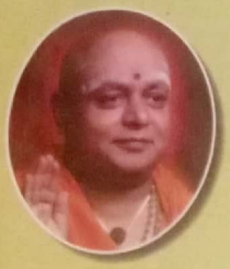


|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust (R.)

BGS INSTITUTE OF TECHNOLOGY

[Affiliated to VTU, Belgaum; Approved by AICTE, New Delhi and Recognized by Govt. of Karnataka)

BG Nagara – 571 448 (Bellur Cross)
Nagamangala Taluk, Mandya District

Practical Record

Name : Manika jain.D

Branch : ECE Sem : 5th

USN : 4BI16EC054

Subject : DSP Lab

CET Code : 142

|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust (R.)

BGS INSTITUTE OF TECHNOLOGY

[Affiliated to VTU, Belgaum; Approved by AICTE, New Delhi and Recognized by Govt. of Karnataka)

BG Nagara – 571 448 (Bellur Cross)
Nagamangala Taluk, Mandya District

Certificate

This is to certify that Mr/Ms. Monika jain D

USN: 4BW 16EC05H *has satisfactorily completed the course of experiments*

in DSP *Laboratory (Course Code:* 15ECL57*)*

prescribed by the Visvesvaraya Technological University, Belagavi for 5th

Semester, BE. Electronics and Communication *Engineering,*

of this College in the year 2018-2019

Record Marks : 12 *Test Marks :* 08

IA Marks : 20

Date : 28/11/18

28/11/18

Staff Incharge

Head of the Department

INDEX

Name of the Student Monika Jain D Class V Sem. Sem. 5th

Expt. No.	Date	Tittle of the Experiment	Page No.	Marks obtained					Sign. of the Staff
1(a)	16/08/18	Introduction	1-10						
1(a)(2)	18/08/18	Generate cos wave in CTS and DTS	11						
1(b)(3)	18-8-18	Generate Sine wave in CTS and DTS	12						
1(a)(4)	23-8-18	Generate Ramp wave in CTS & DTS	13						
1(a)(5)	23-8-18	Generate cos, Sine and ramp wave in CTS	14						
1(a)(6)	23-8-18	Generate cos, Sine and ramp wave in DTS	15						
1(a)(7)	24-8-18	Verification of Sampling theorem	16-19	02	04	02	04	12	8
2(a)	30-8-18	Linear convolution b/w 2 sequence	20						
2(b)	30-8-18	Linear convolution is Commutative	21						
2) (c)	30-8-18	Linear convolution is associative	22						
2) (d)	30-8-18	Linear convolution is distributive	23						
2) (e)	30-8-18	circular convolution b/w 2 sequence	24	02	04	02	04	12	8
2) (f)	30-8-18	Circular convolution is Commutative	25						
2) (g)	30-8-18	Circular convolution is associative	26-27						
2) (h)	30-8-18	Circular convolution is distributive	28-29						
3) (a)	07-9-18	Impulse response in difference eqn	30-31						
3) (b)	07-9-18	Step response in difference eqn	32-33	02	04	02	04	12	8
4) (a)	20-9-18	DFT using built-in function	34-35						
4) (b)	20-9-18	IDFT using built-in function	36						

INDEX

Name of the Student ..... Class ..... Sem. .....

Expt. No.	Date	Tittle of the Experiment	Page No.	Marks obtained				Sign. of the Staff
4) c)	28-9-18	DFT using DFT Equation	37	02	04	02	04	12
4) d)	28-09-18	IDFT using IDFT Equation	38					
5) a)	28-09-18	Analog Butterworth Low pass filter	39-40					
5) b)	28-09-18	Analog Butterworth High pass filter	41					
5) c)	4-10-18	Analog chebyshev Low pass filter	42					
5) d)	4-10-18	Analog chebyshev High pass filter	43	02	04	02	04	12
6) a)	4-10-18	Digital butterworth LPF using bilinear transformation	44-45	02	04	02	04	12
6) b)	4-10-18	Digital butterworth LPF using impulse Invariant Method	46-47					
7) a)	11-10-18	Linearity Properity of DFT	48	02	04	02	04	12
7) b)	11-10-18	Perseval's Theorem of DFT	49					
8) a)	25-10-18	FIR filter using BLACKMAN'S window	50					
8) b)	25-10-18	FIR filter using Hamming window	51					
8) c)	25-10-18	FIR filter using Hanning window	52	02	04	02	04	12
8) d)	25-10-18	FIR filter using Kaiser window	53					
8) e)	25-10-18	FIR filter using rectangular window	54					
9) a)	25-10-18	Auto correlation b/w two sequences	55	02	04	02	04	12
9) b)	25-10-18	Cross correlation b/w two sequence	56					

Name of Experiment ..... Date : .....

Experiment No. ..... Page No.

Expt No	Date	Title of the Experiment	Page No.	Marks obtained				Sign of the staff
		Part-B Hardware Program						
01	15/11/18	Linear Convolution	57	02	04	02	04	(12)
02	15/11/18	Circular Convolution	58	02	04	02	04	(12)
03	15/11/18	Impulse respond	59	02	04	02	04	(12)
04	15/11/18	N-Point DFT	60	02	04	02	04	(12)

Name of Experiment ..Introduction..... Date : 16-08-2018..

Experiment No. ..... Page No. 01

INTRODUCTION TO DSP

DSP means Digital Signal Processing. It is the area of science and engineering that has been developed rapidly over the past 38 years. This rapid development is a result of significant advanced is digital computer technology and integrated circuit fabrication. DSP basically deal with the numerical manipulation of signals and data in digital form having knowledge of elementary operation. Such as digital storage and data in digital form having knowledge storage and delay. Addition, Subtraction and Multiplication by constant one can produce a wide variety of different of different signals.

Block Diagram of DSP

DSP provides an alternative method for processing the analog signal, to perform the processing digitally there is no need for an interface between the analog signal and digital Processor.

This interface is called as analog to digital converter, the output of A/D converter is a digital signal that is appropriate an i/p to the digital domain to the analog domain such as interface is called a digital to analog converter that the signal.

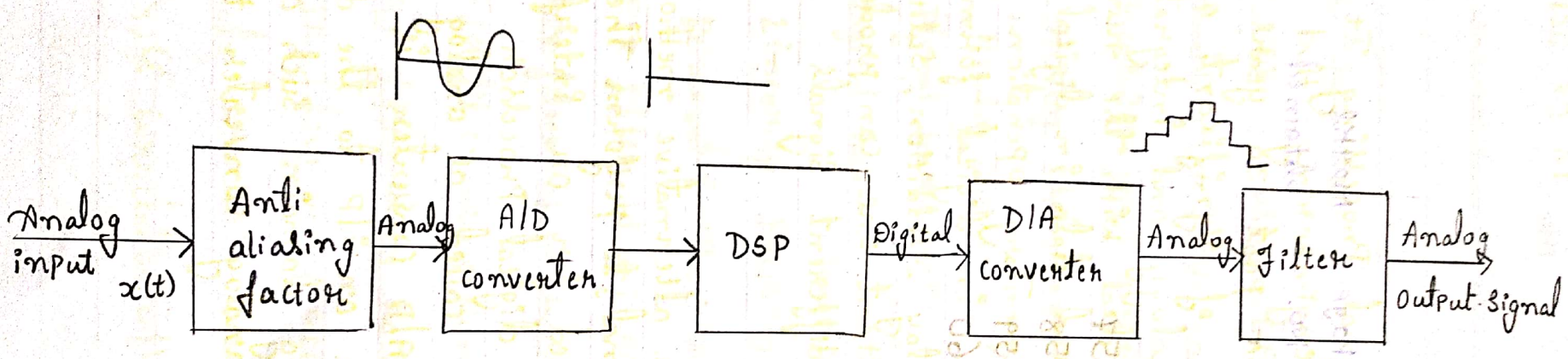


Fig. Block diagram of DSP

Analog to Digital converter :-

Most of the signals like biological signals, radar signals and video signals, communication signals such as audio and video signal by means of its first necessary to convert them into a sequence of number of having finite precision. This procedure is called analog to digital conversion and the corresponding device are called A/D converter.

A/D Conversion has three steps :-

Sampling : This is the conversion of a continous time signal into a discrete time signal into a obtained by taking samples of the continuous time signals at discrete time instant.

Quantisation :- This is the conversion of a discrete time continuous valued signal into a discrete time value signals the valued of each signal sample is represented by a value selected from a finite set of possible values. This difference between the quantized sampled $x(m)$ and the quantized output is called the Quantisation Error.

Coding : In the coding process and discrete values is represented by a bit-bit binary sequence sampling and quantisation are treated in this signal band.

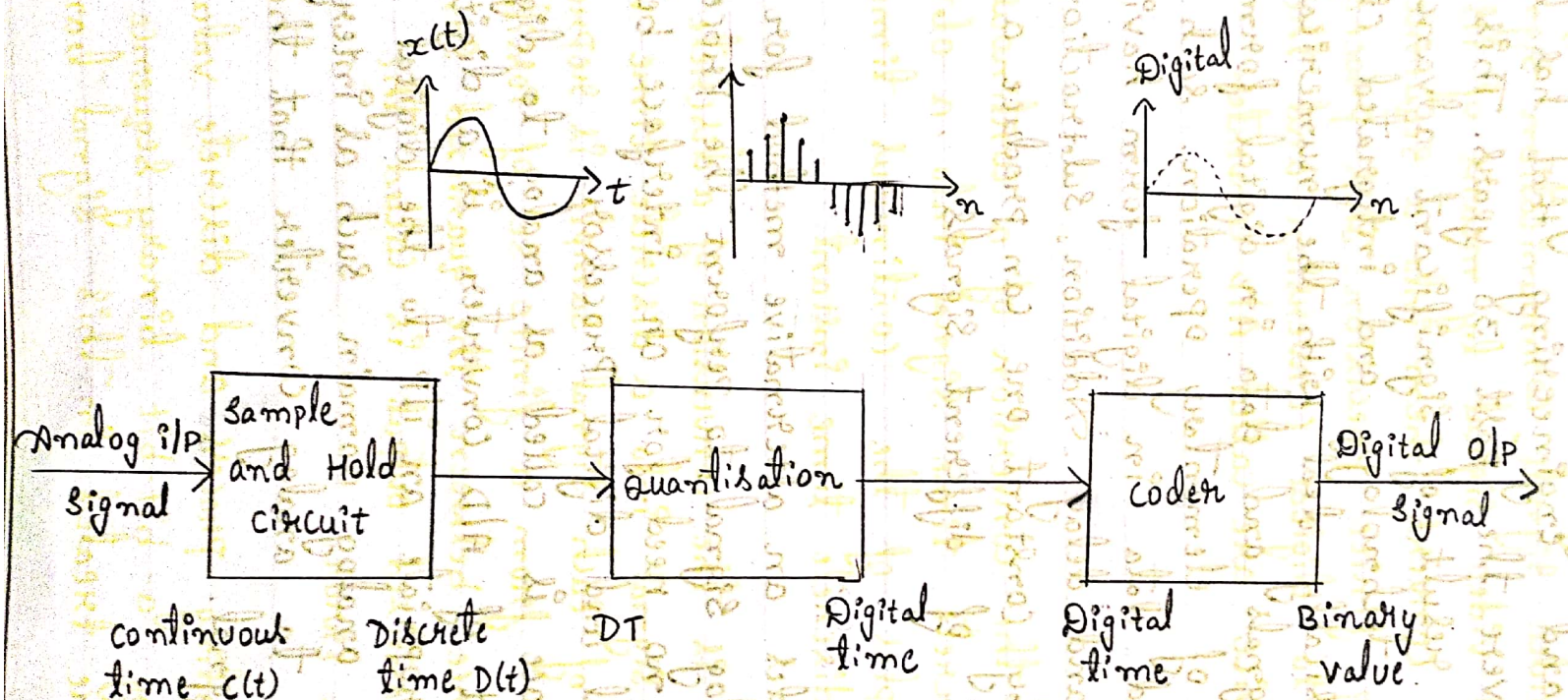


Fig. Block diagram of Analog to Digital converter.

width is finite. The analog signal can be reconstructed from the sufficiently high to avoid the problem commonly called aliasing on the other hand, quantisation is a irreversible process that result in digital distortion is dependent on the accuracy al measured by the number of bits in the A/D conversions process.

The factor affecting the choice of the desired accuracy of the A/D converter are cost & Sampling rate.

Advantages :

- Accuracy.
- Easy to modify the system
- Easy to de-bug
- It has high storage capacity.
- It can be easily simulated.

Limitations :

- Quantisation Error [round-off-Error]
- Bandwidth is limited.
- Power Consumption.

Applications :-

mobile, military - application, Image processing, Digital camera, Computers, medical Instrument. Speech application.

Name of Experiment ..... (6) Date : 16/08/2018

Experiment No. .....

Introduction to MATLAB

MATLAB is a high-performance language for technical computing. It integrator computation and programming in an easy way to use Environment where problems and solutions are Expressed is familiar mathematical notation.

MATLAB is an interactive System where basic data element is an array that doesn't require dimensioning. The name MATLAB Stands for 'Matrix Laboratory' MATLAB was originally written to provide easy access to matrix Software developed by the LINPACK and EISPACK projects in industries. MATLAB and the tool of choice for high-productivity research development and analysis.

MATLAB features a family of add an application Specific Solution called Tool boxes. Very important to most users of MATLAB tool boxes allow you to learn and apply Specialized technology areas in which tool boxes are available include Signal processing, Control Systems, fuzzy logic, Simulation and many others.

MATLAB System consists of 5 major parts :

Desktop tools and development Environment.

This is the set of tools and facilities that help you to use MATLAB functions. It includes desktop

and command windows, editor, debugger and so on.

- MATLAB functions and mathematical functions library.
  This is a vast collection of computational algorithm ranging from elementary functions like Sum, Sine, cosine and Complex

- The MATLAB language.
  This is a high level matrix / array language with control flow statements, functions, data structures, input / output and oriented programming features

- Graphics.
  It has extensive facilities for displaying vectors and matrices of a graphs as well as computing these graphs. It includes high speed level functions for 2D and 3D data visualization, image processing, animation and presentation graphics

- The MATLAB external interface / API.
  This is a library that allows you to write 'c' and FORTAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking) calling MATLAB as a computational engine and for reading and writing MAT-files. This lab manual is intended to get started with MATLAB implementation of DSP algorithm. This test benefied if we wastes Simultaneously on a System.

Name of Experiment ..... (8) ..... Date : .....

Experiment No. .....

MATLAB documentation :

MATLAB provides extensive documentation in both printed and online format to help you learn about and use all of its features If you are a new user, start with the getting Started book, it course all the primary MATLAB features at a high level including many examples.

The MATLAB online help provides task oriented and references information about MATLAB features MATLAB documentation is also available in printed from and in PDI format.

MATLAB ONLINE HELP :

To view the online documentation, Select MATLAB help from the help menu in MATLAB. The MATLAB documentation is organized into their main topics.

Desktop Tools and Development Environment.

- Mathematics - Mathematical operations and data analysis.
- Programming - The MATLAB language and how to develop MATLAB application.
- Graphics - Tools and techniques for plotting graph printing, and programming with handle graphics.
- 3D visualization - Visualization surface and volume data transparency and lighting technique.
- Creating graphical user interface - GUI - building tools and how to write call back fun.

Name of Experiment ⑨ Date :

Experiment No. Page No. 07

- External interface / API - mex files, the MATLAB engine and interfacing to Java, command the part.
- Functions by category - Lists all MATLAB fun grouped into categories.
- Handle graphics property browser - Provide easy occurs to description of graphics object properties.
- External interface / API reference - Converse those fun used by the MATLAB external interfaces providing information as syntax in the calling language, description arguments return values and examples.
- Examples - An index of examples included in the document.
- Release Notes - New features and known problems in current.
- Printable Documentation - PDF Provides various of the documentation suitable for Printing.

Name of Experiment (10) Date : 16/08/18

Experiment No. Page No. 08

Basic Commands used in MATLAB.

clc : clear the command window

Syntax : clc.

Description : clc clears all input and output from the command window display, giving you a "clean screen".

Input : Request user input

Syntax : result = input (prompt)

Description : To evaluate expression, the input fun accesses variables in the current workspace.

Eg clc ;

```
a = input ('enter the first sequence');
b = input ('enter the second sequence');
```

Disp : display text or array.

Syntax - disp (x)

description - command used to displays the contents of x without printing the variable name.

Eg - disp ('The output of the system x=');

Stem : plot discrete

Syntax - stem (x, y)

Description : It plots the data sequence y at values specified by x. The x and y inputs must be vectors or matrix of the same size.

Eg - n=0 : length (A) - 1 ;

stem (n, A);

plot : 2-D line plot
Syntax - Plot (x, y)
Description - plot (x, y) creates a 2-D line plot of the data y versus the corresponding values in x
Eg: plot (x, y).

Subplot : create axes in tilled positions.
Syntax - Subplot (m, n, p)
Description - It divides the current figure into an m-by-n grid and creates an axes in the grid position specified by P.

x-label : Label x-axis
Syntax - xlabel (Str)
Description - It labels the x-axis of the current axes with the string, Str. Each axes graphics object has one predefined x-axis label.
Eg - x-label ('Samples');

y-label : Label y-axis
Syntax - ylabel (Str)
Description - ylabel (Str) labels the y-axis of the current axes with the string, Str. Every axes has one predefined y-axis label.
Eg - y-label ('amplitude');

title : Add title to current axes.
Syntax - title (Str)

Description - title(str) adds the title consisting of a string, str at the top and in the center of the current axes.
Eg - title('impulse');

Length : Length of vector or largest array dimension
Syntax - number of Elements = length(array).
Description - It finds the number of Elements along the largest dimension of an array.

Ones : Create array of all ones.
Syntax : x = ones.
Description - x = Ones returns the scalar 1.

Zeros : Create array of all zeros.
Syntax : x = zeros
Description - Zeros returns the scalar 0.

Cos : Cosine of argument in radians.
Syntax : Y = cos(X)
Description : Y = cos(X) returns the cosine for each Elements of X. The cos function operates Element on arrays function accepts both real and Complex inputs For purely real values or imaginary values of x, cos returns real value in the interval [-1, 1]. For Complex values of x, cos returns complex values. All angles are in radians.

