**LAB1: INTRODUCTION TO MATLAB**

**MATLAB Environment**

MATLAB, short for Matrix Laboratory, is a powerful tool used for numerical computing and visualization. It offers a high-level programming language, an interactive environment, and built-in functions for matrix operations, plotting, data analysis, and more. The MATLAB interface includes the Command Window, Workspace, Command History, and various other tools that facilitate coding and data analysis.

**Basic Arithmetic Operations**

MATLAB can handle a wide range of arithmetic operations, making it easy to perform calculations directly in the Command Window or within scripts. These operations include addition, subtraction, multiplication, division, and exponentiation. Understanding how to execute these operations is fundamental to using MATLAB effectively.

**Variables**

Variables in MATLAB are used to store data values that can be referenced and manipulated throughout your code. You can assign values to variables, perform operations using these variables, and display their values. Variables in MATLAB are dynamically typed, meaning you do not need to declare their type before assigning values to them.

**Simple Plotting**

One of MATLAB's strengths is its ability to create high-quality plots with ease. Plotting functions allow you to visualize data, which can help in analyzing trends and patterns. The basic plotting process involves generating data, creating plots, and customizing the appearance of these plots (e.g., adding titles, labels, and legends).

**Unit Step Function (u(t))**

* **Definition**: The unit step function, denoted as u(t)u(t)u(t), is a piecewise function that equals 0 for t<0 and 1 for t≥0.
* **Mathematical Expression**:
* **Usage**: Often used to represent signals that switch on at a specific time.

**Ramp Function (r(t))**

* **Definition**: The ramp function, denoted as r(t)r(t)r(t), is a linear function that equals 0 for t<0 and t for t≥0.
* **Mathematical Expression**:
* **Usage**: Represents signals that increase linearly with time.

**Unit Impulse Function (δ(t))**

* **Definition**: The unit impulse function, denoted by δ(t), is zero everywhere except at t=0, where it is theoretically infinite, and its integral over all time is 1.
* **Mathematical Expression**:
* **Usage**: Used to model idealized impulses in time series.

**Sinusoidal Signal**

* **Definition**: A sinusoidal signal can be represented in the form x(t)=Acos(ω0t±ϕ) or x(t)=Asin(ω0t±ϕ), where A is the amplitude, ω0 is the angular frequency, and ϕ is the phase shift.
* **Mathematical Expression**: x(t)=Acos(ω0t+ϕ) o x(t)=Asin(ω0t+ϕ)
* **Usage**: Widely used in signal processing, communications, and control systems.

**Sinc Function**

* **Definition**: The sinc function, denoted as sinc(t), is defined as.It is zero at t=±1,±2,±3,…
* **Mathematical Expression**: .

sinc(t)=0 for t=±1,±2,±3,…

* **Usage**: Appears in signal processing and the theory of Fourier transforms.

**Rectangular Signal**

* **Definition**: The rectangular signal, denoted as x(t) can be defined using the rect function. For amplitude A and width T, it can be represented as Arect(t/T).
* **Mathematical Expression**: x(t)=A rect(t/T)
* **Usage**: Often used to model pulses in digital systems.

**Dirac Delta Signal**

* **Definition**: The Dirac delta function, also denoted as δ(t), is a function that is zero everywhere except at zero, and its integral over the entire real line is equal to one.
* **Mathematical Expression**:
* **Usage**: Used in theoretical analysis of systems and signals.

**CODES**

**1.** **Cosine function:**

clc

clear all

close all

t=-pi:0.01:pi;

y=cos(t);

subplot(2,1,1);

plot(t,y);

xlabel('time');

ylabel('y(t)=cos(t)');

title('Continuous plot for Cosine function (Deepak Thapa BCT29)');

p=-pi:0.2:pi;

z=cos(p);

subplot(2,1,2);

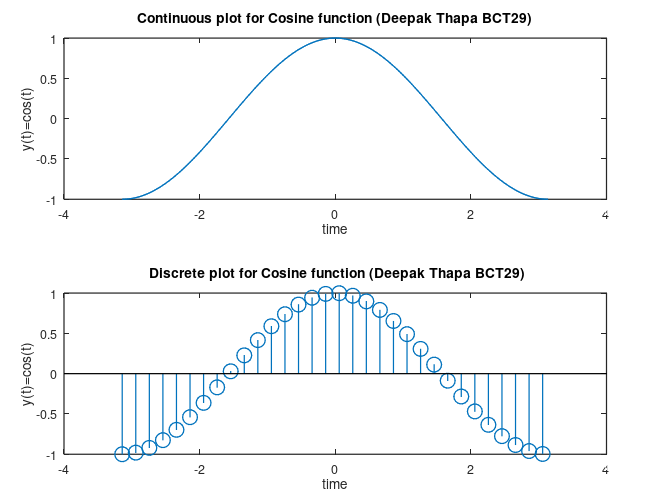
stem(p,z);

xlabel('time');

ylabel('y(t)=cos(t)');

title(' Discrete plot for Cosine function (Deepak Thapa BCT29)');

**Output:**



**2. Sin function:**

clc

clear all

close all

t=0:0.01:10;

y=sin(t);

subplot(2,1,1);

plot(t,y);

xlabel('time');

ylabel('amplitude');

title('Plot for y=sin(t) (Deepak Thapa BCT29)');

p=0:0.2:10;

z=sin(p);

subplot(2,1,2);

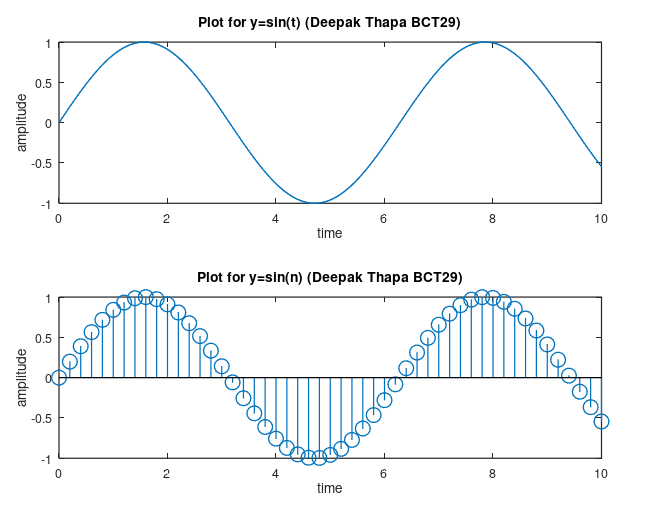
stem(p,z);

xlabel('time');

ylabel('amplitude');

title('Plot for y=sin(n) (Deepak Thapa BCT29)');

