# 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 Coordinator
Dr. Navneet Upadhyay


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)

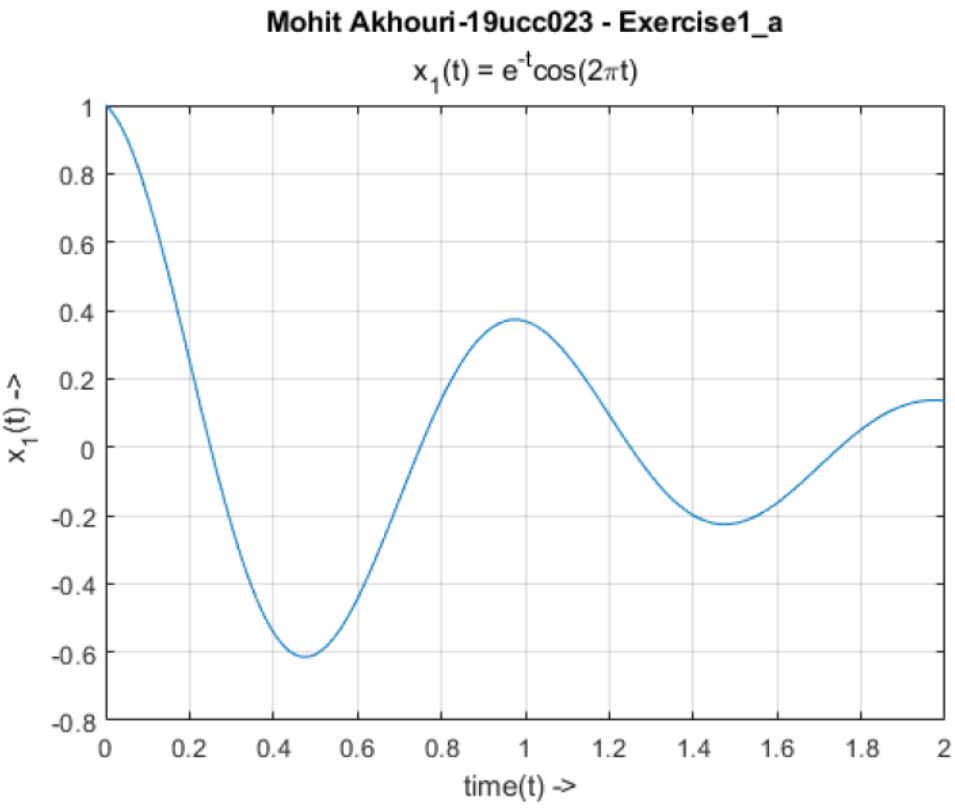



**Figure 1.3** Graph for Exercise 1a - Plotting $x_1$(t)

```
% 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)

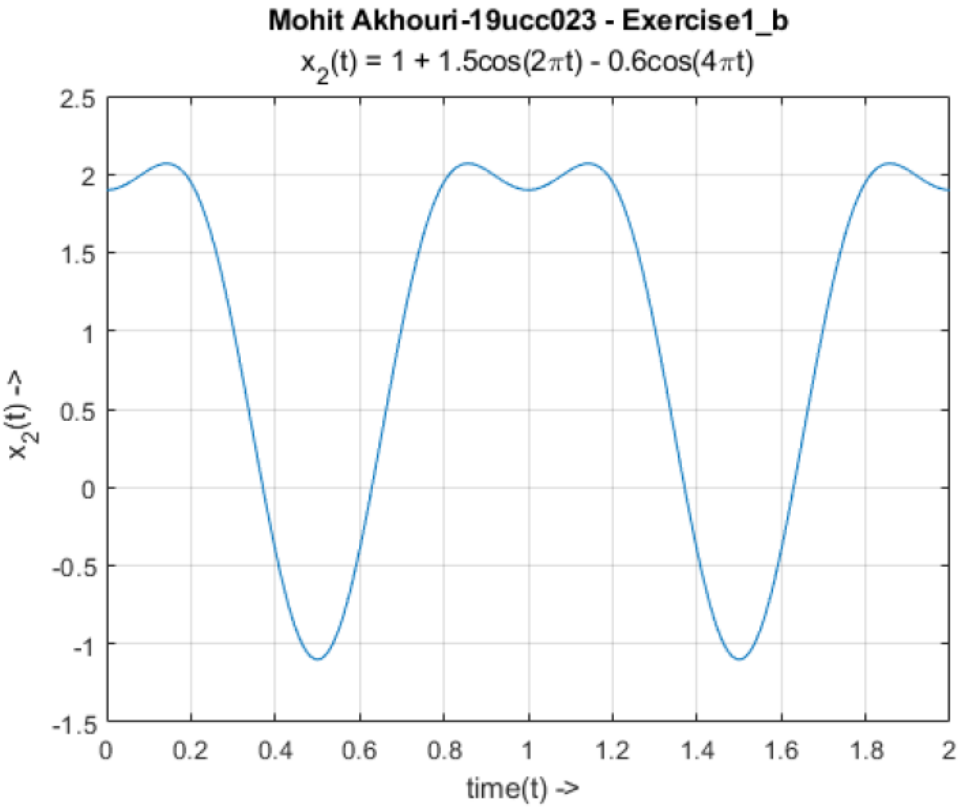



**Figure 1.5** Graph for Exercise 1b - Plotting $x_2$(t)

```
% 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)

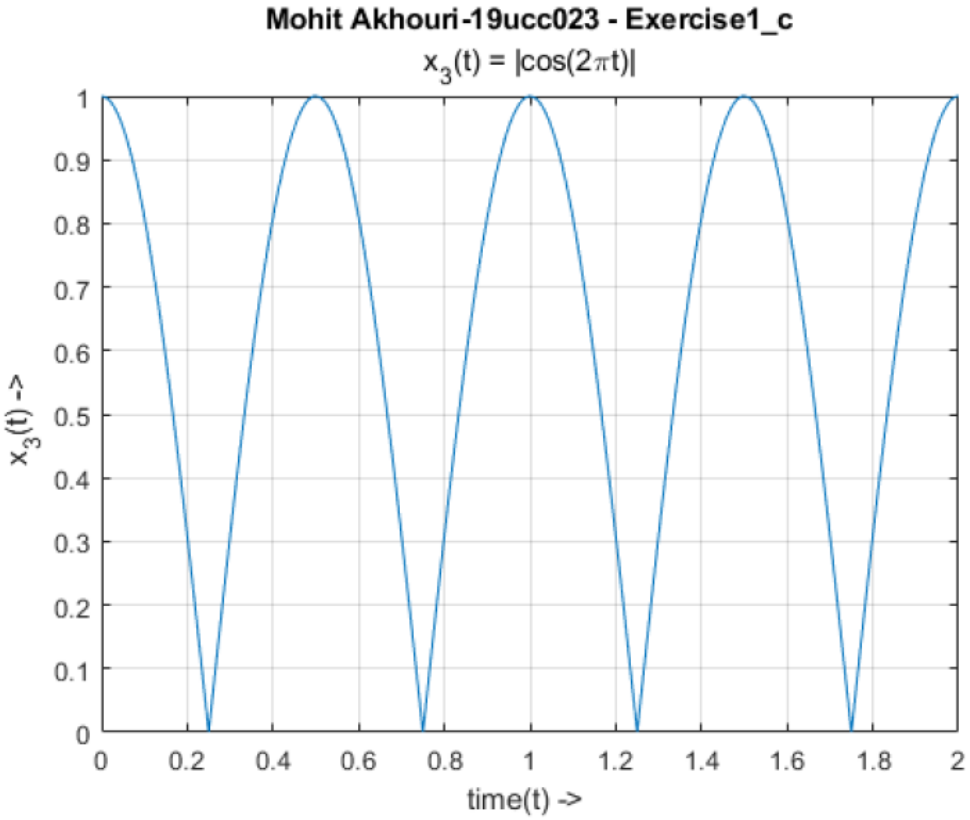