19/08/18

Name of Experiment ... Cos wave ..... Date : 18-8-2018

Experiment No. ... 1(a) ..... Page No. 11

Write a MATLAB code to generate cos wave in continuous-time signal.

```
clc;
f=input ('enter the frequency=');
t=0:0.001:0.1;
y=cos (2*pi*f*t);
disp ('the values of y are =')
disp (y)
plot (t,y)
xlabel ('time');
ylabel ('amplitude');
title ('cose wave');
```

Write a MATLAB code to generate cos wave in discrete-time signal

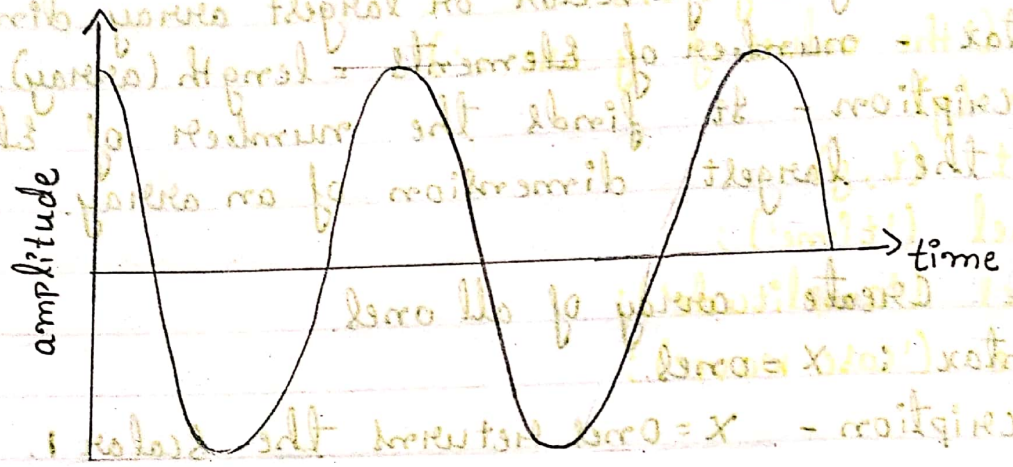
```
clc;
f=input ('enter the input frequency=');
n=0:0.001:0.1;
y=cos (2*pi*f*n);
disp ('the values of y are =')
disp (y)
stem (n,y)
xlabel ('time');
ylabel ('amplitude');
title ('cose wave');
```

BGSIT

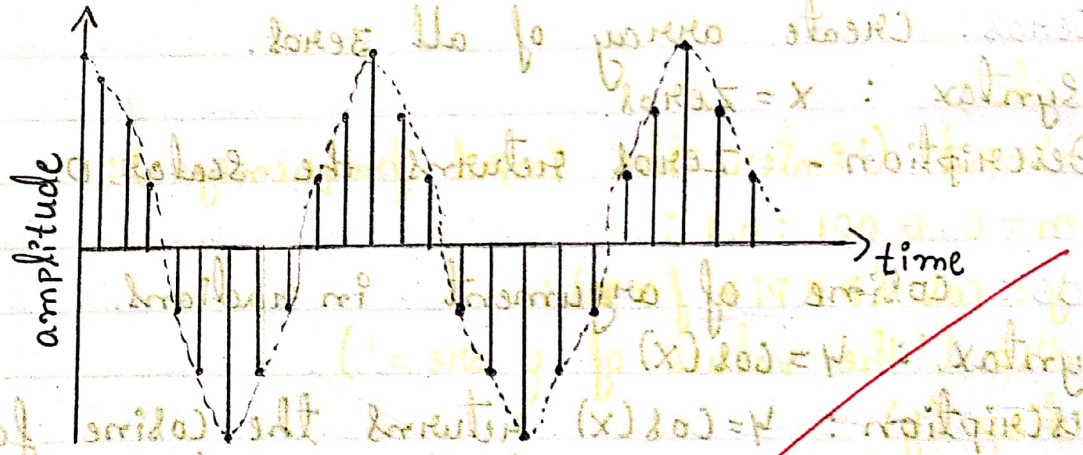
output :

Enter the input frequency = 100

cos wave in CTS



cos wave in DTS.



Name of Experiment ...Sine wave... Date : 18-08-18

Experiment No. ...1(b)... Page No. 12

Write a MATLAB code to generate Sine wave in continuous time Signal.

```
clc;
f = input ('Enter the frequency =');
t = 0: 0.001 : 0.1;
y = sin (2 * pi * f * t);
disp ('the values of y are=')
disp (y)
plot (t,y)
xlabel ('time');
ylabel ('amplitude');
title ('Sine wave');
```

Write a MATLAB code to generate Sine wave in discrete time Signal

```
clc;
f = input ('Enter the input frequency =');
n = 0: 0.001 : 0.1;
y = sin (2 * pi * f * n);
disp ('the values of y are=')
disp (y)
stem (n, y)
xlabel ('time');
ylabel ('amplitude');
title ('Sine wave');
```

Keyword :

Sin = Sine of argument in radians.

Syntax : $y = \sin(x)$

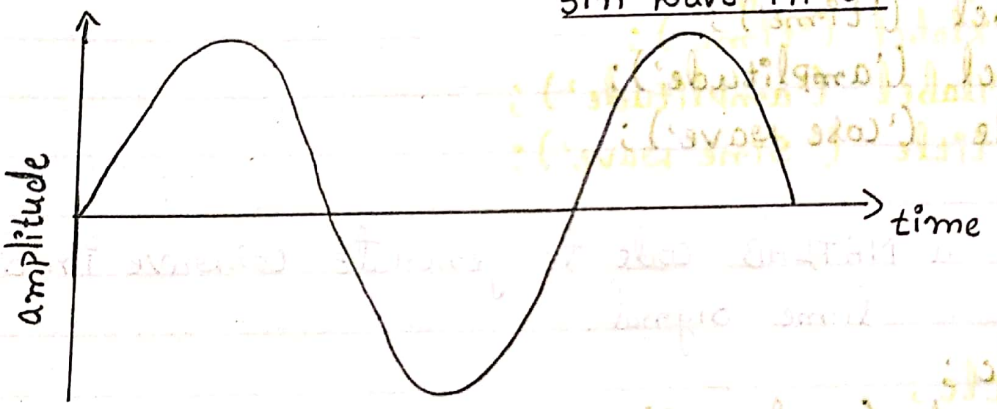
Description : $y = \sin(x)$ returns the circular sine of the element of x. The sin function operates element-wise on arrays. The functions domain and ranges include complex values.

All angles are in radians.

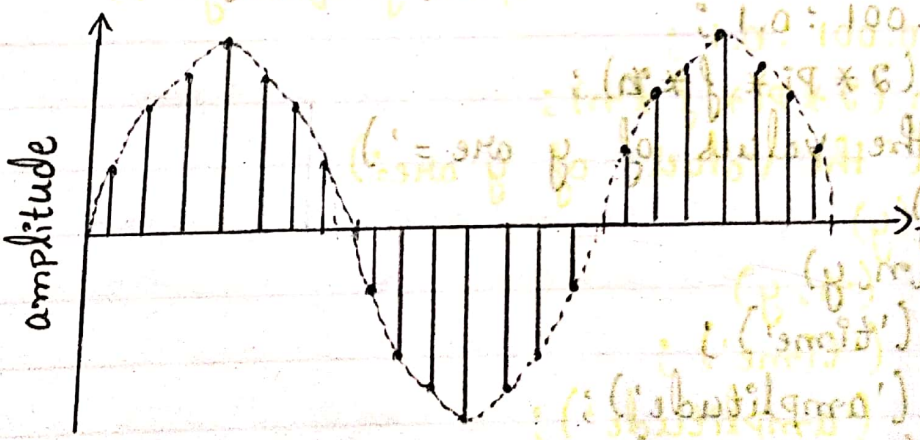
Output :

Enter the input frequency = 100

Sin wave in CTS.



Sin wave in DTS.



Name of Experiment ..... RAMP Wave ..... Date : 23-08-18

Experiment No. ..... 1(c) .....

Write a MATLAB code to generate RAMP wave in continuous time signal.

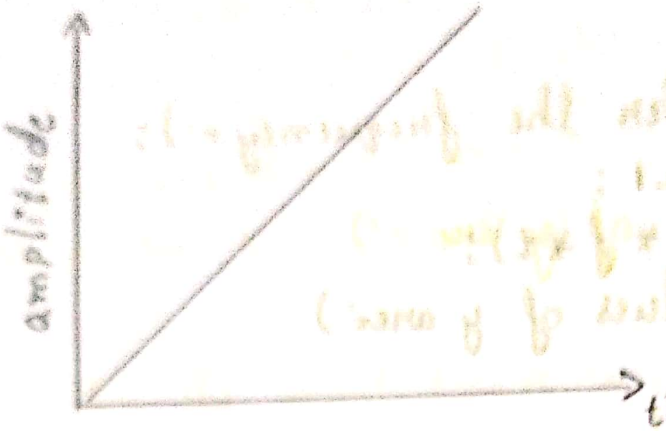
```
clc;
t=0:1:10;
y=t
disp ('the values of y are =')
disp (y)
plot (t, y)
xlabel ('time');
ylabel ('amplitude');
 title ('ramp');
```

Write a MATLAB code to generate Ramp wave in discrete time signal.

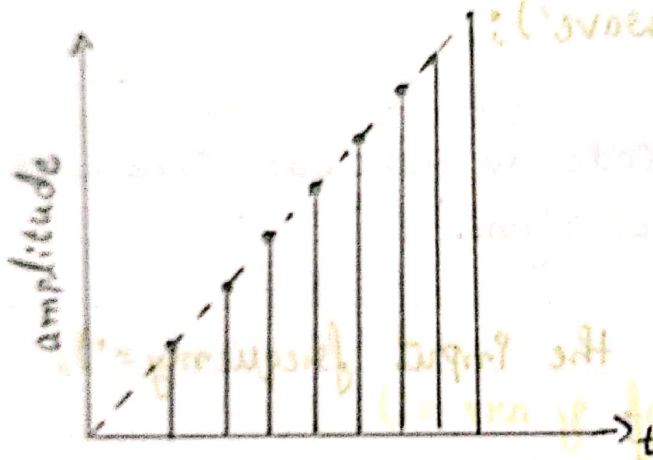
```
clc;
n=0:1:10;
 y=n
disp ('the value of y are =')
disp (y)
stem (n, y)
xlabel ('time');
ylabel ('amplitude');
 title ('ramp');
```

Output :-

Ramp wave in CTS



Ramp wave in DTS



Name of Experiment ....Cos, Sine and ramp wave Date : 23-08-18

Experiment No. ....1(d)

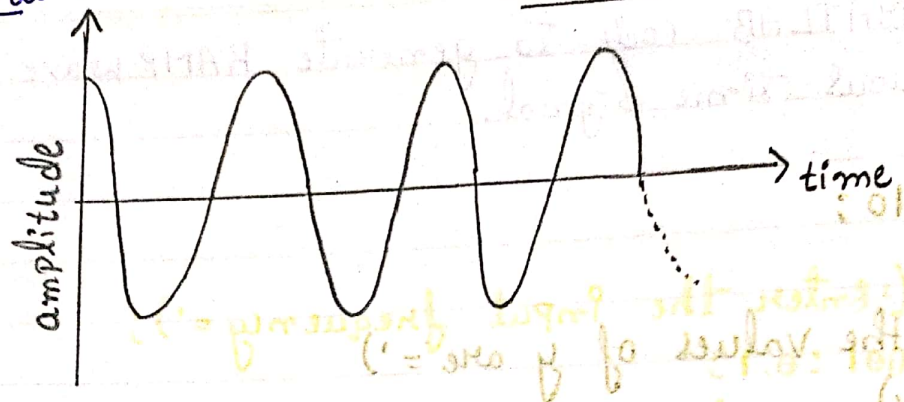
Write a MATLAB code to generate cos, sine and ramp wave in continuous Time signal.

```
clc;
f=input('Enter the input frequency=');
t=0:0.001:0.1;
x=cos(2*pi*f*t);
disp('the values of x are=')
disp(x)
subplot(3,1,1)
plot(t,x)
xlabel('time');
ylabel('amplitude');
title('cosewave');
y=sin(2*pi*f*t);
disp('the values of y are=')
disp(y)
subplot(3,1,2)
plot(t,y)
xlabel('time');
ylabel('amplitude');
title('sine wave');
z=t;
disp('the values of z are=')
disp(z)
subplot(3,1,3)
plot(t,z)
xlabel('time');
ylabel('amplitude');
title('rampwave');
```

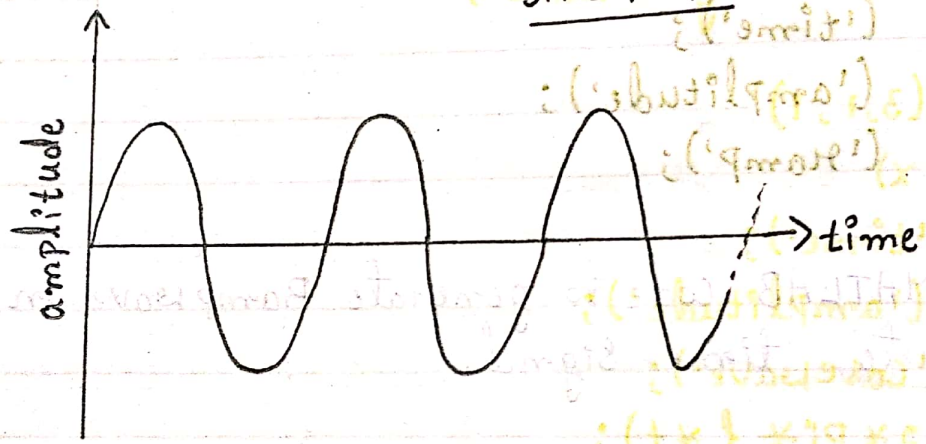
Enter the input frequency = 100

Output :

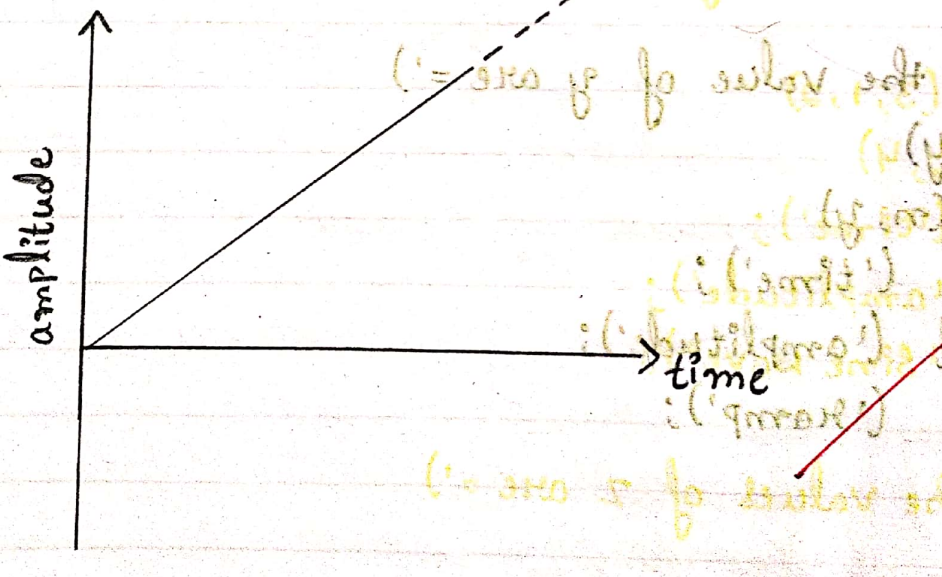
Cos wave



Sine wave



Ramp wave



Name of Experiment ..cos, Sine and Ramp Wave in DT Date : 23-8-18

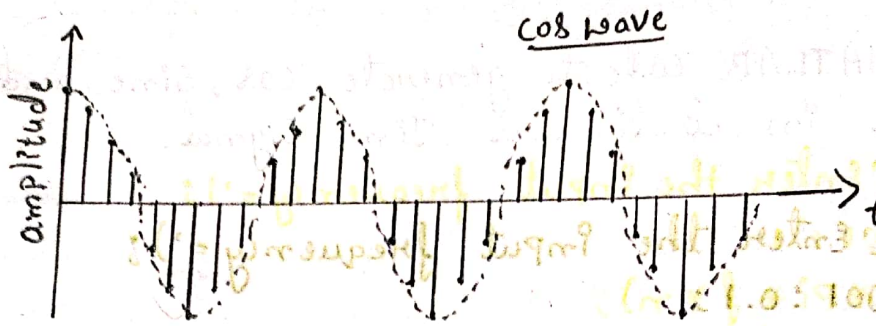
Experiment No. ..1(e)

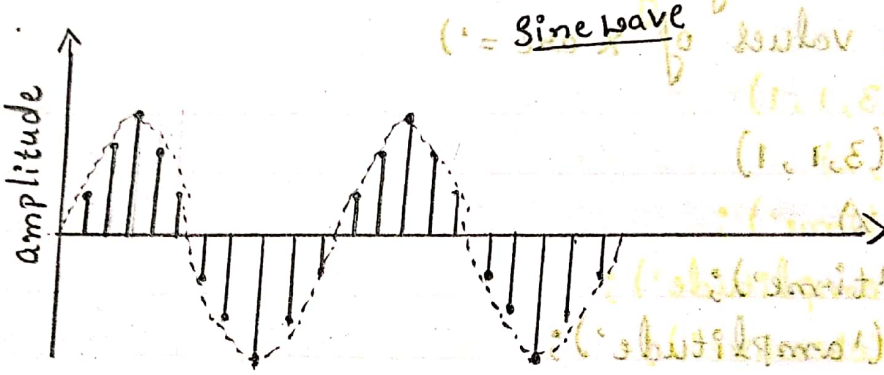
Write a MATLAB code to generate cos, Sine and ramp wave in Discrete time signal.

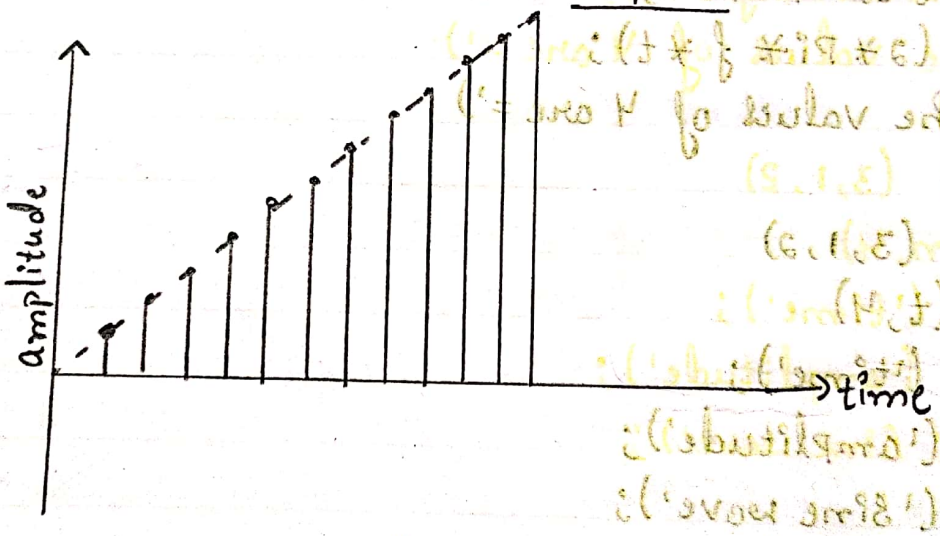
```
clc;
f = input ('enter the input frequency =');
n = 0:0.001 : 1
x = cos (2 * pi * f * n);
disp ('the values of x are =')
disp (x)
subplot (3, 1, 1)
stem (n, x)
xlabel ('time');
ylabel ('amplitude');
title ('cosewave');
Y = sin (2 * pi * f * n);
disp ('the values of Y are =').
disp (Y)
subplot (3,1,2)
stem (n, Y)
xlabel ('time');
ylabel ('amplitude');
title ('Sine wave');
z = n;
disp ('the values of z are =')
disp (z)
subplot (3,1,3)
stem (n, z)
xlabel ('time');
ylabel ('amplitude');
title ('ramp wave');
```

Output :-

Enter the input frequency = 100.