**Output:**



**3. Unit Step Function:**

clc

clear all

close all

t=-20:0.01:20;

a=length(t);

x=1;

while x<=a

if t(x)>0

u(x)=1;

else u(x)=0;

end

x=x+1;

end

subplot(211);

plot(t,u);

axis([-20 20 -1 2])

xlabel('time');

ylabel('amplitude');

title('Plot for unit step function (Deepak Thapa BCT29)');

r=-20:0.5:20;

b=length(r);

x=1;

while x<=b

if r(x)>0

p(x)=1;

else p(x)=0;

end

x=x+1;

end

subplot(212);

stem(r,p);

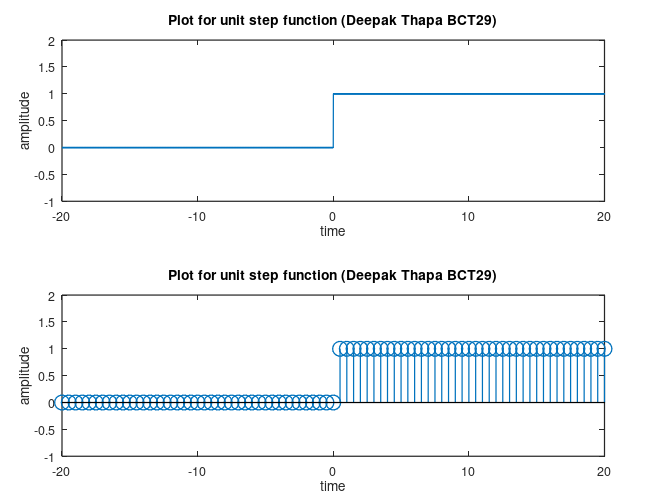
axis([-20 20 -1 2])

xlabel('time');

ylabel('amplitude');

title('Plot for unit step function (Deepak Thapa BCT29)');

**Output:**



**4. Unit Ramp Function**

clc

clear all

close all

t=-20:0.01:20;

a=length(t);

x=1;

while x<=a

if t(x)>0

u(x)=t(x);

else u(x)=0;

end

x=x+1;

end

subplot(211);

plot(t,u);

axis([-20 20 -1 10])

xlabel('time');

ylabel('amplitude');

title('Plot for unit ramp function (Deepak Thapa BCT29)');

r=-20:0.5:20;

b=length(r);

x=1;

while x<=b

if r(x)>0

p(x)=r(x);

else p(x)=0;

end

x=x+1;

end

subplot(212);

stem(r,p);

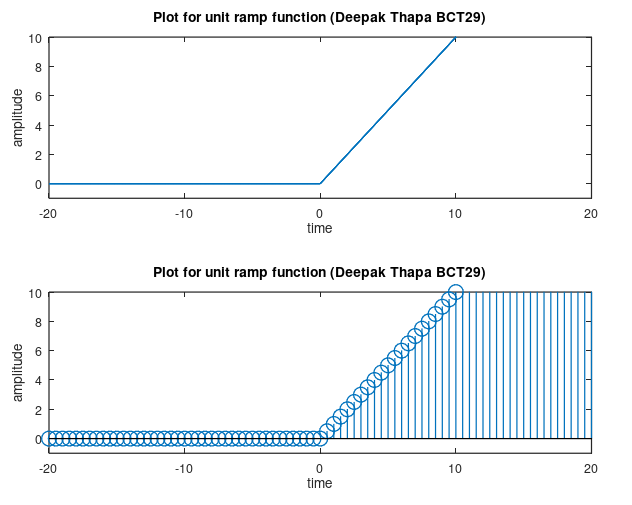
axis([-20 20 -1 10])

xlabel('time');

ylabel('amplitude');

title('Plot for unit ramp function (Deepak Thapa BCT29)');

**Output:**



**5. Sinc function:**

clc

clear all

close all

t=-20:0.01:20;

a=length(t);

x=1;

while x<=a

if t(x)==0

u(x)=1;

else u(x)=sin(t(x))/t(x);

end

x=x+1;

end

subplot(211);

plot(t,u);

axis([-20 20 -1 1.5])

xlabel('time');

ylabel('amplitude');

title('Continuous Plot for unit sinc function (Deepak Thapa BCT29)');

r=-20:0.5:20;

b=length(r);

x=1;

while x<=b

if r(x)==0

p(x)=r(x);

else p(x)=sin(r(x))/r(x);

end

x=x+1;

end

subplot(212);

stem(r,p);

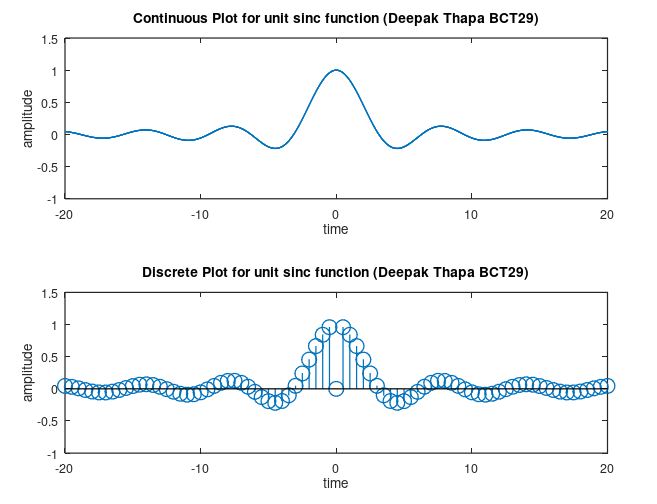
axis([-20 20 -1 1.5])

xlabel('time');

ylabel('amplitude');

title('Discrete Plot for unit sinc function (Deepak Thapa BCT29)');