**Figure 1.7** Graph for Exercise 1c - Plotting $x_3$(t)

### 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);
```

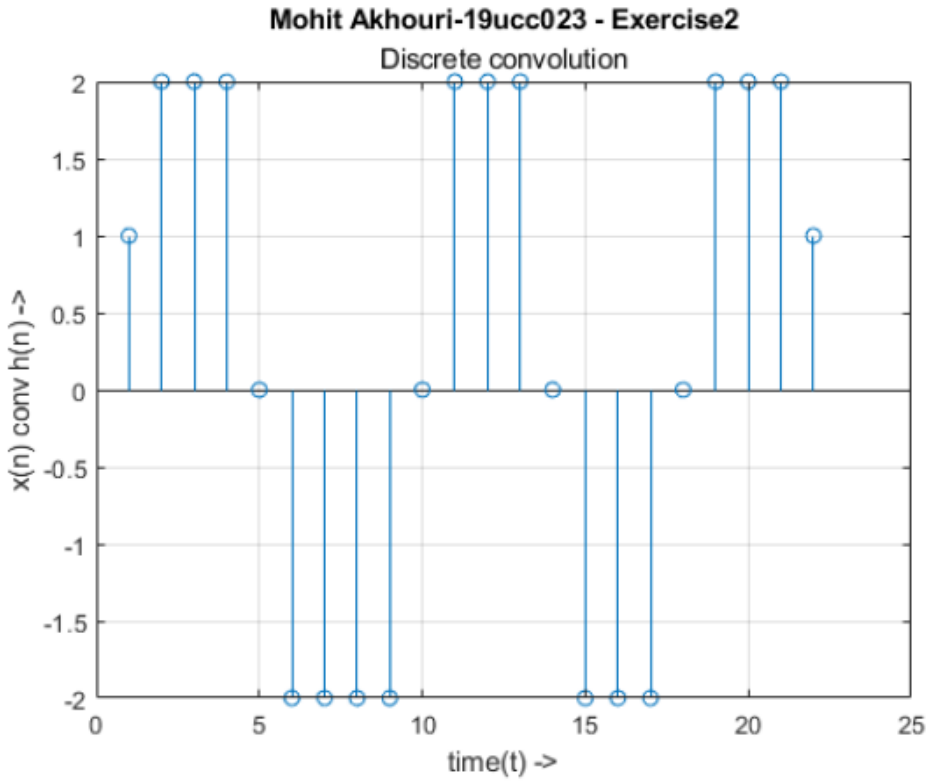



**Figure 1.9** Main code and graph for convolution x and h1

```
% 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);
```

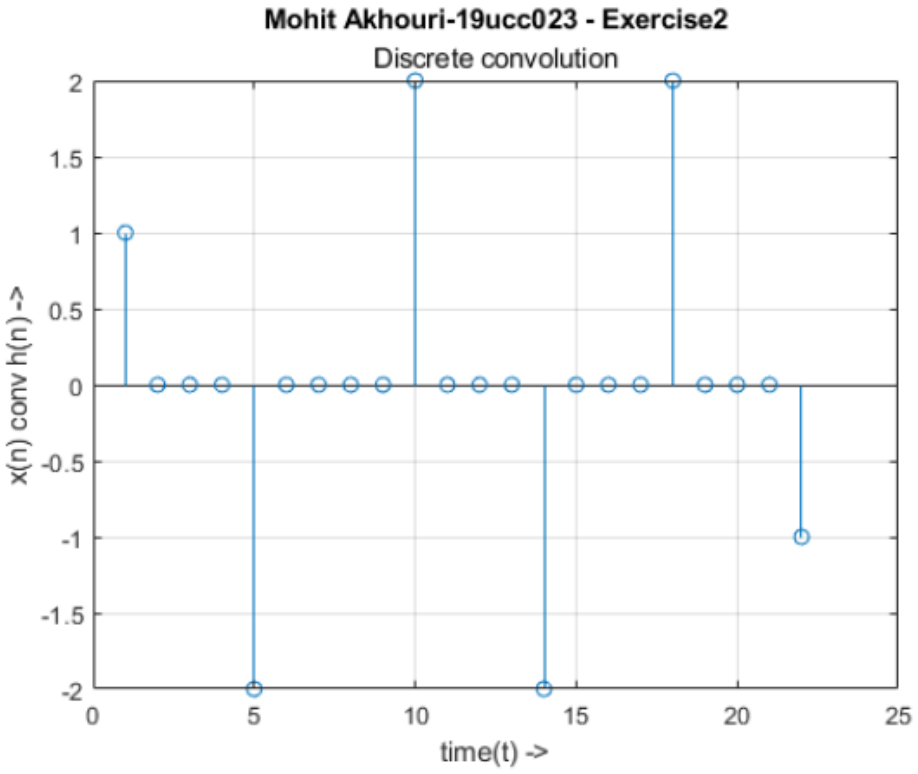



**Figure 1.10** Main code and graph for convolution x and h2

### 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

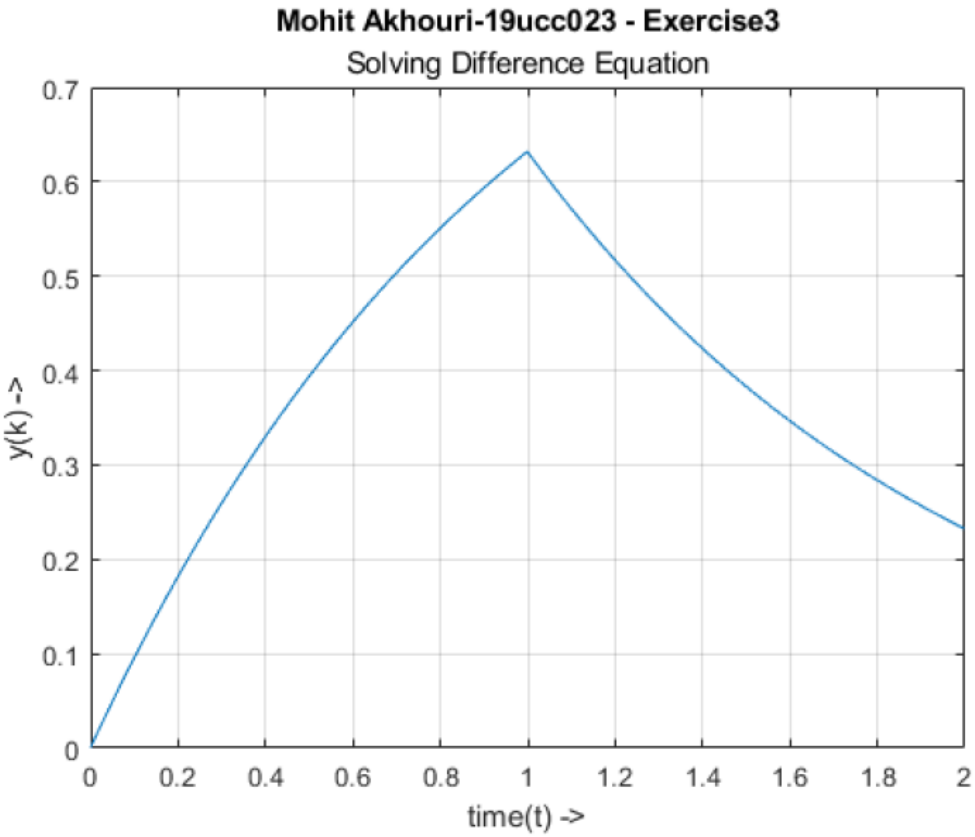