VERIFICATION OF SAMPLING THEOREM.

Aim: To verify the Sampling Theorem.

Theory:

Sampling: It is the process of converting a continuous time signal into a Discrete time signal. It is the first step in conversion of analog signal to digital signal.

Sampling Theorem: It states that Exact reconstruction of a constant time base band signal from its samples is possible if signal is band time and sampling frequency is greater than, twice the signal bandwidth i.e $f = 2W$ 'W' is the signal bandwidth.

Nyquist rate Sampling: The Nyquist Rate is minimum sampling rate required to avoid aliasing equal to twice the highest modulation frequency contains within signal.

Over Sampling: Whenever the sampling rate exceeds the Nyquist rate the condition is known as over sampling.

Under Sampling: Whenever the sampling rate less than the Nyquist rate the condition is known as under sampling.

Name of Experiment ..... Date : 24-08-18

Experiment No. VI Page No. 17

Write a MATLAB code to verify Sampling Theorem

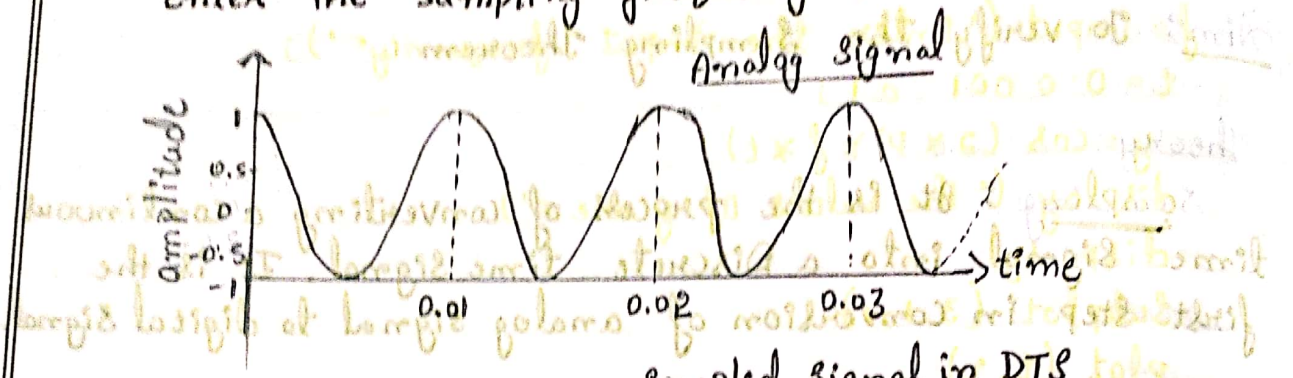
```
clc;
f = input ('Enter the input frequency=');
t = 0:0.001 :0.1;
y = cos (2 * pi * f * t);
display ('the values of y are =')
disp (y)
subplot (3,1,1)
plot (t, y)
xlabel ('time');
ylabel ('amplitude');
title ('analog signal');
fs = input ('Enter the sampling frequency=');
ts = 1/fs;
tn = 0:ts:0.1;
ys = cos (2 * pi * f * tn);
disp ('the values of ys are =')
disp (ys)
subplot (3,1,2)
stem (tn, ys)
xlabel ('time');
ylabel ('amplitude');
title ('sampled signal in DTS');
subplot (3,1,3)
plot (tn, ys)
xlabel ('time');
ylabel ('amplitude');
title ('sampled wave in CTS');
```

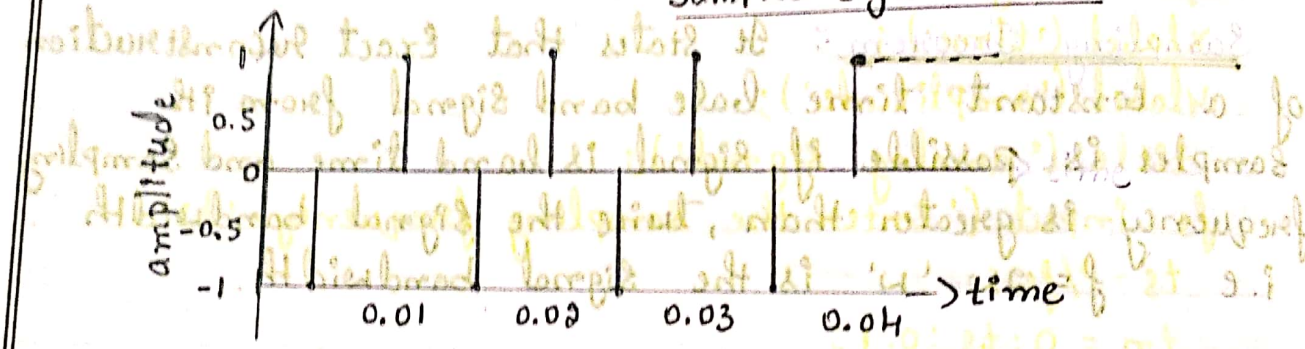
Output:

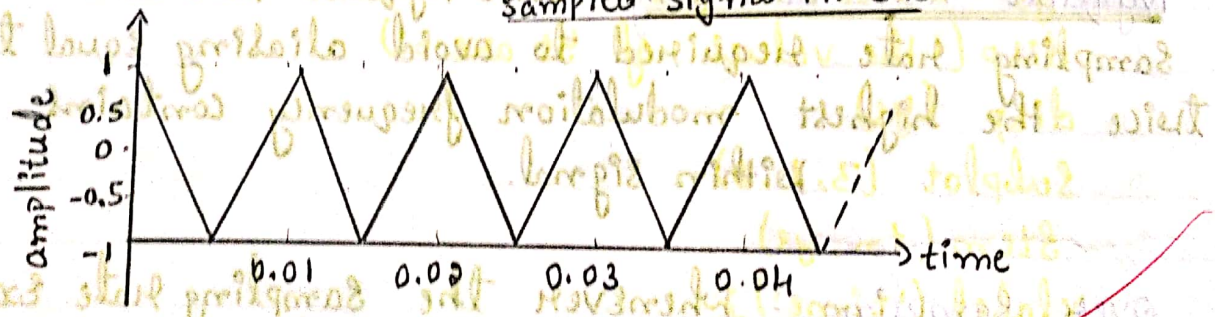
When $f_s = 2f_{max}$ Nyquist rate.

Enter the input frequency = 100

Enter the sampling frequency = 200

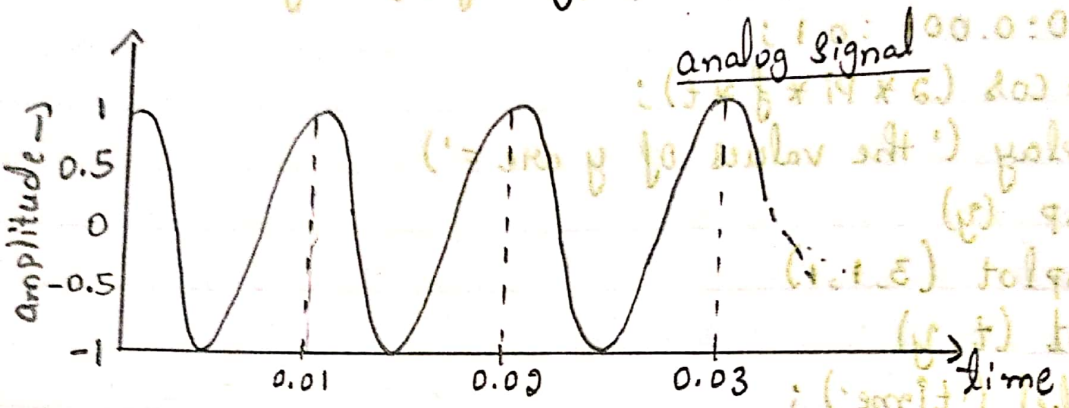


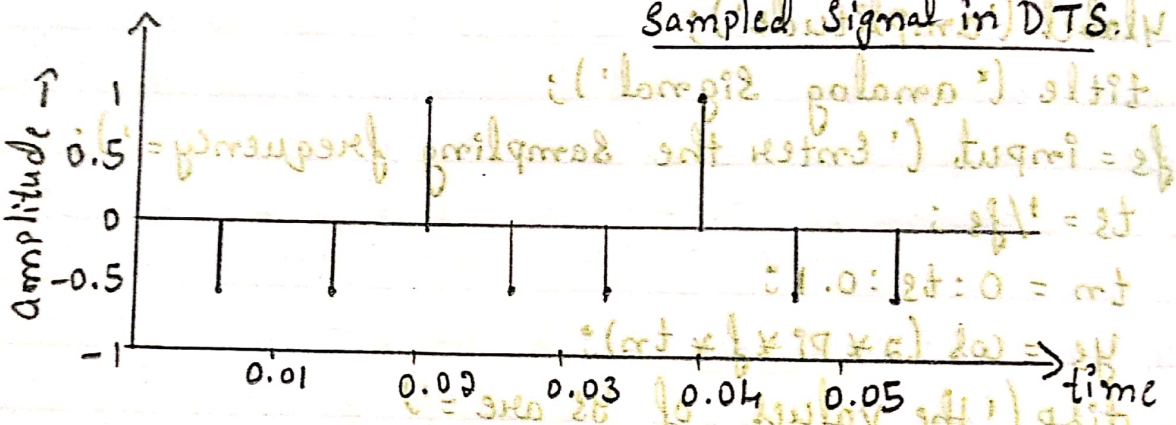


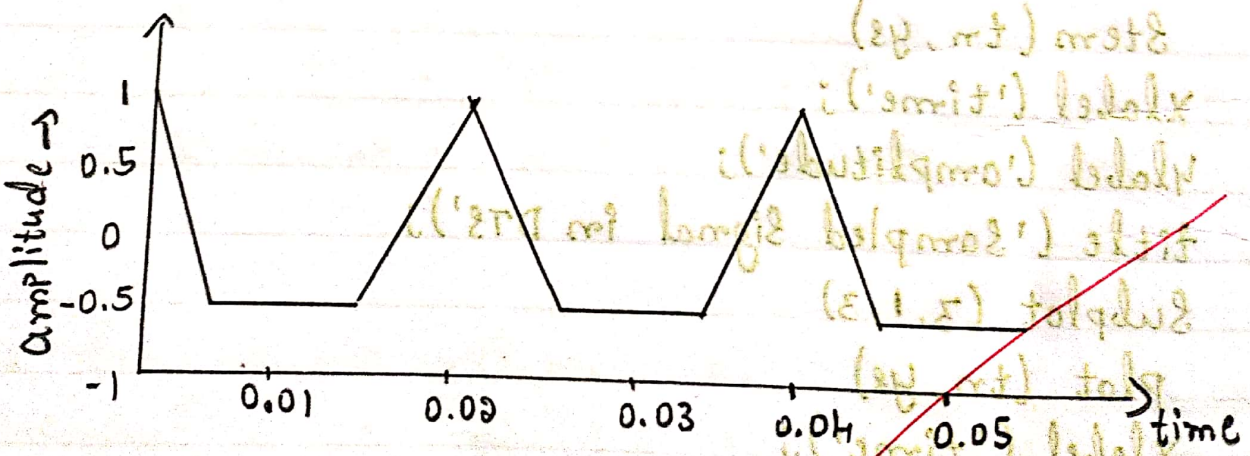


When $f_s < 2f_{max}$ = under sampling.

Enter the input frequency = 100
Enter the sampling frequency = 150.





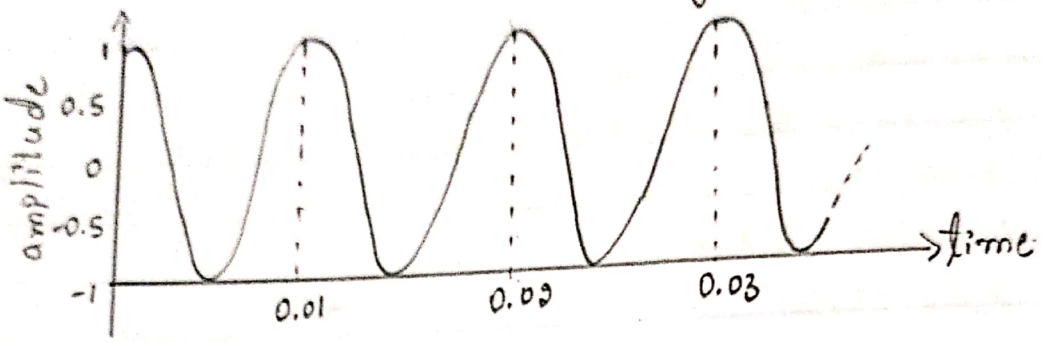


When $f_s > 2f_{max}$: over sampling.

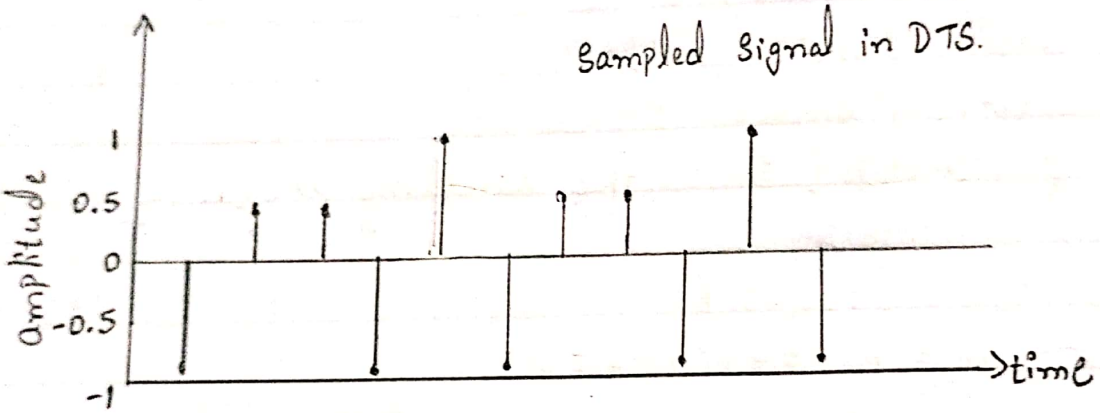
Enter the input frequency = 100

Enter the sampling frequency = 250.

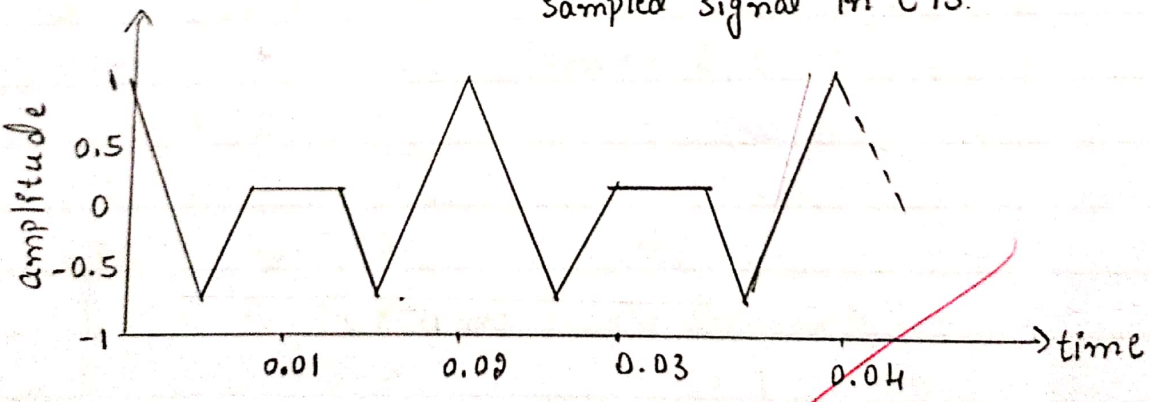
Analog signal



Sampled signal in DTS.



Sampled signal in CTS.



12/12 24/8/18

Name of Experiment ..... Linear convolution Date : 30-08-18

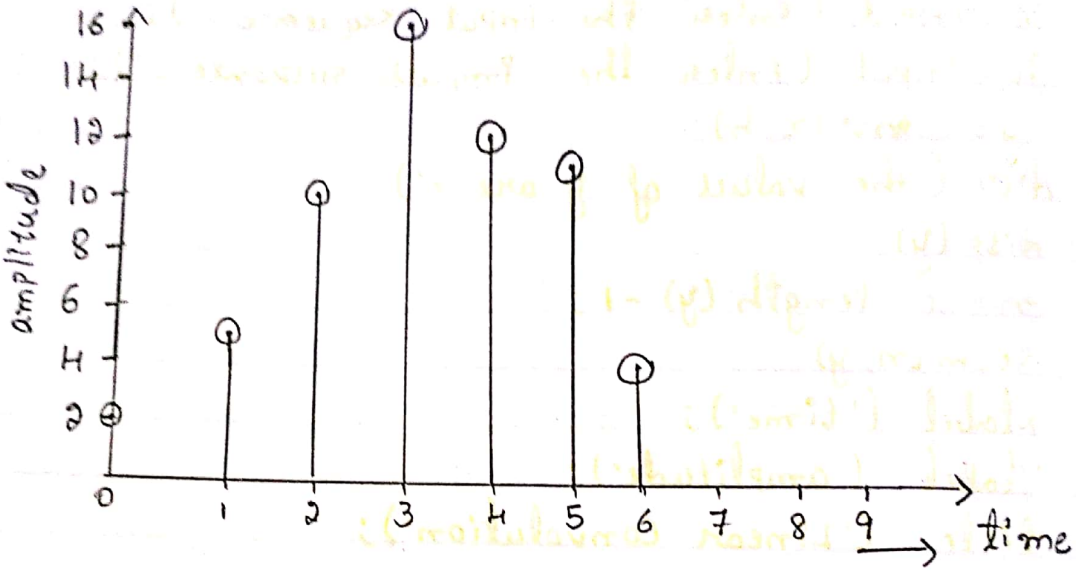
Experiment No. ..... VII 2(a) Page No. 20

Write a MATLAB code to perform Linear convolution between the two sequences.

```
clc;
x = input ('Enter the input sequence =');
h = input ('Enter the impulse response =');
y = conv (x, h);
disp ('the values of y are =')
disp (y)
n = 0: length (y) - 1;
stem (n, y)
xlabel ('time');
ylabel ('amplitude');
title ('Linear convolution');
```

Output :

Enter the input sequence = [1 2 3 4]
Enter the impulse response = [2 1 2 1]
The values of y are = [2 5 10 16 12 11 4]



Verification :

