INSTITUTE OF ENGINEERING

ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT

KALANKI, KATHMANDU

(AFFILIATED TO TRIBHUVAN UNIVERSITY)

ADVANCED COLLEGE
OF ENGINEERING AND MANAGEMENT

LAB REPORT

SUBJECT: Data Communication

LAB NO: 1 to 5

Submitted By:

NAME: Himal Joshi
ROLL NO: ACE079BCT028
DATE: July 29, 2025

Submitted To:

Department of Electronics and Computer Engineering

LAB NO. 1

1.1 THEORY

Data communication is the process of transmitting digital or analog data between devices or systems through various media, such as wired or wireless channels. It involves critical processes like signal generation, modulation, encoding/decoding, and error detection and correction to ensure reliable and efficient data transfer. These processes are essential for applications ranging from telecommunications to internet-based systems, enabling global connectivity and real-time data exchange. Understanding signal behavior, channel characteristics, and error control mechanisms is fundamental to designing robust communication systems.

MATLAB, a high-performance technical computing platform, is extensively used in data communication for its matrix-based computational efficiency and advanced visualization capabilities. With toolboxes like the Signal Processing Toolbox and Communications Toolbox, MATLAB facilitates the simulation of communication systems, signal analysis, and performance evaluation under varying conditions, such as noise or interference. Its intuitive environment allows users to model modulation schemes, analyze channel effects, and develop algorithms for error detection, making it an invaluable tool for both academic learning and practical system design.

In this lab, MATLAB will be employed to explore core data communication concepts, including signal generation, modulation techniques, and channel modeling. Through simulations, we will visualize and analyze signal properties, study the impact of noise, and investigate basic error detection methods. These experiments will provide a deeper understanding of data communication principles and their practical implementation, laying the foundation for advanced studies in communication engineering.

1.2 SOURCE CODE

Listing 1.1: Basic Arithmetic

```

a = 82;
b = 28;
addition = a + b;
subtraction = a - b;
```

```
multiplication = a * b;
division = a / b;
remainder = mod(a, b);

disp(['Addition: ', num2str(addition)]);
disp(['Subtraction: ', num2str(subtraction)]);
disp(['Multiplication: ', num2str(multiplication)]);
disp(['Division: ', num2str(division)]);
disp(['Remainder: ', num2str(remainder)]);
```

Listing 1.2: Basic Matrix Arithmetic

```


A = [1 2; 3 4];
B = [5 6; 7 8];
matrix_add = A + B;
matrix_sub = A - B;
matrix_mul = A * B;


disp('Matrix Addition:');
disp(matrix_add);

disp('Matrix Subtraction:');
disp(matrix_sub);

disp('Matrix Multiplication:');
disp(matrix_mul);
```

Listing 1.3: Basic Trigonometric Operations

```

theta_deg = 45;
theta_rad = deg2rad(theta_deg);
sine_val = sin(theta_rad);
cosine_val = cos(theta_rad);


disp(['sin(', num2str(theta_deg), ') = ', num2str(sine_val)
    ]);
disp(['cos(', num2str(theta_deg), ') = ', num2str(cosine_val
    )]);
```

Listing 1.4: Basic Complex Number Operations

```
z = 3 + 4i;

real_part = real(z);
imag_part = imag(z);
conj_val = conj(z);
abs_val = abs(z);

disp(['Real part: ', num2str(real_part)]);
disp(['Imaginary part: ', num2str(imag_part)]);
disp(['Absolute value: ', num2str(abs_val)]);
disp(['Conjugate: ', num2str(conj_val)]);
```

1.3 OUTPUT

```
>> Basic_Arithmetic
Addition: 110
Subtraction: 54
Multiplication: 2296
Division: 2.9286
Remainder: 26
```

Figure 1.1: Basic Arithmetic

```
>> Matrix
Matrix Addition:
     6     8
    10    12

Matrix Subtraction:
    -4    -4
    -4    -4

Matrix Multiplication:
    19    22
    43    50
```

Figure 1.2: Basic Matrix Arithmetic

```
>> Basic_Trigonometry
sin(45) = 0.70711
cos(45) = 0.70711
```

Figure 1.3: Basic Trigonometric Operation

```
>> Basic_complex_number_operation
Real part: 5
Imaginary part: 4
Absolute value: 6.4031
Conjugate: 5-4i
```

Figure 1.4: Basic Complex Number Operation

LAB NO. 2

2.1 THEORY

A line plot in MATLAB is a fundamental two-dimensional chart that links data points with straight segments, widely utilized to display mathematical functions, track data patterns, or explore variable relationships.

Trigonometric wave graphs, including sine and cosine waves, highlight periodic patterns and are crucial for analyzing signals and waveforms in data communication. MATLAB's integrated `sin` and `cos` functions enable straightforward plotting of these signals across a range of values, offering insights into amplitude, frequency, and phase properties.

MATLAB employs commands like `plot` for smooth line graphs and `stem` for discrete data representations.

2.2 SOURCE CODE

Listing 2.1: Line Graph Visualization

```
a = [5,8];
b = [12,22];
x = [2,3];
y = [2,6];
c = [4,6];
d = [8,12];

subplot(3,1,1);
plot(a, b, '--black', 'LineWidth', 1)
title('LinePlot1/Himal Joshi/ACE079BCT028');
legend('Dashed Line')

subplot(3,1,2);
plot(x, y, '-blue', 'LineWidth', 2)
title('LinePlot2/Himal Joshi/ACE079BCT028');
legend('Solid Line')

subplot(3,1,3);
```

```
stem(c, d, '-red')
title('LinePlot3/Himal Joshi/ACE079BCT028');
legend('Discrete Points')
grid on;
```

Listing 2.2: Waveform Representation (Continuous and Discrete)

```
a = 6;
f = 4;
t = 0:0.015:1;
y = a * sin(2 * pi * f * t);
yy = a * cos(2 * pi * f * t);
subplot(4,2,1)
plot(t, y, '-red')
hold on
plot(t, zeros(size(t)), '-black')
title('Lab2/Himal Joshi/SineWaveContinuous')
legend('Sine Wave')

subplot(4,2,2)
plot(t, yy, '-blue')
hold on
plot(t, zeros(size(t)), '-black')
title('Lab2/Himal Joshi/CosineWaveContinuous')
legend('Cosine Wave')

subplot(4,2,3)
stem(t, y, '-red')
hold on
plot(t, zeros(size(t)), '-black')
title('Lab2/Himal Joshi/SineWaveDiscrete')
legend('Sine Wave')

subplot(4,2,4)
stem(t, yy, '-blue')
hold on
plot(t, zeros(size(t)), '-black')
title('Lab2/Himal Joshi/CosineWaveDiscrete')
legend('Cosine Wave')

subplot(4,2,5)
plot(t, yy, '-blue')
```

```
hold on
plot(t, y, '-red')
hold on
plot(t, zeros(size(t)), '-black')
hold on
title('Lab2/Himal Joshi/SineCosineOverlay')
legend('Cosine Wave', 'Sine Wave')

subplot(4,2,6)
stem(t, yy, '-blue')
hold on
stem(t, y, '-red')
hold on
plot(t, zeros(size(t)), '-black')
hold on
title('Lab2/Himal Joshi/SineCosineDiscrete')
legend('Cosine Wave', 'Sine Wave')
grid on
```

Listing 2.3: Unified Wave Display

```
a = 6;
f = 6;
t = 0:0.015:1;
y = a * sin(2 * pi * f * t);
yy = a * cos(2 * pi * f * t);
plot(t, y, '-red')
hold on
plot(t, zeros(size(t)), '-black')
hold on
plot(t, yy, '-blue')
hold on
stem(t, y, '-green')
hold on
stem(t, yy, '-green')
hold on
title('Lab2/Himal Joshi/CombinedWaveforms')
```

