VISVESVARAYA TECHNOLOGICAL UNIVERSITY,
Jnana Sangama, Belagavi – 590018

KLS VISHWNATHRAO DESHPANDE INSTITUTE OF TECHNOLOGY,
Haliyal – 581 329, Uttara Kannada

LABORATORY MANUAL

Course Title	:	Digital Signal Processing Lab
Course Code	:	BEC502
Year / Semester	:	2nd Year / 5th Sem
Academic Year	:	2025-26
Course In-Charge	:	Prof.Sudheendra Yalagur

Department of Electronics and Communication Engineering

Signature of the Faculty with Date 4/7/25

HoD 04.07.2025

Head of the Department
Dept. of Electronic & Communication Engg.
K.L.S V.D.I.T. HALIYAL (U.K.)

KLS Vishwanathrao Deshpande Institute of Technology

(Accredited by NAAC with "A" Grade)

(Approved by AICTE, New Delhi, Affiliated to VTU, Belagavi)

(Recognized Under Section 2(f) by UGC, New Delhi)

Udyog Vidya Nagar, Haliyal - 581 329, Dist.: Uttara Kannada

www.klsvdit.edu.in | principal@klsvdit.edu.in | hodece@klsvdit.edu.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

College Vision and Mission Statements

Vision

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

Mission

1. To augment innovative pedagogy & kindle quest for interdisciplinary learning & to enhance conceptual understanding.
2. To build competence, professional ethics & develop entrepreneurial thinking.
3. To strengthen industry institute partnership &explore global collaborations.
4. To inculcate culture of socially responsible citizenship.
5. To focus on holistic &sustainable development.

Department Vision,Mission,PEOs and PSOs Statements

Vision

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

Mission

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

PEOs

PEO 1:	To be successful in professional career in electronics, communication and allied industries by acquiring the knowledge in the fundamentals of Electronics and Communication Engineering principles and professional skills.
PEO 2:	To be in a position to analyze real life problems and design socially accepted and economically feasible solutions in the respective fields.
PEO 3:	To exhibit good communication skills in their professional career, lead a team with good leadership traits and good interpersonal relationship with the members related to other engineering streams.
PEO 4:	To involve themselves in lifelong learning and professional development by pursuing higher education and participation in research and development activities.
PEO 5:	To demonstrate professional and ethical responsibilities towards their profession, society and the environment.

PSOs

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

Course Title	**:**	**Digital Signal Processing Laboratory**		
Course Code	**:**	**BEC502**		
Year / Semester	**:**	**3rd Year / 5th Semester**		
Academic Year	**:**	**2025-26**		
Syllabus	**:**	**Sl. No.**	**Content**	**Page No.**
		1	Program to generate the following discrete time signals. a) Unit sample sequence, b)Unit step sequence, c) Exponential sequence, d)Sinusoidal sequence, e) Random sequence	1
		2	Program to perform the following operations on signals. a) Signal addition, b) Signal multiplication, c)Scaling, d) Shifting, e)Folding	5
		3	Program to perform convolution of two given sequences (without using built-in function) and display the signals.	10
		4	Consider a causal system y(n) = 0.9y(n-1)+x(n). a) Determine H(z) and sketch its pole zero plot. b) Plot |H(ejω)| and ∠ H(ejω) c) Determine the impulse response h(n).	13
		5	Computation of N point DFT of a given sequence (without using built-in function) and to plot the magnitude and phase spectrum.	19
		6	Using the DFT and IDFT, compute the following for any two given sequences a)Circular convolution b) Linear convolution	22
		7	Verification of Linearity property, circular time shift property & circular frequency shift property of DFT.	25
		8	Develop decimation in time radix-2 FFT algorithm without using built-in functions.	30
		9	Design and implementation of digital low pass FIR filter using a window to meet the given specifications.	34
		10	Design and implementation of digital high pass FIR filter using a window to meet the given specifications.	38
		11	Design and implementation of digital IIR Butterworth low pass filter to meet the given specifications.	42
		12	Design and implementation of digital IIR Butterworth high pass filter to meet the given specifications.	45
		13	Virtual Lab No-1 Study of FIR filter design using window method (Low-pass and High pass filter).	48
		14	Virtual Lab No-2 Study of IIR (Infinite Impulse Response) filter.	54
CLOs	**:**	• Preparation: To prepare students with fundamental knowledge/overview in the field of Digital Signal Processing. • Core Competence: To equip students with a basic foundation of Signal Processing by delivering the basics of Discrete Fourier Transforms, their properties, efficient computations & the design of digital filters.		

Cos	**:**	**CO1:**	Analyse the different types of signals and systems used in digital signal processing.				
		CO2:	Compute the response of an LTI system using time and frequency domain techniques.				
		CO3:	Develop algorithms for the efficient computations of DFT and IDFT.				
		CO4:	Design of digital FIR filters for the given specifications using different window methods.				
		CO5:	Design of digital IIR digital filters using bilinear transformation method.				
CO / PO Mapping	**:**	**Cos**	**CO1**	**CO2**	**CO3**	**CO4**	**CO5**
		POs / PSOs	1,2,5/2	1,2,5/2	1,2,5/2	1,2,5/2	1,2,5/2
Course In-Charge	**:**	**Prof. Sudheendra Yalagur**					

CO-PO Mapping Matrix

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
1	3	2			2									1
2	3	2			2									1
3	3	2			2									1
4	3	2			2									1
5	3	2			2									1

Experiment No. 1

Program to generate the following Discrete Time Signals

a) Unit impulse sequence b) Unit step sequence c) Exponential sequence d) Sinusoidal sequence e) Ramp sequence

Aim: To generate the following discrete time signals.

a) Unit sample sequence, b) Unit step sequence, c) Exponential sequence, d) Sinusoidal sequence, e) Random sequence

Theory:

Discrete-Time Signals:

In signal processing, discrete-time signals are sequences of numbers that are defined only at specific, discrete time instants. The value of a signal at any given time instant can represent various physical quantities. Below, we discuss the theory behind the five different types of discrete-time signals:

1) Unit Sample Sequence (Impulse Sequence)

Definition:

The **unit sample sequence**, also known as the **impulse sequence**, is a sequence denoted by δ[n] that takes the value 1 at n=0n = 0n=0 and is zero at all other time instances. It is defined mathematically as:

$$\delta[n] = \begin{cases} 1, & \text{if } n = 0 \\ 0, & \text{if } n \neq 0 \end{cases}$$

b) Unit Step Sequence

Definition:

The **unit step sequence**, denoted by u[n], is a sequence that is 0 for n<0 and 1 for n≥0. It is expressed as:

$$u[n] = \begin{cases} 0, & \text{if } n < 0 \\ 1, & \text{if } n \geq 0 \end{cases}$$

c) Exponential Sequence

Definition:

An **exponential sequence** is a sequence where each term is a power of a constant base. It is defined as:

$$x[n] = A \cdot r^n$$

Where:

- A is the amplitude (a constant),
- r is the base of the exponential (which may be a real or complex number),
- n is the discrete-time index.

d) Sinusoidal Sequence

Definition:

A **sinusoidal sequence** is a sequence whose values follow a sine or cosine function. It is mathematically defined as:

$$x[n] = A \cdot \sin(\omega n + \phi)$$

Where:

- A is the amplitude,
- ω is the angular frequency (related to the period),
- n is the discrete time index,
- ϕ is the phase shift.

e) Random Sequence

Definition:

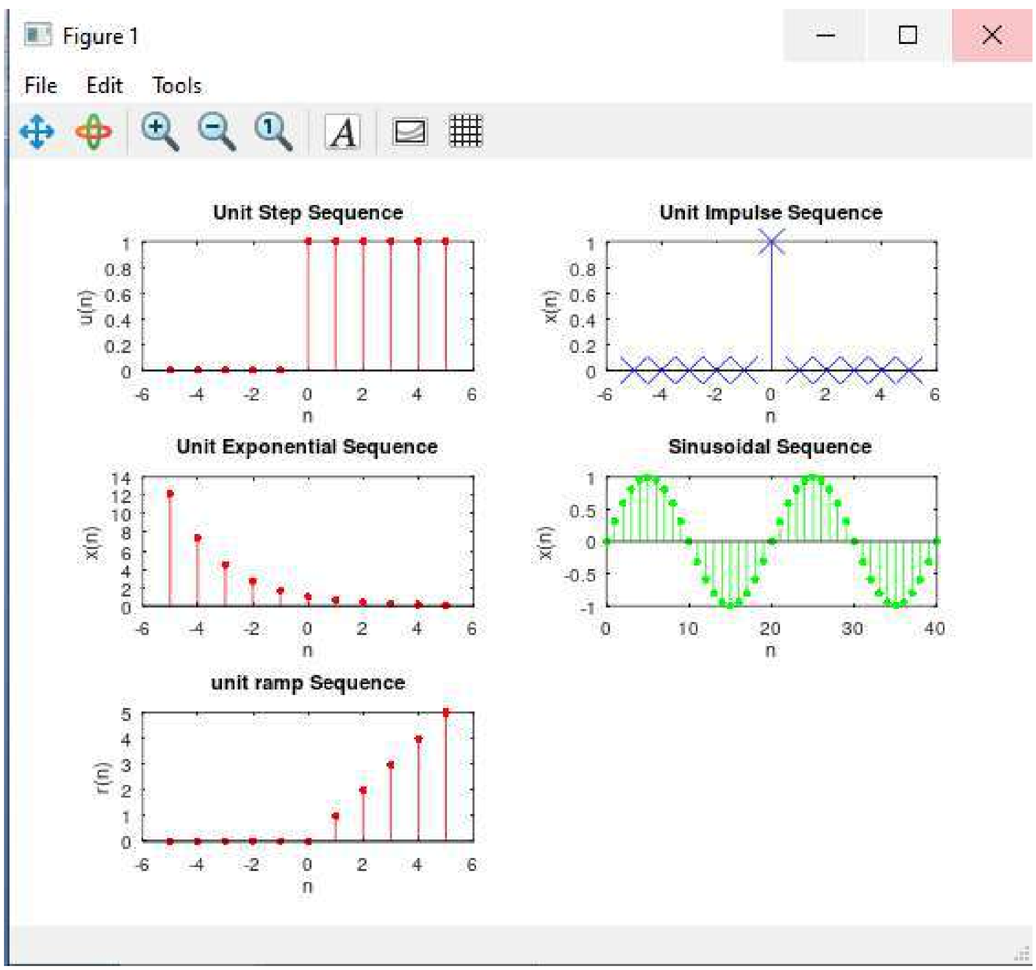
A **random sequence** is a sequence of values that are randomly generated according to some probability distribution. In mathematical terms, a random sequence x[n] is a sequence where each value x[n] is drawn from a probability distribution, such as Gaussian (normal) distribution, uniform distribution, or any other.

Octave Code:

```
%Generation of Unit Step Sequence
clc;
n=-5:1:5
x=(n>=0);
subplot(3,2,1)
stem(n,x, 'r.-', 'MarkerSize', 10)
xlabel('n')
ylabel('u(n)')
title('Unit Step Sequence')

%Generation of Unit Impulse Sequence
clc;
n=-5:1:5
x=(n==0);
subplot(3,2,2)
stem(n,x, 'bx', 'MarkerSize', 10)
xlabel('n')
ylabel('x(n)')
title('Unit Impulse Sequence')

%Generation of Exponential Sequence
clc;
n=-5:5
a=-0.5;
x=exp(a*n);
subplot(3,2,3)
stem(n,x, 'r.', 'MarkerSize', 10)
xlabel('n')
ylabel('x(n)')
title('Unit Exponential Sequence')

%Generation of Sinusoidal Sequence
n=0:1:40
y=sin(0.1*pi*n);
subplot(3,2,4)
stem(n,y, 'g.', 'MarkerSize', 10)
xlabel('n')
ylabel('x(n)')
title('Sinusoidal Sequence')

%Generation of unit ramp Sequence
```

```
n=-5:5;
x=n.*(n>=0);
subplot(3,2,5)
stem(n,x, 'r.', 'MarkerSize', 10)
xlabel('n')
ylabel('r(n)')
title('unit ramp Sequence')
```

Output:



Experiment No. 2

Program to perform the following operations on signals

a) Signal addition b) Signal multiplication c) Scaling d) Shifting e) Folding

Aim: To perform the following operations on signals
a) Signal addition b) Signal multiplication c) Scaling d) Shifting e) Folding

Theory:

In signal processing, various operations can be performed on signals to manipulate or transform them. These operations help in analyzing, processing, or synthesizing signals for different applications, such as communication systems, audio processing, and control systems. Below, we discuss the theory behind five common signal operations: Signal Addition, Signal Multiplication, Scaling, Shifting, and Folding.

a) Signal Addition

Definition:

Signal addition refers to the operation where two or more signals are added together point by point to form a new signal. If x[n] and y[n] are two discrete-time signals, their sum z[n] is given by:

$$z[n] = x[n] + y[n]$$

b) Signal Multiplication

Definition:

Signal multiplication involves multiplying two signals point by point to create a new signal. If x[n] and y[n] are two discrete-time signals, their product z[n] is given by:

$$z[n] = x[n] \cdot y[n]$$

c) Scaling

Definition:

Scaling is the operation where a signal is multiplied by a constant scalar value. If x[n] is a discrete-time signal and A is a constant, the scaled signal y[n] is given by:

$$y[n] = A \cdot x[n]$$

d) Shifting

Definition:

Shifting refers to the operation where a signal is shifted in time by a constant amount. There are two types of shifting: **time shift** and **frequency shift**. In the context of discrete-time signals, if x[n] is the original signal and k is a constant, the time-shifted signal y[n] is defined as:

Right Shift: y[n] = x [n−k]

Left Shift: y[n] = x [n+k]

e) Folding (Time Reversal)

Definition:

Folding, or time reversal, is the operation where a signal is reversed in time. If x[n] is the original signal, its folded version y[n] is given by:

$$y[n] = x[-n]$$

Octave Code:

```
clc;
n1=-4:1:4
a=[1 2 3 4 3 2 0 -3 -1]
n2=-2:1:3
b=[-2 1 0 4 6 -5]
subplot(3,3,1)
stem(n1,a)
title('Sample Sequence 1');
xlabel('n');
ylabel('a[n]');
subplot(3,3,2)
stem(n2,b)
title('Sample Sequence 2');
xlabel('n');
ylabel('b[n]');
u=min(min(n1),min(n2))
v=max(max(n1),max(n2))
w=u:1:v
z1=[]
```

```
temp=1;
for i=1:length(w)
   if (w(i)< min(n1)||w(i)>max(n1));
      z1=[z1 0];
   else
      z1=[z1 a(temp)]
      temp=temp+1;
   end
end
z2=[]
temp=1;
for i=1:length(w)
   if (w(i)< min(n2)||w(i)>max(n2))
      z2=[z2 0]
   else
      z2=[z2 b(temp)]
      temp=temp+1;
   end
end

%Signal Addition

z=z1+z2
subplot(3,3,3)
stem(w,z)
title('Signal Addition');
xlabel('n');
ylabel('Sum[n]');

%Signal multiplication
z=z1.*z2
subplot(3,3,4)
stem(w,z)
title('Signal multiplication');
xlabel('n');
ylabel('Mul[n]');
```

```
%Folding Sequence
clc;
n=-4:1:4
x=[-1 2 3 1 -2 5 2 -2 5]
y=fliplr(x)
subplot(3,3,5)
stem(n,x)
title('Sample Sequence');
xlabel('n');
ylabel('X[n]');
subplot(3,3,6)
stem(n,y)
title('Folding Sequence');
xlabel('n');
ylabel('Y[n]');

%Time Shifting
m=n-2
subplot(3,3,7)
stem(m,x)
title('Time Shifting');
xlabel('n');
ylabel('X[M]');

%Time scaling
a=2.5
scal_x=a.*x
subplot(3,3,8)
stem(n1,scal_x)
title('Time scaling');
xlabel('n');
ylabel('X[N]');
```

Output:

