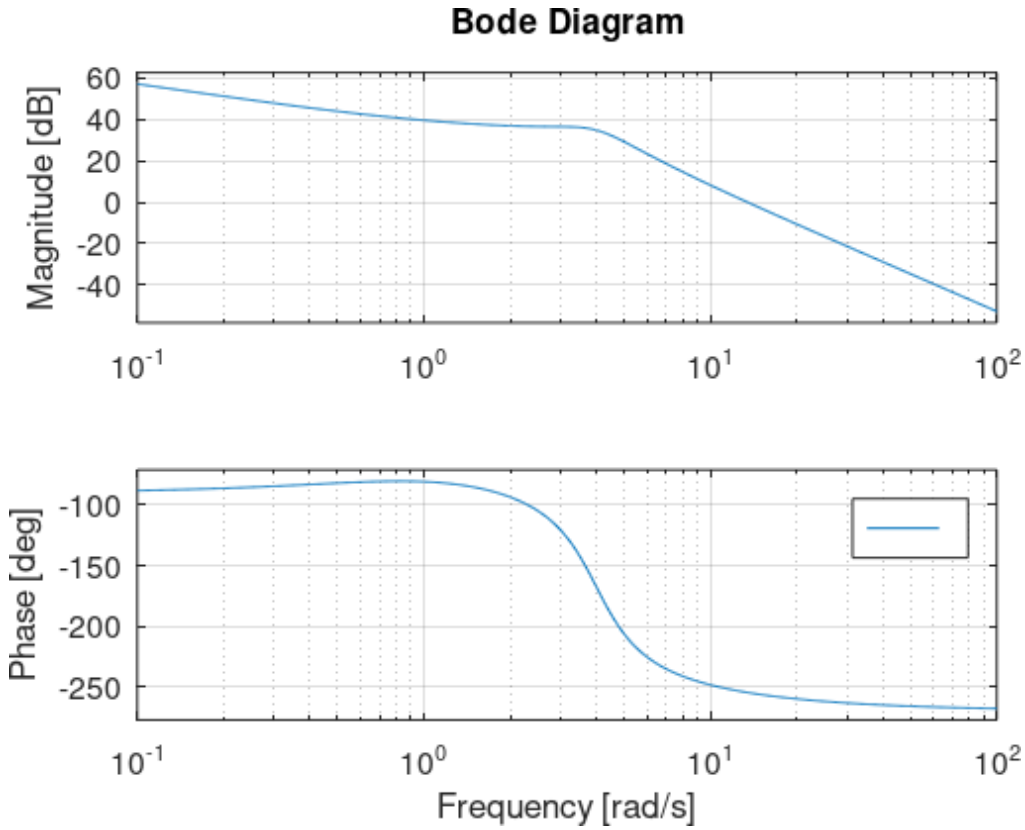
$$G(s)H(s) = \frac{48(s+1)}{(s+20)(s^2+2.4s+16)}$$

Octave Program:

```
clc; clear all; close all; pkg
load control;
s = tf('s'); G1
= 48/s; G2 =
s+1;
G3 = 1/(s + 2);
G4 = 48/((s^2) + (2.4*s) + 16);
% Given Transfer Function TF
= G1*G2*G3*G4;
% Root locus of the Transfer Function
bode(TF);
grid on;

% find Gain and Phase Margin [GM,
PM, Wcp, Wgc]=margin(TF);
GMdB=20*log10(GM); % Gain margin in dB
fprintf('GM=%f, PM=%f, Wgc = %f, Wpc = %f', GMdB,PM,Wcp,Wgc)
```

Results:

GM=-33.640676, PM=285.221465, Wgc = 4.261681, Wpc = 13.497680



Experiment No. 10

Analyze the stability of the given system using Nyquist Plot

Aim:

To write an octave program to analyze the stability of the given system using Nyquist Plot.

Theory:

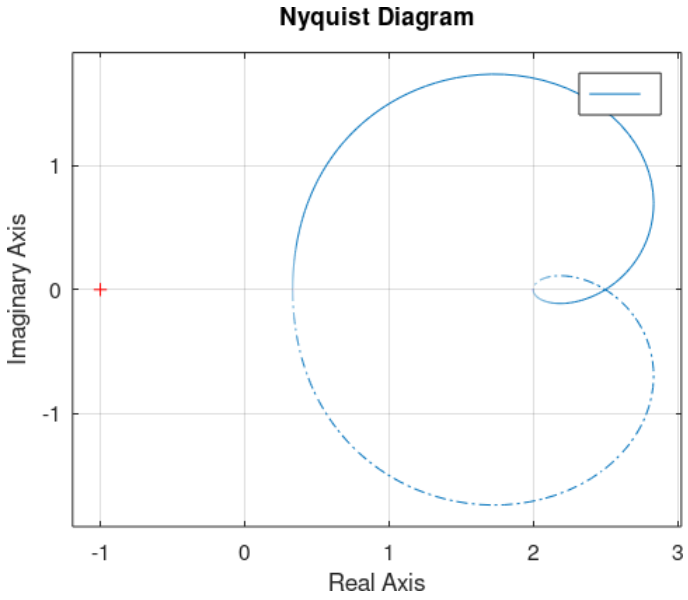
A Nyquist plot is a parametric plot of a frequency response used in automatic control and signal processing. The most common use of Nyquist plots is for assessing the stability of a system with feedback. In Cartesian coordinates, the real part of the transfer function is plotted on the X-axis while the imaginary part is plotted on the Y-axis. The frequency is swept as a parameter, resulting in one point per frequency. The same plot can be described using polar coordinates, where gain of the transfer function is the radial coordinate, and the phase of the transfer function is the corresponding angular coordinate.

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

Octave Program:

```
clc; clear all; close all;
pkg load control;
H = tf([2 5 1],[1 2 3]);
nyquist(H) grid
on
```

Results:



Experiment No. 11

Obtain the time response from state model of a system

Aim:

To write an octave program to obtain the time response from state model of a system.

Theory:

A state-space model is a mathematical representation of a physical system as a set of input, output, and state variables related by first-order differential equations. The state variables define the values of the output variables. The ss model object can represent SISO or MIMO state-space models in continuous time or discrete time.

In continuous-time, a state-space model is of the following form:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Here, x, u and y represent the states, inputs and outputs respectively, while A, B, C and D are the state-space matrices. The ss object represents a state-space model in Octave storing A, B, C and D along with other information such as sample time, I/O names, delays, and offsets.

Example: Create the SISO state-space model defined by the following state-space matrices & obtain the time response.

$$A = \begin{bmatrix} -1.5 & -2 \\ 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \quad C = [0 \quad 1] \quad D = 0$$

Octave Program:

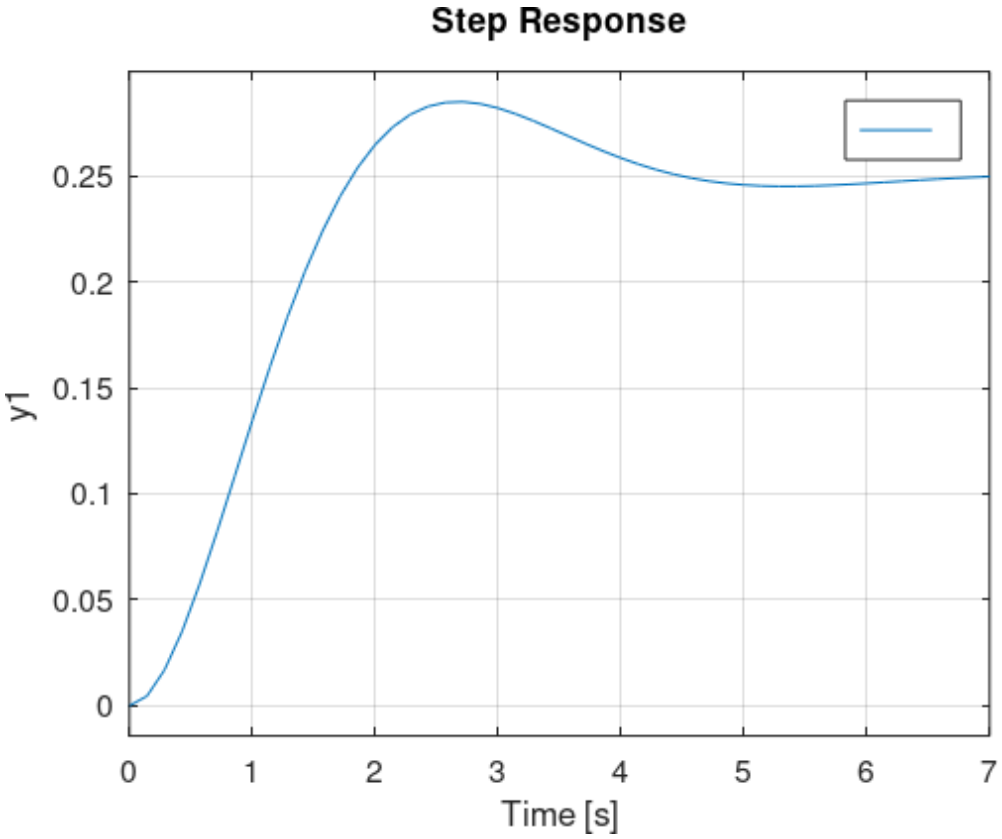
```
clc; clear all; close all;
pkg load control;
A = [-1.5, -2; 1, 0];
B = [0.5; 0];
C = [0, 1];
D = 0;
sys = ss(A,B,C,D)
step(sys)
```

Results:

```
sys.a =
        x1    x2
   x1  -1.5    -2
   x2    1     0
```

```
sys.b =
        u1
   x1  0.5
   x2    0

sys.c =
        x1  x2
   y1    0   1

  sys.d =
        u1
   y1    0
```

Continuous-time model.



Experiment No. 12
Implement PI and PD Controllers

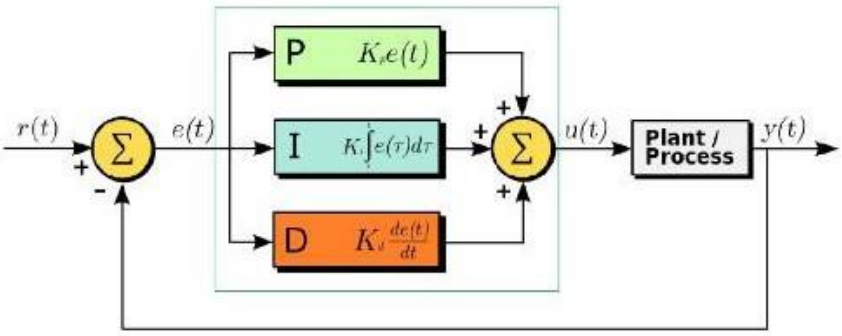
Aim:
To write an octave program to implement PI and PD Controllers.

Theory:
A variation of Proportional Integral Derivative (PID) control is to use only the proportional and integral terms as PI control. The PI controller is the most popular variation, even more than full PID controllers. The value of the controller output u(t) is fed into the system as the manipulated variable input.

$$e(t) = SP - PV$$

$$u(t) = u_{bias} + K_c\, e(t) + \frac{K_c}{\tau_I} \int_0^t e(t)dt$$

The u_{bias} term is a constant that is typically set to the value of u(t) when the controller is first switched from manual to automatic mode. This gives "bumpless" transfer if the error is zero when the controller is turned on. The two tuning values for a PI controller are the controller gain, K_c and the integral time constant τ_I. The value of K_c is a multiplier on the proportional error and integral term and a higher value makes the controller more aggressive at responding to errors away from the set point. The set point (SP) is the target value and process variable (PV) is the measured value that may deviate from the desired value. The error from the set point is the difference between the SP and PV and is defined as e(t) = SP – PV.



The dynamic equations in the Laplace domain and the open-loop transfer function of the DC Motor are the following:

$$s(Js + b)\Theta(s) = KI(s)$$

$$(Ls + R)I(s) = V(s) - Ks\Theta(s)$$

$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2} \qquad [\frac{rad/sec}{V}]$$

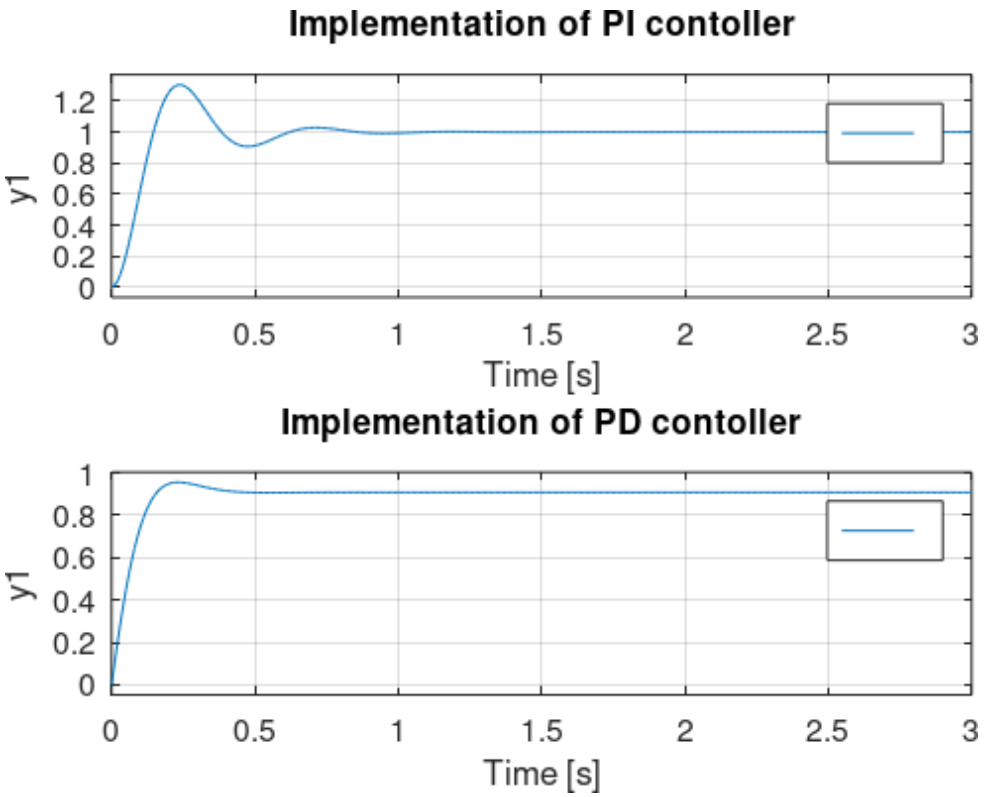
Octave Program:

```
clc; clear all; close all;
pkg load control; J=0.01;
B=0.1;
K=0.01;
R=1;
L=0.5;
s=tf('s');
p_motor = K/((J*s+B)*(L*s+R)+K^2)
%sys=feedback(p_motor, 1);

% Implement PI controller
Kp=100;
Ki=200;
Kd=0;
C=pid(Kp,Ki,Kd);
sys_PI = feedback(C*p_motor,1);
subplot(211);
step(sys_PI, 0:0.01:3)
title('Implementation of PI controller');

% Implement PD controller
Kp=100;
Ki=0;
Kd=5;
C=pid(Kp,Ki,Kd);
sys_PI = feedback(C*p_motor,1);
subplot(212);
step(sys_PI, 0:0.01:3)
title('Implementation of PD controller');
```

Results:

```
Transfer function 'p_motor' from input 'u1' to output ...
                0.01
  y1:  ------------------
     0.005 s^2 + 0.06 s + 0.1001
```

Continuous-time model.



Experiment No. 13

Implement a PID Controller and hence realize an Error Detector

Aim:

To write an octave program to implement a PID Controller and hence realize an Error Detector.

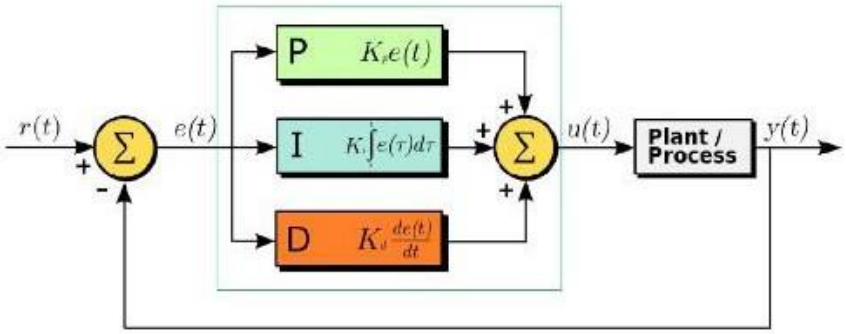
Theory:

A variation of Proportional Integral Derivative (PID) control is to use only the proportional and integral terms as PI control. The PI controller is the most popular variation, even more than full PID controllers. The value of the controller output u(t) is fed into the system as the manipulated variable input.

$$e(t) = SP - PV$$

$$u(t) = u_{bias} + K_c\, e(t) + \frac{K_c}{\tau_I} \int_0^t e(t)dt$$

The u_{bias} term is a constant that is typically set to the value of u(t) when the controller is first switched from manual to automatic mode. This gives "bumpless" transfer if the error is zero when the controller is turned on. The two tuning values for a PI controller are the controller gain, K_c and the integral time constant τ_I. The value of K_c is a multiplier on the proportional error and integral term and a higher value makes the controller more aggressive at responding to errors away from the set point. The set point (SP) is the target value and process variable (PV) is the measured value that may deviate from the desired value. The error from the set point is the difference between the SP and PV and is defined as e(t) = SP – PV.



The dynamic equations in the Laplace domain and the open-loop transfer function of the DC Motor are the following:

$$s(Js + b)\Theta(s) = KI(s)$$

$$(Ls + R)I(s) = V(s) - Ks\Theta(s)$$

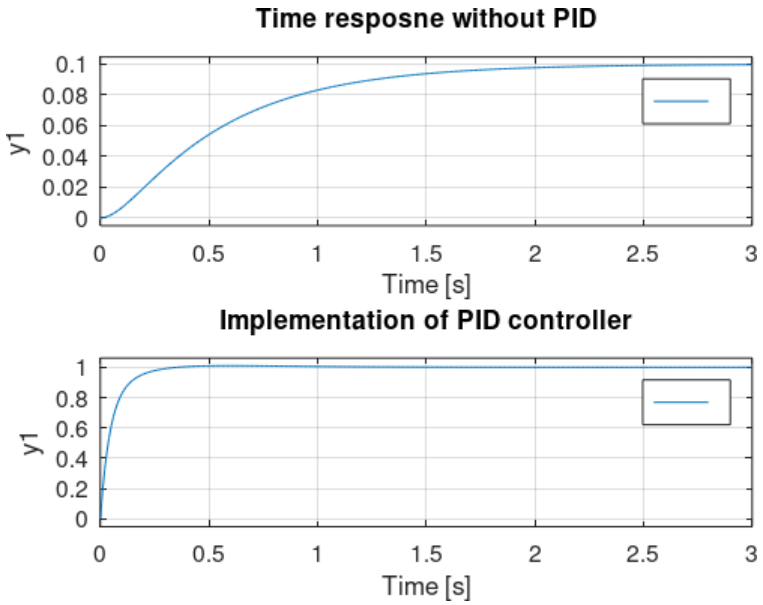
$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2} \qquad [\frac{rad/sec}{V}]$$

Octave Program:

```
clc; clear all; close all;
pkg load control; J=0.01;
B=0.1;
K=0.01;
R=1;
L=0.5;
s=tf('s');
p_motor = K/((J*s+B)*(L*s+R)+K^2)
%sys=feedback(p_motor, 1);

%plot time response without PID Controller
t=0:0.01:3;
subplot(211)
step(p_motor,t)
title('Time resposne without PID');

% Implement PID controller
Kp=100;
Ki=200;
Kd=10;
C=pid(Kp,Ki,Kd);
sys_PI = feedback(C*p_motor,1);
subplot(212);
step(sys_PI, 0:0.01:3)
title('Implementation of PID controller');
```

Results:

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

               0.01
  y1:  ------------------
       0.005 s^2 + 0.06 s + 0.1001

Continuous-time model.
```


Time resposne without PID
y1
Time [s]
Implementation of PID controller
y1
Time [s]

Experiment No. 14

Demonstrate the effect of PI, PD and PID controller on the system response.

Aim:

To write an octave program to demonstrate the effect of PI, PD and PID controller on the system response.

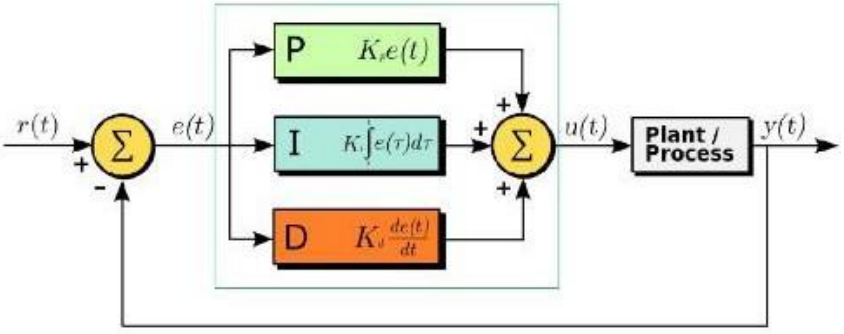
Theory:

A variation of Proportional Integral Derivative (PID) control is to use only the proportional and integral terms as PI control. The PI controller is the most popular variation, even more than full PID controllers. The value of the controller output u(t) is fed into the system as the manipulated variable input.

$$e(t) = SP - PV$$

$$u(t) = u_{bias} + K_c\, e(t) + \frac{K_c}{\tau_I} \int_0^t e(t)dt$$

The u_{bias} term is a constant that is typically set to the value of u(t) when the controller is first switched from manual to automatic mode. This gives "bumpless" transfer if the error is zero when the controller is turned on. The two tuning values for a PI controller are the controller gain, K_c and the integral time constant τ_I. The value of K_c is a multiplier on the proportional error and integral term and a higher value makes the controller more aggressive at responding to errors away from the set point. The set point (SP) is the target value and process variable (PV) is the measured value that may deviate from the desired value. The error from the set point is the difference between the SP and PV and is defined as e(t) = SP – PV.



The dynamic equations in the Laplace domain and the open-loop transfer function of the DC Motor are the following:

$$s(Js+b)\Theta(s) = KI(s)$$

$$(Ls+R)I(s) = V(s) - Ks\Theta(s)$$

$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js+b)(Ls+R)+K^2} \qquad [\frac{rad/sec}{V}]$$

Octave Program:

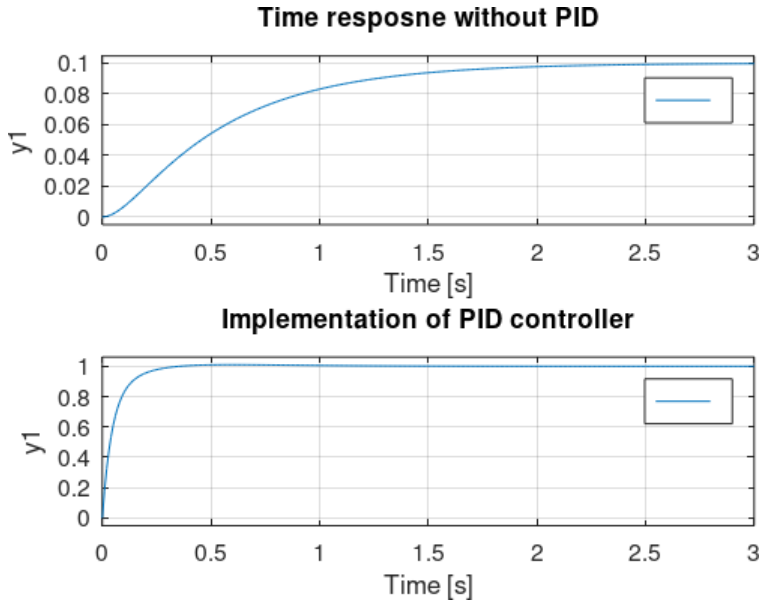
```
clc; clear all; close all;
pkg load control; J=0.01;
B=0.1;
K=0.01;
R=1;
L=0.5;
s=tf('s');
p_motor = K/((J*s+B)*(L*s+R)+K^2)
%sys=feedback(p_motor, 1);

%plot time response without PID Controller
t=0:0.01:3;
subplot(211)
step(p_motor,t)
title('Time resposne without PID');