2.3 OUTPUT

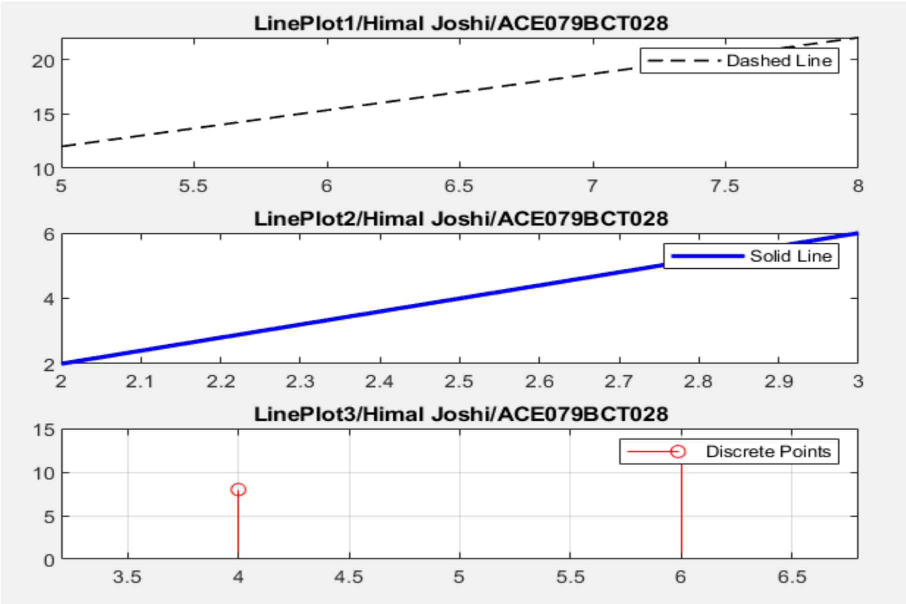

Figure 2.1: Line Graph Visualization

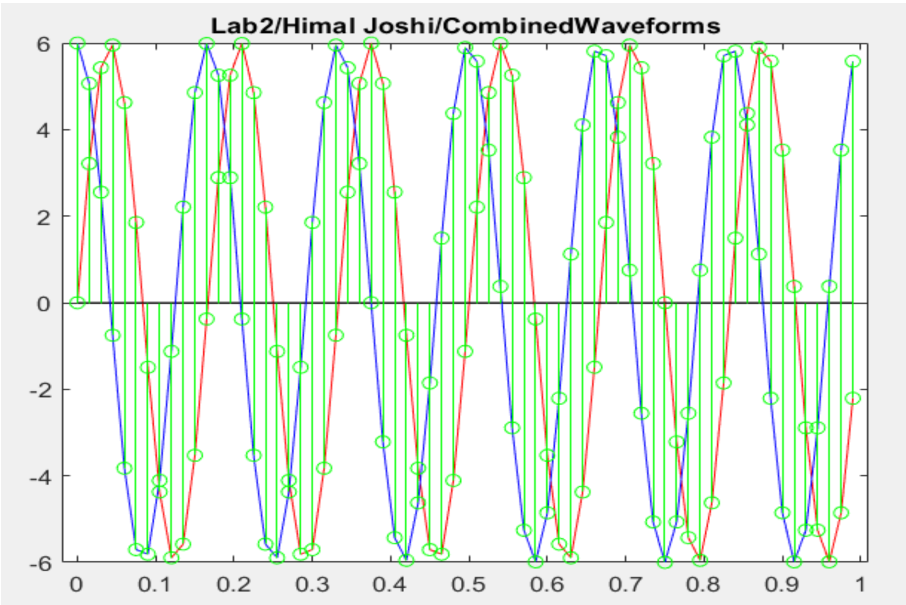

Figure 2.2: Unified Wave Display

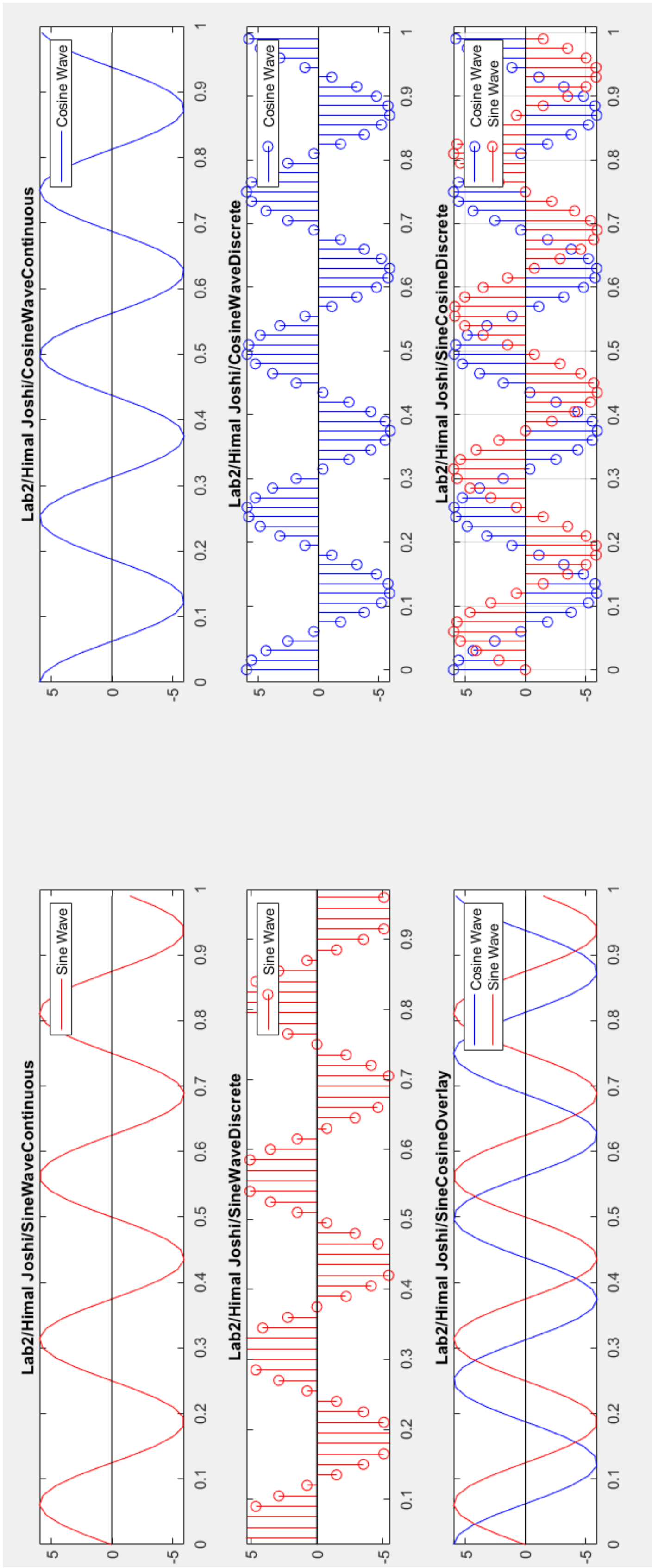

Figure 2.3: Waveform Representation (Continuous and Discrete)

LAB NO. 3

3.1 THEORY

Fundamental signal functions play a key role in data communication and signal processing. The unit step function represents a signal that activates at a specific moment. A square wave, oscillating between two levels periodically, is prevalent in digital circuits. The signum function determines the sign of a given value. The ramp function depicts a signal that increases linearly from zero. The impulse function simulates a brief, intense signal spike for system response analysis. Noise, as random interference, impacts communication clarity negatively.

