Signals Systems and Communication Lab

Laboratory report submitted for the partial fulfillment
of the requirements for the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

by

Mohit Akhouri - 19ucc023

Course Coordinators
Dr. Navneet Upadhyay , Dr. Akash Gupta

Department of Electronics and Communication Engineering
The LNM Institute of Information Technology, Jaipur

February 2021

Contents

List of Figures

;

Chapter *1*

Experiment - 1

1.1 Aim of the experiment

1. To get familiarity with basic commands in MATLAB.
2. To explore the connection between system impulse response and the solution of linear ordinary constant coefficient differential equation.
3. To understand and implement convolution routine for discrete time finite length sequences.

1.2 Software Used

MAT LAB

1.3 Theory

1.3.1 About Matlab :

MATLAB (Matrix Laboratory) is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Figure 1.1 MATLAB Icon

1.3.2 About the function plot(y,x) :

plot(y,x) function takes as argument two sequences y and x and plots them as a graph. We can use other options also like **grid on** to display the grid , **title** to give title to graph and for giving labels to X and Y axis , use **xlabel** and **ylabel** functions.

1.3.3 About Discrete Convolution :

Convolution, one of the most important concepts in electrical engineering, can be used to determine the output a system produces for a given input signal. It can be shown that a linear time invariant system is completely characterized by its impulse response. The sifting property of the discrete time impulse function tells us that the input signal to a system can be represented as a sum of scaled and shifted unit impulses. Thus, by linearity, it would seem reasonable to compute of the output signal as the sum of scaled and shifted unit impulse responses. That is exactly what the operation of convolution accomplishes. Hence, convolution can be used to determine a linear time invariant system's output from knowledge of the input and the impulse response.
The formula for Convolution of two infinite length sequences x and h is :

$$y(m) = \sum_{k=-\infty}^{\infty} x(k).h(m-k) \tag{1.1}$$

In this experiment, we have been given two finite length sequences x and h and we have to write a myconv(x,h) function to calculate the convolution of x and h. So the equation 1.1 transforms to the below equation :

$$y(m) = \sum_{k=1}^{n_x} x(k).h(m-k+1) \tag{1.2}$$

where $1 <= m <= ny$ and **nx** and **ny** are length of sequences x and output sequence y and **nh** is length of sequence h.
$ny = nx + nh - 1$
The above equation is implemented in MATLAB through the concept of **padding with zeros** and **Looping through FOR construct**.

1.4 Code and Results

1.4.1 Exercise 1 : Plotting the signals

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% Code for generating plot of x1(t) = (e^-t)cos(2*pi*t)

t = 0 : 0.001 : 2;
a = exp(-t);
b = cos(2*pi*t);
x1 = a.*b;
plot(t,x1);
xlabel('time(t) -> ');
ylabel('x_{1}(t) -> ');
title('Mohit Akhouri-19ucc023 - Exercise1\_a','x_{1}(t) = e^{-
t}cos(2\pit)');
grid on;
```

Figure 1.2 Code for Exercise 1a - Plotting x_1(t)

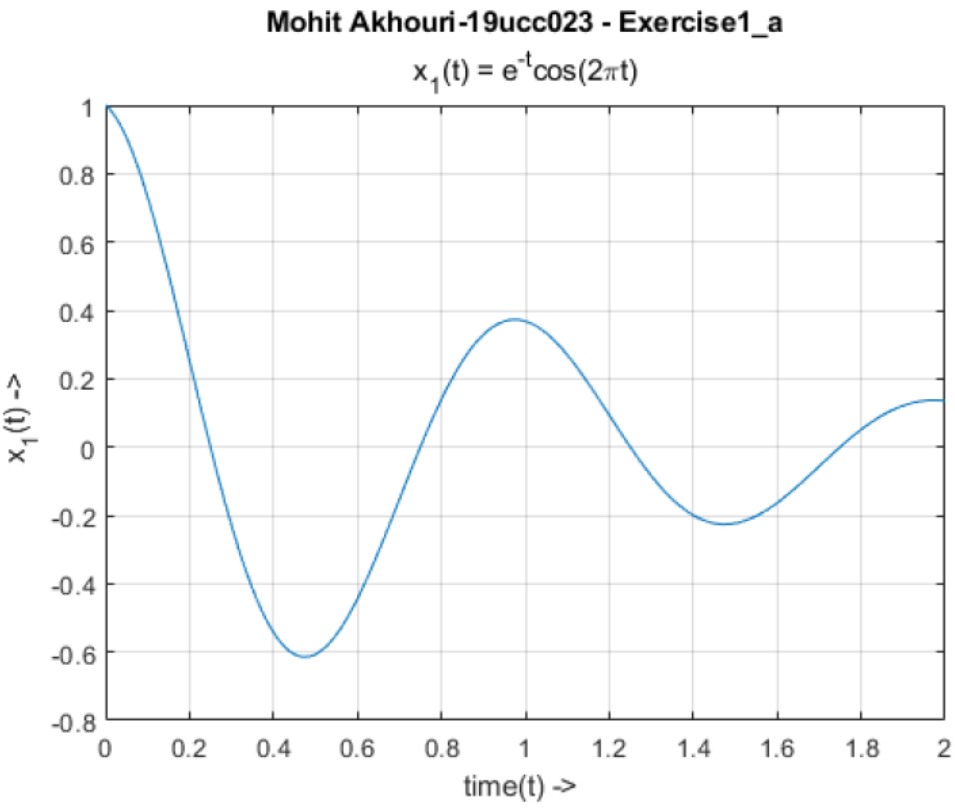


Figure 1.3 Graph for Exercise 1a - Plotting x_1(t)

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% Code for generating plot of x2(t) = 1 + 1.5cos(2*pi*t) -
 0.6cos(4*pi*t)

t = 0 : 0.001 : 2;
x2 = 1 + 1.5*cos(2*pi*t) - 0.6*cos(4*pi*t);
plot(t,x2);
xlabel('time(t) -> ');
ylabel('x_{2}(t) -> ');
title('Mohit Akhouri-19ucc023 - Exercise1\_b','x_{2}(t) = 1 +
 1.5cos(2\pit) - 0.6cos(4\pit)');
grid on;
```

Figure 1.4 Code for Exercise 1b - Plotting x_2(t)

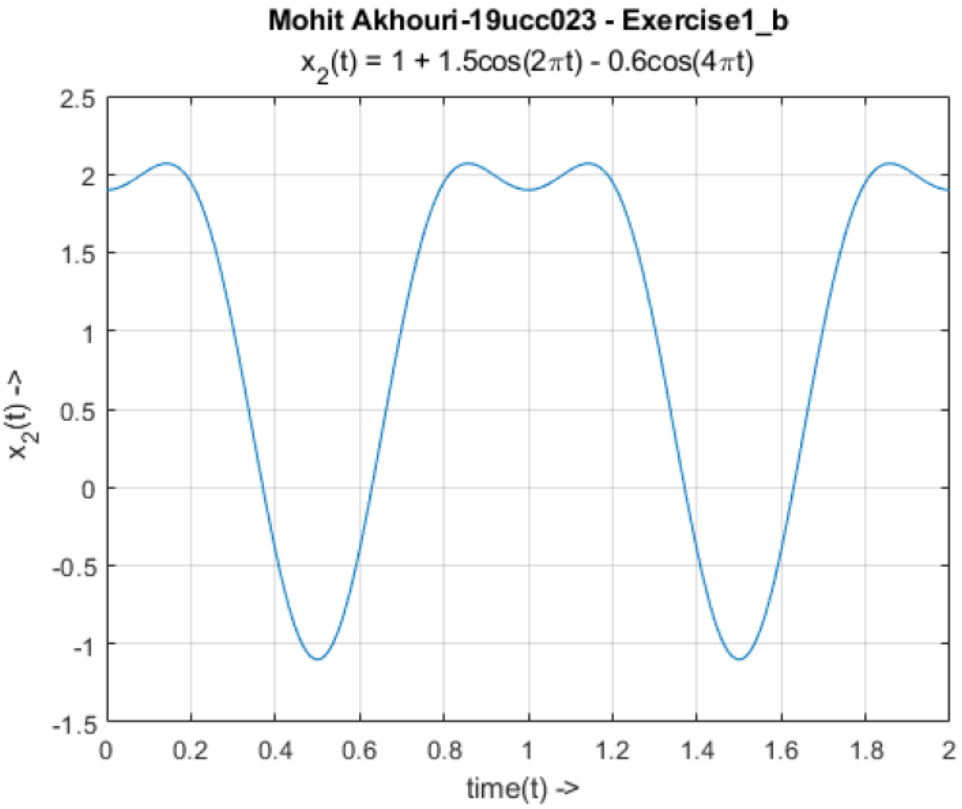


Figure 1.5 Graph for Exercise 1b - Plotting x_2(t)

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% Code for generating plot of x3(t) = |cos(2*pi*t)|

t = 0 : 0.001 : 2;
x3 = abs(cos(2*pi*t));
plot(t,x3);
xlabel('time(t) -> ');
ylabel('x_{3}(t) -> ');
title('Mohit Akhouri-19ucc023 - Exercise1\_c','x_{3}(t) = |
cos(2\pit)|');
grid on;
```

Figure 1.6 Code for Exercise 1c - Plotting x_3(t)

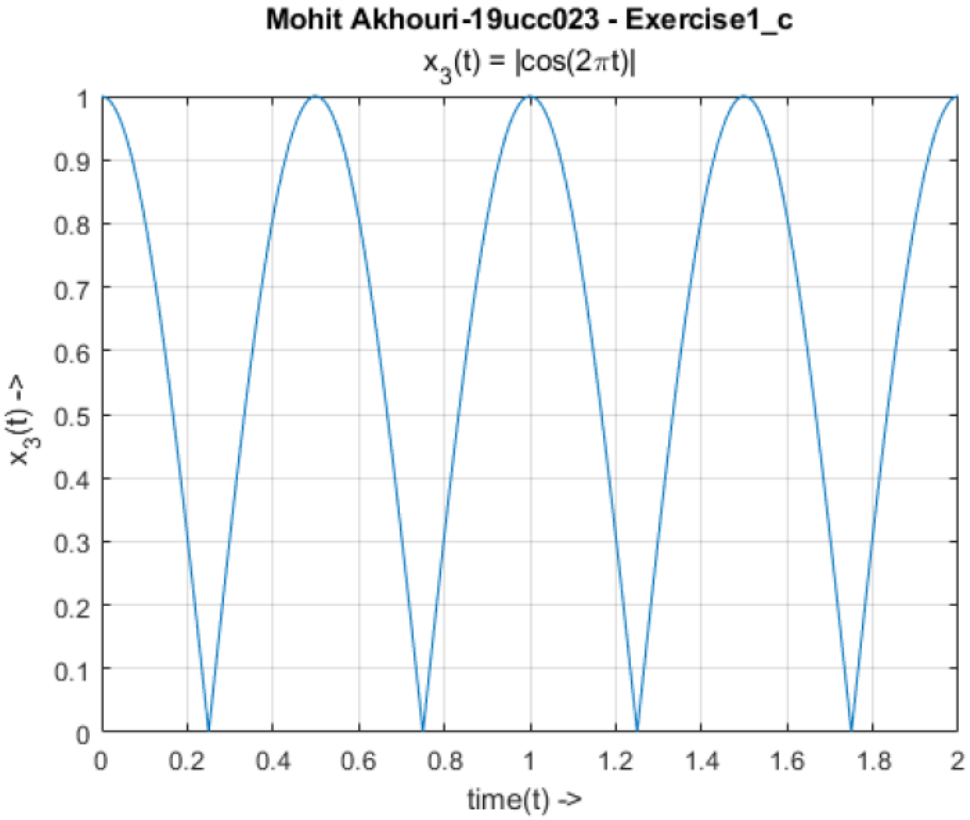


Figure 1.7 Graph for Exercise 1c - Plotting x_3(t)

1.4.2 Exercise 2 : Computing convolution of two finite length sequences

```
function  myconv(x,h)
% Summary of the function myconv(x,h) is :
%   This function takes as input two finite length sequences 'x' and
 'h'
%   and calculate the convulution of x and h

% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

nx = length(x); % length of sequence x
nh = length(h); % length of sequence h
ny = nx+nh-1; % length of output convulated sequence y
y = zeros(1,ny); % output convulated sequence y
h = [h zeros(1,ny-nh+1)]; % padding h with extra zeros

for i=1:ny
    sum=0; %to calculate sum at each iteration
    for j=1:nx
        if (i-j+1>0)
            sum = sum + (x(j)*h(i-j+1));
        end
    end
    y(i)=sum;
end
stem(y);
xlabel('time(t) ->');
ylabel('x(n) conv h(n) ->');
title('Mohit Akhouri-19ucc023 - Exercise2','Discrete convolution');
grid on;
end
```

Published with MATLAB® R2020b

Figure 1.8 Function myconv(x,h) designed for Experiment 2

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% Code for using myconv(x,h) function to calculate the convulution
% of two finite length sequences

x = [1 1 1 1 -1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1];
h1 = [1 1];
myconv(x,h1);
```

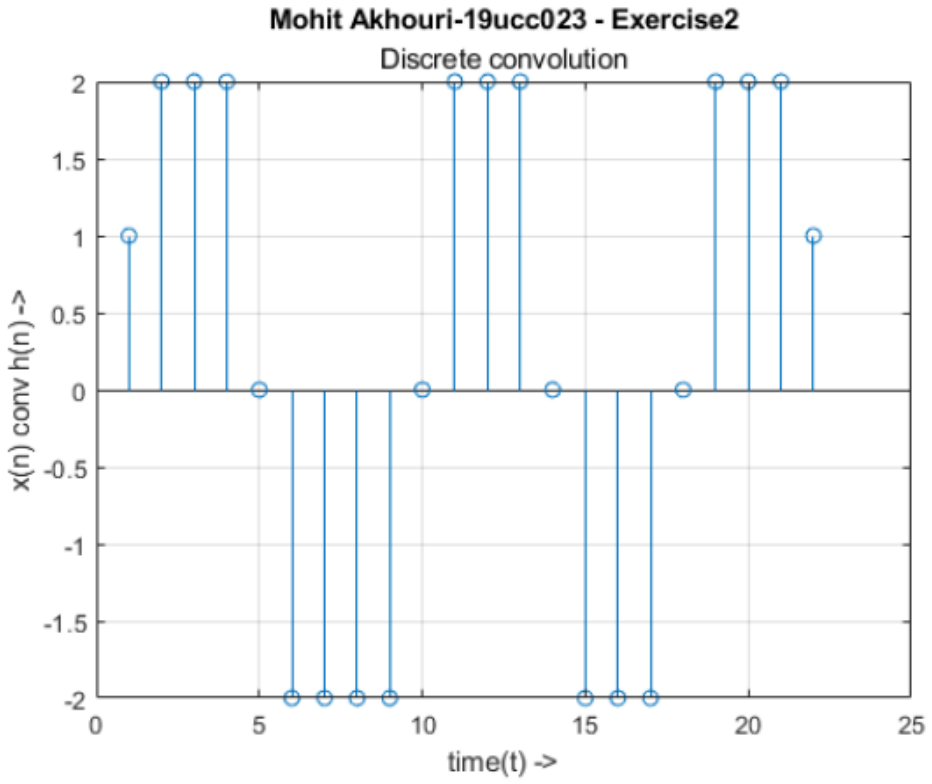


Figure 1.9 Main code and graph for convolution x and h1

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% Code for using myconv(x,h) function to calculate the convulution
% of two finite length sequences

x = [1 1 1 1 -1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1];
h2 = [1 -1];
myconv(x,h2);
```

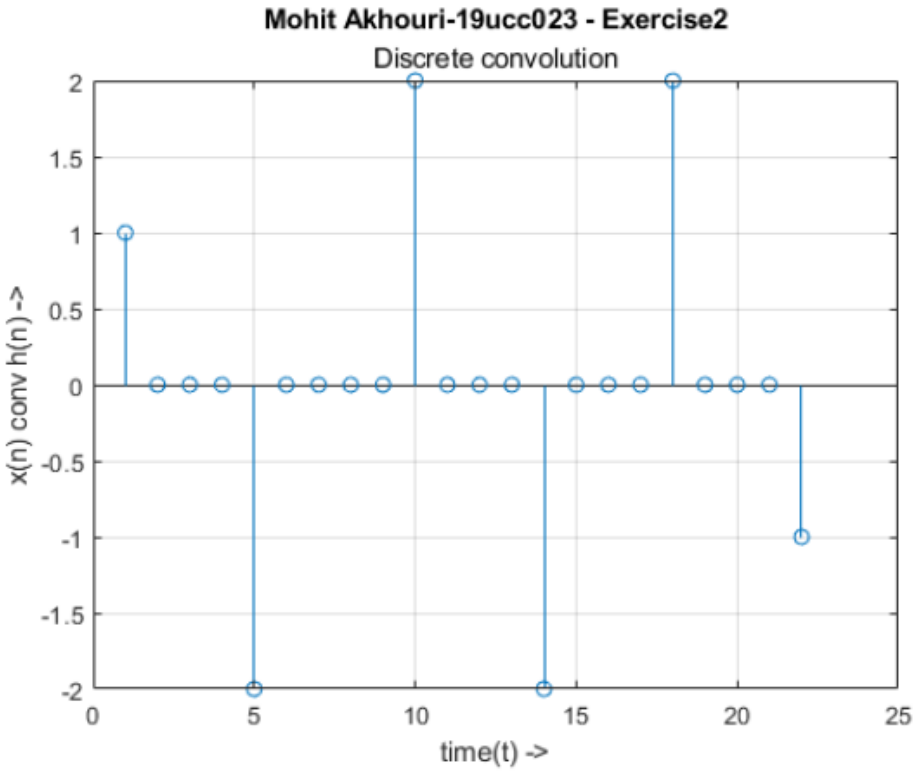


Figure 1.10 Main code and graph for convolution x and h2

1.4.3 Exercise 3 : Solving the Difference equation of RC Filter

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% MATLAB code to solve the difference equation with
% input specified as x(t)

RC=1; %defining time constant = 1
T=0.001;
t=0:T:2-T; %defining range for t
a=1/(1+(RC/T)); %defining 'a'
b=-1/(1+(T/RC)); %defining 'b'
x=(t>=0)+(-1*(t>=1));
y=zeros(1,2000); %filling y with zeros
for k=1:2000 %implementing the algorithm using for loop
    if k==1
        y(k)=a*x(1);
    else
        y(k)=a*x(k)-b*y(k-1);
    end
end
plot(t,y);
ylabel('y(k) ->');
xlabel('time(t) ->');
title('Mohit Akhouri-19ucc023 - Exercise3','Solving Difference
 Equation');
grid on;
```

Figure 1.11 Code for Exercise 3

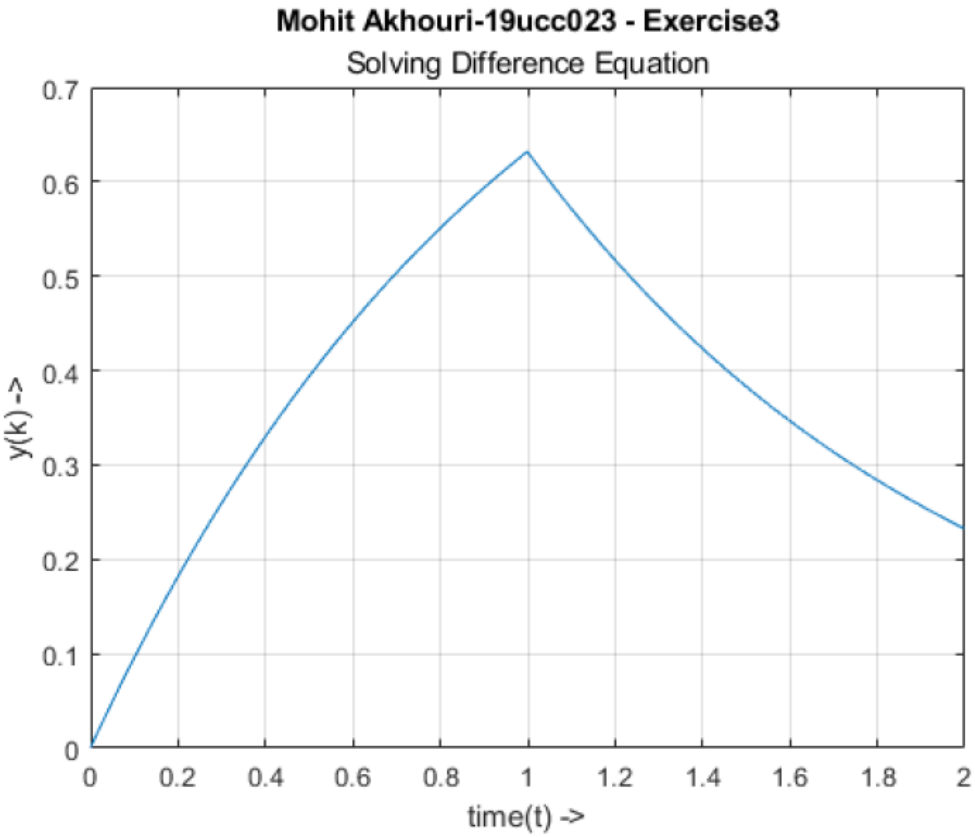


Figure 1.12 Graph of Exercise 3

1.4.4 Exercise 4 : Calculation of Output of RC Filter

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% Calculation output of LTI system