$x(n) = [1\ 2\ 3\ 4]$
$h(n) = [2\ 1\ 2\ 1]$

	1	2	3	4
2	2	4	6	8
1	1	2	3	4
2	2	4	6	8
1	1	2	3	4

$y(n) = \{2, 5, 10, 16, 12, 11, 4\}$

Name of Experiment: linear convolution is commutative 30-08-18

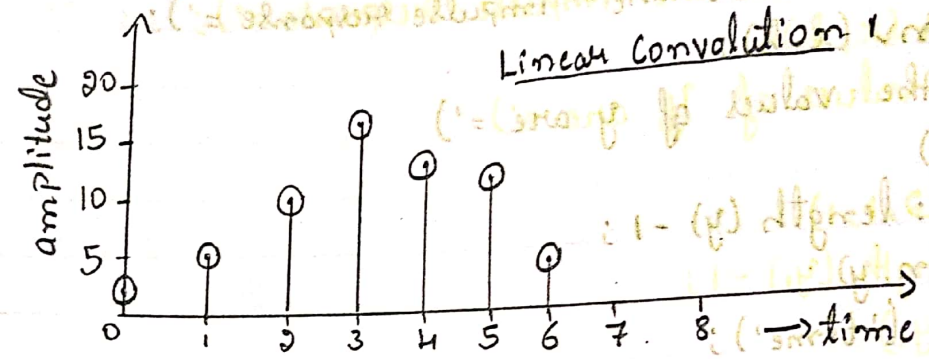
Experiment No.: VIII 2(b) Page No. 21

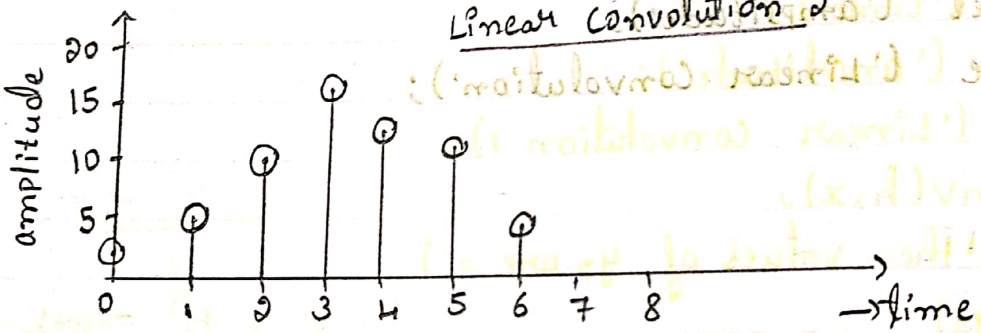
Write a MATLAB code to verify linear convolution is commutative [x(n) * h(n) = h(n) * x(n)]

```
clc;
x = input ('Enter the input sequence =');
h = input ('Enter the impulse response =');
y1 = conv(x, h);
disp('the values of y1 are =')
disp (y1)
subplot (2,1,1)
n=0 : length (y1) -1;
stem (n, y1)
xlabel ('time');
ylabel ('amplitude');
title ('Linear convolution 1);
y2 = conv(h, x);
disp ('the values of y2 are =')
disp (y2)
subplot (2,1,2)
n=0: length(y2) -1;
stem (n, y2)
xlabel ('time');
ylabel ('amplitude');
title ('Linear convolution 2');
if (y1 == y2)
disp ('Linear convolution is commutative')
else
disp ('Linear convolution is not commutative')
end
```

Output:

Enter the input sequence = [1 2 3 4]
Enter the impulse response = [2 1 2 1]
The values of y_1 are = [2 5 10 16 12 11 4]
The values of y_2 are = [2 5 10 16 12 11 4]
linear convolution is Commutative.





Verification:

$x(n) = [1\ 2\ 3\ 4]$

$h(n) = [2\ 1\ 2\ 1]$

$x(n) \longrightarrow$

$h(n)$ ↓	1	2	3	4
2	2	4	6	8
1	1	2	3	4
2	2	4	6	8
1	1	2	3	4

$y_1(n) = \{2, 5, 10, 16, 12, 11, 4\}$

$h(n) \longrightarrow$

$x(n)$ ↓	2	1	2	1
1	2	1	2	1
2	4	2	4	2
3	6	3	6	3
4	8	4	8	4

$y_2(n) = \{2, 5, 10, 16, 12, 11, 4\}$

Name of Experiment ..... associative ..... Date ..... 30-08-18 .....

Experiment No. ..... 9x 2(c) ..... Page No 22

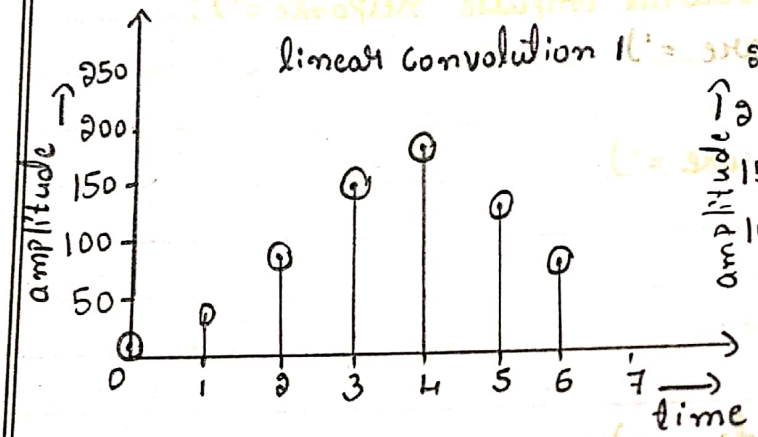
Write a MATLAB code to verify Linear convolution is associative. $[(x(n) * h_1(n)) * h_2(n)] = x_1(n) * [h_1(n) * h_2(n)]$

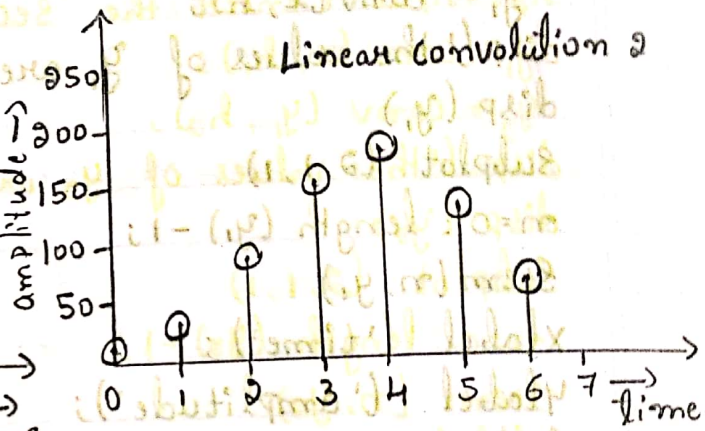
```
clc;
x = input ('Enter the input sequence =');
h1 = input ('Enter the first impulse response =');
h2 = input ('Enter the second impulse response =');
y1 = conv (x, h1);
y2 = conv (y1, h2);
disp ('the values of y2 are =')
disp (y2)
subplot (2,1,1)
n = 0 : length (y2) -1;
stem (n, y2)
xlabel ('time');
ylabel ('amplitude');
title ('Linear convolution 1');
y3 = conv (h1, h2);
y4 = conv (x, y3);
disp ('the values of y4 are=')
disp (y4)
subplot (2,1,2)
n = 0 : length (y4) -1;
stem (n, y4)
xlabel ('time');
ylabel ('amplitude');
title ('Linear convolution 2');
if (y2 == y4)
disp ('Linear convolution is associative')
else
disp ('Linear convolution is not associative')
End.
```

BGSIT

Output :

Enter the input sequence = [1 2 3]
Enter the first impulse response = [2 3 4]
Enter the second impulse response = [3 4 5]
The values of y_2 are = [6 29 86 150 184 133 60]
The values of y_4 are = [6 29 86 150 184 133 60]
linear convolution is associative.





Verification:

$x(n) = [1\ 2\ 3]$
$h_1(n) = [2\ 3\ 4]$
$h_2(n) = [3\ 4\ 5]$

	2	7	16	17	12
3	6	21	48	51	36
4	8	28	64	68	48
5	10	35	80	85	60

$y_2(n) = [6\ 29\ 86\ 150\ 184\ 133\ 60]$

$x(n) \longrightarrow$

$h_1(n)$ ↓	1	2	3
2	2	4	6
3	3	6	9
4	4	8	12

$y_2(n) = [2\ 7\ 16\ 17\ 12]$

Name of Experiment ..... distributive ..... Date : ...30-08-18.....

Experiment No. ..... X 2(d) .....

Write a MATLAB code to verify Linear convolution is distributive $x(n) * [h_1(n) + h_2(n)] = x(n) * h_1(n) + x(n) * h_2(n)$

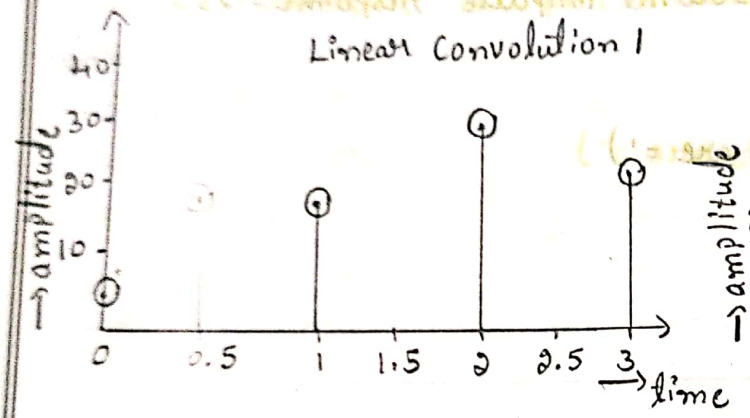
```
clc;
x = input ('Enter the input sequence =');
h1 = input ('Enter the first impulse response =');
h2 = input ('Enter the second impulse response =');
y1 = h1+h2;
y2 = conv (x, y1);
disp ('the values of y2 are =')
disp (y2)
subplot (2,1,1)
n=0: length (y2) -1;
stem (n, y2)
xlabel ('time');
ylabel ('amplitude');
title ('Linear convolution1');
y3 = conv (x, h1);
y4 = conv (x, h2);
y5 = y3+y4;
disp ('the values of y5 are =')
disp (y5)
subplot (2,1,2)
n=0: length (y5) -1;
stem (n, y5)
xlabel ('time');
ylabel ('amplitude');
title ('Linear convolution 2');
if (y2==y5)
disp ('Linear convolution is distributive')
else
disp ('Linear convolution is not distributive')
end
```

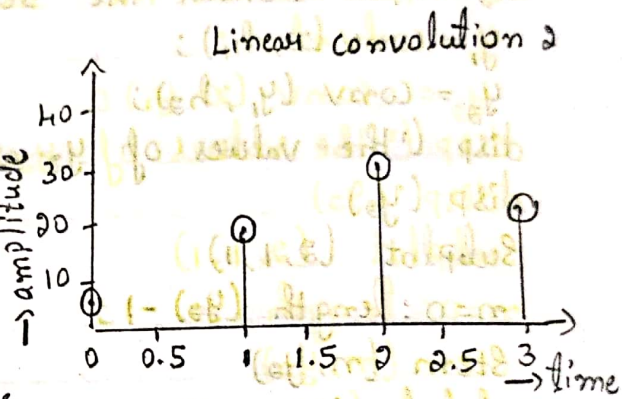
Verified 05/09/18

BGSIT

output :

Enter the input sequence = [1 2 3]
Enter the first impulse response = [3 4]
Enter the second impulse response = [2 3]
The values of y_2 are = [5 17 29 21]
The value of y_5 are = [5 17 29 21]
linear convolution is Distributive





Verification :-

$x(n) = [1 \quad 2 \quad 3]$

$h_1(n) = [3 \quad 4]$

$h_2(n) = [2 \quad 3]$

$h_1 + h_2 = [5 \quad 7]$

	→ x(n)		
	1	2	3
5	5	10	15
7	7	14	21

$y_2(n) = \{5 \quad 17 \quad 29 \quad 21\}$

Name of Experiment ..... Circular Convolution Date : 30-08-18

Experiment No. ..... 2(e) Page No. 24

Write a MATLAB code to perform circular convolution between two sequences.

```
clc;
x = input ('Enter the input sequence =');
h = input ('Enter the impulse response =');
N = max (length (x), length (h));
y = circonv (x, h, N);
disp ('the values of y are =')
disp (y)
n = 0 : length (y) - 1;
stem (n, y)
xlabel ('time');
ylabel ('amplitude');
title ('circular convolution');
```

Keyword:

cconv: Modulo-N circular convolution

Syntax: C = cconv(a, b, n)

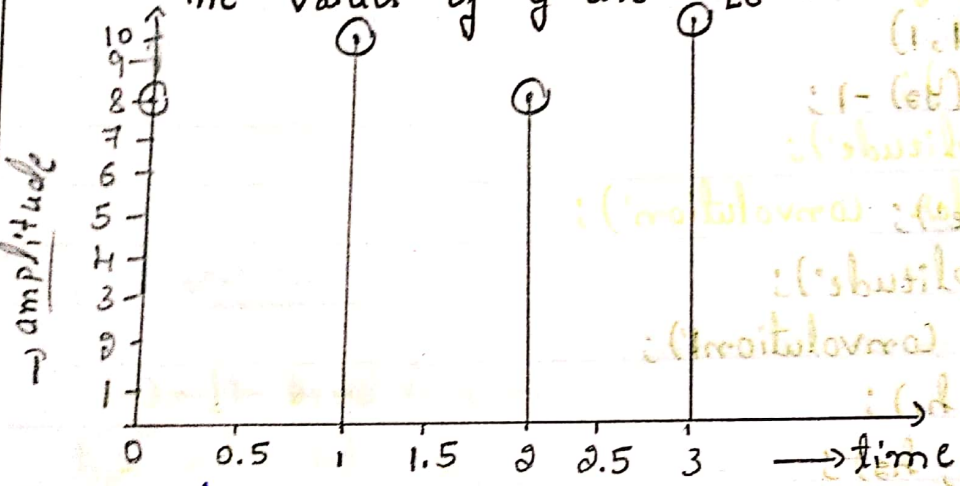
Description: Circular convolution is used to convolve two discrete Fourier transform (DFT) sequences. For very long sequences, circular convolution may be faster than linear convolution.

Output:

Enter the input sequence = [1 2 3]

Enter the impulse response = [1 2 1 2]

The values of y are = [8 10 8 10]



Verification:

$x(n) = \{1, 2, 3\}$

$h(n) = \{1\ 2\ 1\ 2\}$

$y(n) = x(n) \circledast h(n)$

$N = \max(3, 4) = 4$

	1	2	3	0
1	1	2	3	0
2	2	4	6	0
1	1	2	3	0
2	2	4	6	0

$$\begin{bmatrix} 1 & 0 & 3 & 2 \\ 2 & 1 & 0 & 3 \\ 3 & 2 & 1 & 0 \\ 0 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1+0+3+4 \\ 2+2+0+6 \\ 3+4+1+0 \\ 0+6+2+3 \end{bmatrix} = \begin{bmatrix} 8 \\ 10 \\ 8 \\ 10 \end{bmatrix}$$

$y(n) = \{1\ 4\ 7\ 10\ 7\ 6\}$

Name of Experiment ..using commutative..... Date : ....30-08-18.....

Experiment No. .....2(f)..... Page No. 25

Write a MATLAB code to perform circular convolution using commutative.

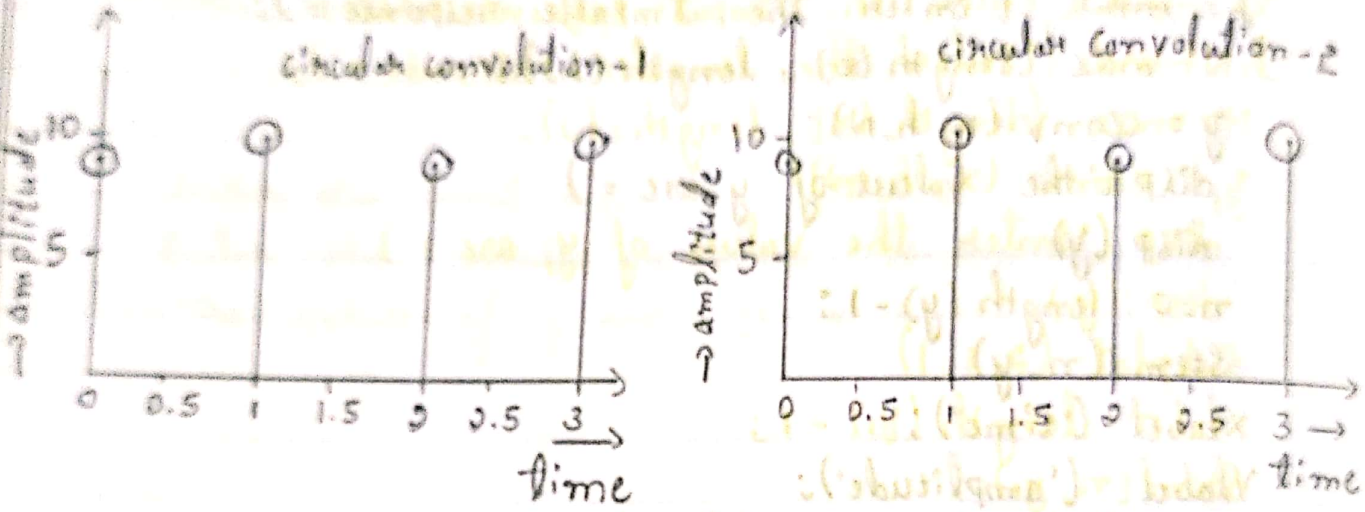
$x(n) \circledast |N^{\nabla} H(n)| = h(n) \circledast |N x(n)|$

```
clc;
x = input ('Enter the input sequence = ');
h = input ('Enter the impulse response = ');
N = max (length (x), length (h));
y1 = cconv (x, h, N)
disp ('Enter the values of y1 are =')
disp (y1)
subplot (2, 1, 1)
n = 0 : length (y1) - 1;
stem (n, y1)
xlabel ('time');
ylabel ('amplitude');
title ('circular convolution 1');
y2 = cconv (h, x, N)
disp ('the values of y2 are =')
disp (y2)
subplot (2, 1, 2)
n = 0 : length (y2) - 1;
stem (n, y2)
xlable ('time');
ylabel ('amplitude');
title ('circular convolution 2')
if (y1 == y2)
disp ('circular convolution is commutative')
else
disp ('circular convolution is not commutative')
End
```

BGSIT

output:

Enter the input sequence = [1 2 3]
enter the impulse response = [1 2 1 2]
the values of y_1 are = [8 10 8 10]
the values of y_2 are = [8 10 8 10]
circular convolution is commutative



verification:

$$y(n) = h(n) \circledast x(n)$$

$$\begin{bmatrix} 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 1+2+1+2 \\ 2+2+6+0 \\ 1+4+3+0 \\ 2+2+6+0 \end{bmatrix}$$

$$= \begin{bmatrix} 8 \\ 10 \\ 8 \\ 10 \end{bmatrix}$$

Name of Experiment ..... associative ..... Date : 30-08-18

Experiment No. ..... 2(g) ..... Page No. 26

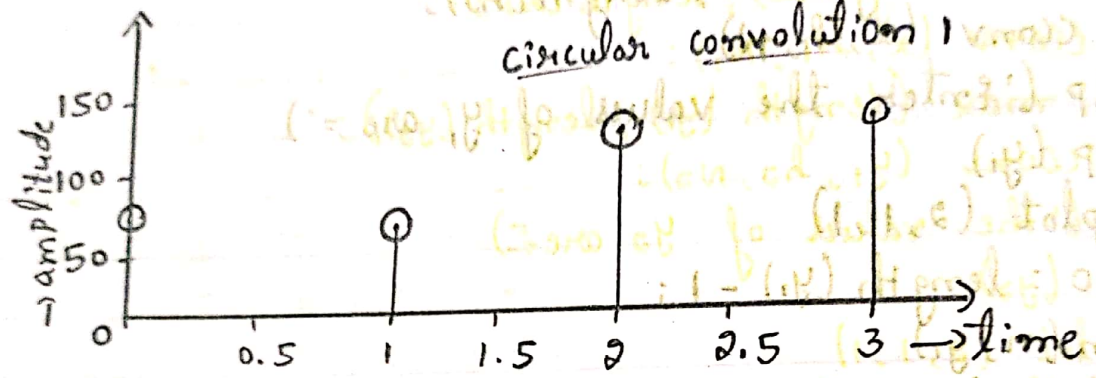
Write a MATLAB code to verify circular convolution is associative $(x(n) \circledast_{N_1} h_1) \circledast_{N_2} h_2 = x \circledast_{N_3} (h_1 \circledast_{N_4} h_2)$