**Figure 1.12** Graph of Exercise 3

### 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

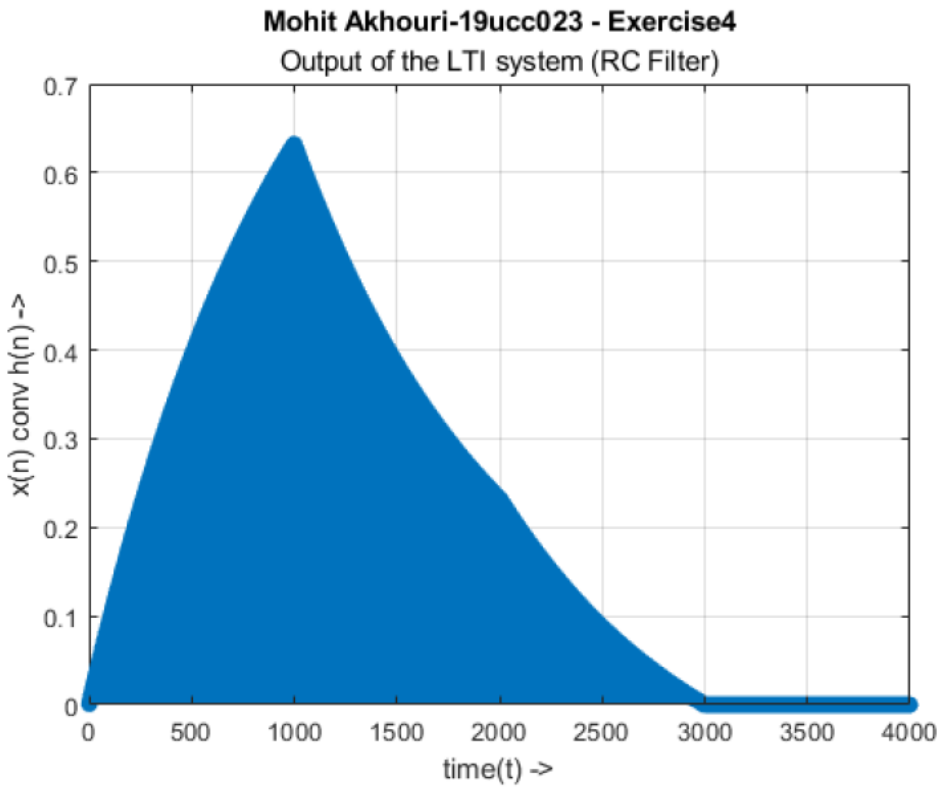



**Figure 1.14** Graph of Exercise 4

## 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**.

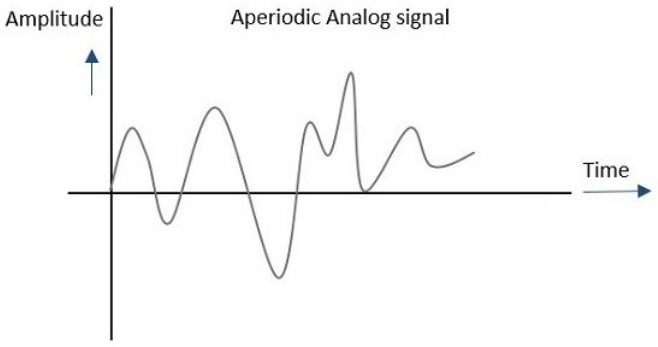



**Figure 2.1** Aperiodic Analog Signal

### 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 $\omega_0$. 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*

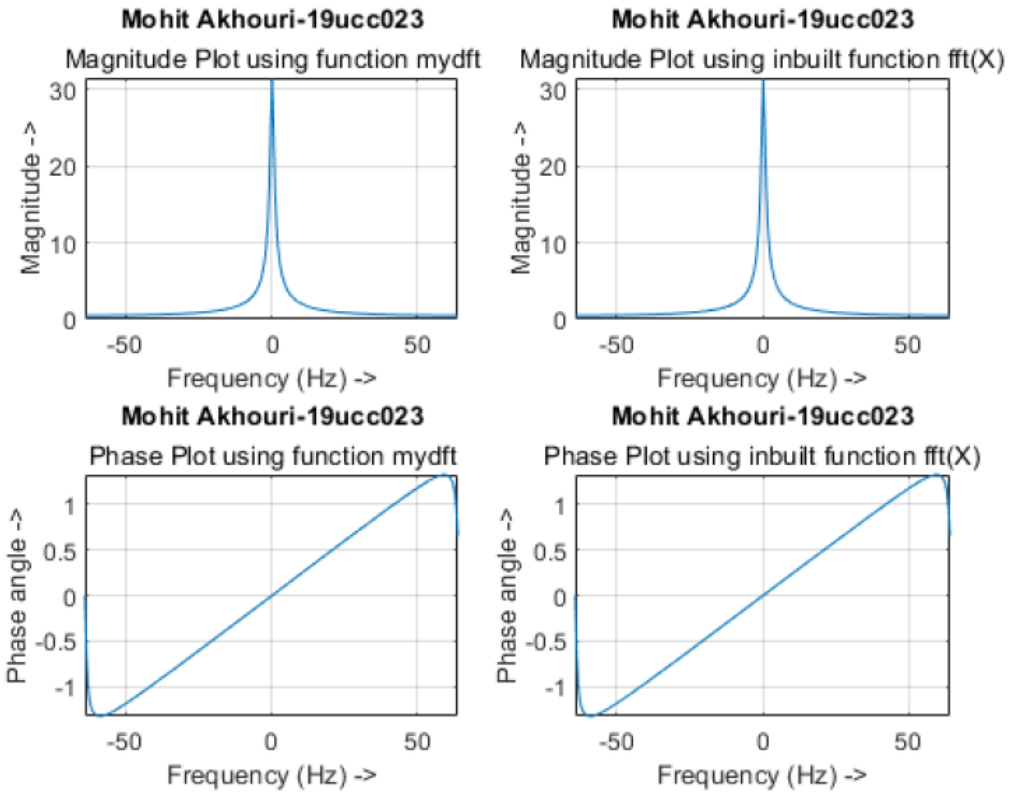