3.2 SOURCE CODE

Listing 3.1: Unit Step Signal Generation

```
t = -2:0.1:2
unitstep = t >= 0;

subplot(2,1,1)
plot(t, unitstep)
legend("Step Signal")
title('Lab3/Himal Joshi/UnitStepContinuous')

subplot(2,1,2)
stem(t, unitstep, '-blue')
legend("Step Signal")
title('Lab3/Himal Joshi/UnitStepDiscrete')
```

Listing 3.2: Square Wave Simulation

```
a = 6;
f = 4;
t = 0:0.01:1;
s = a * sign(sin(2 * pi * f * t));
c = a * sign(cos(2 * pi * f * t));
subplot(2,1,1)
plot(t, s, '-blue')
legend('Square Sine')
title('Lab3/Himal Joshi/SquareSineWave')
```

```
subplot(2,1,2)
plot(t, c, '-blue')
legend('Square Cosine')
title('Lab3/Himal Joshi/SquareCosineWave')
```

Listing 3.3: Signum Function Implementation

```
x = zeros(size(t));
t = -2:0.1:2;
for i = 1:length(t)
    if t(i) > 0
        x(i) = 1;
    elseif t(i) == 0
        x(i) = 0;
    else
        x(i) = -1;
    end
end
subplot(2,1,1)
plot (t,x, '-red')
legend("Signum function")
title('Lab3/Himal Joshi/Ramp - Continuous')

subplot(2,1,2)
stem(t,x, '-red')
legend("Ramp function")
title('Lab3/Himal Joshi/Ramp - Discrete')
```

Listing 3.4: Ramp Function Construction

```
x = zeros(size(t));
t = -2:0.1:2
for i = 1:length(t)
    if t(i) >= 0
        x(i) = t(i);
    else
        x(i) = 0;
    end
end
subplot(2,1,1)
plot(t, x, '-blue')
legend("Ramp Curve")
```

```
title('Lab3/Himal Joshi/RampContinuous')

subplot(2,1,2)
stem(t, x, '-blue')
legend("Ramp Curve")
title('Lab3/Himal Joshi/RampDiscrete')
```

Listing 3.5: Noise Signal Integration

```
a = 5
f = 3
t = 0:0.01:1
s = a * sin(2 * pi * f * t);
c = a * cos(2 * pi * f * t);
x = rand(1, length(t));
z1 = x + s;
z2 = x + c;
subplot(3,1,1)
plot(t, x)
legend('Noise Pattern')
title('Lab3/Himal Joshi/RandomNoise')

subplot(3,1,2)
plot(t, z1)
legend('Noisy Sine')
title('Lab3/Himal Joshi/NoisySineWave')

subplot(3,1,3)
plot(t, z2)
legend('Noisy Cosine')
title('Lab3/Himal Joshi/NoisyCosineWave')
```

Listing 3.6: Impulse Signal

```
t = -2:0.1:2
impulse = t == 0;

subplot(2,1,1)
plot(t, impulse)
legend("Impulse Spike")
title('Lab3/Himal Joshi/ImpulseContinuous')

subplot(2,1,2)
```

```
stem(t, impulse, '-blue')
legend("Impulse Spike")
title('Lab3/Himal Joshi/ImpulseDiscrete')
```