RC=1; %defining time constant = 1
T=0.001;
t=0:T:2-T; %defining range for t
a=1/(1+(RC/T)); %defining 'a'
b=-1/(1+(T/RC)); %defining 'b'
x=[1 zeros(1,1999)]; %defining impulse response del(t)
y=zeros(1,2000); %initializing y
for k=1:2000 %algorithm to find convolution
    if k==1
        y(k)=a*x(1);
    else
        y(k)=a*x(k)-b*y(k-1);
    end
end
%using conv function on h and x
h=y;
x=(t>=0)+(-1*(t>=1));
y1=conv(x,h);
stem(y1);
xlabel('time(t) ->');
ylabel('x(n) conv h(n) ->');
title('Mohit Akhouri-19ucc023 - Exercise4','Output of the LTI system
 (RC Filter)');
grid on;
```

Figure 1.13 Code for Exercise 4

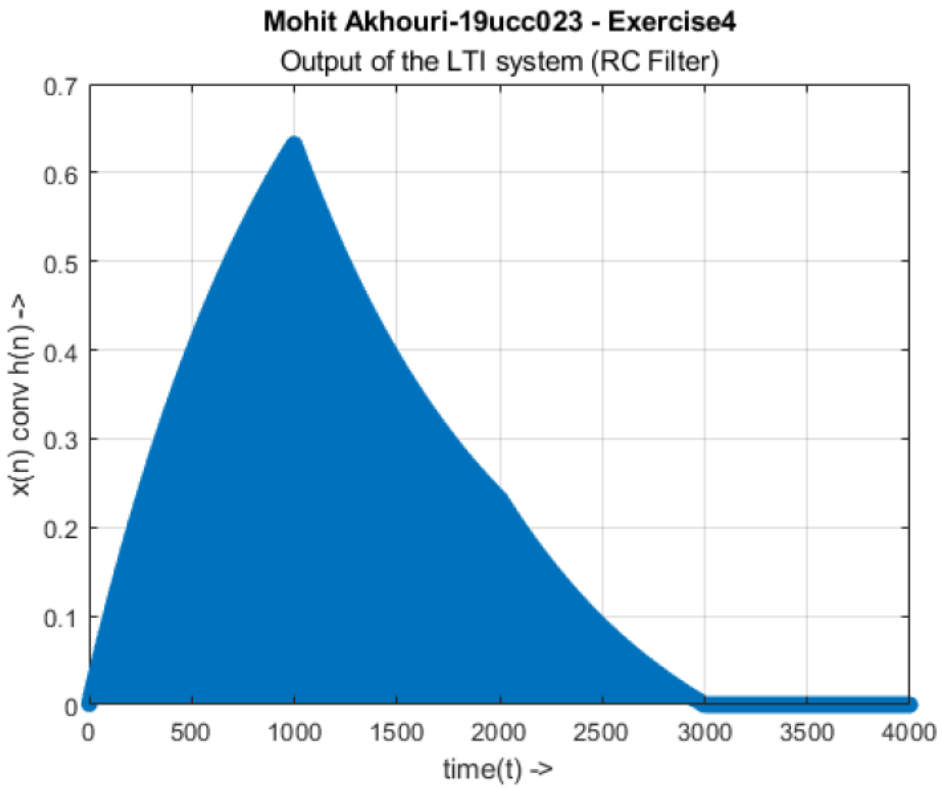


Figure 1.14 Graph of Exercise 4

1.5 Conclusion

In this experiment, We have learnt MATLAB basics and tried to implement basic plots in MATLAB. We have also implemented myconv(x,h) to calculate convolution of two finite length sequences using **for loop** and **concept of padding with zeros** . In exercise 3 and 4, We have analysed the LTI systems and implemented a code to solve difference equation of RC Filter and finding output of RC Filter using convolution of impulse signal and input signal.

Chapter *2*

Experiment - 2

2.1 Aim of the experiment

To compute and plot the Fourier spectra for the aperiodic signals

2.2 Software Used

MATLAB

2.3 Theory

2.3.1 About Aperiodic Signals :

A signal that does not repeat itself after a specific interval of time is called an aperiodic signal. By applying a limiting process, the signal can be expressed as a continuous sum (or integral) of everlasting exponentials. These signals are analysed by means of the **Fourier Transform**.

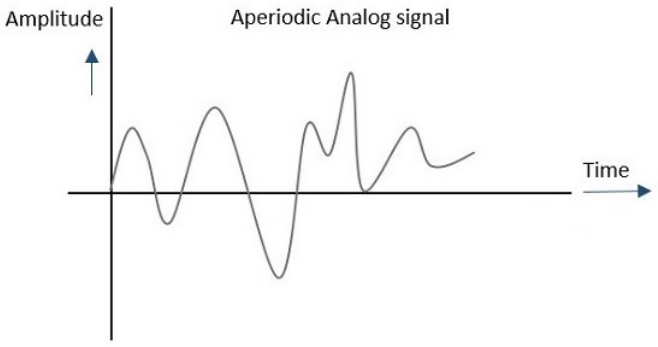


Figure 2.1 Aperiodic Analog Signal

2.3.2 About Fourier Series :

Jean B. Joseph Fourier was a French mathematician who proposed an idea that any periodic signal can be represented by addition of scaled basis signals of different frequencies(harmonics). This idea was later termed as **Fourier Series Representation**. Basically, Fourier Series is an expansion of a periodic function terms of an infinite sum of sines and cosines. It makes use of **orthagonality** relationships of sine and cosine functions. The computation and study of Fourier series is known as harmonic analysis and is extremely useful as a way to break up an arbitrary periodic function into a set of simple terms that can be plugged in, solved individually, and then recombined to obtain the solution to the original problem or an approximation to it to whatever accuracy is desired or practical. The fourier series of a periodic function f(x) of period T is :

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \frac{cos(2\pi kx)}{T} + \sum_{k=1}^{\infty} b_k \frac{sin(2\pi kx)}{T} \tag{2.1}$$

for some set of Fourier coefficients a_k and b_k defined by the integrals :

$$a_k = \frac{2}{T} \int_0^T f(x) \frac{cos(2\pi kx)}{T} dx \tag{2.2}$$

$$b_k = \frac{2}{T} \int_0^T f(x) \frac{sin(2\pi kx)}{T} dx \tag{2.3}$$

2.3.3 About Fourier Transform :

The Fourier transform expresses a function of time (a signal) as a function of frequency.The Fourier transform of a function of time itself is a complex-valued function of frequency, whose complex modulus represents the amount of that frequency present in the original function, and whose complex argument is the phase offset of the basic sinusoid in that frequency.The Fourier transform is called the frequency domain representation of the original signal.For many functions of practical interest one can define an operation that reverses this: the inverse Fourier transformation, also called Fourier synthesis, of a frequency domain representation combines the contributions of all the different frequencies to recover the original function of time. Joseph Fourier introduced the transform in his study of heat transfer, where Gaussian functions appear as solutions of the heat equation. The fourier transform of an aperiodic continuous time signal x(t) is :

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \tag{2.4}$$

Let us consider the samples of X() at regular intervals of ω_0. If X_r is the r^{th} sample the from equation 2.4 we obtain :

$$X_r = \sum_{k=0}^{N_0-1} x_k e^{-jr\Omega_0 k} \tag{2.5}$$

where $x_k = T_s x(kT_s)$,$X_r = X(r\omega_0)$ and $\Omega_0 = \omega_0 T_s$.

2.4 Code and Results

2.4.1 Exercise 1 : Creating function X = mydft(x,t_0,t_s) to generate DFT of x

```
function mydft(x,to,ts)
% This function takes three arguments :
% a sequence 'x', sampling interval 'ts' and to
% This function will calculate the discrete fourier transform of
 signal 'x'

% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

%initializing N and fs
N = to/ts;
fs = 1/ts;
%creating a empty array to store the result using zeros function
X = zeros(1,N);
%loop algorithm to calculate the DFT of signal x1(t)
for k=1:N
    sum = 0;
    for n=1:N
        sum = sum + (x(n).*(exp(-1i*2*pi*(k-1)*(n-1)/N)));
    end
    X(k) = sum;
end
f = linspace(-fs,fs,N); %defining frequency from -fs to fs
%plotting the graphs (magnitude plot and phase plot)
subplot(2,2,1);
plot(f,abs(fftshift(X)));
xlabel('Frequency (Hz) ->');
ylabel('Magnitude ->');
title('Mohit Akhouri-19ucc023','Magnitude Plot using function mydft');
grid on;
subplot(2,2,3);
plot(f,angle(X));
xlabel('Frequency (Hz) ->');
ylabel('Phase angle ->');
title('Mohit Akhouri-19ucc023','Phase Plot using function mydft');
grid on;
```

Published with MATLAB® R2020b

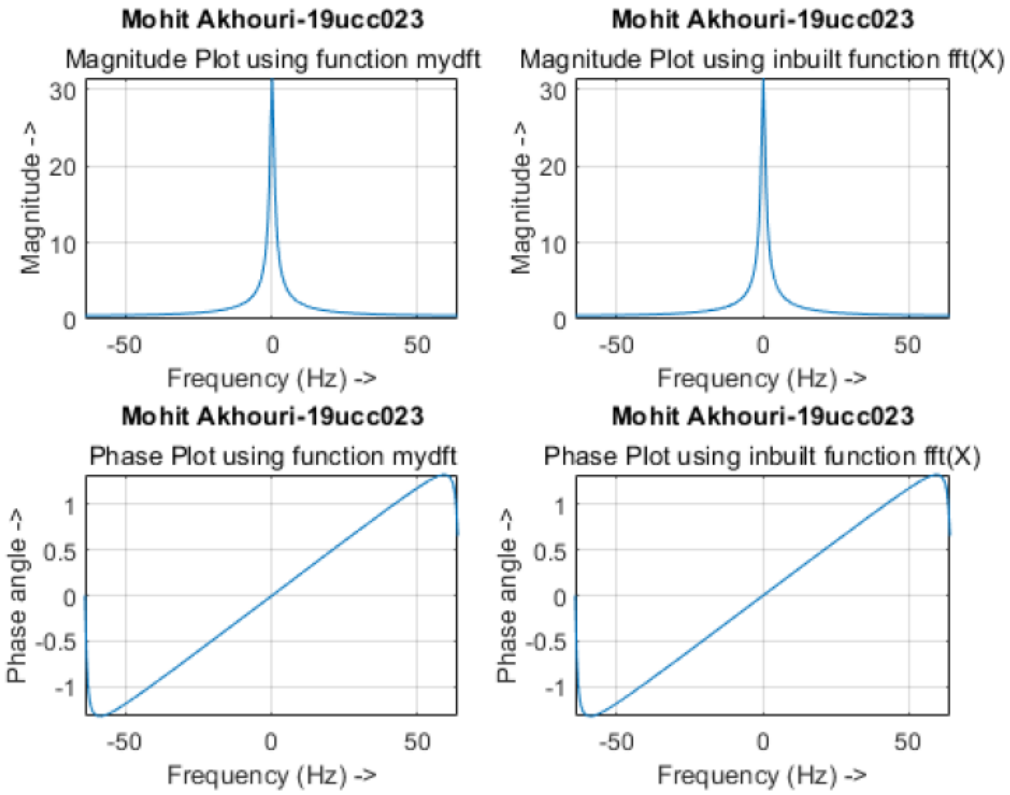
Figure 2.2 Code for the function mydft designed for finding DFT of x

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% Code for calling mydft(x,to,ts) to calculate fourier transform of x

%defining constants to,ts,fs and N
to = 4;
ts = 1/64;
fs = 1/ts;
N = to/ts;
%initializing frequency variable from -fs to fs using linspace
f = linspace(-fs,fs,N);
t = ts:ts:to;
ut = t>=0; %defining u(t) function
xt = (exp(-2*t)).*ut; %defining x1(t)
mydft(xt,to,ts);%calling function mydft to calculate dft of x1(t)
%plotting the graphs (magnitude plot and phase plot)
subplot(2,2,2);
plot(f,abs(fftshift(fft(xt))));
xlabel('Frequency (Hz) ->');
ylabel('Magnitude ->');
title('Mohit Akhouri-19ucc023','Magnitude Plot using inbuilt function
 fft(X)');
grid on;
subplot(2,2,4);
plot(f,angle(fft(xt)));
xlabel('Frequency (Hz) ->');
ylabel('Phase angle ->');
title('Mohit Akhouri-19ucc023','Phase Plot using inbuilt function
 fft(X)');
grid on;
```

Figure 2.3 Code for calculating DFT when T_0 = 4 sec and T_s = 1/64 sec



Published with MATLAB® R2020b

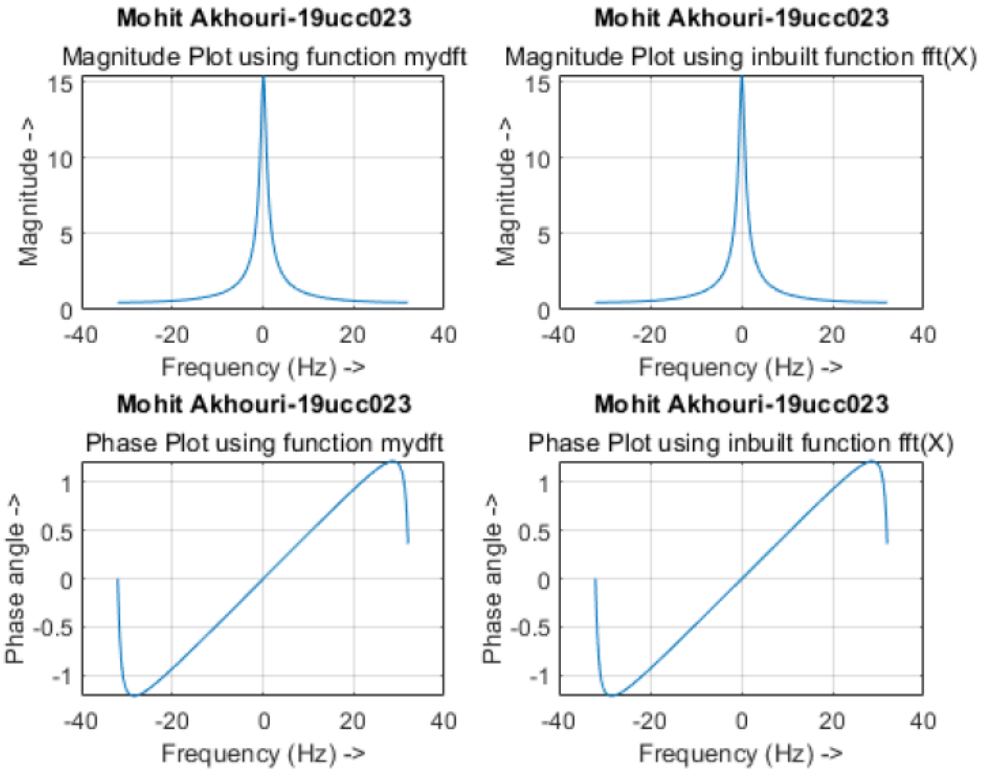
Figure 2.4 Graph of calculated DFT when T_0 = 4 sec and T_s = 1/64 sec

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% Code for calling mydft(x,to,ts) to calculate fourier transform of x

%defining constants to,ts,fs and N
to = 8;
ts = 1/32;
fs = 1/ts;
N = to/ts;
%initializing frequency variable from -fs to fs using linspace
f = linspace(-fs,fs,N);
t = ts:ts:to;
ut = t>=0; %defining u(t) function
xt = (exp(-2*t)).*ut; %defining x1(t)
mydft(xt,to,ts); %calling function mydft to calculate dft of x1(t)
%plotting the graphs (magnitude plot and phase plot)
subplot(2,2,2);
plot(f,abs(fftshift(fft(xt))));
xlabel('Frequency (Hz) ->');
ylabel('Magnitude ->');
title('Mohit Akhouri-19ucc023','Magnitude Plot using inbuilt function
 fft(X)');
grid on;
subplot(2,2,4);
plot(f,angle(fft(xt)));
xlabel('Frequency (Hz) ->');
ylabel('Phase angle ->');
title('Mohit Akhouri-19ucc023','Phase Plot using inbuilt function
 fft(X)');
grid on;
```

Figure 2.5 Code for calculating DFT when T_0 = 8 sec and T_s = 1/32 sec



Published with MATLAB® R2020b

Figure 2.6 Graph of calculated DFT when T_0 = 8 sec and T_s = 1/32 sec

2.5 Conclusion

In this experiment , we learnt the concepts of Aperiodic Signals, Fourier series representation of signals and Fourier transform concept. We tried to implement the **mydft** function to calculate the DFT of a signal 'x'. We implemented many coding concepts of MATLAB like **for loops**, **subplot and plot** and **linspace**. At last we compared the DFT calculated by our function mydft and with the inbuilt function **fft(X)** and understood the concept of Discrete fourier transform .

Chapter *3*

Experiment - 3

3.1 Aim of the experiment

1. Implementation of discrete Fourier transform (DFT) and inverse DFT (IDFT) algorithm

2. Implementation of autocorrelation and cross correlation algorithm

3.2 Software Used

MATLAB

3.3 Theory

3.3.1 About Discrete Fourier transform (DFT) :

In mathematics, the discrete Fourier transform (DFT) converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency. The interval at which the DTFT is sampled is the reciprocal of the duration of the input sequence. It can be said to convert the sampled function from its original domain to the frequency domain. The sequence of N complex numbers x_0, x_1, x_2,..... , x_{N-1} is transformed into an N-periodic sequence of complex numbers:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}, k \in \mathbb{Z} \tag{3.1}$$

3.3.2 About Inverse Discrete Fourier transform (IDFT) :

An inverse DFT is a Fourier series, using the DTFT samples as coefficients of complex sinusoids at the corresponding DTFT frequencies. It has the same sample-values as the original input sequence. The inverse Fourier transform maps the signal back from the frequency domain into the time domain.

The perfect invertibility of the Fourier transform is an important property for building filters which remove noise or particular components of a signals spectrum. The IDFT is given by :

$$X[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[k] e^{j2\pi kn/N}, n \in \mathbb{Z} \tag{3.2}$$

3.3.3 About Correlation :

Correlation is a measure of similarity between two signals. The relationship between two signals indicates whether one depends on the other, both depend on some common phenomenon, or they are independent. Correlation function indicates how correlated two signals are as a function of how much one of them is shifted in time. There are two types of correlation :

1. Auto correlation
2. Cross correlation

3.3.3.1 Auto Correlation function (ACF) :

It is defined as correlation of a signal with itself. Auto correlation function is a measure of similarity between a signal its time delayed version. Consider a random process x(t) (continuous time) ,Its autocorrelation is computed as :

$$R_{xx}(\tau) = \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} x(t) x(t+\tau) dt \tag{3.3}$$

For sampled signal with N samples, ACF is defined as :

$$R_{xx}(m) = \frac{1}{N} \sum_{n=1}^{N-m+1} x[n] x[n+m-1], m = 1, 2, ...N+1 \tag{3.4}$$

3.3.3.2 Cross Correlation function (CCF) :

Cross correlation is the measure of similarity between two different signals. For two wide sense stationary (WSS) processes x(t) and y(t), the CCF is defined as :

$$R_{xy}(\tau) = \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} x(t) y(t+\tau) dt \tag{3.5}$$

For sampled signal with N samples, CCF is defined as :

$$R_{xy}(m) = \frac{1}{N} \sum_{n=1}^{N-m+1} x[n] y[n+m-1], m = 1, 2, ...N+1 \tag{3.6}$$

3.4 Code and Results

3.4.1 Exercise 1(a) : Creating myDFT (x,N) function to compute DFT of signal x(t) :

```
function [X] = myDFT(x,N)
% This function will calculate the DFT of function 'x' with total
 samples N

% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

X=zeros(1,N); %initializing output X
% main loop algorithm for calculating DFT of 'x' starts here
for k=1:N
    sum=0;
    for n=1:N
        sum=sum+(x(n).*(exp(-1j*2*pi*(k-1)*(n-1)/N)));
    end
    X(k)=sum;
end
end
```

Published with MATLAB® R2020b

Figure 3.1 Code for the function myDFT designed for finding DFT of signal x(t)

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% This is the code to perform DFT for
 x(t)=cos(2000*pi*t)+cos(800*pi*t)
% by calling function myDFT(x,N)

% initializing constants fs,N,ts and to
fs = 8000;
N = 128;
ts = 1/fs;
to = N*ts;
t = ts:ts:to; % defining range from 'ts' to 'to'
f = linspace(-fs,fs,N); % defining range from '-fs' to 'fs'
x = cos(2000*pi*t) + cos(800*pi*t); % defining x(t)
subplot(3,1,1);
plot(t,x);
xlabel('time(t) ->');
ylabel('x(t) ->');
title('x(t) = cos(2000\pit) + cos(800\pit)');
grid on;

X = myDFT(x,N); % calculating DFT of 'x' using myDFT
mag = fftshift(abs(X));
ang = angle(X);

subplot(3,1,2);
plot(f,mag);
xlabel('frequency(Hz) ->');
ylabel('Magnitude ->');
title('Magnitude vs. Frequency Plot of X(k)');
grid on;

subplot(3,1,3);
plot(f,ang);
xlabel('frequency(Hz) ->');
ylabel('Phase angle ->');
title('Phase vs. Frequency Plot of X(k)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri - Exercise 1a');
```

Figure 3.2 Code for calculating DFT of signal x(t) with N=128 and F_s = 8000 Hz

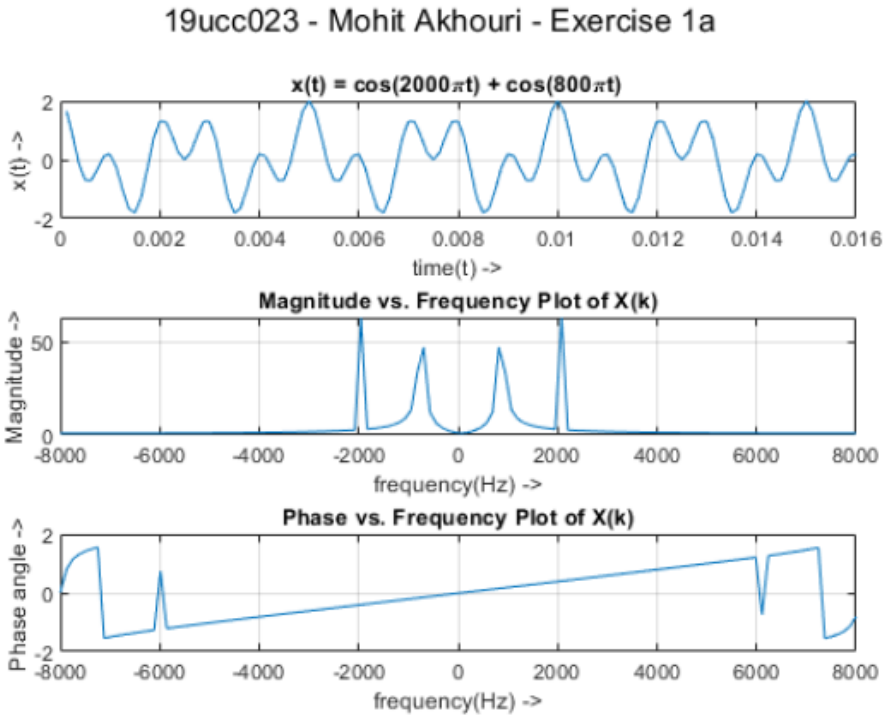


Figure 3.3 Graph of calculated DFT of signal x(t) with N=128 and F_s = 8000 Hz

3.4.2 Exercise 1(b) : Creating myIDFT (X,N) to calculate IDFT of signal x_1[n] = [1 1 1 1] :

```
function [x] = myIDFT(X,N)
% This function will calculate the inverse discrete fourier transform
% (IDFT) of X with total samples N

% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

x=zeros(1,N); % initializing output
% main loop algorithm to calculate IDFT
for n=1:N
    sum=0;
    for k=1:N
        sum=sum+(X(k).*(exp(1j*2*pi*(k-1)*(n-1)/N)));
    end
    sum=sum/N;
    x(n)=sum;
end
x=abs(x);
end
```

Published with MATLAB® R2020b

Figure 3.4 Code for the function myIDFT designed for finding IDFT of signal $x_1[n]$

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% This is the code to perform IDFT for x1[n]={1,1,1,1} with N=4
% by calling function myIDFT(X,N)

%defining constants N
N=4;
x1=[1 1 1 1]; %defining x1[n]
n=0:N-1; %defining range for 'n'

subplot(2,2,1);
stem(n,x1,'Linewidth',1);
xlabel('Total samples (N) ->');
ylabel('x_{1}(n) ->');
title('Plot of x_{1}(n) = [1 1 1 1]');
grid on;