**Output:**



**6. Rectangular Function**

clc

clear all

close all

t=-20:0.01:20;

a=length(t);

x=1;

while x<=a

if t(x)>-10 && t(x)<10

u(x)=1;

else u(x)=0;

end

x=x+1;

end

subplot(211);

plot(t,u);

axis([-20 20 -1 1.5])

xlabel('time');

ylabel('amplitude');

title('Continuous rectangular Function (Deepak Thapa BCT29)');

r=-20:0.5:20;

b=length(r);

x=1;

while x<=b

if r(x)>-10 && r(x)<10

p(x)=1;

else p(x)=0;

end

x=x+1;

end

subplot(212);

stem(r,p);

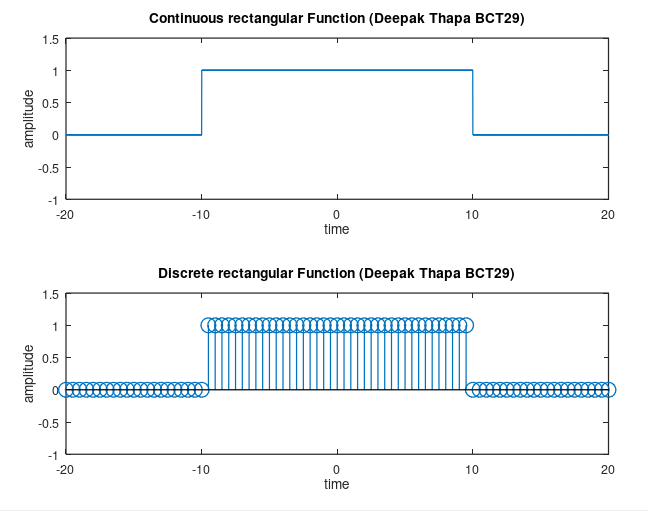
axis([-20 20 -1 1.5])

xlabel('time');

ylabel('amplitude');

title('Discrete rectangular Function (Deepak Thapa BCT29)');

**Output:**



**7. Dirac Delta function:**

clc

clear all

close all

t=-20:0.01:20;

a=length(t);

x=1;

while x<=a

if t(x)==0

u(x)=1;

else u(x)=0;

end

x=x+1;

end

stem(t,u);

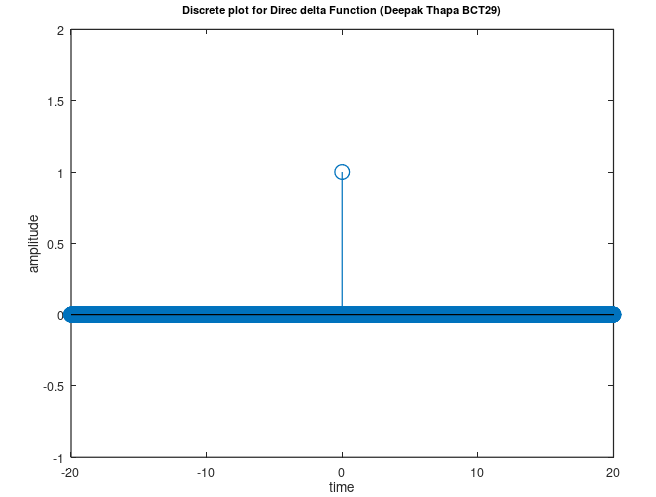
axis([-20 20 -1 2])

xlabel('time');

ylabel('amplitude');

title('Discrete plot for Direc delta Function (Deepak Thapa BCT29)');

**Output:**

****

**DISCUSSION**

The MATLAB outputs demonstrate the visualization of fundamental signal functions, including cosine, sine, unit step, ramp, sinc, rectangular, and Dirac delta functions. Each output illustrates both continuous and discrete forms, providing a comprehensive understanding of these functions' behaviors. The cosine and sine function outputs effectively show their periodic nature and sampling effects. The unit step and ramp function outputs highlight abrupt transitions and linear growth, respectively. The sinc function output reveals its central peak and oscillatory nature, which is crucial for sampling theory, while the rectangular function output depicts pulse shaping. Finally, the Dirac delta function output represents an idealized impulse, essential for theoretical signal analysis. These visualizations collectively enhance the understanding of these fundamental signal processing concepts.

**CONCLUSION**

These plots offer a clear understanding of fundamental signal functions. They provide essential insights into continuous versus discrete representations, transitions, and sampling effects. These visualizations are crucial for both theoretical analysis and practical applications in signal processing.

**LAB 2: PLOT AN EXPONENTIAL FUNCTION, A PIECEWISE FUNCTION, AND ANALYZE HOW THEIR SIGNALS SHIFT AND SCALE OVER TIME, INCLUDING EXAMINING CONVOLUTION EFFECTS.**

**Exponential Function**

An exponential function is a mathematical expression characterized by its rapid increase or decrease. It is defined as f(x) = a \* b^x, where:

* a represents the initial value or scaling factor,
* b is the base of the exponential, and
* x is the exponent or input variable.

Exponential functions exhibit growth when b>1 and decay when 0<b<1. They are commonly used to model phenomena that change at a constant percentage rate, such as population growth, radioactive decay, and compound interest.

**Piecewise Function**

A piecewise function is a mathematical function that is defined by different expressions over different intervals of the input variable. Each piece or segment of the function is valid within a specific range, leading to distinct behaviors in different regions. Piecewise functions are useful for modeling situations where different rules apply in different scenarios, such as tax brackets or electrical systems with varying behaviors.

**Time Shifting**

Time shifting involves adjusting a signal’s position along the time axis by adding or subtracting a constant value to its time variable. This operation effectively delays or advances the signal, which can be critical for aligning signals in time-series analysis, synchronizing data, or testing systems under different timing conditions. Time shifting does not alter the shape or amplitude of the signal but changes its temporal occurrence.

**Time Scaling**

Time scaling refers to modifying the time axis of a signal by multiplying it by a constant factor. This can make the signal occur faster or slower, depending on whether the factor is greater than or less than one. Time scaling affects the signal’s speed but preserves its shape. This technique is valuable in signal processing for adjusting the rate at which a signal progresses or for resampling signals at different rates.