3.3 OUTPUT

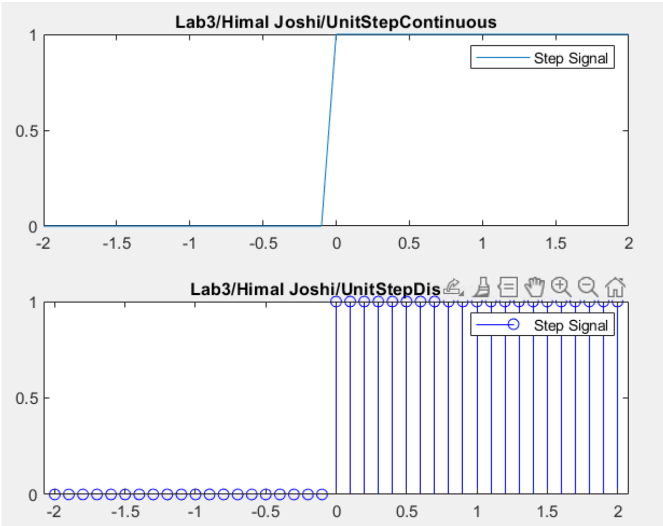

Figure 3.1: Unit Step Signal

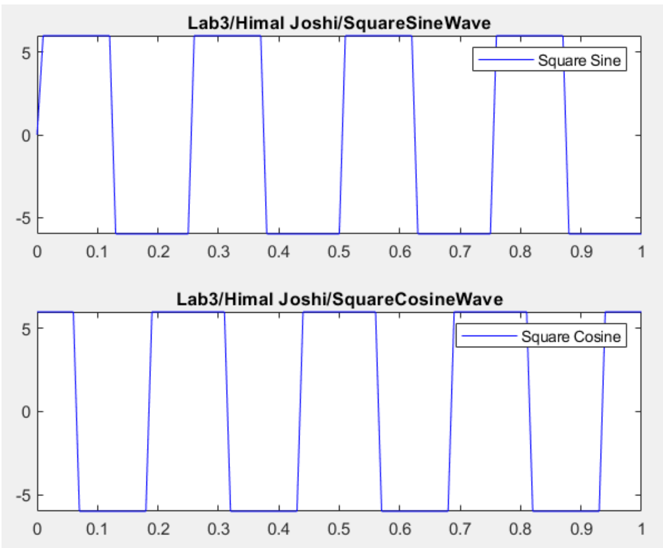

Figure 3.2: Square Wave Simulation

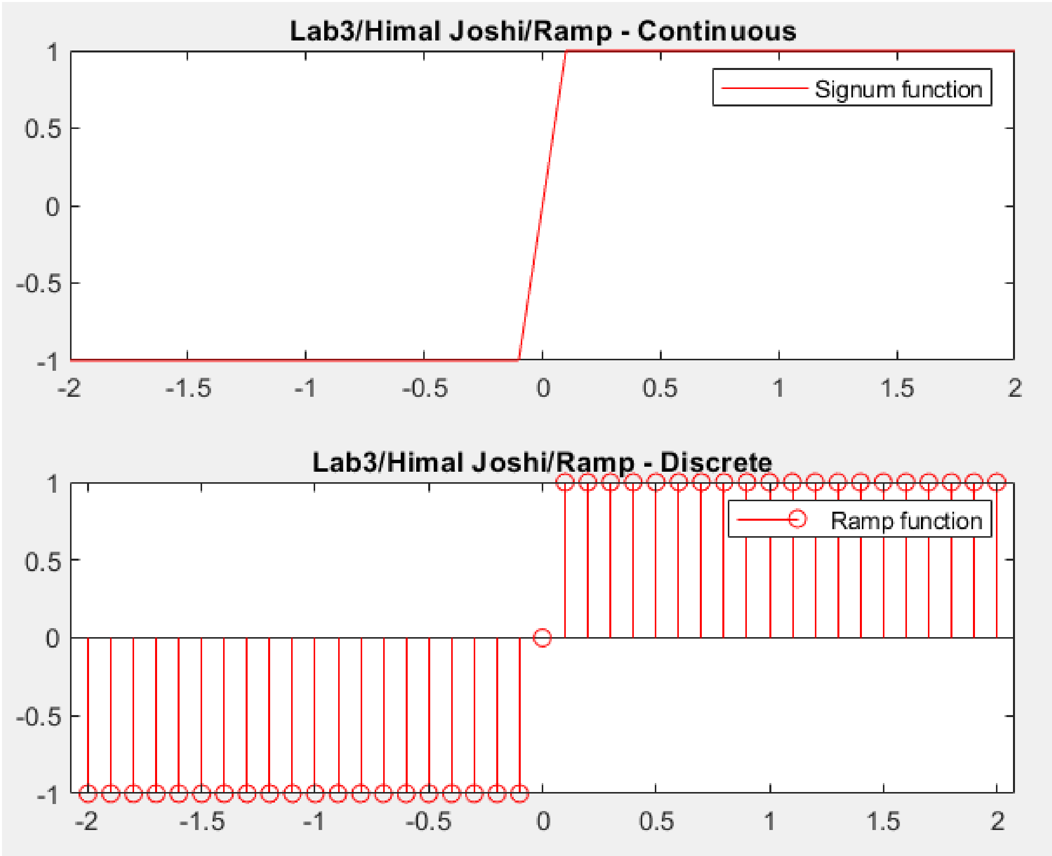

Figure 3.3: Signum Function Implementation

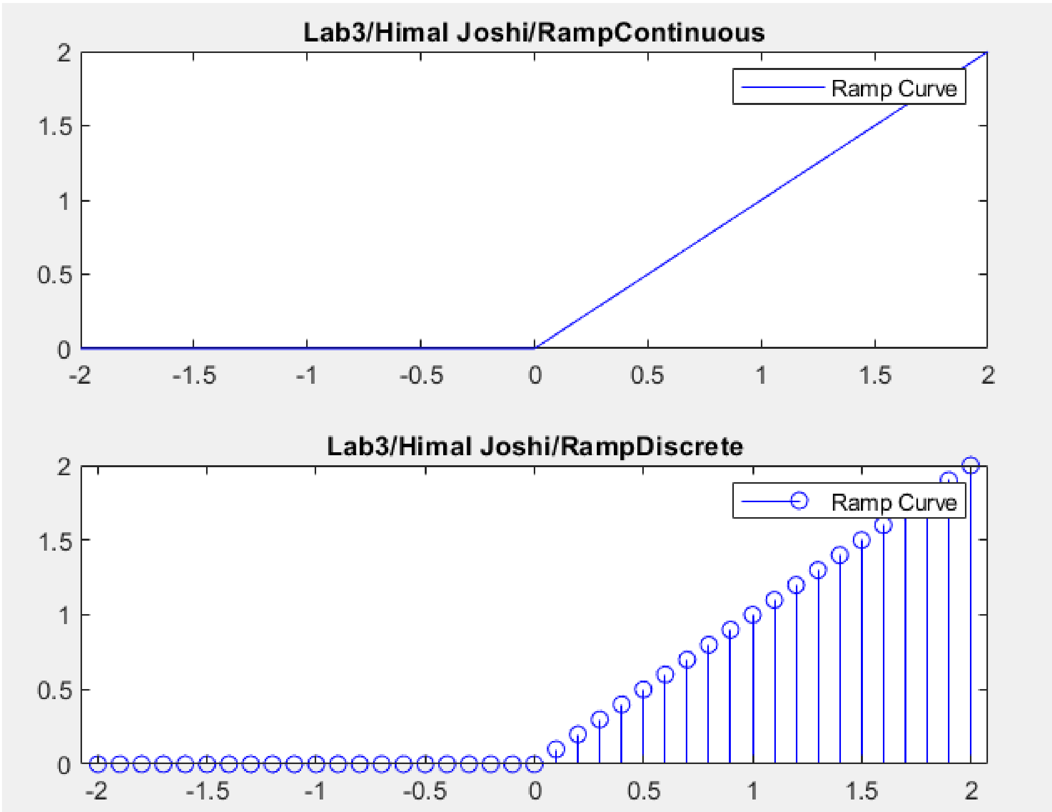

Figure 3.4: Ramp Function Construction

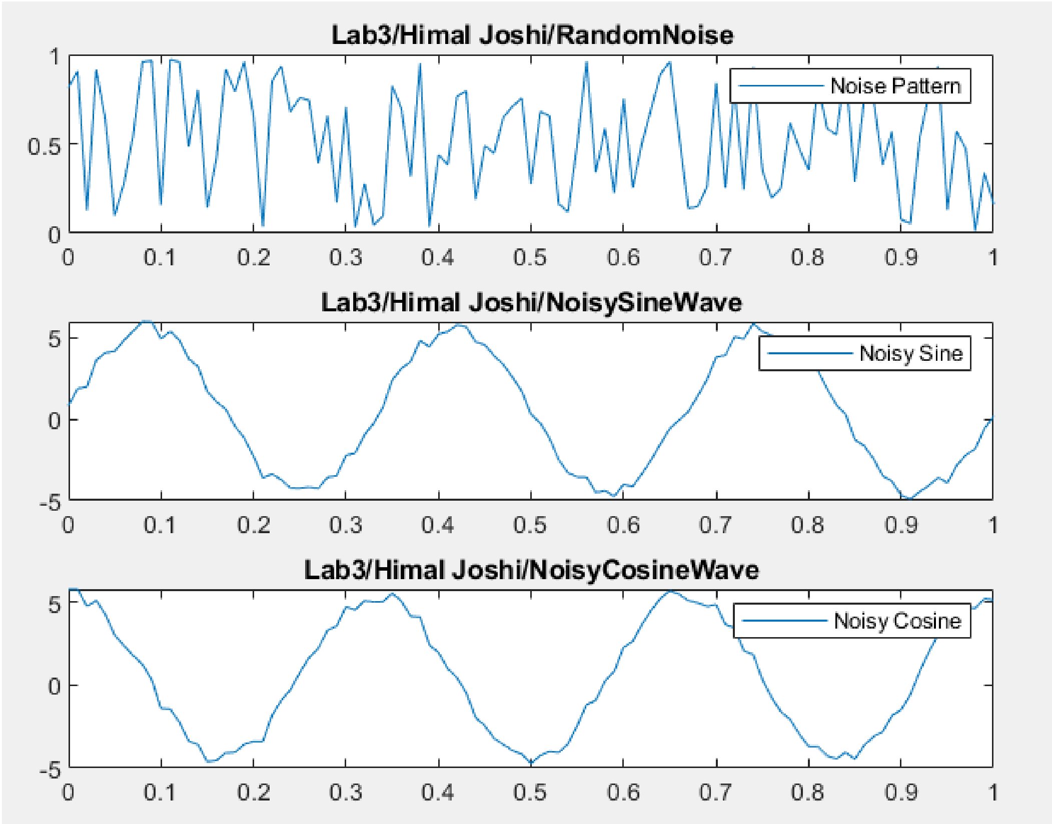

Figure 3.5: Noise Signal Integration

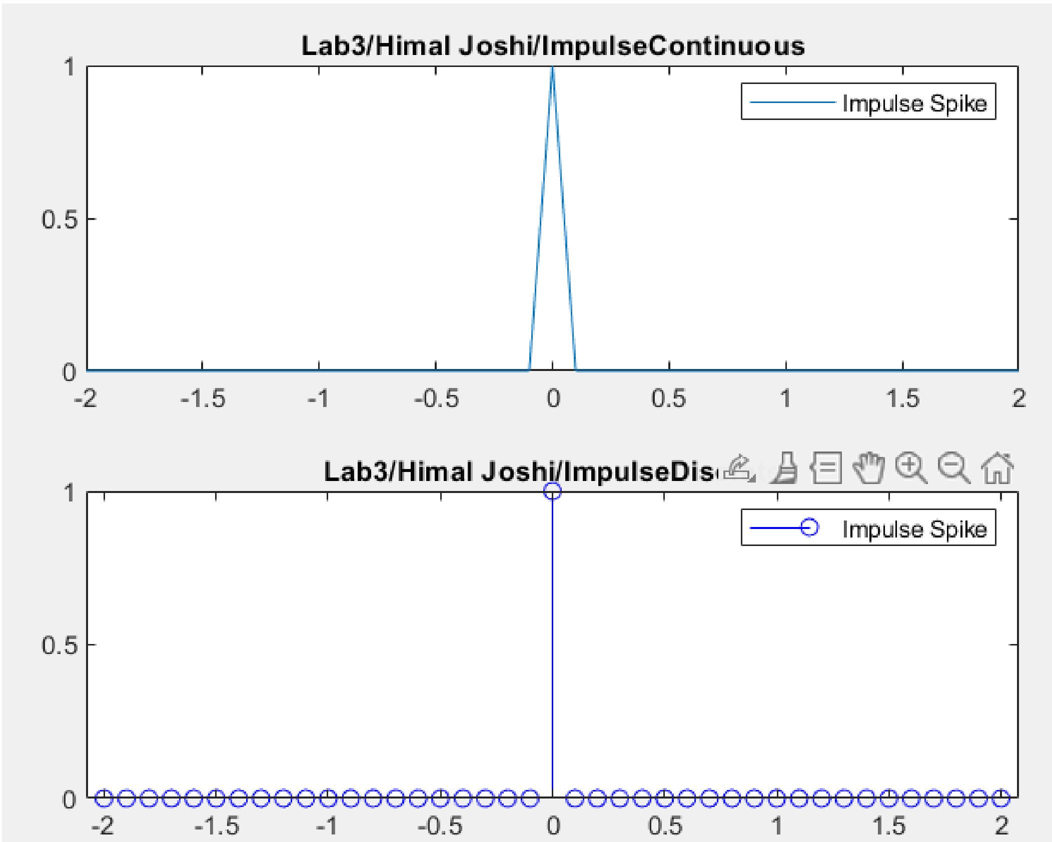

Figure 3.6: Impulse Function

LAB NO. 4

4.1 THEORY

Modulation involves adjusting properties of a high-frequency carrier signal (e.g., amplitude, frequency, or phase) based on a low-frequency message signal, facilitating efficient data transfer over extended distances and various media. The primary modulation types include:

- **Amplitude Modulation (AM)**: Adjusts the carrier's amplitude according to the message signal.
- **Frequency Modulation (FM)**: Modifies the carrier's frequency based on the message signal.
- **Phase Modulation (PM)**: Alters the carrier's phase in response to the message signal.