X1=myDFT(x1,N); % calculating DFT of x1[n] using myDFT
mag=fftshift(abs(X1));
ang=angle(X1);

subplot(2,2,2);
stem(n,mag,'Linewidth',1);
xlabel('Total samples (N) ->');
ylabel('Magnitude ->');
title('Magnitude vs. Samples for X_{1}(k)');
grid on;

subplot(2,2,3);
stem(n,ang,'Linewidth',1);
xlabel('Total samples (N) ->');
ylabel('Phase angle ->');
title('Phase vs. Samples for X_{1}(k)');
grid on;

y1=myIDFT(X1,N); % calculating IDFT of X1[k]

subplot(2,2,4);
stem(n,abs(y1),'Linewidth',1);
xlabel('Total samples (N) ->');
ylabel('y_{1}(n) ->');
title('y_{1}(n) obtained after IDFT of X_{1}(k)');
grid on;

sgtitle('19ucc023 - Mohit Akhouri - Exercise 1b');
```

Figure 3.5 Code for calculating IDFT of signal $x_1[n]$ with N=4

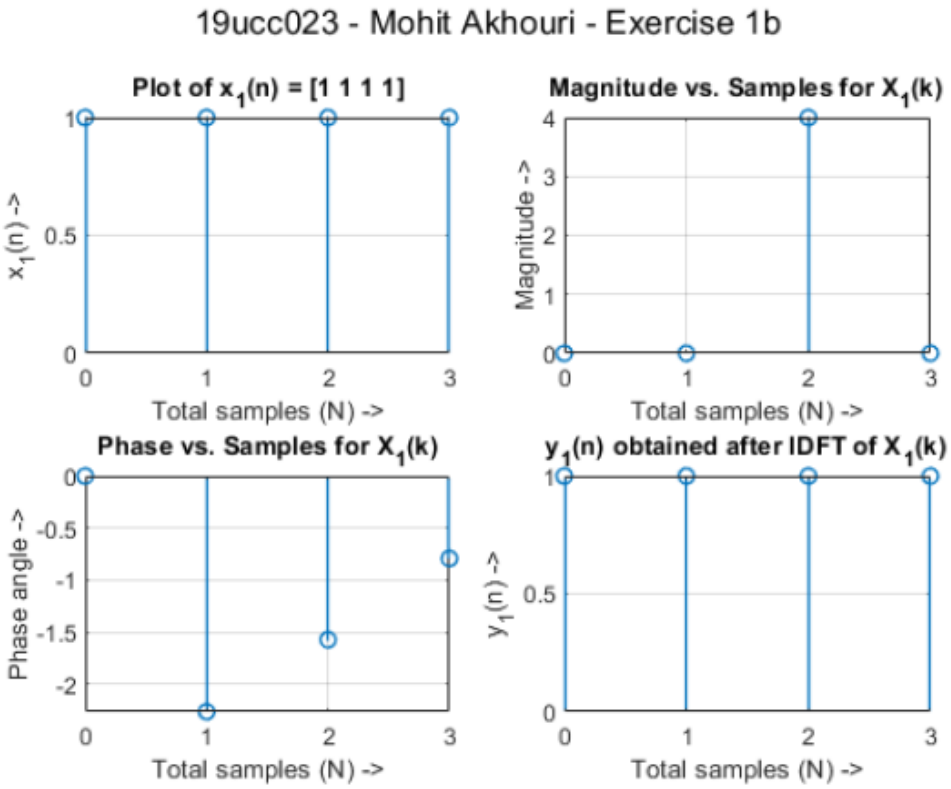


Figure 3.6 Graph of calculated IDFT of signal $x_1[n]$ with N=4

3.4.3 Exercise 2: Plotting the magnitude and phase spectra of speech signal (dove.wav) :

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% This is the code to calculate and plot the magnitude and phase
% spectra of speech signal 'dove.wav'

[x,fs]=audioread('dove.wav'); % reading audio signal
% defining constants ts,N,to and f
ts=1/fs;
N=length(x);
to=N*ts;
f=linspace(-fs,fs,N); % defining rangde of 'f'
t=ts:ts:to; % defining range of 't'

subplot(3,1,1);
plot(t,x);
xlabel('time(t) ->');
ylabel('x(t) ->');
title('x(t) = dove.wav');
grid on;

X=myDFT(x,N); % calculating DFT of audio signal
mag=fftshift(abs(X));
ang=angle(X);

subplot(3,1,2);
plot(f,mag);
xlabel('frequency(Hz) ->');
ylabel('Magnitude ->');
title('Magnitude vs. Frequency Plot of X(k)');
grid on;

subplot(3,1,3);
plot(f,ang);
xlabel('frequency(Hz) ->');
ylabel('Phase angle ->');
title('Phase vs. Frequency Plot of X(k)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri - Exercise 2');
```

Figure 3.7 Code for calculating DFT of speech signal (dove.wav) using myDFT function

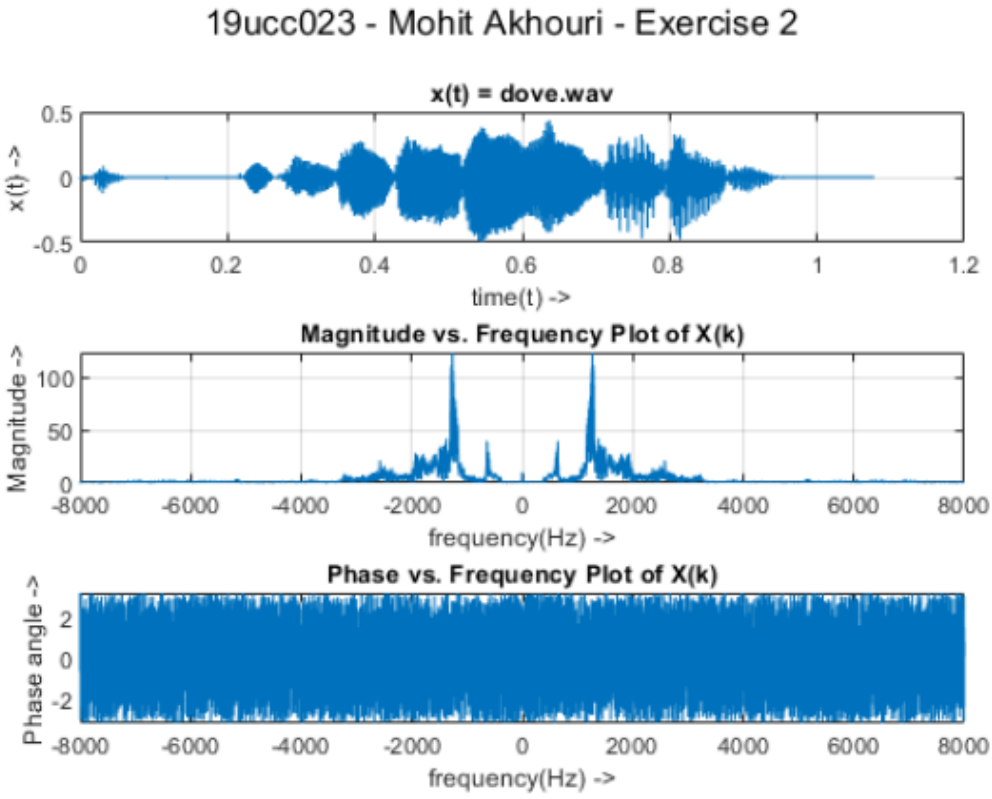


Figure 3.8 Graph of Magnitude and phase spectra of speech signal (dove.wav)

3.4.4 Exercise 3: MATLAB program to calculate the CCF of signals x[n] and y[n] :

```
function [Rxy] = myXCORR(x,y)
% This function will calculate the cross correlation of two signals
% 'x' and 'y'

% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% defining constants nx,ny and N
nx=length(x);
ny=length(y);
N=nx+ny-1;
Rxy=zeros(1,N); % initializing Rxy

flip(y); % flipping y

x=[x zeros(1,N-nx)]; % padding with right zeros
y=[zeros(1,N-ny) y]; % padding with left zeros

% main loop algorithm to calculate cross-correlation of 'x' and 'y'
for m=1:N+1
    sum=0;
    for n=1:N-m+1
        sum=sum+(x(n)*y(n+m-1));
    end
    sum=sum/N;
    Rxy(m)=sum;
end

end
```

Published with MATLAB® R2020b

Figure 3.9 Code for function myXCORR(x,y) to perform the CCF of signals x[n] and y[n]

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% This code will calculate the cross correlation of two discrete
 signals
% 'x' and 'y' where y=x+w and w=randn(1,N) , a zero-mean , unit
 variance
% of the Gaussian random process

% defining constants f,fs,N and n
f=1;
fs=200;
N=1024;
n=1:N;
x=sin((2*pi*f*n)/fs); % defining x[n]
subplot(2,2,1);
plot(n,x);
xlabel('samples (n) ->');
ylabel('x[n] ->');
title('x[n] = sin((2\pifn)/F_{s})');
grid on;

w=randn(1,N); % defining Gaussian random process
y=x+w; % defining y[n]

subplot(2,2,2);
plot(n,y);
xlabel('samples (n) ->');
ylabel('y[n] ->');
title('y[n] = x[n]+w[n]');
grid on;

Rxy_inbuilt=xcorr(x,y); % calculating cross correlation of 'x' and 'y'
 using inbuilt function
subplot(2,2,3);
plot(Rxy_inbuilt);
xlabel('samples (m) ->');
ylabel('R_{xy}[m] ->');
title('R_{xy}[m] = xcorr(x,y)');
grid on;

Rxy_myfunc=myXCORR(x,y); % calculating cross correlation of 'x' and
 'y' using myXCORR function
subplot(2,2,4);
plot(Rxy_myfunc);
xlabel('samples (m) ->');
ylabel('R_{xy}[m] ->');
title('R_{xy}[m] = myXCORR(x,y)');
grid on;

sgtitle('19ucc023 - Mohit Akhouri - Exercise 3');
```

Figure 3.10 Code for performing CCF of x[n] and y[n] using myXCORR(x,y)

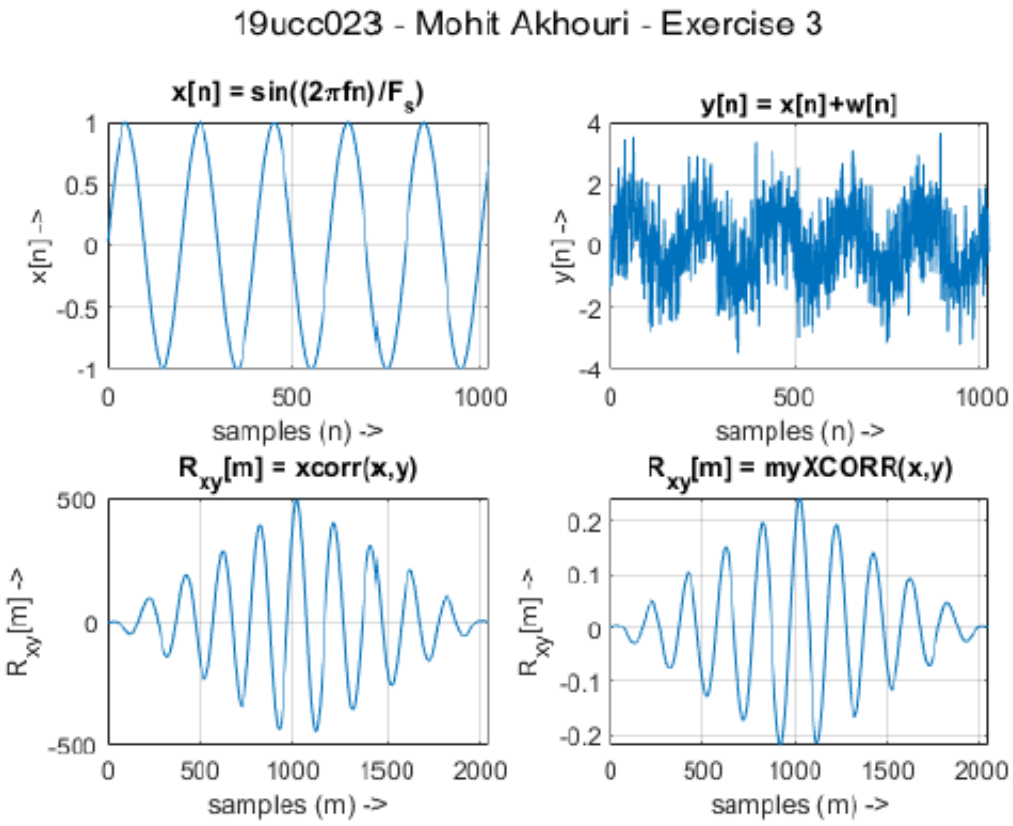


Figure 3.11 Graph for performed CCF of x[n] and y[n] using myXCORR(x,y)

3.5 Conclusion

In this experiment, we learnt about the transforms DFT for converting a signal from time domain to frequency domain and IDFT for the reverse process of converting signal from frequency domain to time domain . We also learnt about the Correlation algorithms (Cross Correlation and Auto Correlation functions) to calculate the similarity between two signals . We also created **myDFT(x,N)** and **myIDFT(X,N)** to perform DFT and IDFT respectively . We also learnt how to process an audio signal in MATLAB using **audioread()** function and how to plot its magnitude and phase spectra. Finally we created **myXCORR(x,y)** function to calculate cross-correlation of two signals x[n] and y[n].

Chapter *4*

Experiment - 4

4.1 Aim of the experiment

1. To generate two periodic signals $x_1(t)$ and $x_2(t)$
2. To compute and plot the Fourier spectra for the aforementioned periodic signals
3. To illustrate the Gibb's phenomenon

4.2 Software Used

MATLAB

4.3 Theory

4.3.1 About Fourier Series :

Jean Baptiste Joseph Fourier,a French mathematician and a physicist, was born in Auxerre, France. He initialized Fourier series, Fourier transforms and their applications to problems of heat transfer and vibrations. The Fourier series, Fourier transforms and Fourier's Law are named in his honour.
To represent any periodic signal x(t), Fourier developed an expression called Fourier series. This is in terms of an infinite sum of sines and cosines or exponentials. Fourier series uses orthogonality condition. The computation and study of Fourier series is known as harmonic analysis and is extremely useful as a way to break up an arbitrary periodic function into a set of simple terms that can be plugged in, solved individually, and then recombined to obtain the solution to the original problem or an approximation to it to whatever accuracy is desired or practical.In particular, since the superposition principle holds for solutions of a linear homogeneous ordinary differential equation, if such an equation can be solved in the case of a single sinusoid, the solution for an arbitrary function is immediately available by expressing the original function as a Fourier series and then plugging in the solution for each sinusoidal component.

The Fourier Series of a periodic signal x(t) with period T is given by :

$$x(t) = \sum_{k=-\infty}^{+\infty} D_k e^{jk\omega_o t} \tag{4.1}$$

where $\omega_o = \frac{2\pi}{T}$ and D_k is the k^{th} fourier series coeffiecient.
The Fourier Series coefficient D_k is calculated by :

$$D_k = \frac{1}{T} \int_T x(t) e^{-jk\omega_o t} dt \tag{4.2}$$

In order to compute D_k discretely, approximating the aforementioned finite integral as :

$$D_k = \frac{1}{N} \sum_{n=0}^{N-1} x(nT_s) e^{-jk\Omega_o n} \tag{4.3}$$

where $\Omega_0 = \omega_o T_s$ where T_s is the sampling interval and $N = \frac{T}{T_s}$ is number of samples in one period T.

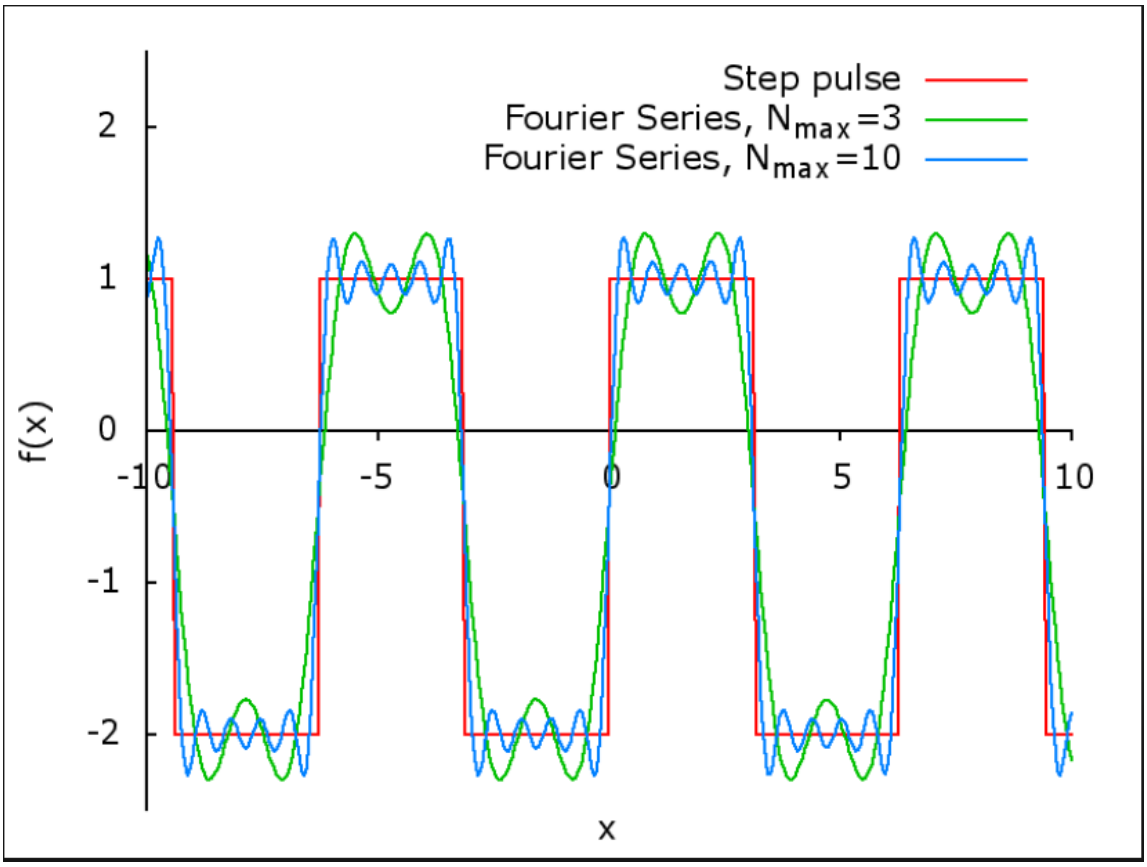


Figure 4.1 Fourier Series of a step pulse

4.3.2 About Gibb's Phenomenon :

In mathematics, the Gibbs phenomenon, discovered by **Henry Wilbraham** (1848) and rediscovered by **J. Willard Gibbs** (1899), is the peculiar manner in which the Fourier series of a piecewise continuously differentiable periodic function behaves at a jump discontinuity. The nth partial sum of the Fourier series has large oscillations near the jump, which might increase the maximum of the partial sum above that of the function itself. The overshoot does not die out as n increases, but approaches a finite limit. This sort of behavior was also observed by experimental physicists, but was believed to be due to imperfections in the measuring apparatus. This is one cause of ringing artifacts in signal processing. The Gibbs phenomenon involves both the fact that Fourier sums overshoot at a jump discontinuity, and that this overshoot does not die out as more terms are added to the sum.
The equation of the approximation of the original periodic signal x(t) is defined as :

$$x_M(t) = \sum_{k=-M}^{M} D_k e^{jk\omega_o t} \tag{4.4}$$

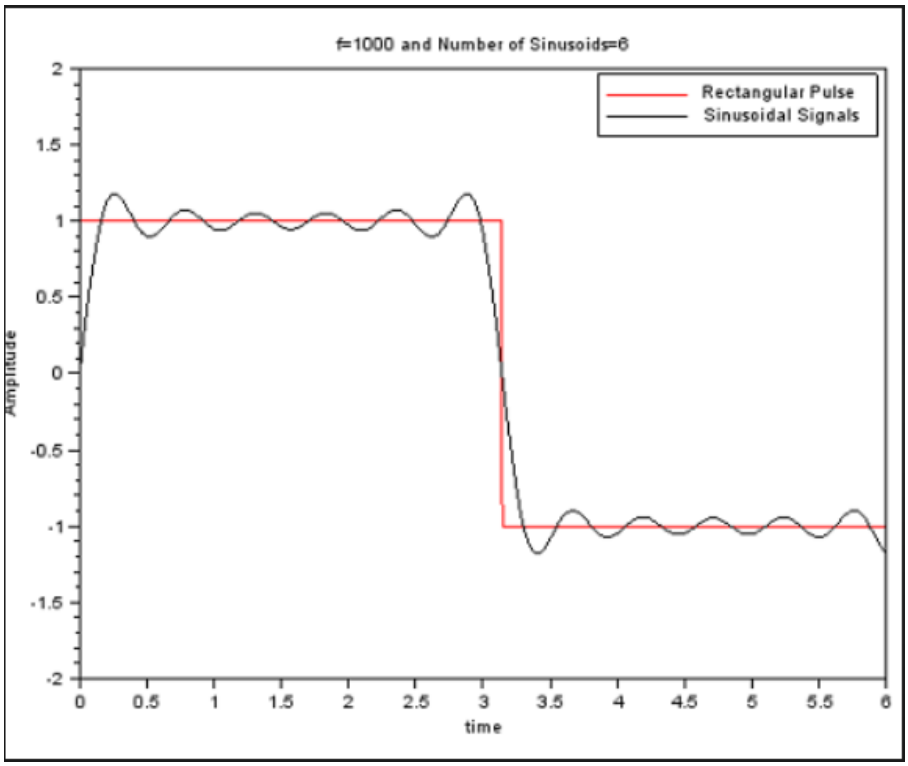


Figure 4.2 Illustration of Gibb's phenomenon

4.4 Code and Results

4.4.1 Exercise 1 : Plotting periodic signals $x_1(t)$ and $x_2(t)$ and plotting Fourier spectra :

```
function [Dk] = Fourier_Series_Coeff(x,N,num)
% This function will calculate the first 'num' coefficients of
% periodic signals, num = 1,2,3 ... 10...

% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

Dk = zeros(1,num); % initializing output variable Dk for storing
 Fourier series coefficients

% running loop to find first 'num' fourier series coefficients
for k=1:num
    sum=0;
    for n=1:N
        sum=sum+(x(n).*exp(-1j*(k-1)*(n-1)*2*pi/N));
    end
    sum=sum/N;
    Dk(k)=sum;
end

end
```

Published with MATLAB® R2020b

Figure 4.3 Code for the function Fourier_Series_Coeff to calculate the 10 fourier series coefficients D_k

```
function [xt] = Fourier_Spectra(Dk,T,t,num)
% This function will calculate and return the fourier spectra of
 signal
% with time period T for number of fourier series coefficients
% equal to 'num'

% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

xt = 0; % initializing output variable xt to store fourier spectra

% loop for finding fourier spectra of signal with 'num' Fourier
 coefficients
for k=1:num
    xt = xt + (Dk(k) .* exp(1j*(k-1)*t*2*pi/T));
end

end
```

Published with MATLAB® R2020b

Figure 4.4 Code for function Fourier_Spectra to plot fourier spectra of signals $x_1(t)$ and $x_2(t)$

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% plotting periodic signals x1(t) and x2(t)
N = 256; % defining total number of samples
T = 2; % defining bound on time
t = linspace(0,T,N); % defining the time(t) axis
num = 10; % number of fourier series coefficients to be calculated

% loop for defining periodic signals x1(t) and x2(t)
for i=1:length(t)
    if(t(i)<=1)
        x1(i)=exp(-t(i)/2);
        x2(i)=1;
    elseif(t(i)>1 & t(i)<=2)
        x1(i)=0;
        x2(i)=-1;
    end
end
% plotting signals x1(t) and x2(t)
figure;
subplot(2,1,1);
plot(t,x1,'Linewidth',1.5);
xlabel('time(t) ->');
ylabel('x_{1}(t) ->');
title('Plotting x_{1}(t) = e^{-t/2}');
grid on;
subplot(2,1,2);
plot(t,x2,'Linewidth',1.5);
xlabel('time(t) ->');
ylabel('x_{2}(t) ->');
title('Plotting x_{2}(t) = 1 ( 0 <= t <= T/2 ) , -1 ( T/2 < t <= T) ,
 T=2');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

Dk1=Fourier_Series_Coeff(x1,N,num); % calculating 10 fourier series
 coefficients for signal x1(t)
Dk2=Fourier_Series_Coeff(x2,N,num); % calculating 10 fourier series
 coefficients for signal x1(t)
mag_Dk1=abs(Dk1); % calculating magnitude of Dk1
ang_Dk1=angle(Dk1); % calculating phase angle of Dk1
mag_Dk2=abs(Dk2); % calculating magnitude of Dk2
ang_Dk2=angle(Dk2); % calculating phase angle of Dk2

% plotting magnitude and phase plot of Dk1
figure;
subplot(2,1,1);
stem(mag_Dk1,'Linewidth',1.5);
xlabel('Coefficient (k) ->');
ylabel('Magnitude of D_{k} ->');
```

1

Figure 4.5 Part 1 of the code for plotting periodic signals, calling functions to calculate D_k and fourier spectra of signals $x_1(t)$ and $x_2(t)$

```
title('Magnitude plot of D_{k} of signal x_{1}(t)');
grid on;
subplot(2,1,2);
stem(ang_Dk1,'Linewidth',1.5);
xlabel('Coefficient (k) ->');
ylabel('Phase angle of D_{k} ->');
title('Phase plot of D_{k} of signal x_{1}(t)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% plotting magnitude and phase plot of Dk2
figure;
subplot(2,1,1);
stem(mag_Dk2,'Linewidth',1.5);
xlabel('Coefficient (k) ->');
ylabel('Magnitude of D_{k} ->');
title('Magnitude plot of D_{k} of signal x_{2}(t)');
grid on;
subplot(2,1,2);
stem(ang_Dk2,'Linewidth',1.5);
xlabel('Coefficient (k) ->');
ylabel('Phase angle of D_{k} ->');
title('Phase plot of D_{k} of signal x_{2}(t)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

figure;
xt1=Fourier_Spectra(Dk1,T,t,num); % Fourier spectra of signal x1(t)
xt2=Fourier_Spectra(Dk2,T,t,num); % Fourier spectra of signal x2(t)

% plotting fourier spectra of signals x1(t) and x2(t) for Dk, k=0-9
subplot(2,1,1);
plot(t,xt1,'Linewidth',1.5);
xlabel('time(t) ->');
ylabel('x_{1}(t) ->');
title('Fourier spectra of x_{1}(t)');
grid on;
subplot(2,1,2);
plot(t,xt2,'Linewidth',1.5);
xlabel('time(t) ->');
ylabel('x_{2}(t) ->');
title('Fourier spectra of x_{2}(t)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```

Figure 4.6 Part 2 of the code for plotting periodic signals, calling functions to calculate D_k and fourier spectra of signals $x_1(t)$ and $x_2(t)$

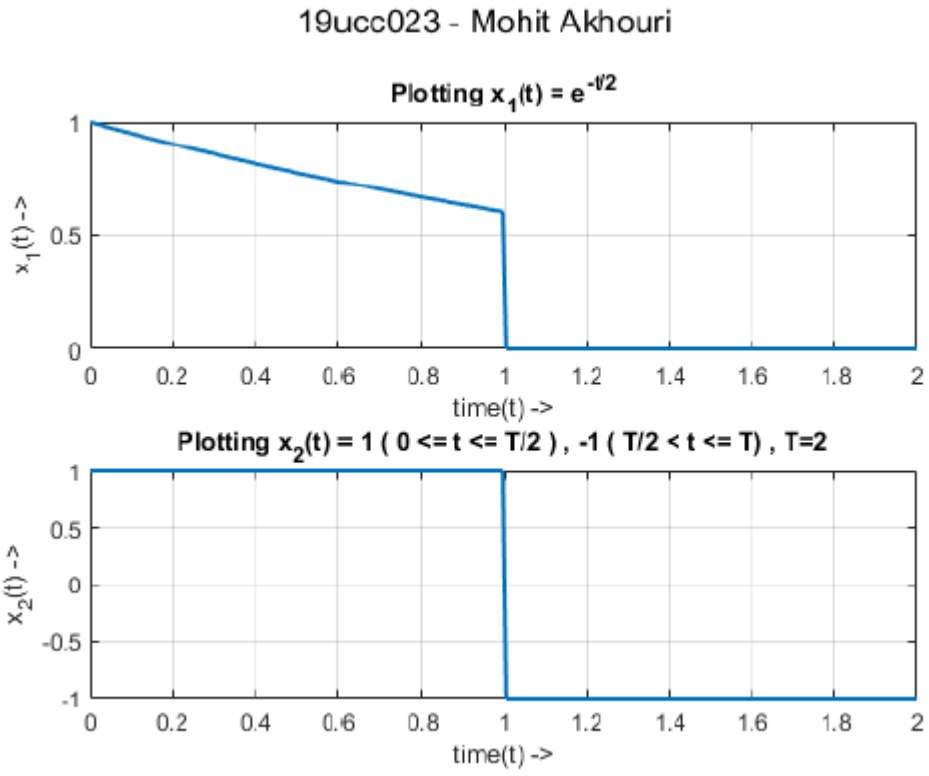


Figure 4.7 Plot of the periodic signals $x_1(t)$ and $x_2(t)$

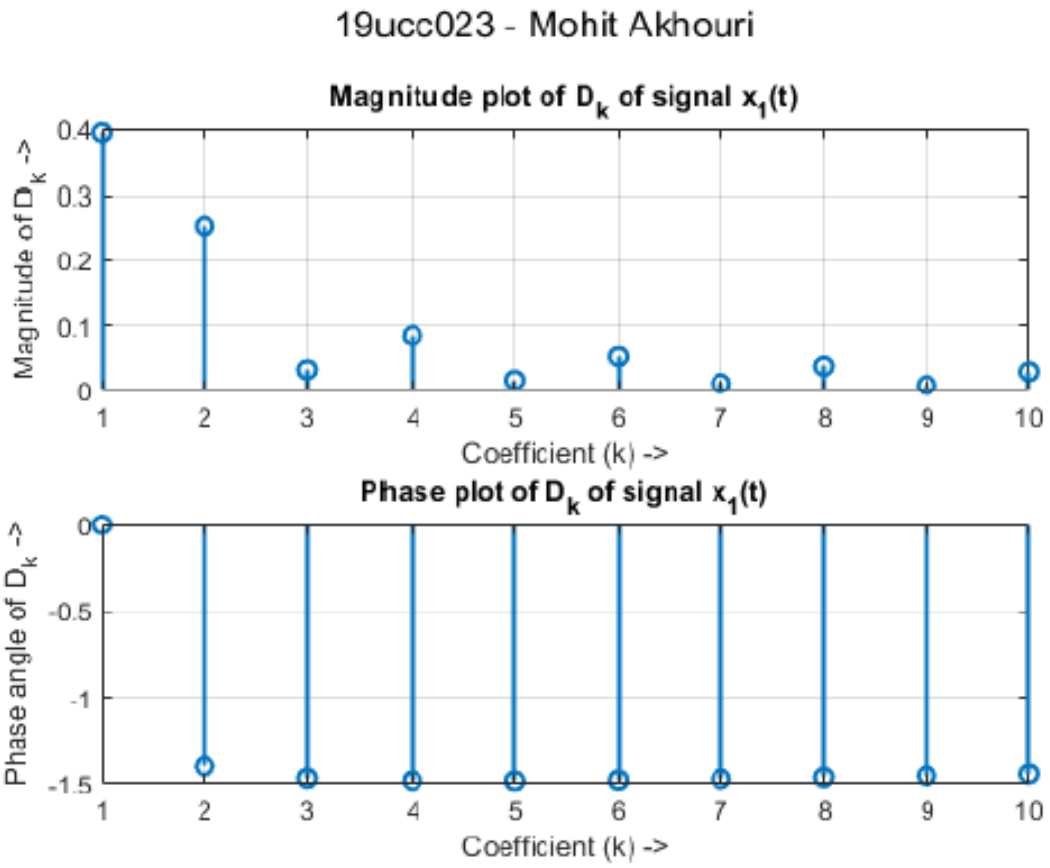


Figure 4.8 Plot of the magnitude and phase spectra of D_k for signal $x_1(t)$

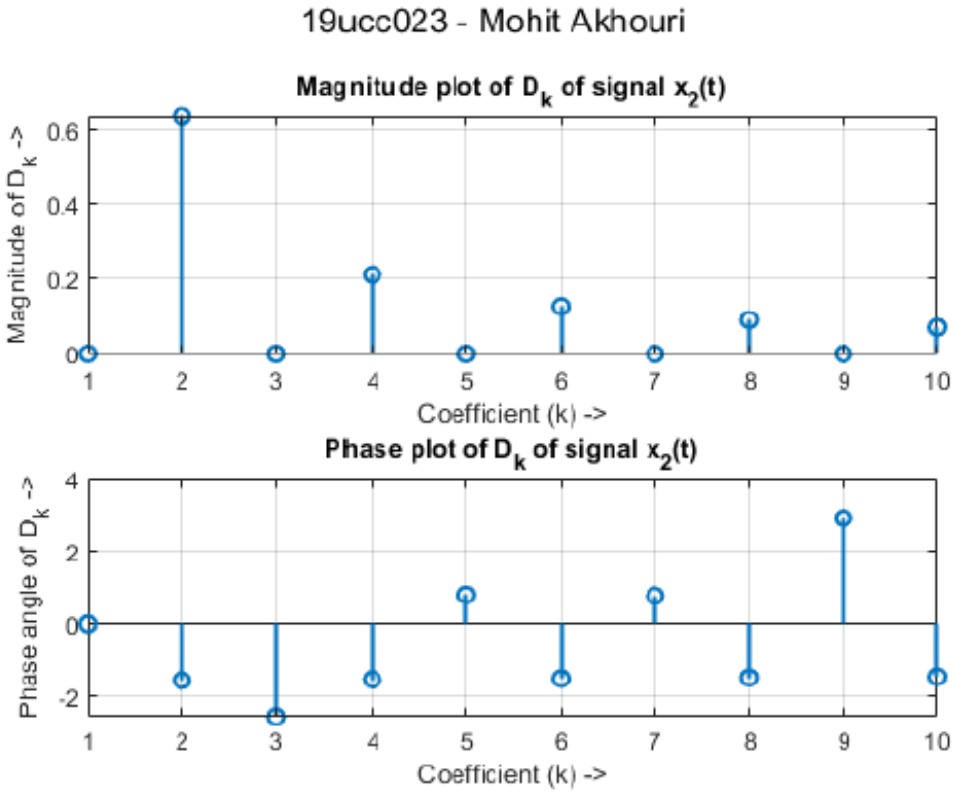


Figure 4.9 Plot of the magnitude and phase spectra of D_k for signal $x_2(t)$

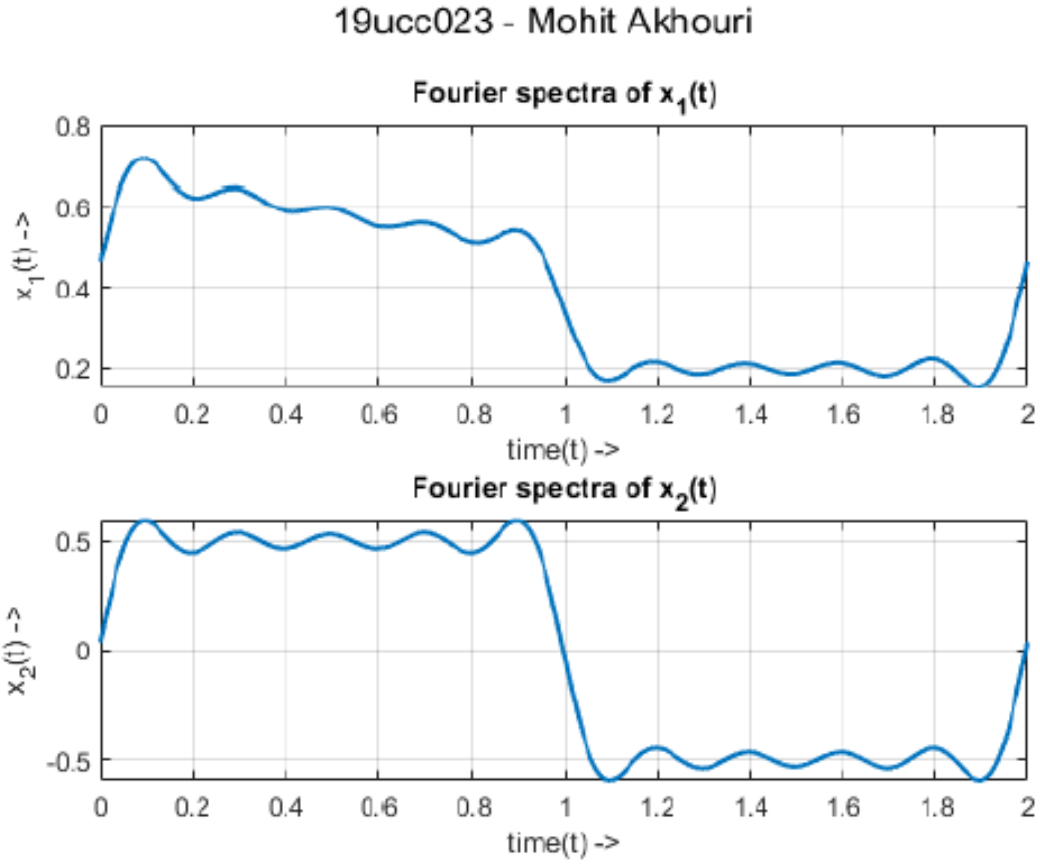


Figure 4.10 Plot of the Fourier spectra for signals $x_1(t)$ and $x_2(t)$ with 10 fourier series coefficients

4.4.2 Exercise 2: Illustration of Gibb's phenomenon for given signals for M=19 and M=99:

```
function [Dk] = Fourier_Series_Coeff_Gibbs(x,N,M)
% This function will calculate the Fourier Series Coefficient
% required for the Gibb's phenomenon that is Dk , where k = -M to M

% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

size = 2*M+1; % defining size of Dk variable
Dk = zeros(1,size); % initializing Dk variable to store coefficients

% running loop from k=-M to M to find fourier series coefficients
for k=-M:M
    sum=0;
    for n=1:N
        sum=sum+(x(n).*exp(-1j*k*(n-1)*2*pi/N));
    end
    sum=sum/N;
    Dk(k+M+1)=sum;
end

end
```

Published with MATLAB® R2020b

Figure 4.11 Code for the function to calculate Fourier Series Coefficient (D_k) used in function Gibbs_Phenomenon function

```
function [xm] = Gibbs_Phenomenon(x,t,T,N,M)
% This function will illustrate the gibb's phenomenon for signal x(t)
% for a value of M ( in this exp, M=19 and M=99 )

% This Gibb's phenomenon function returns the approximated signal
 xm(t)
% for original signal x(t) where M is a number which is given as 19
 and 99
% in this experiment

% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

Dk=Fourier_Series_Coeff_Gibbs(x,N,M); % calculating Dk used for
 calulating xm(t)
xm=0; % initializing output variable xm to store approximated signal
 xm(t)

% running loop to calculate approximated signal xm(t)
for k=-M:M
    xm=xm+(Dk(k+M+1).*exp(1j*k*t*2*pi/T));
end
end
```

Published with MATLAB® R2020b

Figure 4.12 Code for the function (Gibbs_Phenomenon) to illustrate the Gibb's phenomenon for M=19 and M=99

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% This script will illustrate the Gibb's phenonmenon on the signals
% x1(t) and x2(t)

N = 256; % defining constant N ( total samples )
T = 2; % defining bound on time axis
t = linspace(0,T,N); % defining time axis ( time from 0 to T )

% loop for defining periodic signals x1(t) and x2(t)
for i=1:length(t)
    if(t(i)<=1)
        x1(i)=exp(-t(i)/2);
        x2(i)=1;
    elseif(t(i)>1 & t(i)<=2)
        x1(i)=0;
        x2(i)=-1;
    end
end

xm1_19 = Gibbs_Phenomenon(x1,t,T,N,19); % Gibb's phenomenon for signal
 x1(t) when M=19
xm1_99 = Gibbs_Phenomenon(x1,t,T,N,99); % Gibb's phenomenon for signal
 x1(t) when M=99

% plotting approximated signal xm(t) for periodic signal x1(t)
figure;
subplot(2,1,1);
plot(t,xm1_19,'Linewidth',1.5');
xlabel('time(t) ->');
ylabel('x_{M}(t) ->');
title('approximation x_{M}(t) for periodic signal x_{1}(t) for M=19');
grid on;
subplot(2,1,2);
plot(t,xm1_99,'Linewidth',1.5);
xlabel('time(t) ->');
ylabel('x_{M}(t) ->');
title('approximation x_{M}(t) for periodic signal x_{1}(t) for M=99');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

xm2_19 = Gibbs_Phenomenon(x2,t,T,N,19); % Gibb's phenomenon for signal
 x2(t) when M=19
xm2_99 = Gibbs_Phenomenon(x2,t,T,N,99); % Gibb's phenomenon for signal
 x2(t) when M=99

% plotting approximated signal xm(t) for periodic signal x2(t)
figure;
subplot(2,1,1);
plot(t,xm2_19,'Linewidth',1.5');
```