**Convolution**

Convolution is a mathematical process that combines two functions to produce a third function, representing the way one function affects another. In signal processing, convolution describes how an input signal is transformed by a system's response. It is a fundamental operation for analyzing and designing systems, filtering signals, and understanding the interaction between input and output in various applications, such as audio processing and image filtering.

**CODES**

**1. Exponential Signal**

clc

clear all

close all

n=0:40;

a=1.1;

c=2;

y=c\*a.^n;

subplot (211);

plot(n,y);

xlabel('n------->');

ylabel('y------->');

title('Continuous plot for y=ca^n (Deepak Thapa BCT29)');

m=0:40;

b=1.1;

d=2;

t=d\*b.^m;

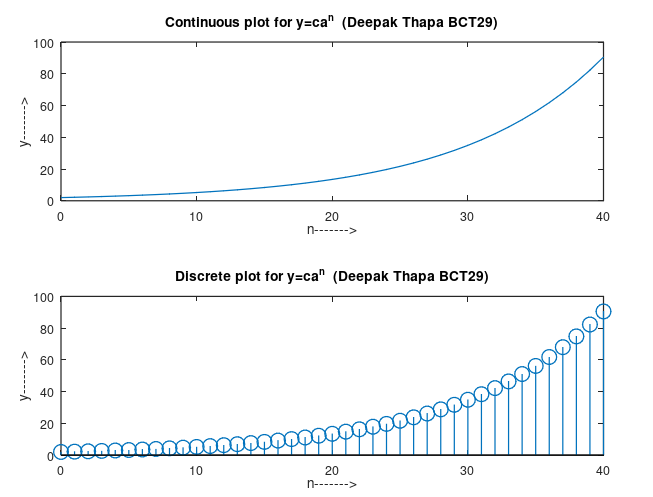
subplot (212);

stem(m,t);

xlabel('n------->');

ylabel('y------->');

title('Discrete plot for y=ca^n. (Deepak Thapa BCT29));

**Output:**

**2. Piecewise signal**

clc

clear all

close all

t0=-5:0.01:-3;

x0=zeros(size(t0));

t1=-3:0.01:-2;

x1=ones(size(t1));

t2=-2:0.02:-1;

x2=-t2-3;

t3=-1:0.01:1;

x3=ones(size(t3));

t4=1:0.02:2;

x4=zeros(size(t4));

x=[x0 x1 x2 x3 x4];

t=[t0 t1 t2 t3 t4];

plot (t,x,'LineWidth',1);

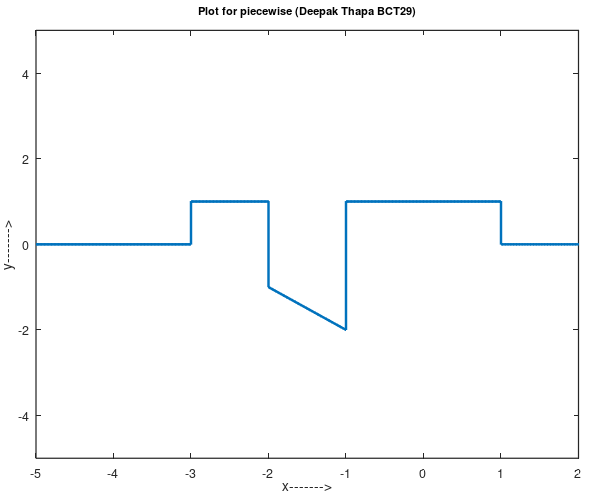
axis ([-5 2 -5 5]);

xlabel('x------->');

ylabel('y------->');

title('Plot for piecewise (Deepak Thapa BCT29)');

**Output:**



**3. Variable Sin function**

clc

clear all

close all

t=0:0.01:4\*pi;

y=sin(t);

subplot(511);

plot(t,y);

axis([0 4\*pi -2 2]);

xlabel('x------->');

ylabel('y------->');

title( plot of sint.');

t=0:0.01:4\*pi;

y=sin(2\*t);

subplot(512);

plot(t,y);

axis([0 4\*pi -2 2]);

xlabel('x------->');

ylabel('y------->');

title( plot of sin2t.');

t=0:0.01:4\*pi;

y=sin(t/2);

subplot(513);

plot(t,y);

axis([0 4\*pi -2 2]);

xlabel('x------->');

ylabel('y------->');

title( plot of sin(t/2).');

t=0:0.01:4\*pi;

y=sin(t-2);

subplot(514);

plot(t,y);

axis([0 4\*pi -2 2]);

xlabel('x------->');

ylabel('y------->');

title( plot of sin(t-2).');

t=0:0.01:4\*pi;

y=sin(t+2);

subplot(515);

plot(t,y);