This technique is crucial in data communication for adapting signals to transmission media, enabling signal multiplexing, and enhancing resilience against noise and interference.

A function $f(t)$ is deemed **even** if it meets the condition:

$$f(-t) = f(t) \quad \text{for all } t$$

Its graph shows symmetry around the y-axis, with examples like $\cos(t)$ and t^2.

A function $f(t)$ is considered **odd** if it satisfies:

$$f(-t) = -f(t) \quad \text{for all } t$$

Its graph is symmetric about the origin, with examples including $\sin(t)$ and t^3.

Any function can be decomposed into even and odd components, a concept valuable in signal analysis, especially for Fourier series and transforms.

4.2 SOURCE CODE

Listing 4.1: Sinc Function Plotting

```
t = -12:0.12:12;

sinc_func = sin(t) ./ t;
sinc_func(t == 0) = 1;

subplot(2,1,1)
plot(t, sinc_func)
legend("Sinc Wave")
title('Lab4/Himal Joshi/SincContinuous')
xlabel('t')
ylabel('sinc(t)')

subplot(2,1,2)
stem(t, sinc_func, 'g')
legend("Sinc Wave")
title('Lab4/Himal Joshi/SincDiscrete')
xlabel('t')
ylabel('sinc(t)')
```

Listing 4.2: Exponential Decay Visualization

```
t = 0:0.12:12;

exp_decay = 6 * exp(-t);

subplot(2,1,1)
plot(t, exp_decay)
legend("Decay Curve")
title('Lab4/Himal Joshi/ExpDecayContinuous')
xlabel('t')
ylabel('e^{-t}')

subplot(2,1,2)
stem(t, exp_decay, 'g')
legend("Decay Curve")
title('Lab4/Himal Joshi/ExpDecayDiscrete')
xlabel('t')
ylabel('e^{-t}')
```

Listing 4.3: Even and Odd Component Analysis

```
t = -3:0.015:3;
x = zeros(size(t));

for i = 1:length(t)
    if t(i) >= 0
        x(i) = 3;
    else
        x(i) = 0;
    end
end

xe = (x + fliplr(x)) / 2;
xo = (x - fliplr(x)) / 2;

figure
subplot(3,2,1)
plot(t, x)
title('Lab4/Himal Joshi/BaseFunction')
xlabel('t')
ylabel('f(t)')

subplot(3,2,2)
stem(t, x, 'r')
title('Lab4/Himal Joshi/BaseFunctionDiscrete')
xlabel('t')
ylabel('f(t)')

subplot(3,2,3)
plot(t, xe)
title('Lab4/Himal Joshi/EvenComponent')
xlabel('t')
ylabel('f_e(t)')

subplot(3,2,4)
stem(t, xe, 'r')
title('Lab4/Himal Joshi/EvenComponentDiscrete')
xlabel('t')
ylabel('f_e(t)')

subplot(3,2,5)
```

```
plot(t, xo)
title('Lab4/Himal Joshi/OddComponent')
xlabel('t')
ylabel('f_o(t)')

subplot(3,2,6)
stem(t, xo, 'r')
title('Lab4/Himal Joshi/OddComponentDiscrete')
xlabel('t')
ylabel('f_o(t)')
```

Listing 4.4: Phase Modulation Demonstration

```
am = 5
fm = 3
t = -1:0.001:1

msg = am * cos(2 * pi * fm * t);

subplot(3,1,1)
plot(t, msg, '-blue')
hold on
plot(t, zeros(size(t)), '-black')
title('Lab4/Himal Joshi/MessageSignal')
legend('Cosine Message')

ac = 12;
fc = 50;

carrier = ac * cos(2 * pi * fc * t);

subplot(3,1,2)
plot(t, carrier, '-blue')
hold on
plot(t, zeros(size(t)), '-black')
title('Lab4/Himal Joshi/CarrierWave')
legend('Cosine Carrier')

kp = 1;
beta = kp * am;

PM = ac * cos(2 * pi * fc * t + beta * cos(2 * pi * fm * t))
```

```
    ;

subplot(3,1,3)
plot(t, PM, '-blue')
hold on
plot(t, zeros(size(t)), '-black')
title('Lab4/Himal Joshi/PhaseModulatedSignal')
legend('Phase Mod Signal')
```

Listing 4.5: Frequency Modulation Execution

```
am = 5
fm = 3
t = -1:0.001:1

msg = am * cos(2 * pi * fm * t);

subplot(3,1,1)
plot(t, msg, '-red')
hold on
plot(t, zeros(size(t)), '-black')
title('Lab4/Himal Joshi/MessageWave')
legend('Cosine Message')

ac = 10;
fc = 50;

carrier = ac * cos(2 * pi * fc * t);

subplot(3,1,2)
plot(t, carrier, '-red')
hold on
plot(t, zeros(size(t)), '-black')
title('Lab4/Himal Joshi/CarrierFrequency')
legend('Cosine Carrier')

kf = 5;
beta = (kf * am) / fm;

FM = ac * cos(2 * pi * fc * t + beta * sin(2 * pi * fm * t))
    ;

```

```
subplot(3,1,3)
plot(t, FM, '-red')
hold on
plot(t, zeros(size(t)), '-black')
title('Lab4/Himal Joshi/FrequencyModulatedSignal')
legend('Freq Mod Signal')
```

Listing 4.6: Amplitude Modulation

```
t = -1.5:0.0015:1.5;  % Changed from -1:0.001:1

a = 6;  % Changed from 5
f = 6;  % Changed from 5
y = a * sin(2 * pi * f * t);

subplot(3,1,1)
plot(t, y, '-green')  % Changed from -red
hold on
plot(t, zeros(size(t)), '-black')
title('Lab4/Himal Joshi/MessageSineWave')
legend('Sine Wave')

hf = 60;  % Changed from 50
a2 = 6;   % Changed from 5
carrier = a2 * sin(2 * pi * hf * t);

subplot(3,1,2)
plot(t, carrier, '-green')  % Changed from -red
hold on
plot(t, zeros(size(t)), '-black')
title('Lab4/Himal Joshi/CarrierSineWave')
legend('Carrier Wave')

am_signal = (1 + (y / a)) .* carrier;

subplot(3,1,3)
plot(t, am_signal, '-green')  % Changed from -red
hold on
plot(t, zeros(size(t)), '-black')
title('Lab4/Himal Joshi/AmplitudeModulatedWave')
legend('AM Signal')
```