1

Figure 4.13 Part 1 of the code for illustration of Gibb's phenomenon on periodic signals $x_1(t)$ and $x_2(t)$

```
xlabel('time(t) ->');
ylabel('x_{M}(t) ->');
title('approximation x_{M}(t) for periodic signal x_{2}(t) for M=19');
grid on;
subplot(2,1,2);
plot(t,xm2_99,'Linewidth',1.5);
xlabel('time(t) ->');
ylabel('x_{M}(t) ->');
title('approximation x_{M}(t) for periodic signal x_{2}(t) for M=99');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```

Figure 4.14 Part 2 of the code for illustration of Gibb's phenomenon on periodic signals $x_1(t)$ and $x_2(t)$

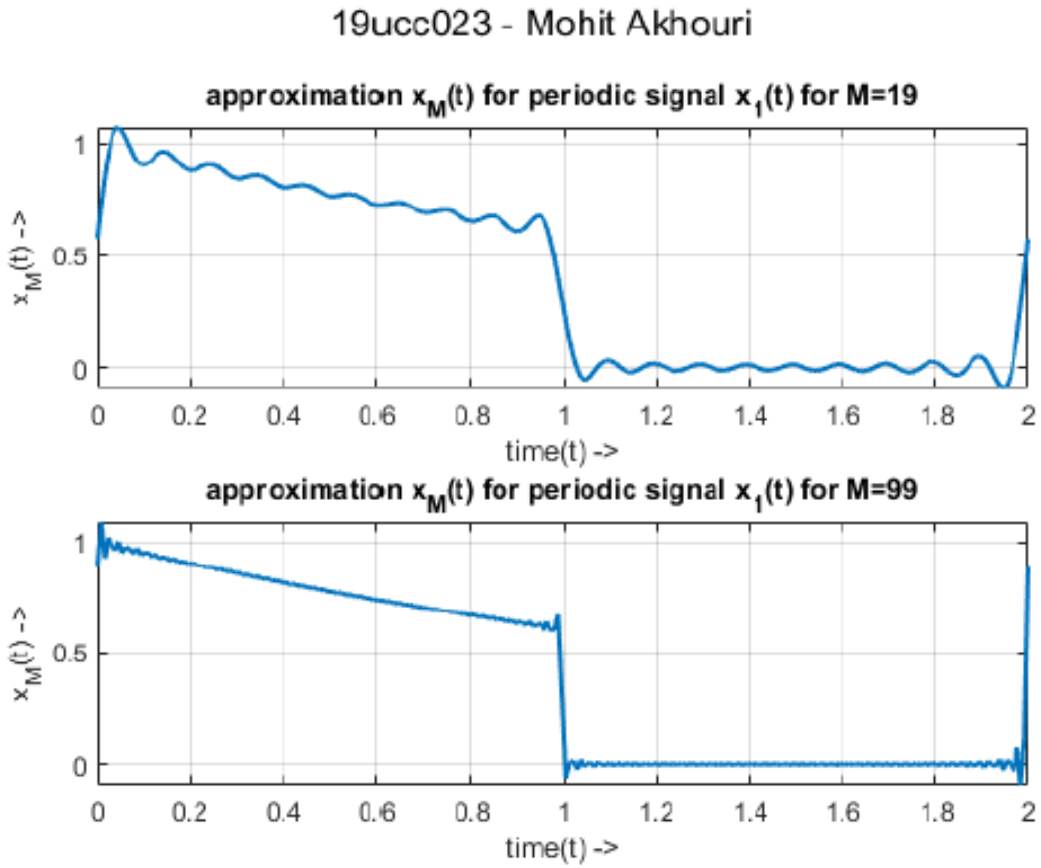


Figure 4.15 Graph of illustration of Gibb's phenomenon on signal $x_1(t)$ for M=19 and M=99

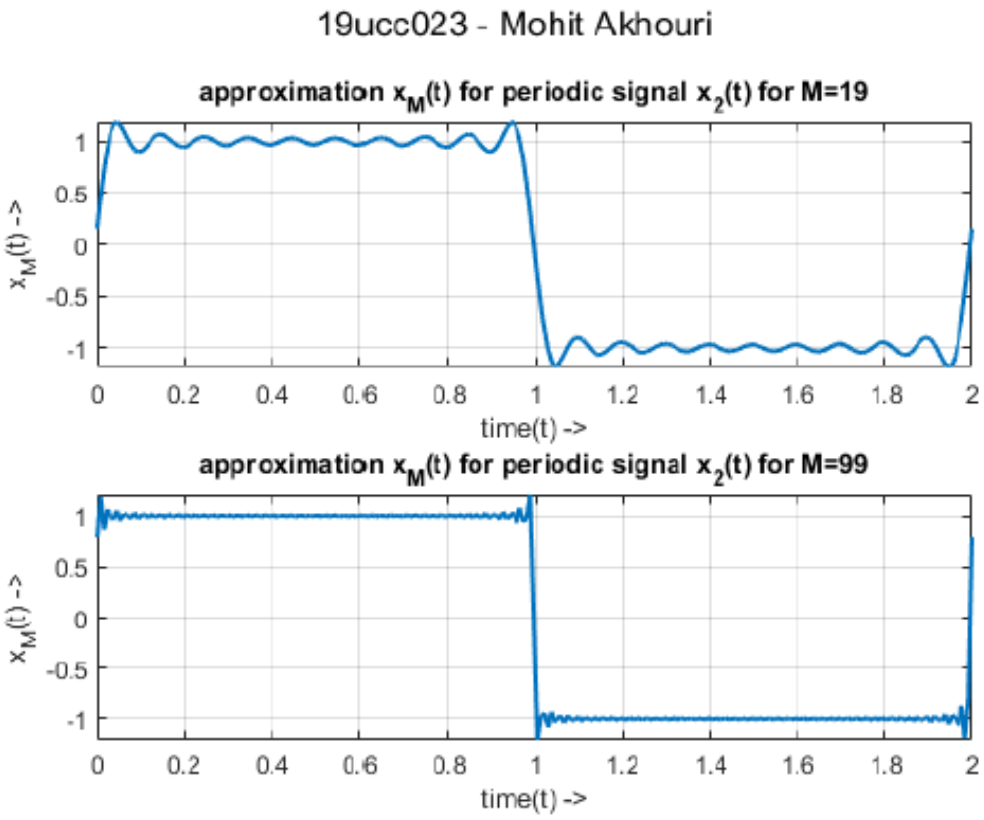


Figure 4.16 Graph of illustration of Gibb's phenomenon on signal $x_2(t)$ for M=19 and M=99

4.5 Conclusion

In this experiment, we learnt about the concept of Fourier Series and the representation of Fourier series of periodic signals. We also implemented the concept of plotting periodic signals , exponential and square wave signals. We learnt the concept of Fourier series coefficients D_k and implemented an algorithm to compute them. We also implemented algorithm to compute the Fourier Spectra of periodic signals.

We also learnt about the **Gibb's phenomenon** and how it can be used to obtain the **approximated version** of the original periodic signal. We implemented an algorithm to illustrate the Gibb's phenomenon for the two periodic signals (exponential and square wave) for two values of M , that is for M=19 and M=99.

Chapter *5*

Experiment - 5

5.1 Aim of the experiment

1. To verify the convolution property of Fourier transform
2. To verify time shifting property of Fourier transform
3. To verify frequency shifting property of Fourier transform

5.2 Software Used

MATLAB

5.3 Theory

5.3.1 About Fourier transform :

In mathematics, a Fourier transform (FT) is a mathematical transform that decomposes functions depending on space or time into functions depending on spatial or temporal frequency, such as the expression of a musical chord in terms of the volumes and frequencies of its constituent notes. The term Fourier transform refers to both the frequency domain representation and the mathematical operation that associates the frequency domain representation to a function of space or time.

The Fourier transform of a function of time is a complex-valued function of frequency, whose magnitude (absolute value) represents the amount of that frequency present in the original function, and whose argument is the phase offset of the basic sinusoid in that frequency. The Fourier transform is not limited to functions of time, but the domain of the original function is commonly referred to as the time domain. There is also an inverse Fourier transform that mathematically synthesizes the original function from its frequency domain representation, as proven by the Fourier inversion theorem.

5.3.2 Convolution property of Fourier transform :

The convolution of two functions $x_1(t)$ and $x_{2](t)}$ is written as h(t)=x_1*x_2 defined by :

$$h(t) = \int_{-\infty}^{\infty} x_1(\tau)x_2(t-\tau) \tag{5.1}$$

The convolution property states that convolution in time domain results in multiplication in frequency domain.

$$x_1(t) * x_2(t) = X_1(\omega).X_2(\omega) \tag{5.2}$$

5.3.3 Time shifting property of Fourier transform :

Time shifting property is as follows :

$$x(t) <-> X(\omega) \tag{5.3}$$

$$x(t-t_o) <-> e^{-j\omega t_o}X(\omega) \tag{5.4}$$

5.3.4 Frequency shifting property of Fourier transform :

Frequency shifting property is as follows :

$$x(t) <-> X(\omega) \tag{5.5}$$

$$e^{j\omega t_o}.x(t) <-> X(\omega-\omega_o) \tag{5.6}$$

5.4 Code and Results

5.4.1 Exercise 1 : Verifying convolution property of Fourier transform

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% This code will verify the convolution property of Fourier transform

to = 25;
ts = 1/10;
N = to/ts;
t = ts:ts:to; % defining range for t
fs = 1/ts;
f = linspace(-fs,fs,N);
xt = (t>=0 & t<10); % defining first pulse x(t)
ht = (t>=0 & t<10); % defining second pulse h(t)

figure;
subplot(2,1,1);
plot(t,xt,'Linewidth',1.5);
xlabel('Time(t)->');
ylabel('x(t) ->');
title('Plotting rectangular signal x(t) of length 10');
grid on;
subplot(2,1,2);
plot(t,ht,'Linewidth',1.5);
xlabel('Time(t)->');
ylabel('h(t)->');
title('Plotting rectangular signal h(t) of length 10');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% Finding convolution in time domain
conv_xh_t = conv(xt,ht); % using function conv to calculate
 convolution in time domain

% plotting figures;
figure;
subplot(2,1,1);
plot(conv_xh_t);
xlabel('time(t) ->');
ylabel('x(t) conv h(t)');
title('Convolution of x(t) and h(t) in time domain');
grid on;

% Finding convolution in frequency domain
conv_xh_f = abs(ifft(fft(xt).*fft(ht))); % taking inverse fft of
 multiplication of fft
% of x(t) and h(t) in frequency domain

% plotting figures;
subplot(2,1,2);
plot(conv_xh_f);
xlabel('frequency(Hz) ->');
```

Figure 5.1 Part 1 of Code for verification of convolution property of Fourier transform

```
ylabel('x(\omega) x h(\omega)');
title('Multiplication (Convolution) of x(\omega) and h(\omega) in
 Frequency domain');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 5.2 Part 2 of Code for verification of convolution property of Fourier transform

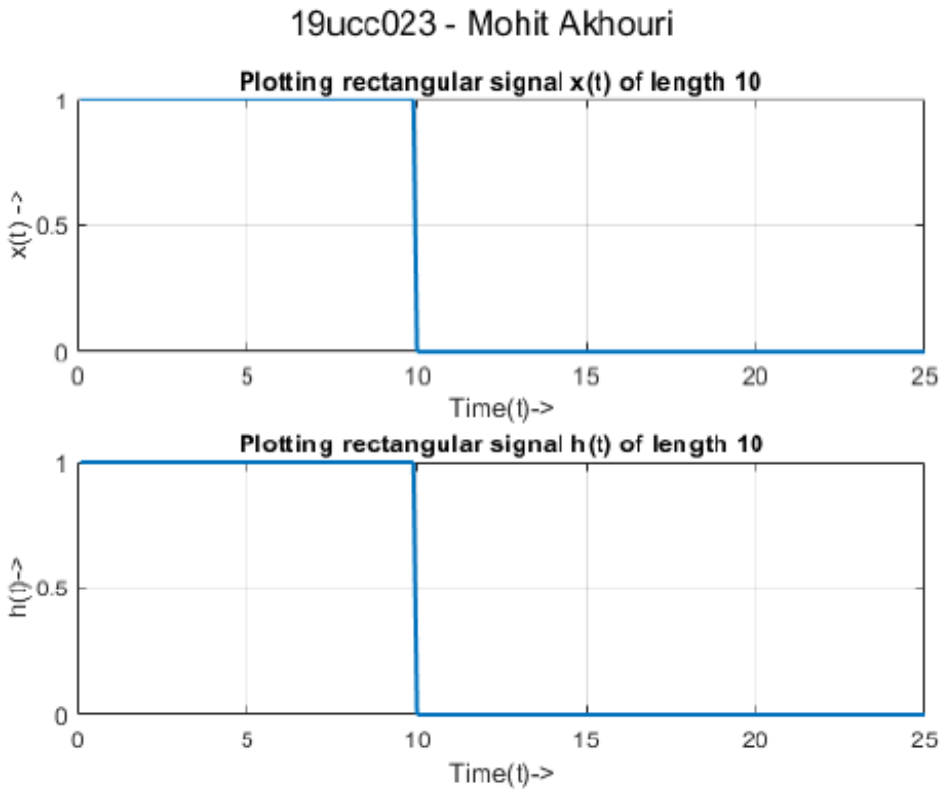


Figure 5.3 Graph of plotted pulses x(t) and h(t) of length 10

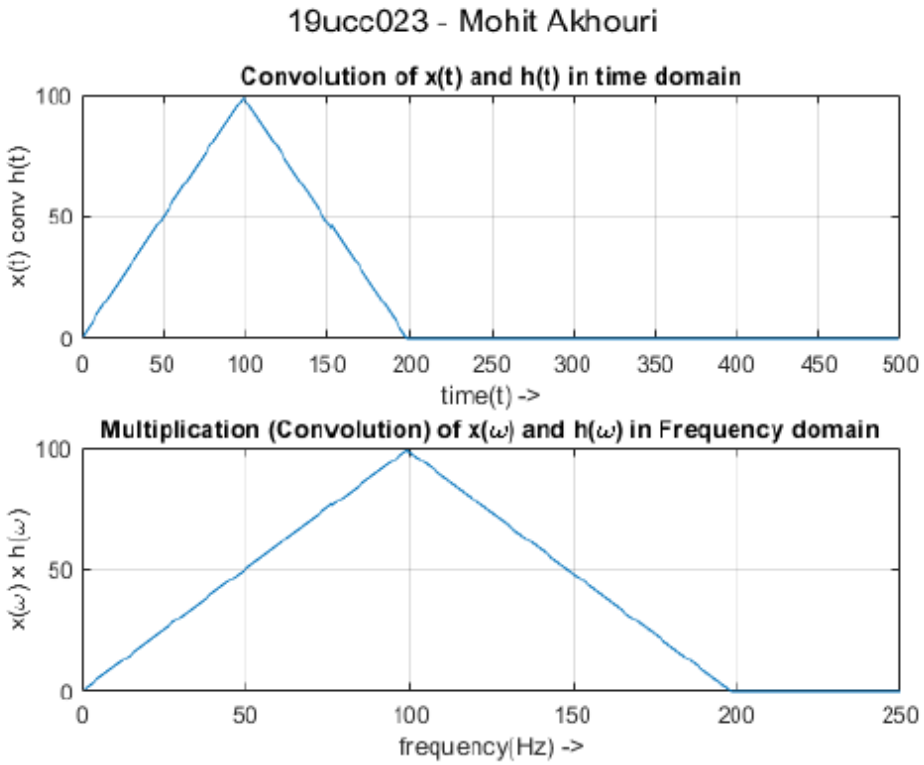


Figure 5.4 Graph of verification of convolution property of Fourier transform

5.4.2 Exercise 2: Verifying Time shifting property of Fourier transform

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% This code will verify the Time shifting property of Fourier
 transform

to = 25; % defining To
ts = 1/10; % defining Ts
N = to/ts; % defining range N
t = ts:ts:to; % defining range for t
fs = 1/ts; % defining frequency fs
f = linspace(-fs,fs,N); % defining frequency f
Xt = (t>=0 & t<10); % defining pulse X(t)

% plotting original signal X(t)
figure;
subplot(2,1,1);
plot(t,Xt,'Linewidth',1.5);
xlabel('Time(t)->');
ylabel('x(t) ->');
title('Plotting rectangular signal X(t) of length 10');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

To = 5; % defining value by which signal is to be plotted
Ft = fftshift(fft(Xt)); % taking fft of X(t)
shift_Ft = exp(-1j*2*pi*f*To).*Ft; % Shifting fourier transform
Shift_Ift = abs(ifft(shift_Ft)); % Taking inverse fourier transform
t1 = t + To;
Shift_Ift = Xt;

% plotting X(t-5)
figure;
subplot(2,1,1);
plot(t1,Shift_Ift,'Linewidth',1.5);
xlabel('time(t)->');
ylabel('X(t-5)->');
title('Shifting X(t) by 5 units to right, Plotting X(t-5)');
grid on;


To = -5; % defining value by which signal is to be plotted
Ft = fftshift(fft(Xt)); % taking fft of X(t)
shift_Ft = exp(-1j*2*pi*f*To).*Ft; % Shifting fourier transform
Shift_Ift = abs(ifft(shift_Ft)); % Taking inverse fourier transform
t2 = t + To;
Shift_Ift = Xt;

% plotting X(t+5)
subplot(2,1,2);
plot(t2,Shift_Ift,'Linewidth',1.5);
```

Figure 5.5 Part 1 of Code for verification of Time shifting property of Fourier transform

```
xlabel('time(t)->');
ylabel('X(t+5)->');
title('Shifting X(t) by 5 units to left, Plotting X(t+5)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 5.6 Part 2 of Code for verification of Time shifting property of Fourier transform

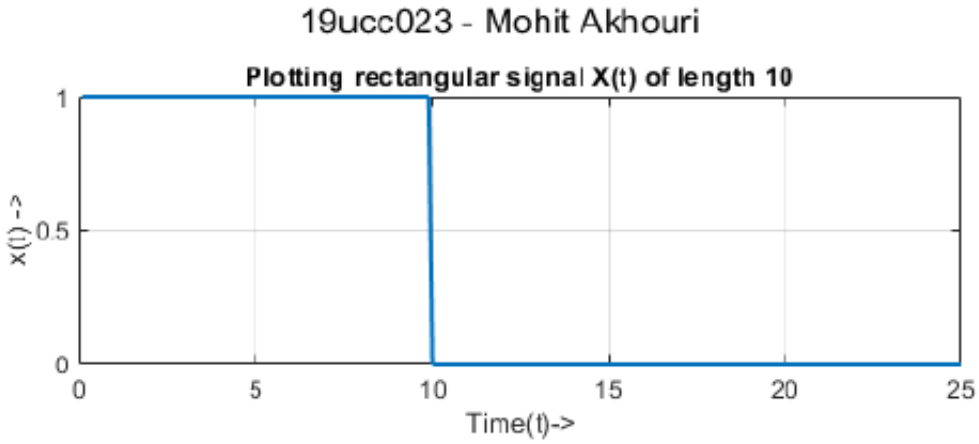


Figure 5.7 Graph of plotted pulse X(t) of length 10

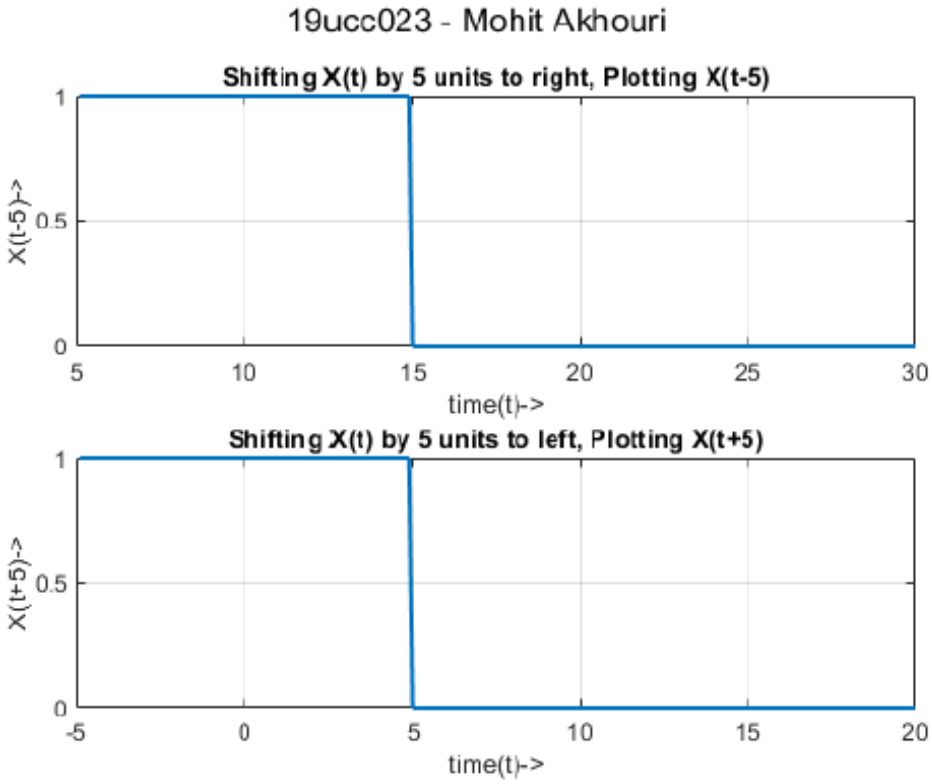


Figure 5.8 Graph of verification of Time shifting property by plotting X(t+5) and X(t-5)

5.5 Conclusion

In this experiment, we learnt about the Time shifting, Convolution and Frequency shifting property of Fourier transform and implemented functions for verification of Time shifting and Convolution property of Fourier transform.

Chapter *6*

Experiment - 6

6.1 Aim of the experiment

1. To add two sinusoidal signals
2. Eliminating the higher frequency component using fifth order low pass Butterworth filter
3. Eliminating the lower frequency component using fifth order high pass Butterworth filter

6.2 Software Used

MATLAB

6.3 Theory

6.3.1 About Butterworth filter:

The Butterworth filter is a type of signal processing filter designed to have a frequency response as flat as possible in the passband. It is also referred to as a maximally flat magnitude filter. It was first described in 1930 by the British engineer and physicist Stephen Butterworth in his paper entitled ”On the Theory of Filter Amplifiers”.
The frequency response of the Butterworth filter is maximally flat (i.e. has no ripples) in the passband and rolls off towards zero in the stopband. When viewed on a logarithmic Bode plot, the response slopes off linearly towards negative infinity. A first-order filter’s response rolls off at 6 dB per octave (20 dB per decade) (all first-order lowpass filters have the same normalized frequency response). A second-order filter decreases at 12 dB per octave, a third-order at 18 dB and so on. Butterworth filters have a monotonically changing magnitude function with , unlike other filter types that have non-monotonic ripple in the passband and/or the stopband.

Compared with a Chebyshev Type I/Type II filter or an elliptic filter, the Butterworth filter has a slower roll-off, and thus will require a higher order to implement a particular stopband specification, but

Butterworth filters have a more linear phase response in the pass-band than Chebyshev Type I/Type II and elliptic filters can achieve.

To use butterworth filter in MATLAB , use commands **butter(ord,Wn,'low')** for low pass filtering and **butter(ord,Wn,'high')** for high pass filtering and then pass them through function **filter()** to get the filtered output signal.

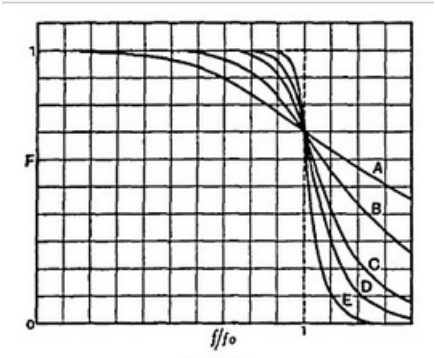


Figure 6.1 Frequency response plot from Butterworth's 1930 paper

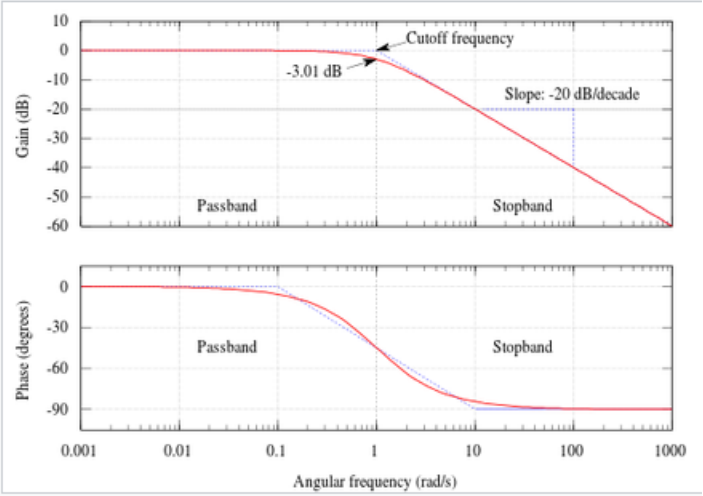


Figure 6.2 The Bode plot of a first-order Butterworth low-pass filter

6.4 Code and Results

6.4.1 Illustration of use of high pass and low pass Butterworth filter

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% This code will plot two sinusoidal signals and make use of
 butterworth
% filter

Fs = 50; % defining frequency Fs
Ts = 1/Fs; % defining Ts ( Sampling time period )
NyqFreq = Fs/2; % defining nyquist frequency
To = 10; % defining total duration ( To )
t = Ts:Ts:To; % defining time interval ( t )
N = length(t); % defining total number of samples N

% defining constants A1,A2,freq1,freq2
A1 = 20;
A2 = 30;
freq1 = 2;
freq2 = 20;

x1 = A1*sin(2*pi*freq1*t); % defining signal x1(t)
x2 = A2*sin(2*pi*freq2*t); % defining signal x2(t)

sum = x1+x2; % defining sum of x1(t) and x2(t)

% plotting figures - x1,x2 and sum
figure;
subplot(2,1,1);
plot(t,x1);
xlabel('time(t)->');
ylabel('x_{1}(t)->');
title('plotting x_{1}(t)');
grid on;
subplot(2,1,2);
plot(t,x2);
xlabel('time(t)->');
ylabel('x_{2}(t)->');
title('plotting x_{2}(t)');
sgtitle('19ucc023 - Mohit Akhouri');
grid on;

figure;
plot(t,sum);
xlabel('time(t)->');
ylabel('x_{1}(t) + x_{2}(t)->');
title('19ucc023 - Mohit Akhouri','Plotting the sum of x_{1}(t) and
 x_{2}(t)');
grid on;

f=linspace(-NyqFreq,NyqFreq,N); % defining range for frequency
Magn_Sum=fftshift(fft(sum,N)); % taking fft of signal sum and fftshift
```