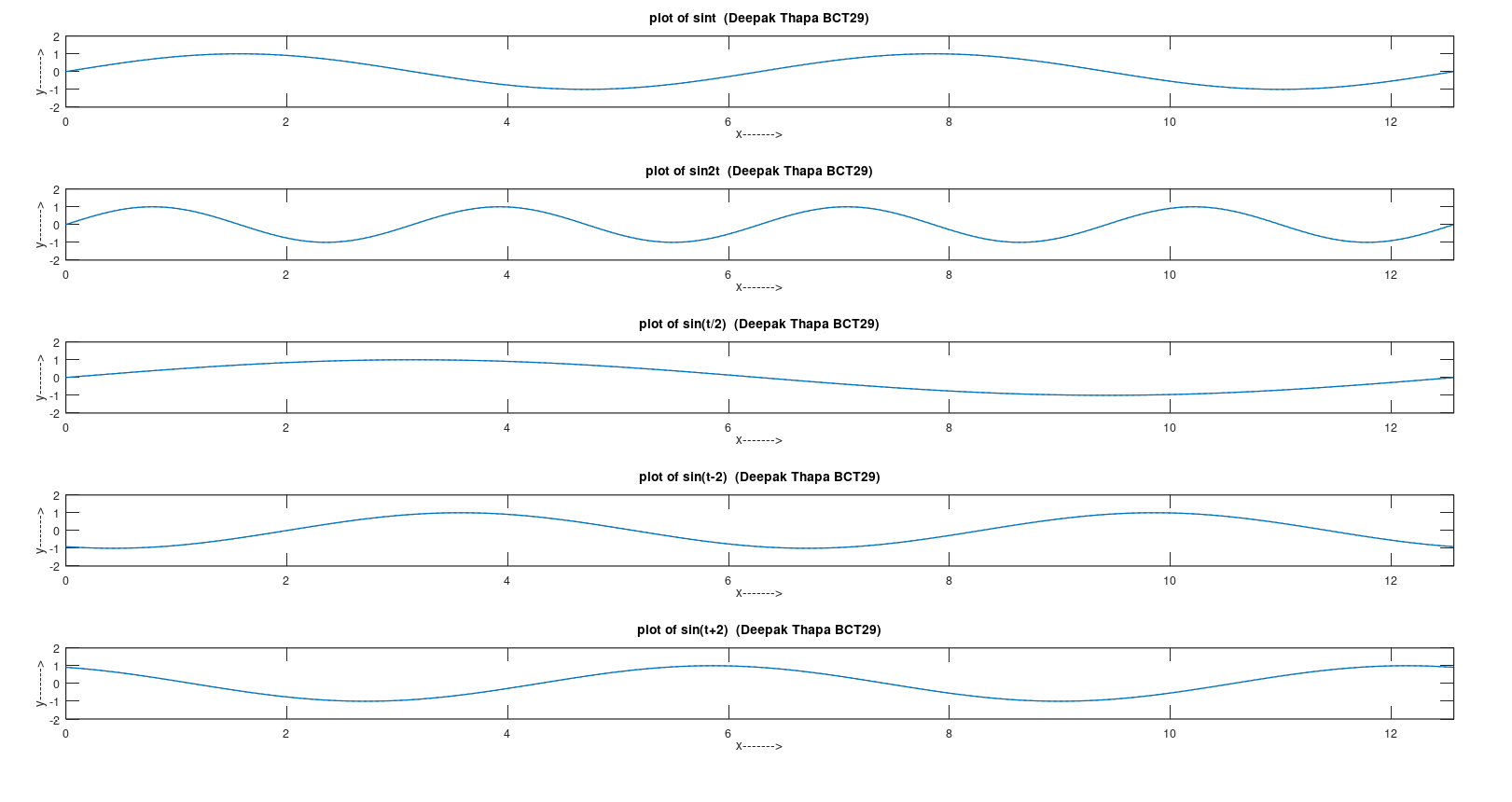
axis([0 4\*pi -2 2]);

xlabel('x------->');

ylabel('y------->');

title( plot of sin(t+2).');

**Output:**



**4. Convolution**

clc

clear all

close all

t1=0:4;

t2=0:6;

t3=0:11;

unitstep=t1<=4;

x=unitstep;

h=1.4.^t2;

subplot(311);

stem(t1,x);

xlabel('n------>');

ylabel('x------>');

title('x[n] (Deepak Thapa BCT29)');

axis([0 10 0 2]);

subplot(312);

stem(t2,h);

xlabel('n------>');

ylabel('h------>');

title('h[n] (Deepak Thapa BCT29)');

axis([0 10 0 10]);

y=conv(x,h);

subplot(313);

stem(y);

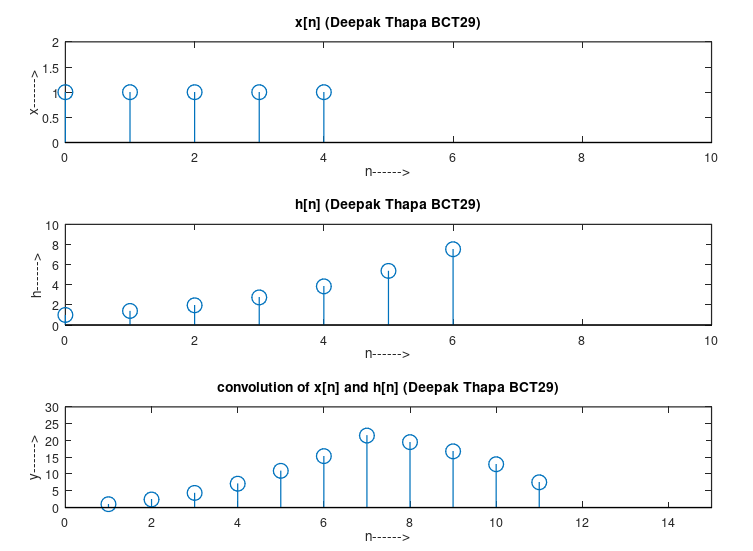
xlabel('n------>');

ylabel('y------>');

title( convolution of x[n] and h[n] (Deepak Thapa BCT29)');

axis([0 15 0 30]);

**Output:**

****

**DISCUSSION**

The MATLAB outputs provided demonstrate the visualization of different signal types and operations. The exponential signal output includes plots of both continuous and discrete forms, effectively showcasing the growth behavior. The piecewise signal output illustrates different segments with distinct characteristics combined into a single plot, emphasizing the transitions between segments. The variable sine function output features multiple transformations, highlighting changes in frequency and phase. Lastly, the convolution output visualizes the process between a unit step function and an exponential function, clearly showing the result of their combination. These visualizations collectively enhance the understanding of these fundamental signal processing concepts.

**CONCLUSION**

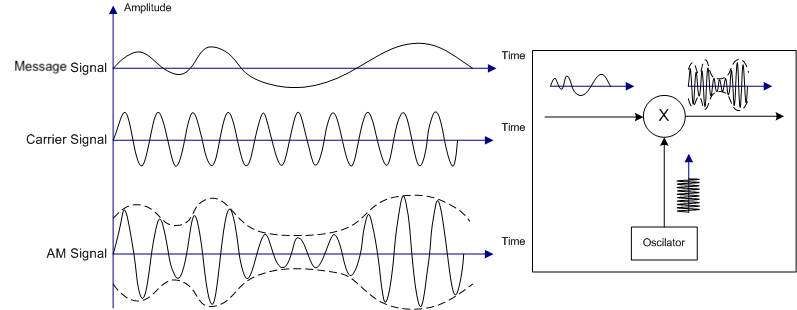
These visualizations offer insights into exponential growth, piecewise functions, variable sine waves, and convolution operations. They provide a clear understanding of how these signals behave and interact, which is crucial for analysis in signal processing and system design.