4.3 OUTPUT

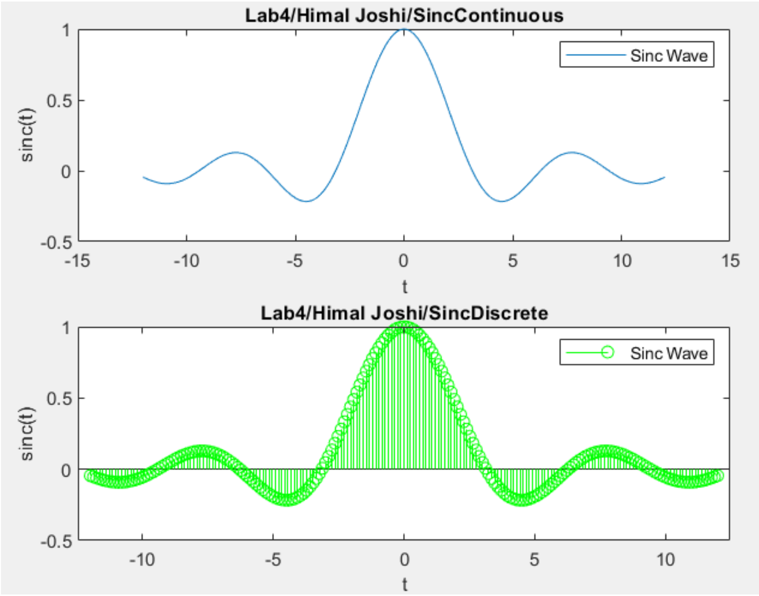

Figure 4.1: Sinc Function Plotting

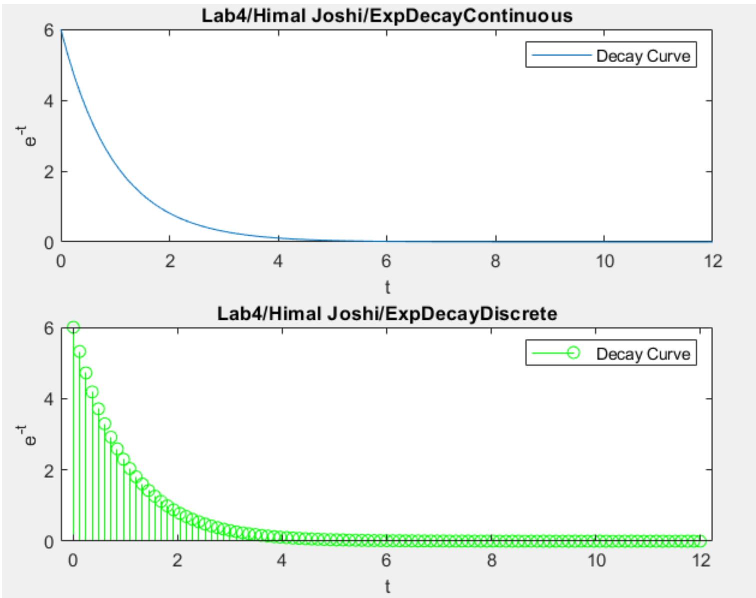

Figure 4.2: Exponentially Decaying Visualization

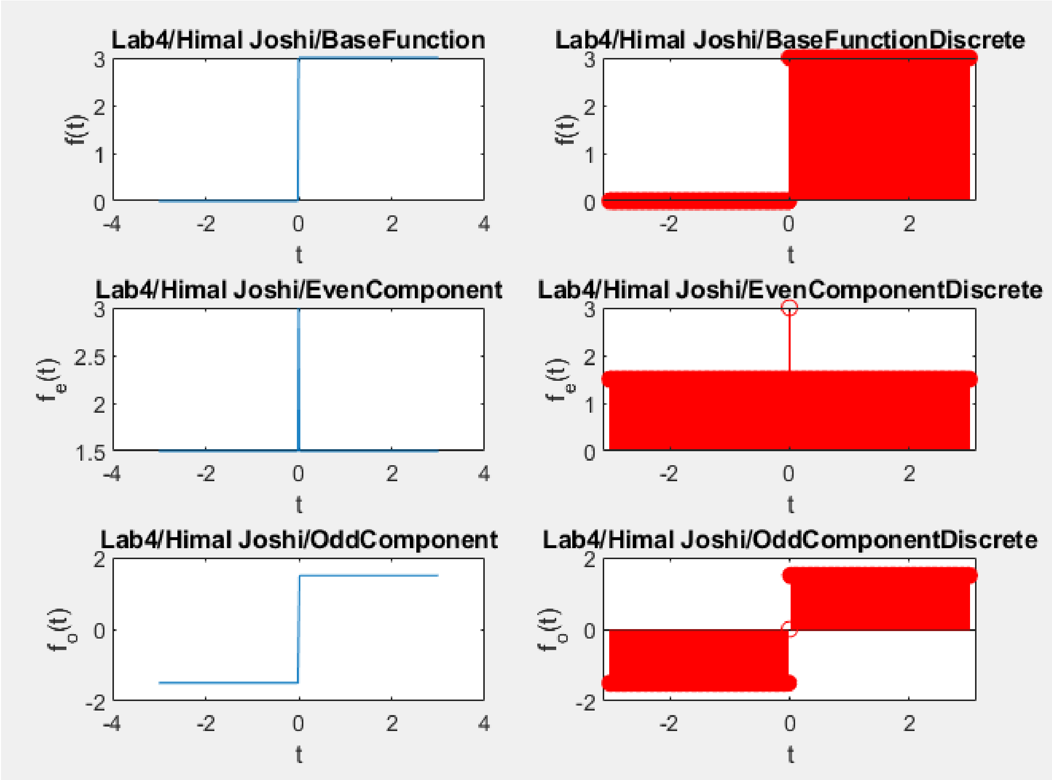

Figure 4.3: Even and Odd Component Analysis

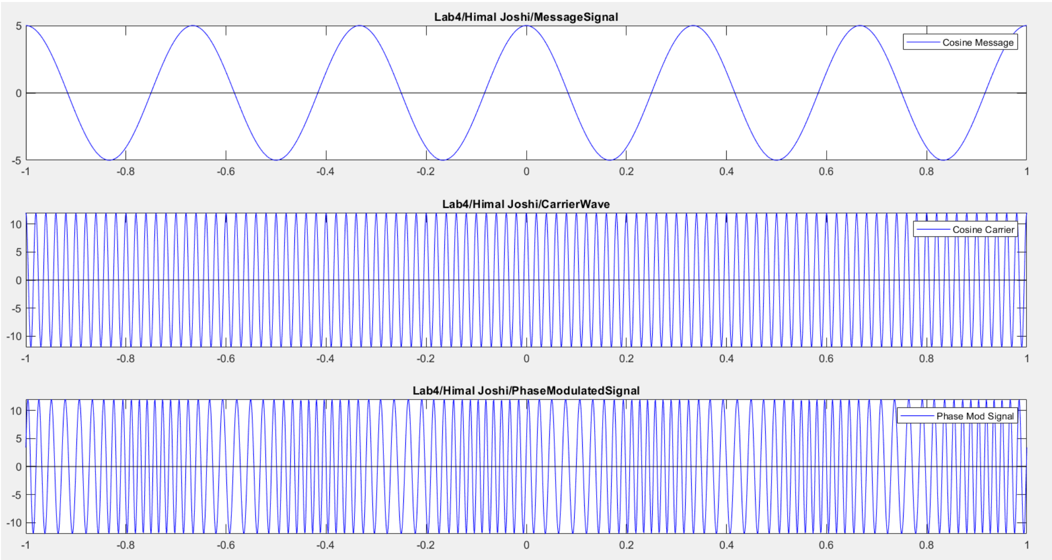

Figure 4.4: Phase Modulation Demonstration

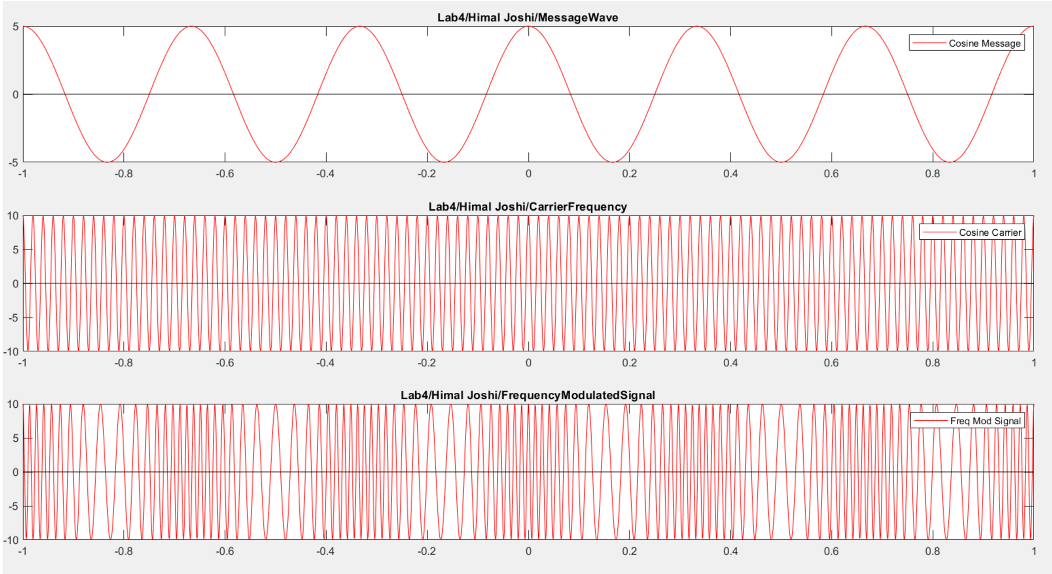

Figure 4.5: Frequency Modulation Execution

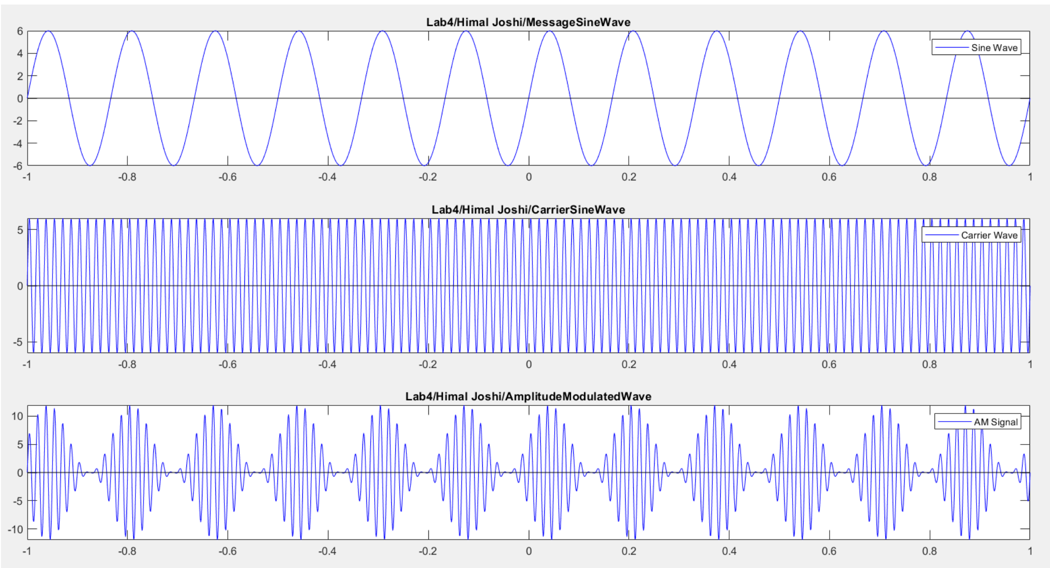

Figure 4.6: Amplitude Modulation

LAB NO. 5

5.1 THEORY

Digital modulation entails modifying the attributes of a high-frequency carrier signal (e.g., amplitude, frequency, or phase) based on a digital message signal, typically consisting of discrete values like 0 and 1. It is a cornerstone of digital communication systems for transmitting binary data over analog channels.

The main digital modulation types explored in this lab include:

- **Amplitude Shift Keying (ASK)**: Modulates the carrier amplitude, with one level for logic '1' and another (often zero) for logic '0', controlling the carrier's presence.
- **Frequency Shift Keying (FSK)**: Shifts the carrier frequency between two values based on the message, e.g., a higher frequency for '1' and a lower for '0'.
- **Phase Modulation (PM)**: Adjusts the carrier phase according to the digital signal, with Binary Phase Shift Keying (BPSK) using $+\theta$ for '1' and $-\theta$ for '0'.

Digital modulation provides benefits like improved noise resistance, optimized bandwidth usage, and simplified signal processing. These methods are integral to modern systems such as wireless networks, modems, and satellite communications.

In signal analysis, digital message signals like square waves are often odd functions due to their origin symmetry, similar to sine waves:

$$f(-t) = -f(t)$$

This symmetry aids in their analysis and representation using techniques like the Fourier Transform.

This lab highlights how basic modulation concepts extend to digital signaling, enabling effective binary data transmission on carrier waves.

5.2 SOURCE CODE

Listing 5.1: Amplitude Shift Keying (ASK) Signal Generation