% Implement PID controller
Kp=100;
Ki=200;
Kd=10;
C=pid(Kp,Ki,Kd);
sys_PI = feedback(C*p_motor,1);
subplot(212);
step(sys_PI, 0:0.01:3)
title('Implementation of PID controller');
```

Results:

Transfer function 'p_motor' from input 'u1' to output ...

$$y1: \ \frac{0.01}{0.005\ s^2 + 0.06\ s + 0.1001}$$

Continuous-time model.



Experiment No. 15

Virtual Lab Experiment – 1

Website: http://vlabs.iitkgp.ac.in/gps/ctrl/Exp12/index.html

Aim:

To plot the response of a unity feedback system for different values of damping ratio (ζ) and calculate its rise time (tr), settling time (ts), & prcent; maximum overshoot.

Theory:

If the output or some part of the output is returned to the input side and utilized as part of the system input, then it is known as feedback. Feedback plays an important role in order to improve the performance of the control systems. Negative feedback reduces the error between the reference input R(s) and system output. The figure shows the block diagram of the negative feedback control system. Since H(s) = 1, hence it is a unity feedback system.

Second order system:

The closed loop transfer function of the second order unity feedback system is given by,

$$\frac{Y(s)}{R(s)} = \frac{(s)}{1+(s)} = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$$

The expression for the response of the second order system can be written as,

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

When ζ = 0, the system is undamped.
When ζ = 1, the system is critically damped. When
0 < ζ < 1, the system is under damped.

Unit step response of second order system:

Apply unit step signal at the input of the second order system,

$$r(t) = u(t)$$

Taking laplace transform on the both sides,

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

Case 1 : when ζ = 0 i.e. system is undamped
The equation 1 becomes,

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

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

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

Taking the inverse laplace transform on both sides, we have,

$$Y(t) = (1 - \cos(\omega_n t))\, u(t)$$

This equation shows that the unit step response of an undamped system is a continuous time signal of constant amplitude and frequency.

Case 2 : when $\zeta = 1$ i.e. system is critically damped The equation 1 becomes,

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

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

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

After the partial fraction, taking the inverse laplace transform on both the sides, we have,

$$Y(t) = (1 - e^{-\omega_n t} - \omega_n t e^{-\omega_n t})\, u(t)$$

When the system is critically damped then, the unit step response of the second order system would try to reach the steady state step input.

Case 3 : when $0 < \zeta < 1$ i.e. system is under damped

The equation 1 can be written as,

$$Y(s) = \frac{\omega_n^2}{((s+\zeta\omega_n)^2+ \omega_n^2(1-\zeta^2))} R(s)$$

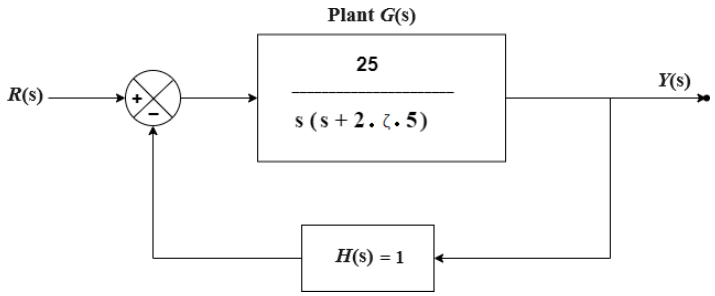
Substituting $R(s) = \frac{1}{s}$

$$Y(s) = \frac{\omega_n^2}{((s+\zeta\omega_n)^2+ \omega_n^2(1-\zeta^2))}$$

The shows, when the system is under damped then, the unit step response of the system is having damped oscillations i.e. response of decreasing amplitude.

Procedure:

Unity feedback control system:

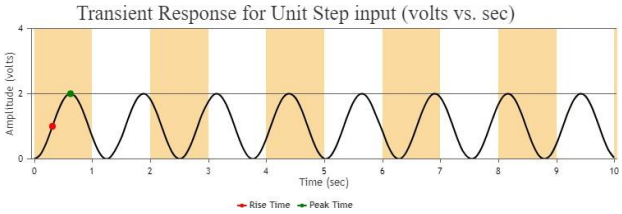


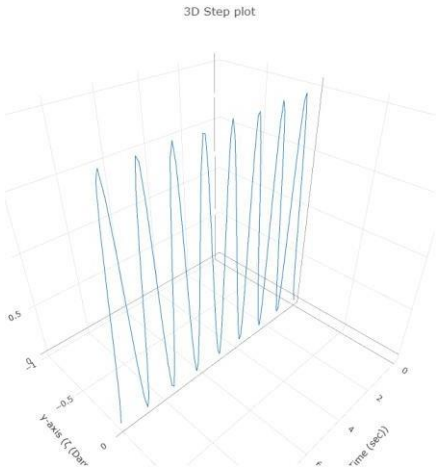
Steps to perform the simulation:

1. First click on "Problem 1" button and select the input signal as Unit Step. In the given unity feedback system, set the value of ζ (damping ratio) through the up-down arrows in the input box in denominator of plant transfer function G (s).
2. ζ can be varied from 0 to 1. After the value of ζ is set click on the 'RUN' button to observe the system response for the particular input signal and & zeta value.
3. Click on 'Download Plot' button to download the response plot. Click on 'Ok' button.
4. Click on '3D-Plot' button then on '3D Step' to observe the 3D response plot for that particular input signal, where x-axis represents time (sec), y-axis represents ζ (damping ratio) and z-axis represents the system response. Hover on the plot area and click on the camera icon in the top right corner to download this 3D plot. Click on 'Ok' button.
5. Change the value of ζ and repeat step 2-4.

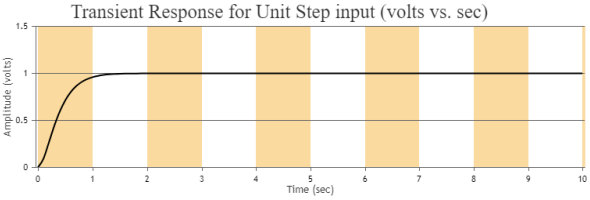
Results:

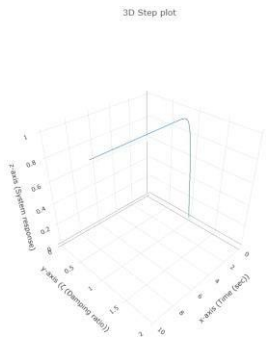
Case 1 : when ζ = 0 i.e. system is undamped



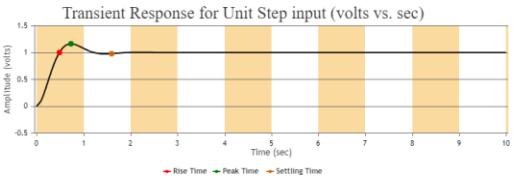


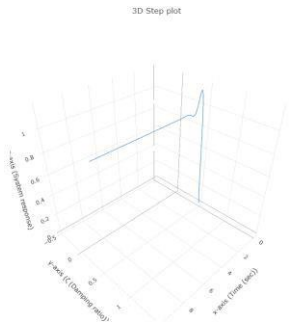
Case 2 : when ζ = 1 i.e. system is critically damped





Case 3 : when $0 < \zeta < 1$ i.e. system is under damped





Experiment No. 16

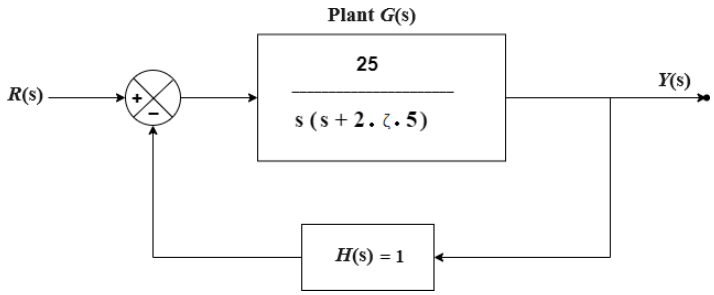
Virtual Lab Experiment – 2

Website: http://vlabs.iitkgp.ac.in/gps/ctrl/Exp12/index.html

Aim:

To plot the root locus diagram of a unity feedback system and plot the output response of the system for different values of amplifier gain(k).

Theory:



Plot the root loci for the unity feedback system shown. Where,

$$G(s) = KG_1(s) = \frac{K}{(s+1)(s+2)}$$

Assume that the amplifier gain (K) is varied from 0 to 50. Indicate the value of the gain K for which the root locus crosses the imaginary axis. Plot the output response for K = 0.4, 2, 6 and 12. The characteristic equation of the system (considered in problem-2) is

$$1 + KG_1(s)\,H(s) = 0$$

The root locus is the path of the roots of characteristic equation traced out in s-plane as gain K is changed from 0 to ∞ and it is symmetrical about the real axis.

Construction of Root Locus:

Branch

The root locus branches start at the open loop poles and end at open loop zeros. So, the number of root locus branches N is equal to the number of finite open loop poles P or the number of finite open loop zeros Z, whichever is greater. Mathematically, the number of root locus branches N can be written as

$$N = P, \text{ if } P \geq Z$$

$$N = Z, \text{ if } Z > P$$

If odd number of open loop poles and zeros exist to the left side of a point on the real axis, then that point is on the root locus branch.

Centroid (α) :

$$\alpha = \frac{\Sigma \text{ Real part of finite open loop poles} - \Sigma \text{ Real part of finite open loop zeros}}{P-Z}$$

Angle of asymptotes (θ) :

$$\theta = \frac{(2q+1)180^0}{P-Z}$$

where q = 0, 1, 2,...(P-Z)-1

Break-away and Break-in points :

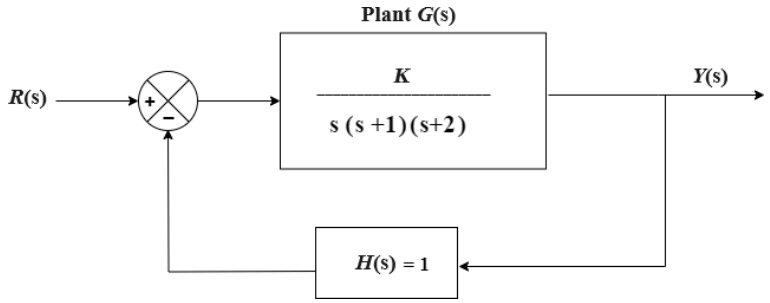
If there exists a real axis root locus branch between two open loop poles, then there will be a break-away point in between these two open loop poles. If there exists a real axis root locus branch between two open loop zeros, then there will be a break-in point in between these two open loop zeros.

Note : Break-away and break-in points exist only on the real axis root locus branches.

Steps to find break-away and break-in points :

1) Write K in terms of s from the characteristic equation.
2) Differentiate K with respect to s and make it equal to zero. Substitute these values of s in the equation (found from step 1).
3) The values of s for which the K value is positive are the break points.

Procedure:



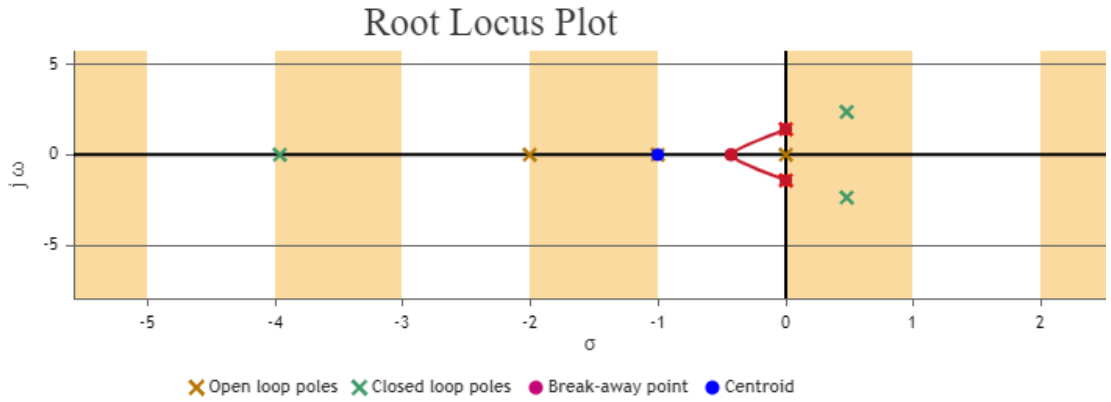
Plot the root loci for the unity feedback system:

1. First click on "Problem 2" button and select the input signal as Unit Step. In the given unity feedback system, set the value of K (amplifier gain) through the up-down arrows in the input box in numerator of plant transfer function G (s).
2. K can be varied from 1 to 50. After the value of K is set click on the 'RUN' button, then on 'Root Locus' to observe the root locus plot of the system for that particular amplifier gain. The plot can be zoomed by selecting an area under the plot. Click on 'Ok' button to clear it. Similarly, click on the 'RUN' button, then on 'Output Response' to observe the system response for the particular input signal and K value.

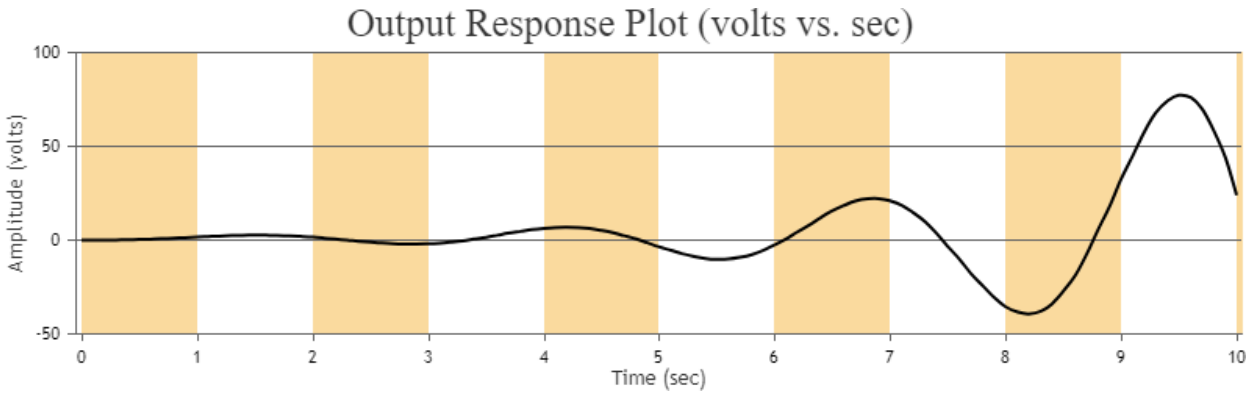
3. The plots can be downloaded by clicking on 'Download Plot' button. Click on 'Ok' button.
4. Change the value of gain K and repeat step 2-3.

Results:

Root Locus Plot:



Output Response:



Lab Manual 3

Communication Laboratory (BECL404)

CO's And PO's Mapping Chart

Sl. No.	Description	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
1	Understand the basic concepts of RF transmitters and Receivers.	2	2												2
2	Illustrate the AM and FM modulation generation and detection using suitable electronic circuits.	2	2	1											2
3	Design and test the sampling, Multiplexing and pulse modulation techniques using electronic hardware.	2	2	1											2
4	Design and demonstrate the electronic circuits used for RF transmitters and receivers.	2	2	1											2

Evaluation:

Non Integrated Lab		
CIE		
Particulars	**Marks**	**Total**
Performance	5	10 * 14 = 140 30 (Reduced)
Journal	2	
Viva Voce	3	
Lab IA		
Particulars	**Marks**	**Total**
IA	100	20 (Reduced)
Total (CIE +Lab IA)		50

Mapping of Experiments with CO, PO and PSO

S.No.	Experiment Details	CO	PO	PSO
1	Design and test a high-level collector Modulator circuit and Demodulation the signal using diode detector.	1	1,2	2
2	Test the Balanced Modulator / Lattice Modulator (Diode ring)	1	1,2	2
3	Design a Frequency modulator using VCO and FM demodulator using PLL (Use IC566 and IC565).	2	1,2,3	2
4	Design and plot the frequency response of Pre-emphasis and Deemphasis Circuits	2	1,2,3	2
5	Design and test BJT/FET Mixer	4	1,2,3	2
6	Design and test Pulse sampling, flat top sampling and reconstruction	3	1,2,3	2
7	Design and test Pulse amplitude modulation and demodulation	3	1,2,3	2
8	Generation and Detection of Pulse position Modulation	3	1,2,3	2
9	Generation and Detection of Pulse Width Modulation	3	1,2,3	2
10	PLL Frequency Synthesizer	4	1,2,3	2
11	Data formatting and Line Code Generation	3	1,2,3	2
12	PCM Multiplexer and Demultiplexer	3	1,2,3	2
13	Virtual Lab: Amplitude Modulation and De-modulation	1	1,2,3	2
14	Virtual Lab: Frequency Modulation and De modulation waveforms	1	1,2,3	2

EXPERIMENT WISE LESSON PLAN

Experiment No.1	
Name	Design and test a high-level collector Modulator circuit and Demodulation the signal using diode detector.
Objectives	Students will able to design and analyze a high-level AM collector modulator and recover the modulated signal using a diode envelope detector.
Experiment No.2	
Name	Test the Balanced Modulator / Lattice Modulator (Diode ring)
Objectives	Students will able to study and verify the operation of a balanced (diode ring) modulator for DSB-SC signal generation.
Experiment No.3	

Name	Design a Frequency modulator using VCO and FM demodulator using PLL (Use IC566 and IC565).
Objectives	Students will able to design and implement FM generation using IC566 VCO and demodulation using IC565 PLL.
Experiment No.4	
Name	Design and plot the frequency response of Pre-emphasis and Deemphasis Circuits
Objectives	Students will able to design pre-emphasis and de-emphasis networks and analyze their frequency response characteristics.
Experiment No. 5	
Name	Design and test BJT/FET Mixer
Objectives	Students will able to design and evaluate a BJT/FET mixer circuit for frequency conversion applications.
Experiment No. 6	
Name	Design and test Pulse sampling, flat top sampling and reconstruction
Objectives	Students will able to implement pulse and flat-top sampling techniques and reconstruct the original signal.
Experiment No. 7	
Name	Design and test Pulse amplitude modulation and demodulation
Objectives	Students will able to generate and demodulate a PAM signal and study its characteristics.
Experiment No. 8	
Name	Generation and Detection of Pulse position Modulation
Objectives	Students will able to generate and detect a PPM signal and analyze its time-domain behaviour.
Experiment No.9	
Name	PLL Frequency Synthesizer
Objectives	Students will able to generate and demodulate a PWM signal and study its performance.
Experiment No.10	
Name	Data formatting and Line Code Generation
Objectives	Students will able to design and analyze a PLL-based frequency synthesizer for stable frequency generation.

Experiment No.11	
Name	Data formatting and Line Code Generation
Objectives	Students will able to implement various line coding techniques for digital data transmission.
Experiment No.12	
Name	PCM Multiplexer and Demultiplexer
Objectives	Students will able to design and verify a PCM multiplexing and demultiplexing system.
Experiment No.13	
Name	Virtual Lab: Amplitude Modulation and De-modulation
Objectives	Students will able to simulate and analyze AM generation and demodulation waveforms using virtual tools.
Experiment No.14	
Name	Virtual Lab: Frequency Modulation and De modulation waveforms
Objectives	Students will able to simulate and study FM generation and demodulation waveforms using virtual tools.

LIST OF EXPERIMENTS

LAB SAFETY & USAGE INSTRUCTIONS

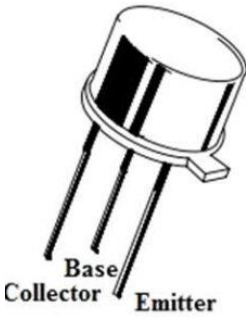
1. Students must enter the lab only during allotted lab hours with proper ID.
2. Maintain discipline and silence inside the laboratory.
3. Follow the instructions given by the lab instructor at all times.
4. No food or drinks (except water) are allowed in the lab.
5. Bags should be kept in designated areas.
6. Ensure power supply is **OFF** before making or modifying any circuit connections.
7. Never touch live wires or exposed terminals.
8. Use only the specified voltage and current ratings for components.
9. Check all connections before switching ON the power supply.
10. Report any electric shock or faulty equipment immediately.
11. Insert components properly without bending or damaging pins.
12. Avoid short circuits by ensuring proper placement of wires and components.
13. Use color-coded jumper wires for clarity (e.g., red for Vcc, black for ground).
14. Do not apply excessive force while inserting components.
15. Remove connections carefully after completing the experiment.
16. Handle instruments such as CRO, Function Generator, Power Supply, and Multimeter with care.
17. Do not change instrument settings abruptly.
18. Ensure proper grounding of equipment before use.
19. Switch OFF all devices after completing the experiment.
20. Return all components and tools to their designated place.

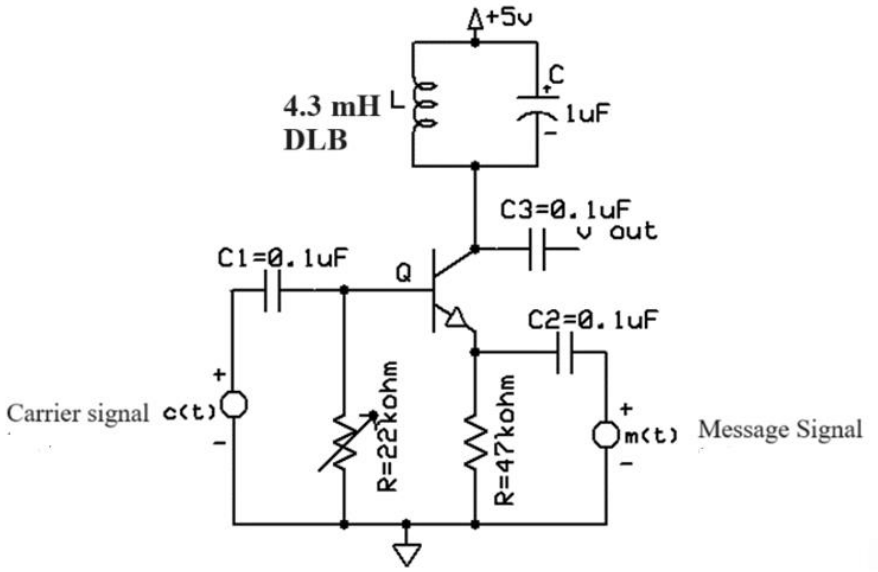
Experiment No: 1
Design and test a high-level collector Modulator circuit and Demodulation the signal using diode detector.

Apparatus Required:

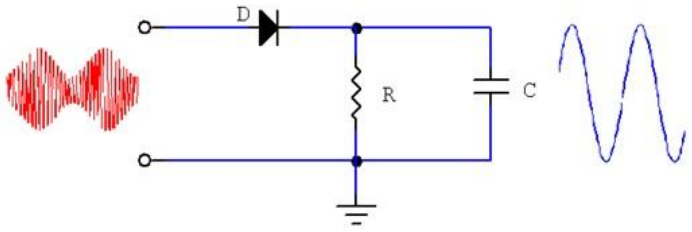
Sl. No	Components	Quantity
1	Transistor SL100	01
2	Resistor22KΩ (pot), 47KΩ	01
3	Capacitor:1μF, 0.1μF	03
4	Inductor 4.3mH from DLB	01
5	Bread board Connecting wire	01
6	CRO (40MHz), Signal generator(1MHz), DC supply(30V)	01

Circuit Diagram:

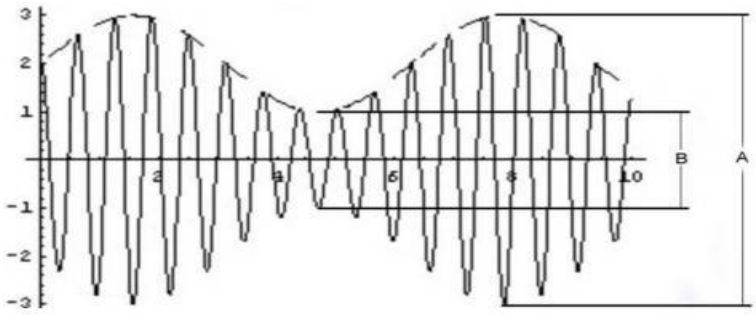




Demodulation

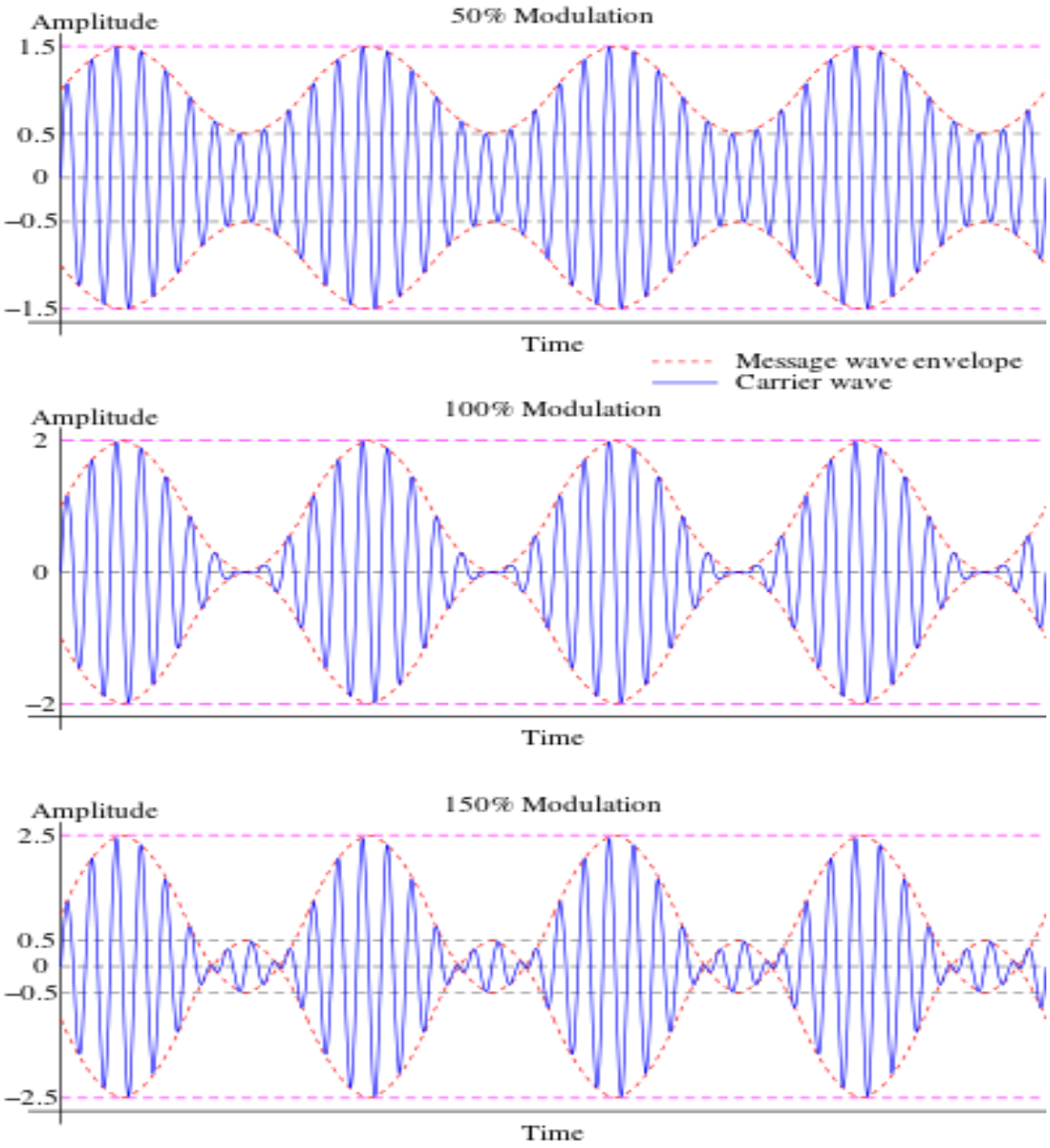


Waveforms:



where, A = Vmax and B = Vmin

From waveform modulation index, μ is given by $\mu = \frac{Vmax - Vmin}{Vmax + Vmin}$



Design: Reconstruction circuit: concider frequency of message signal as fm =237 Hz, this is the cutoff frequency of LPF. Choose C = 0.1uF and find R using $f_m = \frac{1}{2\pi RC}$

Tuned Circuit frequency is $f_c = \frac{1}{2\pi\sqrt{LC}}$ carrier wave frequency fc=2.37KHz Assume capacitor =1uF then L= 4.3mH

SL100 configuration	
Transistor Polarity	NPN
Collector−Emitter Voltage (VCEO)	50V
Collector−Base Voltage (VCBO)	60V
Continuous Collector Current (Ic)	0.5A
Emitter Base Voltage (VEBO)	5V
Power Dissipation (Pd)	800mW
Operating Temperature Range	-65 - 200°C
Output Capacitance (Cobo)	20pF
DC Current Gain (hFE)	40-300

Theory

The **amplitude modulation definition** is, an amplitude of the carrier signal is proportional to (in accordance with) the amplitude of the input modulating signal.

Amplitude modulation (AM) in a tuned amplifier circuit using a bipolar junction transistor (BJT) involves varying the amplitude of an input signal to carry information. The BJT acts as an amplifier, and the tuned circuit filters out unwanted frequencies, leaving only the desired modulated signal. By adjusting the biasing of the BJT, the modulation depth can be controlled, allowing for faithful reproduction of the modulating signal. Overall, the tuned amplifier circuit amplifies the modulated signal while filtering out noise and unwanted frequencies, enabling efficient transmission or reception of modulated signals.

the detector is a demodulator, It recovers the original signal (what was the modulating signal at the transmitter end) from the received AM signal. The detector consists of a simple half-wave rectifier which rectifies the received AM signal. This is followed by a **low pass filter** which removes (bypasses) the high-frequency carrier waveform the received signal. The resultant output of the low pass filter will be the original input (modulating) signal.

Procedure

1. Rig up the circuit as shown in the figure.
2. Set the amplitude of c(t) =7.5Vppand m(t) = 1.5Vpp using different signal generator.
3. Set the frequency of message signal m(t) =237Hz &carrier wave signal c(t) = 2.37KHzto get the AM wave.
4. Tune the 22KΩ pot till get clear waveforms of AM in CRO.
5. Note down the Vmax & Vmin.
6. Calculate the modulation index μ along with values of Vm & Vc
7. Repeat step number 4 and 5 for various value of Vmax &Vmin by varying amplitude of modulating signalm(t).
8. **Note**: If out is not getting doing all above procedure, then by pass both C1=C3=0.1uF filter capacitor.

Observation

Fc = _____Hz, Fm = _____Hz

V_{max} in volts	V_{min} in Volts	Modulation Index $\mu = \frac{V_{max}-V_{min}}{V_{max}+V_{min}}$	Amplitude of $V_m = \frac{V_{max}-V_{min}}{2}$	Amplitude of $V_c = \frac{V_{max}+V_{min}}{2}$

Note: Make sure Amplitude of carrier signal Vc is constant

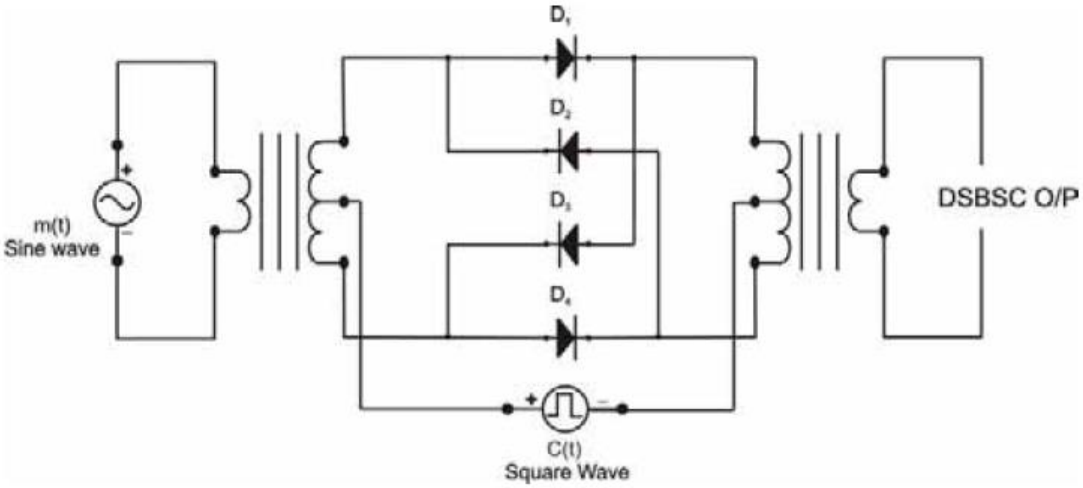
Result: Designed and verified Amplitude modulation and demodulation circuit.

Experiment No: 2
Test the Balanced Modulator / Lattice Modulator (Diode ring)

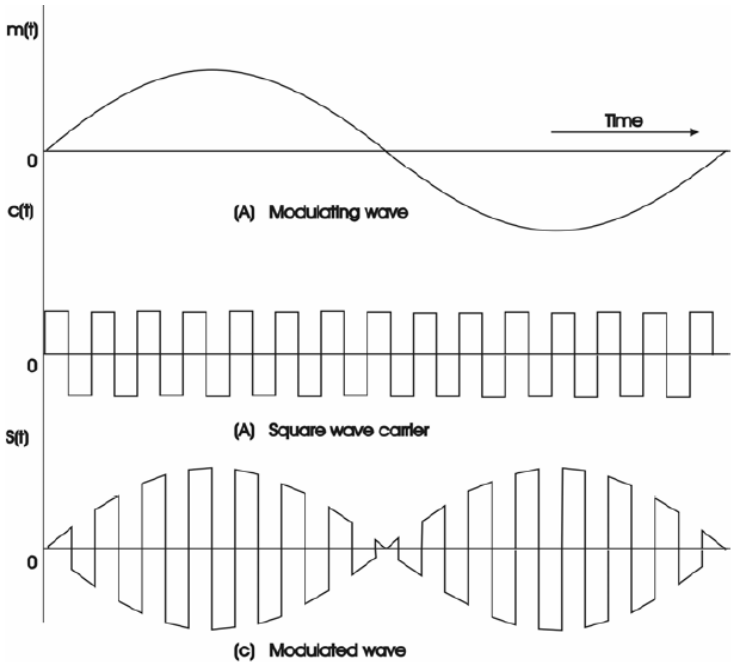
Apparatus Required:

Sl No	Components	Quantity
1	D1, D2, D3, D4 -0A79 or 1N914, or 1N4148 or BAT48	04
2	Inter stage Audio Transformer or Impedance Transformer models among any one TY-141P, or TY145P, or LT44 or 42TM018	02
3	Bread board, Connecting wire	01
4	CRO (40MHz), Signal generator(1MHz), DC supply(30V)	01

Circuit Diagram:



Waveforms



Theory:

A Diode Ring Modulator, also known as a Lattice Modulator, is a type of balanced modulator commonly used in electronic communications. Its primary function is to produce double-sideband suppressed-carrier (DSBSC) signals, which involve suppressing the radio frequency carrier while preserving the sum and difference frequencies at the output. This modulation technique allows for

efficient transmission of information while conserving power.

The Diode Ring Modulator typically consists of four diodes arranged in a ring configuration, hence the name "ring modulator." These diodes are interconnected in such a way that they form a closed loop or ring structure. The input signals, usually the carrier signal and the modulating signal, are applied to opposite pairs of diodes in the ring.

The output waveform of the Diode Ring Modulator lacks the carrier signal but contains all the information present in a traditional amplitude modulated (AM) signal. This makes it an efficient means of transmitting information, especially in applications where power saving is crucial.

Procedure

1. Connections are made as shown in figure
2. Apply modulating signal (Sine Wave Vm = 2Vp) with frequency fm = 1K Hz, and carrier signal (Square Wave Vmc = 3Vp) with frequency fc = 10 KHz (fc = 10fm).
3. Observe the phase reversal of 180^o at each Zero Crossing modulating signal in the output DSBSC signal.

Observation:
Voltage: _____
Frequency: _____

Result: Verified Test the Balanced Modulator using Diode ring.

Experiment No: 3

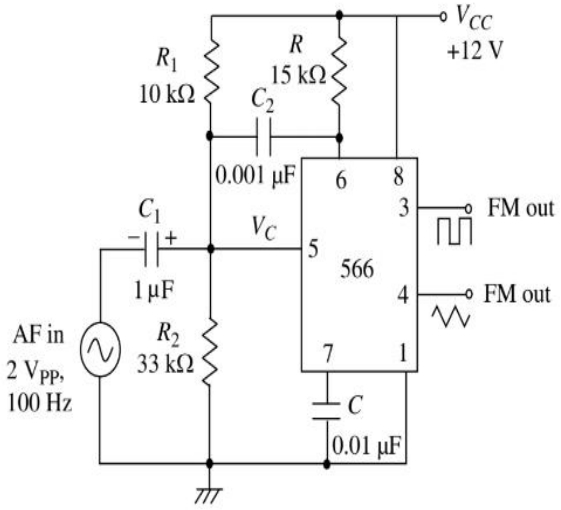
Design a Frequency modulator using VCO and FM demodulator using PLL(Use IC566 and IC565).

Apparatus Required:

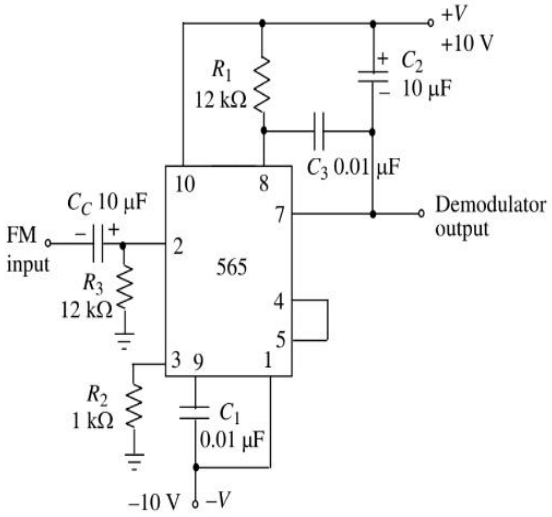
Sl No	Components	Quantity
1	PLL 565	01
2	Resistor, 12K, 12K, 1KΩ	01
3	Capacitor: 10uF, 0.01uF, 0.01uF	01
4	Bread board, Connecting wire	01
5	CRO (40MHz), Signal generator(1MHz), DC supply(30V)	01

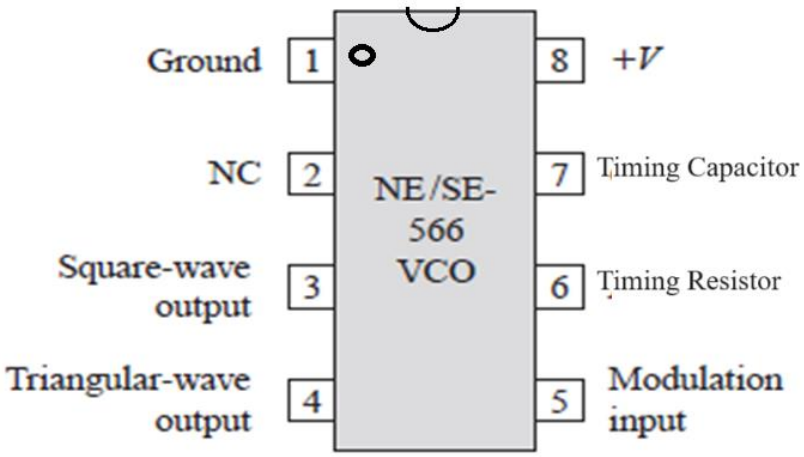
Circuit Diagram:

Modulation



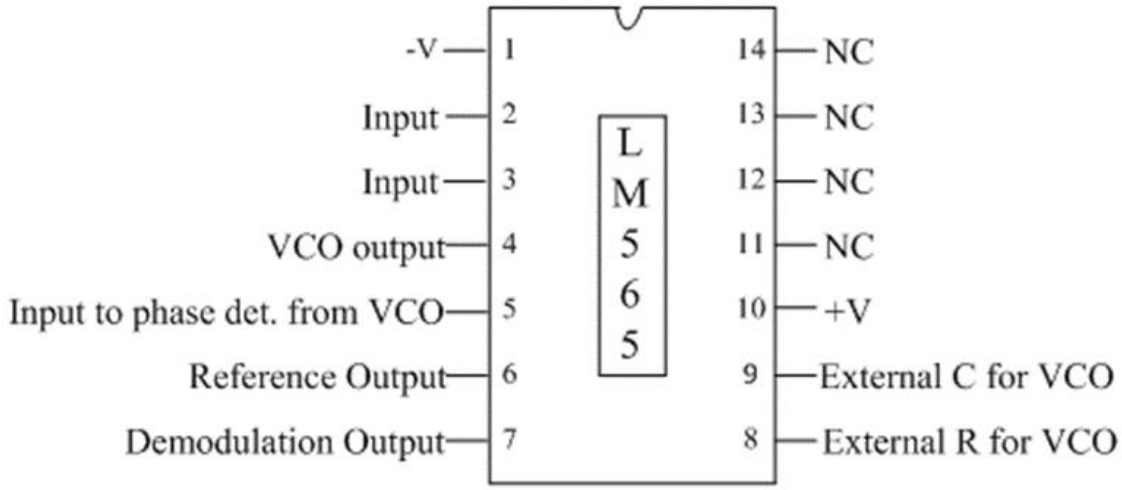
Demodulation





Specifications: VCO IC LM SE566:

Maximum operating Voltage—24V
Input voltage — 3V (P-P)
Maximum Operating Frequency - 1 MHz
The frequency can be controlled using current, voltage, resistor or capacitor
Operating temperature— 0oC to +70oC
Power dissipation—300mv.



Specification of 565

Power dissipation Max is1400mW
Availability of linear triangle signal through in-phase zero crossings
Operating voltage Max is ±12V
Temperature ranges from -55ºC to +125ºC
TTL & DTL compatible with the input of phase detector & output of square wave
Hold in the range can be adjustable from ±1% to above ±60%
The range of Operating Voltage ranges from ±5V to ±12V
The current power supply is 12.5mA
Max operating frequency of VCO is 500 KHz

Theory

In Frequency Modulation (FM), frequency of the carrier is varied in accordance with the instantaneous amplitude of the modulating signal. FM is being used very popularly in television and radio transmission systems. FM has an advantage of the least noise interference because the frequency is not susceptible to electrical disturbances. However, its bandwidth requirement is very high compared to AM.

FM can be generated by IC 566. It is a Voltage Controlled Oscillator (VCO) which generates a triangular wave and a square wave decided by the values of the resistor and capacitor externally connected to it. Typical amplitude of triangular wave is 2.4 V and that of square wave is 5 V.

For the demodulation of FM signal, the centre frequency of the VCO of the demodulator. The variation of the input frequency from the centre frequency is converted into the variation of the voltage by the phase comparator of the demodulator. FM input is coupled to the pin 2 using R3 Cc network as shown in figure.

Major advantage of FM demodulator using PLL is linearity. Linearity is governed by the voltage to frequency characteristic of the VCO within the PLL. As the frequency deviation of the incoming signal swings over a small portion of the PLL bandwidth, and the characteristic of the VCO can be made relatively linear, the distortion levels from PLL demodulators are normally very low.

Design of Modulator

As per the data sheet: $f_o = \dfrac{5(V_{CC} - V_C)}{2V_{CC}RC}$

Also, $3/4V_{CC} < V_C < V_{CC}$ and $2\ \text{k}\Omega < R < 20\ \text{k}\Omega$.

Take $V_{CC} = +12$ V, $f_o = 5$ kHz and $V_C = 3/4V_{CC} = 9$ V.

Choose $R = 15\ \text{k}\Omega$. Then, we get $C \approx 0.01\ \mu\text{F}$.

Take $C_1 = 1\ \mu\text{F}$ and $C_2 = 0.001\ \mu\text{F}$.

i.e., $V_{CC} \times \dfrac{R_2}{R_1 + R_2} = \dfrac{3}{4}V_{CC}$

Choose $R_1 = 10\ \text{k}\Omega$. Then, we get $R_2 = 30\ \text{k}\Omega$. Take $R_2 = 33\ \text{k}\Omega$ as standard.

Design of Demodulator

Design: Let $+V = +10$ V and $-V = -10$ V

Let the centre frequency of the FM be $f_o = \dfrac{0.3}{R_1C_1} = 2.5$ kHz

Take $C_1 = 0.01\ \mu\text{F}$. Then we get, $R_1 = 12\ \text{k}\Omega$.

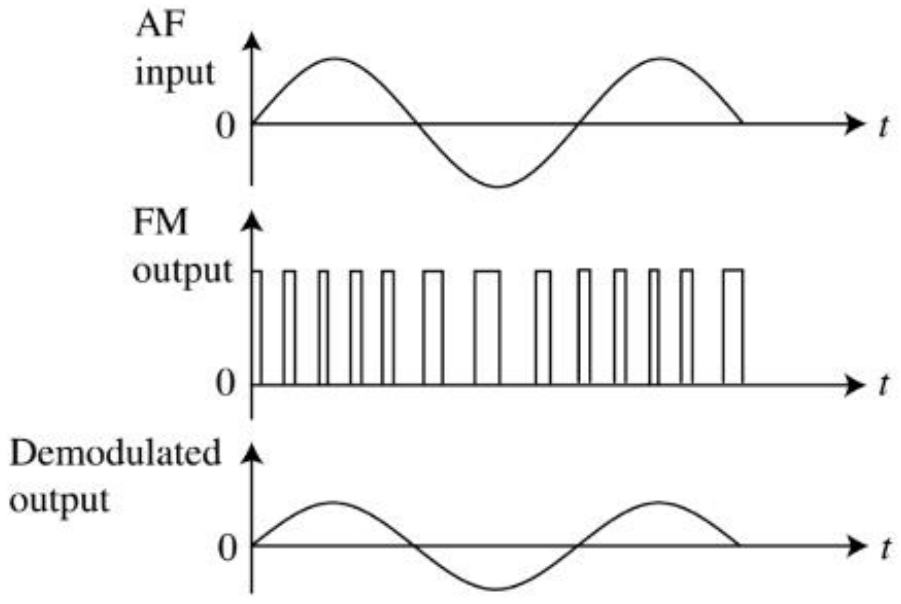
The value of R_1 satisfies the required condition $2\ \text{k}\Omega < R_1 < 20\ \text{k}\Omega$, as per data sheet. Take $C_C = 10\ \mu\text{F}$ and $R_3 = 12\ \text{k}\Omega$ to couple the input to the IC.

For demodulator circuit, since centre frequency is same as that of modulator, frequency determining components are the same.

Take same set of coupling capacitor and resistor.

Take $R_2 = 1\ \text{k}\Omega$, $C_2 = 10\ \mu\text{F}$ and $C_3 = 0.01\ \mu\text{F}$

Waveforms



Procedure

1 Set up the circuit on a breadboard. Without feeding the modulating signal, note the amplitude and frequency of the output square wave at pin 3. Note free-running frequency fo.

2 Apply the modulating signal of 2 Vpp, 100 Hz. Observe the FM output at pin 3 and pin 4, on CRO screen.

3 Note maximum and minimum frequencies, fmax and fmin of FM output. Calculate frequency deviation Δf = fmax - fmin. Calculate the modulation index mi = Δf/fm where fm is modulating signal frequency.

4 Set up FM demodulator and apply the FM signal to it. Observe the demodulated output.

Observation

AC in (v)	A_m in (v)	F_{max} in Hz	F_{min} in Hz	Δf	modulation index mi

Result: Verified FM modulation and demodulation circuit using PLL.

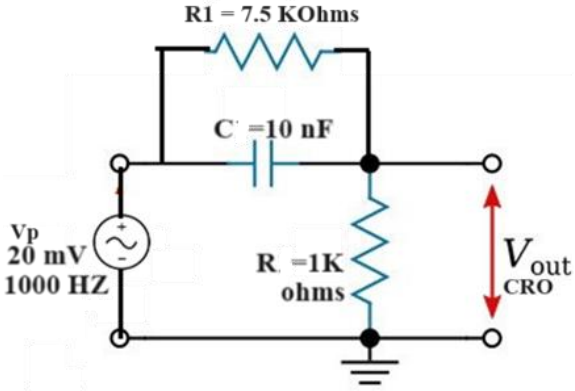
Experiment No: 4
Design and plot the frequency response of Pre-emphasis and Deemphasis Circuits

Apparatus Required:

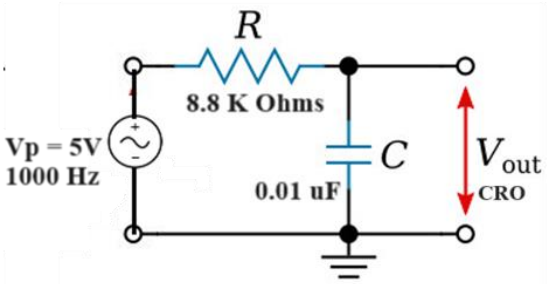
Sl No	Components	Quantity
1	Opamp uA741	01
2	Capacitor 0.01uF,	01
3	Resistors: 1K, 7.5k, 8.8 kΩ	01
4	Bread board, Connecting wire	01
5	CRO (40MHz), Signal generator(1MHz), DC supply(30V)	01

Circuit Diagram

requency response of Pre-emphasis Circuit



equency response of Deemphasis Circuit

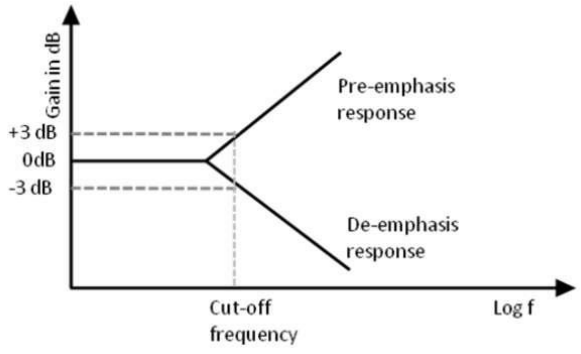


Design:

$Use formula to claculate cut of frequency of both High pass and Low pass Filter by f_c = \frac{1}{2\pi RC}$

R1 assume that 10 times of output impedance of function generator 600 ohms ie R1=10x600=6K ohms assumed that 7.5K ohms.

Nature of Graph



Theory

The pre-emphasis and de-emphasis help to improve the quality of any communication especially audio signals on the transmitter and receiver sides. The presence of noise is also an issue in FM and we know that noise usually has higher amplitude and higher frequency. When the amplitude of a high-frequency noise is higher than the current component in the

modulation signal, it causes high-frequency interference. To deal with this issue, most FM circuits use a technique called pre-emphasis during transmission and de-emphasis during receiving. Pre-emphasis and de-emphasis circuits are commonly used in FM transmitters and receivers to improve the output signal-to-noise ratio.

The pre-emphasis circuit is actually a high pass filter and de-emphasis circuit a low pass filter. The amount of pre-emphasis and de-emphasis used is defined by the time constant of a simple RC filter circuit. Simple pre-emphasis and de-emphasis circuits using op-amp are given in the diagram

Procedure

1) Connect the circuit as per circuit diagram as shown in Pre-emphasis Circuit.
2) Apply the sinusoidal signal of amplitude 20mV as input signal to pre-emphasis circuit.
3) Then bty increasing the input signal frequency from 500Hz to 20KHz, observe the output voltage (Vo) and calculate gain (20log (Vo /Vi)).
4) Plot the graph between gain Vs frequency.
5) Repeat above steps 2 to 4 for de-emphasis circuit by applying the sinusoidal signal of 5V as input signal.

Observation

For Pre-emphasis Vi = Vp = 20mV

Frequency F in Hz	Output Voltage Vo in mV	Output gain in dB = 20log (Vo/Vi)

For De-emphasis Vi = Vp = 5V

Frequency F in Hz	Output Voltage Vo in mV	Output gain in dB = 20log (Vo/Vi)

Result: Designed and verified and plot the frequency response of Pre-emphasis and Deemphasis Circuits

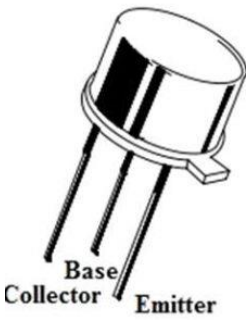
Experiment No: 5

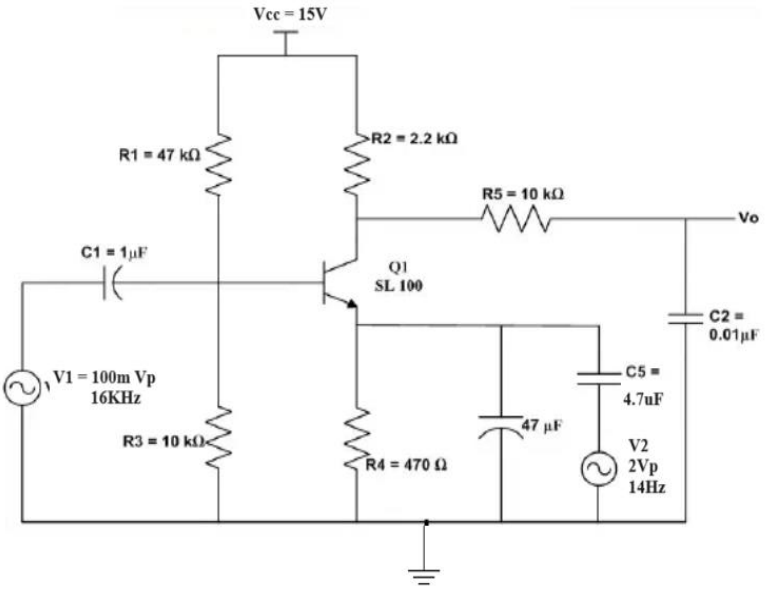
To Design and test BJT/FET Mixer

Apparatus Required:

Sl No	Components	Quantity
1	Transistor SL100	01
2	Resistors 47K, 10K, 470, 2.2K, 22K, 10KΩ	01
3	Capacitors: 1uF, 47uF, 4.7uF, 47uF, 0.01uF	01
4	Bread board Connecting wire	-
5	CRO (40MHz), Signal generator(1MHz), DC supply(30V)	-

Circuit Digram





Design

Given, $\mathbf{V_{CE} = 5\ V}$ and $\mathbf{I_C = 2\ mA}$ Assume β = 100

$\mathbf{V_{CC} = 2V_{CE}} = 2 \times 5 = 10V$

Let $V_{RE} = 10\%\ V_{CC} = 1$ V

$R_E = V_{RE} / (I_C + I_B)$

$I_B = I_C / \beta = 2$ mA / 100 = 20 μA

$R_E = 1 / (2\ m = 20\ \mu) = 495\ \Omega$

Choose $\mathbf{R_E = 470\ \Omega}$

Apply KVL to collector loop

$V_{CC} - I_C R_C - V_{CE} - V_E = 0$

$R_C = (V_{CC} - V_{CE} - V_E) / I_C = (10 - 5 - 1) / 2\ m$

$R_C = 2$ kΩ Choose $\mathbf{R_c = 2.2\ k\Omega}$

$_{BE} = 0.6$)

$R_2 = 8.8$ kΩ choose $\mathbf{R_2 = 10\ k\Omega}$

$R_1 = (V_{CC} - V_{R2}) / I_{R1} = (10 - 1.6) / 200\ \mu A$

$R_1 = 42$ kΩ choose $\mathbf{R_1 = 47k\Omega}$

$X_{CE} << R_E$

$X_{CE} = R_E / 10$

$1 / (2\pi f C_E) = 470 / 10$ let f = 100 Hz

$C_E = 33\ \mu F$ choose $\mathbf{C_E = 47\ \mu F}$

Note : Choose coupling cpacitors in such way that Reactance of coupling capacitors Xc1 and Xc5 should be less than 15Ω

Theory

In electronics, a mixer, or frequency mixer, is an electrical circuit that creates new frequencies from two signals applied to it. In its most common application, two signals are applied to a mixer, and it produces new signals at the sum and difference of the original frequencies. Active mixers use an amplifying device (such as a transistor or vacuum tube) that may increase the strength of the product signal. Active mixers improve isolation between the ports, but may have higher noise and more power consumption. Mixers may also be classified by their topology:

An unbalanced mixer, in addition to producing a product signal, allows both input signals to pass through and appear as components in the output.

A single balanced mixer is arranged with one of its inputs applied to a balanced (differential) circuit so that either the local oscillator (LO) or signal input (RF) is suppressed at the output, but not both.

A double balanced mixer has both its inputs applied to differential circuits, so that neither of the input signals and only the product signal appears at the output.[1] Double balanced mixers are more complex and require higher drive levels than unbalanced and single balanced designs.

Procedure

1. Connections are made as shown in the circuit diagram.
2. Apply the input signals as mentioned in the circuit diagram.
3. Observe the output waveforms in CRO
4. Measure the output frequency, it has to be equal to $\Delta f = f_1 - f_2$
5. Repeat the steps 3 and 4 by decreasing frequency of V2 in the step of 1KHz.

Observation

Frequncy F1 in Hz	Frequency F2 in Hz	Output frequency $\Delta f = f_1 - f_2$ in Hz
16Hz	14Hz	
16Hz	13Hz	
16Hz	12Hz	
16Hz	11Hz	
16Hz	10Hz	
16 Hz	09Hz	

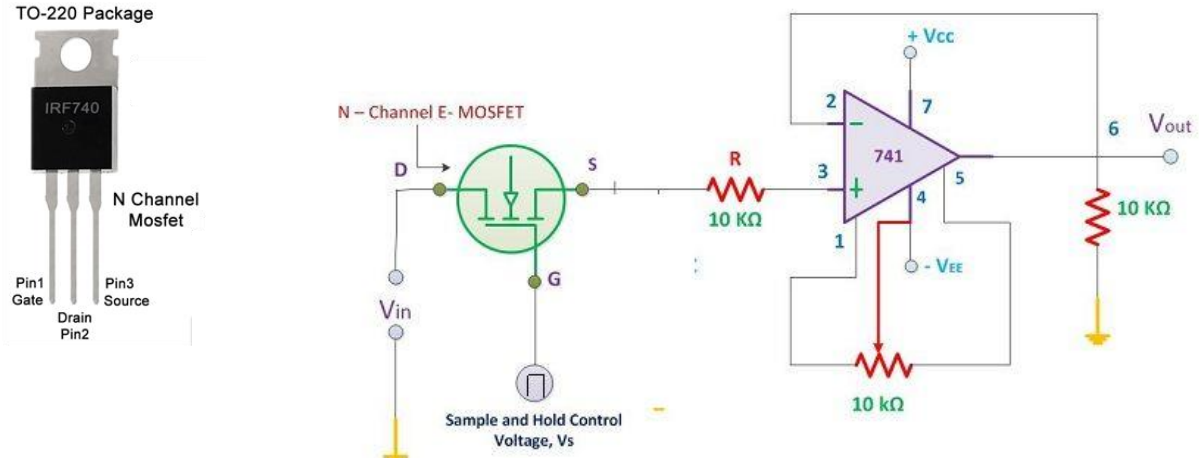
Result: Design and tested BJT frequency Mixer

Experiment No: 6
Design and test Pulse sampling, flat top sampling and reconstruction.

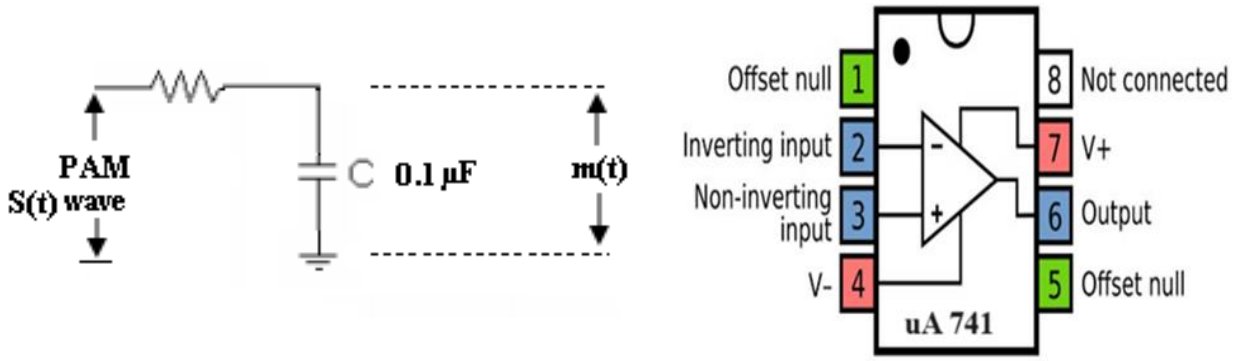
Apparatus Required:

Sl No	Components	Quantity
1	n-EMOSFET (IRF740), OP-AMP(µA741)	1
2	Resistor22KΩ, 10KΩ(POT),47KΩ	1
3	Bread board, connecting wires	1
4	CRO, Function Generator and DC Supply	1
5	Capacitor 0.1uF (Electrolyte)	1

Circuit: Pulse sampling circuit diagram



Reconstruction of Pulse sampling circuit diagram and Pin Diagram of uA 741



Design

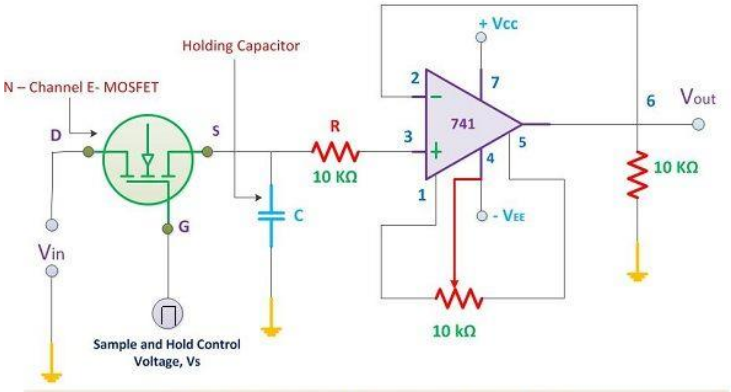
Sampling circuit is a voltage follower circuit and assume uA741 output current is 20 mA hnece assume R and RL vlaue is to assume 1K to 10KΩ as per avilability and C electrolyte capacitor of 0.1uF.

Reconstruction circuit: concider frequency of message signal as fm =100 Hz, this is the cutoff frequency of LPF.

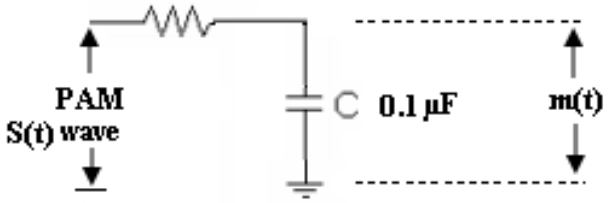
Choose C = 0.1uF and find R using $f_m = \frac{1}{2\pi RC}$

n-EMOSFET (IRF740) Specification	**uA741 Specification**
Maximum Drain Source Voltage -- 400 V Maximum Continuous Drain Current --10 A Maximum Gate Source Voltage -- ±20V Power Dissipation -- 125 W Channel Type -- N Category -- Power MOSFET Frequency range -- 1Hz-150kHz	Supply Voltage: ±18V Voltage Gain (Open Loop):200,000 (or 106 dB) CMRR: 90dB Differential voltage amplification: 200V/mv Supply Current: 1.5mA Available in 8-Pin PDIP, Unity Gain Bandwidth (GBW): 1 MHz Input resistance Ri=2M Ohm Output resistance Ro=75ohm

Circuit: Flat top sampling



Reconstruction of Flat top sampling



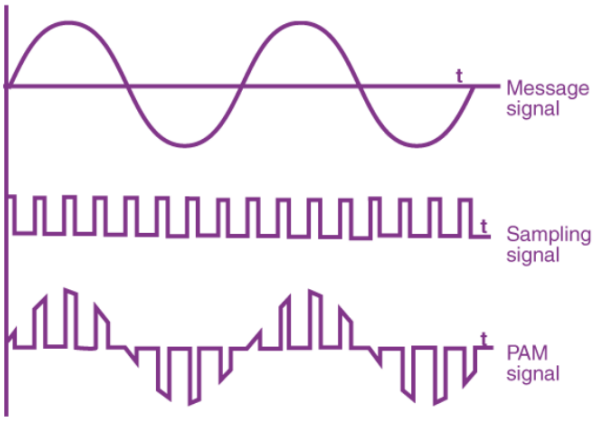
Design

Sampling circuit is a voltage follower circuit and assume uA741 output current is 20 mA hnece assume R and RL vlaue is to assume 1K to 10KΩ as per avilability.

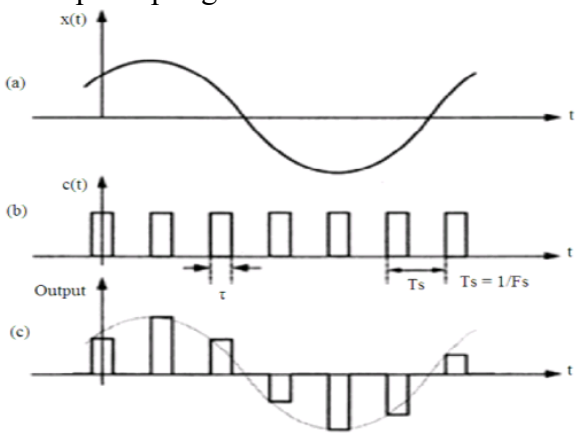