**Figure 2.2** Code for the function mydft designed for finding DFT of x

```
% 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*

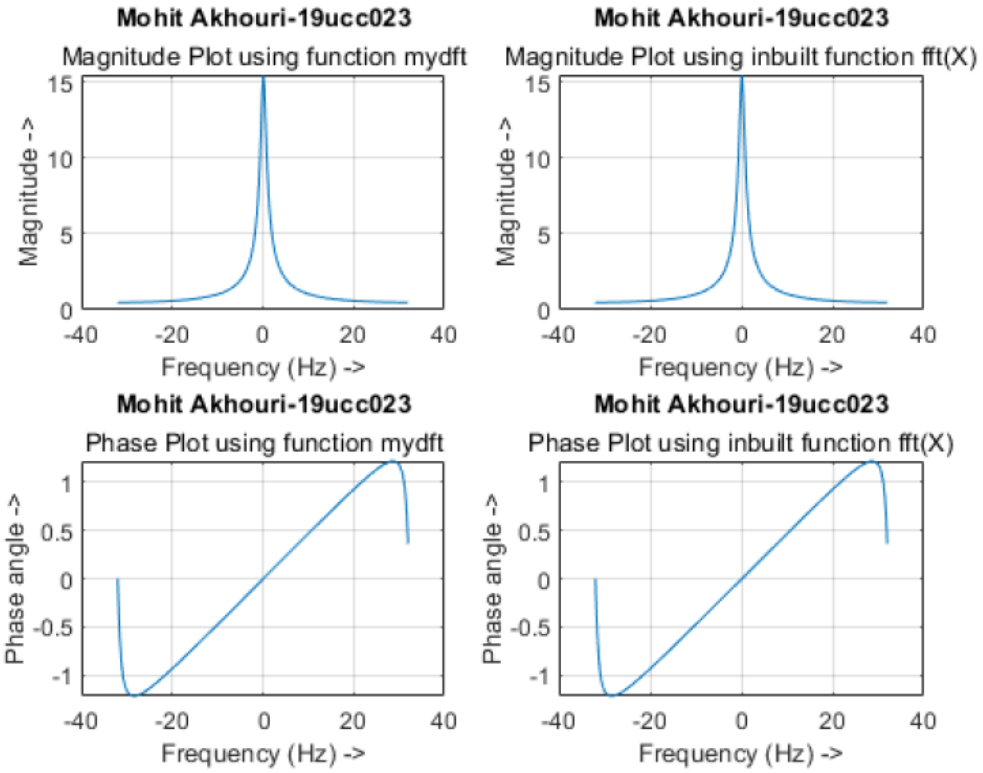
**Figure 2.4** Graph of calculated DFT when $T_0$ = 4 sec and $T_s$ = 1/64 sec

```
% 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

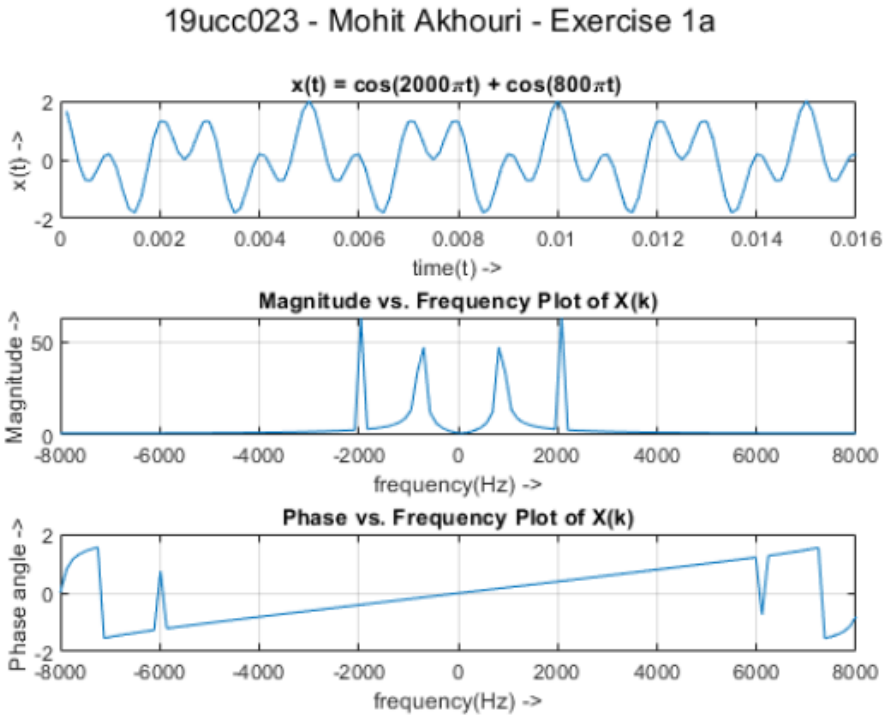



**Figure 3.3** Graph of calculated DFT of signal x(t) with N=128 and $F_s$ = 8000 Hz

### 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

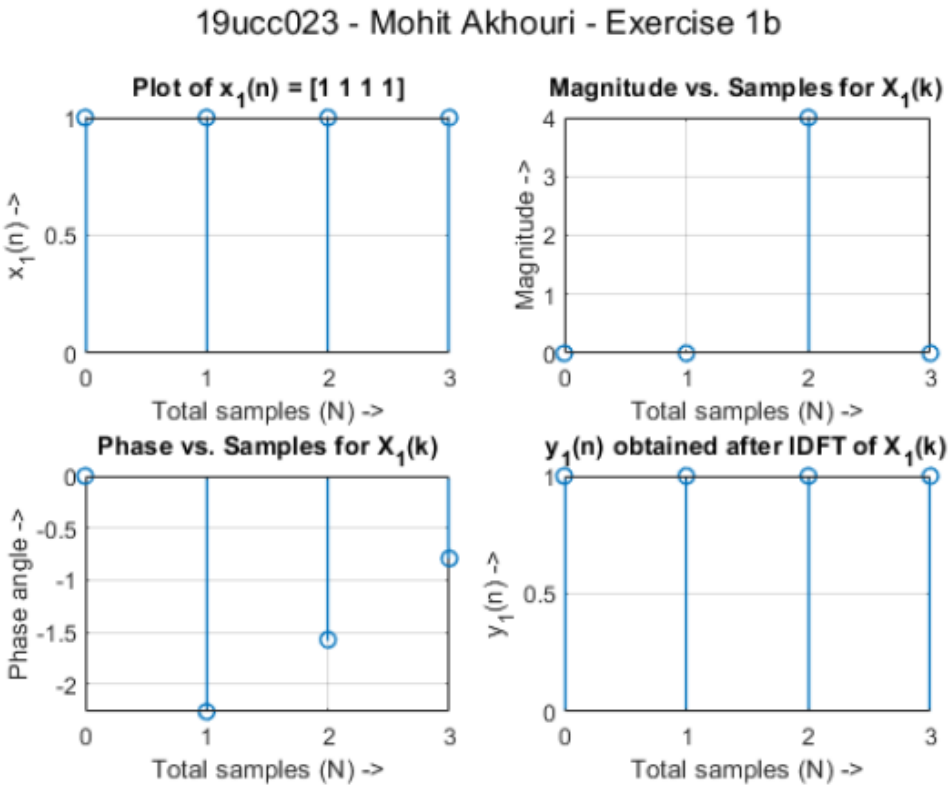