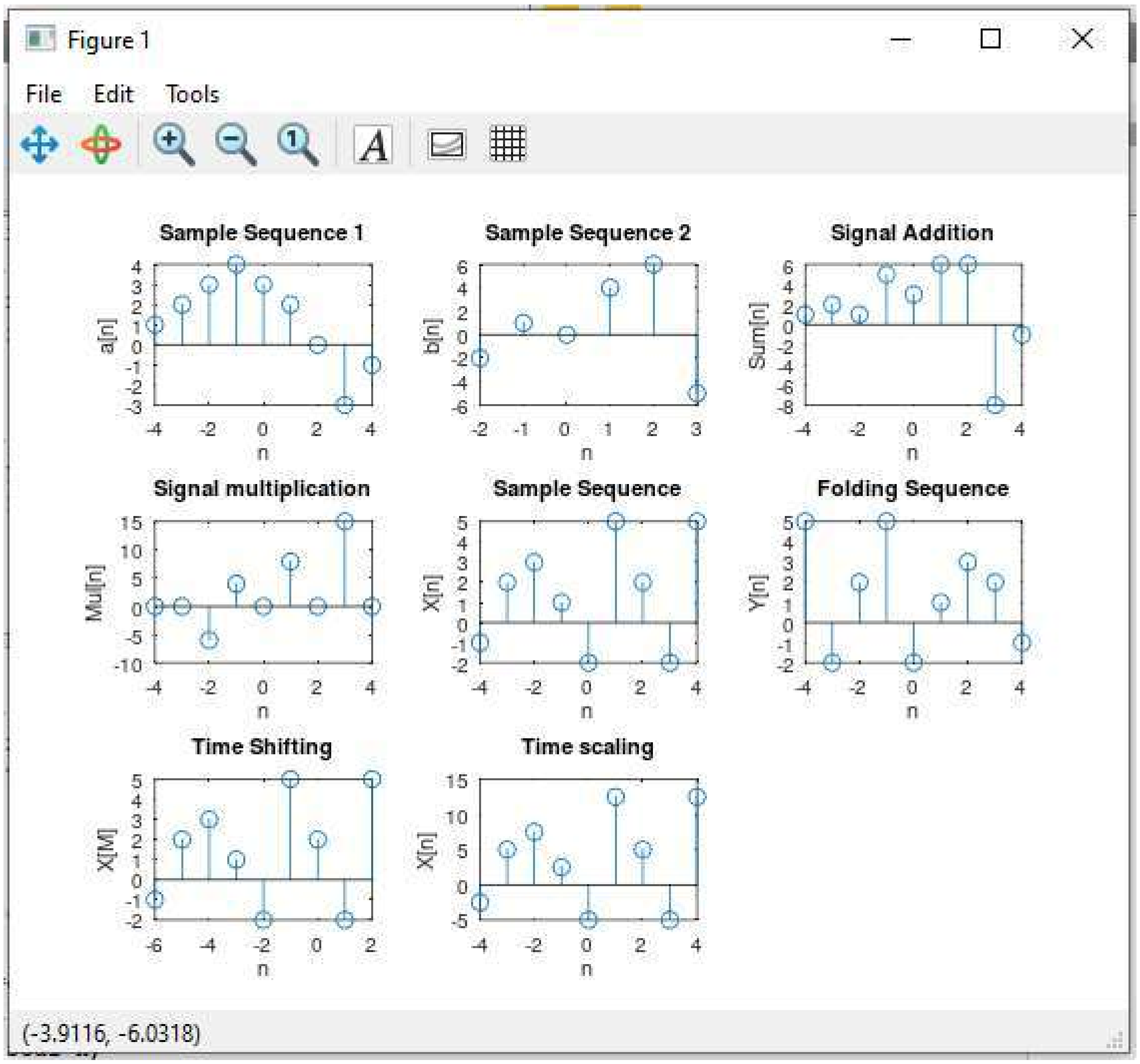


Figure 1
File Edit Tools
Sample Sequence 1
a[n]
n
Sample Sequence 2
b[n]
Signal Addition
Sum[n]
Signal multiplication
Mul[n]
Sample Sequence
X[n]
Folding Sequence
Y[n]
Time Shifting
X[M]
Time scaling
(-3.9116, -6.0318)

Experiment No. 3

Program to perform convolution of two given sequences (without using built-in function) and display the signals

Aim: To perform convolution of two given sequences (without using built-in function) and display the signals.

Theory:

Convolution of Two Sequences:

Convolution is a mathematical operation that combines two sequences to produce a third sequence. It is a fundamental tool in signal processing, especially in the analysis and design of linear systems. In the context of discrete-time signals, the convolution of two sequences x[n] and h[n] results in a new sequence y[n], which represents the output of a system when h[n] is the system's impulse response and x[n] is the input signal.

The convolution operation is denoted as:

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

Where:

- x[n] is the input sequence,
- h[n] is the impulse response sequence (also referred to as the filter),
- y[n] is the output sequence (result of the convolution),
- k is the summation index, which runs over the entire range of the sequences.

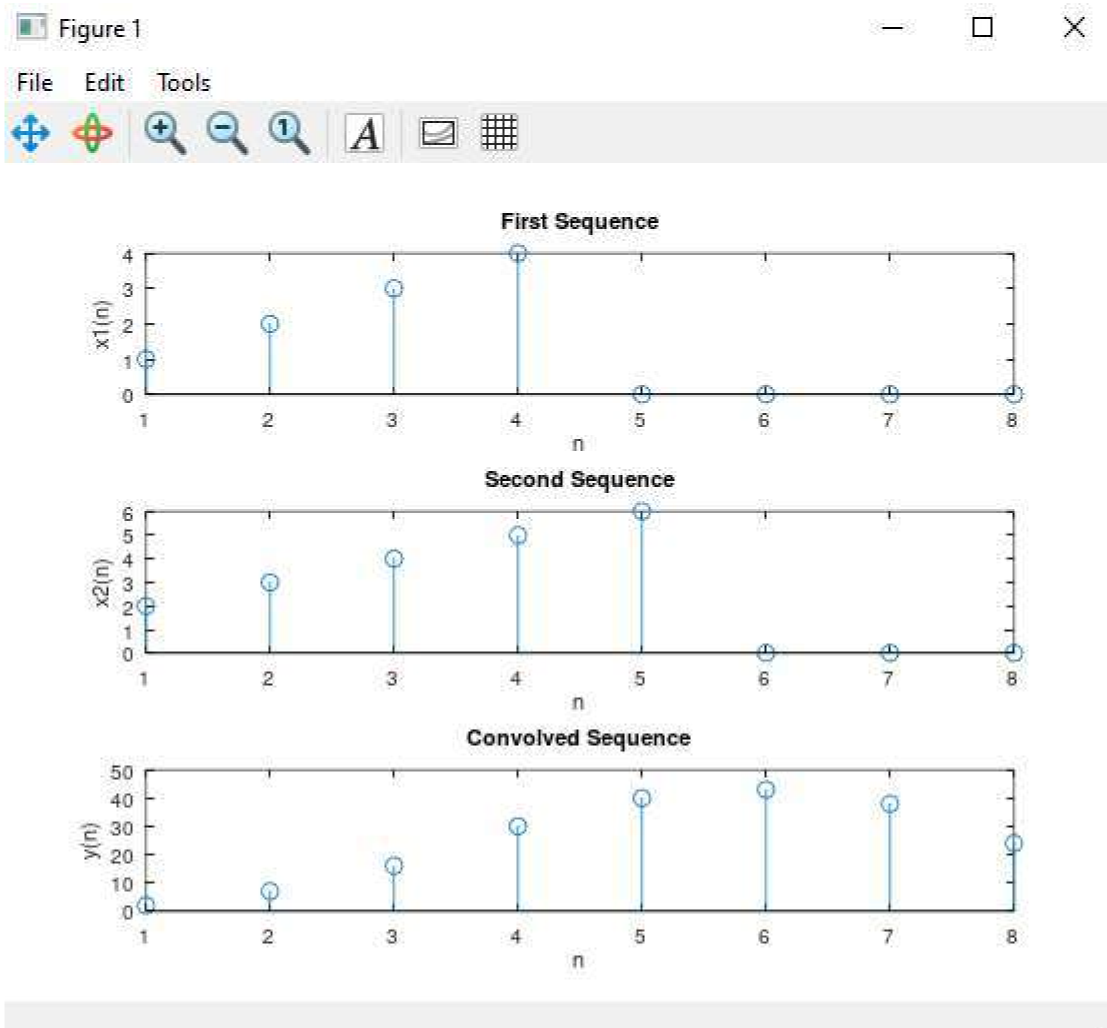
The convolution sum expresses how each value of the output sequence y[n] is a weighted sum of the input sequence values, where the weights are determined by the shifted and flipped values of the impulse response sequence h[n].

Octave Code:

```
clc; clear all; close all;
x1=input('Enter Input Sequence for Signal x1(n): ');
n1=length(x1);
x2=input('Enter Input Sequence for Signal x2(n): ');
n2=length(x2);
N=n1+n2-1; %Length of Convolved Sequence
%Zero padding to make sequences of length N
x1=[x1 zeros(1,N-n1)];
x2=[x2 zeros(1,N-n2)];
%Initializing Output sequence of zeros.
y = zeros(1,N);
for n = 1:N
for k = 1:n
y(n)=y(n)+x1(k)*x2(n-k+1);
end
end
subplot(3,1,1);
stem(x1);
xlabel('n');
ylabel('x1(n)');
title('First Sequence')
subplot(3,1,2);
stem(x2);
xlabel('n');
ylabel('x2(n)');
title('Second Sequence')
subplot(3,1,3);
stem(y);
xlabel('n');
ylabel('y(n)');
title('Convolved Sequence')
disp('Convolved sequence:');
disp(y);
```

Output:

Enter Input Sequence for Signal x1(n): [1 2 3 4]
Enter Input Sequence for Signal x2(n): [2 3 4 5 6]
Convolved sequence:
 2 7 16 30 40 43 38 24



Experiment No. 4

Consider a causal system y(n) = 0.9y(n-1)+x(n).

a) Determine H(z) and sketch its pole zero plot.
b) Plot |H(ejω)| and ∠ H(ejω)
c) Determine the impulse response h(n).

Aim: Consider a causal system y(n) = 0.9y(n-1)+x(n).
a) Determine H(z) and sketch its pole zero plot.
b) Plot |H(ejω)| and ∠ H(ejω)
c) Determine the impulse response h(n).

Theory:

1. Pole-Zero Plots

A pole-zero plots is a graphical representation of the poles and zeros of a system's transfer function in the complex plane. It helps to analyze the system's frequency response, stability, and behavior.

- Zeros: Points in the complex plane where the transfer function H(z) or H(s) is zero. These are the values of z or s where the numerator of the system's transfer function equals zero.
- Poles: Points where the transfer function becomes infinite, i.e., the denominator of H(z) or H(s) equals zero.

In a discrete-time system, the transfer function is often represented as:

$$H(z) = \frac{B(z)}{A(z)}$$

where B(z) and A(z) are polynomials in z, and the zeros are the roots of B(z), while the poles are the roots of A(z).

The system's stability can be inferred from the pole locations:

- If the poles are inside the unit circle (for discrete systems), the system is stable.
- If poles lie outside the unit circle, the system is unstable.

2. Magnitude and Phase Response |H(e^jω)| and ∠H(e^jω)

The **frequency response** of a system describes how the system responds to sinusoidal inputs of different frequencies. For a discrete-time system, the frequency response is given by evaluating the transfer function H(z) on the unit circle in the z-plane.

$$H(e^{j\omega}) = \frac{B(e^{j\omega})}{A(e^{j\omega})}$$

Magnitude Response |H(e^jω)|: It represents the amplitude scaling that the system applies to an input sinusoid at frequency ω. It can be computed as:

$$|H(e^{j\omega})| = \left|\frac{B(e^{j\omega})}{A(e^{j\omega})}\right|$$

Phase Response ∠H(e^jω): It represents the phase shift the system introduces to the input sinusoidal signal at frequency ω. It is given by:

$$\angle H(e^{j\omega}) = \arg\left(\frac{B(e^{j\omega})}{A(e^{j\omega})}\right)$$

3. Impulse Response h(n)

The **impulse response** h(n) of a system is the output when the input is a discrete unit impulse function δ(n). The impulse response fully characterizes a linear time-invariant (LTI) system. It is related to the transfer function H(z) via the inverse Z-transform:

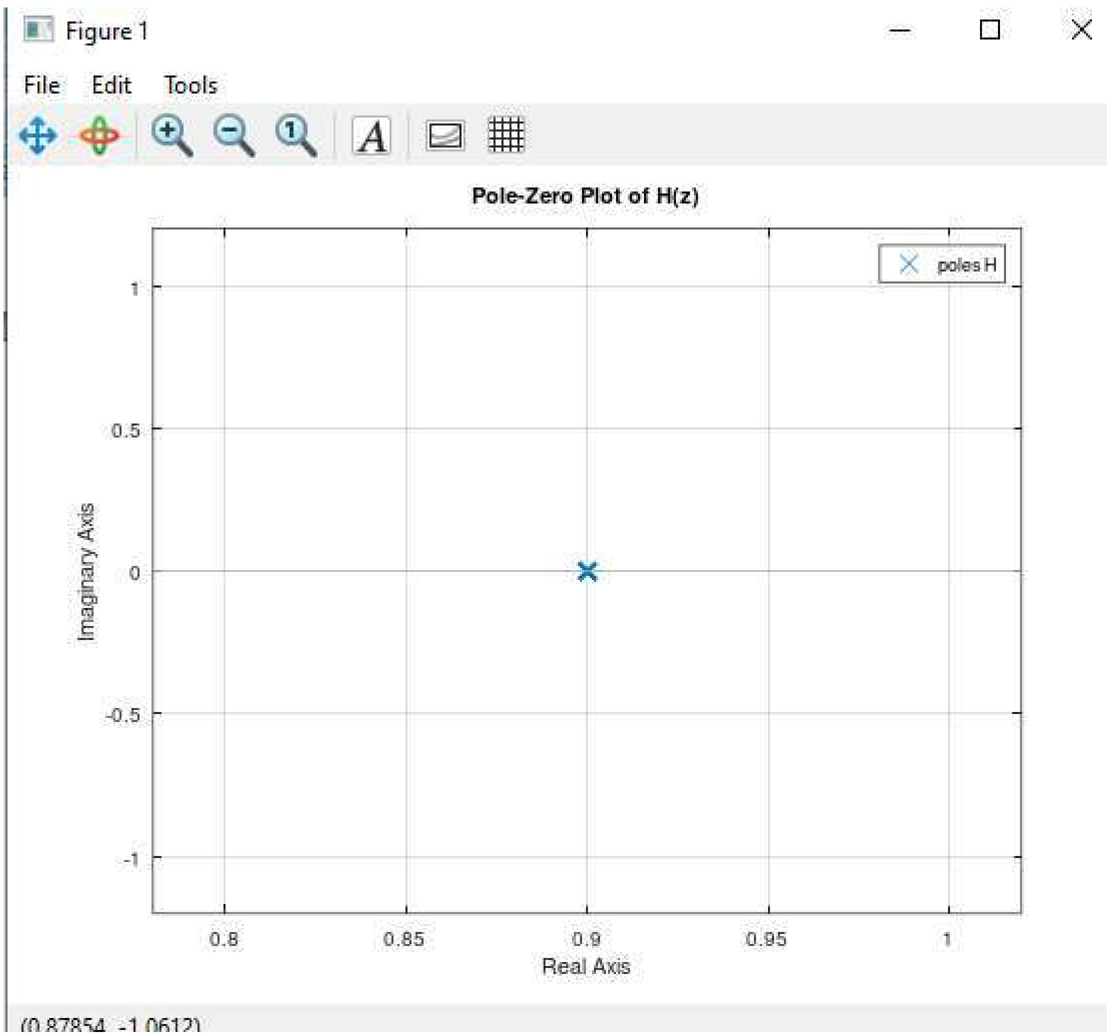
$$h(n) = \mathcal{Z}^{-1}[H(z)]$$

Octave Code:

```
clc, close all, pkg load control;
% Part A: Transfer Function and Pole-Zero Plot
numerator = [1]; % z
denominator = [1, -0.9]; % z - 0.9
% Discrete-time transfer function (zpk form)
H = zpk([], [0.9], 1); % Zero at origin, pole at 0.9
% Pole-Zero plot
figure;
pzmap(H);
title('Pole-Zero Plot of H(z)');
grid on;
% Part B: Frequency Response
omega = linspace(-pi, pi, 512); % Frequency from -pi to pi
H_freq = freqz(numerator, denominator, omega);
magnitude = abs(H_freq);
phase = angle(H_freq);
% Plot Magnitude
figure;
subplot(2, 1, 1);
plot(omega, magnitude);
title('Magnitude |H(e^{j\omega})|');
xlabel('\omega (radians/sample)');
ylabel('|H(e^{j\omega})|');
grid on;
% Plot Phase
subplot(2, 1, 2);
plot(omega, phase);
title('Phase \angle H(e^{j\omega})');
xlabel('\omega (radians/sample)');
ylabel('\angle H(e^{j\omega}) (radians)');
grid on;
% Part C: Impulse Response
n = 0:20; % Sample range
h = (0.9).^n; % Impulse response for n >= 0
% Plot Impulse Response
figure;
```

```
stem(n, h, 'filled');
title('Impulse Response h(n)');
xlabel('n');
ylabel('h(n)');
grid on;
```