**LAB 3 Amplitude and Frequency Modulation**

**THEORY**

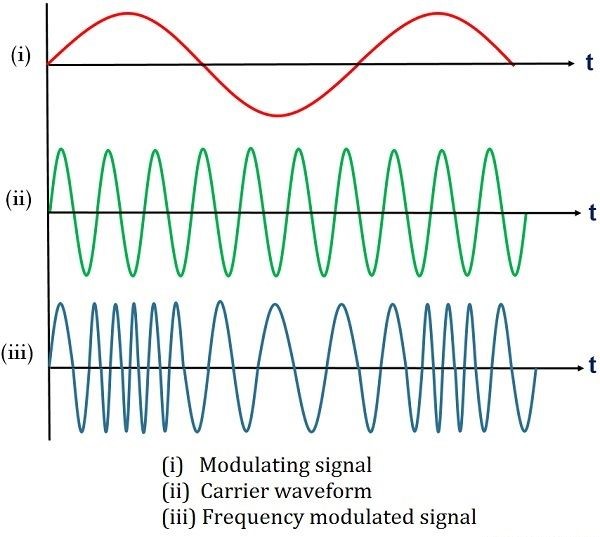
**Amplitude Modulation**

Amplitude modulation is a technique used to encode information in a carrier wave by varying its amplitude. In AM, the amplitude of the carrier signal is proportional to the message signal (modulating signal). This technique allows the transmission of information such as audio or data over radio waves. The key components of AM are the carrier signal, modulating signal, and the resulting modulated signal. The modulated signal has varying amplitude while maintaining a constant frequency, which enables the transmission of the modulating signal's information.



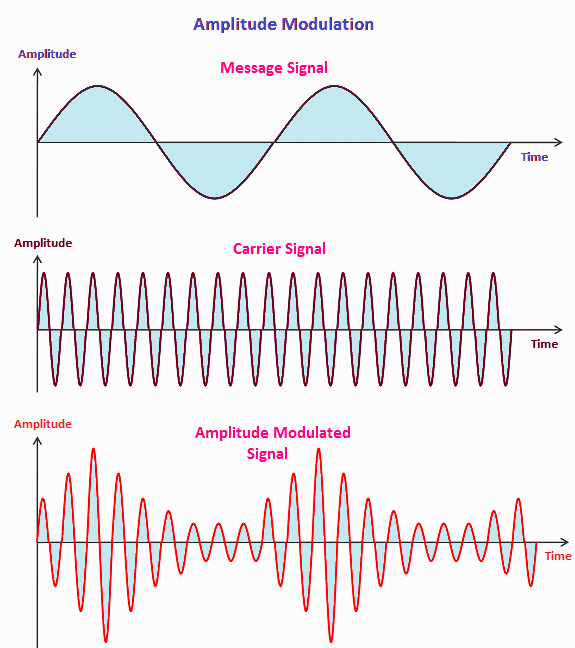
**Frequency Modulation**

Frequency modulation is a method of encoding information by varying the frequency of the carrier signal according to the amplitude of the modulating signal. In FM, the frequency of the carrier wave changes in proportion to the amplitude of the message signal. This results in a signal where the frequency deviation represents the information being transmitted. FM is commonly used in radio broadcasting and other communication systems due to its resilience to signal degradation and noise, providing clearer and more reliable transmission of audio and data.



**Amplitude Modulation vs Frequency Modulation**

Amplitude Modulation (AM) and Frequency Modulation (FM) are two techniques for encoding information onto a carrier signal. AM involves varying the amplitude of the carrier wave in proportion to the modulating signal while keeping its frequency constant, resulting in a signal that is more susceptible to noise since noise affects amplitude. In contrast, FM modulates the frequency of the carrier wave according to the amplitude of the modulating signal, maintaining a constant amplitude and offering better resistance to noise, though it requires a larger bandwidth. While AM is commonly used in radio broadcasting due to its simplicity and lower bandwidth requirement, FM provides superior sound quality and clarity, making it preferable for applications where noise immunity is crucial.

****

**CODES**

**1. Amplitude Modulation**

clc

clear all;

close all;

t=0:0.001:1;

fm=input('enter the message frequency:');%20

fc=input('enter the carrier frequency:');%200

mi=input('modulation index= ');%1

A=5;

Sm=A\*sin(2\*pi\*fm\*t);%message signal

Sc=A\*sin(2\*pi\*fc\*t);%carrier signal

Sam=(A+mi\*Sm).\*sin(2\*pi\*fc\*t);%Am signal

subplot(3,1,1);

plot(Sm);

xlim([0 1000]);

xlabel('time');

ylabel('amplitude');

title('plot for message signal (Deepak Thapa BCT29)');

subplot(3,1,2);

plot(Sc);

xlim([0 1000]);

xlabel('time');

ylabel('amplitude');

title('plot for carrier signal (Deepak Thapa BCT29)');

subplot(3,1,3);

plot(Sam);

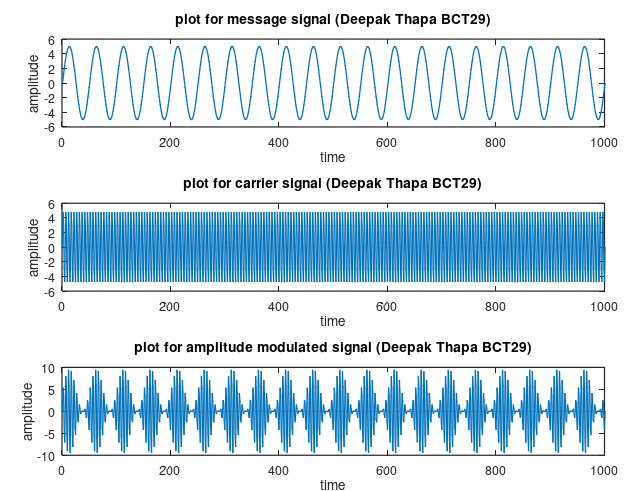
xlim([0 1000]);

xlabel('time');

ylabel('amplitude');

title('plot for amplitude modulated signal (Deepak Thapa BCT29)');