Figure 6.3 Part 1 of Code for working of Butterworth filter and plotting of Frequency and Time domain spectra

```
Magn_Sum_Modified=10*log10(abs(Magn_Sum)); % taking log scale (dB
 scale) of fft of signal sum
figure;
% plotting the magnitude spectra of signal sum
plot(f,Magn_Sum_Modified);
xlabel('Frequency(Hz)->');
ylabel('Magnitude->');
title('19ucc023 - Mohit Akhouri','Magnitude plot of fourier transform
 of signal sum (x_{1}(t)+x_{2}(t))');
grid on;

f_low=2; % defining lower cutoff frequency
f_high=20; % defining upper cutoff frequency

Wn_low=(2*f_low/Fs); % defining Wn for lower cutoff frequency
Wn_high=(2*f_high/Fs); % defining Wn for upper cutoff frequency

order=5; % defining the order of butterworth filter

[b_low,a_low] = butter(order,Wn_low,'low'); % low pass butterworth
 filter
[b_high,a_high] = butter(order,Wn_high,'high'); % high pass
 butterworth filter

nv_low=filter(b_low,a_low,sum); % filtering low pass frequency
nv_high=filter(b_high,a_high,sum); % filtering high pass frequency

mag_lowpass=fftshift(fft(nv_low,N)); % taking magnitude of fft of low
 pass signal
mag_lowpass_modified=10*log10(abs(mag_lowpass)); % taking db scale of
 magnitude of fft of low pass signal

mag_highpass=fftshift(fft(nv_high,N)); % taking magnitude of fft of
 high pass signal
mag_highpass_modified=10*log10(abs(mag_highpass)); % taking db scale
 of magnitude of fft of high pass signal

figure;
% plotting low pass and high pass signals
subplot(2,1,1);
plot(f,mag_lowpass);
xlabel('Frequency (Hz)->');
ylabel('Magnitude->');
title('Magnitude plot of low pass butterworth signal');
grid on;
subplot(2,1,2);
plot(f,mag_highpass);
xlabel('Frequency (Hz)->');
ylabel('Magnitude->');
title('Magnitude plot of high pass butterworth signal');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 6.4 Part 2 of Code for working of Butterworth filter and plotting of Frequency and Time domain spectra

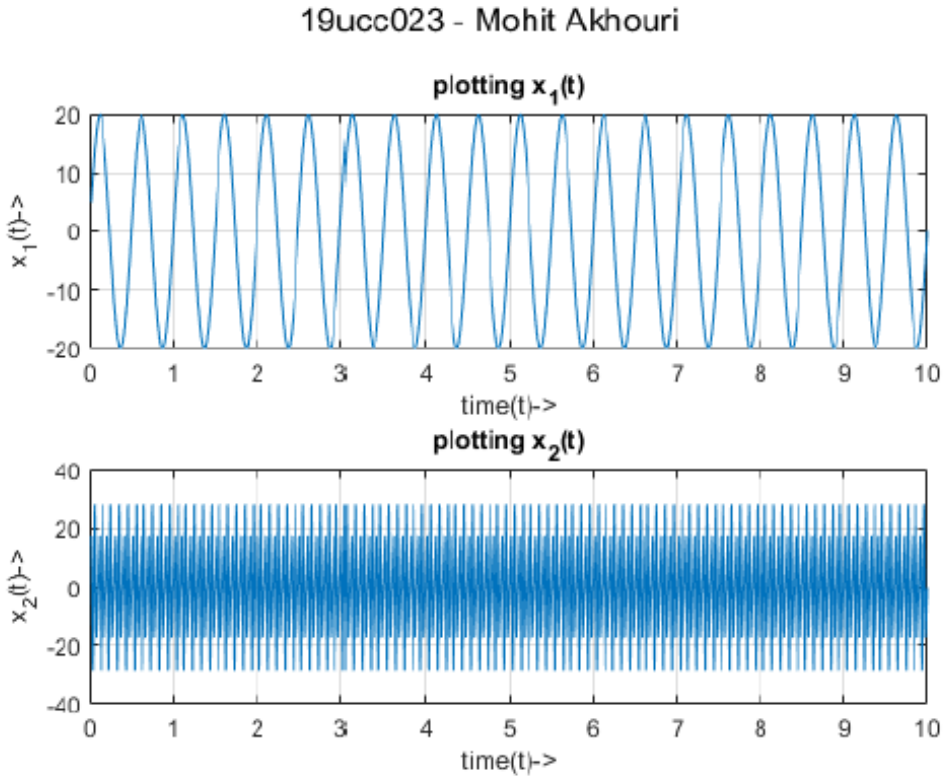


Figure 6.5 Graph of Plotted sinusoidal signals $x_1(t)$ and $x_2(t)$

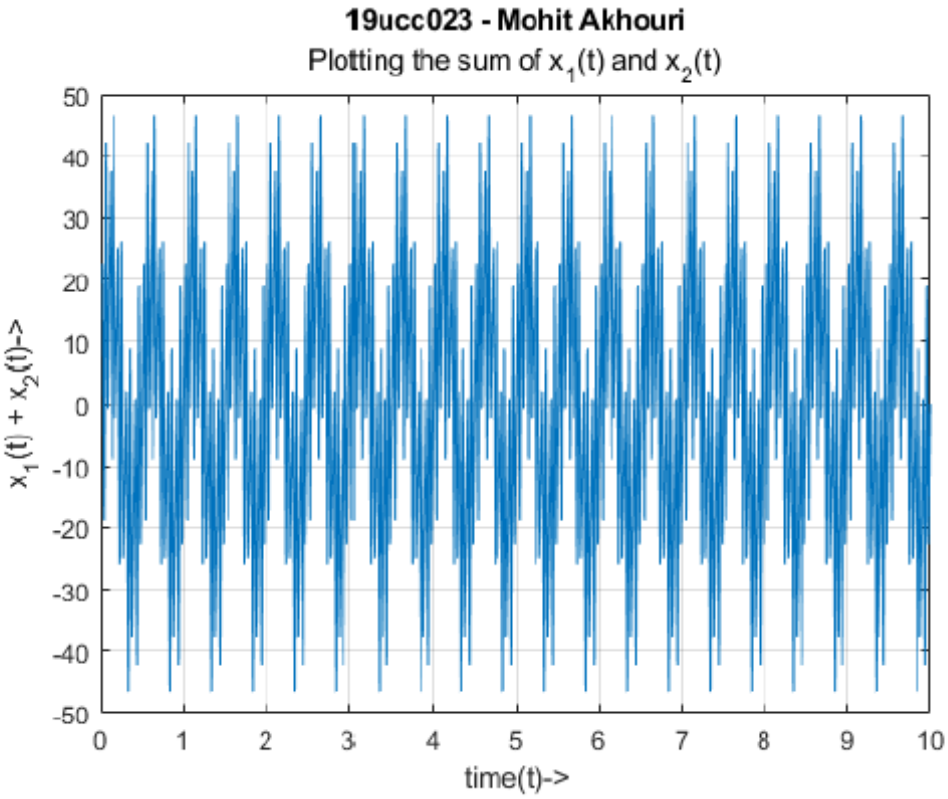


Figure 6.6 Graph of Plotted signal sum = $x_1(t)$ + $x_2(t)$

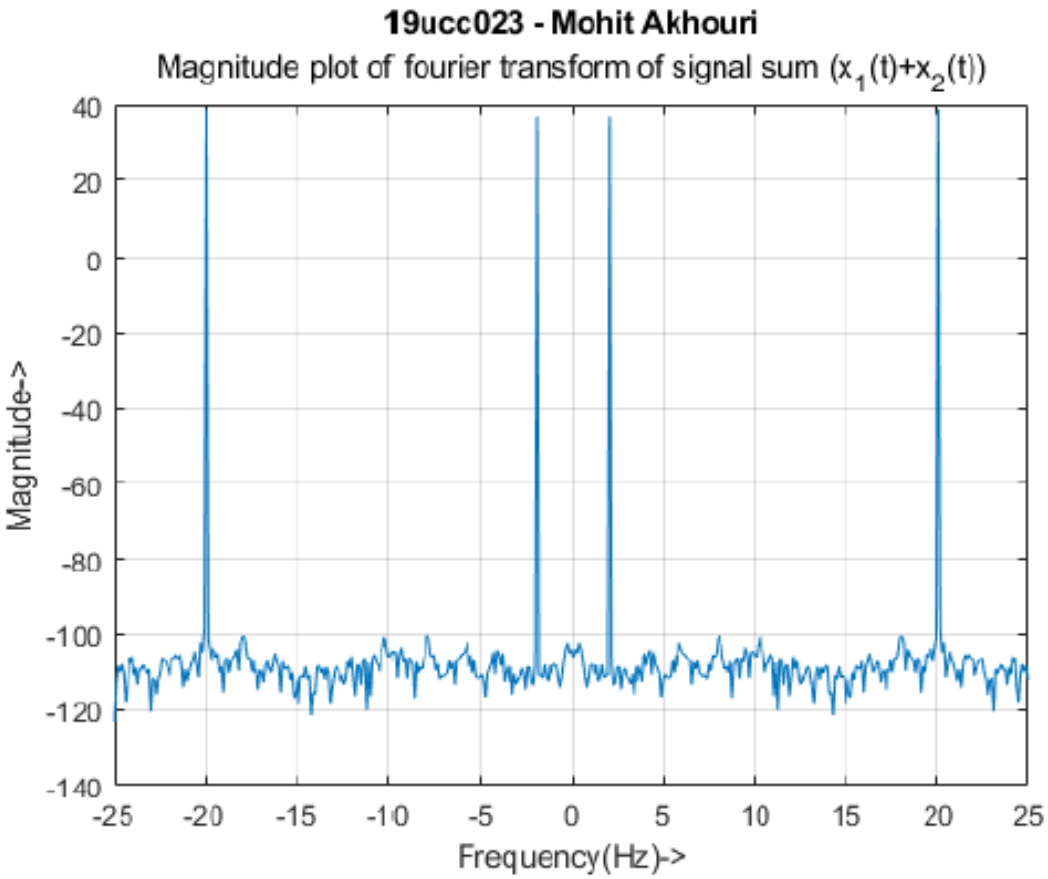


Figure 6.7 Magnitude vs. Frequency plot for the signal sum = $x_1(t)$ + $x_2(t)$

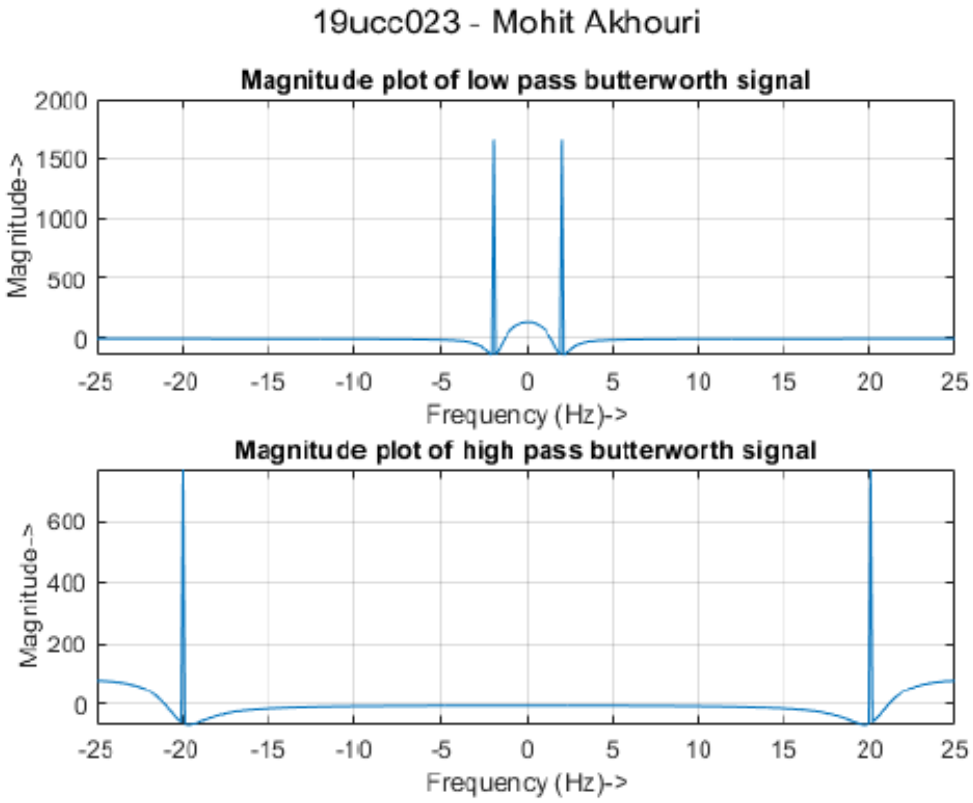


Figure 6.8 Graph of low pass and high pass signal obtained on passing signal **sum** through Butterworth filter

6.5 Conclusion

In this experiment, We have studied about the **sinusoidal signals** and how to plot them . We have used two frequencies for signals **2 Hz** and **20 Hz** and plotted their time domain representation. We calculated the sum of the two sinusoidal signals and plotted its time domain and frequency domain representation.

We also studied about **Low pass** and **High pass Butterworth filter** and how it can be used for filtering process, that is elimination of **lower frequency component** and **higher frequency component** . We studied MATLAB commands to use butterworth filter, **butter( )** and **filter( )**.

Chapter *7*

Experiment - 7

7.1 Aim of the experiment

1. To generate amplitude modulated signals (=0.3,0.5 and 1) and Demodulate the modulated wave.
2. To implement a Double Side Band - Suppressed Carrier (DSB-SC) modulator using a sampler (switch) and a band-pass filter and recover the modulating signal.

7.2 Software Used

MATLAB

7.3 Theory

7.3.1 About AM Modulation :

Amplitude modulation (AM) is a modulation technique used in electronic communication, most commonly for transmitting messages with a radio carrier wave. In amplitude modulation, the amplitude (signal strength) of the carrier wave is varied in proportion to that of the message signal, such as an audio signal. This technique contrasts with angle modulation, in which either the frequency of the carrier wave is varied as in frequency modulation, or its phase, as in phase modulation.
In Amplitude Modulation the amplitude of high frequency sine wave (carrier) is varied in accordance with the instantaneous value of the modulating signal.

$$m(t) = A_m cos(2f_m t) \tag{7.1}$$

$$c(t) = Acos(2\pi f_c(t)) \tag{7.2}$$

$$S_{AM}(t) = [A_c + A_m cos(2\pi f_m t)]cos(2\pi f_c(t)) \tag{7.3}$$

Where equation 7.1 is **modulating signal**, equation 7.2 is **carrier signal** and equation 7.3 is equation of **Amplitude modulated wave** . A_m is amplitude of modulating signal, A_c is amplitude of carrier signal, f_m is frequency of modulating signal and f_c is frequency of carrier signal.

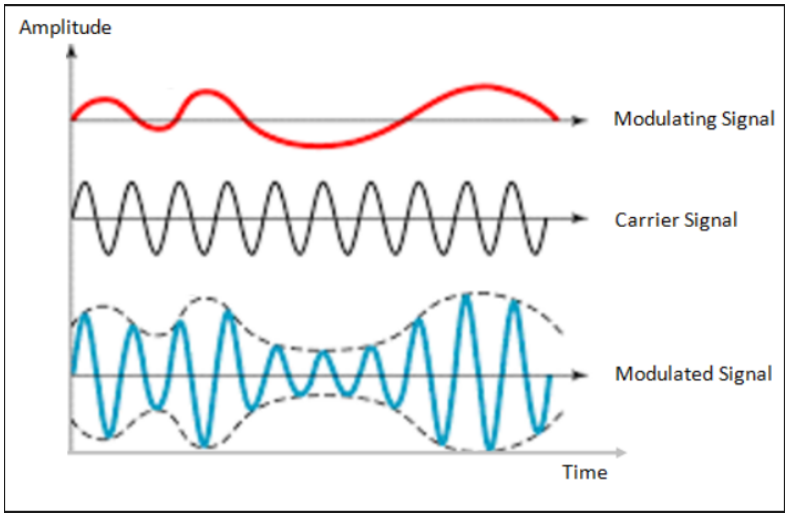


Figure 7.1 AM modulation of message signal

7.3.2 About Modulation Index :

Modulation index is also known as modulation depth is defined for the carrier wave to describe the modu- lated variable of carrier signal varying with respect to its unmodulated level. It is represented as follows:

$$\mu = \frac{m_p}{A} = \frac{V_{max} - V_{min}}{V_{max} + V_{min}} \tag{7.4}$$

where, m_p is the peak value of modulating signal and A is carrier amplitude, V_{max} and V_{min} are maximum and minimum values of the envelope.

7.3.3 About DSB-SC Modulation :

Double-sideband suppressed-carrier transmission (DSB-SC) is transmission in which frequencies produced by amplitude modulation (AM) are symmetrically spaced above and below the carrier frequency and the carrier level is reduced to the lowest practical level, ideally being completely suppressed. In the DSB-SC modulation, unlike in AM, the wave carrier is not transmitted; thus, much of the power is distributed between the side bands, which implies an increase of the cover in DSB-SC, compared to AM, for the same power use. DSB-SC transmission is a special case of double-sideband reduced carrier transmission. It is used for radio data systems. This mode is frequently used in Amateur radio voice communications, especially on High-Frequency bands. The basic transmit signal for DSB-SC is :

$$S_{DSB-SC}(t) = m(t)cos(2\pi f_c t) \tag{7.5}$$

where f_c is the carrier frequency. The fourier transform S(f) the modulator output is related to the Fourier transform M(f) of the message signal by :

$$S(f) = \frac{1}{2}M(f - f_c) + \frac{1}{2}M(f + f_c) \tag{7.6}$$

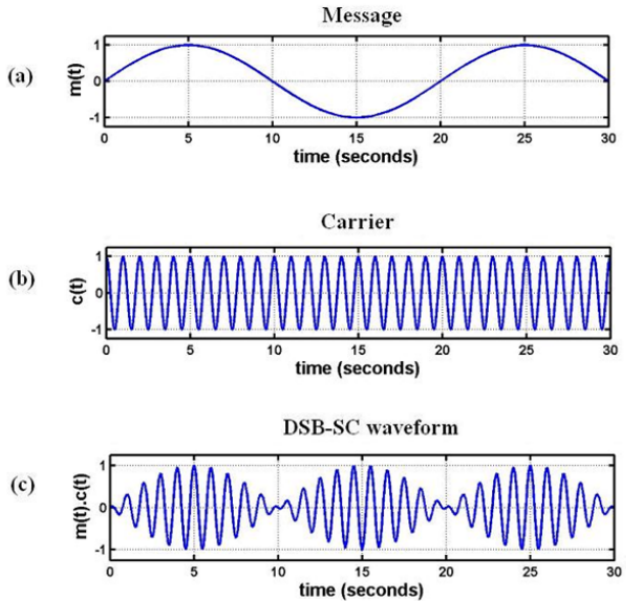


Figure 7.2 DSB-SC Modulation of message signal

7.3.4 About DSB-SC Demodulation :

Demodulation is recovering the message signal back from the modulated signal. Demodulation has two basic types.

- Non Coherent or Envelop Detection
- Coherent or Product Detection

Envelop detection is mainly reserved for Double side band (DSB) and Vestigial side band (VSB) type modulation. The **Product or Coherent detection** can be used in every AM type.

7.3.5 About Coherent detection:

The same carrier signal (which is used for generating SSB-SC wave) is used to detect the message signal. In this process, the message signal can be extracted from SSB-SC wave by multiplying it with a carrier, having the same frequency and the phase of the carrier used in SSB-SC modulation. The resulting signal is then passed through a Low Pass Filter. The output of this filter is the desired message signal.

7.4 Code and Results

7.4.1 AM Wave generation in time and frequency domain

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% This code will generate AM wave forms in time and frequency domains

Am = 10; % defining Am modulating amplitude
fm = 200; % defining fm modulating frequency
fc = 2000; % defining fc carrier frequency

t = linspace(0,0.02,300); % defining range for time axis

% generating AM wave form for 30% modulation
mu1 = 0.3; % 30% modulation index
Ac1 = Am/mu1; % defining carrier amplitude
m1t = Am*cos(2*pi*fm*t); % defining message signal
c1t = Ac1*cos(2*pi*fc*t); % defining carrier signal
am1 = (Ac1 + Am*cos(2*pi*fm*t)).*cos(2*pi*fc*t); % AM signal for 30%
 modulation

% plotting figures
figure;
subplot(3,1,1);
plot(t,m1t);
xlabel('time(t)->');
ylabel('m(t) ->');
title('Message signal');
grid on;
subplot(3,1,2);
plot(t,c1t);
xlabel('time(t)->');
ylabel('c(t) ->');
title('Carrier signal');
grid on;
subplot(3,1,3);
plot(t,am1);
xlabel('time(t)->');
ylabel('X_{AM}(t) ->');
title('X_{AM}(t) for 30% modulation');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% generating AM wave form for 50% modulation
mu2 = 0.5; % 50% modulation index
Ac2 = Am/mu2; % defining carrier amplitude
m2t = Am*cos(2*pi*fm*t); % defining message signal
c2t = Ac2*cos(2*pi*fc*t); % defining carrier signal
am2 = (Ac2 + Am*cos(2*pi*fm*t)).*cos(2*pi*fc*t); % AM signal for 50%
 modulation

% plotting figures
figure;
```

Figure 7.3 Part 1 of Code for AM wave generation

```
subplot(3,1,1);
plot(t,m2t);
xlabel('time(t)->');
ylabel('m(t) ->');
title('Message signal');
grid on;
subplot(3,1,2);
plot(t,c2t);
xlabel('time(t)->');
ylabel('c(t) ->');
title('Carrier signal');
grid on;
subplot(3,1,3);
plot(t,am2);
xlabel('time(t)->');
ylabel('X_{AM}(t) ->');
title('X_{AM}(t) for 50% modulation');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% generating AM wave form for 100% modulation
mu3 = 1; % 100% modulation index
Ac3 = Am/mu3; % defining carrier amplitude
m3t = Am*cos(2*pi*fm*t);  % defining message signal
c3t = Ac3*cos(2*pi*fc*t); % defining carrier signal
am3 = (Ac3+ Am*cos(2*pi*fm*t)).*cos(2*pi*fc*t); % AM signal for 100%
 modulation

% plotting figures
figure;
subplot(3,1,1);
plot(t,m3t);
xlabel('time(t)->');
ylabel('m(t) ->');
title('Message signal');
grid on;
subplot(3,1,2);
plot(t,c3t);
xlabel('time(t)->');
ylabel('c(t) ->');
title('Carrier signal');
grid on;
subplot(3,1,3);
plot(t,am3);
xlabel('time(t)->');
ylabel('X_{AM}(t) ->');
title('X_{AM}(t) for 100% modulation');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% frequency respose for 30% modulation
fft1=fftshift(fft(am1));
fft1_modified=(abs(fft1));
```