Output:



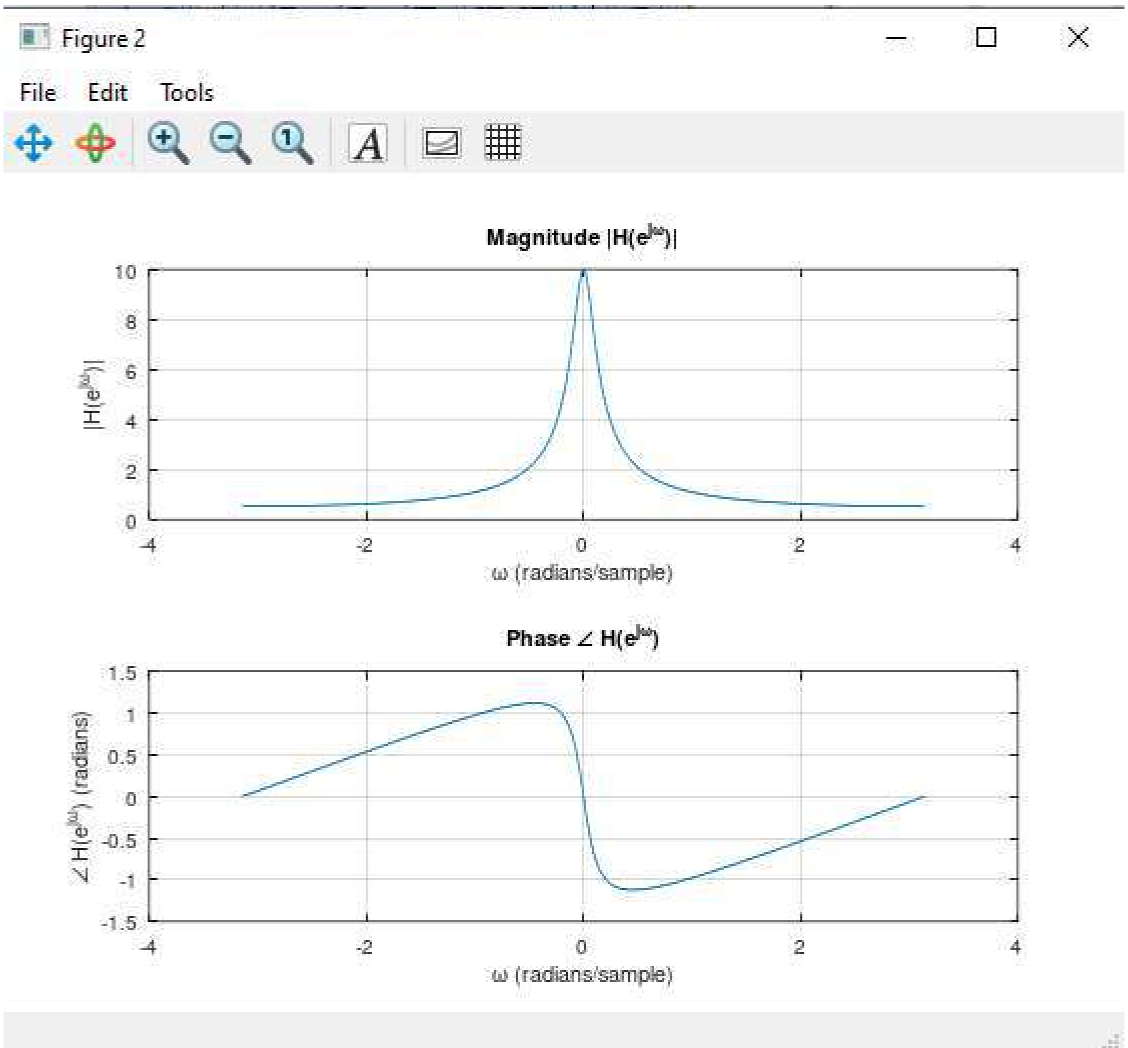


Figure 2
File
Edit
Tools
Magnitude |H(e^jω)|
|H(e^jω)|
ω (radians/sample)
Phase ∠ H(e^jω)
∠ H(e^jω) (radians)
ω (radians/sample)

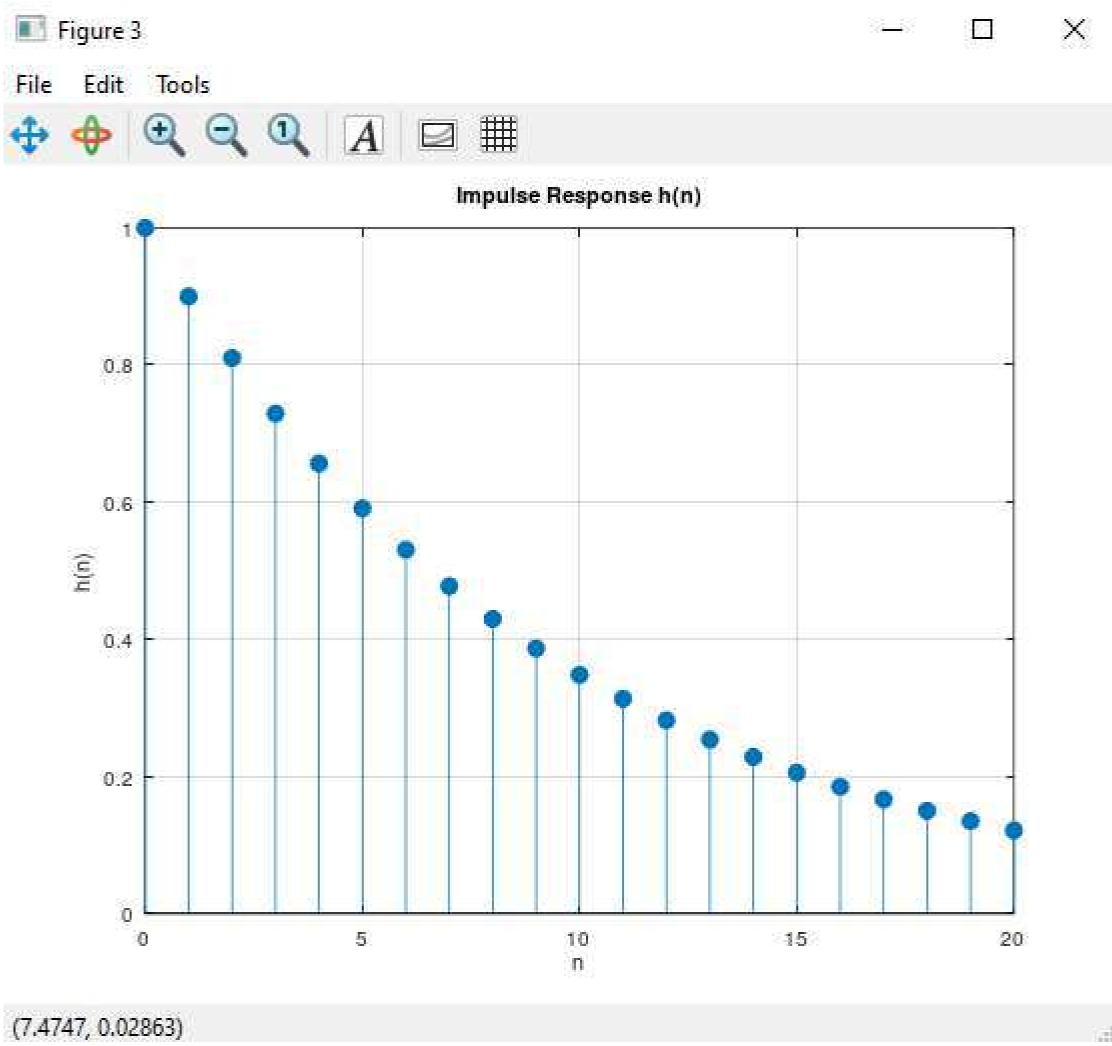


Figure 3
File Edit Tools
Impulse Response h(n)
h(n)
n
0
5
10
15
20
0
0.2
0.4
0.6
0.8
1
(7.4747, 0.02863)

Experiment No. 5

Computation of N point DFT of a given sequence (without using built-in function) and to plot the magnitude and phase spectrum

Aim: To Compute N Point DFT of a given sequence and plot magnitude and phase spectrum.

Theory:

The Discrete Fourier Transform (DFT) is a mathematical transformation used in signal processing to analyze the frequency content of discrete-time signals. It converts a finite sequence of equally spaced samples in the time domain into a sequence of complex numbers representing the signal's frequency components.

For a given sequence x(n), where n ranges from 0 to N−1, the N-point DFT is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, 2, \ldots, N-1$$

Where:

- X(k) is the k-th frequency bin (or frequency component) in the DFT output.
- x(n) is the time-domain sequence (input signal) of length N.
- N is the total number of samples.
- k represents the index of the DFT output, and it corresponds to discrete frequency bins.
- e^(−j2π/Nkn) is a complex exponential (the basis function for the DFT), representing the frequency components of the signal.

Octave Code:

```
clc, close all, clear all;
% Define the input sequence
x = [1, 2, 3, 4]; % Example sequence
N = length(x);    % Number of points in DF
% Initialize the DFT output
X = zeros(1, N);  % DFT output array
% Compute the N-point DFT
for k = 0:N-1
   for n = 0:N-1
      X(k+1) = X(k+1) + x(n+1) * exp(-1j * 2 * pi * k * n / N);
   end
end
% Magnitude and Phase Spectrum
magnitude = abs(X);
phase = angle(X);
% Frequency bins for plotting
frequencies = (0:N-1) * (2 * pi / N); % Frequency range
% Plotting the Magnitude Spectrum
figure;
subplot(2, 1, 1);
stem(frequencies, magnitude, 'filled');
title('Magnitude Spectrum');
xlabel('Frequency (radians/sample)');
ylabel('|X(k)|');
grid on;
%Plotting the Phase Spectrum
subplot(2, 1, 2);
stem(frequencies, phase, 'filled');
title('Phase Spectrum');
xlabel('Frequency (radians/sample)');
ylabel('angle X(k) (radians)');
grid on;
```

Output:

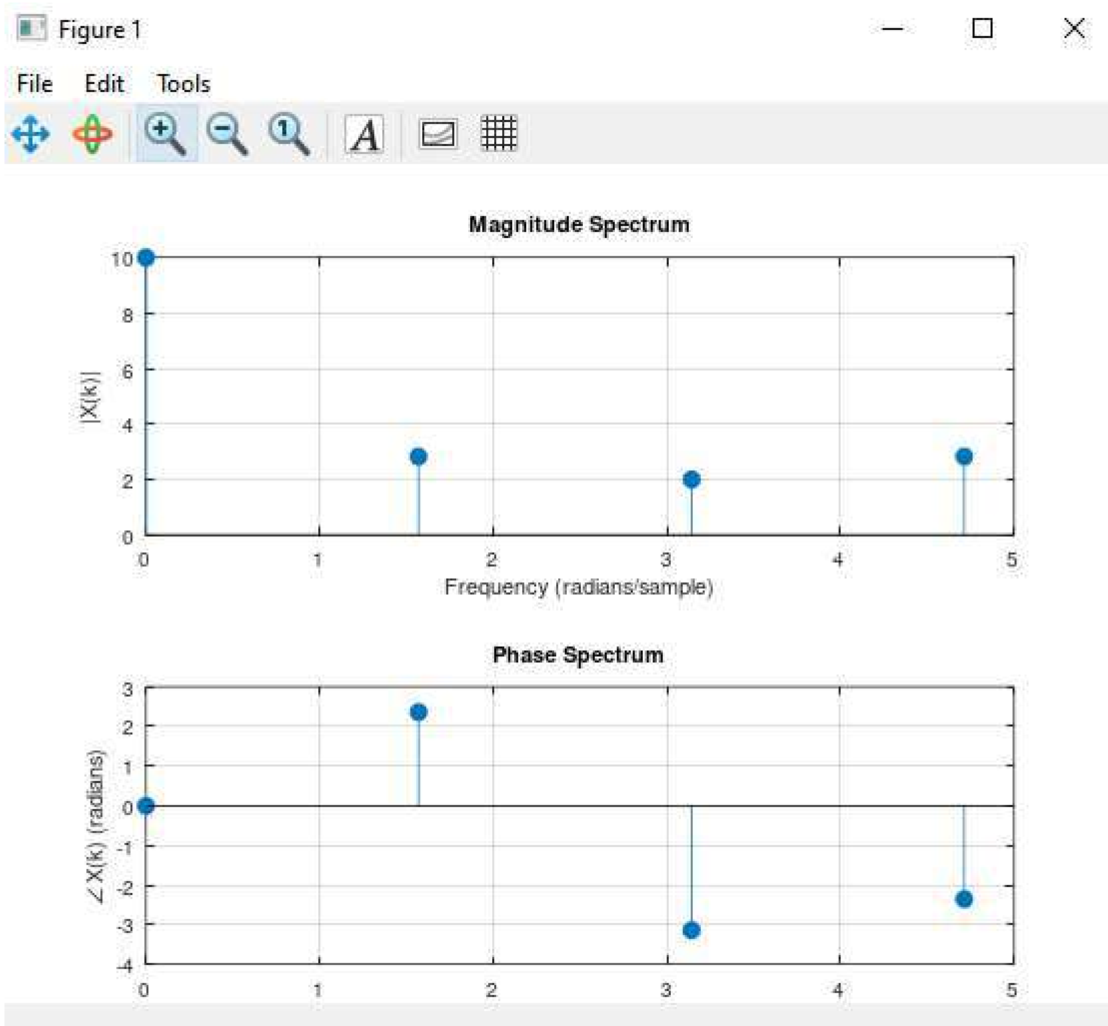


Figure 1
File Edit Tools
Magnitude Spectrum
|X(k)|
Frequency (radians/sample)
Phase Spectrum
∠X(k) (radians)

Experiment No. 6

Using the DFT and IDFT, compute the following for any two given sequences

a) Circular convolution
b) Linear convolution

Aim: To compute Circular and Linear Convolution using DFT and IDFT.

Theory:

1. Linear Convolution

Linear convolution refers to the standard convolution operation applied between two signals in the time domain. It is commonly used in signal processing and linear systems analysis. The **linear convolution** of two discrete-time signals x(n) and h(n) is defined as:

$$y(n) = (x * h)(n) = \sum_{m=-\infty}^{\infty} x(m)h(n-m)$$

Where:

- y(n) is the output signal.
- x(n) is the input signal.
- h(n) is the impulse response (or filter) of the system.
- n is the discrete-time index.