**Output:**



**2. Frequency Modulation**

clc

clear all;

close all;

t=0:0.001:1;

fm=input('enter the message frequency:');%10

fc=input('enter the carrier frequency:');%100

mi=input('modulation index= ')%5

Sm=sin(2\*pi\*fm\*t);%messag signal

Sc=sin(2\*pi\*fc\*t);%carrier signal

Sfm=sin(2\*pi\*fc\*t+(mi.\*sin(2\*pi\*fm\*t)));%fm wave

subplot(3,1,1);

plot(Sm);

xlim([0 1000]);

xlabel('time');

ylabel('amplitude');

title('plot for message signal (Deepak Thapa BCT29)');

subplot(3,1,2);

plot(Sc);

xlim([0 1000]);

xlabel('time');

ylabel('amplitude');

title('plot for carrier signal (Deepak Thapa BCT29)');

subplot(3,1,3);

plot(Sfm);

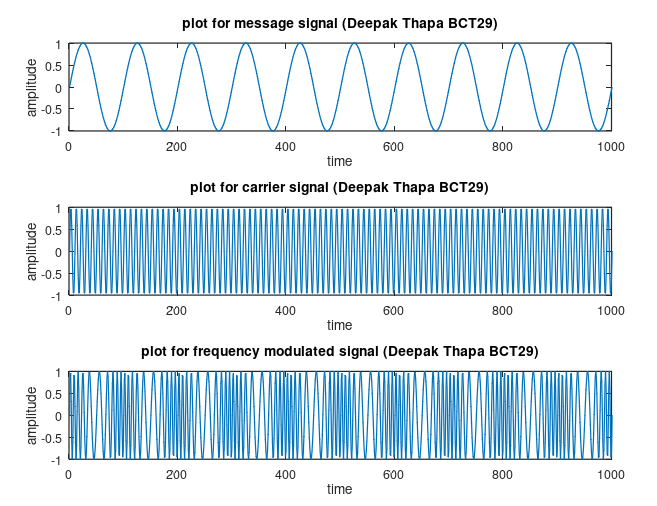
xlim([0 1000]);

xlabel('time');

ylabel('amplitude');

title('plot for frequency modulated signal (Deepak Thapa BCT29)');

**Output:**



**DISCUSSION**

The MATLAB outputs illustrate both amplitude and frequency modulation techniques. For amplitude modulation, the output includes plots of the message signal, carrier signal, and the modulated signal, clearly showing how the amplitude of the carrier wave varies in accordance with the message signal. In the case of frequency modulation, the output provides plots of the message signal, carrier signal, and the frequency-modulated signal, effectively demonstrating how the frequency of the carrier wave is adjusted based on the message signal. These visual representations collectively enhance the understanding of the principles and applications of amplitude and frequency modulation.

**CONCLUSION**

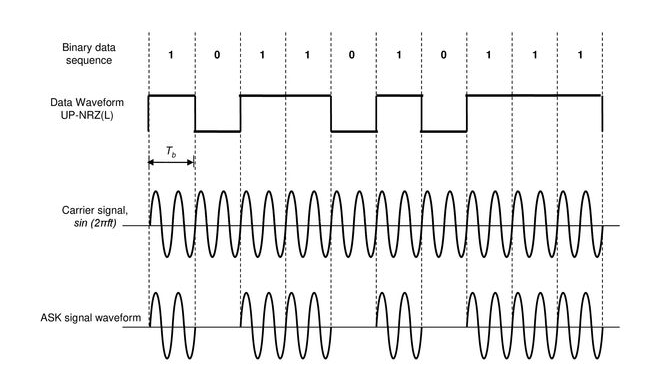
These visualizations effectively demonstrate the principles of amplitude and frequency modulation. They provide a clear view of how modulation affects carrier signals, helping in understanding signal processing techniques for communication systems.

**LAB 4 Amplitude Shift Keying, Frequency Shift Keying and Phase Shift Keying**

**THEORY**

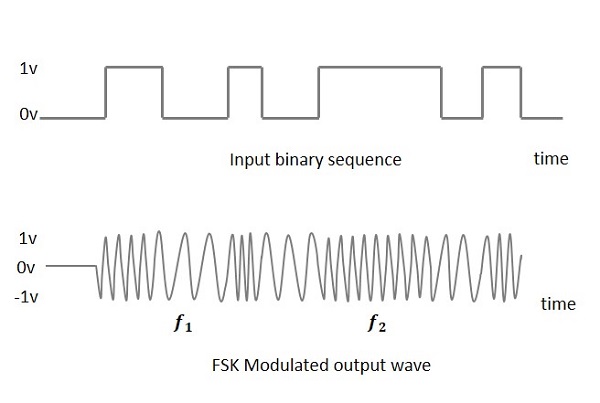
**Amplitude Shift Keying**

Amplitude Shift Keying (ASK) is a type of Amplitude Modulation, which represents the binary data in the form of variations in the amplitude of a signal. Any modulated signal has a high frequency carrier. The binary signal when ASK modulated, gives a zero value for Low input while it gives the carrier output for High input. The following figure represents ASK modulated waveform along with its input.

****

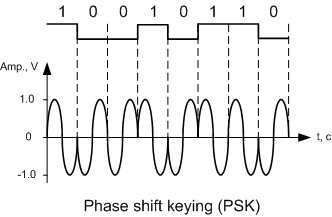
**Frequency Shift Keying**

Frequency Shift Keying (FSK) is a digital modulation method where the carrier signal's frequency changes based on variations in the digital signal. In FSK modulation, a high frequency output corresponds to a binary high input, while a low frequency output corresponds to a binary low input. These binary states are referred to as Mark and Space frequencies. The diagram below illustrates the waveform resulting from FSK modulation, alongside its corresponding input signal.



**Phase Shift Keying**

Phase Shift Keying (PSK) is a digital modulation method where the phase of the carrier signal is altered by adjusting the sine and cosine inputs at specific intervals. PSK is extensively utilized in applications such as wireless LANs, biometrics, contactless operations, RFID, and Bluetooth communications. The diagram below illustrates the modulated output waveform of PSK, corresponding to its input signal.