Reconstruction circuit: concider frequency of message signal as fm =100 Hz abd this is the cutoff frequency of LPF.

Choose C = 0.1uF and find R using $f_m = \frac{1}{2\pi RC}$

Waveforms Pulse sampling



Flat top sampling



Theory

The sampling theorem can be defined as the conversion of an analog signal into a discrete form by taking the sampling frequency as twice the input analog signal frequency. Input signal frequency denoted by Fm and sampling signal frequency denoted by Fs. If the sampling frequency (Fs) equals twice the input signal frequency (Fm), then such a condition is called the Nyquist Criteria for sampling. When sampling frequency equals twice the input signal frequency is known as "Nyquist rate". If the sampling frequency (Fs) is less than twice the input signal frequency, such criteria called an Aliasing effect.

In flat-top sampling or rectangular pulse sampling, the top of the samples remains constant and is equal to the instantaneous value of the baseband signal x(t) at the start of sampling. During transmission, noise is introduced at top of the transmission pulse which can be easily removed if the pulse is in the form of flat top. Here, the top of the samples are flat i.e. they have constant amplitude. Hence, it is called as flattop sampling or practical sampling. Flat top sampling makes use of sample and hold circuit."

Procedure

1. Before wiring the circuit checks all the components using multi meter.
2. As per design set the values and do the connections as shown in circuit diagram.
3. Set the carrier amplitude or sampling signal to around 4 Vp and frequency, fs = 1KHz.
4. Set the message signal amplitude to around 2Vp and frequency, fm= 100 HZ.
5. Connect the CRO at the pin number 6 of OP-AMP and observe the waveform for both circuits.
6. Connect this output to the reconstruction filter and observe the waveforms.

Observation:

t = _____

T_s = _____

Result: Verified and tested Pulse sampling, flat top sampling and reconstruction circuits.

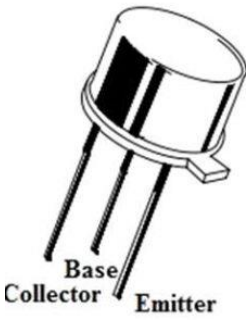
Experiment No: 7
Design and test Pulse amplitude modulation and demodulation.

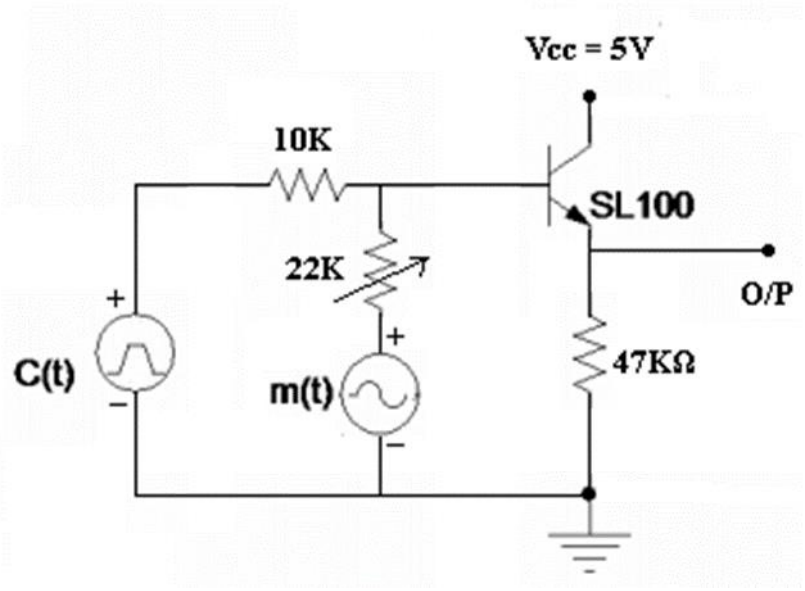
Apparatus Required:

Sl No	Components	Specification
1	Transistor	SL100
2	Resistor	22K, 10K,47K
3	Capacitor	0.1µF
4	Bread boardConnecting wire	--
5	CRO (40MHz), Signal generator(1MHz), DC supply(30V)	--

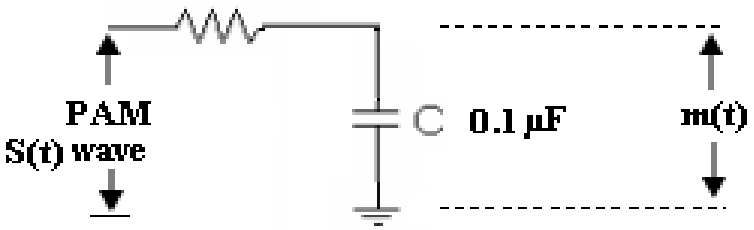
Circuit diagram

Pulse Amplitude Modulation Circuit





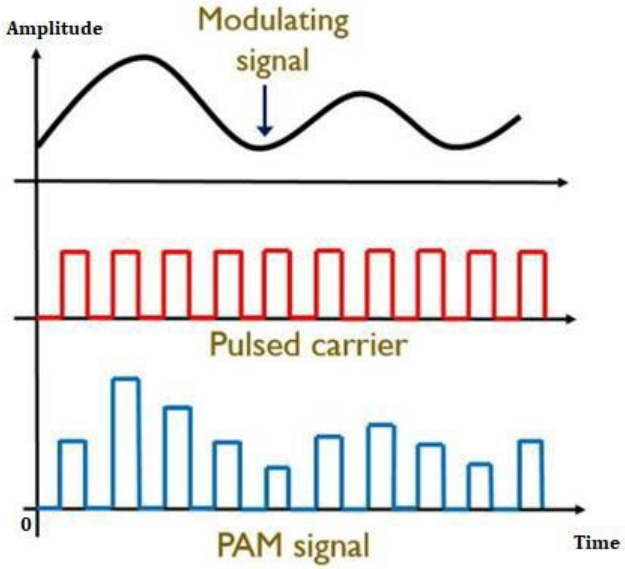
Demodulation circuit

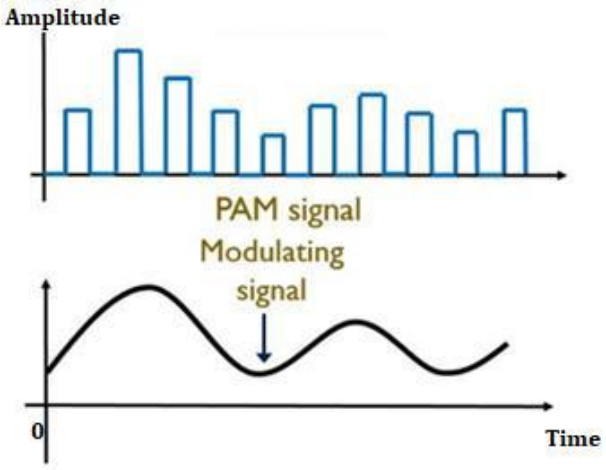


Design

Reconstruction circuit: concider frequency of message signal as fm =100 Hz abd this is the cutoff frequency of LPF. Choose C = 0.1uF and find R using $f_m = \frac{1}{2\pi RC}$

Waveforms





Theory :

Pulse amplitude modulation is a technique in which the amplitude of each pulse is controlled by the instantaneous amplitude of the modulation signal. It is a modulation system in which the signal is sampled at regular intervals and each sample is made proportional to the amplitude of the signal at the instant of sampling. This technique transmits the data by encoding in the amplitude of a series of signal pulses. In PAM, the signal amplitudes can be changed based on the modulating signal. The pulse train works like a periodic switching signal toward the modulator. Once it is switched ON, and then allows the samples of modulating signals to supply toward the output. The pulse train's periodic time is called the sampling period.

PAM is mostly applied in non-based modulating transmission of digital data and applications replaced by pulse-code modulation and pulse-position modulation. Particularly all phone modems faster than 300 bit/s use quadrature amplitude modulation.

Procedure

1. Before wiring the circuit checks all the components using multi meter.
2. As per design set the values and do the connections as shown in circuit diagram.
3. Set the pulsed carrier amplitude to around 5V (p-p) and frequency, fc = 1 KHz.
4. Set the message signal amplitude to around 3 V (p-p) and frequency, fm = 100HZ.
5. Check the modulated and demodulated output waveform.

Result: Verified and tested Pulse Amplitude Modulation circuit and its reconstruction circuit.

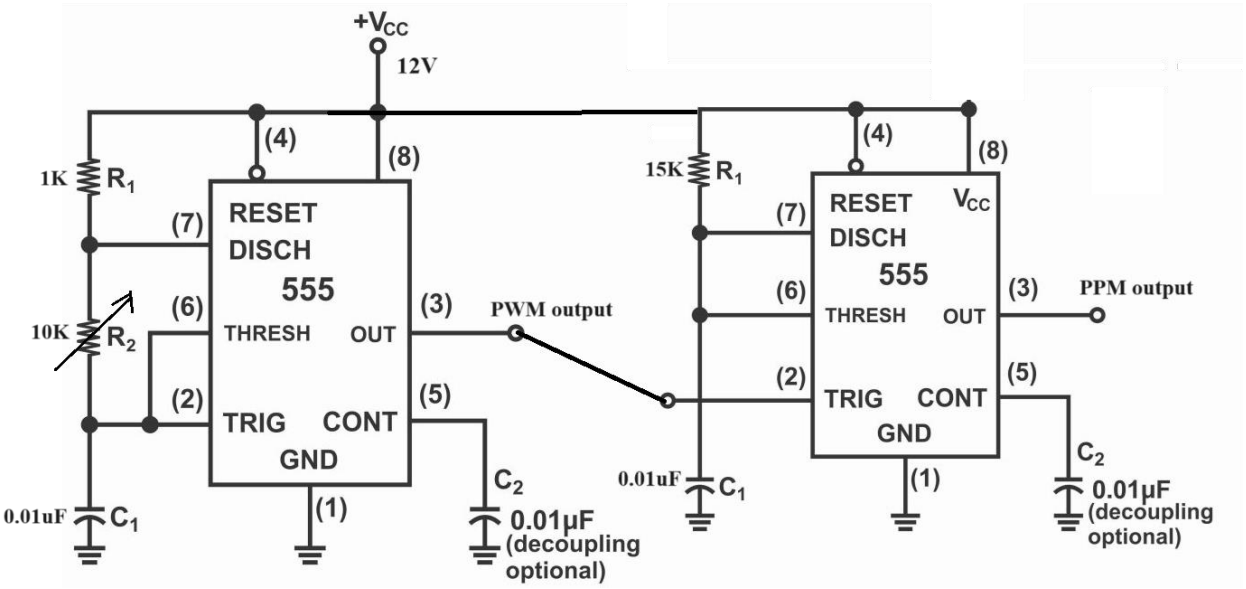
Experiment No: 8
Generation and Detection of Pulse position Modulation

Apparatus Required:

Sl No	Components	Quantity
1	555 Timer	02
2	Diode 1N4148 Fast Switching Diode	01
3	Resistor, 1K, 10K,15K,	01
4	Capacitor 0.01uF	4
5	Bread board , Connecting wire	-
6	CRO (40MHz), Signal generator(1MHz), DC supply(30V)	01

Circuit Diagram

Pulse Position Modulation Circuit



Design

Design of Summing amplifier PPM

Design of 555 timer

555 timer function as monostable multivibrator we have $T = \frac{1}{6.8K} = 1.1RC = 1.1 \times R_1 \times 0.01\mu$

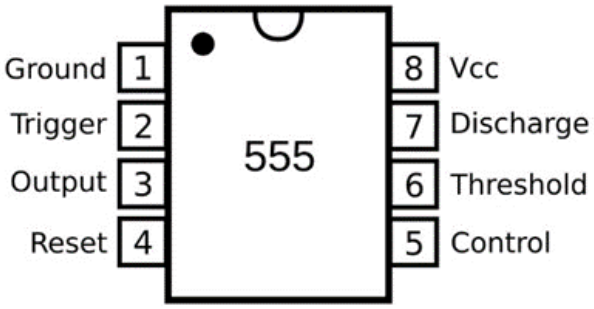
R1 = 13.6KΩ use R1=15KΩ

555 timer function as an astable multivibrator

$T_{on} = 0.693(R_1 + R_2) \times C_1$ *and* $T_{off} = 0.693(R_1 \times C_1)$ *we know that* $f = \frac{1}{T} = \frac{1}{T_{on} + T_{off}}$

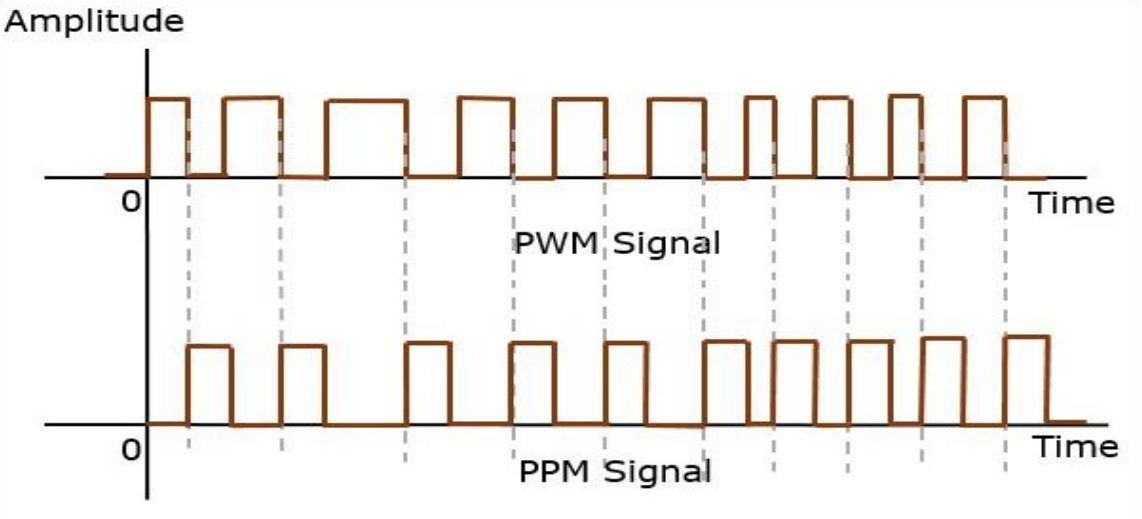
$= \frac{1.44}{(R_1 + 2R_2)C_1}$ chose R1 as 1 KΩ resistor, R2 as a 10 KΩ Potentiometer and C as 10nF (0.01μF) Capacitorto get F= 6.87 KHz and $Duty\ cycle = \frac{T_{on}}{T_{on}+T_{off}} = \frac{76.26\mu}{145.6\mu} = 52.38\%$

Specification of 555 Timer	
Supply voltage (VCC)	4.5 to 15 V



Supply current (VCC = +15 V)	10 to 15 mA
Output current (maximum)	200 mA
Maximum Power dissipation	600 mW
Max Frequency (Astable)	500-kHz to 2-MHz
Compatibility	TTL and CMOS logic
Operating temperature	0 to 75 °C

Waveform of PPM



Theory

Pulse position modulation is a modulation technique in which the position of pulse varies according to instantaneous value of amplitude of sampled modulating signal. The 555 timer IC is used for PPM generation. It is an integrated circuit used in a variety of timer, delay, pulse generation, and oscillator applications. A mono-stable vibrator is used in PPM for converting Pulse Width modulated signal to Pulse position modulated signal. We require one a stable multi-vibrator to generate PWM signal and one monostable multi-vibrator to get desire PPM

The monostable is negative edge triggered. Hence, corresponding to each trailing edge of PWM

signal, the monostable output goes high. It remains high for a fixed time decided by its own R1C1 components. Thus, as the trailing edges of the PWM signal keep shifting in proportion with the modulating signal, the PPM pulses also keep shifting, as shown in Waveform of PWM and PPM. This modulation technique finds its application in air traffic control systems, in radio control and in military applications.

Procedure

1 Make the connection as per Circuit diagram.
2 First check output of astable multivibrator and observe PWM wave by varying Pot R2
3 Observe the PWM output waveforms for different duty cycle.
4 Next connect output of PWM Pin #3 is fed to triggering input of IC 555 timer Pin #2 to result is PPM.
5 The output is taken at terminal Pin #3 of second timer 555 IC.
6 The PPM wave is observed on CRO by varying Pot R2 from astable555Timer

Result: Verified the output of the Pulse width Modulation and Pulse Position Modulation circuits using astable and monostable operating modes 555 timers.

Experiment No: 9

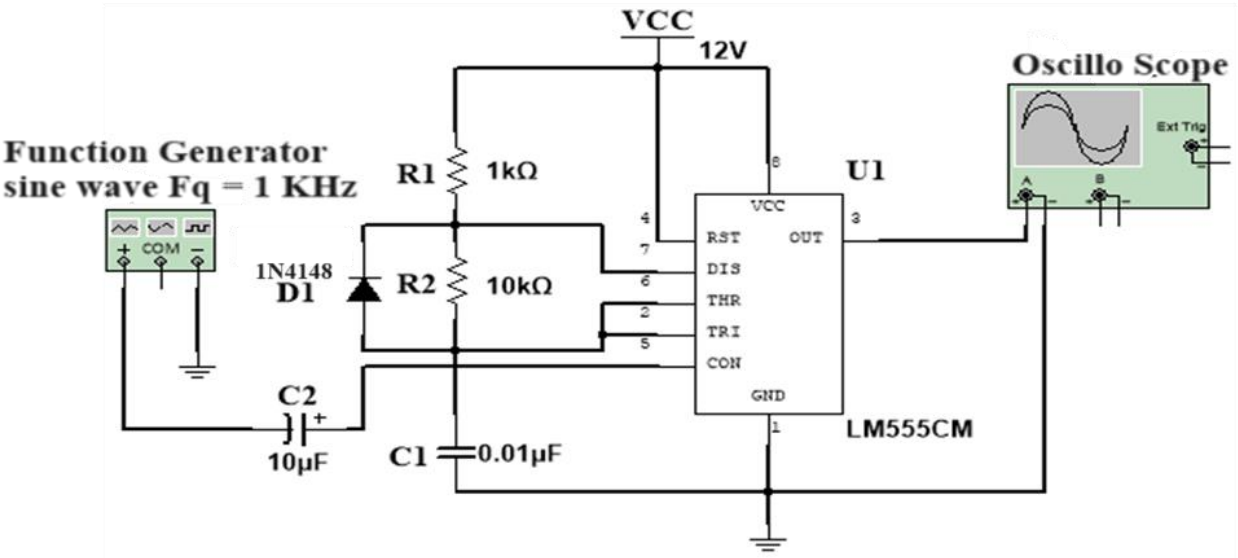
Generation and Detection of Pulse Width Modulation

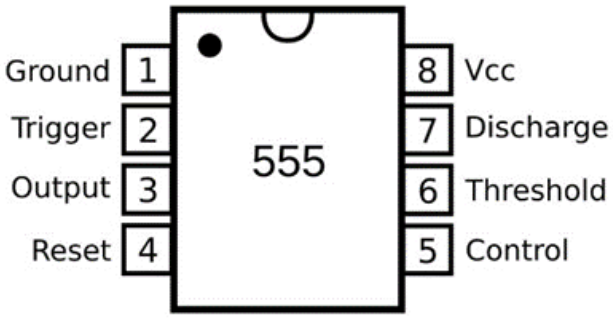
Apparatus Required:

Sl No	Components	Quantity
1	555 Timer	01
2	Diode 1N4148	01
3	Resistor, 10KΩ, 1KΩ	01
4	Capacitor 0.01uF	01
5	Bread board	01
6	CRO (40MHz), Signal generator(1MHz), DC supply(30V)	01
7	Connecting wires	-

Circuit Diagram

Pulse width Modulation Circuit





Design

Design of a 555 timer as an Astable Multivibrator for PWM

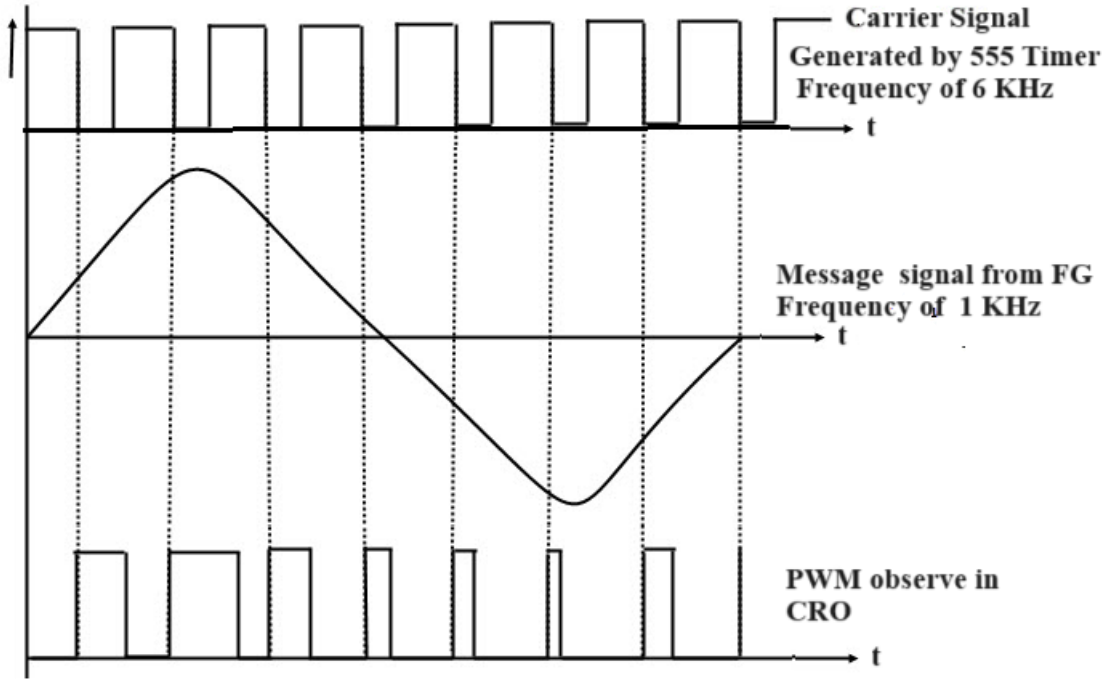
$$T_{on} = 0.693(R_1 + R_2) \times C_1 \; and \; T_{off} = 0.693(R_1 \times C_1) \; we \; know \; that \; f = \frac{1}{T} = \frac{1}{T_{on} + T_{off}}$$

$= \frac{1.44}{(R_1 + 2R_2)C_1}$ chose R1 as 1 KΩ resistor, R2 as a 10 KΩ Potentiometer and C as 10nF

(0.01µF) Capacitorto get F= 6.87 KHz and$Duty\ cycle = \frac{T_{on}}{T_{on}+T_{off}} = \frac{76.26\mu}{145.6\mu} = 52.38\%$

Specification of 555 Timer	
Supply voltage (VCC)	4.5 to 15 V
Supply current (VCC = +15 V)	10 to 15 mA
Output current (maximum)	200 mA
Maximum Power dissipation	600 mW
Max Frequency (Astable)	500-kHz to 2-MHz
Compatibility	TTL and CMOS logic
Operating temperature	0 to 75 °C

Waveform of PWM



Theory

PWM stands for pulse width modulation, a process which involves the control of the pulse widths, or the ON/OFF periods or logical outputs that's generated from a particular source such as an oscillator circuit or microcontroller. As we know that the ON time of the IC 555 PWM circuit is decided by the time taken by its capacitor to charge at the 2/3rd Vcc level through pin#7 resistors R1 and R2, and the OFF time is determined by the discharging time of the capacitor below 1/3rd Vcc through R2.

The variable voltage at control input (pin 5) of IC555 in a stable mode then the width of output

pulse will change accordingly. For low frequency audio signal to the control input, internally the control voltage will change the reference voltage of comparator so charging time of capacitor changes every time and so width of o/p pulse also changes. So here as shown in figure the audio signal is applied at the control input through positive clamper circuit. The values of R1, R2 and C1 are chosen to generate pulse train of around 6.87 KHz. The width of o/p pulse will change in accordance with the amplitude of audio signal as shown in figure in Circuit Diagram.

Procedure

1. Make the connection as per Circuit diagram.
2. Observe the output of pulse of train in CRO with frequency of 6.8KHz without connecting function generator.
3. Now Set the M(t) = 4Vpp with frequency of 1 KHzfrom signal generator and connect to 555 timers.
4. The output is taken at terminal 3 of timer 555 IC observe the PWM output waveforms in CRO.
5. By varying amplitude of M(t) in the function generator in the step of 0.5VPP observe the change of width of the pulses.

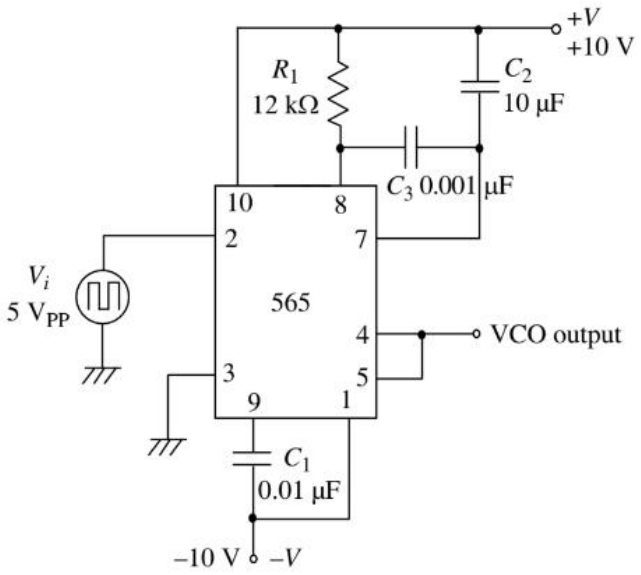
Result: Verified the output of the Pulse width Modulation circuits using Astable mode 555 timers.

Experiment No: 10
To realize PLL Frequency Synthesizer

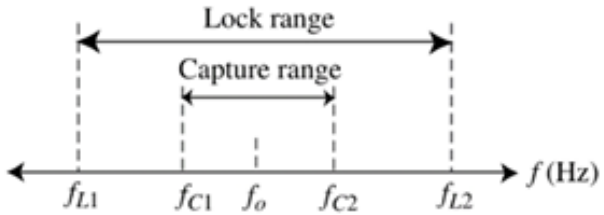
Apparatus Required:

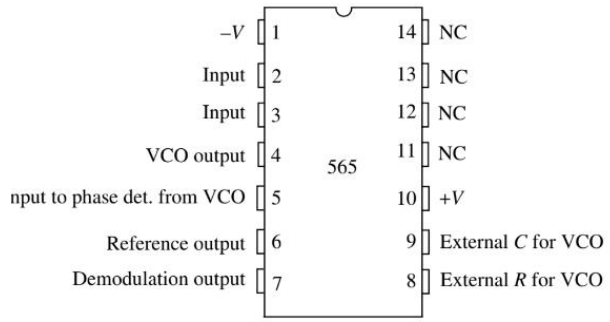
Sl No	Components	Quantity
1	CRO, Function Generator, Breadboard	1
2	Resistance: 12K,	1
3	Capacitor: 0.001uF, 10uF and 0.01uF	1
4	RPS and connecting wires	1

Circuit Diagram



Nature of Graph and Pin Diagram





Design: Take $+V = +10$ V and $-V = -10$ V

Assume the free running frequency $f_o = \dfrac{1.2}{4R_1C_1} = 2.5$ kHz

Take $C_1 = 0.01$ μF. Then we get, $R_1 = 12$ kΩ.

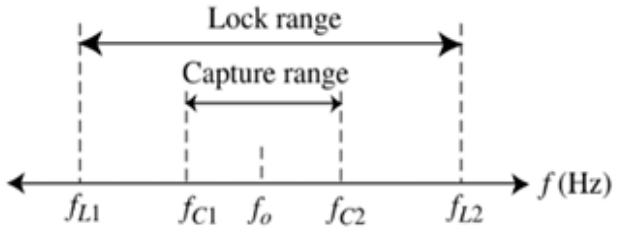
The value of R_1 satisfies the required condition $2\ \text{k}\Omega < R_2 < 20\ \text{k}\Omega$

Take $C_3 = 0.001$ μF and $C_2 = 10$ μF

The range of frequencies that the loop will remain in lock

$$= \pm\frac{8f_o}{V} = \frac{8 \times 2.5 \times 10^3}{10 - (-10)} = 1 \text{ kHz}$$

Graph



Therefore, lock range, $f_L = 2$ kHz

The range of frequencies that the loop will acquire lock

$$= \frac{\sqrt{f_L}}{\sqrt{2 \times \pi \times 3.6 \times 10^3 \times C_2}} = \frac{\sqrt{10^3}}{\sqrt{2\pi \times 3.6 \times 10^3 \times 10 \times 10^{-6}}} = 66.5 \text{ Hz}$$

Therefore, capture range, $f_C = 133$ Hz.

Theory

A frequency synthesizer is an electronic circuit used to generate precise and stable output frequencies based on a reference frequency or multiple reference frequencies. It's widely used in communication systems, radar systems, and various electronic devices.

A frequency synthesizer generates an output frequency by combining and manipulating the frequency of one or more reference signals using various techniques such as phase-locked loops (PLLs), direct digital synthesis (DDS), or fractional-N synthesis.

Initially, the PLL is unlocked, and the output frequency may differ from the reference frequency. The phase detector compares the phases of the reference and output signals, generating an error voltage proportional to the phase difference. The LPF filters and smoothes this error voltage to provide a DC voltage that represents the frequency error. This voltage is then fed to the VCO, which adjusts its frequency in response to minimize the phase error. As the loop continues to operate, the PLL locks, maintaining a stable phase and frequency relationship between the reference and output signals.

Procedure

1. Setup the circuit and observe the output at pin 4 or pin 5 and note down the VCO frequency. It is the free running frequency f0 without any input signal.
2. Apply a signal input to pin 2 square waves of 5 Vpp, 1KHz and slowly vary its frequency from low to high. and note down fc1 and fL2.
3. Keep on increasing input signal frequency up to the point when VCO frequency starts following the input frequency. Note down the frequency at which the process starts. This frequency is the lower capture frequency fc1 of the PLL.
4. Keep increasing the signal frequency further. A frequency is obtained when the VCO frequency stops following the input signal frequency. This frequency is the upper lock-in frequency fL2. Observe that above the upper lock-in frequency the VCO signal will not at all follow the input frequency variation.
5. Now start reducing the input frequency slowly up to the value when again the VCO frequency starts following the input frequency. This frequency is upper capture frequencyfc2.

6. Keep reducing the signal frequency further to the value when VCO frequency stops following the input signal frequency and will not follow further. This value of the frequency is lower lock-in frequencyfL1.
7. Mark the obtained value on straight line. calculate lock range fL = fL2 -fL1 and capture range fc = fc2 – fc1.

Observation

$\mathbf{f_{C1}}$ =
$\mathbf{f_{C2}}$ =
$\mathbf{f_{L1}}$ =
$\mathbf{f_{L2}}$ =
$\mathbf{f_C = f_{C2} - f_{C1}}$
$\mathbf{f_L = f_{L2} – f_{L1}}$

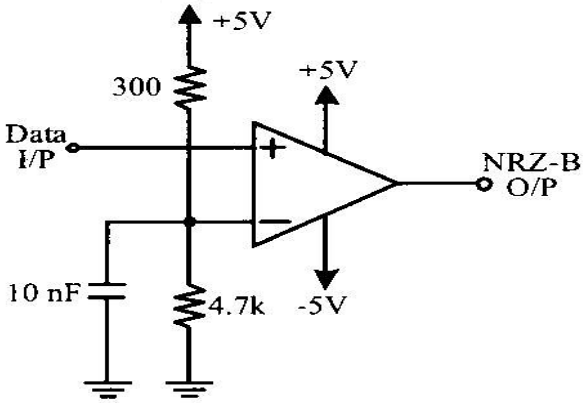
Result: Free running frequency, f_0 = _____Hz, Lock Range f_L = _____Hz
Capture Range f_c= _____Hz

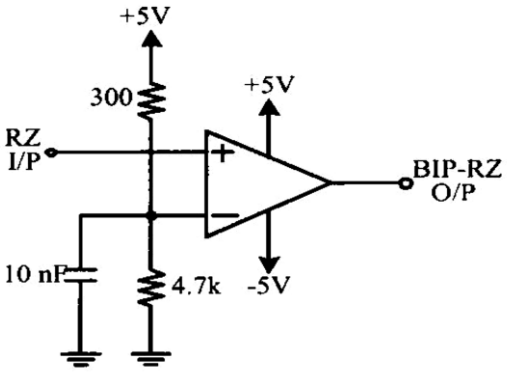
Experiment No: 11
Data formatting and Line Code Generation

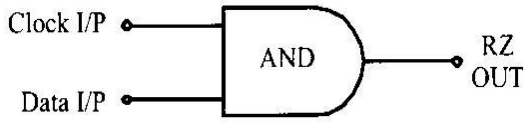
Apparatus Required:

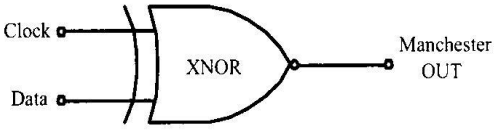
Sl No	Components	Quantity
1	Op-Amp µA741	02
2	Resistor, 300, 4.7KΩ	01
3	AND gate (7408) XNOR Gate(74266)	01
4	Capacitor 10 nF or 0.01uF	01
5	Bread board ,Connecting wire	-
6	CRO (40MHz), Signal generator(1MHz), DC supply(30V)	01

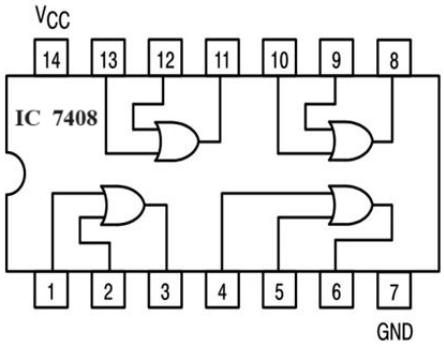
Circuit Diagram

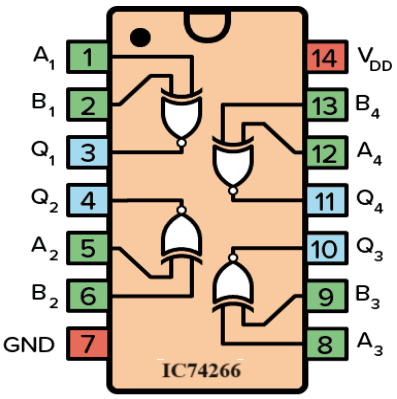


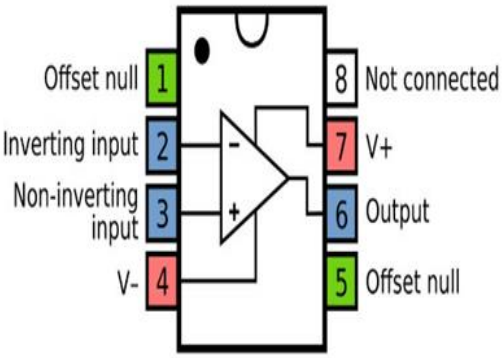




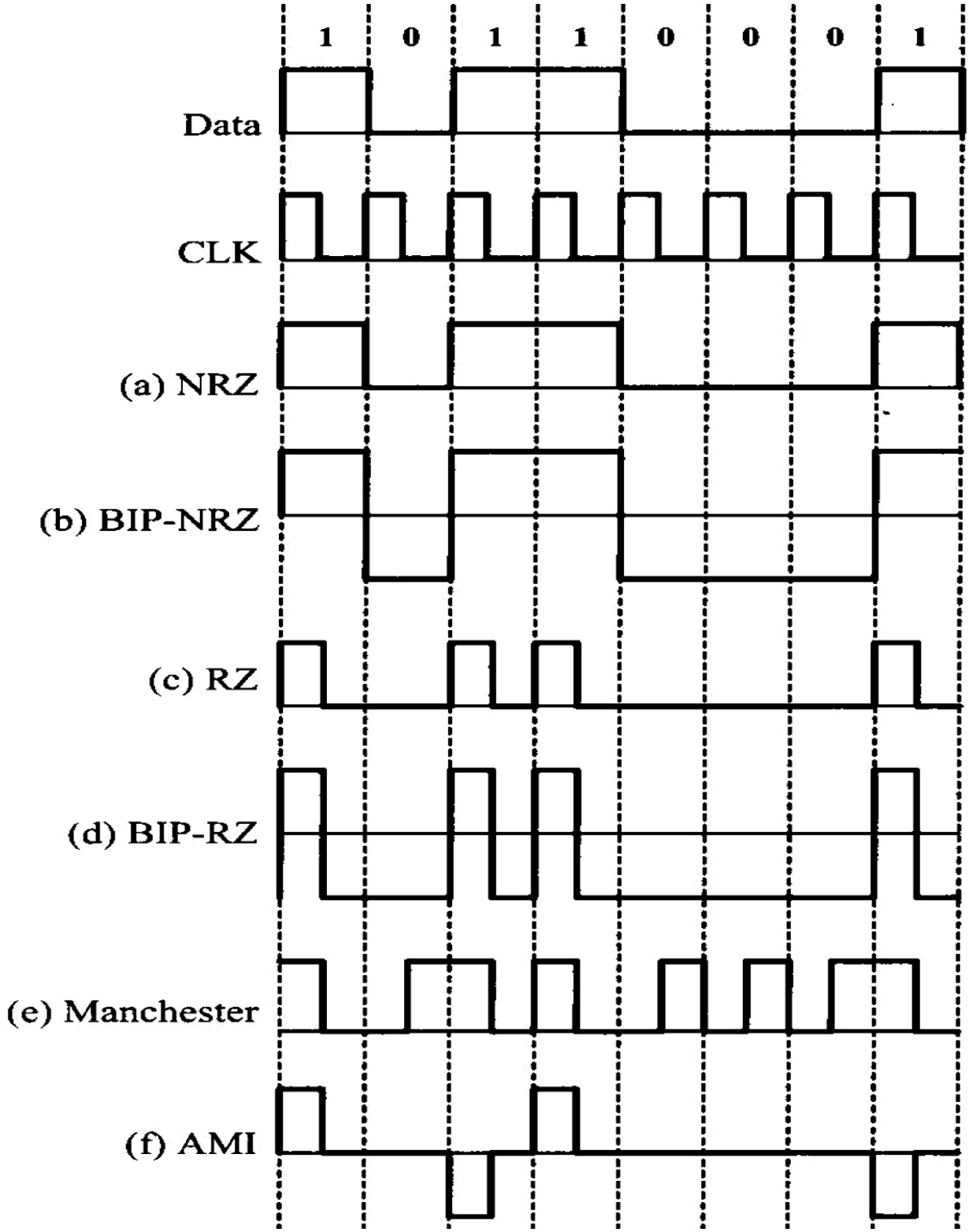








Waveforms


1 0 1 1 0 0 0 1
Data
CLK
(a) NRZ
(b) BIP-NRZ
(c) RZ
(d) BIP-RZ
(e) Manchester
(f) AMI

Theory

Line coding is the process of converting digital data to digital signals. Serial data is to handle by customized protocols like SPI, I2C etc. These protocols are usually based on line codes. The most common types of line encoding are NRZ (Non-Return to Zero), Manchester code, AMI (Alternate Mark Inversion) etc.

Unipolar: presence of pulse represents a 1 and the absence of pulse represents a 0
Polar: a High in data is represented by a positive pulse, while a Low in data is represented by a negative pulse.
Bipolar Signalling: which has three voltage levels namely +, - and 0. Such a signal is called as duo-binary signal.

1‟ and 0 can be represented in various formats in different levels and waveforms. The selection of coding technique depends on system band width, system ability to pass dc level information, error checking facility. Non return to Zero (level): The NRZ(L) waveform simply goes low for one bit time to represent a data „0‟ and high to represent data „1‟.For lengthy data the clock is lost in asynchronous mode. The maximum rate at which NRZ can change is half the data clock, when alternate 0‟s and 1‟s are there. DC Level: A length data will have only a dc level as its waveform, a dc voltage cannot be used in circuits which involve transformers like telephone, AC coupled amplifiers, capacitors, filter etc.

Procedure

1. Before wiring the circuit checks all the components using multi meter and IC tester.
2. As per design set the values and do the connections as shown in circuit diagram.
3. Set the Data using mono clock pulse wherever necessary and observe the output.
4. Set the Data using mono clock pulse and continuous clock pulse wherever necessary and observe the output.
5. Draw the output waveform.

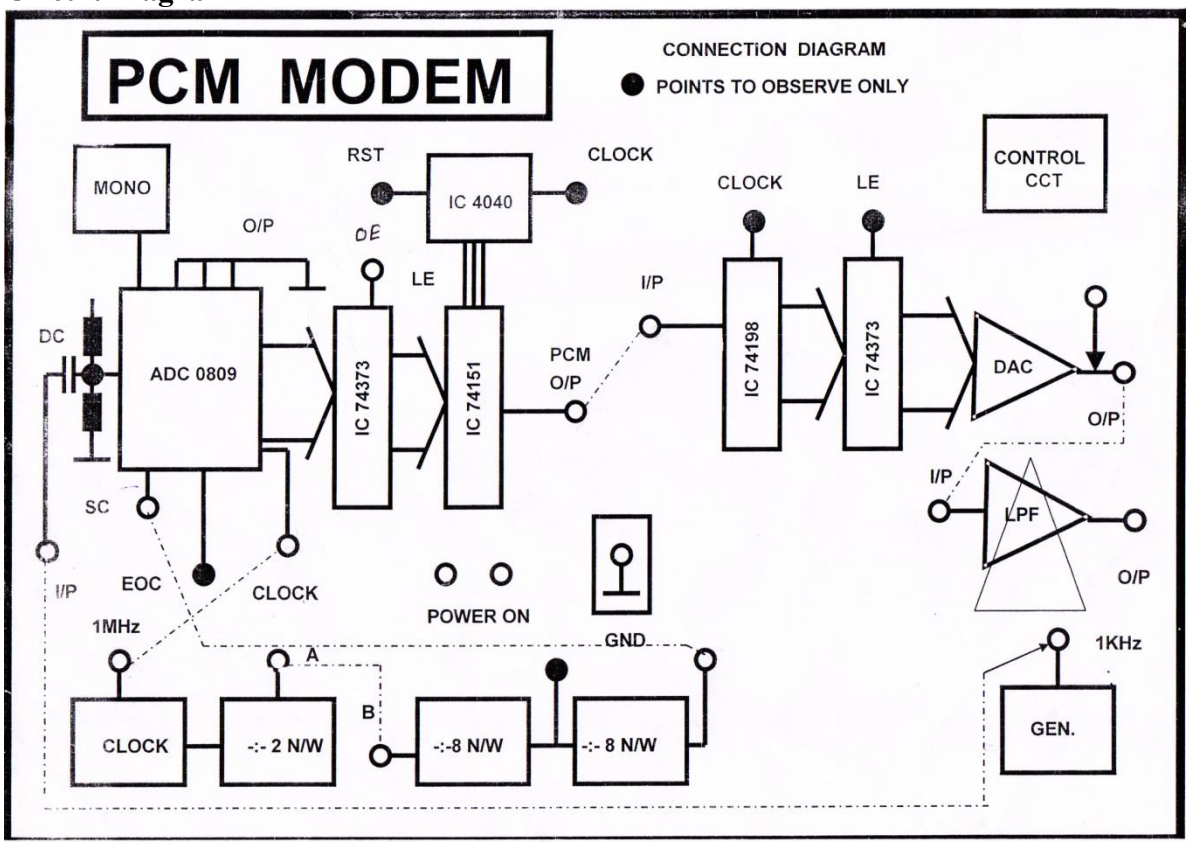
Result: Data formatting and Line Code Generation implemented and verified.

Experiment No: 12
PCM Multiplexer and Demultiplexer

Apparatus Required:

Sl No	Components	Quantity
1	PCM MODEM Kit	01
2	Connecting Wires	--
3	CRO	01

Circuit Diagram



Theory

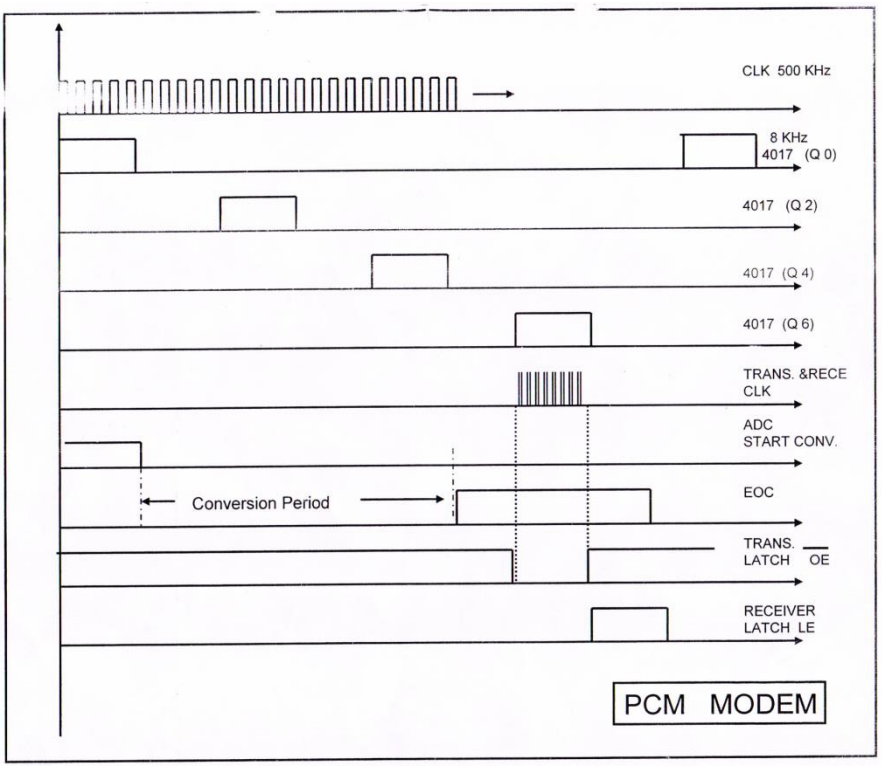
Pulse code Modulation is a digital transmission of samples of analog signal. In PCM Generator we have sampler, Analog to digital converter & parallel to serial data converter & serial transmission. In PCM Receiver there is serial reception of data, this serially Received data is then converted to parallel from & then fed to digital to analog converter. The output of DAC is fed to low pass filter & we get transmitted analog signal. PCM performance as an analog communication system depends primarily on the quantization noise introduced by ADC.

All control signals are derived from basic clock. To derive control signals IC7408, 7432, 4017, are used. After a analog to digital conversion signals fed to Multiplexer IC. Control for IC 74151 is from IC4040. Every time IC4040 counter gives 8 combination & transmits S bit data corresponding to every sample.

In Receiver section we have used shift register IC 7498 in serial in & parallel out form. For synchronization clock at Receiver must be came to clock at transmitter. After serial reception of data output of shift register is latched using 74373 & fed to DAC (R-2R Ladder). Output of DAC is fed to filter.

To observe stable waveform on CRO the sampling frequency must be exactly integer multiple of signal frequency. In our kit signal frequency is obtain by frequency divider & filter circuit.

Waveforms



Procedure

1. Switch on “Power ON” switch red LED should glow.
2. Observe 1 MHz clock O/p signal on panel.
3. Connect this 1 MHz clock to ADC 0809 clock input.
4. Observe point ‘A’ O/p from -: - 2 Network & connect it to point ‘B’ i.e. I/P to -: -Network.
5. Observe O/p of both --: -- 8 network & calculate their frequencies.
6. Connect O/p of 2nd-:-8 network to start conversion I/p of ADC 0809i.e. ‘SC’ point on panel.
7. Observe ‘SC’ point & ‘EOC’ point on panel simultaneously on dual trace oscilloscope & find out conversion period of A/D. Conversion period Time bet been falling edge of ‘SC’ & rising edge of ‘EOC’.
8. Observe ‘EOC’ point & ‘CIE point i.e. O/p enable pin of 74373 latch simultaneously on dual trace scope. Also observe ‘RST’ point &’CLK’ point of IC 4040 along with ‘OE’.
9. Observe ‘PCM OUTPUT’ point & connect it to I/p to receiver. i e. serial I/p to shift register 74198

10. Observe CLK of 74198 along CLK of 4040 dual trace scope.
11. Observe 'LE' i. e. latch enable pin of 74373 latch next to shift register, with 'CLK of 74198.
12. Now you can draw timing diagram of the system.
13. First Connect fixed frequency sine wave to I/p of ADC 0809.
14. Observe DAC O/p. Calculate its frequency & peak to peak amplitude.
15. Observe this with I/p fixed frequency sine wave.
16. Connect DAC O/p to I/p of LPF filter & observe O/p of filter.
17. Externally, use variable frequency source & first keep frequency of sine wave minimum. Observe DAC O/p along with I/p sine wave. Now slowly increase frequency of sine wave to verify Nyquist criteria & to observe aliasing effect.

Result: Design and verified PAM Multiplexer and Demultiplexer

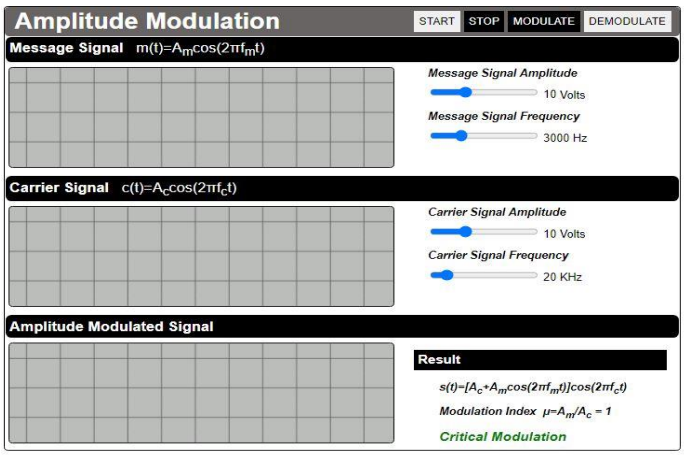
Experiment No: 13
Virtual Lab: To Virtually Simulate and Interpret Amplitude Modulation and De-modulation.

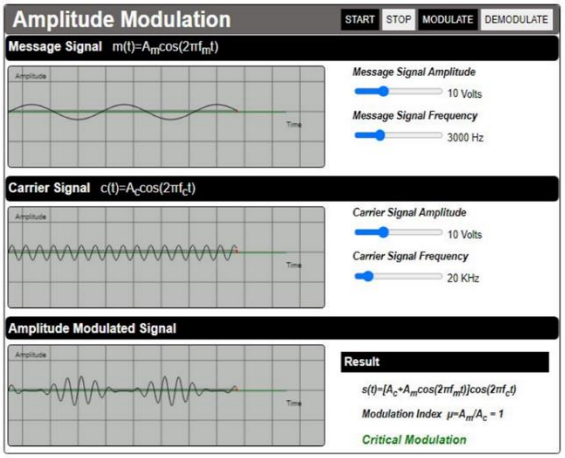
Procedure:

These procedure steps will be followed on the simulator.

1) After reading the theory and attending the pretest, click the "Simulation" tab
2) The interactive simulator will be displayed
3) Set the Message Signal Amplitude and Frequency
4) Set the Carrier Signal Amplitude and Frequency
5) Click on "Modulate" button. This will Show the Modulated Message Signal Simulation Screen.
6) Click on "Start" button. This will start the Simulation.
7) Click on "Demodulate" button. This will Show the Demodulated Message Signal.
8) Click on "Stop" button to view the graph in a Static state
9) In the Results Section you can view the modulation Index and type of modulation (with respect to modulation index)
10) The post-test questions will be displayed, attempt the questions to check the understanding about the experiment.
11) Note the conclusions from the experiment performed.

Simulation





Link: https://kcgcollege.ac.in/Virtual-Lab/Electronics-and-Communication-Engineering/

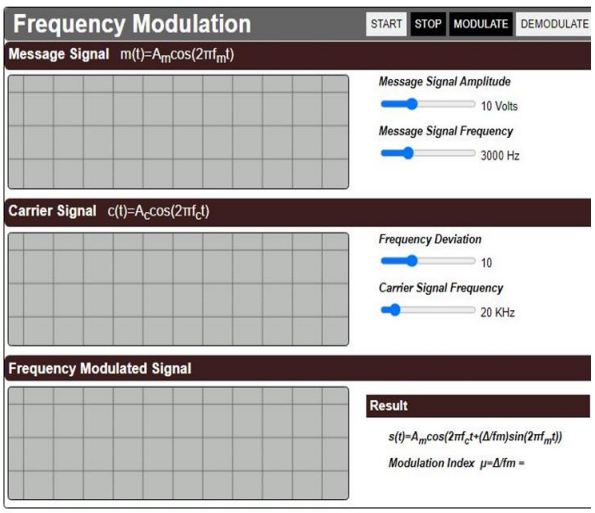
Experiment No: 14
Virtual Lab: To Simulate virtually and Interpret Frequency Modulation and De modulation

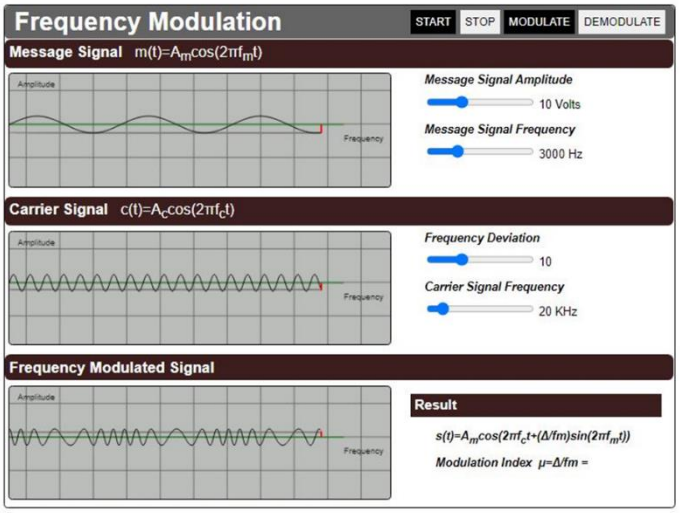
Procedure:

These procedure steps will be followed on the simulator

1) After going through the theory and pretest, click the "Simulation" tab
2) The simulator will display the interactive Simulator
3) Set the Message Signal Amplitude and Frequency
4) Set the Carrier Signal Amplitude and Frequency
5) Click on "Modulate" button. This will Show the Modulated Message Signal.
6) Click on "Start" button. This will Start the Simulation.
7) Click on "Demodulate" button. This will Show the Demodulated Message Signal.
8) Click on "Stop" button to view the graph in a Statis state
9) In the Results Section you can view the modulation Index and type of modulation (with respect to modulation index)
10) The simulator will display the interactive questions, attempt the questions.
11) Note the conclusions from the experiment performed.

Simulation





Link: https://kcgcollege.ac.in/Virtual-Lab/Electronics-and-Communication-Engineering/Exp-2/

Lab Manual 4

Microcontroller Laboratory

(BECL456A)

CO's And PO's Mapping Chart

Sl. No.	Description	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
1	Write a Assembly Language / C Programs in 8051 for solving simple problems that manipulate input data using different instructions	1		2		2								2	
2	Develop testing and experimental procedures on 8051 Microcontroller, analyze their operation under different cases.		1	2		2								2	
3	Develop programs for 8051 Microcontroller to implement real world problems.		1	2		2								2	
4	Develop Microcontroller applications using external hardware interface.		1	2		2								2	

Evaluation:

Non Integrated Lab		
CIE		
Particulars	**Marks**	**Total**
Performance	5	10 * 23 = 230 scaled down to 30
Journal	2	
Viva Voce	3	
Lab IA		
Particulars	**Marks**	**Total**
IA	100	20 (Reduced)
Total (CIE +Lab IA)		50

Mapping of Experiments with CO, PO and PSO

S.No.	Experiment Details	CO	PO	PSO
1	Write an ALP to move a block of n bytes of data from source (20h) to destination (40h) using Internal-RAM.	1	1,3,5	1
2	Write an ALP to move a block of n bytes of data from source (2000h) to destination (2050h) using External RAM.	1	1,3,5	1
3	Write an ALP To exchange the source block starting with address 20h, (Internal RAM) containing N (05) bytes of data with destination block starting with address 40h (Internal RAM).	1	1,3,5	1
4	Write an ALP to exchange the source block starting with address 10h (Internal memory), containing n (06) bytes of data with destination block starting at location 00h (External memory).	1	1,3,5	1
5	Write an ALP to add the byte in the RAM at 34h and 35h, store the result in the register R5 (LSB) and R6 (MSB), using Indirect Addressing Mode.	1	1,3,5	1
6	Write an ALP to subtract the bytes in Internal RAM 34h & 35h store the result in register R5 (LSB) & R6 (MSB).	1	1,3,5	1
7	Write an ALP to multiply two 8-bit numbers stored at 30h and 31h and store16- bit result in 32h and 33h of Internal RAM.	1	1,3,5	1
8	Write an ALP to perform division operation on 8-bit number by 8-bit number.	1	1,3,5	1
9	Write an ALP to separate positive and negative in a given array.	2	2,3,5	1
10	Write an ALP to separate even or odd elements in a given array.	2	2,3,5	1
11	Write an ALP to arrange the numbers in Ascending & Descending order.	2	2,3,5	1
12	Write an ALP to find Largest & Smallest number from a given array starting from 20h & store it in Internal Memory location 40h.	2	2,3,5	1
13	Write an ALP for Decimal UP-Counter.	2	2,3,5	1
14	Write an ALP for Decimal DOWN-Counter.	2	2,3,5	1
15	Write an ALP for Hexadecimal UP-Counter.	2	2,3,5	1
16	Write an ALP for Hexadecimal DOWN-Counter.	2	2,3,5	1
17	Write an 8051 C program to find the sum of first 10 Integer Numbers.	1	1,3,5	1
18	Write an 8051 C program to find Factorial of a given number.	1	1,3,5	1
19	Write an 8051 C program to find the Square of a number (1 to 10) using Look-Up Table.	2	2,3,5	1
20	Write an 8051 C program to count the number of Ones and Zeros in two consecutive memory locations.	2	2,3,5	1
21	Write an 8051 C Program to rotate stepper motor in Clock & Anti-Clockwise direction.	4	2,3,5	1
22	Write an 8051 C program to Generate Sine & Square waveforms using DAC interface.	4	2,3,5	1
23	Virtual Lab: http://vlabs.iitkgp.ac.in/rtes/exp12/index.html	3	2,3,5	1

EXPERIMENT WISE LESSON PLAN

Experiment No.1	
Name	Write an ALP to move a block of n bytes of data from source (20h) to destination (40h) using Internal-RAM.
Objectives	Students will able to transfer a block of N bytes from one location of internal RAM to another.
Experiment No.2	
Name	Write an ALP to move a block of n bytes of data from source (2000h) to destination (2050h) using External RAM.
Objectives	Students will able to transfer a block of N bytes from one location of external RAM to another using MOVX instruction.
Experiment No.3	
Name	Write an ALP To exchange the source block starting with address 20h, (Internal RAM) containing N (05) bytes of data with destination block starting with address 40h (Internal RAM).
Objectives	Students will able to interchange two blocks of data stored in internal RAM.
Experiment No.4	
Name	Write an ALP to exchange the source block starting with address 10h (Internal memory), containing n (06) bytes of data with destination block starting at location 00h (External memory).
Objectives	Students will able to exchange data between internal RAM and external RAM.
Experiment No. 5	
Name	Write an ALP to add the byte in the RAM at 34h and 35h, store the result in the register R5 (LSB) and R6 (MSB), using Indirect Addressing Mode.
Objectives	Students will able to add two numbers stored in memory using indirect addressing mode
Experiment No. 6	
Name	Write an ALP to subtract the bytes in Internal RAM 34h & 35h store the result in register R5 (LSB) & R6 (MSB).
Objectives	Students will able to subtract two numbers stored in internal RAM and store the result.
Experiment No. 7	
Name	Write an ALP to multiply two 8-bit numbers stored at 30h and 31h and store16- bit result in 32h and 33h of Internal RAM.
Objectives	Students will able to multiply two 8-bit numbers and obtain a 16-bit result.

Experiment No. 8	
Name	Write an ALP to perform division operation on 8-bit number by 8-bit number.
Objectives	Students will able to divide one 8-bit number by another and obtain quotient and remainder.
Experiment No.9	
Name	Write an ALP to separate positive and negative in a given array.
Objectives	Students will able to separate positive and negative numbers from a given array.
Experiment No.10	
Name	Write an ALP to separate even or odd elements in a given array.
Objectives	Students will able to separate even and odd numbers from a given array.
Experiment No.11	
Name	Write an ALP to arrange the numbers in Ascending & Descending order.
Objectives	Students will able to arrange a given set of numbers in ascending and descending order.
Experiment No.12	
Name	Write an ALP to find Largest & Smallest number from a given array starting from 20h & store it in Internal Memory location 40h.
Objectives	Students will able to find the largest and smallest numbers in a given array.
Experiment No.13	
Name	Write an ALP for Decimal UP-Counter.
Objectives	Students will able to implement a decimal up-counter using 8051.
Experiment No.14	
Name	Write an ALP for Decimal DOWN-Counter.
Objectives	Students will able to implement a decimal down-counter using 8051.
Experiment No.15	
Name	Write an ALP for Hexadecimal UP-Counter.
Objectives	Students will able to implement a hexadecimal up-counter.
Experiment No.16	
Name	Write an ALP for Hexadecimal DOWN-Counter.
Objectives	Students will able to implement a hexadecimal down-counter.

Experiment No.17	
Name	Write an 8051 C program to find the sum of first 10 Integer Numbers.
Objectives	Students will able to calculate the sum of the first 10 natural numbers using 8051 C.
Experiment No.18	
Name	Write an 8051 C program to find Factorial of a given number.
Objectives	Students will able to compute the factorial of a given number using 8051 C.
Experiment No.19	
Name	Write an 8051 C program to find the Square of a number (1 to 10) using Look-Up Table.
Objectives	Students will able to find the square of a number using a lookup table.
Experiment No.20	
Name	Write an 8051 C program to count the number of Ones and Zeros in two consecutive memory locations.
Objectives	Students will able to count the number of 1s and 0s in given memory locations.
Experiment No.21	
Name	Write an 8051 C Program to rotate stepper motor in Clock & Anti-Clockwise direction.
Objectives	Students will able to rotate a stepper motor in clockwise and anticlockwise directions.
Experiment No.22	
Name	Write an 8051 C program to Generate Sine & Square waveforms using DAC interface.
Objectives	Students will able to generate sine and square waveforms using DAC interfacing.
Experiment No.23	
Name	Virtual Lab: http://vlabs.iitkgp.ac.in/rtes/exp12/index.html
Objectives	Students will able to Interface 4x4 switch matrix with the microcontroller

LIST OF EXPERIMENTS

LAB SAFETY & USAGE INSTRUCTIONS

1. Students must handle the microcontroller board and components carefully to avoid physical damage.
2. Ensure the development board is properly connected to the computer before powering it ON.
3. Upload programs only after verifying the circuit connections and selected device settings.
4. Do not disconnect cables or remove components while the power supply is ON.
5. Shut down the system properly and switch OFF the power supply after completing the experiment.

I. ASSEMBLY LANGUAGE PROGRAMMING

DATA TRANSFER PROGRAM

1. Write an ALP to move block of n bytes of data from source(20h) to destination(40h) using Internal RAM

AIM: To move block of data between two sets of memory location

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions:

i) MOV: Copy instruction
Syntax: MOV destination operand, source operand
Example: MOV A, B; copy the content of B register to A register. A register holds the copy of B register. Source operand is not altered after execution of instruction.

ii) MOVX: This instruction transfers data from external memory location to accumulator
Syntax: movx operand1, operand2
Example: movx a, @dptr

iii) INC: increment instruction
This instruction increases the content of operand by 1
Syntax: inc operand
Example: inc r0

iv) DJNZ: Decrement Jump on No Zero
This instruction decrements the content of operand by 1. After decrementing if operand is not equal to zero then control is transferred to specified location

Assembler Directives:

i) Org: It is used to indicate the beginning address
ii) End: assembler Directive
It indicates the end of program

Program:

```
ORG 0000h
MOV R2, #05h        ; Loads 05h into R2
MOV R1, #20h        ; Loads 20h into R1
MOV R0, #40h        ; Loads 40h into R0
UP: MOV A, @R1      ; Moves data from the internal RAM location pointed by R1 into the A.
MOV @R0, A          ; Stores the contents of A into the internal RAM location pointed by R0.
INC R1              ; Increments R1 to point to the next source memory location.
INC R0              ; Increments R0 to point to the next source memory location.
DJNZ R2, UP         ; Decrements R2 by 1; if R2 ≠ 0, control jumps back to UP to repeat the loop.
END
```

Implementation:

Source			Destination	
Memory Address	Data		Memory Address	Data
D:0x20h	10	→	D:0x40h	10
D:0x21h	20	→	D:0x41h	20
D:0x22h	30	→	D:0x42h	30
D:0x23h	40	→	D:0x43h	40
D:0x24h	50	→	D:0x44h	50

Output:

Before Execution:
D:0x20h:10 D:0x40h:00
D:0x21h:20 D:0x41h:00
D:0x22h:30 D:0x42h:00
D:0x23h:40 D:0x43h:00
D:0x24h:50 D:0x44h:00

After Execution:
D:0x20h:10 D:0x40h:10
D:0x21h:20 D:0x41h:20
D:0x22h:30 D:0x42h:30
D:0x23h:40 D:0x43h:40
D:0x24h:50 D:0x44h:50

Conclusion: Block of data is moved from internal memory location 20h to internal memory location 40h and implemented using Keil.

Output:
Before Execution: **After Execution:**

2. Write an ALP to move block of n bytes of data from source(2000h) to destination(2050h) using External RAM

Aim: To move block of data between two sets of memory location

Objective: To realize efficient method of data interchange over move instruction.

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Program:

```
ORG 0000H
MOV R4, #05H          ; Loads 05H into R4
MOV DPTR, #2050H      ; Loads 2050H into the Data Pointer (DPTR).
MOV R0, DPL           ; Copies the lower byte of DPTR into R0
MOV R1, DPH           ; Copies the higher byte of DPTR into R1
MOV DPTR, #2000H      ; Loads 2000H into the Data Pointer (DPTR).
MOV R2, DPL           ; Copies the lower byte of DPTR into R2
MOV R3, DPH           ; Copies the higher byte of DPTR into R3
UP: MOV DPL, R2       ; Loads R2 into DPL
MOV DPH, R3           ; Loads R3 into DPL
MOVX A, @DPTR         ; Reads a byte from external memory (source) addressed by DPTR into
Accumulator A.
MOV DPL, R0           ; Loads destination low address from R0 into DPL.
MOV DPH, R1           ; Loads destination high address from R1 into DPH.
MOVX @DPTR, A         ; Writes the contents of A into external memory
INC R0
INC R2
DJNZ R4, UP           ; Decrements R4 by 1; if R4 ≠ 0, control jumps back to UP to repeat the loop.
END
```

Output

Before Execution:

0x2000h:10	0x2050h:00
0x2001h:20	0x2051h:00
0x2002h:30	0x2052h:00
0x2003h:40	0x2053h:00
0x2004h:50	0x2054h:00

After Execution:

0x2000h:10	0x2050h:10
0x2001h:20	0x2051h:20
0x2002h:30	0x2052h:30
0x2003h:40	0x2053h:40
0x2004h:50	0x2054h:50

Conclusion

The block of data is transferred from 2000h memory location to 2050h memory location.

Output:

Before Execution:

After Execution:

3. Write an ALP to exchange the source block starting with address 20h containing N(05) bytes of data with destination block starting at location 00h

Aim: To exchange data between two sets of memory location

Objective: To realize efficient method of data interchange over move instruction.

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Program:

```
ORG 0000H
MOV R0, #20H          ; Initialize source memory pointer
MOV R1, #40H          ; Initialize destination memory pointer
MOV R2, #05H          ; Initialize iteration counter

BACK:
MOV A, @R0            ; Get data from source memory pointer
MOV B, @R1
MOV @R0, B
MOV @R1, A
INC R0
INC R1          ; Store data in destination memory pointer
DJNZ R2, BACK         ; Decrement counter and repeat if not zero
END
```

Output:

Before Execution:

D:0x20h:10	D:0x00h:50
D:0x21h:20	D:0x01h:60
D:0x22h:30	D:0x02h:70
D:0x23h:40	D:0x03h:80
D:0x24h:50	D:0x04h:90

After Execution:

D:0x20h:50	D:0x00h:10
D:0x21h:60	D:0x01h:20
D:0x22h:70	D:0x02h:30
D:0x23h:80	D:0x03h:40
D:0x24h:90	D:0x04h:50

Conclusion: Exchange of data between two sets of memory location is implemented.

Output:

Before Execution:

After Execution:

4. Write an ALP to exchange the source block starting with address 10h containing N(06) bytes of data with destination block starting at location 00h

Aim: To exchange data between two sets of memory location

Objective: To realize efficient method of data interchange over move instruction.

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Program:

```
ORG 0000H
MOV R0, #10H              ; Initialize source memory pointer
MOV DPTR, #8000H          ; Initialize destination memory pointer
MOV R1, #06H              ; Initialize iteration counter
BACK:
MOV A, @R0                ; Get data from source memory pointer
MOV  R2, A
MOVX A, @DPTR
XCH A, R2
MOV @DPTR, A
MOV A, R2
MOV @R0, A
INC R0
INC DPL
DJNZ R1, BACK             ; Decrement counter and repeat if not zero
END
```

Output

Before Execution:

D:0x10h:10 X:0x8000h:50
D:0x11h:20 X:0x8001h:60
D:0x12h:30 X:0x8002h:70
D:0x13h:40 X:0x8003h:80
D:0x14h:50 X:0x8004h:90
D:0x15h:60 X:0x8005h:0A

After Execution:

D:0x10h:50 X:0x8000h:10
D:0x11h:60 X:0x8001h:20
D:0x12h:70 X:0x8002h:30
D:0x13h:80 X:0x8003h:40
D:0x14h:90 X:0x8004h:50
D:0x15h:0A X:0x8005h:60

Conclusion: Exchange of data between two sets of memory location is implemented.

Output:

Before Execution:

After Execution:

ARITHMETIC AND LOGICAL OPERATION

5. Write an ALP to add the byte in the RAM at 34h and 35h, store the result in the register R5 and R6 using Indirect Addressing Mode.

Aim: To perform addition of two 8-bit numbers

Objective: To study and understand add and addc instruction and apply it to perform addition of two 8-bit numbers

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions:

i) ADD: Addition Instruction
Syntax: ADD destination operand, source operand
Note: Destination Operand is an A register whereas source operand can be register, Immediate number, Address of internal memory or register indirect addressing mode
Example: ADD A, r0
Operation: A=A + destination operand
Flags affected: C,AC,OV

ii) ADD: Add along with Carry
Syntax: ADDC destination operand, source operand
Example: ADDC A, r0
Operation: A=A+r0+C
Flags affected: C,AC,OV

iii) CLR: Clear Carry Flag

Addressing Modes:

i) Direct addressing mode:
Example: MOV A,40h

ii) Indirect Addressing Mode
Example: MOV A,@R0

Program:

```
ORG 0000h
MOV R0, #34h     ; Loads 34H into R0;
MOV R1, #35h     ; Loads 35H into R1;
MOV A, @R0       ; Moves the data from the internal RAM location pointed by R0 (34H) into the A.
ADD A, @R1       ; Adds the contents of the internal RAM location pointed by R1 (35H) to
Accumulator A.
MOV R5, A        ; Stores the sum from the Accumulator into register R5.
MOV B, #00h      ; Clears register B by loading 00H.
JNC LAST         ; If CY = 0, control jumps to LAST (no carry generated).
INC B
LAST: MOV R6, B ; Copies the contents of register B into R6
END
```

Output

D:0x34h:10
D:0x35h:20
R5:30
R6:00

Conclusion: Two 8-bit numbers are added and result is stored in two different registers which holds Sum and Carry respectively

Output:

6. Write an ALP to Subtract the byte in the RAM at 34h and 35h, store the result in the register R5 and R6

Aim: To perform subtraction of two 8-bit numbers

Objective: To study and understand SUBB instruction and apply it to perform addition of two 8-bit numbers

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions:

i) SUBB: Subtraction Instruction
Syntax: SUBB destination operand, source operand

Note: Destination Operand is an A register whereas source operand can be register, Immediate number, Address of internal memory or register indirect addressing mode

Example: SUBB A, r0
Operation: A=A-destination operand - C
Flags affected: C,AC,OV

Program:

```
ORG 0000H
MOV R0, #34h    ; Loads 34H into R0;
MOV R1, #35h    ; Loads 35H into R1;
MOV A, @R0      ; Moves the data from the internal RAM location pointed by R0 (34H) into the A.
SUBB A, @R1     ; Subtracts the contents of the internal RAM location pointed by R1 (35H) to
Accumulator A.
MOV R5, A       ; Stores the sum from the Accumulator into register R5.
MOV B, #00h     ; Clears register B by loading 00H.
JNC LAST        ; If CY = 0, control jumps to LAST (no carry generated).
INC B
LAST: MOV R6, B ; Copies the contents of register B into R6
END
```

Output

D:0x34h:20
D:0x35h:10
R5:10
R6:00

Conclusion:
Subtraction of two 8-bit numbers are performed and implemented using Keil

Output:

7. Write an ALP to perform multiplication of two 8-bit numbers stored at 30h and 31h and store 16-bit result in 32h and 33h of internal RAM

Aim: To perform multiplication and division of two 8-bit numbers

Objective: To study and understand mul instruction

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions:

i) MUL: Multiplication instruction

Syntax: MUL operand1operand2

Example: MUL AB

Operation: Lower Byte of product in A reg and Higher byte product in B reg

Implementation:

Multiplication Program:

```
ORG 0000H
MOV R0, #30h          ; Load the address of the first operand into R0
MOV R1, #31h          ; Load the address of the second operand into R1
MOV A, @R0            ; Load the first operand into accumulator
MOV B, @R1            ; Load the second operand into B register
MUL AB                ; Multiply the contents of accumulator and B, result in AB
MOV R0, #32h
MOV R1, #33h
MOV @R2, A            ; Store the low byte of the result in address 32h
MOV @R3, B            ; Store the high byte of the result in address 33h
 END
```

Output

D:0x30h:01

D:0x31h:02

R2:02

R3:00

Conclusion: Multiplication of two 8-bit numbers are performed and 16-bit product is stored in two Registers.

Output:

8. Write an ALP to perform division operation on 8-bit number by 8-bit number

Aim: To perform division of two 8-bit numbers

Objective: To study and understand div instruction

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions:

i) DIV: Division instruction

Syntax: DIV operand1operand2

Example: DIV AB

Operation: A is Dividend and B holds Divisor. After Division operation A holds Quotient and B holds Remainder

Program:

```
ORG 0000h
MOV R0, #30h          ; Load the dividend address into R0
MOV R1, #31h          ; Load the divisor address into R1
MOV A, @R0            ; Load the dividend into accumulator
MOV B, @R1            ; Load the divisor into B register
DIV AB                ; Divide the contents of accumulator (dividend) by B (divisor),
                      quotient in A, remainder in B
MOV @R2, A            ; Store the quotient in address 32h
MOV @R3, B            ; Store the remainder in address 33h
END
```

Output

D:0x30:40

D:0x31:05

D:0x32:08

D:0x32:00

Conclusion: Division of two 8-bit numbers are performed and implemented using Keil

Output:

9. Write an ALP to Separate Positive and Negative in a given array

Aim: To Separate Positive and Negative Numbers in a given array

Objective: To study and understand positive and negative numbers

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions:

i) JNC:
Syntax: JNC relative address
Example: JNC last
Description: This instruction checks for A=0, if A is not 0 then control is transferred to specified location.

ii) JMP:
Syntax: JMP relative address
Example: JMP Positive
Description: It transfers program control to a specified memory address by loading the Program Counter with that address.

iii) RLC:
Syntax: RLC relative address
Description: Rotate Left through Carry. It performs 9-bit rotation (8 bits + carry)
A = 45H → 0100 0101
CY = 1
After RLC A:
A = 8BH → 1000 1011
CY = 0

Program:

```
ORG 0000H
MOV R0, #20H          ; Loads R0 with address 20H, to the start of the internal RAM array.
MOV R5, #08H          ; Loads R5 with 08H, representing the no of elements in the array.
MOV DPTR, #8000H      ; Loads DPTR with address 8000H, to store positive numbers.
MOV R1, DPL
MOV R2, DPH
MOV DPTR, #8050H      ; Loads DPTR with address 8050H, to store negative numbers.
MOV R3, DPL
MOV R4, DPH
UP:
   MOV A, @R0         ; Moves the data pointed to by R0 into the Accumulator.
   MOV B, A           ; Copies the accumulator content into register B
   RLC A              ; Rotates accumulator left through carry to check the sign bit.
   JNC POSITIVE       ; Jumps to POSITIVE if carry is not set (number is positive).
   JMP NEGATIVE       ; Jumps to NEGATIVE if carry is set (number is negative).

POSITIVE:
   MOV DPL, R1
   MOV DPH, R2
   MOVX @DPTR, A      ; Stores the positive number into external memory
   INC R1             ; Increments positive memory pointer.
```

```
   JMP LAST          ; Jumps to LAST to continue loop execution.
NEGATIVE:
   MOV DPL, R3
   MOV DPH, R4
   MOVX @DPTR, A  ; Stores the negative number into external memory.
   INC R3            ; Increments negative memory pointer.
LAST:
   INC R0            ; Points R0 to the next array element.
   DJNZ R5, UP       ; Decrements R5 and jumps to UP if R5 is not zero.
END
```

Output:

D: 0X20: 02h, 69h, 03h, 96h, 79h, 78h, 05h, 06h

X: 0X8000: 02h, 69h, 03h, 79h, 78h, 05h, 06h

X: 0X8050: 96h

Conclusion: Positive and Negative Numbers are separated in a given array of Numbers is implemented using Keil.

Output:

10. Write an ALP to Separate Even or Odd elements in a given array

Aim: To Separate Even or Odd elements in a given array

Objective: To study and understand even and odd numbers

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions:

i) RRC
Syntax: RRC A
Description: Rotate Right through Carry; performs 9-bit rotation (8 accumulator bits + carry).
Example:
A=45H→01000101
CY = 1
After RRC A:
A = A2H → 1010 0010
CY = 1

Program:

```
ORG 0000H
MOV R0, #20H          ; Loads R0 with address 20H, to the start of the internal RAM array.
MOV R5, #08H          ; Loads R5 with 08H, representing the no of elements in the array.
MOV DPTR, #8000H      ; Loads DPTR with address 8000H, to store even numbers.
MOV R1, DPL
MOV R2, DPH
MOV DPTR, #8050H      ; Loads DPTR with address 8050H, to store odd numbers.
MOV R3, DPL
MOV R4, DPH

UP:
   MOV A, @R0         ; Moves the data pointed to by R0 into the Accumulator.
   MOV B, A           ; Copies the accumulator content into register B
   RRC A              ; Rotates accumulator right through carry to check the carry bit.
   JNC EVEN           ; Jumps to EVEN if carry is not set (number is positive).
   JMP ODD            ; Jumps to ODD if carry is set (number is negative).

EVEN:
   MOV DPL, R1
   MOV DPH, R2
   MOVX @DPTR, A      ; Stores the even number into external memory
   INC R1             ; Increments even memory pointer.
   JMP LAST           ; Jumps to LAST to continue loop execution.

ODD:
   MOV DPL, R3
   MOV DPH, R4
   MOVX @DPTR, A      ; Stores the odd number into external memory.
   INC R3             ; Increments odd memory pointer.
LAST:
   INC R0             ; Points R0 to the next array element.
```

```
    DJNZ R5, UP                ; Decrements R5 and jumps to UP if R5 is not zero.
END
```

Output:

D: 0X20: 01h, 02h, 03h, 04h, 05h, 06h, 07h, 08h

X: 0X8000: 02h, 04h, 06h, 08h

X: 0X8050: 01h, 03h, 05h, 07h

Conclusion: Even and Odd Numbers are separated in a given array of Numbers is implemented using Keil.

Output:

11. Write an ALP to arrange the numbers in ascending and descending orders

Aim: To sort given numbers in ascending order

Objective: To understand bubble sort technique and sort given array in ascending and descending order using bubble sort technique.

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions:

i) CJNE**:** The CJNE instruction compares the first two operands and branches to the specified destination if their values are not equal. If the values are the same, execution continues with the next instruction.
Syntax: CJNE operand1, operand2, reladdr
Example: CJNE A,B,next

The Carry bit (C) is set if operand1 is less than operand2, otherwise it is cleared

ii) DEC**:**
Syntax: DEC register.
Description: This instruction decrements the content of specified register by 1

iii) JNC:
Syntax: JNC relative address
Example: JNC last
Description: This instruction checks for Carry flag=0, if carry flag is 0 then control is transferred to specified location.

iv) JC:
Syntax: JC relative address
Example: JC last 1
Description: This instruction checks for Carry flag=1, if carry flag is 1 then control is transferred to specified location.

Implementation:

Bubble Sort Algorithm

Note: If number of elements to be sorted are 'n' then it consists of 'n-1' number of passes(iterations) and comparisons
steps:
1. starting with first element compare first element with next element of array
2. if current element is greater than the next element of the array swap them
3. if the current element is less than the next element of the array, move to next element

let i) n=06
ii) array= 5,1,6,2,4,3

Let's consider an array with values {5, 1, 6, 2, 4, 3}
Below, we have a pictorial representation of how bubble sort will sort the given array.

Ascending Order Program:

```
ORG 0000h
MOV R1, #06h            ; Loads 06H into R1;
MOV R2, #05h            ; Loads 05H into R2;

LOOP1: MOV R3, #05h ; Loads 05H into R3
MOV DPTR, #9000h    ; Loads 9000H into DPTR

LOOP: MOVX A, @DPTR ; Reads a byte from DPTR into A.
MOV B, A            ; Copies the value from A into register B for comparison.
INC DPTR
MOVX A, @DPTR       ; Reads the next byte from DPTR into A.
CJNE A, B, NEXT     ; Compares A and B; if not equal, jumps to NEXT.