2. Circular Convolution

Circular convolution is a variation of convolution where the sequences are considered to be periodic, i.e., wrapped around in a circular fashion. In circular convolution, the signal is effectively "wrapped" at the ends, and no signal outside the period is considered.
The **circular convolution** of two discrete-time sequences x(n) and h(n), each of length N, is given by:

$$y(n) = (x \circledast h)(n) = \sum_{m=0}^{N-1} x(m)h((n-m) \mod N)$$

Where:

- y(n) is the output sequence of the same length N.
- x(n) and h(n) are the two periodic sequences.
- The modulo operation (n−m) mod N ensures that the indices "wrap around" for periodic behavior.

Octave Code:

```
clc, close all, clear all;
%Define two example sequences
x = [1,2,3];  % First sequence
h = [0.5,1];   % Second sequence
% Lengths of the sequences
N = max(length(x), length(h)); % Length for DFT (circular convolution)
L = length(x) + length(h) - 1; % Length for linear convolution
% Circular Convolution using DFT and IDFT
X = fft(x, N); % DFT of x
H = fft(h, N); % DFT of h
% Circular convolution result
Y_circular = X.* H; % Element-wise multiplication in frequency domain
y_circular = ifft(Y_circular); % IDFT to get the circular convolution result
% Linear Convolution using DFT and IDFT
x_padded = [x, zeros(1, L - length(x))]; % Zero-pad x
h_padded = [h, zeros(1, L - length(h))]; % Zero-pad h
X_linear = fft(x_padded, L); % DFT of x padded
H_linear = fft(h_padded, L); % DFT of h padded
% Linear convolution result
Y_linear = X_linear .* H_linear; % Element-wise multiplication in frequency domain
y_linear = ifft(Y_linear); % IDFT to get the linear convolution result
disp('Circular Convolution Result:');
disp(y_circular);
disp('Linear Convolution Result:');
disp(y_linear)
subplot(2, 1, 1);
stem(0:N-1, real(y_circular), 'filled');
title('Circular Convolution Result');
xlabel('n');
ylabel('y[n]');
grid on
subplot(2, 1, 2);
stem(0:L-1, real(y_linear), 'filled');
title('Linear Convolution Result');
xlabel('n');
ylabel('y[n]');
grid on;
```

Result:

```
Circular Convolution Result:
   3.5000   2.0000   3.5000
Linear Convolution Result:
   0.5000   2.0000   3.5000   3.0000
```

Output:

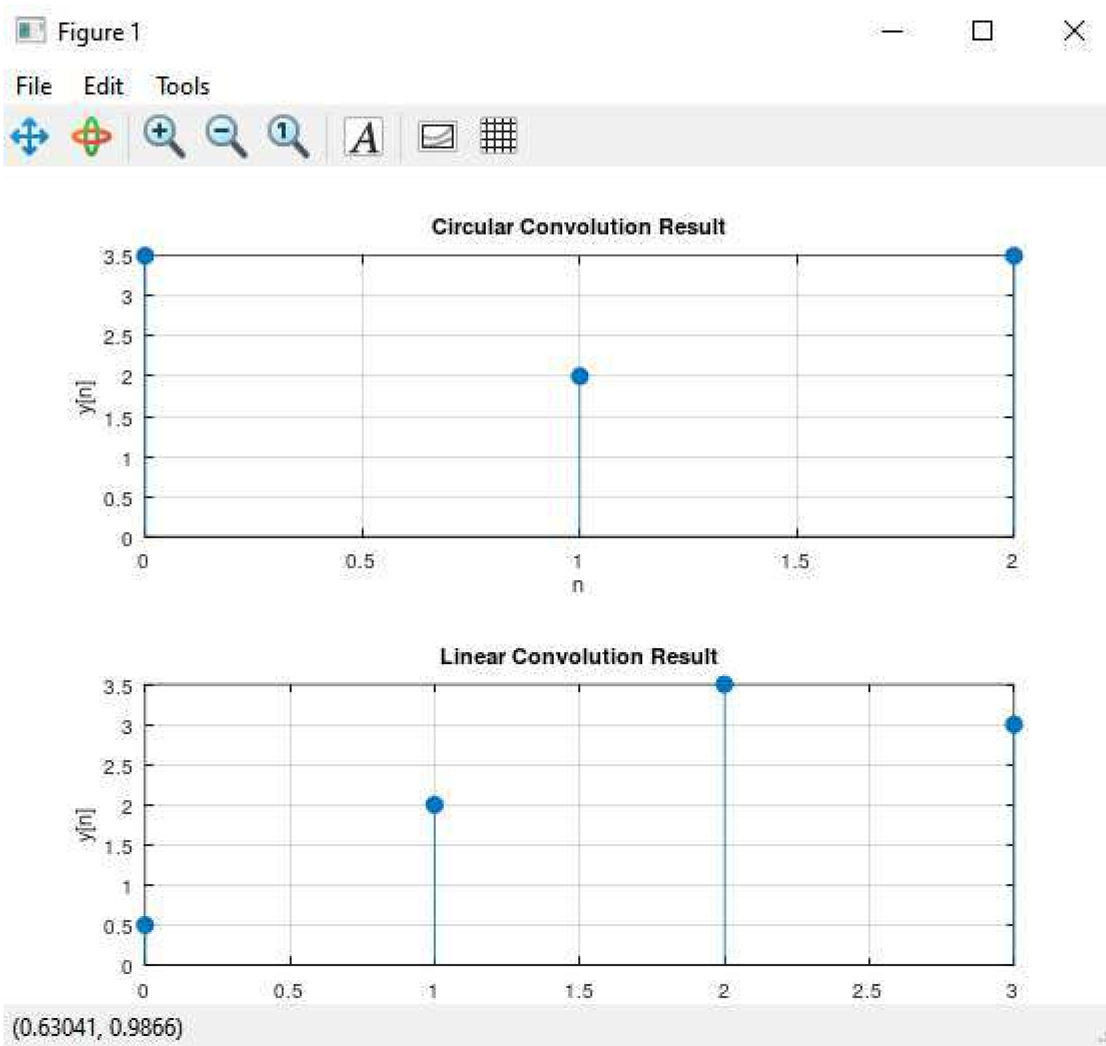


Figure 1
File
Edit
Tools
Circular Convolution Result
y[n]
n
0
0.5
1
1.5
2
2.5
3
3.5
Linear Convolution Result
(0.63041, 0.9866)

Experiment No. 7

Verification of Linearity property, circular time shift property & circular frequency shift property of DFT.

Aim: To Verify Linearity property, circular time shift property & circular frequency shift property of DFT.

Theory:

1. Linearity Property of DFT

The DFT is a linear operator, meaning that it satisfies the principle of superposition. This property states that the DFT of a weighted sum of sequences is the weighted sum of their individual DFTs.
Mathematically:
Let x1(n) and x2(n) be two discrete-time sequences, and let a1 and a2 be arbitrary constants. The DFT of the linear combination of these sequences is:

$$\mathcal{X}(k) = \mathcal{X}_1(k) + \mathcal{X}_2(k)$$

This property implies that if you have a combination of signals, you can compute their DFTs separately and then combine them using the same weights. It is an essential property for simplifying calculations in signal processing.

2. Circular Time Shift Property of DFT

The **Circular Time Shift** property refers to the effect of shifting a sequence in time on its DFT. A time shift of a discrete sequence corresponds to a linear phase shift in its DFT.

Mathematically:

Let x(n) be a sequence with its DFT X(k). If the sequence is shifted by n0 samples, the new sequence becomes x(n−n0). The DFT of the shifted sequence is given by:

$$\mathcal{X}_{n_0}(k) = X(k)e^{-j\frac{2\pi}{N}kn_0}$$

This property shows that a shift in the time domain results in a multiplication of the DFT by a complex exponential term with a frequency-dependent phase shift. The magnitude of the DFT remains unchanged, but the phase is modified.

3. Circular Frequency Shift Property of DFT

The **Circular Frequency Shift** property describes the effect of shifting a sequence in the frequency domain on its DFT. A frequency shift in the frequency domain corresponds to a circular shift in the time domain.

Mathematically:

Let x(n) be a sequence with its DFT X(k). If the sequence is multiplied by a complex exponential in the time domain (i.e., a frequency shift), the DFT of the new sequence is:

$$\mathcal{Y}(k) = X(k - k_0)$$

This property shows that multiplying the signal in the time domain by a complex exponential corresponds to shifting the DFT in the frequency domain. The shift is circular, meaning that the frequency components "wrap around" when the index exceeds N−1N-1N−1, the maximum index in the DFT.

Octave Code:

```
clc, clear all, close all;
% Define parameters
N = 8; % Length of DFT
n = 0:N-1; % Sample indices
% Define two sequences for testing
x1 = [1, 2, 3, 4, 0, 0, 0, 0]; % First sequence
x2 = [0, 1, 0, 1, 0, 0, 0, 0]; % Second sequence
% Define coefficients for linearity
alpha = 2; % Coefficient for x1
beta = 3;  % Coefficient for x2
% 1. Linearity Property
% DFT of individual sequences
X1 = fft(x1);
```

```
X2 = fft(x2);
% DFT of the linear combination
X_linear_combination = fft(alpha * x1 + beta * x2);
% Verify linearity
X_combined = alpha * X1 + beta * X2;
% Display results
disp('Linearity Verification:');
disp('DFT of linear combination:');
disp(X_linear_combination);
disp('Linear combination of DFTs:');
disp(X_combined);
% 2. Circular Time Shift Property
% Time shift by 2 samples
shift = 2;
x_shifted = circshift(x1, shift); % Circularly shift x1 by 2
X_shifted = fft(x_shifted); % DFT of the shifted sequence
X_original = fft(x1); % DFT of the original sequence
% Verify circular time shift property
% Expected: X_shifted = X_original * e^(-j*2*pi*shift/N*n)
% Display results
disp('Circular Time Shift Verification:');
disp('DFT of shifted sequence:');
disp(X_shifted);
disp('Expected result:');
expected_shifted = X_original .* exp(-1j * 2 * pi * shift / N * n);
disp(expected_shifted);
% 3. Circular Frequency Shift Property
% Frequency shift by 1 sample
freq_shift = 1;
X_freq_shifted = circshift(X_original, freq_shift); % Circularly shift DFT
% Verify circular frequency shift property
% Expected: x[n] * e^(j*2*pi*freq_shift*n/N)
% Display results
disp('Circular Frequency Shift Verification:');
disp('DFT of frequency shifted sequence:');
disp(X_freq_shifted);
disp('Expected result:');
```

```
expected_freq_shifted = X_original .* exp(1j * 2 * pi * freq_shift * n / N);
disp(expected_freq_shifted);
% Plotting results for verification
figure;
% Linearity Property
subplot(3, 1, 1);
stem(n, real(X_linear_combination), 'filled', 'DisplayName', 'Linear Combination');
hold on;
stem(n, real(X_combined), 'filled', 'DisplayName', 'Combined DFT');
title('Linearity Property Verification');
xlabel('k');
ylabel('Magnitude');
legend;
grid on;
% Circular Time Shift Property
subplot(3, 1, 2);
stem(n, real(X_shifted), 'filled', 'DisplayName', 'Shifted DFT');
hold on;
stem(n, real(expected_shifted), 'filled', 'DisplayName', 'Expected DFT');
title('Circular Time Shift Property Verification');
xlabel('k');
ylabel('Magnitude');
legend;
grid on;
% Circular Frequency Shift Property
subplot(3, 1, 3);
stem(n, real(X_freq_shifted), 'filled', 'DisplayName', 'Frequency Shifted DFT');
hold on;
stem(n, real(expected_freq_shifted), 'filled', 'DisplayName', 'Expected DFT');
title('Circular Frequency Shift Property Verification');
xlabel('k');
ylabel('Magnitude');
legend;
grid on;
```

Output:

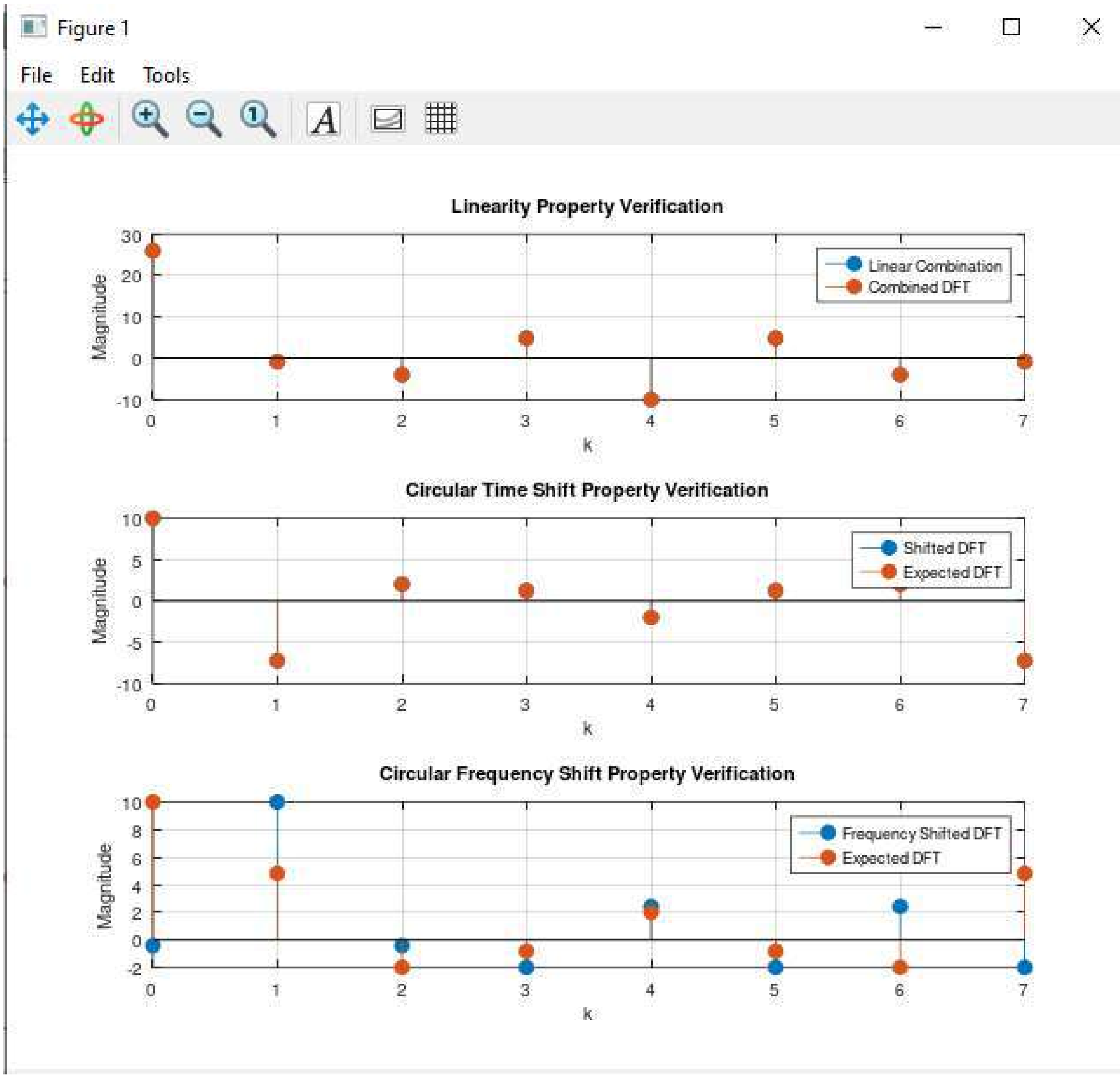


Figure 1
File Edit Tools
Linearity Property Verification
Linear Combination
Combined DFT
Magnitude
k
Circular Time Shift Property Verification
Shifted DFT
Expected DFT
Magnitude
k
Circular Frequency Shift Property Verification
Frequency Shifted DFT
Expected DFT
Magnitude
k

Experiment No. 8

Develop decimation in time radix-2 FFT algorithm without using built-in functions.

Aim: To Develop decimation in time radix-2 FFT algorithm.