Figure 7.4 Part 2 of Code for AM wave generation

```
% frequency response for 50% modulation
fft2=fftshift(fft(am2));
fft2_modified=(abs(fft2));

% frequency response for 100% modulation
fft3=fftshift(fft(am3));
fft3_modified=(abs(fft3));

% plotting figures
figure;
subplot(3,1,1);
plot(fft1_modified);
xlabel('frequency(Hz)->');
ylabel('X_{AM}(\omega)->');
title('X_{AM}(\omega) for 30% modulation');
grid on;
subplot(3,1,2);
plot(fft2_modified);
xlabel('frequency(Hz)->');
ylabel('X_{AM}(\omega)->');
title('X_{AM}(\omega) for 50% modulation');
grid on;
subplot(3,1,3);
plot(fft3_modified);
xlabel('frequency(Hz)->');
ylabel('X_{AM}(\omega)->');
title('X_{AM}(\omega) for 100% modulation');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 7.5 Part 3 of Code for AM wave generation

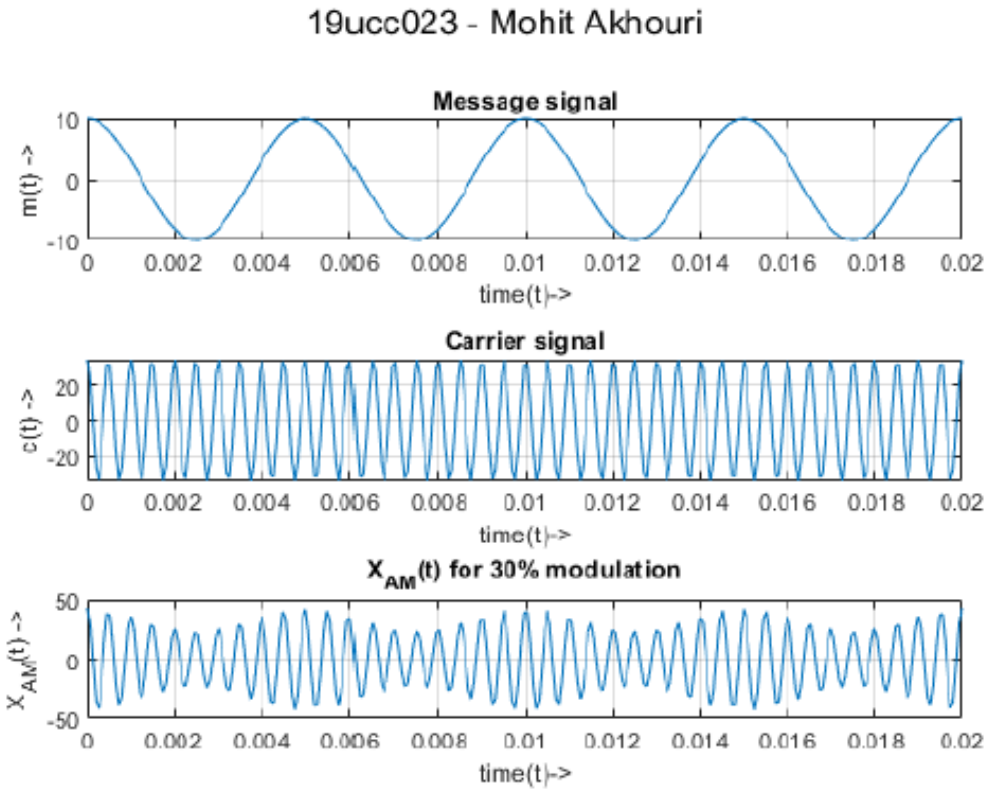


Figure 7.6 AM wave generation for 30% modulation

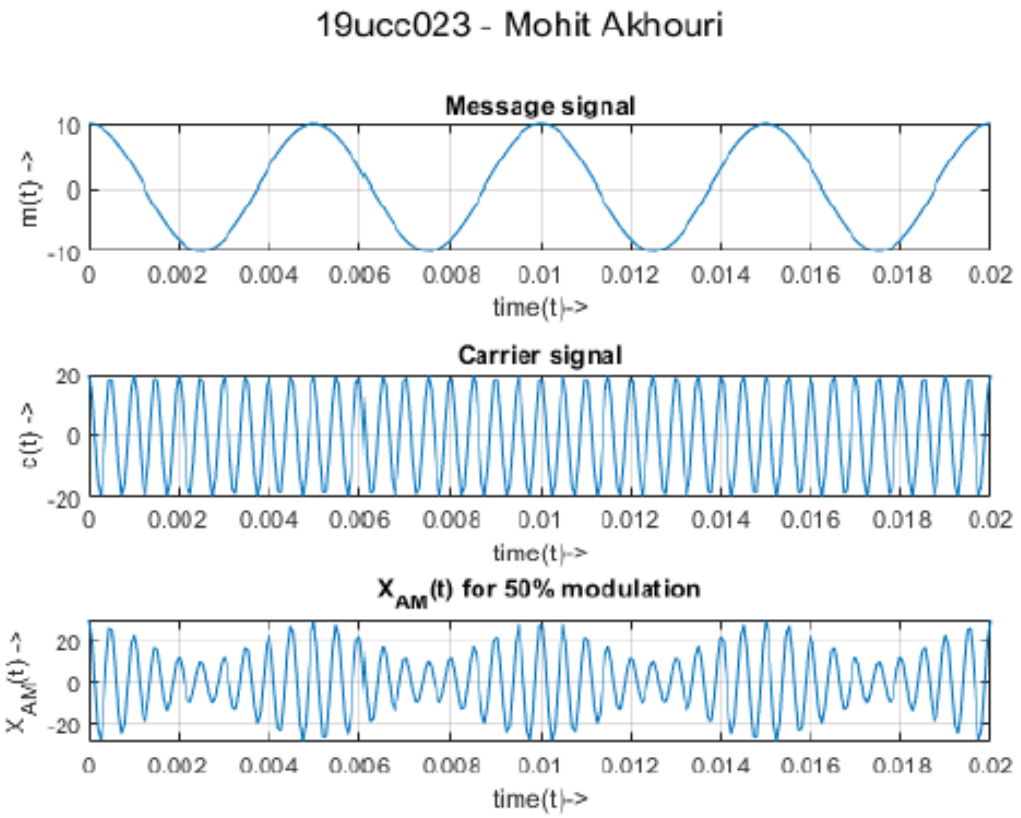


Figure 7.7 AM wave generation for 50% modulation

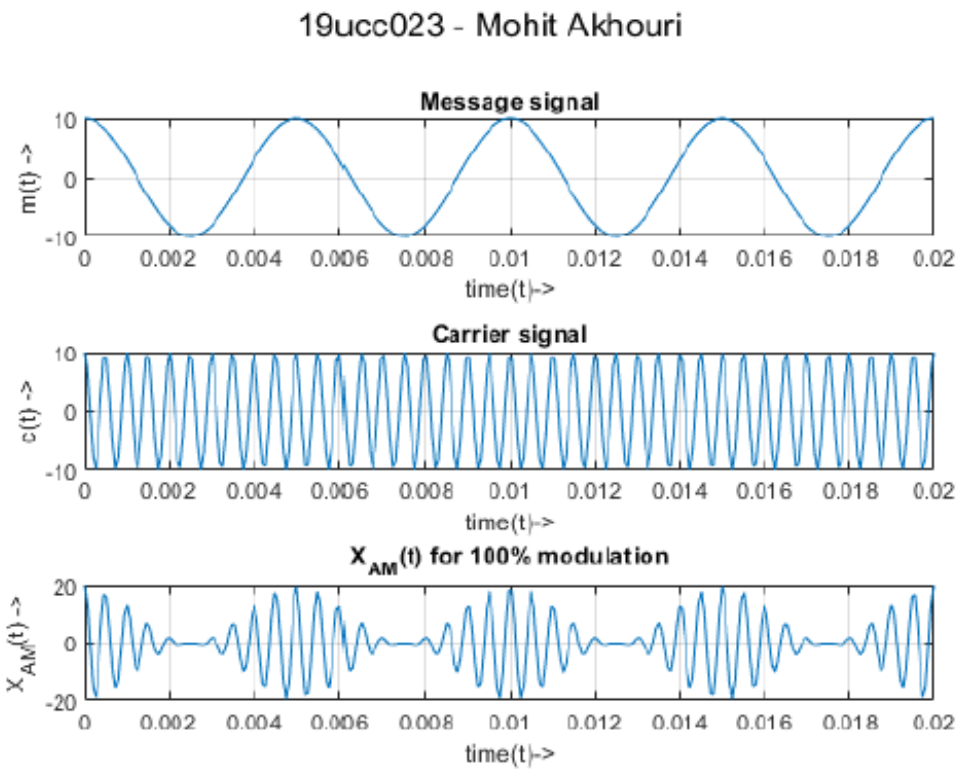


Figure 7.8 AM wave generation for 100% modulation

7.4.2 Illustration of Over-Modulation for modulation index 1.2 in AM waves

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% this code will illustrate the over modulation in case of AM
 modulated
% signal for mu = 1.2
Am = 10; % defining modulating amplitude
fm = 200; % defining modulating frequency
fc = 2000; % defining carrier frequency
t = linspace(0,0.01,300); % defining time axis
mu = 1.2; % modulation index = 1.2
Ac = Am/mu; % defining carrier amplitude
mt = Am*cos(2*pi*fm*t); % defining message signal
ct = Ac*cos(2*pi*fc*t); % defining carrier signal

% plotting figures
figure;
subplot(2,1,1);
plot(t,mt);
xlabel('time(t)->');
ylabel('m(t)->');
title('Message signal');
grid on;
subplot(2,1,2);
plot(t,ct);
xlabel('time(t)->');
ylabel('c(t)->');
title('Carrier signal');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

am = (Ac + Am*cos(2*pi*fm*t)).*cos(2*pi*fc*t); % AM modulation for mu
 = 1.2
% plotting figures
figure;
subplot(2,1,1);
plot(t,am);
xlabel('time(t)->');
ylabel('X_{AM}(t)->');
title('X_{AM}(t) for modulation index 1.2');
grid on;

demod1=abs(hilbert(am)); % obtaining demodulation for AM signal
% plotting figures
subplot(2,1,2);
plot(t,demod1);
xlabel('time(t)->');
ylabel('demodulated m(t)->');
title('Demodulated Distorted message signal (m(t))');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 7.9 Code for illustration of Over Modulation for modulation index = 1.2

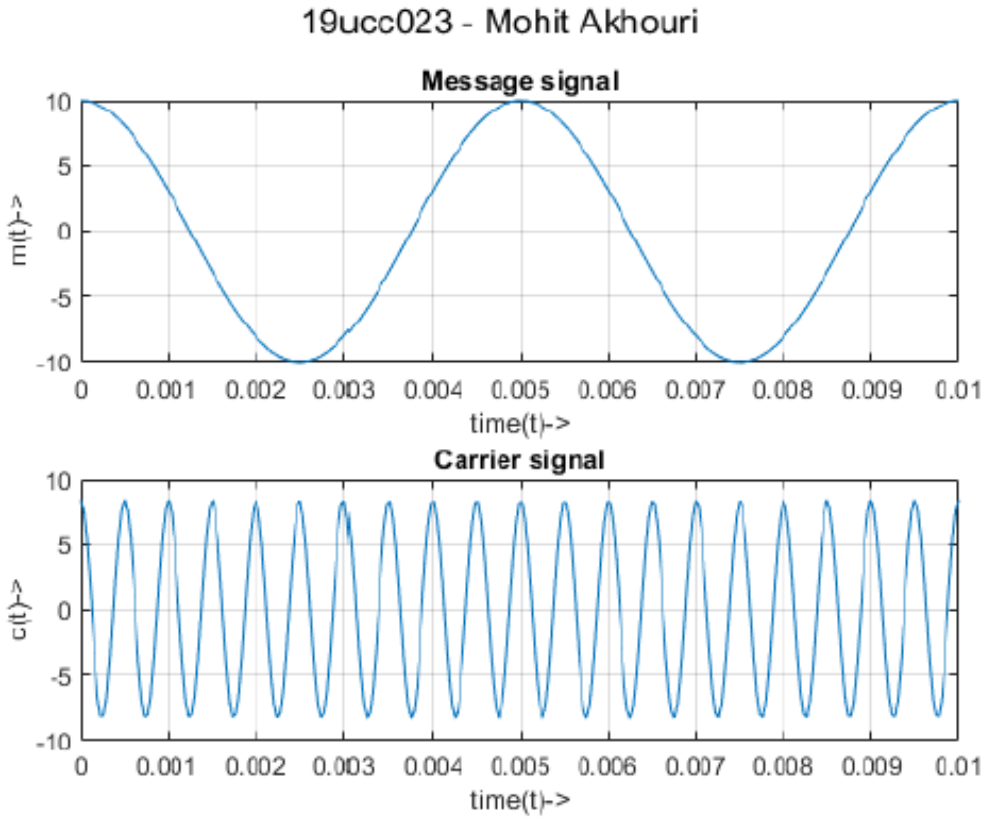


Figure 7.10 Graph of message signal and carrier signal

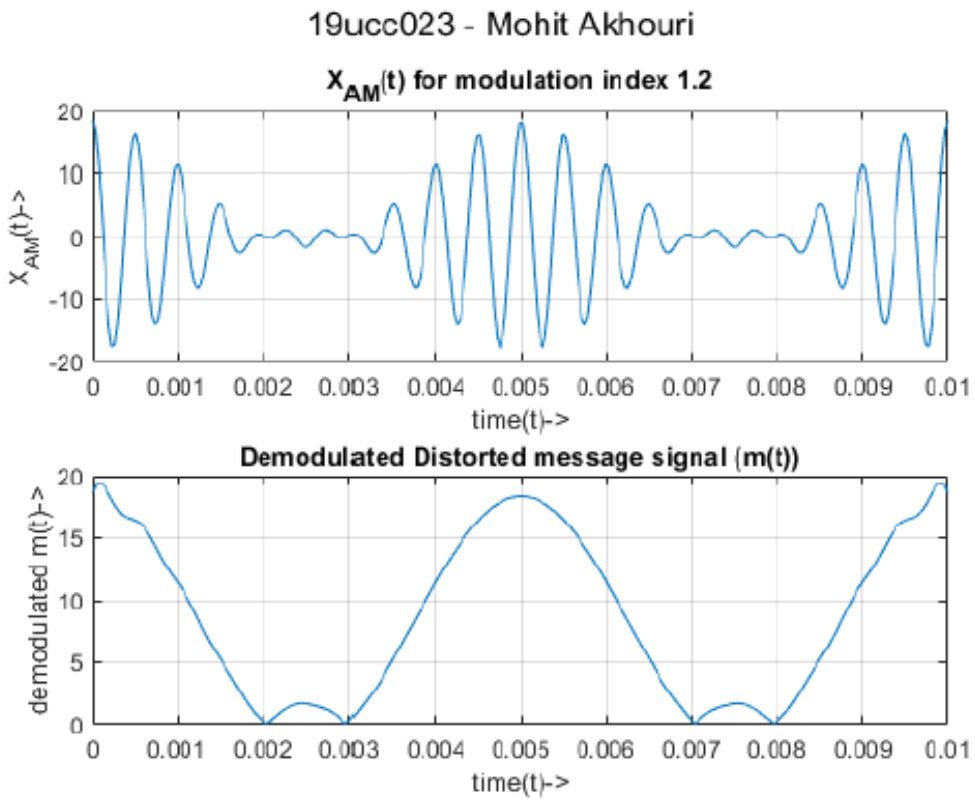


Figure 7.11 Graph of AM wave and Demodulated signal for modulation index = 1.2

7.4.3 Generation of DSB-SC wave in time and frequency domain

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% this code will generate DSB-SC signals for 30%,50% and 100%
 modulation

Am = 5; % defining modulating amplitude
Ac = 5; % defining carrier amplitude
fm = 1; % defining modulating frequency
fc = 20; % defining carrier frequency
mu = Am/Ac; % defining modulation index

t = linspace(0,5,500); % defining time axis (t)
mt = Am*cos(2*pi*fm*t); % defining message signal
ct = Ac*cos(2*pi*fc*t); % defining carrier signal

% plotting figures
subplot(3,1,1);
plot(t,mt);
xlabel('time(t)->');
ylabel('m(t)->');
title('Message signal');
grid on;
subplot(3,1,2);
plot(t,ct);
xlabel('time(t)->');
ylabel('c(t)->');
title('Carrier signal');
grid on;

dsbsct = mt.*(Ac*cos(2*pi*fc*t)); % defining DSB-SC signal for mu=1
% plotting figures
subplot(3,1,3);
plot(t,dsbsct);
xlabel('time(t)->');
ylabel('X_{DSB-SC}(t)->');
title('X_{DSB-SC}(t) for 100% modulation');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

fft1=fftshift(fft(dsbsct));
fft1_modified=(abs(fft1));

% plotting figure
figure;
plot(fft1_modified);
xlabel('frequency(Hz)->');
ylabel('X_{DSB-SC}(\omega)->');
title('19ucc023 - Mohit Akhouri','X_{DSB-SC}(\omega) for 100%
 modulation');
grid on;
```

Figure 7.12 Code for generation of DSB-SC signal in time and frequency domain

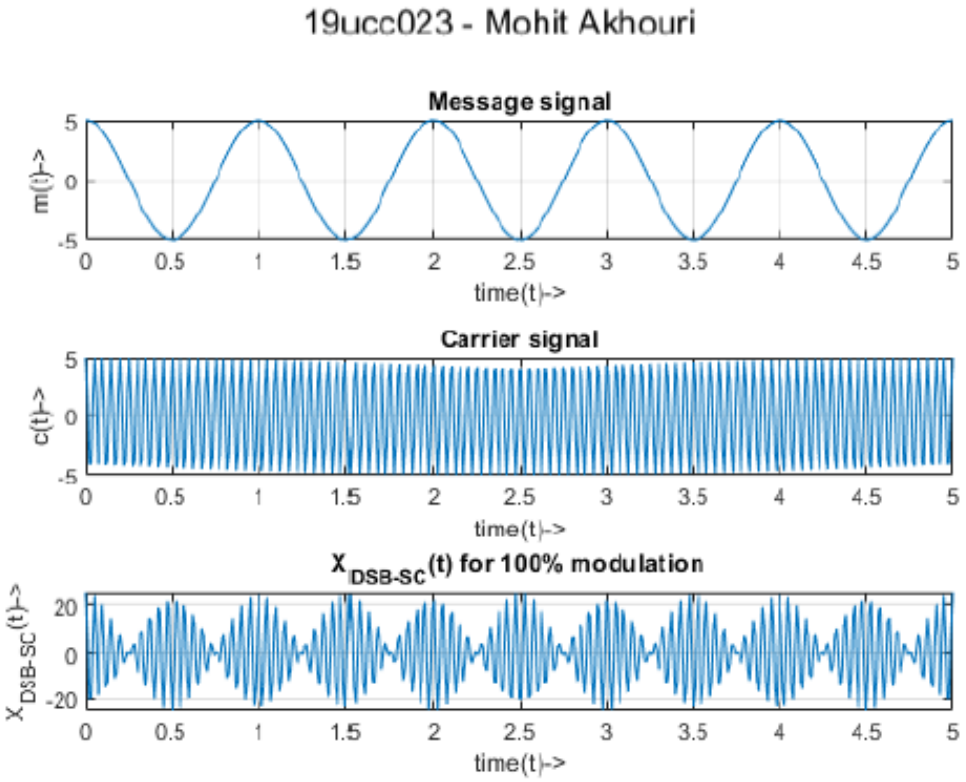


Figure 7.13 Graph of message signal, carrier signal and DSB-SC signal in time domain

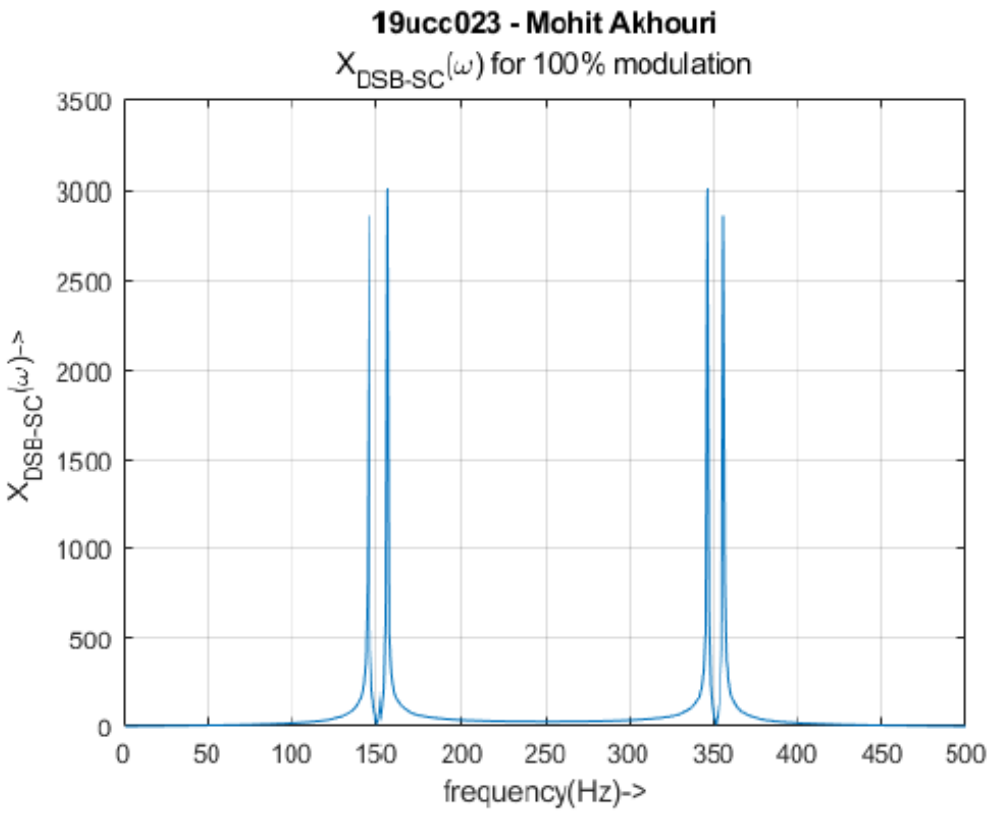


Figure 7.14 Graph of DSB-SC signal in frequency domain

7.5 Conclusion

In this experiment, we touched upon the concepts of Communication and studied two types of modulation techniques **DSB-SC** and **DSB-C or AM**. We learnt their equations and how to generate them using message signal and carrier signal in **time domain** and **frequency domain** using MATLAB. We also learnt about the detection techniques - namely , **Coherent detection** and **Non-coherent detection** for getting back the message signal. We learnt about the MATLAB function hilbert() and how it can be used for demodulation.

Chapter *8*

Experiment - 8

8.1 Aim of the experiment

1. To generate frequency modulated signal and demodulate it (without MATLAB inbuilt function)

8.2 Software Used

MATLAB

8.3 Theory

8.3.1 About Frequency modulation (FM) :

Frequency modulation (FM) is the encoding of information in a carrier wave by varying the instantaneous frequency of the wave. The technology is used in telecommunications, radio broadcasting, signal processing, and computing. In this process, the frequency of the carrier signal varies linearly with the message signal.