```
clc;
clear all;
close all;

Amp = 5;
fp = 4;
fc = 50;
t = 0:0.001:1;

x = Amp * sin(2 * pi * fc *t);
subplot(3 , 1, 1);
plot(t,x);
grid on;
xlabel('time');
ylabel('amplitude');
title('Carrier Signal/Himal Joshi/ACE079BCT009');
legend("carrier wave");


y = (Amp/2) * (sign(sin(2 * pi * fp * t)) + 1);

subplot(3, 1, 2);
plot(t, y);
grid on;
xlabel('time');
ylabel('amplitude');
title('Himal Joshi/Message Signal');
legend('message signal');

a = x .* y;
subplot(3, 1, 3);
plot(t, a);
grid on;
xlabel('time');
ylabel('amplitude');
title('Himal Joshi/ASK Signal');
legend('ASK signal');
```

Listing 5.2: Frequency Shift Keying (FSK) Signal Generation

```
clc;
clear all;
close all;

Amp = 5;
fp = 4;
fc1 = 50;
fc2 = 100;
t = 0 : 0.001 : 1;

c1 = (Amp/2) * sin(2 * pi * fc1 * t);
c2 = (Amp/2) * sin(2 * pi * fc2 * t);

subplot(4, 1, 1);
plot(t, c1);
grid on;
xlabel('time');
ylabel('amplitude');
title('Himal Joshi/Carrier 1 Signal');
legend("carrier 1 wave");

subplot(4, 1, 2);
plot(t, c2);
grid on;
xlabel('time');
ylabel('amplitude');
title('Himal Joshi/Carrier 2 Signal');
legend("carrier 2 wave");

m = (Amp/2) * (sign(sin(2 * pi * fp * t)) + 1);

subplot(4, 1, 3);
plot(t, m);
grid on;
xlabel('time');
ylabel('amplitude');
title('Himal Joshi/Message Signal');
legend("message signal");

a = zeros(size(t));
```

```
for i = 1:length(t)
    if m(i) == 0
        a(i) = c2(i);
    else
        a(i) = c1(i);
    end
end

subplot(4, 1, 4);
plot(t, a);
grid on;
xlabel('time');
ylabel('amplitude');
title('Himal Joshi/FSK Signal');
legend("FSK signal");
```

Listing 5.3: Phase Shift Keying (PSK) Signal Generation

```
clc;
clear all;
close all;


Amp = 5;
fm = 2;
fc = 10;
t = 0:0.001:1;


x = Amp .* sin(2 * pi * fc * t);
subplot(3, 1, 1);
plot(t, x);
grid on;
xlabel('time');
ylabel('amplitude');
title('Himal Jsohi/Carrier Signal');


y = sign(sin(2 * pi * fm * t));
subplot(3, 1, 2);
plot(t, y);
grid on;
```

```
xlabel('time');
ylabel('amplitude');
title('Himal Joshi/Message Signal');
legend("message signal");


a = x .* y;
subplot(3, 1, 3);
plot(t, a);
grid on;
xlabel('time');
ylabel('amplitude');
title('Himal Joshi/PSK Signal');
```