**Figure 3.6** Graph of calculated IDFT of signal $x_1[n]$ with N=4

### 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

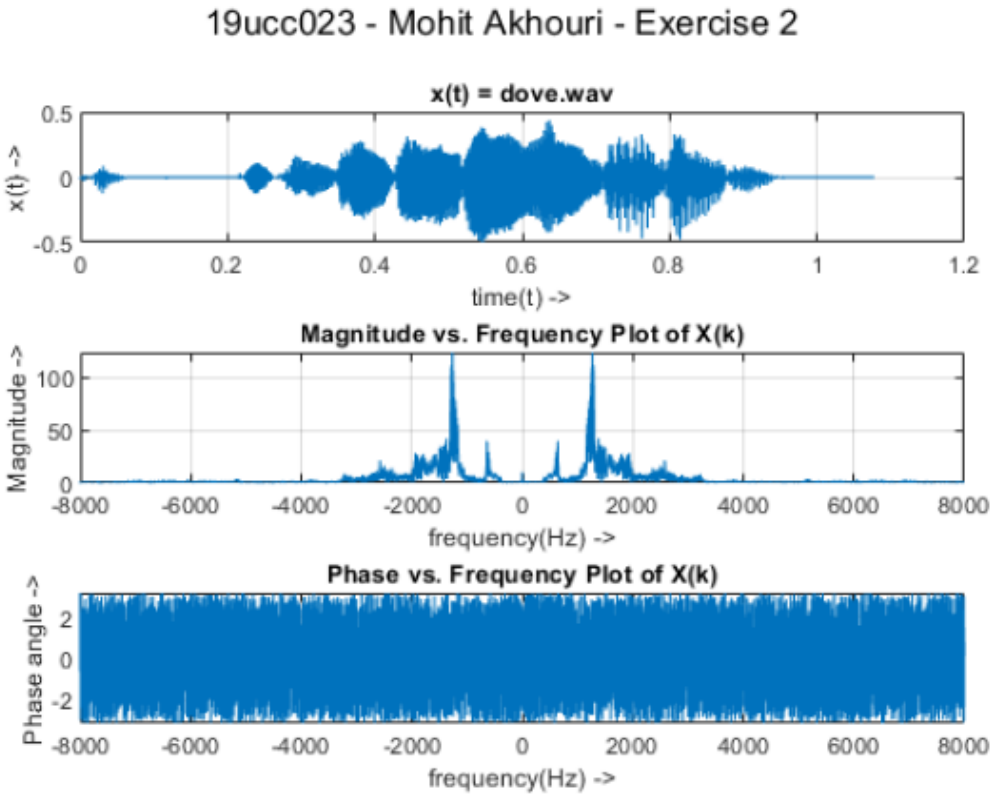



**Figure 3.8** Graph of Magnitude and phase spectra of speech signal (dove.wav)

### 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)

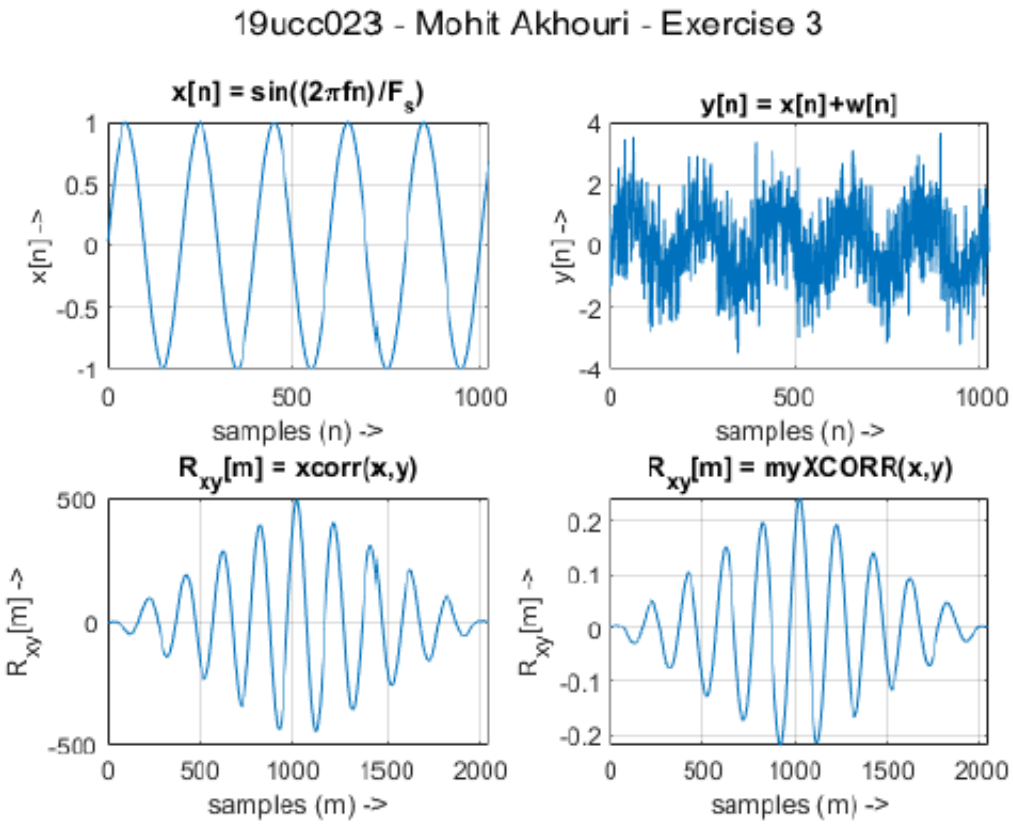



**Figure 3.11** Graph for performed CCF of x[n] and y[n] using myXCORR(x,y)