Frequency modulation is widely used for FM radio broadcasting. It is also used in telemetry, radar, seismic prospecting, and monitoring newborns for seizures via EEG, two-way radio systems, sound synthesis, magnetic tape-recording systems and some video-transmission systems. In radio transmission, an advantage of frequency modulation is that it has a larger signal-to-noise ratio and therefore rejects radio frequency interference better than an equal power amplitude modulation (AM) signal. For this reason, most music is broadcast over FM radio. Frequency modulation and phase modulation are the two complementary principal methods of angle modulation.

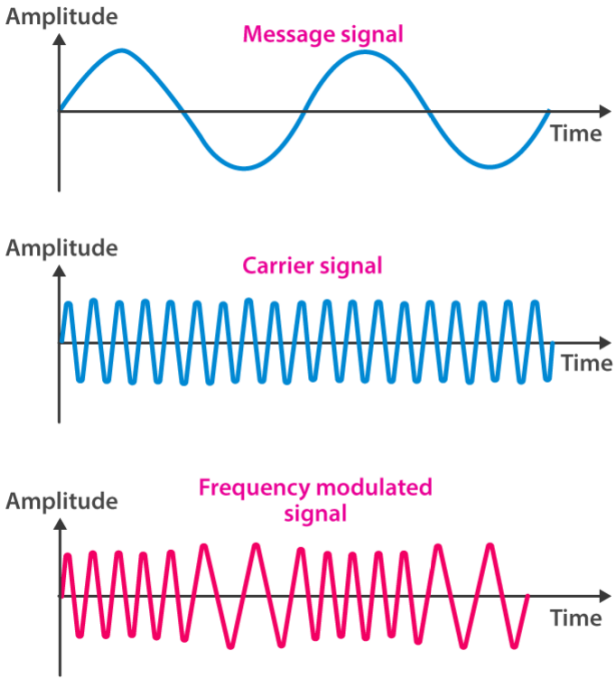


Figure 8.1 FM Modulation

Consider a message signal defined by S(t) = $A_m cos(2\pi f_m t)$. The instantaneous frequency of the FM signal is expressed as :

$$f(t) = f_c + \Delta f cos(2\pi f_m t) \tag{8.1}$$

where Δf = $k_f A_m$ is the maximum frequency deviation that occurs in the carrier frequency.

8.3.2 About Modulation index (β):

The FM modulation index is equal to the ratio of the frequency deviation to the modulating frequency. From the formula and definition of the modulation index, it can be seen that there is no term that includes the carrier frequency and this means that it is totally independent of the carrier frequency. Modulation index (β) is :

$$\beta = \frac{\Delta f}{f_m} = \frac{k_f A_m}{f_m} \tag{8.2}$$

The equation β is a dimensionless quantity since k_f has the units of $volt^{-1}\ second^{-1}$. The FM signal is given by :

$$S_{FM}(t) = A_c cos[(2\pi f_c t + \beta sin(2\pi f_m t)] \tag{8.3}$$

8.3.3 About Frequency DeModulation (FM) :

Demodulation is of three types :

- Frequency discrimination
- Phase Shift Discrimination
- Phase-Locked-Loop detector (PLL)

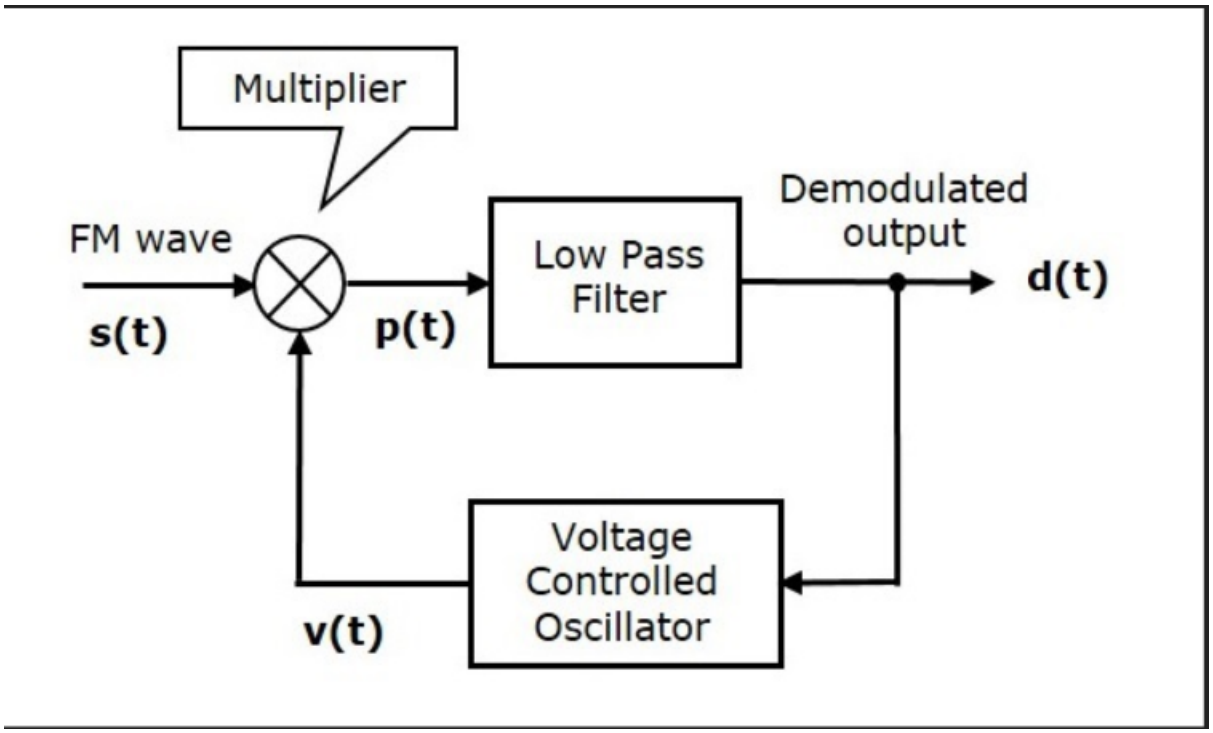


Figure 8.2 FM Demodulation

8.4 Code and Results

8.4.1 FM wave generation in time domain and frequency domain

```
% Name = Mohit Akhouri
% Roll no = 19UCC023
% SSC LAB Batch D1 - Monday ( 2-5 pm )

% This code will generate FM ( frequency modulated ) wave in time
 domain
% and frequency domain

t = linspace(0,1,500); % defining range for time axis
fc = 12000; % defining carrier frequency
fm = 4500; % defining message frequency
am = 1; % defininining message amplitude
ac = 1; % defining carrier amplitude
Beta1 = 1; % defining modulation index 1
Beta2 = 5; % defining modulation index 2
Beta3 = 7; % defining modulation index 3
Beta4 = 10; % defining modulation index 4
Kf1 = (Beta1*fm)/am; % calculating max. frequency deviation for beta 1
Kf2 = (Beta2*fm)/am; % calculating max. frequency deviation for beta 2
Kf3 = (Beta3*fm)/am; % calculating max. frequency deviation for beta 3
Kf4 = (Beta4*fm)/am; % calculating max. frequency deviation for beta 4

mt = am*cos(2*pi*fm*t); % defining message signal
ct = ac*cos(2*pi*fc*t); % defining carrier signal

% plotting the message signal and carrier signal
figure;
subplot(2,1,1);
plot(t,mt);
xlabel('time(t)->');
ylabel('m(t)->');
title('Message Signal m(t)');
grid on;
subplot(2,1,2);
plot(t,ct);
xlabel('time(t)->');
ylabel('c(t)->');
title('Carrier Signal c(t)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

FM1t = ac*cos(2*pi*fc*t + Beta1*sin(2*pi*fm*t)); % definig FM wave for
 beta 1
FM2t = ac*cos(2*pi*fc*t + Beta2*sin(2*pi*fm*t)); % definig FM wave for
 beta 2
FM3t = ac*cos(2*pi*fc*t + Beta3*sin(2*pi*fm*t)); % definig FM wave for
 beta 3
FM4t = ac*cos(2*pi*fc*t + Beta4*sin(2*pi*fm*t)); % definig FM wave for
 beta 4

fs = 2*fc; % defining sampling frequency
N = length(t); % defining N ( total samples )
```

Figure 8.3 Part 1 of Code for FM wave generation

```
ts = 1/fs; % defining sampling time
f=linspace(-fs/2,fs/2,N); % defining frequency interval

% plotting FM wave in time domain and frequency domain for beta 1
figure;
subplot(2,1,1);
plot(t,FM1t);
xlabel('time(t)->');
ylabel('X_{FM1}(t)->');
title('X_{FM1}(t) (Frequency modulated) signal for \beta = 1');
grid on;
subplot(2,1,2);
plot(f,abs(fftshift(fft(FM1t))));
xlabel('Frequency(Hz)->');
ylabel('X_{FM1}(\omega)->');
title('Magnitude plot of X_{FM1}(\omega) in frequency domain for \beta
 = 1');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% plotting FM wave in time domain and frequency domain for beta 2
figure;
subplot(2,1,1);
plot(t,FM2t);
xlabel('time(t)->');
ylabel('X_{FM2}(t)->');
title('X_{FM2}(t) (Frequency modulated) signal for \beta = 5');
grid on;
subplot(2,1,2);
plot(f,abs(fftshift(fft(FM2t))));
xlabel('Frequency(Hz)->');
ylabel('X_{FM2}(\omega)->');
title('Magnitude plot of X_{FM2}(\omega) in frequency domain for \beta
 = 5');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% plotting FM wave in time domain and frequency domain for beta 3
figure;
subplot(2,1,1);
plot(t,FM3t);
xlabel('time(t)->');
ylabel('X_{FM3}(t)->');
title('X_{FM3}(t) (Frequency modulated) signal for \beta = 7');
grid on;
subplot(2,1,2);
plot(f,abs(fftshift(fft(FM3t))));
xlabel('Frequency(Hz)->');
ylabel('X_{FM3}(\omega)->');
title('Magnitude plot of X_{FM3}(\omega) in frequency domain for \beta
 = 7');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 8.4 Part 2 of Code for FM wave generation

```
% plotting FM wave in time domain and frequency domain for beta 4
figure;
subplot(2,1,1);
plot(t,FM4t);
xlabel('time(t)->');
ylabel('X_{FM4}(t)->');
title('X_{FM4}(t) (Frequency modulated) signal for \beta = 10');
grid on;
subplot(2,1,2);
plot(f,abs(fftshift(fft(FM4t))));
xlabel('Frequency(Hz)->');
ylabel('X_{FM2}(\omega)->');
title('Magnitude plot of X_{FM4}(\omega) in frequency domain for \beta
 = 10');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 8.5 Part 3 of Code for FM wave generation

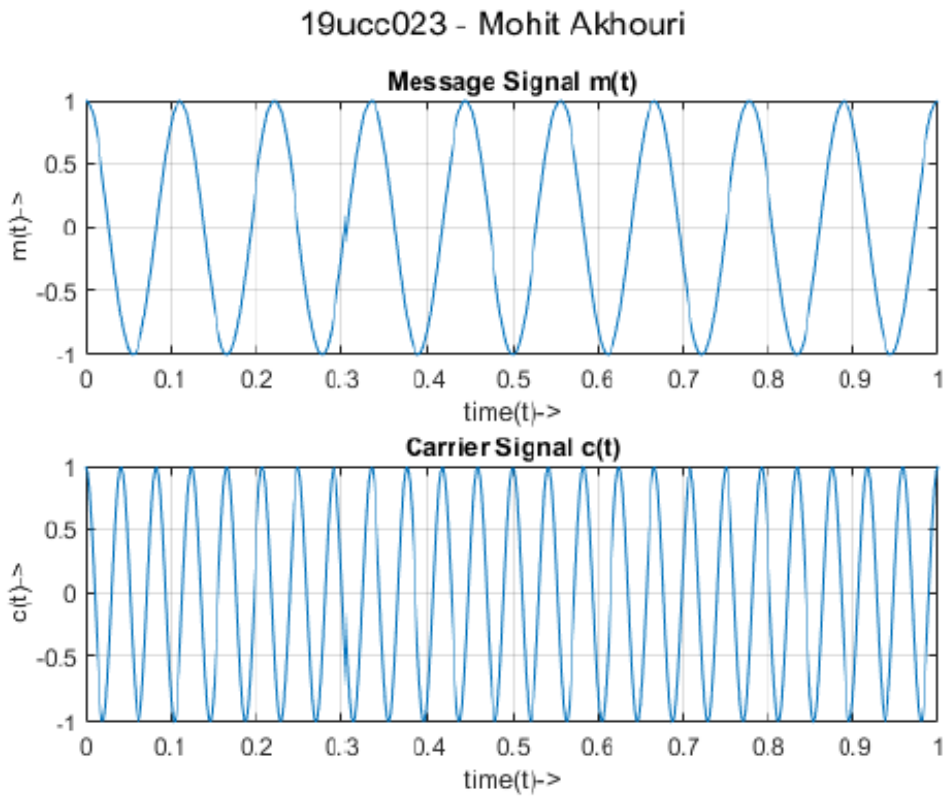


Figure 8.6 Message signal and carrier signal in time domain

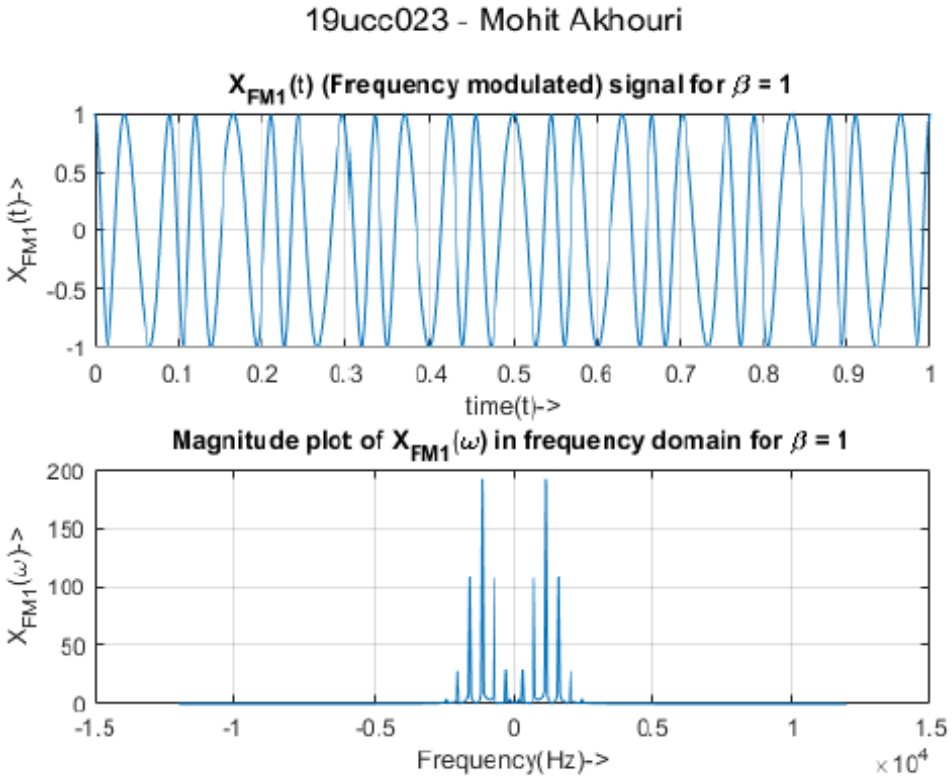


Figure 8.7 FM wave for $\beta = 1$

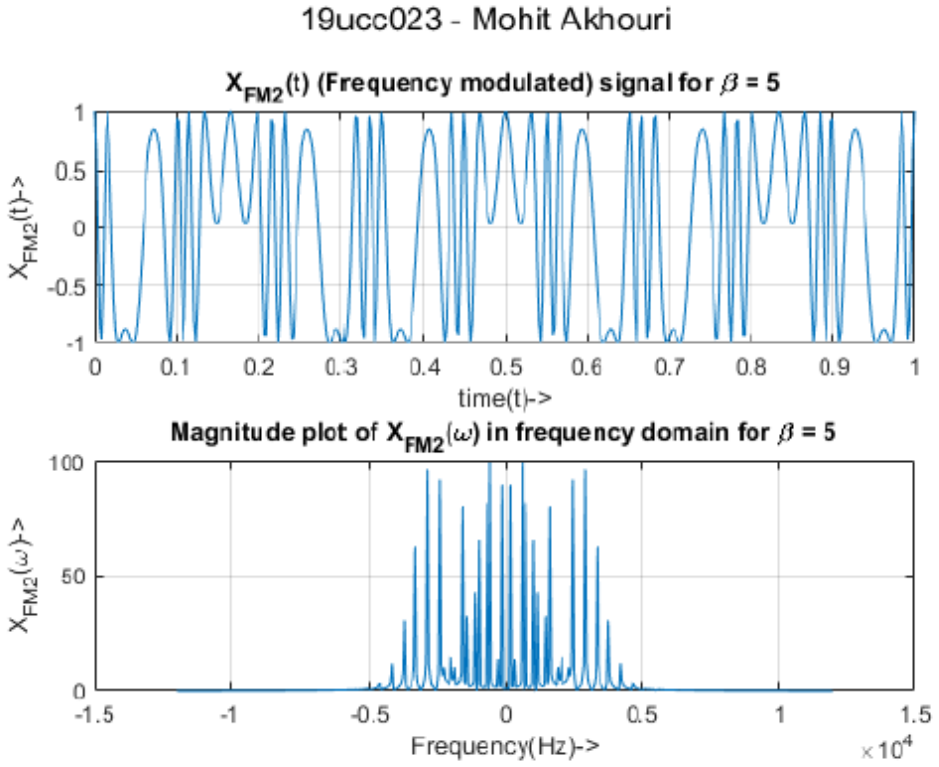


Figure 8.8 FM wave for $\beta = 5$

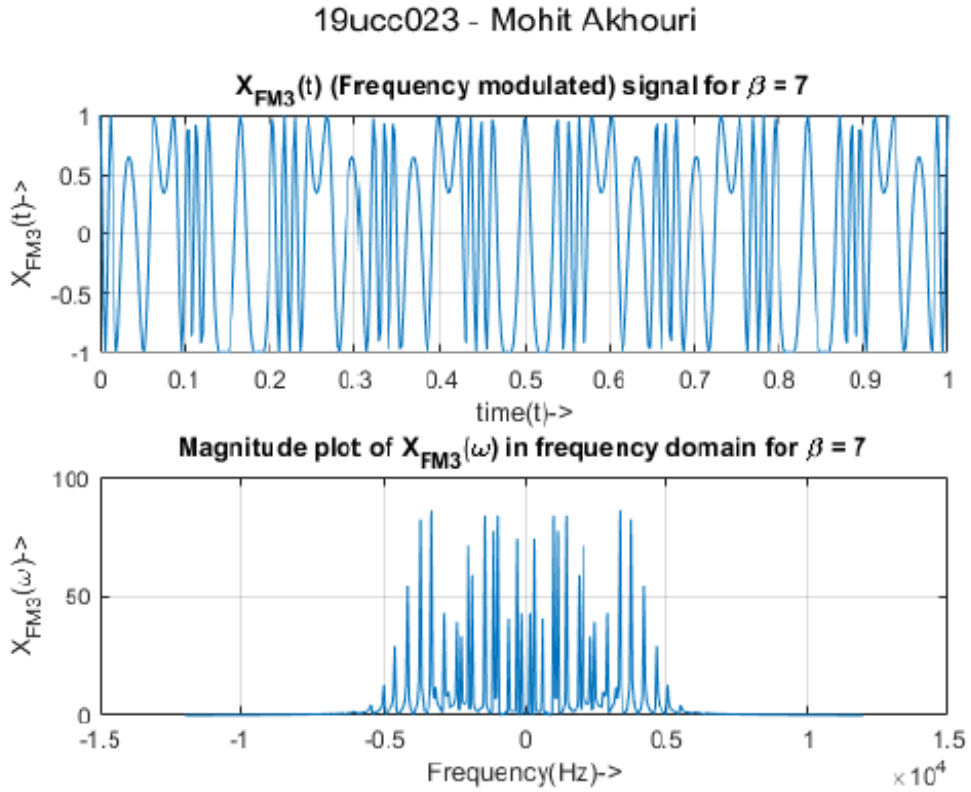


Figure 8.9 FM wave for $\beta = 7$

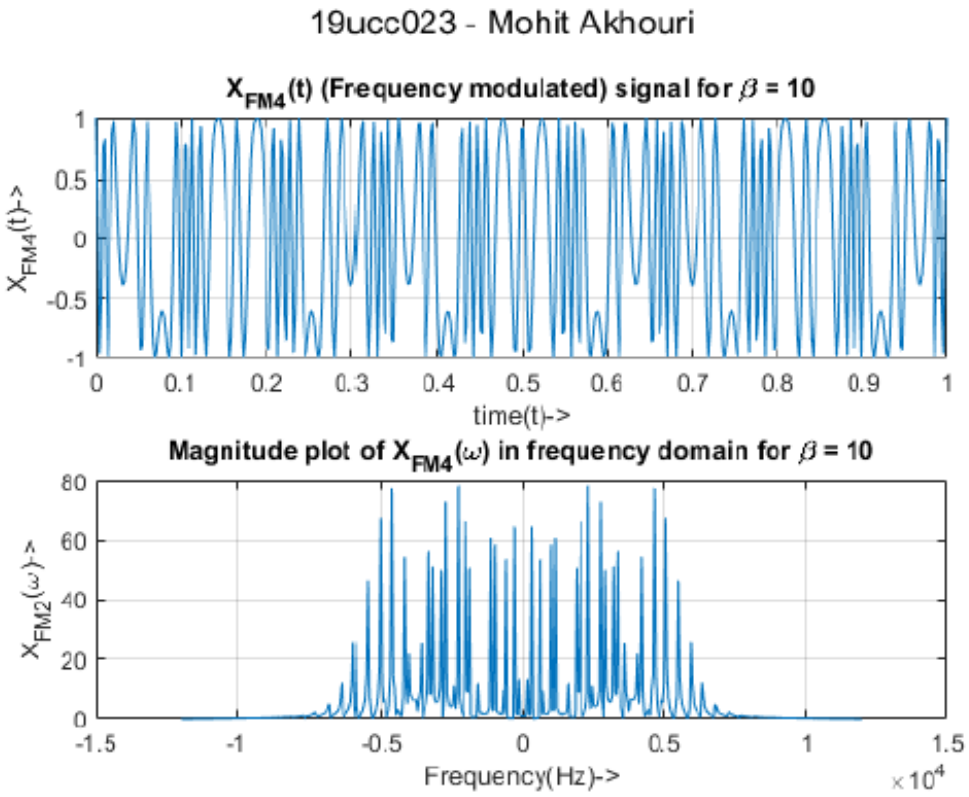


Figure 8.10 FM wave for $\beta = 10$

8.5 Conclusion

In this experiment, I studied about the FM wave modulation and demodulation for different value of modulation index (β) and how to demodulate them using inbuilt MATLAB function **fmdemod**.