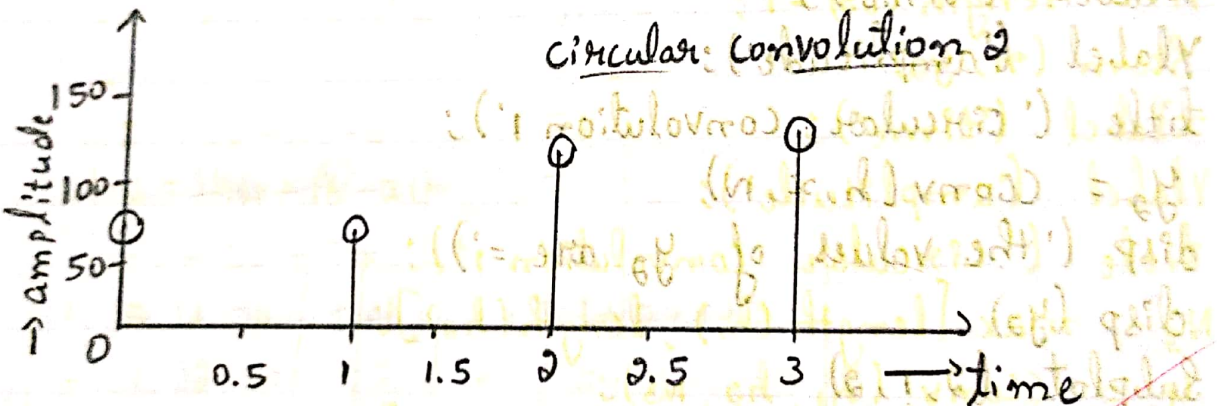
```
clc;
x = input ('Enter the input sequence =');
h1 = input ('Enter the first impulse response =');
h2 = input ('Enter the second impulse response =');
N1 = max [length (x), length (h1)];
y1 = cconv (x, h1, N1);
N2 = max (length (y1), length (h2));
y2 = cconv (y1, h2, N2);
disp ('the values of y2 are:')
disp (y2)
subplot (2, 1, 1)
n = 0: length (y2) - 1;
stem (n, y2)
xlabel ('time');
ylabel ('amplitude');
title ('circular convolution 1');
N3 = max [length (h1), length (h2)];
y3 = cconv (h1, h2, N3);
N4 = max [length (x), length (y3)];
y4 = cconv (x, y3, N4);
disp ('the values are y4 are:')
disp (y4)
subplot (2, 1, 2)
n = 0: length (y4) - 1;
stem (n, y4)
xlabel ('time');
ylabel ('amplitude');
```

BGSIT

output :

```
Enter the input sequence = [1 3 4 1]
Enter the first impulse response = [4 3]
Enter the second impulse response = [3 2 1 0]
The values of y2 are [78 75 112 113]
The values of y4 are [78 75 112 113]
Circular convolution is associative.
```





```
title ('The circular convolution 2');
if (y2==y4)
disp ('circular convolution is associative')
else
disp ('circular convolution is not associative')
end
```

Verification:

$x(n) = \{1, 3, 4, 1\}$ $h_1(n) = \{4, 3\}$ $h_2(n) = \{3, 2, 1, 0\}$

$N_1 = \max[4, 2]$

$N_1 = 4$

$$y_1 = (x \circledast_4 h_1) = \begin{bmatrix} 1 & 1 & 4 & 3 \\ 3 & 1 & 1 & 4 \\ 4 & 3 & 1 & 1 \\ 1 & 4 & 3 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 7 \\ 16 \\ 25 \\ 05 \end{bmatrix}$$

$N_2 = \max(4, 4)$

$N_2 = 4$

$$\begin{bmatrix} 7 & 16 & 25 & 5 \\ 15 & 7 & 16 & 25 \\ 25 & 15 & 7 & 16 \\ 16 & 25 & 15 & 7 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 21+32+25+0 \\ 45+14+16+0 \\ 75+30+7+0 \\ 48+50+15+0 \end{bmatrix} = \begin{bmatrix} 78 \\ 75 \\ 112 \\ 113 \end{bmatrix}$$

Name of Experiment .... distributive ..... Date : 30-08-18

Experiment No. 2(b) Page No. 28

Write a MATLAB code to verify circular convolution is distributive $[x \circledast_{N_1} [h_1 + h_2]] = x \circledast_{N_2} h_1 + x \circledast_{N_3} h_2$

```
clc;
x = input ('Enter the input sequence:');
h1 = input ('Enter the first impulse response =');
h2 = input ('Enter the second impulse response =');
y1 = h1 + h2;
N1 = max (length (x), length (y1));
y2 = cconv (x, y1, N1);
disp ('the values of y2 are =')
disp (y2)
subplot (2,1,1)
n=0 : length (y2) - 1;
stem (n, y2)
xlabel ('time');
ylabel ('amplitude');
title (' circular convolution-1');
N2 = max (length (x), length (h1));
y3 = cconv (x, h1, N2);
N3 = max (length (x), length (h2));
y4 = cconv (x, h2, N3);
y5 = y3 + y4;
disp ('The values of y5 are')
disp (y5)
subplot (2,1,2)
n = 0 : length (y5) - 1;
xlabel ('time');
ylabel ('amplitude');
```

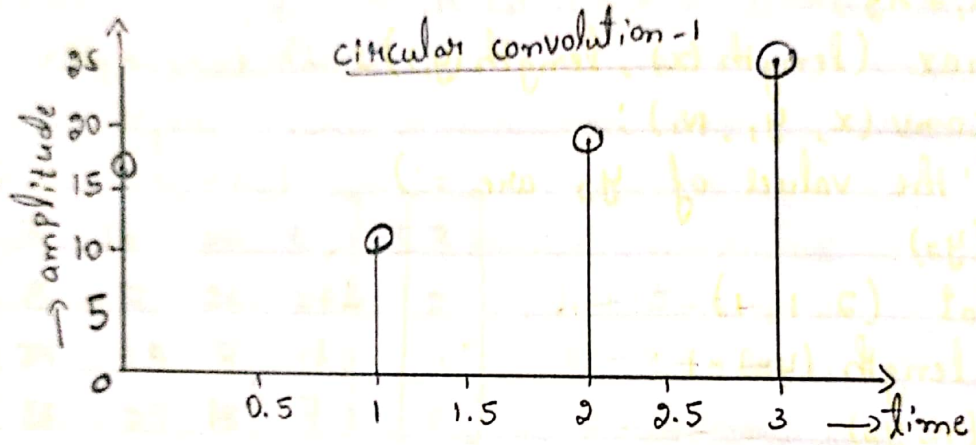
BGSIT

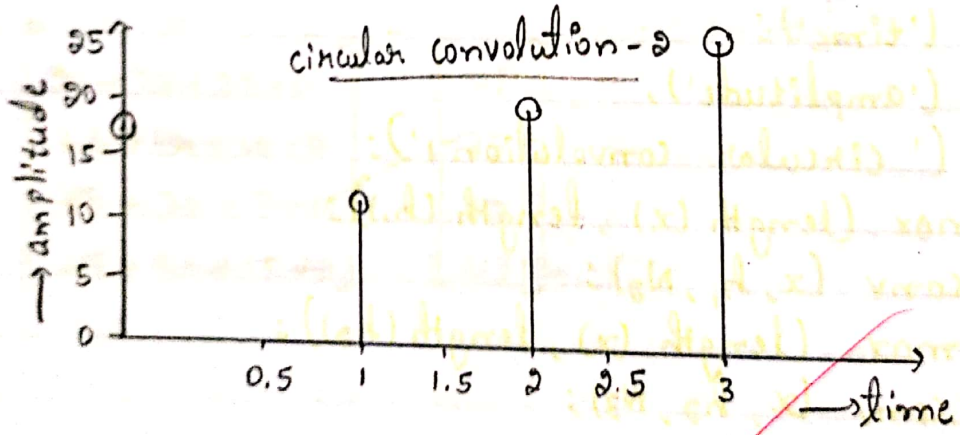
```
title ('circular convolution 2');
if (y2==y5)
disp ('circular convolution is distributive')
Else
disp ('circular convolution is not distributive')
End
```

12/12
Done
20/08/18

output :

Enter the input sequence = [1 2 3 4]
Enter the first impulse response = [1 2]
Enter the second impulse response = [3 1]
The values of y_3 are [16 11 18 25]
The values of y_5 are [16 11 18 25]





Verification:

$x(n) = \{1, 2, 3, 4\}$

$h_1(n) = \{1, 2\}$

$h_2(n) = \{3, 1\}$

LHS $[x \circledast_N |h_1 + h_2|]$

$h_1 + h_2 = \{4, 3\}$

$x \circledast_N |h_1 + h_2|$

$$\begin{bmatrix} 1 & 4 & 3 & 2 \\ 2 & 1 & 4 & 3 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 4+12+0+0 \\ 8+3+0+0 \\ 12+6+0+0 \\ 16+9+0+0 \end{bmatrix}$$

$$= \begin{bmatrix} 16 \\ 11 \\ 18 \\ 25 \end{bmatrix}$$

Name of Experiment Impulse response in diff Eqn Date : 07-09-18

Experiment No. 3) (a) Page No. 30

Write a MATLAB code to find impulse response in a given difference Equation

```
clc;
num = input ('Enter the numerator co-efficients');
den = input ('Enter the denominator co-efficients');
impulse = [ones (1,1), zeros (1, N-1)];
N= input ('Enter the number of samples');
h = filter (num, den, impulse);
disp ('the values of impulse response are')
disp (h)
subplot (2,1,1)
n=0 : length (impulse)-1;
stem (n, impulse)
xlable ('time');
Ylabel ('amplitude');
title ('Impulse funition');
subplot (2,1,2)
n=0 : length (h)-1;
stem (n, h)
xlabel ('time');
Ylabel ('amplitude');
title ('impulse response');
```

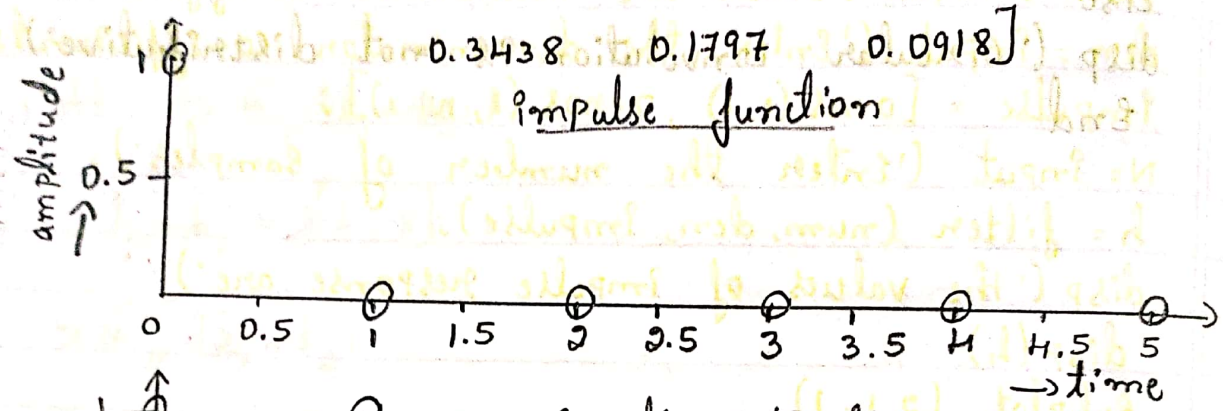
Output:

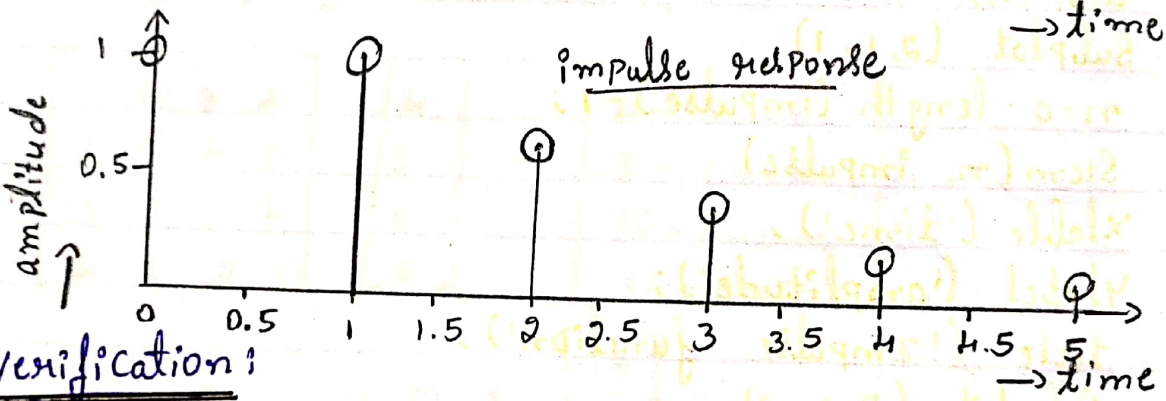
Enter the numerator coefficients = [1 1/4]

Enter the denominator co-efficients = [1, -3/4, 1/8]

Enter the number of samples = 6

The impulse response values are [1.0000 1.0000 0.6250 0.3438 0.1797 0.0918]





Verification:

$$y(n) - \frac{3}{4}y(n-1) + \frac{1}{8}y(n-2) = x(n) + \frac{1}{4}x(n-1)$$

$$y(n) = x(n) + \frac{1}{4}x(n-1) + \frac{3}{4}y(n-1) - \frac{1}{8}y(n-2)$$

Let $x(n) = \delta(n)$

$$y(n) = \delta(n) + \frac{1}{4}\delta(n-1) + \frac{3}{4}y(n-1) - \frac{1}{8}y(n-2)$$

Let $N=6$, $n=0,1,2,3,4,5$ & Assume initial values are zero

$N=0$ $\quad y(0) = \delta(0) + \frac{1}{4}\delta(0-1) + \frac{3}{4}y(0-1) - \frac{1}{8}y(0-2)$

$y(0) = 1$

$N=1$ $\quad y(1) = \delta(1) + \frac{1}{4}\delta(1-1) + \frac{3}{4}y(1-1) - \frac{1}{8}y(1-2)$

$y(1) = \frac{1}{4} + \frac{3}{4} = 1$

$N=2 \quad y(2) = \delta(2) + \frac{1}{4}\delta(2-1) + \frac{3}{4}y(2-1) + \frac{-1}{8}y(3-2)$

$= \frac{3}{4}(1) - \frac{1}{8} \quad = 5/8 = 0.625$

$N=3 \quad y(3) = \delta(3) + \frac{1}{4}\delta(3-1) + \frac{3}{4}y(3-1) - \frac{1}{8}y(3-2)$

$= 3/4 * 0.625 - \frac{1}{8} \times 1 = 0.3437$

$N=4 \quad y(4) = \delta(4) + 1/4\,\delta(4-1) + \frac{3}{4}y(4-1) - \frac{1}{8}y(4-2)$

$= 3/4 * 0.347 - \frac{1}{8} * 0.625 = 0.17965.$

$N=5, y(5) = \delta(5) + \frac{1}{4}\delta(5-1) + \frac{3}{4}y(5-1) - \frac{1}{8}y(5-2)$

$y(5) = \frac{3}{4} \times 0.17965 - \frac{1}{8} \times 0.3437 = 0.09177$

Apply z-transform

$Y(z) - \frac{3}{4}z^{-1}Y(z) + \frac{1}{8}z^{-2}Y(z) = X(z) + \frac{1}{4}z^{-1}X(z)$

$Y(z)\left[1 - \frac{3}{4}z^{-1} + \frac{1}{8}z^{-2}\right] = X(z)\left[1 + \frac{1}{4}z^{-1}\right]$

$H(z) = \frac{Y(z)}{X(z)} = \frac{1 + \frac{1}{4}z^{-1}}{1 - \frac{3}{4}z^{-1} + \frac{1}{8}z^{-2}} = \frac{[1 \quad 1/4]}{[1 \quad -3/4 \quad 1/8]}$

Name of Experiment: Step response in a diff eqn Date: 07-09-18

Experiment No.: 3) b Page No. 32

Write a MATLAB code to find a step response in a given difference equation

```
clc;
num = input ('Enter the numerator co-efficients');
den = input ('Enter the denominator co-efficients');
Step = ones(1,N);
y = filter(num, den, Step);
disp ('the step response values are')
disp(y)
subplot (2,1,1)
n=0: length (Step) -1;
stem(n, step)
xlable ('time');
ylabel ('amplitude');
title ('Step function');
subplot (2,1,2)
n=0: length (y) -1;
stem (n, y)
xlabel ('time');
ylabel ('amplitude');
title ('Step response');
```

Keyword :

Filter : 1-D digital filter

Syntax : y = filter (b, a, x)

Description : The filter function filters a data sequence using a digital filter which works for both real and complex inputs. The filter is a direct form II. transposed implementation of the standard difference equation.

y = filter (b, a, x) filters the data in vector x with the filter described by numerator co-efficient b and denominator coefficient vector a. If a(1) is not equal to 1, filter normalizes the filter co-efficients by a(1). If a(1) equals 0, filter returns an Error.