## 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.

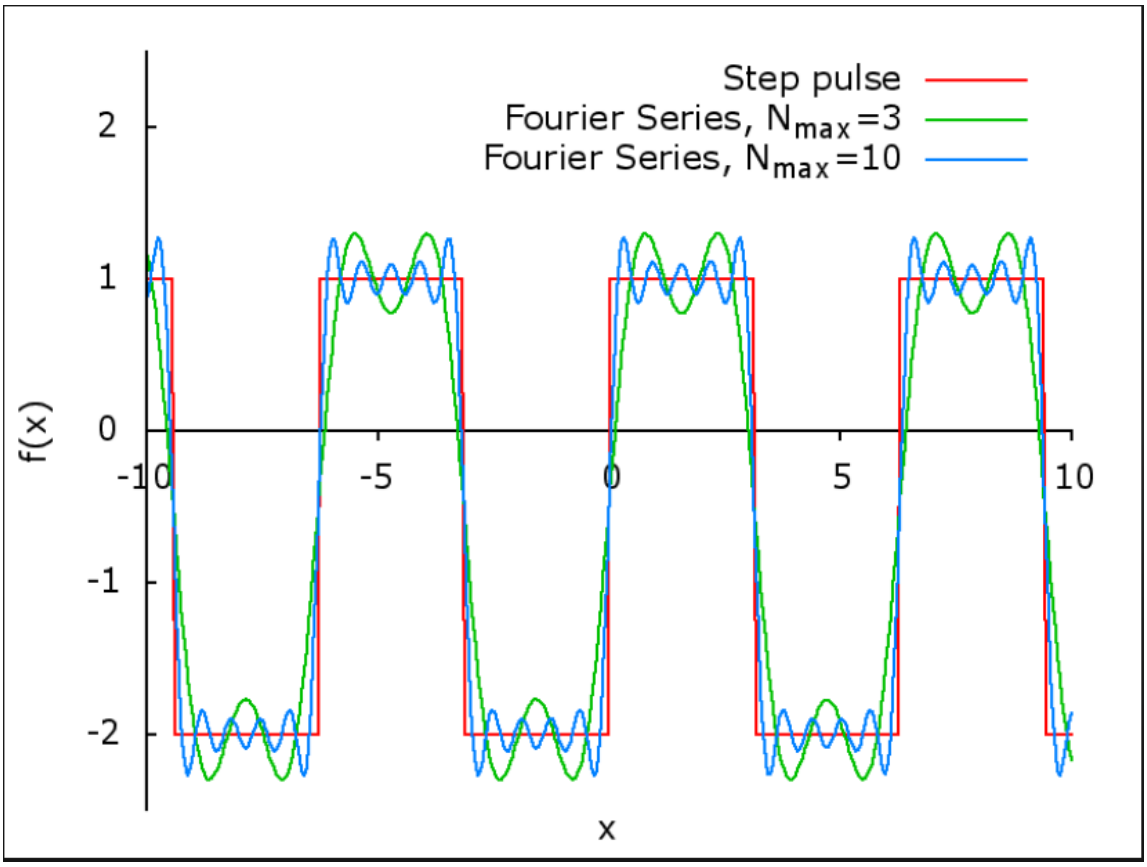



**Figure 4.1** Fourier Series of a step pulse

### 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 :

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

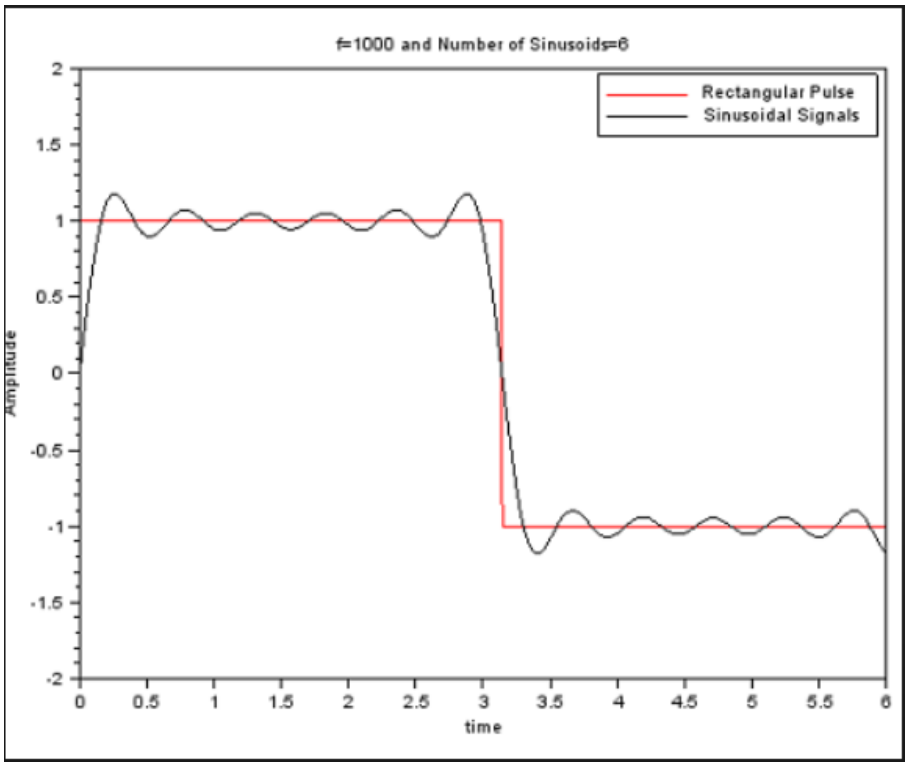



**Figure 4.2** Illustration of Gibb's phenomenon

## 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 $x_2(t)$

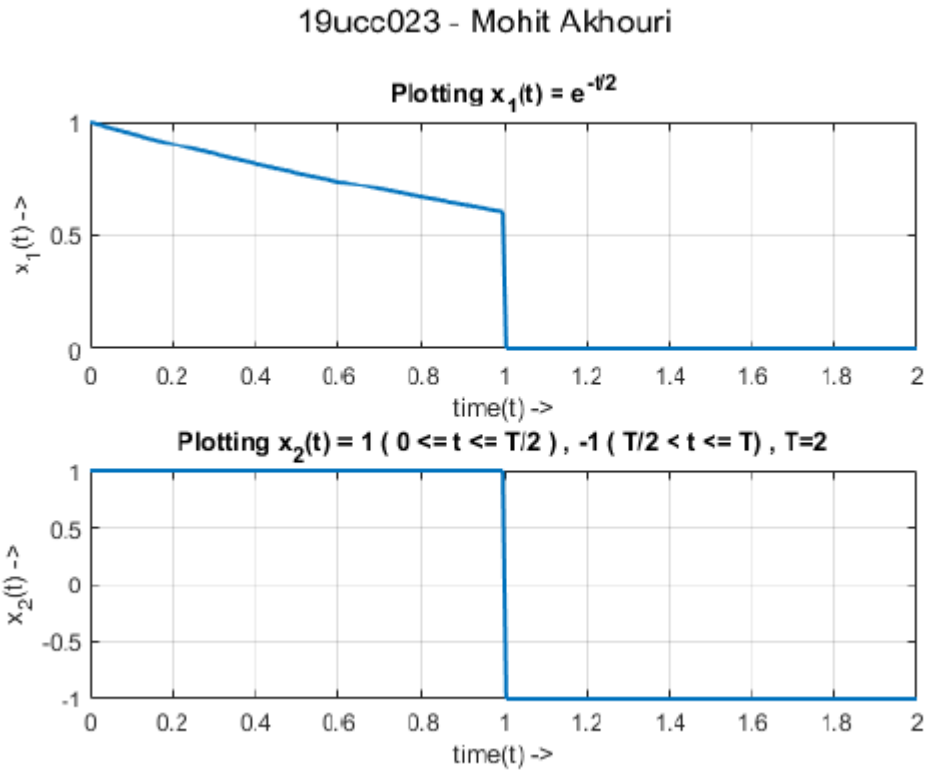