Theory:

The **Decimation in Time (DIT)** algorithm is one of the most widely used methods for efficiently computing the Fast Fourier Transform (FFT) of a sequence. It is based on the principle of breaking down a larger problem into smaller sub problems, and it is particularly well-suited for sequences whose length is a power of two. The Radix-2 FFT algorithm, which is a specific case of the DIT approach, operates on an input sequence whose length N is a power of 2 (i.e., N=2^m)

1. Structure of the DIT Radix-2 FFT

The key to the DIT Radix-2 FFT is the "decimation" step, where the input sequence is recursively divided into two smaller sequences. This is done by splitting the sequence into even and odd indexed elements:

Step 1: Divide and Conquer

Given a sequence x[n] of length N = 2^m, the input sequence is split into two sub-sequences:

- One containing the even-indexed elements: x[0],x[2],x[4]
- The other containing the odd-indexed elements: x[1],x[3],x[5]

This division process reduces the problem size by a factor of 2, and we continue this division recursively until we have sequences of length 1.

Step 2: FFT of Sub-sequences

Once the sequence has been divided into smaller sub-sequences of length 1, the DFT of each length-1 sequence is trivially computed. For length-1 sequences, the DFT is simply the value of the sequence itself, as X[0] = x[0] and so on.

Step 3: Combine Results Using the Butterfly Operation

After computing the DFTs of the smallest sub-sequences, the next step is to combine the results using the **butterfly operation**, which forms the basis of the FFT algorithm. The butterfly operation combines the results of two smaller DFTs to compute the corresponding DFTs of the larger sequence.

For a sequence of length N, the butterfly operation for the combined result of the k-th element of the FFT can be written as:

$$X[k] = X_{\text{even}}[k] + W_N^k \cdot X_{\text{odd}}[k]$$

$$X[k + N/2] = X_{\text{even}}[k] - W_N^k \cdot X_{\text{odd}}[k]$$

.

Octave Code:

```
Clc, clear all, close all;
function X = radix2_fft(x)
  % Check if the length of x is a power of 2
  N = length(x);
  if mod(log2(N), 1) != 0
    error('Length of input must be a power of 2');
  end
  % Recursive FFT computation
  if N == 1
    X = x; % Base case
    return;
  end
  % Split into even and odd parts
  X_even = radix2_fft(x(1:2:N)); % FFT of even-indexed elements
  X_odd = radix2_fft(x(2:2:N));  % FFT of odd-indexed elements
  % Combine results
  X = zeros(1, N); % Initialize the result array
  for k = 0:N/2-1
    twiddle_factor = exp(-1j * 2 * pi * k / N) * X_odd(k + 1); % Twiddle factor
    X(k + 1) = X_even(k + 1) + twiddle_factor; % Combine even and odd
    X(k + N/2) = X_even(k + 1) - twiddle_factor; % Combine even and odd
  end
end
```

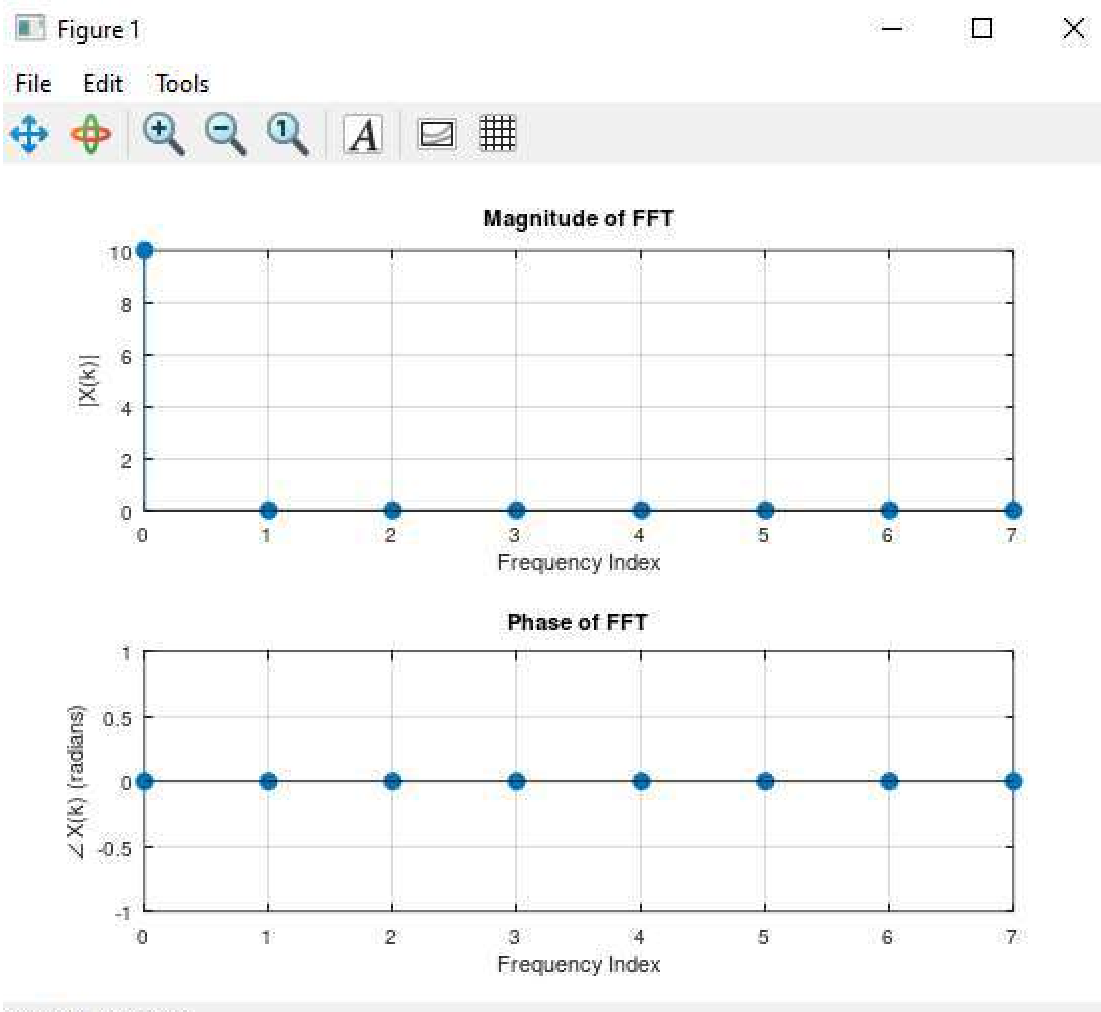
```
% Example Usage
N = 8; % Length of the input signal (must be a power of 2)
x = [4, 3, 2, 1, 0, 0, 0, 0]; % Example input sequence
% Compute the FFT
X = radix2_fft(x);
% Display results
disp('FFT Result:');
disp(X);


figure;
subplot(2, 1, 1);
stem(0:N-1, abs(X), 'filled');
title('Magnitude of FFT');
xlabel('Frequency Index');
ylabel('|X(k)|');
grid on;

subplot(2, 1, 2);
stem(0:N-1, angle(X), 'filled');
title('Phase of FFT');
xlabel('Frequency Index');
ylabel('angle X radians)');
grid on;
```

Result:

FFT Result:

10 0 0 0 0 0 0 0

Output:



Experiment No. 9

Design and implementation of digital low pass FIR filter using a window to meet the given specifications

Aim: To design and implement digital low pass FIR Filter using window.

Theory:

Linear-phase is required throughout the passband of the filter to preserve the shape of the given signal in the passband. A causal IIR filter cannot give linear-phase characteristics and only special types of FIR filters that exhibit center symmetry in its impulse response give the linear-phase. An FIR filter with impulse response h[n] can be obtained as follows:

$h[n] = h_d[n],\ 0 \le n \le N-1$

$= 0$, otherwise…………..(a)

The impulse response $h_d[n]$ is truncated, since we are interested in causal FIR filter. It is possible to write above equation alternatively as

$h[n] = h_d[n]w[n]$……………..(b)

Where w[n] is said to be a rectangular window defined by,

$w[n] = 1,\ 0 \le n \le N-1$

$= 0$, otherwise

Taking DTFT on both the sides of equation(b), we get

$H(\omega) = H_d(\omega) * W(\omega)$

Hamming Window:

The impulse response of an N-term Hamming Window is defined as follows:

$$w_{Ham}(n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$$

Octave Code:

```
clc; clear all; close all;
%accept filter parameters from user
N=input('Enter the window length N:');
fc=input('Enter the cut-off frequency fc(Hz):');
Fs=input('Enter the sampling frequency Fs(Hz):');
Wc=2*fc/Fs;
Wh=hamming(N);
%generate a FIR filter based on hamming window
b=fir1(N-1,Wc,'low',Wh);
[h,W]=freqz(b,1,256);
mag=20*log10(abs(h));
%display values
disp('Hamming Window Coefficients:');
disp(Wh);
disp('Unit Sample Response of FIR filter h[n]:');
disp(b);
%plot frequency response
freqz(b);
title('Hamming Filter Frequency Response');
```

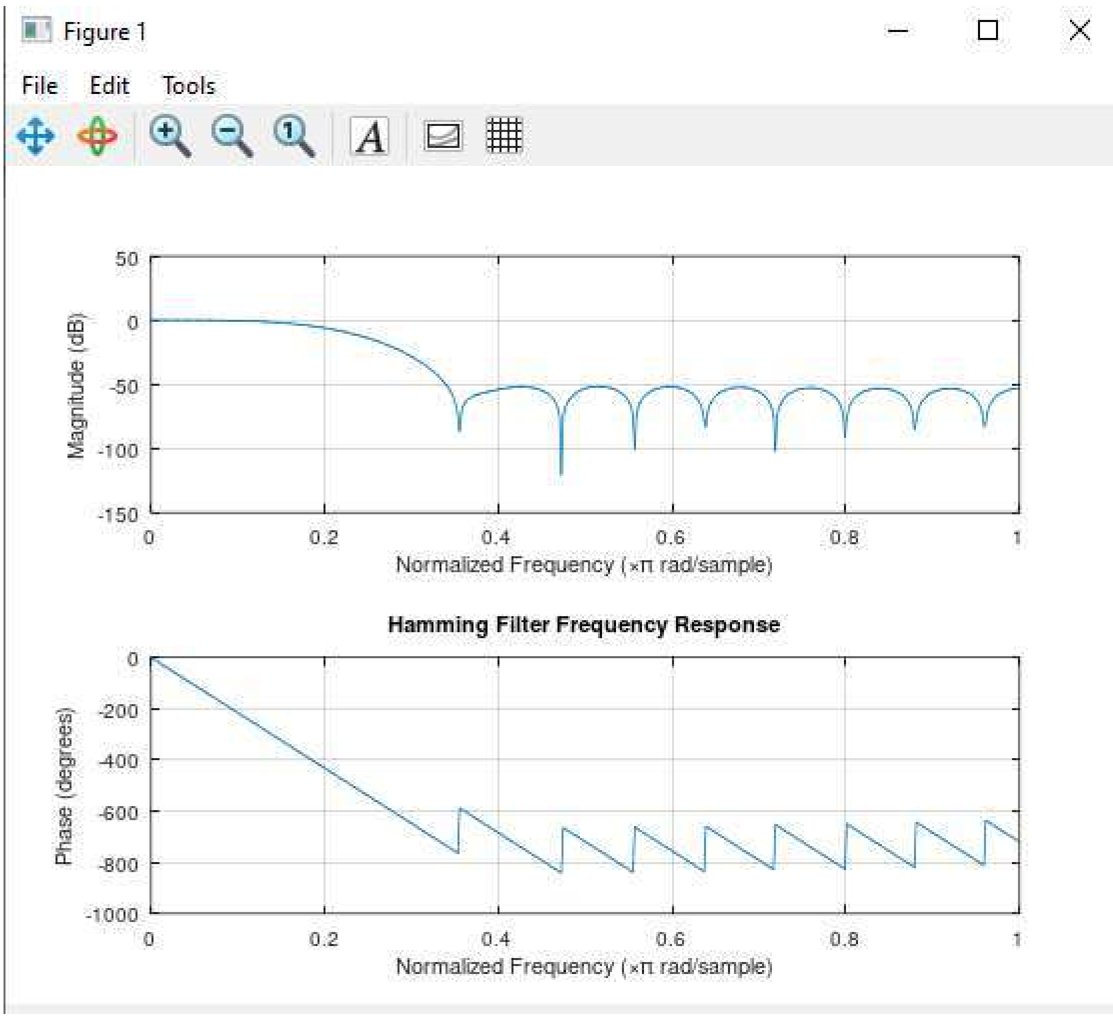
Output:

```
Enter the window length N: 25
Enter the cut-off frequency fc(Hz):200
Enter the sampling frequency Fs(Hz):2000
Hamming Window Coefficients:
   0.080000
   0.095674
   0.141628
   0.214731
   0.310000
   0.420943
   0.540000
   0.659057
   0.770000
   0.865269
   0.938372
   0.984326
   1.000000
   0.984326
   0.938372
   0.865269
   0.770000
   0.659057
   0.540000
   0.420943
   0.310000
   0.214731
   0.141628
   0.095674
   0.080000
Unit Sample Response of FIR filter h[n]:
 Columns 1 through 11:

   1.9841e-03   1.5442e-03  -1.3772e-04  -4.6127e-03  -1.1773e-02  -1.8000e-02  -1.6344e-02
6.4104e-04   3.6477e-02   8.7228e-02   1.4120e-01

 Columns 12 through 22:
```

1.8267e-01 1.9825e-01 1.8267e-01 1.4120e-01 8.7228e-02 3.6477e-02 6.4104e-04 -1.6344e-02 -1.8000e-02 -1.1773e-02 -4.6127e-03

Columns 23 through 25:

-1.3772e-04 1.5442e-03 1.9841e-03
>>



Experiment No.10

Design and implementation of digital high pass FIR filter using a window to meet the given specifications

Aim: To design and implement digital High Pass FIR Filter using window.