**CODES**

**1. ASK**

clc;

clear all;

close all;

t=0:0.001:1;

fp=input('enter the frequency of pulse/msg:');%10

fc=input('enter the carrier frequency:');%100

amp=input('enter amplitude of message and carrier signal: ');%4

m=(amp/2).\*square(2\*pi\*fp\*t)+(amp/2);

c=amp.\*sin(2\*pi\*fc\*t);

w=c.\*m;

subplot(3,1,1);

plot(t,m);

xlabel('time');

ylabel('amplitude');

title('plot for message signal (Deepak Thapa BCT29)');

subplot(3,1,2);

plot(t,c);

xlabel('time');

ylabel('amplitude');

title('plot for carrier signal (Deepak Thapa BCT29)');

subplot(3,1,3);

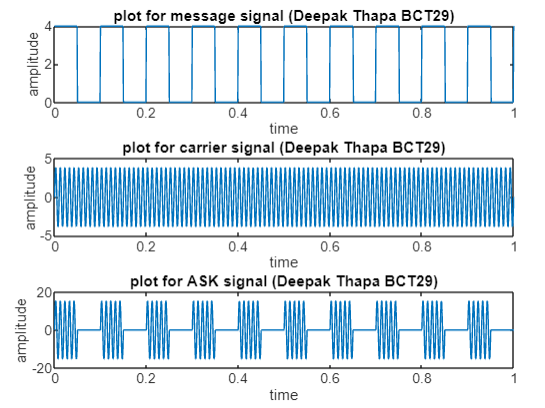
plot(t,w);

xlabel('time');

ylabel('amplitude');

title('plot for ASK signal (Deepak Thapa BCT29)');

**Output:**



**2. FSK**

clc

clear all

close all

fp=input("Enter the frequency of pulse message:");%5

fc1=input("Enter the carrier frequency 1:");%10

fc2=input("Enter the carrier frequency 2:");%30

amp=input("Enter the amplitude for both carrier and message:");%4

amp=amp/2;

t=0:0.001:1;

c1=amp.\*sin(2\*pi\*fc1\*t); %1st carrier

c2=amp.\*sin(2\*pi\*fc2\*t); %2nd carrier

m=amp.\*square(2\*pi\*fp\*t)+amp; %generating message pulse

for i=0:1000

if m(i+1)==0

mm(i+1)=c2(i+1);

else

mm(i+1)=c1(i+1);

end

end

subplot(411)

plot(t,m)

xlabel('time');

ylabel('amplitude');

title('message signal (Deepak Thapa BCT29)')

subplot(412)

plot(t,c1)

xlabel('time');

ylabel('amplitude');

title('carrier frequncy 1 (Deepak Thapa BCT29)')

subplot(413)

plot(t,c2)

xlabel('time');

ylabel('amplitude');

title('carrier frequency 2 (Deepak Thapa BCT29)')

subplot(414)

plot(t,mm)

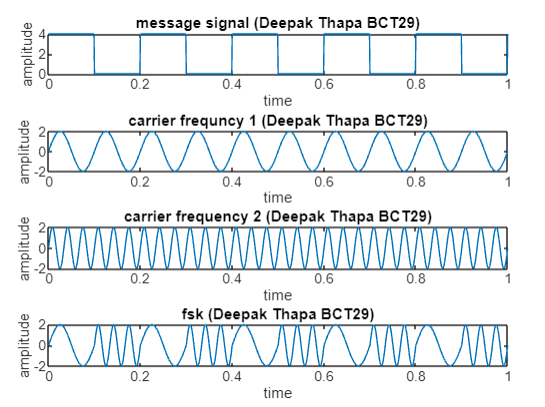
xlabel('time');

ylabel('amplitude');

title(['fsk 5' ...

'(Deepak Thapa BCT29)'])

**Output:**



**3.PSK**

clc;

clear all;

close all;

t=0:0.001:1;

fm=input('enter the frequency of pulse/msg:');%10

fc=input('enter the carrier frequency:');%60

amp=input('enter amplitude of message and carrier signal: ');%3

c=amp.\*sin(2\*pi\*fc\*t);

m=square(2\*pi\*fm\*t);

x=c.\*m;

subplot(3,1,1);

plot(t,m);

xlabel('time');

ylabel('amplitude');

title('Plot for message signal (Deepak Thapa BCT29)');

subplot(3,1,2);

plot(t,c);

xlabel('time');

ylabel('amplitude');

title('Plot for carrier signal (Deepak Thapa BCT29)');

subplot(3,1,3);

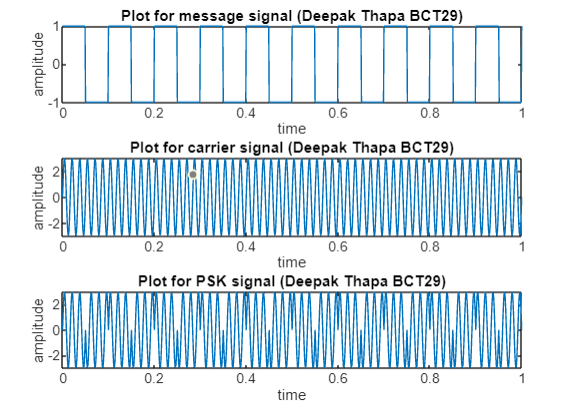
plot(t,x);

xlabel('time');

ylabel('amplitude');

title('Plot for PSK signal (Deepak Thapa BCT29)');

**OUTPUT**



**DISCUSSION**

The MATLAB outputs provide an insightful exploration of various modulation techniques, specifically Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), and Phase Shift Keying (PSK). The ASK output effectively demonstrates how the amplitude of a carrier signal can be modulated by a rectangular message signal. In the FSK output, the modulation is achieved by toggling between two distinct carrier frequencies in response to the message signal, thereby altering the frequency. Lastly, the PSK output showcases phase modulation by adjusting the phase of the carrier signal based on a square wave message signal. These outputs collectively highlight the fundamental principles and applications of each modulation technique.

**CONCLUSION**

These output signals provide insights into ASK, FSK, and PSK modulation techniques, highlighting their effects on carrier signals. They offer a practical understanding of how different modulation methods influence signal properties, which is essential for communication system design and analysis.