**Figure 4.7** Plot of the periodic signals $x_1(t)$ and $x_2(t)$

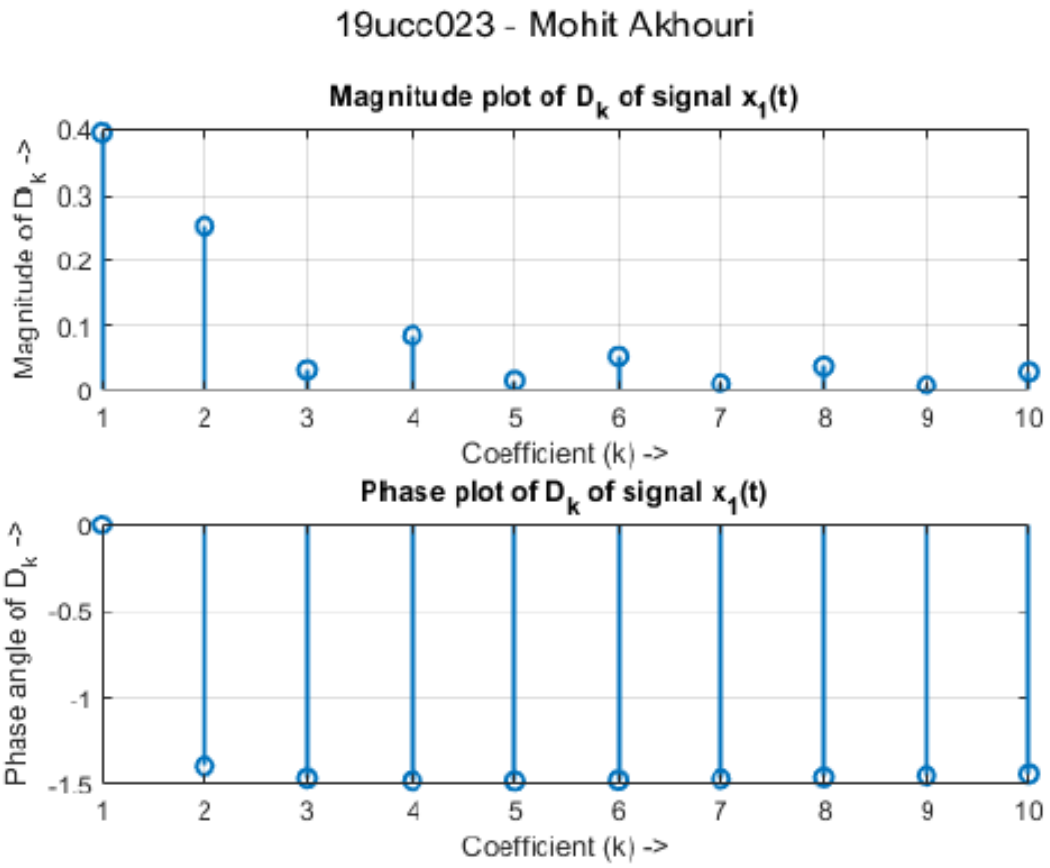



**Figure 4.8** Plot of the magnitude and phase spectra of $D_k$ for signal $x_1(t)$

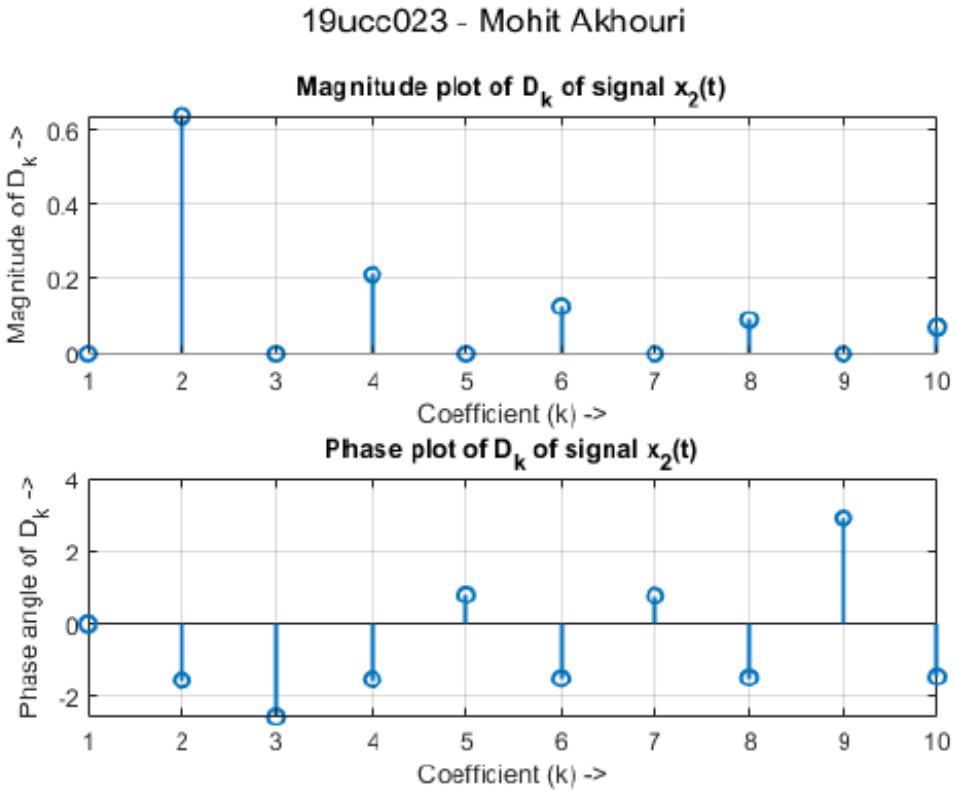



**Figure 4.9** Plot of the magnitude and phase spectra of $D_k$ for signal $x_2(t)$

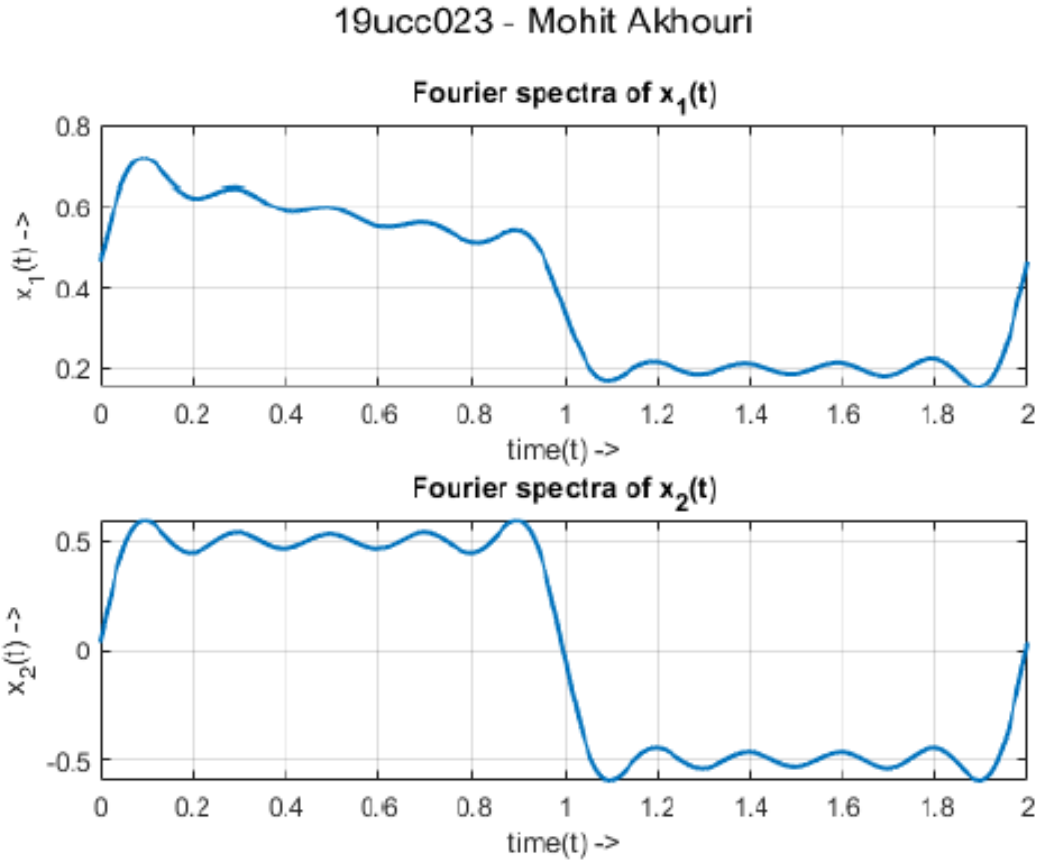