Ex : data = [1 : 0.2 : 4] ;

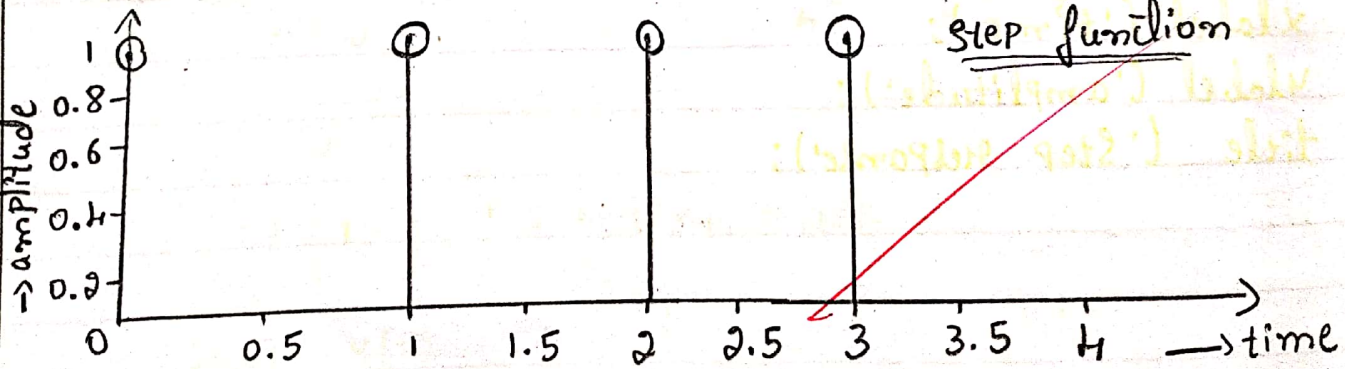
output :

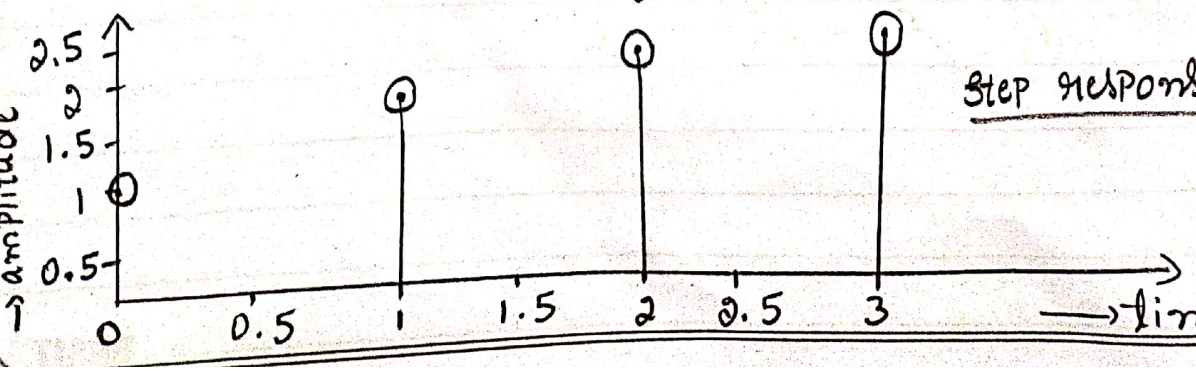
Enter the numerator co-efficients = [1]

Enter the denominator co-efficients = [1 -3/4 1/8]

The step response values are

1.0000 1.7500 2.1875 2.4219.





Verification:

$$y(n) - \frac{3}{4} y(n-1) + \frac{1}{8} y(n-2) = x(n)$$

$$y(n) = x(n) + \frac{3}{4} y(n-1) - \frac{1}{8} y(n-2)$$

Let $N = 4$, $n = 0, 1, 2, 3$.

$x(n) = u(n)$ & Assume initial values are zero

$$y(n) = \frac{3}{4} y(n-1) - \frac{1}{8} y(n-2) + u(n)$$

$n=0$, $y(0) = \frac{3}{4} y(0-1) - \frac{1}{8} y(0-2) + u(0) = 0 - 0 + 1 = 1$

$n=1$, $y(1) = \frac{3}{4} y(1-1) - \frac{1}{8} y(1-2) + u(1) = \frac{3}{4}(1) + 1 = \frac{7}{4} = 1.75$

$n=2$, $y(2) = \frac{3}{4} y(2-1) - \frac{1}{8} y(2-2) + u(2) = \frac{3}{4}(1.75) - \frac{1}{8}(1) + 1 = 2.187$

$n=3$, $y(3) = \frac{3}{4} y(3-1) - \frac{1}{8} y(3-2) + u(3) = \frac{3}{4}(2.187) - \frac{1}{8} \times 1.75 + 1 = 2.421$

$$y(n) = \{1, 1.75, 2.187, 2.421\}$$

Apply z-transform

$$Y(z) - \frac{3}{4} z^{-1} Y(z) + \frac{1}{8} z^{-2} Y(z) = X(z)$$

$$Y(z) \left[1 - \frac{3}{4} z^{-1} + \frac{1}{8} z^{-2}\right] = X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - \frac{3}{4} z^{-1} + \frac{1}{8} z^{-2}}$$

$$= \frac{[1]}{[1 \quad -3/4 \quad 1/8]}$$

Keywords:

FFT : (Fast fourier tranform)

Syntax : Y= fft (x,n)

Description : Y= fft(x,n) returns the n-point DFT. fft(x) is equivalent to fft (x,n) where n is the size of x in the first nonsingleton dimension. If the length of x is less than n, x is padded with trailing zeros to length n. If the length of x is greater than n, the sequence x is truncated. When x is a matrix, the length of the columns are adjusted in the same manner.

abs : Absolute value and complex magnitude

Syntax : abs(x)

Description : abs(x) returns an array Y such that each Element of Y is the absolute value of the corresponding Element of x. If x is complex, abs(x) returns the complex modulation (magnitude) which is the same as

Sqrt (real(x) .^ 2 + imag(x).^2)

angle : phase angle

Syntax : P = angle (z)

Description : P = angle (z) returns the phase angles, in radians, for each element of complex array z. The angles lie between $\pm\pi$.

For example complex z, the magnitude R and phase theta is given by

R = abs(z)

theta = angle(z) and the statement z = R .* exp(i * theta) converts back to the original complex z.

ifft : Inverse fast fourier transform

Syntax : ifft (x, n)

Description : Y = ifft (x, n) returns the n-Point inverse DFT of vector X.

Name of Experiment DFT using built-in function Date : 20-09-18

Experiment No. 4.a) Page No. 35

Write a MATLAB code to find DFT for a given sequence and also plot magnitude and phase using built-in-functions.

```
clc;
x = input ("enter the input sequence = ");
N = input ('enter the number of DFT points = ');
x = fft (x, N);
disp ('The values of x are =')
disp (x)
M = abs(x)
disp ('The magnitude values are =')
disp (M)
subplot (2, 1, 1)
n=0 : length (M)-1;
stem (n, M)
xlabel ('time');
ylabel ('amplitude');
title ('Magnitude');
P= angle (X);
disp ('the phase values are')
disp (P)
subplot (2, 1, 2)
n=0 : length (P)-1;
stem(n, P)
xlabel ('time');
ylabel ('amplitude');
title ('phase');
```

BGSIT

output :

enter the input sequence = [1 2 3 4]

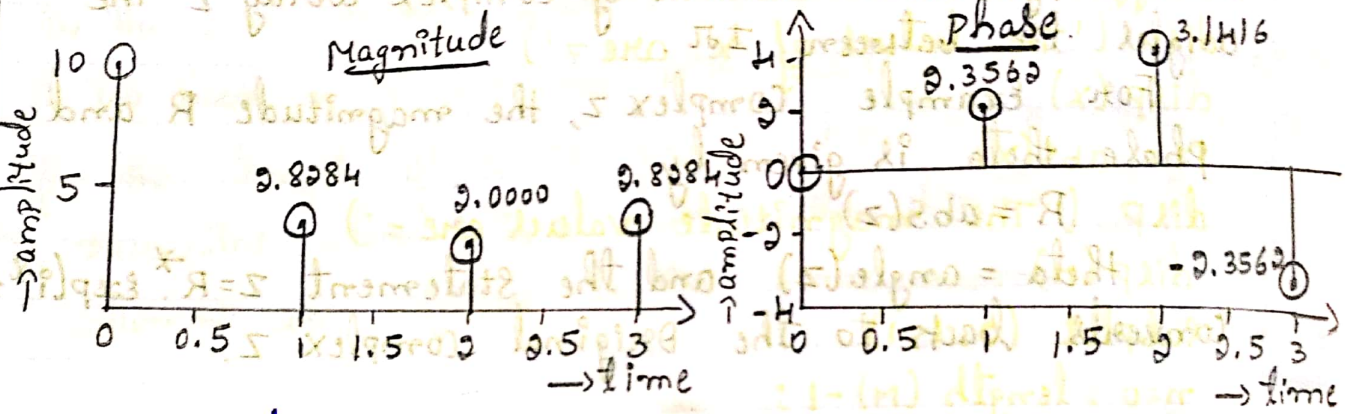
enter the number of dft points = 4.

The values of x are = 10.0000 + 0.0000i -2.0000 +2.0000i -2.0000 +0.0000i -2.0000-2.0000i

The magnitude value are =

10.0000 2.8284 2.0000 2.8284

The phase values are 0 2.3562 3.1416 -2.3562.

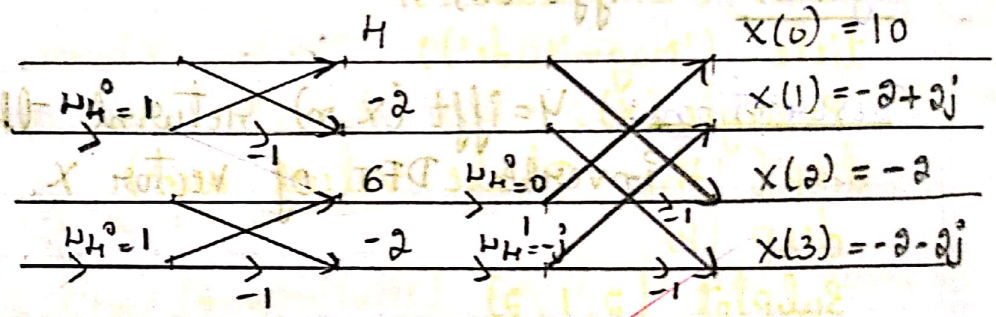


Verification :

$x(n) = \{1, 2, 3, 4\}$

B.R

00	00	x(0) = 1
01	10	x(2) = 3
10	01	x(1) = 2
11	11	x(3) = 4



$X(K) = \{10, -2+2j, -2, -2-2j\}$

$|X(K)| = \{10, 2.8284, 2, 2.8284\}$

$\angle X(K) = \tan^{-1}(b/a) = \{0, 2.3562, 3.1416, -2.3562\}$

Name of Experiment: IDFT using built-in function Date: 20-09-18

Experiment No.: 4 b)

Write a MAT-LAB code to find IDFT for the given DFT Signal using built-in-function.

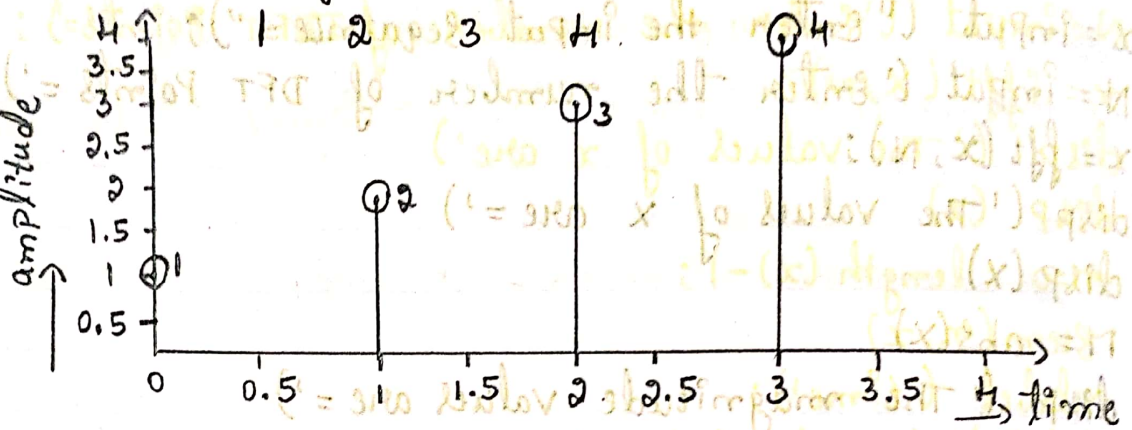
```
clc;
x=input ('Enter the DFT Sequence:');
N=input ('Enter the Number of IDFT Points=');
x=ifft (X,N);
disp ('The values of x are')
disp (x)
n=0 : length (x)-1;
stem(n,x)
xlabel ('time');
ylabel ('amplitude');
title ('IDFT Points');
```

output :

Enter the DFT sequence = [10 -2+2j -2 -2-2j]

Enter the number of idft points = 4.

The values of x are =



Verification :

$x(k) = \{10, -2+j2, -2, -2-j2\}$

B.R

00	00	x(0) = 10	8 → 4 → 1/4 → x(0) = 1
01	10	x(2) = -2	12 → 8 → 1/4 → x(1) = 2
10	01	x(1) = -2+j2	-4 → 12 → 1/4 → x(2) = 3
11	11	x(3) = -2-j2	4j → 16 → 1/4 → x(3) = 4

$\therefore x(n) = \{1, 2, 3, 4\}$

Name of Experiment: DFT using DFT Equation Date: 28-09-18

Experiment No.: 4 c) Page No. 37

Write a MATLAB code to find DFT for a given sequence & also find magnitude & phase using DFT Eqn.

```
clc;
x=input ('Enter the input sequence=');
N=input ('Enter the number of DFT Points=');
X=zeros (1,N);
for K=0:N-1;
for n=0:N-1;
X(K+1)=X(K+1)+x(n+1)*Exp((-i)*2*Pi*K*n/N);
End
End
disp ('the values of x are')
disp (X)
m=abs (X)
disp ('the magnitude values are')
disp (m)
Subplot (2,1,1)
n=0:length (m)-1;
Stem (m,m)
Xlabel ('time');
Ylabel ('amplitude');
title ('magnitude');
P=angle (X);
disp ('the phase values are')
disp (P)
Subplot (2,1,2)
n=0:length (P)-1;
Stem (n,P)
xlabel ('time');
Ylabel ('amplitude');
title ('Phase');
```

BGSIT

output :

Enter the input sequence = [1 2 1 0]

Enter the number of DFT Points = 4.

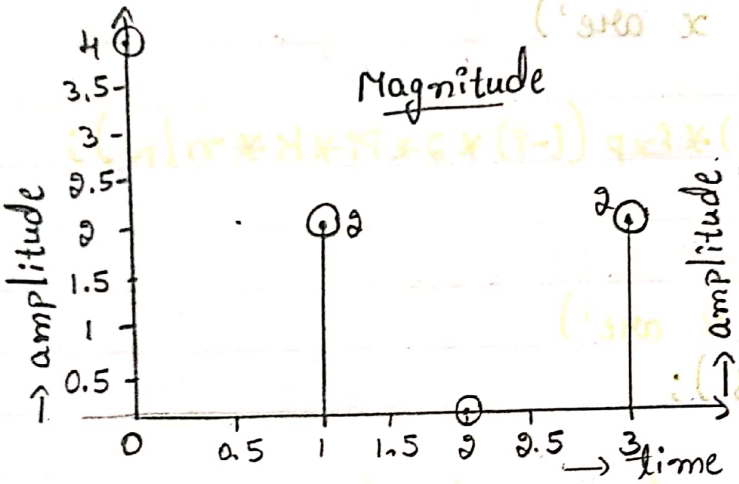
The values of X are

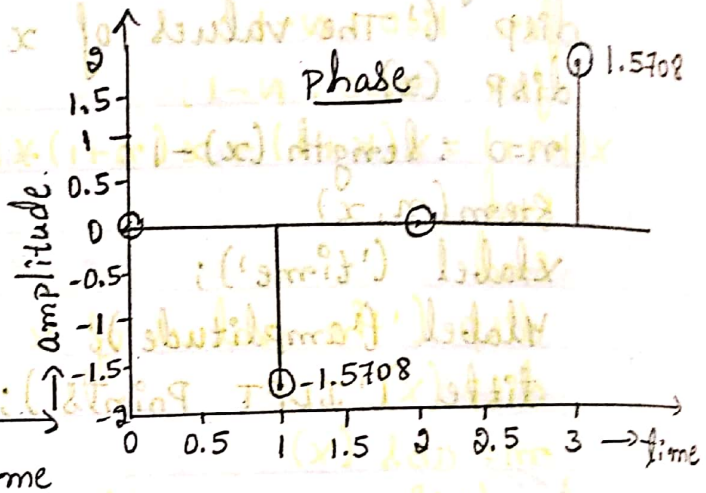
4.0000+0.0000i, 0.0000-2.0000i, 0.0000+0.0000i, -0.0000+2.0000i.

m = 4.0000 2.0000 0 2.0000

The magnitude values are 4.0000 2.0000 0 2.0000

The phase values are 0 -1.5708 0 1.5708





Verification using Matrix Method :

$x(n) = \{1, 2, 1, 0\}$

$N = 4 \quad X_N = W_N x_N$

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^4 & W_4^6 \\ W_4^0 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & 1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ -2j \\ 0 \\ 2j \end{bmatrix}$$

Magnitude = {4, 2, 0, 2}

Phase = {0, -1.5707, 0, 1.5707}

Name of Experiment: IDFT using IDFT Equation Date: 28-09-18

Experiment No: 4 d) Page No: 38

Write a MATLAB code to find IDFT for a given sequence using IDFT Equation.

```
clc;
x = input ('Enter the dft sequence =');
N = input ('Enter the Number of idft Points =');
x = zeros (1, N);
for n=0 : N-1;
for K=0 : N-1;
x(n+1) = x(n+1) + X(K+1) * Exp((i) * 2 * Pi * K * n / N);
xn = x. /N;
End
End
disp ('The values of xn are');
disp (xn)
n=0 : length (xn) - 1;
Stem (n, xn)
xlabel ('time');
ylabel ('amplitude');
title ('idft sequence');
```

12/12 28/9/18

BGSIT

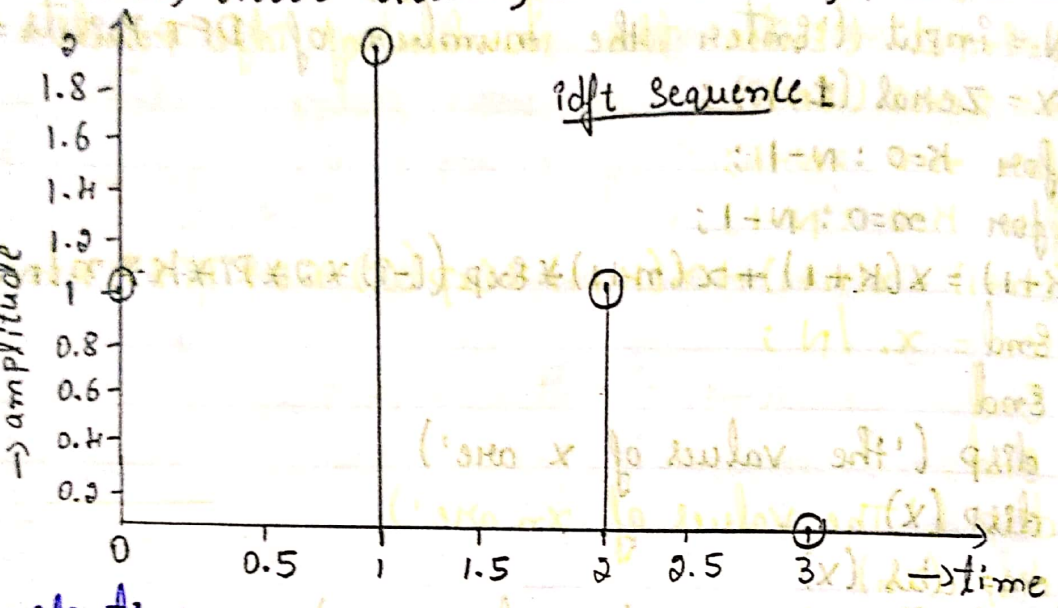
Output :

Enter the input sequence = [4 -2j 0 2j]

Enter the number of idft points = 4

The value of x_n are

1.0000 + 0.0000i, 2.0000 - 0.0000i, 1.0000 + 0.0000i, 0.0000 + 0.0000i



Verification :

$X(K) = \{4, -j2, 0, j2\}$

$x_N = \frac{1}{N} W_N^* X_N$.