5.3 OUTPUT

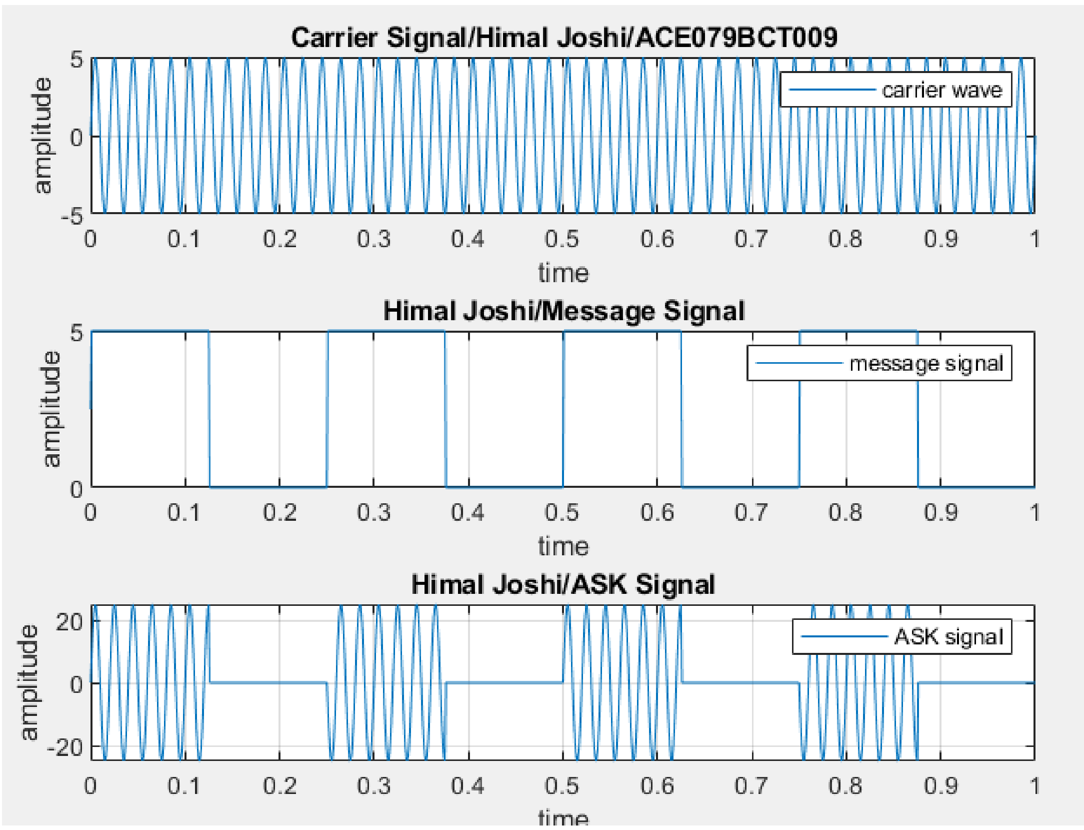

Figure 5.1: Frequency Modulation

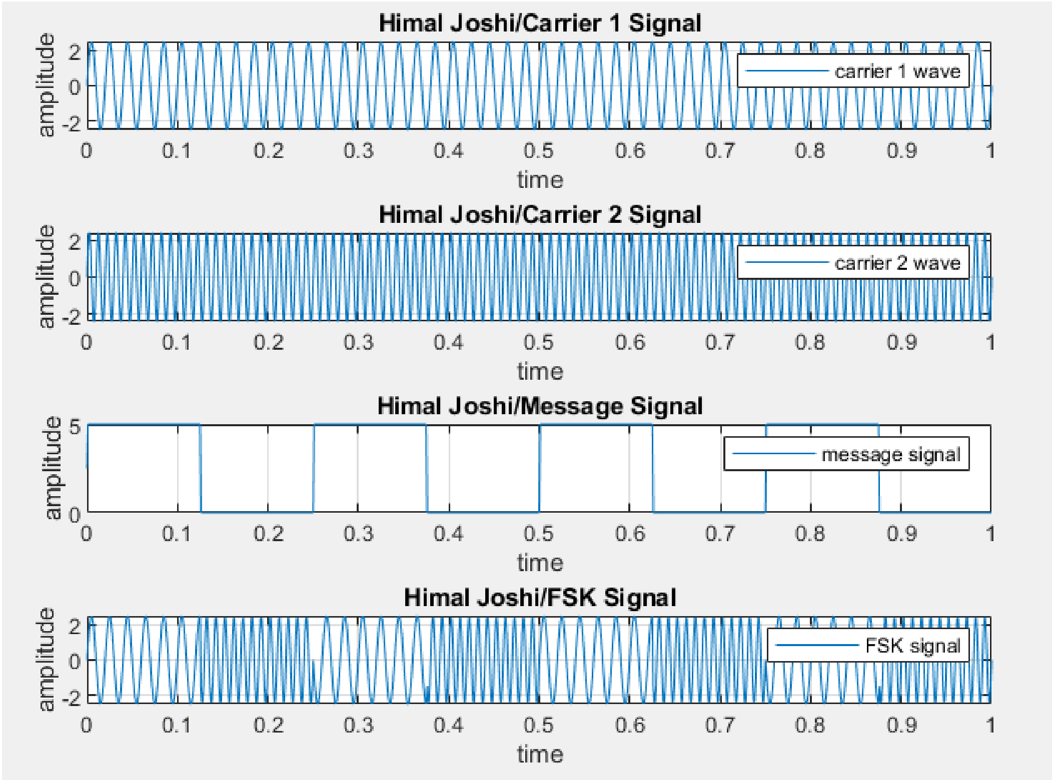

Figure 5.2: Amplitude Modulation

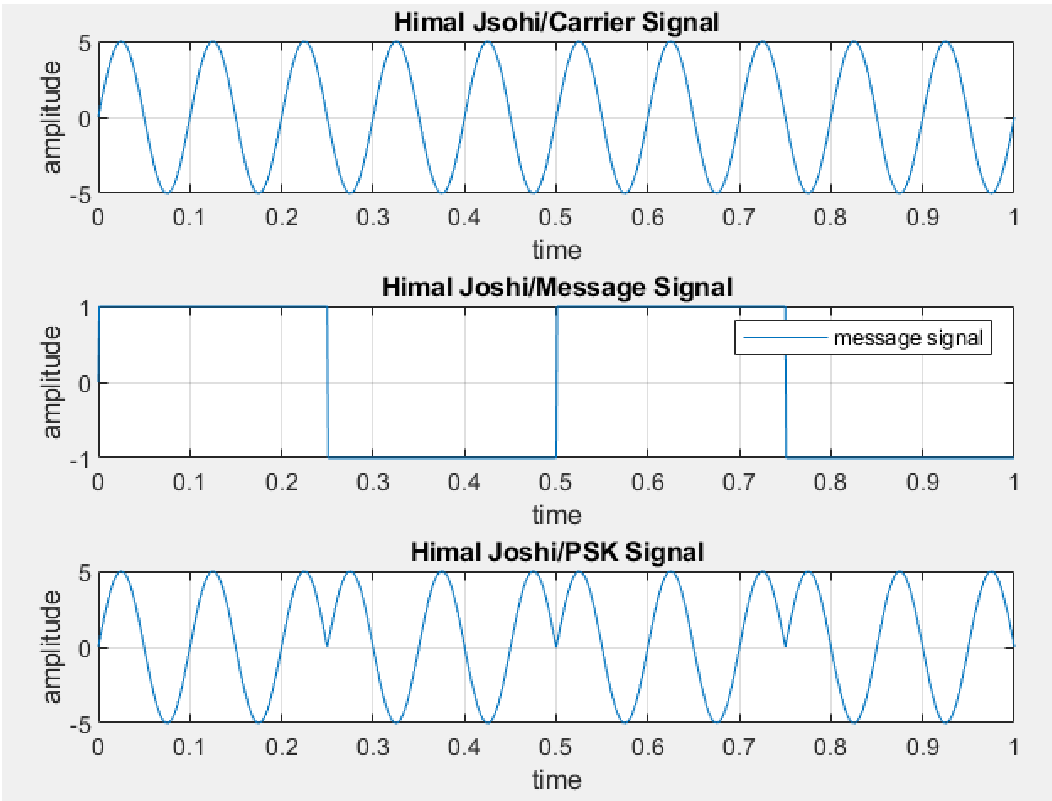

Figure 5.3: Phase Modulation