NEXT: JNC LAST      ; Jump if No Carry; if A ≥ B, control jumps to LAST
XCH A, B            ; Exchanges the contents of A and register B.
MOVX @DPTR, A       ; Stores the value in A into DPTR.
DEC DPL
MOV A, B            ; Moves the contents of register B into A.
MOVX @DPTR, A       ; Stores the value in A into external memory
INC DPTR

LAST: DJNZ R3, LOOP ; Decrements R3; if not zero, jumps back to LOOP for next comparison.
DJNZ R2, LOOP1      ; Decrements R2; if not zero, jumps back to LOOP1 to repeat passes.
END
```

Output:

Before Execution:
X:0x9000h:05
X:0x9001h:01
X:0x9002h:06
X:0x9003h:02
X:0x9004h:04
X:0x9005h:03

After Execution:
X:0x9000h:01
X:0x9001h:02
X:0x9002h:03
X:0x9003h:04
X:0x9004h:05
X:0x9004h:06

Output:

Before Execution:

After Execution:

Descending Order Program:

```
ORG 0000h
MOV R1, #06h            ; Loads 06H into R1;
MOV R2, #05h            ; Loads 05H into R2;

LOOP1: MOV R3, #05h ; Loads 05H into R3
MOV DPTR, #9000h    ; Loads 9000H into DPTR

LOOP: MOVX A, @DPTR ; Reads a byte from DPTR into A.
MOV B, A            ; Copies the value from A into register B for comparison.
INC DPTR
MOVX A, @DPTR       ; Reads the next byte from DPTR into A.
CJNE A, B, NEXT     ; Compares A and B; if not equal, jumps to NEXT.

NEXT: JC LAST       ; Jump if Carry; if A< B, control jumps to LAST
XCH A, B            ; Exchanges the contents of A and register B.
MOVX @DPTR, A       ; Stores the value in A into DPTR.
DEC DPL
MOV A, B            ; Moves the contents of register B into A.
MOVX @DPTR, A       ; Stores the value in A into external memory
INC DPTR

LAST: DJNZ R3, LOOP ; Decrements R3; if not zero, jumps back to LOOP for next comparison.
DJNZ R2, LOOP1      ; Decrements R2; if not zero, jumps back to LOOP1 to repeat passes.
END
```

Output

Before Execution:

X:0x9000h:05
X:0x9001h:01
X:0x9002h:06
X:0x9003h:02
X:0x9004h:04
X:0x9005h:03

After Execution:

X:0x9000h:06
X:0x9001h:05
X:0x9002h:04
X:0x9003h:03
X:0x9004h:02
X:0x9004h:01

Conclusion:

The given array is sorted in ascending and descending order using bubble sort method and implemented using Keil

Output:

Before Execution:

After Execution:

12. Write an ALP to find largest and smallest number from a given array starting from 20h and store it in Internal memory location 40h

Aim: To find largest element in a given array

Objective: To study and understand how to find largest and smallest element in an array

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions:

i) JC

Syntax: JC relative address

Description: This instruction checks for Carry flag=1, if carry flag is 1 then control is transferred to specified location.

Implementation:

Steps:

1. If n is size of array
2. Then n-1 number of comparisons are required
3. Let Array= 10,80,60,20,30
4. Let B register to store a largest number
5. Copy Current element (1^{st}) and next element(2^{nd}) in B and A register respectively
6. Compare 1^{st} element(B) and 2^{nd} element(A)
7. If 2^{nd} element(A) is larger than 1^{st}element(B),then move larger element to B register
8. Else move to next element.

Largest number Program:

```
ORG 0000h
MOV R2, #04h      ; Loads 04h into R2
MOV R0, #20h      ; Loads 20h into R0
MOV A, R0         ; Copies the contents of R0 (20H) into the Accumulator A.
MOV B, A          ; Copies the value in A into register B

LOOP:
INC R0
MOV A, @R0        ; Moves the data from R0 into Accumulator A.
CJNE A, B, NEXT   ; Compares A with B; if not equal, control jumps to NEXT.

NEXT: JC LAST     ; if A < B, control jumps to LAST
MOV B, A          ; Copies A into B

LAST: DJNZ R2, LOOP  ; Decrements R2 by 1; if R2 ≠ 0, jumps back to LOOP.
MOV R1, #40h         ; Loads 40H into R1
MOV @R1, B           ; Stores the final value in B into 40H.
END
```

Output

Before Execution:
D: 0X20h: 10, 20, 30, 40, 50
D: 0X40h: 00, 00, 00, 00, 00

After Execution:
D: 0X20h: 0Ah, 14h, 1Eh, 28h, 32h
D: 0X40h: 32h, 00, 00, 00, 00

Output:

Before Execution:

After Execution:

Smallest Number Program:

```
ORG 0000h
MOV R2, #04h      ; Loads 04h into R2
MOV R0, #20h      ; Loads 20h into R0
MOV A, R0         ; Copies the contents of R0 (20H) into the Accumulator A.
MOV B, A          ; Copies the value in A into register B

LOOP:
INC R0
MOV A, @R0        ; Moves the data from R0 into Accumulator A.
CJNE A, B, NEXT   ; Compares A with B; if not equal, control jumps to NEXT.

NEXT: JNC LAST    ; if A > B, control jumps to LAST
MOV B, A          ; Copies A into B

LAST: DJNZ R2, LOOP  ; Decrements R2 by 1; if R2 ≠ 0, jumps back to LOOP.
MOV R1, #40h         ; Loads 40H into R1
MOV @R1, B           ; Stores the final value in B into 40H.
END
```

Output

Before Execution:
D: 0X20h: 10, 20, 30, 40, 50
D: 0X40h: 00, 00, 00, 00, 00

After Execution:
D: 0X20h: 0Ah, 14h, 1Eh, 28h, 32h
D: 0X40h: 0Ah, 00, 00, 00, 00

Conclusion:
Largest and smallest element is found in a given array

Output:

Before Execution:

After Execution:

COUNTER OPERATION PROGRAMS

13. Write an ALP Decimal up counter

Aim: To perform Decimal up counter operation

Objective: To study and understand Counter operation in Up mode

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions: Program can be implemented by using 'inc' and 'dec' instruction.

Theory: Counter is a device that is used to store the number of times the event has occurred

Counter operation modes: i) Up mode
ii) Down Mode
iii) Up-Down Mode

Up counter Program:

```
ORG 0000h
MOV A, #00h          ; Loads 00h into Register A.
MOV B, #0Ah          ; Loads 0Ah into Register B.
MOV R0, #40h         ; R0 acts as a pointer to internal RAM
MOV R1, #0Ah         ; Loads 0Ah into R1.
UP: CJNE A, B, NEXT  ; Compares the contents of A with register B.
NEXT: JNC LAST       ; If CY = 0, control jumps to label LAST; otherwise execution continues.
MOV @R0, A           ; Load the contents of A to the internal RAM location pointed by R0
INC R0
INC A
DJNZ R1, UP          ; Decrements R1 by 1; if R1 ≠ 0, program jumps to label UP to repeat the
loop.
LAST: NOP            ; Performs no action
END
```

Output:

Before Execution:
D:0x40h:00
D:0x41h:00
D:0x42h:00
D:0x43h:00
D:0x44h:00
D:0x45h:00
D:0x46h:00
D:0x47h:00
D:0x48h:00
D:0x49h:00

After Execution:
D:0x40h:00
D:0x41h:01
D:0x42h:02
D:0x43h:03
D:0x44h:04
D:0x45h:05
D:0x46h:06
D:0x47h:07
D:0x48h:08
D:0x49h:09

Conclusion: Decimal up counter operation is performed for counting numbers in up mode from 00 to 99 (0 to 9).

Output:

Before Execution: **After Execution:**

14. Write an ALP Decimal Down counter

Aim: To perform Decimal Down counter operation

Objective: To study and understand Counter operation in Down mode

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Instructions: Program can be implemented by using 'inc' and 'dec' instruction.

Down Counter Program:

```
ORG 0000h
MOV A, #09h          ; Loads 09h into Register A.
MOV B, #00h          ; Loads 00h into Register B.
MOV R0, #40h         ; R0 acts as a pointer to internal RAM
MOV R1, #0Ah         ; Loads 0Ah into R1.
UP: CJNE A, B, NEXT  ; Compares the contents of A with register B.
NEXT: JC LAST        ; If CY = 1, control jumps to label LAST; otherwise execution continues.
MOV @R0, A           ; Load the contents of A to the internal RAM location pointed by R0
INC R0
DEC A
DJNZ R1, UP          ; Decrements R1 by 1; if R1 ≠ 0, program jumps to label UP to repeat the
loop.
LAST: NOP            ; Performs no action
END
```

Output:

Before Execution:

D:0x40h:00
D:0x41h:00
D:0x42h:00
D:0x43h:00
D:0x44h:00
D:0x45h:00
D:0x46h:00
D:0x47h:00
D:0x48h:00
D:0x49h:00

After Execution:

D:0x40h:09
D:0x41h:08
D:0x42h:07
D:0x43h:06
D:0x44h:05
D:0x45h:04
D:0x46h:03
D:0x47h:02
D:0x48h:01
D:0x49h:00

Conclusion: Down counter program is implemented using Keil micro vision 2

Output:

Before Execution:

After Execution:

15. Write an ALP HexaDecimal Up Counter

Aim: To perform HexaDecimal Up counter operation

Objective: To study and understand Counter operation in Up mode

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Program:

```
ORG 0000h
MOV A, #00h            ; Loads 00h into Register A.
MOV B, #10h            ; Loads 10h into Register B.
MOV R0, #40h           ; R0 acts as a pointer to internal RAM
MOV R1, #10h           ; Loads 10h into R1.
UP: CJNE A, B, NEXT    ; Compares the contents of A with register B.
NEXT: JNC LAST         ; If CY = 0, control jumps to label LAST; otherwise execution continues.
MOV @R0, A             ; Load the contents of A to the internal RAM location pointed by R0
INC R0
IN
C A
DJNZ R1, UP            ; Decrements R1 by 1; if R1 ≠ 0, program jumps to label UP to repeat the
loop.
LAST: NOP              ; Performs no action
END
```

Output:

D:0x40h:00
D:0x41h:01
D:0x42h:02
D:0x43h:03
D:0x44h:04
D:0x45h:05
D:0x46h:06
D:0x47h:07
D:0x48h:08
D:0x49h:09
D:0x4Ah:0A
D:0x4Bh:0B
D:0x4Ch:0C
D:0x4Dh:0D
D:0x4Eh:0E
D:0x4Fh:0F

Conclusion: Hexadecimal Up Counter Operation is implemented using Keil software.

Output:

16. Write an ALP Hexa Decimal Down Counter

Aim: To perform Hexa Decimal Down counter operation

Objective: To study and understand Counter operation in Down mode

Software Tool Used: Keil micro vision 2

Coding Language: 8051 Assembly Language

Program:

```
ORG 0000h
MOV A, #0Fh          ; Loads 0Fh into Register A.
MOV B, #00h          ; Loads 00h into Register B.
MOV R0, #40h         ; R0 acts as a pointer to internal RAM
MOV R1, #10h         ; Loads 10h into R1.
UP: CJNE A, B, NEXT  ; Compares the contents of A with register B.
NEXT: JC LAST        ; If CY = 1, control jumps to label LAST; otherwise execution continues.
MOV @R0, A           ; Load the contents of A to the internal RAM location pointed by R0
INC R0
DEC A
DJNZ R1, UP          ; Decrements R1 by 1; if R1 ≠ 0, program jumps to label UP to repeat the
loop.
LAST: NOP            ; Performs no action
END
```

Output

D:0x40h:0F
D:0x41h:0E
D:0x42h:0D
D:0x43h:0C
D:0x44h:0B
D:0x45h:0A
D:0x46h:09
D:0x47h:08
D:0x48h:07
D:0x49h:06
D:0x4Ah:05
D:0x4Bh:04
D:0x4Ch:03
D:0x4Dh:02
D:0x4Eh:01
D:0x4Fh:00

Conclusion: Hexadecimal Down Counter Operation is implemented using Keil software.

Output:

II. C-PROGRAMMING

1. **Write an 8051 C Program to find sum of first 10 integer Numbers**

Aim: To find sum of first 10 integer Numbers

Objective: To study and understand addition of first 10 integers numbers

Software Tool Used: Keil micro vision 2

Coding Language: C-Programming Language

Program:

```
#include<reg51.h>

void main() {
   unsigned char i;
   unsigned int sum = 0;

                                        // Loop to add numbers from 1 to 10
   for (i = 1; i <= 10; i++)
{
      sum += i; // Adding each number to the sum
}

                                         // Displaying the sum
                                            // You need to write your own code to display the sum,
                                            // depending on your hardware setup (LEDs, LCD, Serial
                                               monitor, etc.)
}
```

Output

Sum=10

Conclusion: Sum of first 10 integer numbers is implemented using C-Programming Language

Output:

2. **Write 8051 C Program to find factorial of a given number**

Aim: To find sum of factorial of given number

Objective: To study and understand factorial of given number

Software Tool Used: Keil micro vision 2

Coding Language: C-Programming Language

Program:

```
#include<reg51.h>

unsigned int factorial (unsigned int num)
{
   unsigned int fact = 1;
   unsigned int i;

                                                  // Calculate factorial
   for (i = 1; i <= num; i++)
{
       fact *= i;
 }

   return fact;
}

void main() {
   unsigned int number = 5;      // Change this number to find factorial of a different number
   unsigned int result;

                                  // Calculate factorial of the given number
   result = factorial(number);
}
```

Output:
Number=5
Result=120

Conclusion: factorial of number is implemented using C-Programming Language

Output:

3. **Write an 8051 C Program to find the Square of a number(1 to 10) using Look-Up Table**

Aim: To find the Square of a number(1 to 10) using Look-Up Table

Objective: To study and understand Square of a number(1 to 10) using Look-Up Table

Software Tool Used: Keil micro vision 2

Coding Language: C-Programming Language

Program:

```
#include <reg51.h>

                                        // Lookup table for squares of numbers from 1 to 10
unsigned char squares[] = {1, 4, 9, 16, 25, 36, 49, 64, 81, 100};

void main()
{
   unsigned char number = 5;          // Change this number to find square of a different
                                         number
   unsigned char square;

                                        // Checking if the number is within range (1 to 10)
   if (number >= 1 && number <= 10)
{
                                        // Getting the square from the lookup table
      square = squares[number - 1];

                                  // Displaying the square
                                     // You need to write your own code to display the square,
                                     // depending on your hardware setup (LEDs, LCD, Serial monitor, etc.)
   }
else
{
                                       // Number out of range
                                       // You can handle this case as per your requirement
   }
}
```

Output:
Number=5
Square=25

Conclusion: Square of a number(1 to 10) using Look-Up Table is implemented using C-Programming Language

Output:

4. **Write an 8051 C Program to count the number of Ones and Zeros in two Consecutive memory locations**

Aim: To find the Square of a number (1 to 10) using Look-Up Table

Objective: To study and understand Square of a number (1 to 10) using Look-Up Table

Software Tool Used: Keil micro vision 2

Coding Language: C-Programming Language

Program:

```
#include <reg51.h>
unsigned char countOnesZeros(unsigned char memLocation1, unsigned char memLocation2)
{
   unsigned char data1 = *((unsigned char*)memLocation1);     // Read data from
                                                              first memory location
   unsigned char data2 = *((unsigned char*)memLocation2);     // Read data from
                                                                    second memory location

   unsigned char countOnes = 0;
   unsigned char countZeros = 0;

                                                  // Counting ones and zeros in the first byte
   for (unsigned char i = 0; i < 8; i++)
{
      if (data1 & (1 << i))
{
         countOnes++;
}
else
{
         countZeros++;
}
}

                                                // Counting ones and zeros in the second byte
   for (unsigned char i = 0; i < 8; i++)
{
      if (data2 & (1 << i))
{
         countOnes++;
}
else {
         countZeros++;
      }
   }

                                                // Return the count of ones and zeros
   return ((countOnes << 4) | countZeros);      // Store count of ones in higher nibble
                                                      and count of zeros in lower nibble
}

void main() {
   unsigned char memLocation1 = 0x30; // Change this to the address of first memory
                                         location
   unsigned char memLocation2 = 0x31; // Change this to the address of second
```

memory location

```
    unsigned char result = countOnesZeros(memLocation1, memLocation2);
}
```

Output:

Ones=4
Zeros=4

Conclusion: Number of ones and zeros are counted and stored in two different memory location

Output:

HARDWARE INTERFACING PROGRAMS

1. **Write an 8051 C Program to rotate stepper motor in Clock and Anticlockwise direction**

Aim: To rotate stepper motor in Clock and Anticlockwise direction

Objective: To study and understand Stepper Motor Rotation

Software Tool Used: Keil micro vision 2

Coding Language: C-Programming Language

Program: Stepper Motor in Clockwise Direction

```
#include<reg52.h>
#include<absacc.h>
void delay(void);
main()
{
unsigned char a=0x01;
XBYTE [0X0403]=0X80;
while(1)
{
XBYTE[0X0400]=a;
delay();
a=a<<1;
if(a==0X00)
{
a=0X01;
}
else
{
}
}
}
void delay(void)
{
int i,j;
for(i=0;i<100;i++)
{
for(j=0;j<1000;j++)
{
}
```

```
}
}
```

Program: ANTICLOCKWISE DIRECTION

```
#include<reg52.h>
#include<absacc.h>
void delay(void);
main()
{
unsigned char a=0x08;
XBYTE [0X0403]=0X80;
while(1)
{
XBYTE[0X0400]=a;
delay();
a=a>>1;
if(a==0X00)
{
a=0X08;
}
else
{
}
}
}
void delay(void)
{
int i,j;
for(i=0;i<100;i++)
{
for(j=0;j<1000;j++)
{
}
}
}
```

Conclusion: Stepper motor rotation in Clock wise and Anti Clock wise Direction

2. **Write an 8051 C Program to Generate Sine and Square wave using DAC interface**

Aim: Write a C program to generate a sine waveform and Square Waveform using DAC interface

Objective: To study and understand Sine and Square wave generation

Software Tool Used: Keil micro vision 2

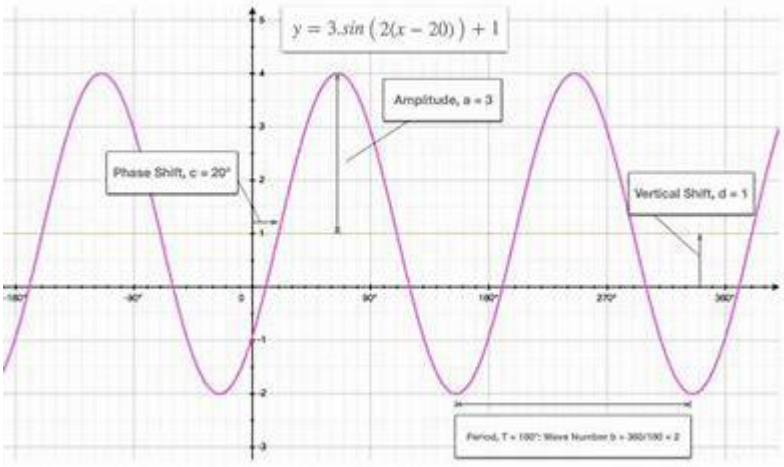
Coding Language: C-Programming Language

Program: Generation of Sine Wave

```
#include <reg52.h>
void delay (void);
main()
{
TMOD=0x81;
while(1)
{
P0=0x80;
delay();
P0=0xC0;
delay();
P0=0xEE;
delay();
P0=0xFF;
delay();
P0=0xEE;
delay();
P0=0xC0;
delay();
P0=0x80;
delay();
P0=0x40;
delay();
P0=0x11;
delay();
P0=0x00;
delay();
P0=0x11;
delay();
P0=0x40;
delay();
```

```
}
}
void delay(void)
{
TH0=0xf9;
TL0=0xf6;
TR0=1;
while(TF0==1)
{
}
TR0=0;
TF0=0;
}
```

Output:

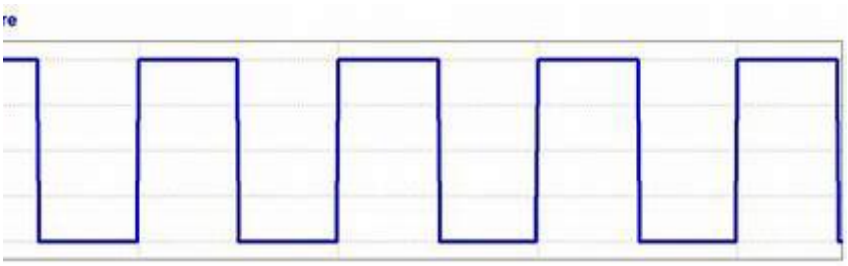


Output:

Program: Generation of Square Wave

```
#include <reg51.h>
#define FREQ_HZ 1000 // Frequency in Hz
#define TIMER_CLOCK_FREQ 12000000     // Timer clock frequency in Hz (for example,
                                         assuming 12 MHz crystal oscillator)

void main() {
   unsigned int timer_reload_value;

                                         // Calculate reload value for timer
                                         // Reload value = Timer clock frequency /
                                                          (2 * desired frequency)
   timer_reload_value = TIMER_CLOCK_FREQ / (2 * FREQ_HZ);

                                         // Configure Timer 1 in Mode 2 (8-bit
                                            auto-reload mode)
   TMOD |= 0x20;

                                                          // Load initial value into timer
   TH1 = (unsigned char)(timer_reload_value >> 8);
   TL1 = (unsigned char)timer_reload_value;

                                                          // Start Timer 1
   TR1 = 1;

   while (1) {
      // Your main application code here
      // For continuous square wave generation, you might not need additional code here
   }
}
```

Output:



Conclusion: Generation of Sine wave and Square Wave is implemented using 8051 C-Programming Language

Output:

Virtual Lab

URL: http://vlabs.iitkgp.ac.in/rtes/exp12/index.html

Objective:
This experiment deals with the interfacing of a switch matrix and then generating patterns that can be viewed in the LED matrices.

Tools Required:
Hardware:

- LED Matrix
- ATMEGA16 microcontroller
- Switch Matix
- Crystal Oscilator
- Capacitors
- Resistors

Software:

- Proteus
- AVR Studio

Theory
ABOUT MICROCONTROLLERS FROM THE AVR FAMILY:

The AVR architecture is based upon modified Harvard architecture where program and data are stored in separate physical memory systems. The AVR family can be briefly classified as

- Tiny AVR
- Mega AVR
- XMEGA AVR
- Application specific AVR
- FPSLIC
- 32-bit AVR

Atmel's AVRs have a two stage, single level pipeline design. This means the next machine instruction is fetched as the current one is executing. Most instructions take just one or two clock cycles, making AVRs relatively fast among the eight-bit microcontrollers. The AVR processors were designed for efficient execution of compiled C code in mind and have several built-in pointers for the task.

KNOWLEDGE ON ATMEGA16

ATmega16 is an 8-bit high performance microcontroller from Atmel's Mega AVR family with low power consumption. The architecture of ATmega16 is based on enhanced RISC (Reduced Instruction Set Computing) architecture with 131 powerful instructions. Most of the instructions are executed in one machine cycle. It can work on a maximum frequency of 16MHz.

FEATURES OF ATMEGA 16

- 16 kilo bytes of Flash memory.
- 512 bytes of EEPROM.
- 1 kilo bytes of SRAM.
- 131 instructions set.
- 32 x 8 bit general purpose working registers.
- On chip 2 cycle multiplier.
- Programming locks for software security.
- Two 8-bit timers.
- One 16 bit timer.
- 8 channel 10-bit ADC.

- On chip Analog comparator.
- Internal calibrated RC Oscillator.
- 32 programmable I/O lines.
- Programmable serial USART.
- Max Speed : 16 MHz(ATmega16).
- Watchdog timer.
- USB controller support.
- Ethernet controller support.
- LCD controller support.
- DMA controller.

Here, each I/O port has 3 registers associated with each it. These three registers are :

- DDRx
- PORTx
- PINx ; Where x stands for A, B, C, D

BASIC KNOWLEDGE OF AVR STUDIO SOFTWARE

Atmel AVR Studio is an Integrated Development Environment (IDE) for developing and debugging embedded Atmel AVR applications. It enables full control execution of programs on the AT90S In-Circuit Emulator or on the built-in AVR Instruction Set Simulator. It provides a project management tool, source file editor, simulator, assembler and front-end compiler for C/C++ programming, emulator and on-chip debugger. The AVR Studio gives a seamless and easy-to-use environment to write, build, and debug C/C++ and assembly code.

AVR Studio supports source level execution of assembly programs assembled with the Atmel Corporation's AVR Assembler and C programs compiled with compilers such as IAR Embedded Workbench, Code Vision AVR C compiler, GCC(GNU), etc. In AVR studio 5 there is an integrated C compiler, and need not be installed separately.

The assembler translates assembly source code into object code. The generated object code can be used as input to a simulator such as the ATMEL AVR Simulator or an emulator such as the ATMEL AVR In-Circuit Emulator. The assembler also generates a programmable hex code which can be programmed directly into the program memory of an AVR microcontroller. The assembler generates fixed code allocations, consequently no linking is necessary.

AVR Studio 4(or higher version) has a modular architecture which allows even more interaction with 3rd party software vendors. GUI plug-ins and other modules can be written and hooked to the system.