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \frac{1}{4} \begin{bmatrix} W_4^{-0} & W_4^{-0} & W_4^{-0} & W_4^{-0} \\ W_4^{-0} & W_4^{-1} & W_4^{-2} & W_4^{-3} \\ W_4^{-0} & W_4^{-2} & W_4^{-4} & W_4^{-6} \\ W_4^{-0} & W_4^{-3} & W_4^{-6} & W_4^{-9} \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix}$$

$$= \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & +j \end{bmatrix} \begin{bmatrix} 4 \\ -2j \\ 0 \\ 2j \end{bmatrix} \Rightarrow \frac{1}{4} \begin{bmatrix} 4 - 2j + 0 + 2j \\ 4 + 2 + 0 + 2 \\ 4 + 2j + 0 - 2j \\ 4 - 2 + 0 + 2 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 4 \\ 8 \\ 4 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

Keywords :

1) Buttord : Butterworth filter order and cut-off frequency

Syntax : [n, Wn] = buttord (Wp, Ws, Rp, Rs)

[n, Wn] = buttord (Wp, Ws, Rp, Rs, 's')

Description : buttord calculates the minimum order of a digital or analog Butterworth filter required to meet a set of filter design specifications.

2) Butter : Butterworth filter design.

Syntax : [Z, P, K] = butter (n, Wn)

Description : butter designs lowpass, bandpass, highpass and bandstop digital and analog Butterworth filters. Butterworth filters are characterized by a magnitude response that is maximally flat in the passband and monotomic overall.

3) freqs : frequency response of analog filters.

Syntax : h = freqs (b, a, w)

Description : freqs (b, a, w) returns the complex frequency response of the analog filter specified by co-efficient vectors b and a. freqs Evaluates the frequency response along the imaginary axis in the Complex plane at the angular frequencies in rad/s specified in real vector w, where w is a vector containing more than one frequency.

4) **freqz** : Frequency response of filter
Syntax : [h, w] = freqz (hfilt)
Description : [h, w] = freqz (hfilt) returns the frequency response h and the corresponding frequencies w at which the filter response of hfilt is computed. The frequency response is evaluated at 8192 points equally spaced around the upper half of the unit circle.

5) **cheb1ord** : chebyshev Type 1 filter order.
Syntax : [n, Wp] = cheb1ord (Wp, Ws, Rp, Rs)
[n, Wp] = cheb1ord (Wp, Ws, Rp, Rs, 's')
Description : cheb1ord calculates the minimum order of a digital or analog Chebyshev Type 1 filter required to meet a set of filter design specifications.

6) **cheb1** : chebyshev Type 1 filter using specification object.
Syntax : hd = design (d, 'cheby1')
Description : hd = design (d, 'cheby1') designs a type 1 chebyshev IIR digital filter using the specifications supplied in the object d. Depending on the filter specification object, cheby1 may or may not be a valid design method. Use design methods with the filter specification object to determine if a chebyshev type 1 filter design is possible.

BGSIT

Name of Experiment .... Butterworth LPF ..... Date : 28-09-18

Experiment No. ..... 5(a) ..... Page No. 40

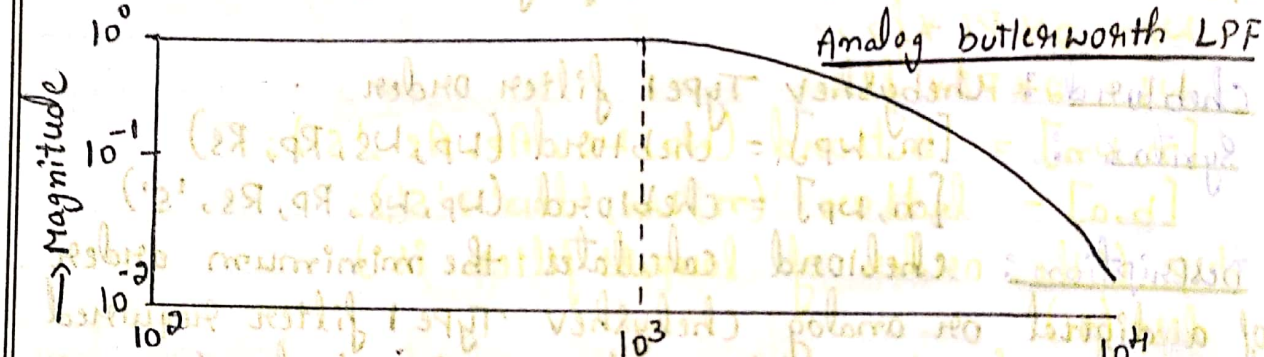
Write a MATLAB code to design analog butterworth lowpass filter.

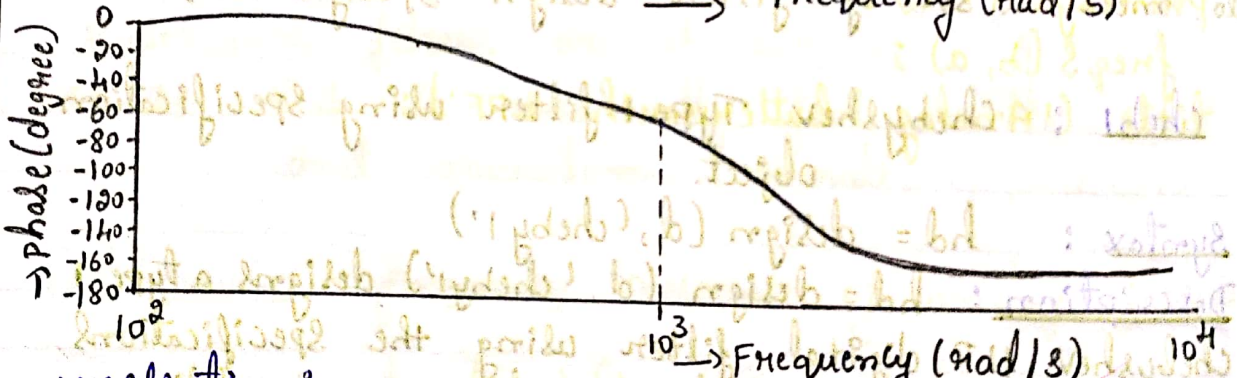
```
clc;
fp=input('enter the passband edge frequency in Hz');
fs=input('enter the stopband edge frequency in Hz');
Ap=input('enter the Passband attenuation in dB=');
As=input('enter the Stopband attenuation in db=');
  wp= 2*Pi*fp;
  ws = 2*Pi*fs;
  [n, wn] = buttord (wp, ws, Ap, As, 's');
  [b,a] = butter (n, wn, 'low', 's');
disp ('the order of lowpass filter is')
 disp (n)
printsys (b,a);
  freqs (b, a);
title ('Analog butterworth LPF');
```

Output :

Enter the Passband Edge frequency in hz = 250
Enter the Stopband Edge frequency in hz = 750
Enter the Passband attenuation in db = 3
Enter the Stopband attenuation in db = 15
The order of Low pass filter is = 2.

$$\text{num/den} = \frac{4012915.1817}{s^2 + 2832.9897\,s + 4012915.1817}$$





Verification :-

$A_p = -3$ dB $A_s = -15$ dB $f_p = 250$ Hz $f_s = 750$ Hz.

$$N = \frac{\log\left(\frac{10^{-A_p/10} - 1}{10^{-A_s/10} - 1}\right)}{2\log\left(\frac{\Omega_p}{\Omega_s}\right)}$$

$\Omega_p = 2\pi f_p = 2\pi \times 250 = 500\pi$ rad/sec

$\Omega_s = 2\pi f_s = 2\pi \times 750 = 1500\pi$ rad/sec.

$$N = \frac{\log\left(\frac{10^{-(-3/10)} - 1}{10^{-(-15/10)} - 1}\right)}{2\log\left(\frac{500\pi}{1500\pi}\right)} \Rightarrow N = 1.5657$$

$\boxed{N = 2}$ //

Name of Experiment .... Butterworth HPF Date : 28-09-18

Experiment No. 5 (b) Page No. 41

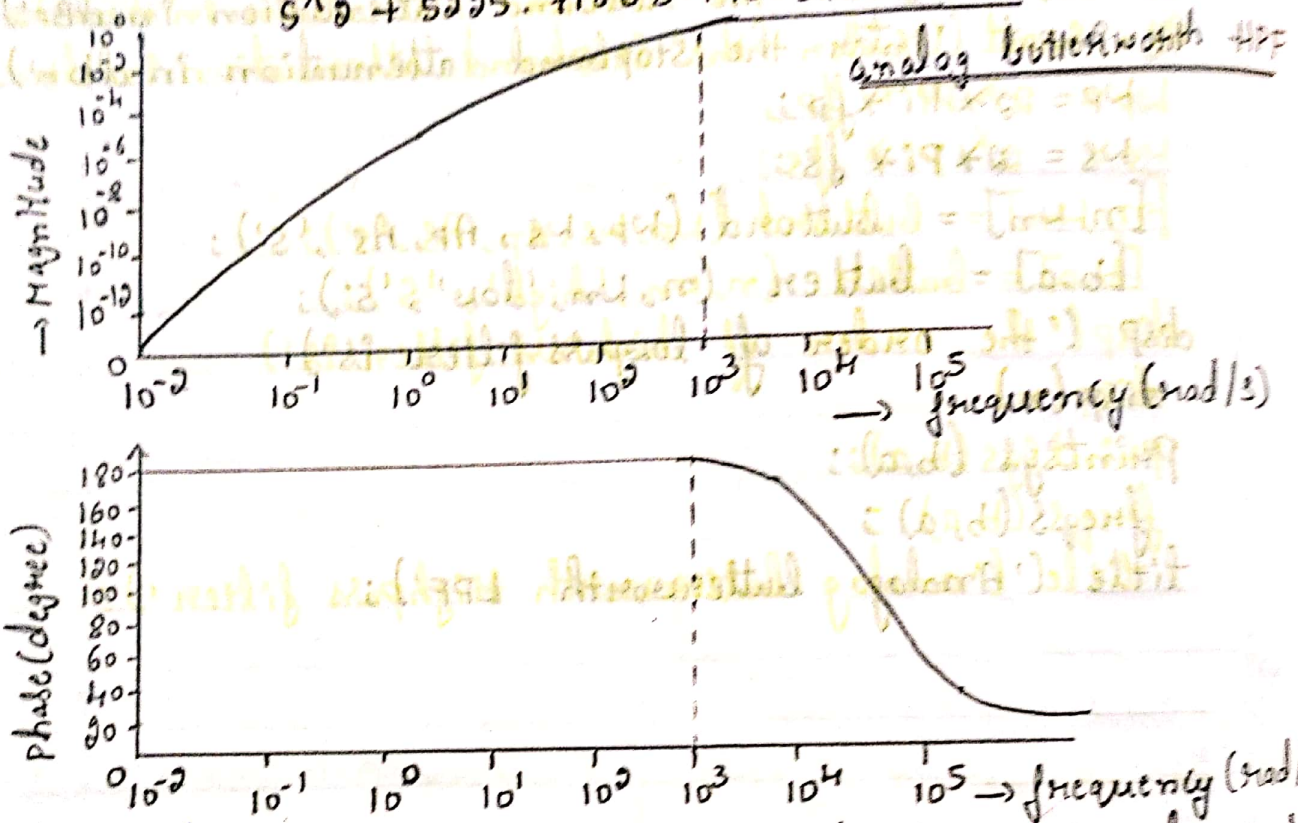
Write a MATLAB code to design analog Butterworth Highpass filter.

```
clc;
fp=input ('Enter the Passband edge frequency in hz=');
fs=input ('Enter the Stopband edge frequency in hz=');
Ap = input ('Enter the Passband attenuation in db=');
As = input ('Enter the Stopband attenuation in db=');
Wp = 2* pi* fp;
Ws = 2* pi* fs;
[n, Wn] = buttord (Wp, Ws, Ap, As, 's');
[b,a] = butter (n, Wn, 'high', 's');
disp('the order of highpass filter is')
disp(n)
printSys(b,a);
freqs (b,a) ;
title ('analog butterworth Highpass filter');
```

output :

Enter the passband edge frequency in Hz = 750
Enter the stopband edge frequency in Hz = 250
Enter the passband edge attenuation in db = 3
Enter the stopband attenuation in db = 15
The order of Highpass filter is 2.

num/den = $\frac{s^2}{s^2 + 5225.7186\,s + 13654067.2368}$



Verification :- $A_p = -3$ dB $A_s = -15$ dB $f_p = 750$ Hz $f_s = 250$ Hz.

$$N = \frac{\log\left(\frac{10^{-A_p/10} - 1}{10^{-A_s/10} - 1}\right)}{2\log\left(\frac{\omega_p}{\omega_s}\right)}$$

$\omega_p = 2\pi f_p = 2\pi \times 750 = 1500\pi$ rad/sec

$\omega_s = 2\pi f_s = 2\pi \times 250 = 500\pi$ rad/sec.

$$N = \frac{\log\left(\frac{10^{-(-3)/10} - 1}{10^{-(-15)/10} - 1}\right)}{2\log\left(\frac{1500\pi}{500\pi}\right)}$$

$N = 1.559$

$\boxed{N = 2}$ //

Name of Experiment ..... Chebyshev LPF ..... Date : 4-10-18

Experiment No. ..... 5 (c) ..... Page No. 42

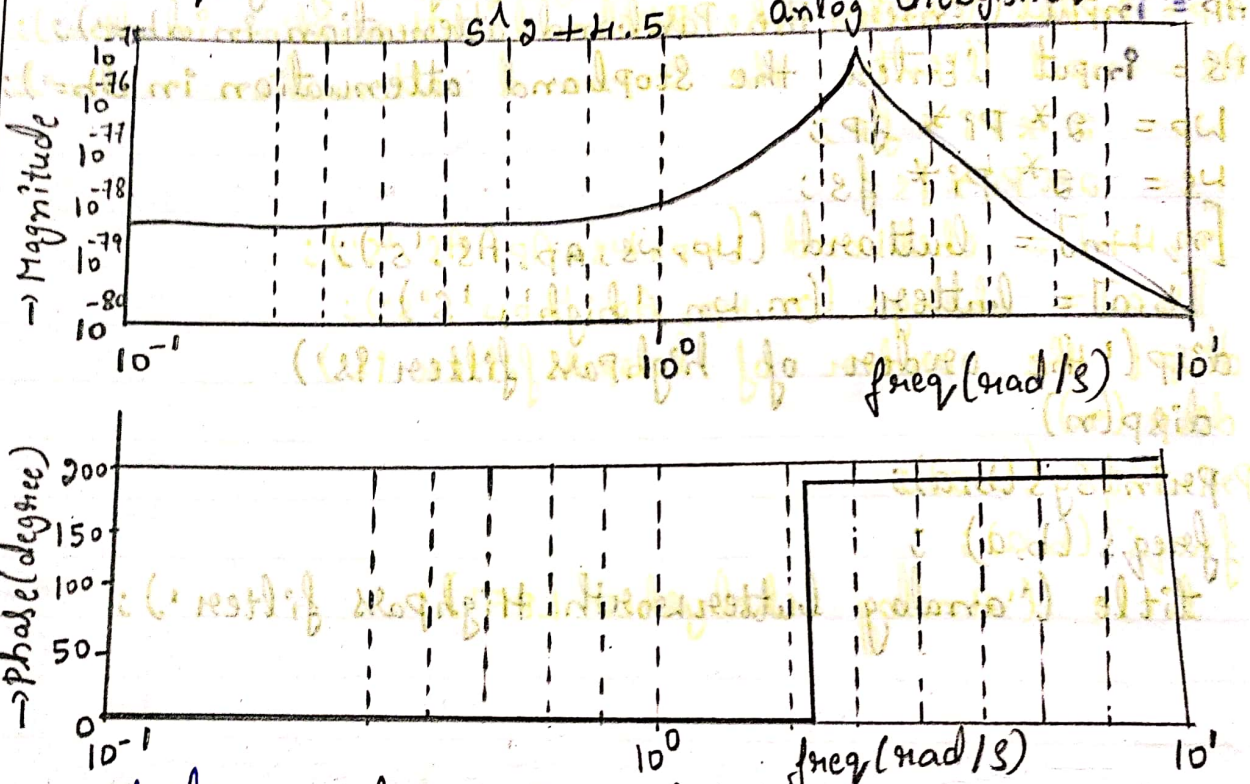
Write a MATLAB code to design Analog chebyshev Lowpass filter.

```
clc;
fp=input ('Enter the Passband Edge frequency in Hz=');
fs=input('Enter the Stopband Edge frequency in Hz =');
Ap= input ('Enter the Passband attenuation in db=');
As= input ('Enter the Stopband attenuation in db=');
wp= 2*Pi*fp;
ws= 2*Pi*fs;
[n, wn] = cheb1ord [wp, ws, Ap, As, 's');
[b,a]= cheby1 (n, wp, Ap, 'low', 's');
disp('the order of lowpassfilter is')
disp (n)
Printsys (b,a);
freqs (b,a);
title ('Analog chebyshev LPF');
```

output :

```
Enter the Passband edge frequency in Hz = 250
Enter the Stopband edge frequency in Hz = 750
Enter the Passband attenuation in db = 3
Enter the Stopband attenuation in db = 15
The order of LPF is 2.
```

num/den = $\frac{1.998 \text{ e-}078}{s^2 + 1.5}$



Verification: $f_p = 250$ Hz $f_s = 750$ Hz $A_p = -3$ $A_s = -15$

$$N = \frac{\cosh^{-1}\sqrt{\left(\frac{10^{-A_s/10}-1}{10^{-A_p/10}-1}\right)}}{\cosh^{-1}\left(\Omega_s/\Omega_p\right)}$$

$\Omega_p = 2\pi f_p = 2\pi \times 250 = 500\pi$ rad/sec

$\Omega_s = 2\pi f_s = 2\pi \times 750 = 1500\pi$ rad/sec

$$N = \frac{\cosh^{-1}\sqrt{\left(\frac{10^{-(-15)/10}-1}{10^{-(-3)/10}-1}\right)}}{\cosh^{-1}\left(1500\pi/500\pi\right)} \Rightarrow N = 1.360$$

select N = 2

Write a MATLAB code to design Analog chebyshev Highpass filter.