Theory:

Linear-phase is required throughout the passband of the filter to preserve the shape of the given signal in the passband. A causal IIR filter cannot give linear-phase characteristics and only special types of FIR filters that exhibit center symmetry in its impulse response give the linear-phase. An FIR filter with impulse response h[n] can be obtained as follows:

$h[n] = h_d[n],\ 0 \le n \le N-1$

$= 0$, otherwise…………..(a)

The impulse response $h_d[n]$ is truncated, since we are interested in causal FIR filter. It is possible to write above equation alternatively as

$h[n] = h_d[n]w[n]$……………..(b)

Where w[n] is said to be a rectangular window defined by,

$w[n] = 1,\ 0 \le n \le N-1$

$= 0$, otherwise

Taking DTFT on both the sides of equation(b), we get

$H(\omega) = H_d(\omega)*W(\omega)$

Hamming Window:

The impulse response of an N-term Hamming Window is defined as follows:

$$w_{Ham}(n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$$

Octave Code:

```
clc; clear all; close all;
%accept filter parameters from user
N=input('Enter the window length N:');
fc=input('Enter the cut-off frequency fc(Hz):');
Fs=input('Enter the sampling frequency Fs(Hz):');
Wc=2*fc/Fs;
Wh=hamming(N);
%generate a FIR filter based on hamming window
b=fir1(N-1,Wc,'high',Wh);
[h,W]=freqz(b,1,256);
mag=20*log10(abs(h));
%display values
disp('Hamming Window Coefficients:');
disp(Wh);
disp('Unit Sample Response of FIR filter h[n]:');
disp(b);
%plot frequency response
freqz(b);
title('Hamming Filter Frequency Response');
```

Output:

```
Enter the window length N:33
Enter the cut-off frequency fc(Hz):300
Enter the sampling frequency Fs(Hz):1000
Hamming Window Coefficients:
   0.080000
   0.088839
   0.115015
   0.157524
   0.214731
   0.284438
   0.363966
   0.450258
   0.540000
   0.629742
```

0.716034
0.795562
0.865269
0.922476
0.964985
0.991161
1.000000
0.991161
0.964985
0.922476
0.865269
0.795562
0.716034
0.629742
0.540000
0.450258
0.363966
0.284438
0.214731
0.157524
0.115015
0.088839
0.080000

Unit Sample Response of FIR filter h[n]:

Columns 1 through 11:

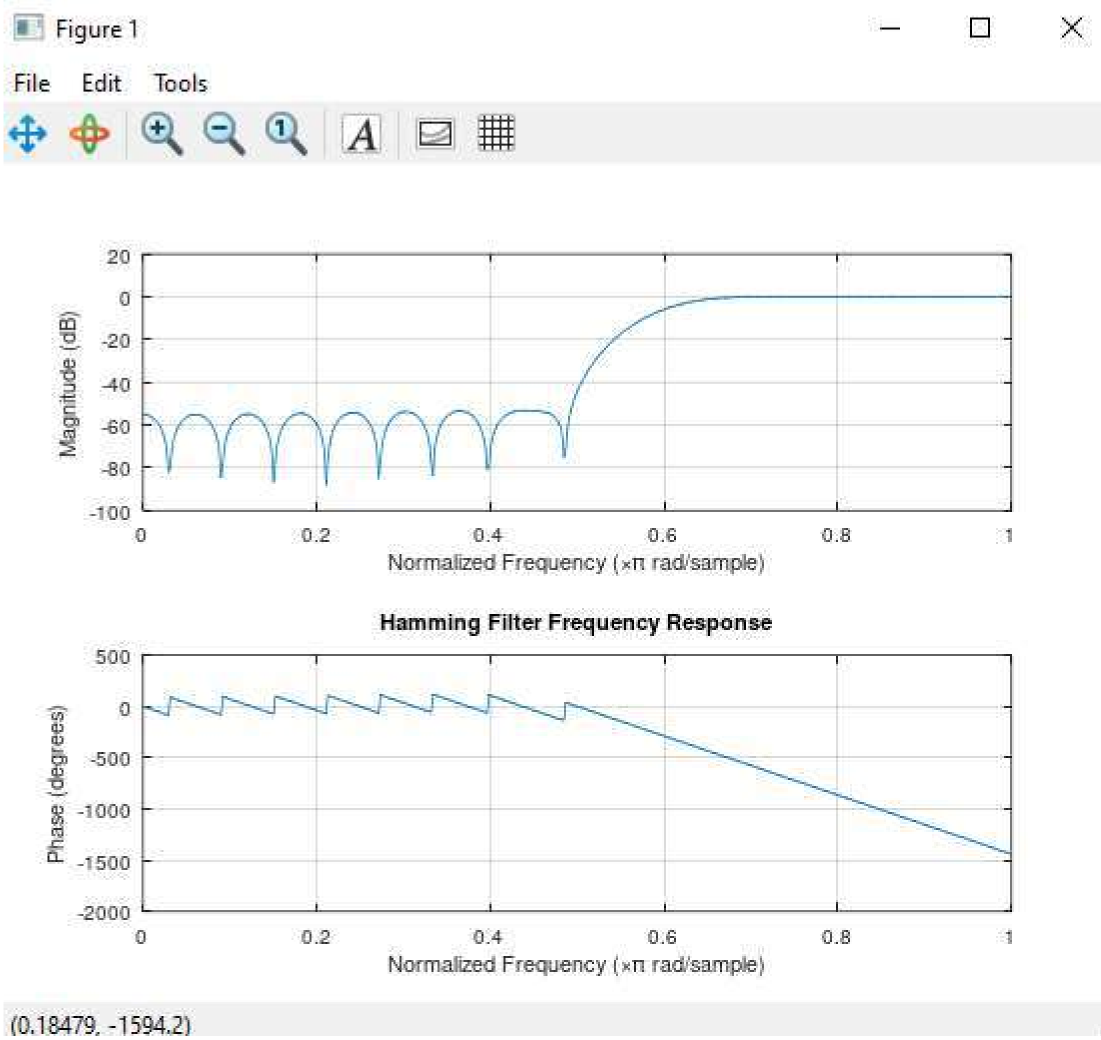
1.5339e-03 -8.6637e-05 -2.4470e-03 2.3868e-03 3.1723e-03 -7.9003e-03 3.5503e-04 1.4987e-02 -1.3038e-02 -1.6313e-02 3.6300e-02

Columns 12 through 22:

-7.7613e-04 -6.5155e-02 5.8200e-02 8.9422e-02 -3.0006e-01 4.0059e-01 -3.0006e-01 8.9422e-02 5.8200e-02 -6.5155e-02 -7.7613e-04

Columns 23 through 33:

```
   3.6300e-02  -1.6313e-02  -1.3038e-02   1.4987e-02   3.5503e-04  -7.9003e-03   3.1723e-03
2.3868e-03  -2.4470e-03  -8.6637e-05   1.5339e-03
>>
```



Experiment No.11

Design and implementation of digital IIR Butterworth low pass filter to meet the given specifications.

Aim: To design and implement of digital IIR Butterworth low pass filter to meet the given specifications.

Theory:

A desired frequency response is approximated by a transfer function expressed as a ratio of polynomials. This type of transfer function yields an impulse response of infinite duration. Therefore, the analog filters are commonly referred to as infinite impulse response (IIR) filters.

The main classes of analog filters are:

1. Butterworth Filter.
2. Chebyshev Filter.

These filters differ in the nature of their magnitude response as well as in their design and implementation.

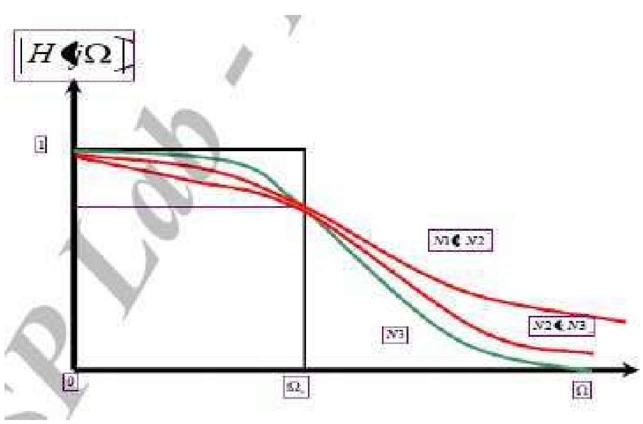
Butterworth Filters:

Butterworth filters have very smooth passband, which we pay for with a relatively wide transition region. A butterworth filter is characterized by its magnitude frequency response,

$$|H(j\Omega)| = 1 / [1 + (\Omega/\Omega_c)2N]^{1/2}$$

Where N is the order of the filter and Ω_c is defined as the cut-off frequency, where the filter magnitude is $1/\sqrt{2}$ times the dc gain ($\Omega = 0$).

$$\text{Order, N} = \frac{log_{10}[\frac{10^{0.1As\,db}-1}{10^{0.1Ap\,db}-1}]}{2\,log\frac{\Omega s}{\Omega p}}$$

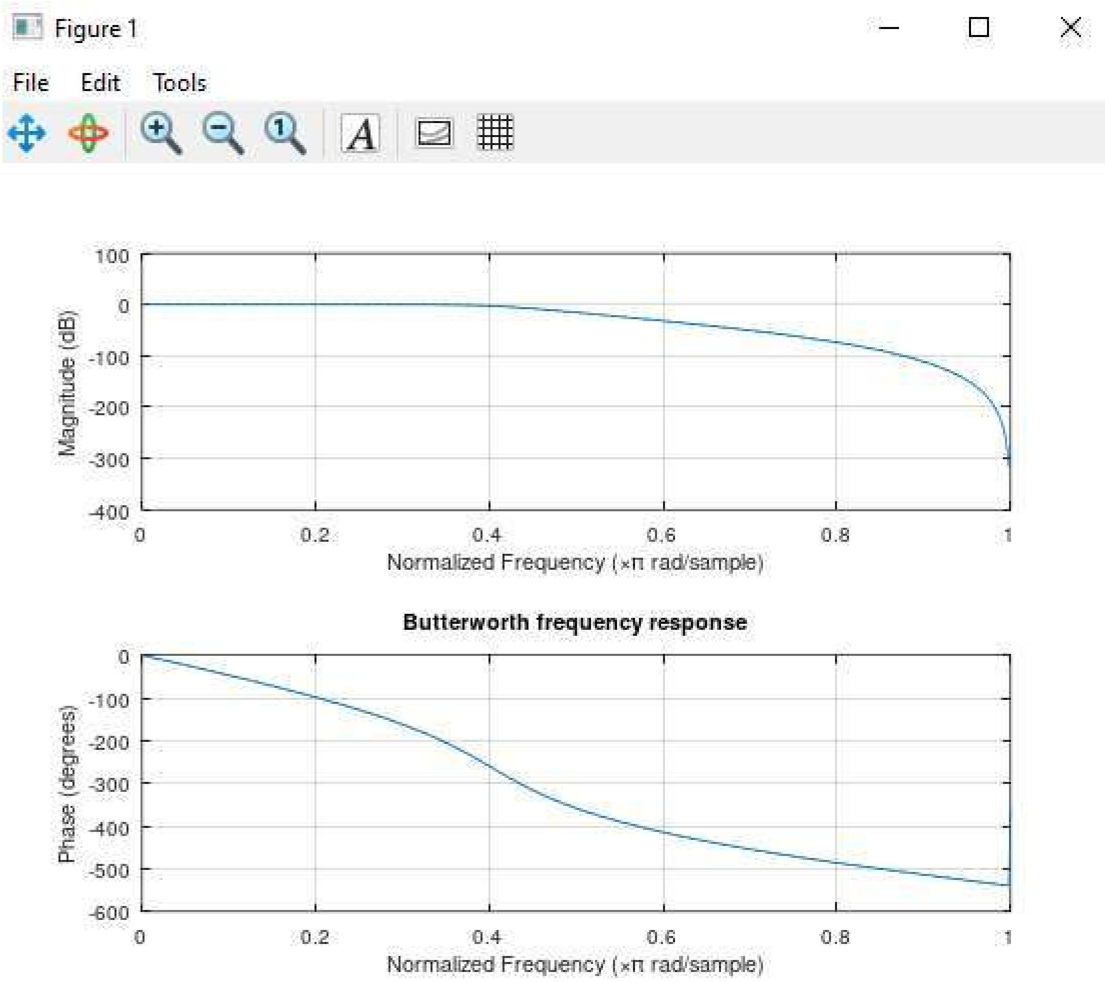


Octave Code:

```
clc; clear all; close all;
Rp=input('Enter passband attenuation in dB:');
Rs=input('Enter stopband attenuation in dB:');
fp=input('Enter passband frequency in Hz:');
fs=input('Enter stopband frequency in Hz:');
Fs=input('Enter sampling frequency in Hz:');
ftype=input('Enter type of filter:','s');
%Calculate sampled frequency values
wp=2*fp/Fs;
ws=2*fs/Fs;
[N,wn]=buttord(wp,ws,Rp,Rs);
%Butterworth filter design
[z,p]=butter(N,wn,ftype);
%display filter parameters
disp('Order of Butterworth filter:');
disp(N);
disp('Butterworth window cutoff frequency:');
disp(wn);
%plot frequency response of filter
freqz(z,p);
title('Butterworth frequency response');
```

Output:

Enter passband attenuation in dB:1
Enter stopband attenuation in dB:15
Enter passband frequency in Hz:1500
Enter stopband frequency in Hz:2000
Enter sampling frequency in Hz:8000
Enter type of filter:low
Order of Butterworth filter:
6
Butterworth window cutoff frequency:
0.4088



>>

Experiment No.12

Design and implementation of digital IIR Butterworth high pass filter to meet the given specifications

Aim: To Design and implement of digital IIR Butterworth high pass filter to meet the given specifications

Theory:

A desired frequency response is approximated by a transfer function expressed as a ratio of polynomials. This type of transfer function yields an impulse response of infinite duration. Therefore, the analog filters are commonly referred to as infinite impulse response (IIR) filters.

The main classes of analog filters are:

1. Butterworth Filter.
2. Chebyshev Filter.

These filters differ in the nature of their magnitude response as well as in their design and implementation.

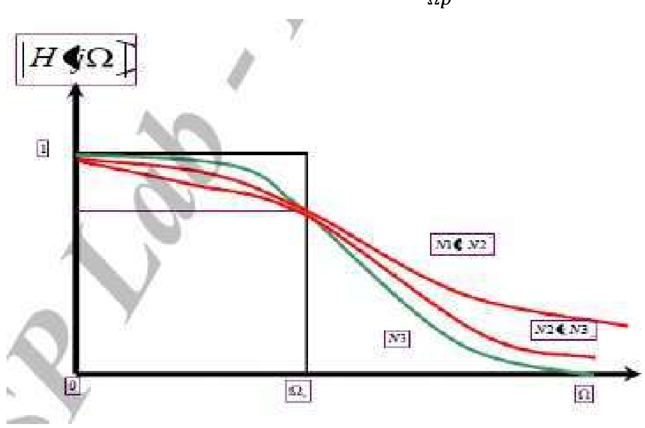
Butterworth Filters:

Butterworth filters have very smooth passband, which we pay for with a relatively wide transition region. A butterworth filter is characterized by its magnitude frequency response,

$$|H(j\Omega)| = 1 / [1 + (\Omega/\Omega_c)2N]^{1/2}$$

Where N is the order of the filter and Ω_c is defined as the cut-off frequency, where the filter magnitude is $1/\sqrt{2}$ times the dc gain ($\Omega = 0$).

$$\text{Order, N} = \frac{log_{10}[\frac{10^{0.1As\,db}-1}{10^{0.1Ap\,db}-1}]}{2\,log\frac{\Omega s}{\Omega p}}$$

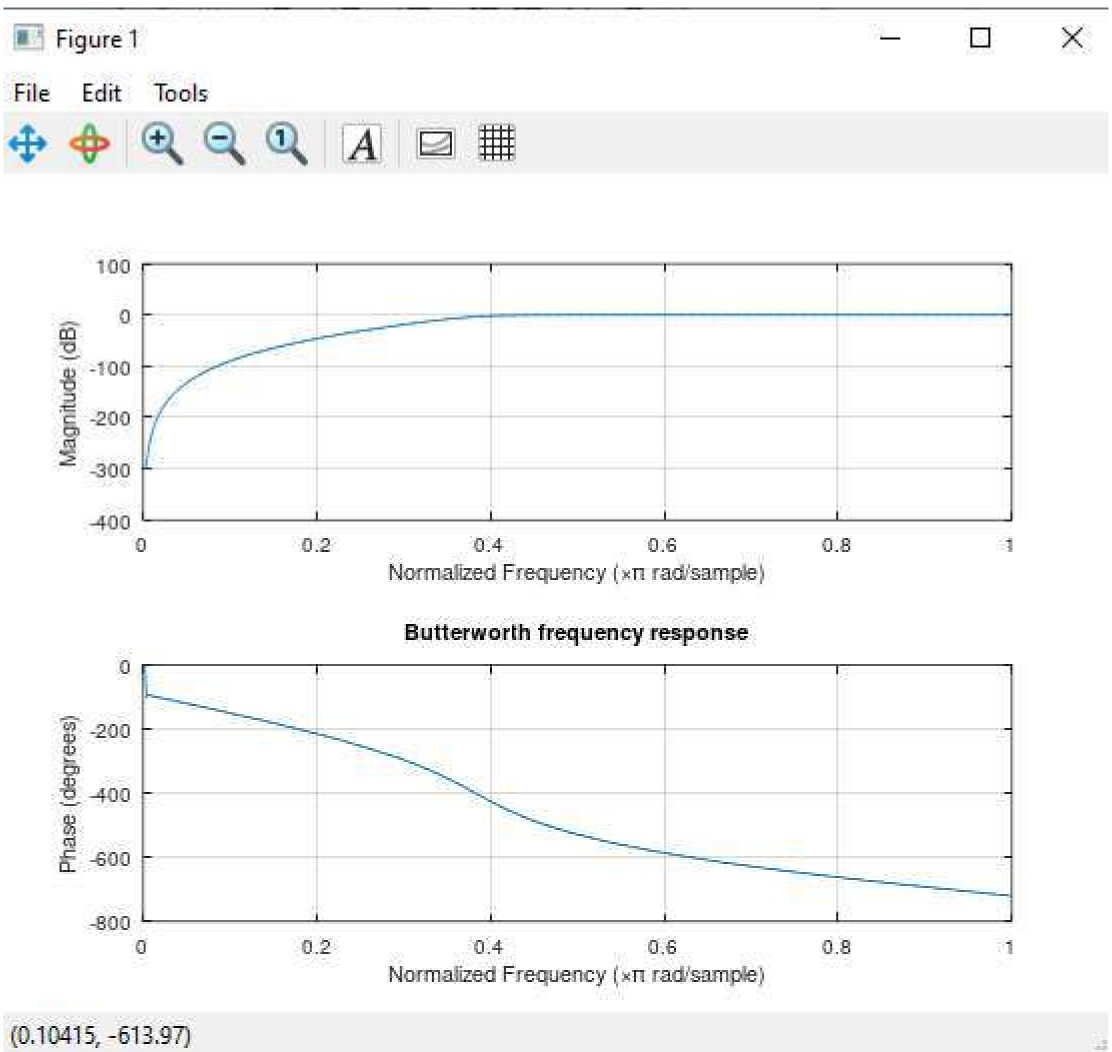


Octave Code:

```
clc; clear all; close all;
Rp=input('Enter passband attenuation in dB:');
Rs=input('Enter stopband attenuation in dB:');
fp=input('Enter passband frequency in Hz:');
fs=input('Enter stopband frequency in Hz:');
Fs=input('Enter sampling frequency in Hz:');
ftype=input('Enter type of filter:','s');
%Calculate sampled frequency values
wp=2*fp/Fs;
ws=2*fs/Fs;
[N,wn]=buttord(wp,ws,Rp,Rs);
%Butterworth filter design
[z,p]=butter(N,wn,ftype);
%display filter parameters
disp('Order of Butterworth filter:');
disp(N);
disp('Butterworth window cutoff frequency:');
disp(wn);
%plot frequency response of filter
freqz(z,p);
title('Butterworth frequency response');
```