**Figure 4.10** Plot of the Fourier spectra for signals $x_1(t)$ and $x_2(t)$ with 10 fourier series coefficients

### 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 $x_2(t)$

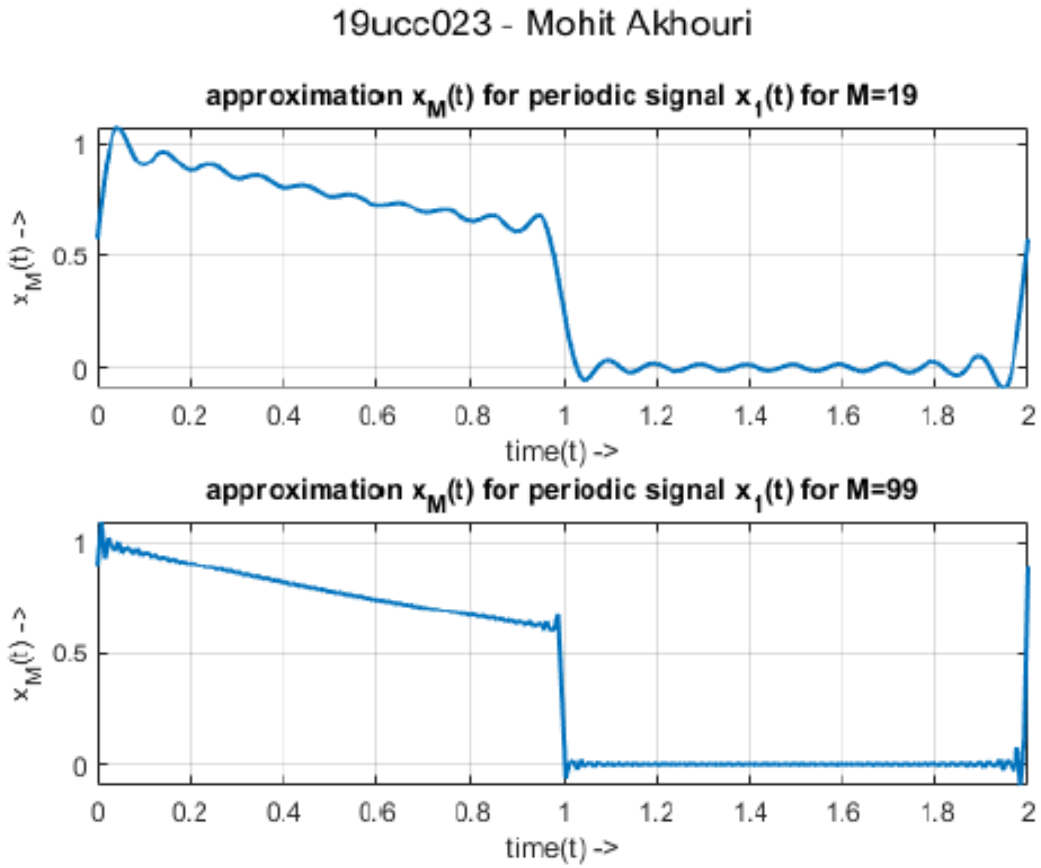



**Figure 4.15** Graph of illustration of Gibb's phenomenon on signal $x_1(t)$ for M=19 and M=99

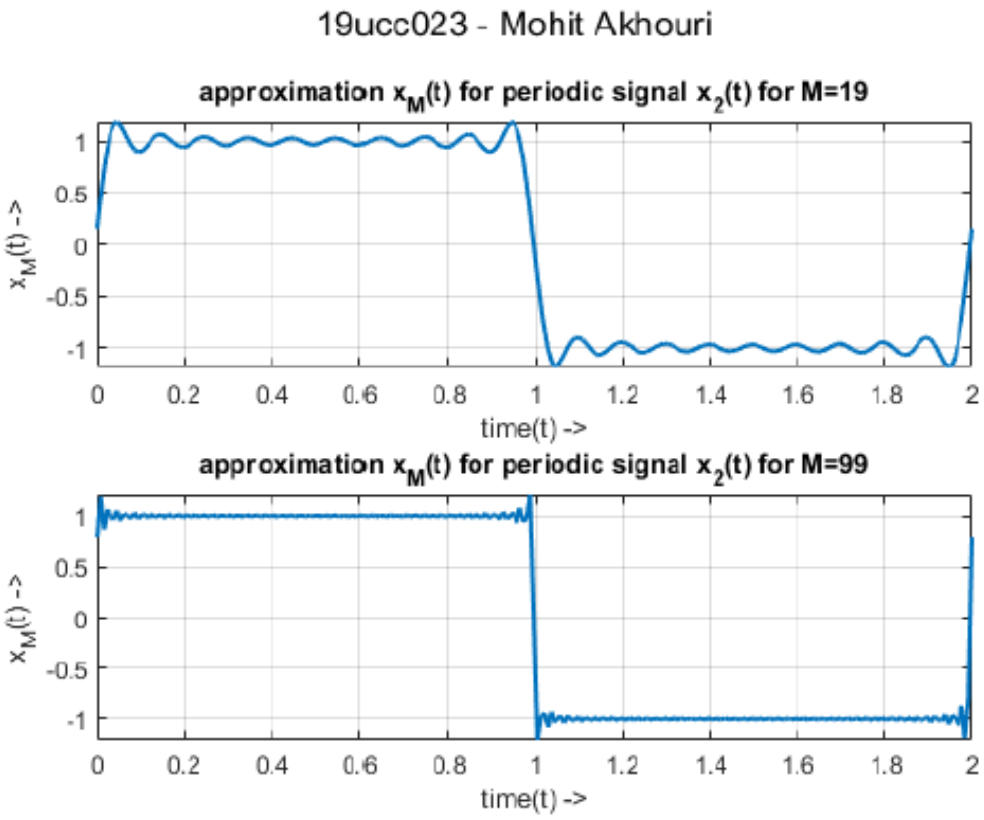



**Figure 4.16** Graph of illustration of Gibb's phenomenon on signal $x_2(t)$ for M=19 and M=99

## 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.