Output:

Enter passband attenuation in dB:2
Enter stopband attenuation in dB:19
Enter passband frequency in Hz:1500
Enter stopband frequency in Hz:2000
Enter sampling frequency in Hz:8000
Enter type of filter:high
Order of Butterworth filter:
7
Butterworth window cutoff frequency:
0.3863



>>

Virtual Lab No. 1

Study of FIR filter design using window method: Lowpass and Highpass filter.

Link: http://vlabs.iitkgp.ac.in/dsp/exp8/index.html

Objective:

- Basics of filter designs and different types of filter designing techniques.
- Different types of window functions.
- Designing of Lowpass and highpass FIR filters using these window functions

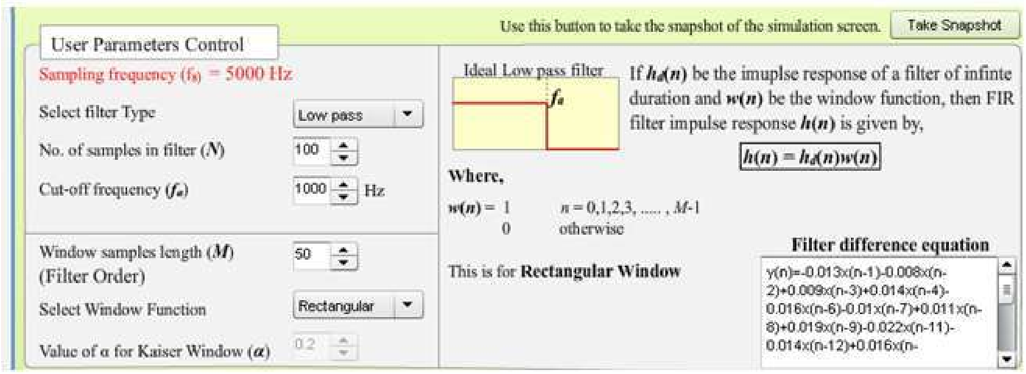
Procedure:

1. Click on the simulator tab SIMULATORIt will open the workspace.

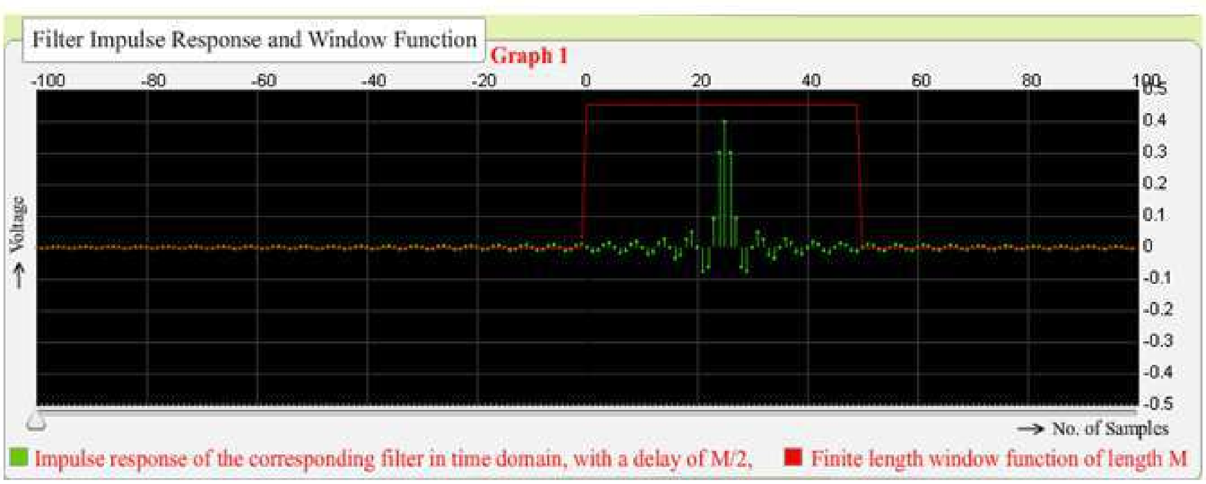
2. See the movie in experiment page by pressing help button to understand how the different steps, as mentioned next, are to be executed..

3. User controls like filter selection, no. of samples in the ideal filter in time domain, cutoff frequency, window length (filter order) and window type selection are given to compare the effect of different windows for designing FIR filters.

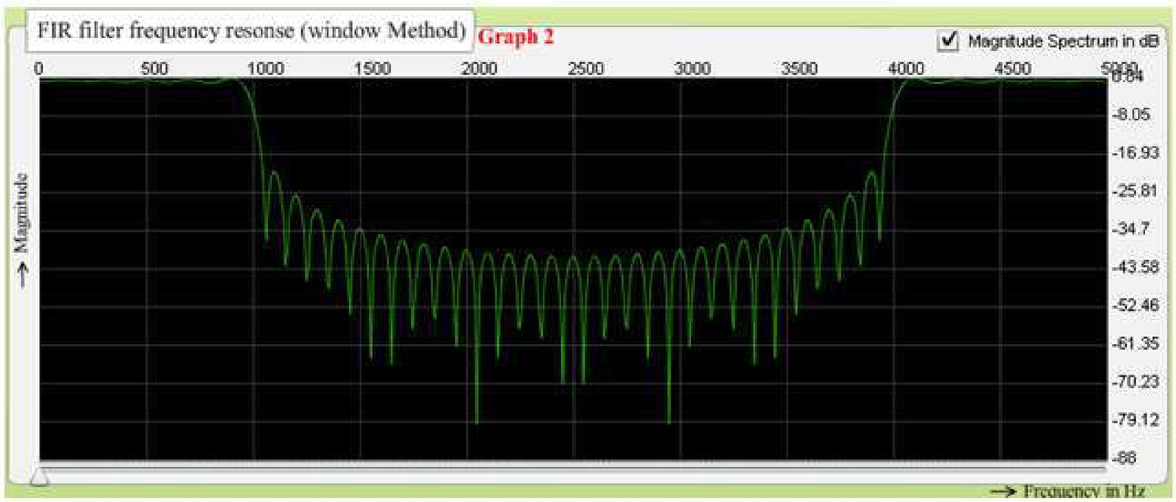
4. In this experiment we have provided two types of filters Lowpass and Highpass filters. The sampling frequency is set to 5000 Hz, you can vary cut-off frequency (fa) from 1000 Hz to 3000 Hz and window length (Filter order M) from 0 to N. You can choose following window functions: Rectangular, Barlett, Hamming, Hanning, Blackman and Kaiser Windows. If you choose Kaiser window function there is a parameter called a through which you can change shape of the Kaiser window.

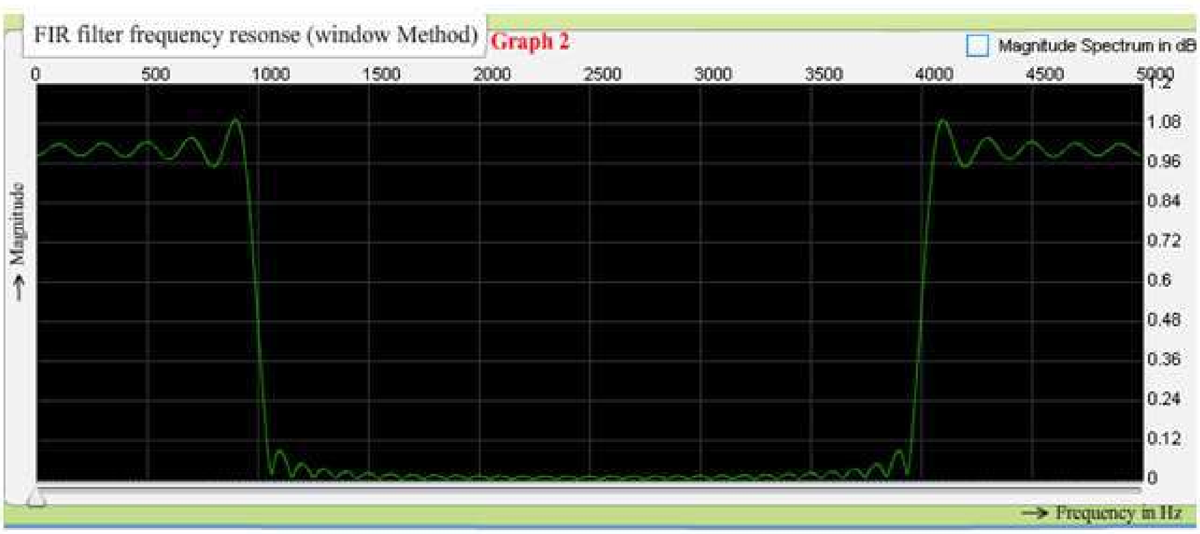


5. Graph 1 plots the impulse response of the filter (Green) in time domain with a time shift of M/2 and Window function (Red) . A slider is given along the x-axis by this you can read sample by sample value for impulse response of the filter.

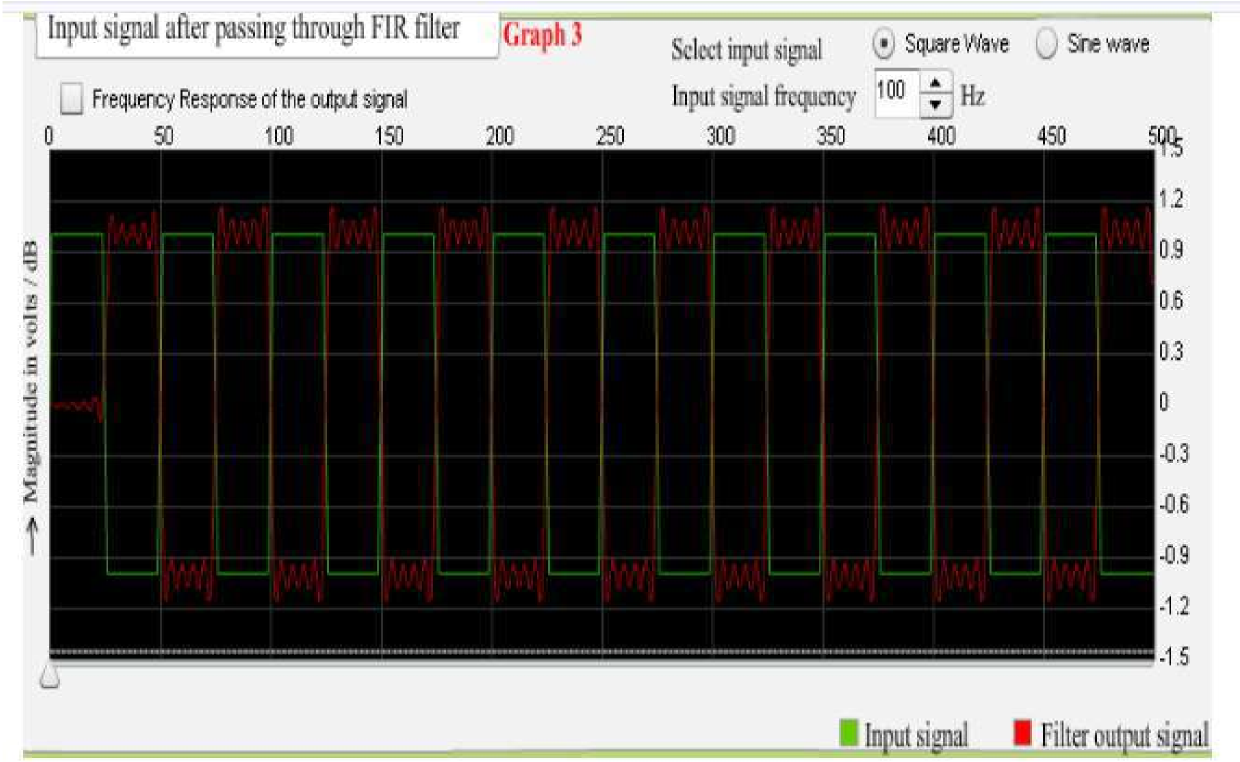


6. Graph 2 plots the magnitude spectrum of FIR filter scale the x-axis range from 0 to 5000 Hz. Here also a slider is provided along the x-axis by which you can read the values at different frequencies. A check box is provided at right top corner on Graph 2 to switch between magnitude spectrum in Decibel and absolute value.

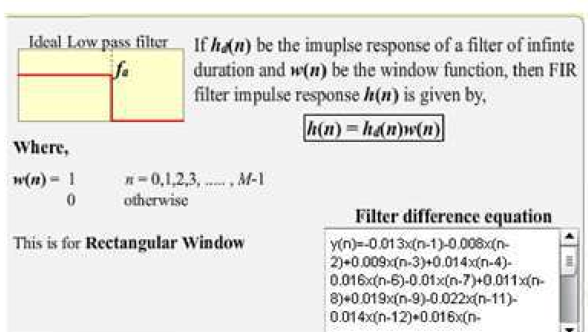




7. Graph no. 3 showing the input signal before and after passing through the filter. In the same graph you can observe the frequency response of the output signal just by checking the checkbox provides.


Input signal after passing through FIR filter
Graph 3
Select input signal
Square Wave
Sine wave
Frequency Response of the output signal
Input signal frequency
100
Hz
0
50
100
150
200
250
300
350
400
450
500
1.5
1.2
0.9
0.6
0.3
0
-0.3
-0.6
-0.9
-1.2
-1.5
→ Magnitude in volts / dB
Input signal
Filter output signal

9.You can choose between sine or square signal as input signal and select input signal frequency.



10.When experiment loads first time the default values are set in all user controls. Initially for the Lowpass filter fs = 5000 Hz, fa = 1000 Hz and M = 50. Rectangular window is being selected by default. Square wave is the input by default and its frequency if 100 Hz.

11.Keeping filter cutoff frequency constant change the input signal frequency and note down input signal and output signal peak to peak values. Which will give you table no. 1 for your report generation.

12.Similarly, choose Highpass filter and repeat the step 10 -12 to note down another set of data which will give you Table no. 2.

13.Now change the window function and do the earlier steps which will give you number of tables.

14.From each table of data you have input and output signal peak to peak values, compute the gain in decible and plot the gain Vs frequency plot which will give you the same plot what you have with filter frequency response (Graph 2).

15.Use take snapshot button to take the screenshot of the experiment space.

16.Complete the Observation and discuss the results in report generation.

17.Then click Yes I have finished my Experiment button to submit your report.

Virtual Lab No. 2

Study of Infinite Impulse Response (IIR) filter

Link: http://vlabs.iitkgp.ac.in/dsp/exp10/index.html

Objective:

- Basics of IIR filter designing and its implimentation.
- Filter designing techniques like Butterworth, Chebyshev 1, Chebyshev 2, Elliptic etc.

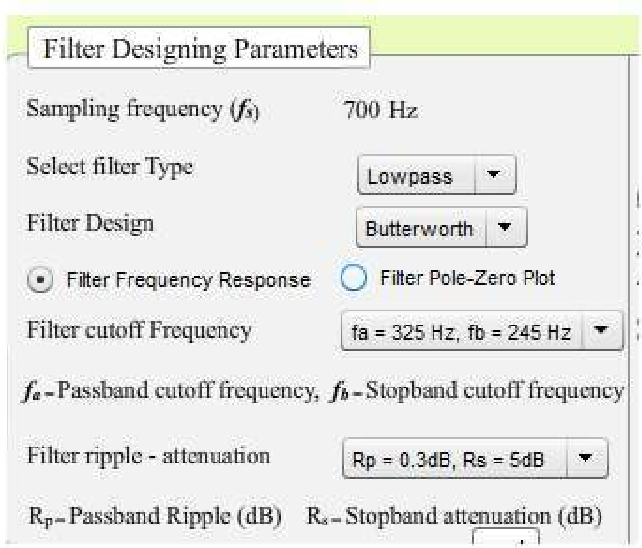
Procedure:

1. Click on the Simulator tab SIMULATORIt will open the workspace.

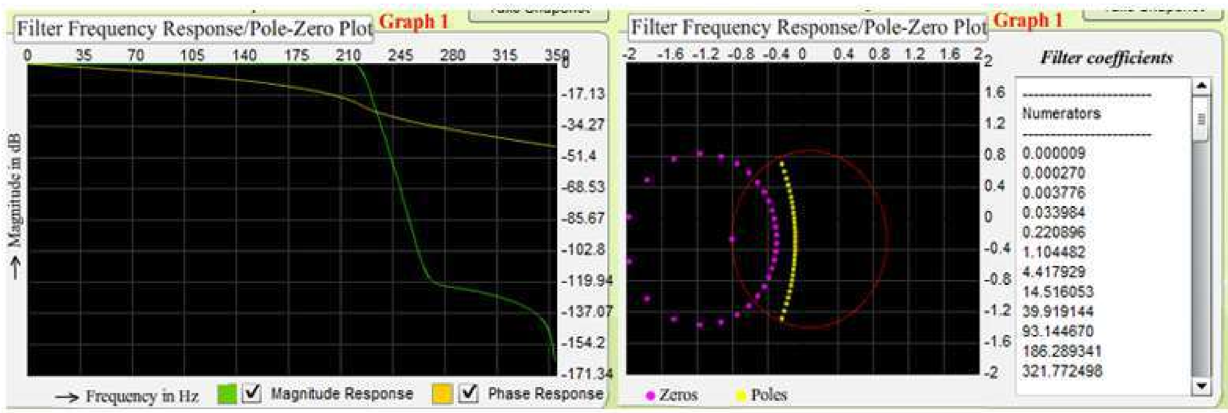
2. See the movie in experiment page by pressing help button ? to understand how the following steps are to be executed.

3. In this experiment we have provided two types of filters Lowpass and Highpass filter. The sampling frequency is set to 700 Hz.

4. We have provided user controls on top left side of the simulation screen. Here you have an option to change filter design technique like Butterworth, Chebyshiv 1 etc. You can change filter passband and stopband frequency from the dropdown menu also the passband ripple and stopband attenuation. You can choose between filter frequency response or pole zero plot.

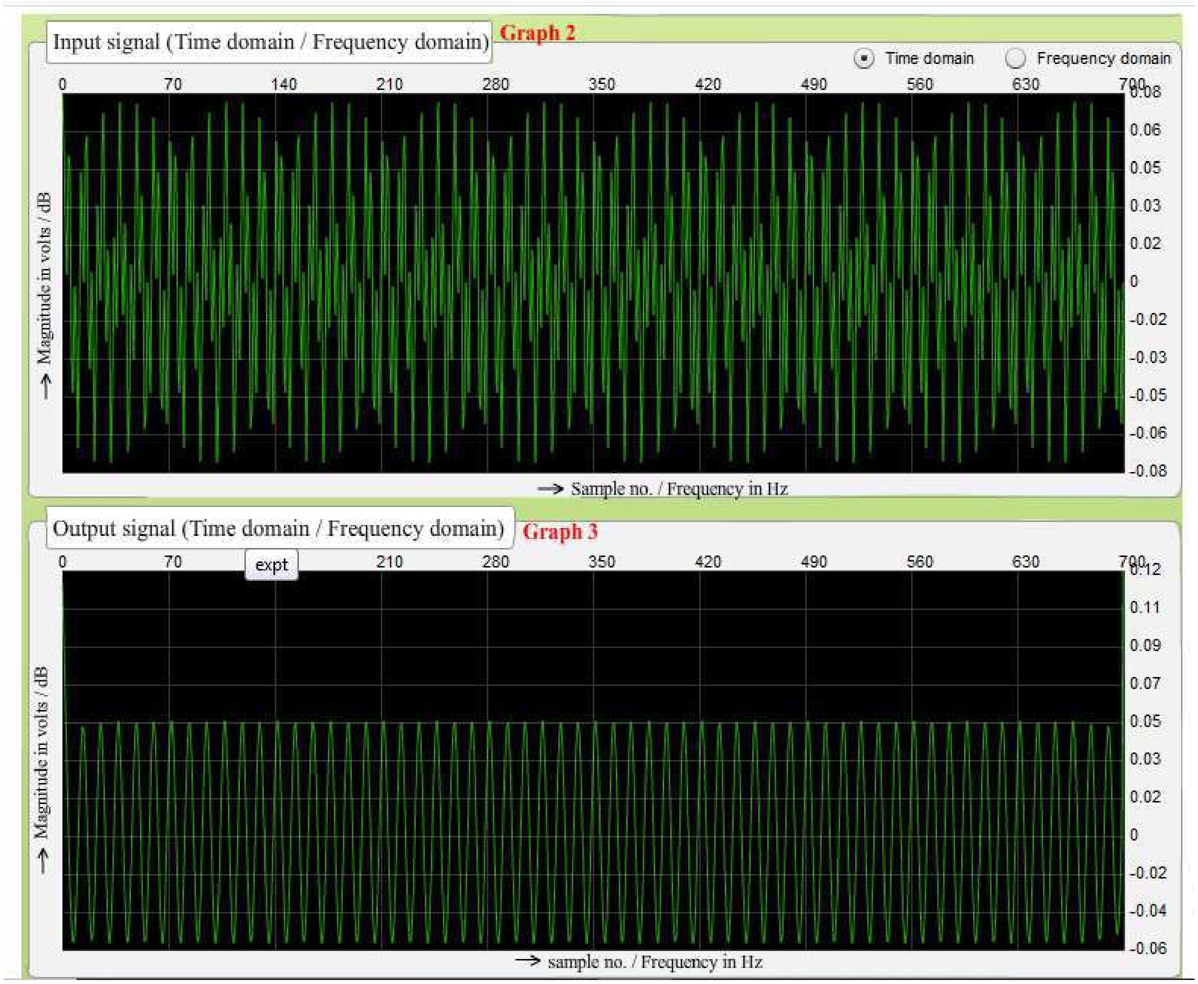


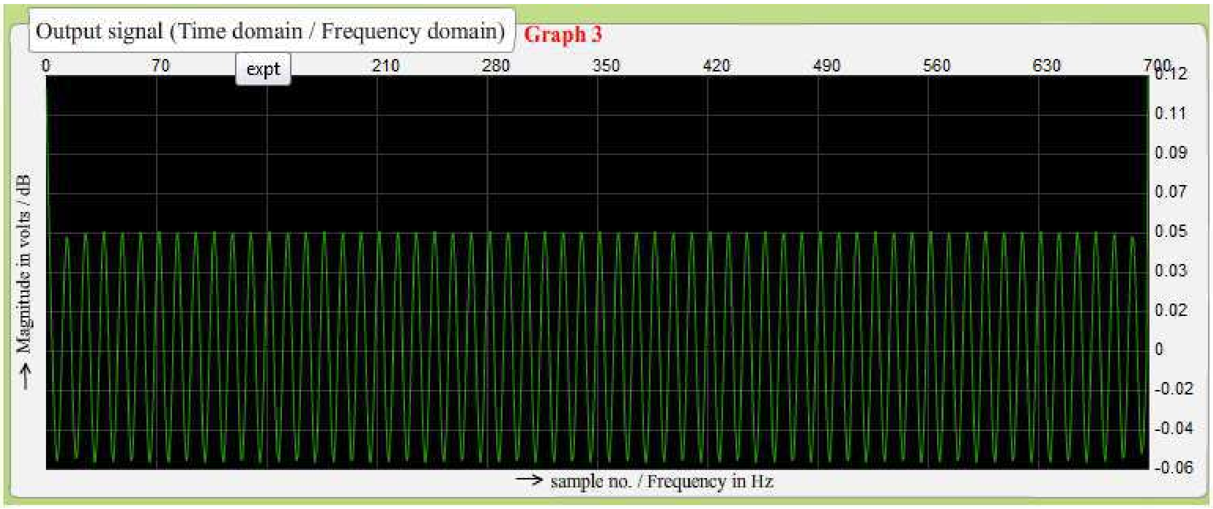
5. Graph 1 plots the filter frequency response and pole zero plot. Two radio buttons has been provided to change the plot from frequency response to pole zero plot vice versa. For frequency response plot you can choose between magnitude or phase response or both by selecting the corresponding checkbox given below the plot. In pole zero plot mode you will get the filter coefficients you can use these filter coefficients for filter designing in any other software or programming.

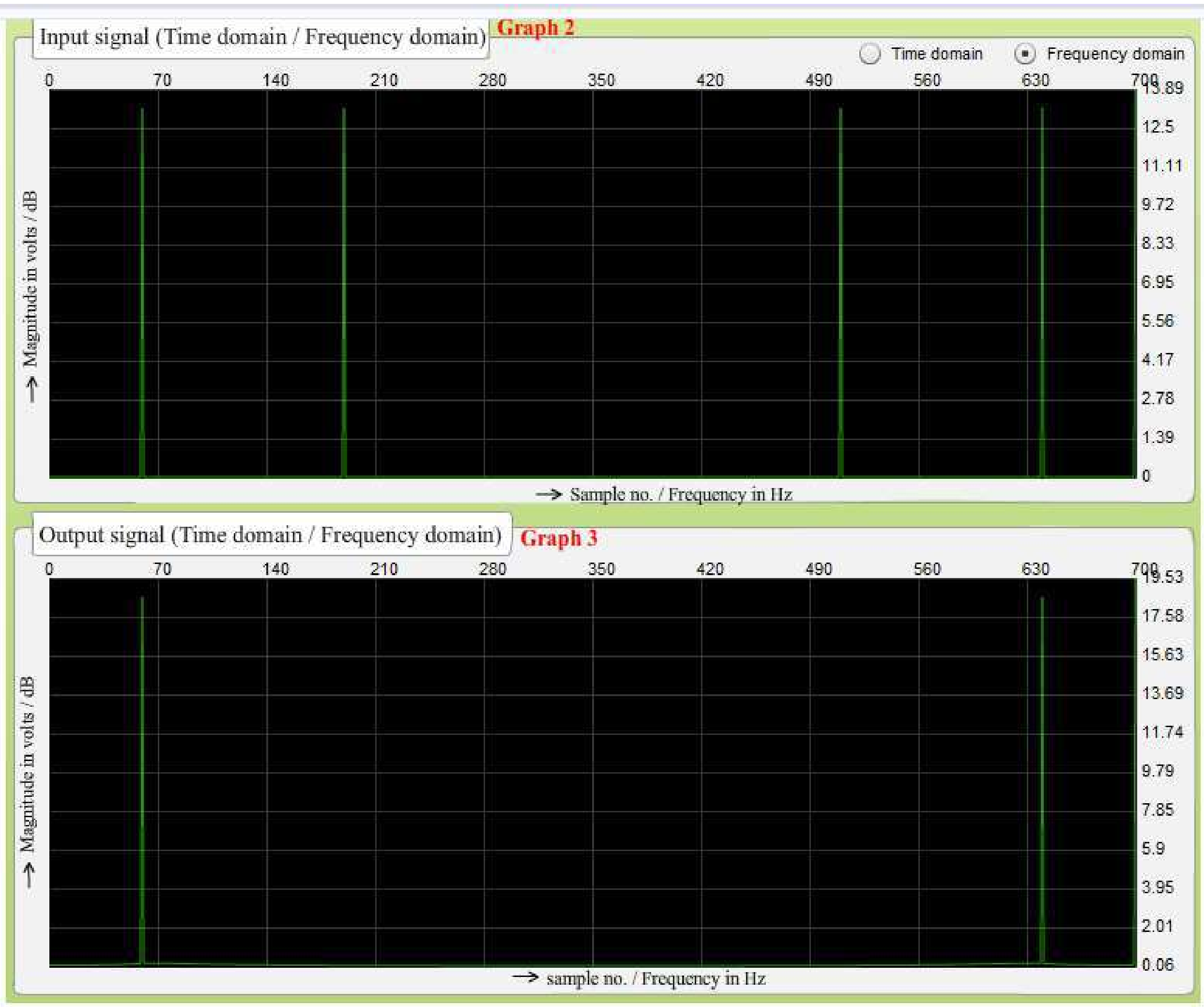


6.Here we have taken sum of two sinusoidals of different frequencies (60Hz & 190Hz) as input so that user can easily understand the filter operation by choosing appropriate cutoff frequencies.

7.Graph 2 and 3 plots input and output signal respectively and also the corresponding frequency domain plots. Here we have given two radio buttons so that user can change from time domain to frequency domain plot.


Input signal (Time domain / Frequency domain)
Graph 2
Time domain
Frequency domain
0 70 140 210 280 350 420 490 560 630 700
0.08 0.06 0.05 0.03 0.02 0 -0.02 -0.03 -0.05 -0.06 -0.08
→ Magnitude in volts / dB
→ Sample no. / Frequency in Hz
Output signal (Time domain / Frequency domain)
Graph 3
expt
0 70 210 280 350 420 490 560 630 700
0.12 0.11 0.09 0.07 0.05 0.03 0.02 0 -0.02 -0.04 -0.06
→ Magnitude in volts / dB
→ sample no. / Frequency in Hz


Output signal (Time domain / Frequency domain)
Graph 3
expt
0
70
210
280
350
420
490
560
630
700
0.12
0.11
0.09
0.07
0.05
0.03
0.02
0
-0.02
-0.04
-0.06
→ Magnitude in volts / dB
→ sample no. / Frequency in Hz



8.Select lowpass filter and Butterworth filter design observe the filter frequency response, pole zero positions by changing cutoff frequency. Simultaneously see the output and compare with input signal. Write the discussion in the report portion of this experiment. Similarly change the filter design to remaining three types of designing techniques and change the cutoff frequency and observe the output.

9.Now change the passband ripple and stopband attenuation to another values and observe the results.

10.Repeat the step 8 and 9 for high pass also and observe the output signal and compare with input signal.

11.Use take snapshot button to take the screenshot of the experiment space

12.Complete the Observation and discuss the results in report generation.

13.Then click Yes I have finished my Experiment button to submit your report.