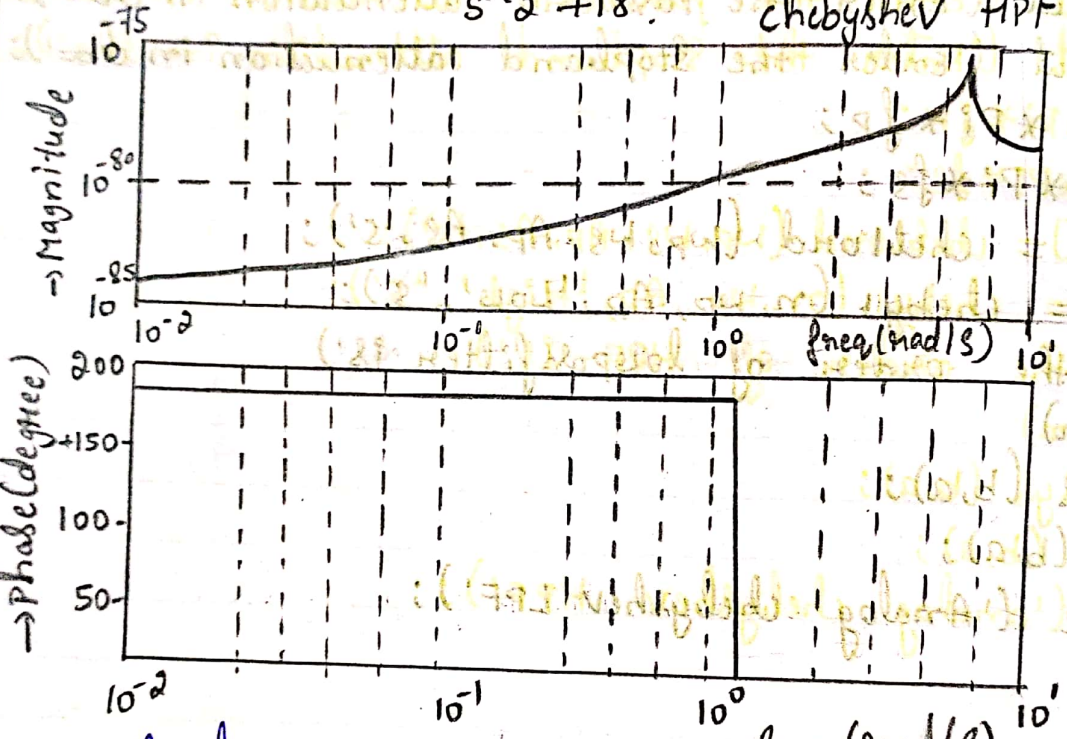
```
clc;
fp=input ('Enter the Passband frequency in Hz=');
fs=input ('Enter the Stopband frequency in Hz=');
Ap=input ('Enter the passband attenuation in db=');
As=input ('Enter the Stopband attenuation in db=');
Wp = 2*Pi*fp;
Ws = 2*Pi*fs;
[n,Wn] = cheb1ord (Wp,Ws, Ap, As,'s');
[b,a] = cheby1 (n, Wp, Ap, 'High','s');
disp ('the order of HPF is')
disp (n)
printsys (b,a);
freqs (b,a);
title ('analog chebyshev HPF');
```

12/12 4/10/18

Output :

Enter the Passband edge frequency in Hz = 750
Enter the stop band edge frequency in Hz = 250
Enter the passband attenuation in db = 3
Enter the stopband attenuation in db = 15.
The order of HPF is 2.

num/den = $\frac{2.8853e^{-0.79}\ s^2}{s^2 + 18}$



Verification:- $f_p = 750$ Hz $f_s = 250$ Hz $A_p = -3$ $A_s = -15$

$$N = \frac{\cosh^{-1}\sqrt{\frac{10^{-A_s/10}-1}{10^{-A_p/10}-1}}}{\cosh^{-1}(w_s/w_p)}$$

$w_p = 2\pi f_p = 2\pi \times 750 = 1500\pi$

$w_s = 2\pi f_s = 2\pi \times 250 = 500\pi$

$$N = \frac{\cosh^{-1}\sqrt{\frac{10^{-(15)/10}-1}{10^{-(3)/10}-1}}}{\cosh^{-1}(500\pi/1500\pi)}$$

$N =$

$N =$

Name of Experiment: Digital Butterworth LPF using bilinear transformation Date: 11-10-18

Experiment No: 6(a) Page No: 44

Write MATLAB code to design digital butterworth Lowpass filter using bilinear transformation

```
clc;
fs = input ('Enter the sampling frequency in Hz=');
Fp = input ('Enter the Passband Edge frequency in Hz=');
Fs = input ('Enter the Stopband Edge frequency in Hz=');
Ap = input ('Enter the passband Attenuation in db=');
As = input ('Enter the Stopband Attenuation in db=');
Fp1 = 2*pi*(Fp/fs);
Fs1 = 2*pi*(Fs/fs);
Wp = 2*tan(Fp1/2);
Ws = 2*tan(Fs1/2);
[n, Wn] = buttord (Wp, Ws, Ap, As, 's');
[b, a] = butter (n, Wn, 'low', 's');
disp ('the order of low pass filter is')
disp(n)
disp ('System function in analog')
printsys (b, a, 's');
fs=1;
[z, p] = bilinear [b, a, fs]
disp ('System function in digital')
printsys (z, p, 'z');
freqz (z, p);
title ('Digital butterworth LPF');
```

BGSIT

Output :

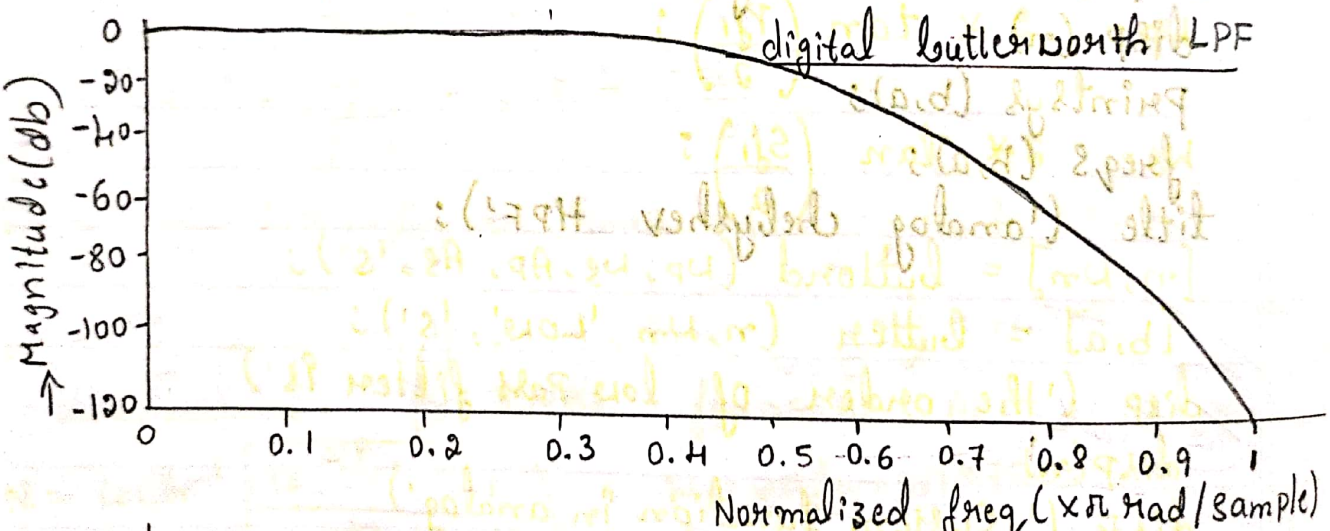
Enter the sampling frequency in Hz = 2000
Enter the passband edge frequency in Hz = 500
Enter the stopband frequency in Hz = 750
Enter the passband attenuation in db = 3
Enter the stopband attenuation in db = 15
The order of Lowpass filter is 2.

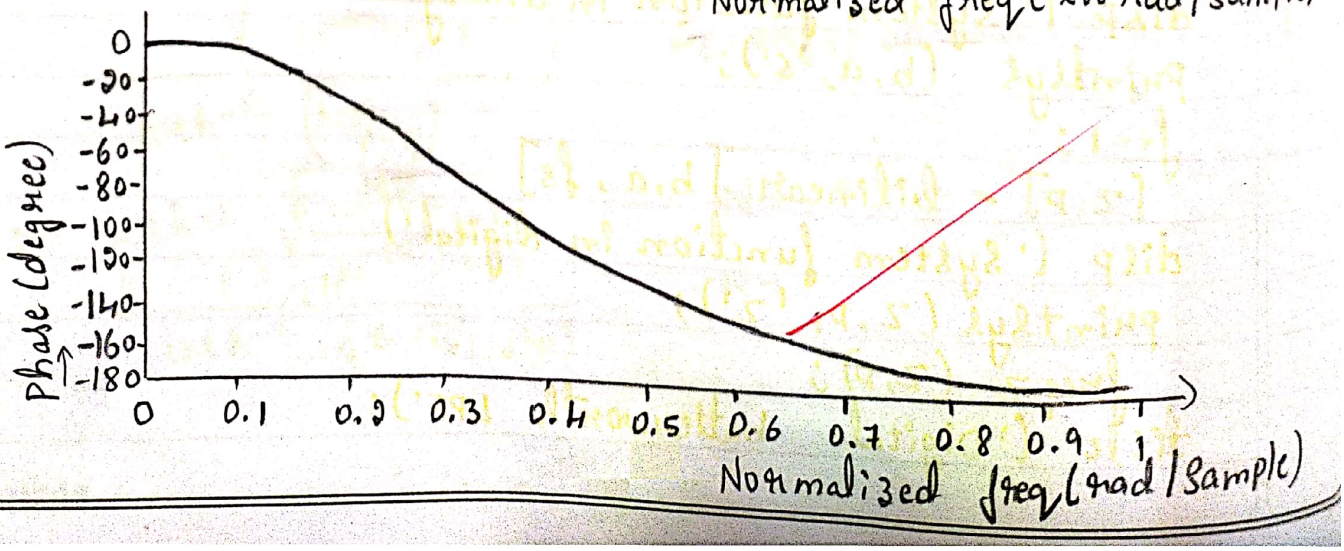
System function in analog.

$$\text{num/den} = \frac{11.213}{s^2 + 2.907\,s + 11.213}$$

System function in digital.

$$\text{num/den} = \frac{0.30053\,z^2 + 0.60106\,z + 0.30053}{z^2 + 0.030385\,z + 0.17174}$$





Keyword:

bilinear: Bilinear transformation method for analog to digital conversion.

Syntax: [zd, Pd, kd] = bilinear (z, P, K, fs)

[zd, Pd, kd] = bilinear (z, P, K, fs, fp)

Description: The bilinear transformation is a mathematical mapping of variables. In digital filtering it is a standard method the S or analog plans into the Z or Digital plans

Verification:

$fs = 2000$ $\quad Pf = 500$ $\quad Sf = 750$ $\quad Ap = -3$ $\quad As = -15$

$$Pf_1 = 2 * Pi * \left(\frac{Pf}{fs}\right) = 2 * \pi * \frac{500}{2000} = 0.5\pi \text{ rad}$$

$$Sf_1 = 2 * Pi * \left(\frac{Sf}{fs}\right) = 2 * \pi * \frac{750}{2000} = 0.75\pi \text{ rad.}$$

$T = 1$ sec.

$$Wp = \frac{2}{T} \tan\left(\frac{Pf_1}{2}\right) = \frac{2}{T} \tan\left(\frac{0.5\pi}{2}\right) = 2 \text{ rad/sec}$$

$$Ws = \frac{2}{T} \tan\left(\frac{Sf_1}{2}\right) = \frac{2}{T} \tan\left(\frac{0.75\pi}{2}\right) = 4.828 \text{ rad/sec.}$$

$$N = \frac{\log\left(\frac{10^{-Ap/10} - 1}{10^{-As/10} - 1}\right)}{2 \log\left(\frac{Wp}{Ws}\right)} = \frac{\log\left(\frac{10^{-(-3)/10} - 1}{10^{-(-15)/10} - 1}\right)}{2 \log\left(2/4.828\right)}$$

$N = 1.944$

select $\langle N = 2 \rangle$

Name of Experiment ... Digital butterworth LPF using impulse invariant Method Date : 11-10-18

Experiment No. 6(b) Page No. 46

Write a MATLAB code to design digital butterworth LPF using impulse invariant method.

```
clc;
fs = input ('Enter the Sampling frequency in Hz =');
Pf = Input ('Enter the Pass Edge frequency in Hz =');
Sf = Input ('Enter the Stopband frequency in Hz=');
AP = input ('Enter the Passband Attenuation in db=');
As = Input ('Enter the Stopband Attenuation in db=');
Wp = 2 * Pi * (Pf/fs);
Ws = 2 * Pi * (Sf/fs);
[n, Wn] = buttord (Wp, Ws, Ap, As, 's');
[b,a] = butter (n, Wn, 'Low', 's');
disp ('The order of Lowpass filter is')
disp(n)
disp ('System function in analog')
printsys (b, a, 's');
fs=1;
[z, P] = impinvar (b,a, fs)
disp ('System function in digital');
printsys (z, P, 'z');
freqz (z, P);
title ('Digital butterworth LPF');
```

BGSIT

Keyword :

Impinvar : Impulse Invariance method for analog to digital filter conversion.

Syntax : [bz, az] = impinvar (b, a, fs)

[bz, az] = impinvar (b, a, fs, tol)

Description : [bz, az] = impinvar (b, a, fs) creates a digital filter with numY & DenY co-efficients bz & az respectively whose impulse response is equal to the impulse response of the analog filter with co-efficient b and a.

Output :

Enter the sampling frequency in Hz = 5000

Enter the Passband edge frequency in Hz = 500

Enter the Stopband Edge frequency in Hz = 750

Enter the passband attenuation in db = 3

Enter the Stopband attenuation in db = 15

The order of lowpass filter is 5.

System function in analog.

$$\text{num/den} = \frac{0.13438}{S^5 + 2.1661\,S^4 + 2.3461\,S^3 + 1.5704\,S^2 + 0.64966\,S +}$$

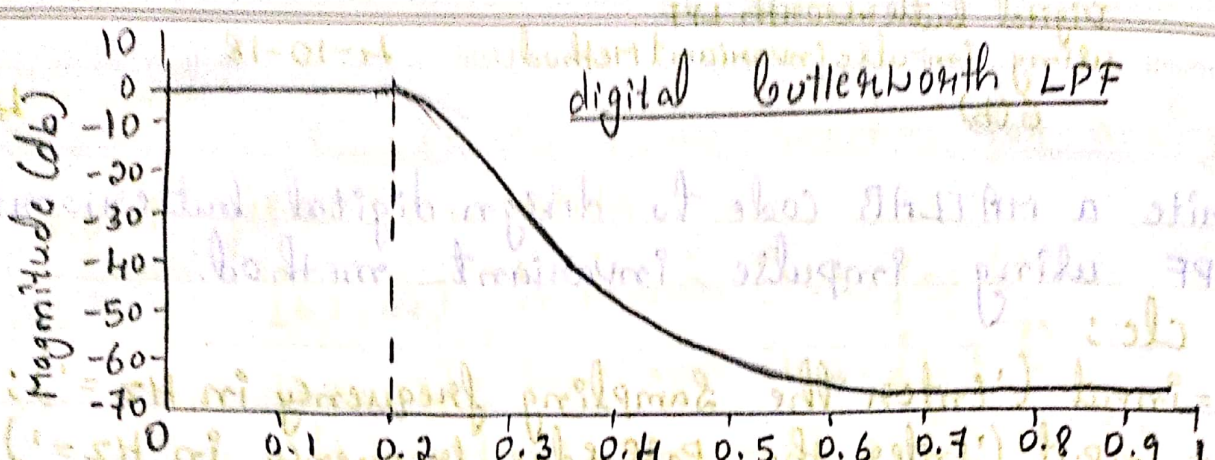
Z = 0.0000 0.0036 0.0248 0.0161 0.0010 0 0.13438

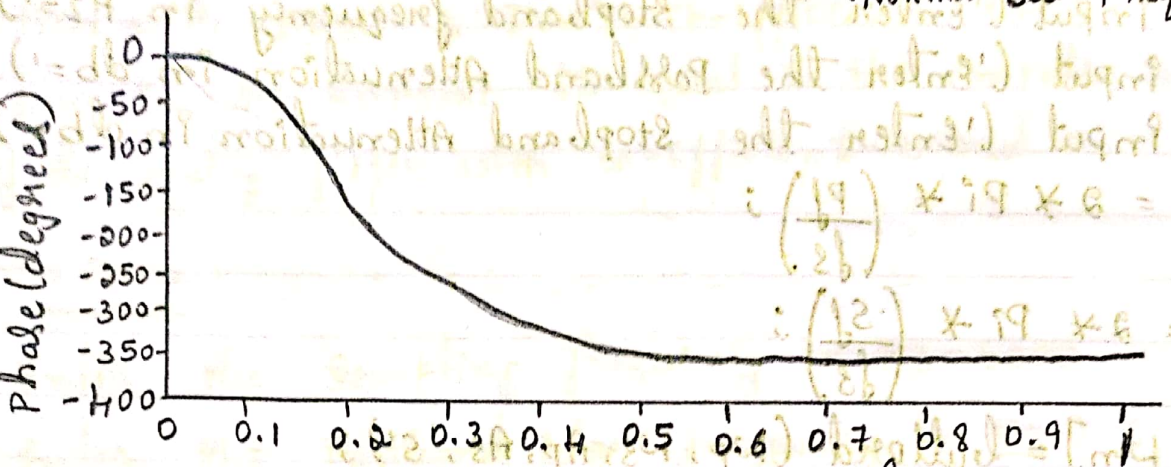
P = 1.0000 -2.8946 3.6252 -2.3850 0.8144 -0.1146

System function in digital.

$$\text{num/den} = \frac{4.4409\,e-16\,z^5 + 0.0035716\,z^4 + 0.024819\,z^3 + 0.01613\,z^2 + 0.00097505\,z}{z^5 - 2.8946\,z^4 + 3.6252\,z^3 - 2.385\,z^2 + 0.81443z - 0.1146}$$

digital butterworth LPF





Verification :-

$fs = 5000$ · $Ap = -3$ $As = -15$ $Pf = 500$ $Sf = 750$

$$Wp = 2*Pi*\left(\frac{Pf}{fs}\right) = 2*\pi*\left(\frac{500}{5000}\right) = 0.2\pi \text{ rad}$$

$$Ws = 2*Pi*\left(\frac{Sf}{fs}\right) = 2*\pi*\left(\frac{750}{5000}\right) = 0.3 \text{ rad}$$

$$N = \frac{\log\left(\frac{10^{-(-3)/10}-1}{10^{-(-15)/10}-1}\right)}{2\log\left(\frac{0.2\pi}{0.3\pi}\right)}$$

$N = 4.27$

Select $\langle N = 5 \rangle$.

Name of Experiment ..DFT is linear.. Date : 11-10-2018

Experiment No. ..7(a)..

Write a MATLAB code to verify DFT is linear.

$a x_1(n) + b x_2(n) \xleftrightarrow{DFT} a X_1(K) + b X_2(K)$.

```
clc;
x1 = input ('Enter the first input Sequence =');
x2 = input ('Enter the second input Sequence =');
N = input ('Enter the number of DFT Points =');
a = 1;
b = 1;
y1 = a.*x1 + b.*x2;
Y1 = fft (y1, N);
disp ('The values of Y1 are'):
disp (Y1)
X1 = fft (a.*x1, N);
X2 = fft (b.*x2, N);
Y2 = X1 + X2;
disp ('The values of y2 are')
disp (Y2)
if (Y1==Y2)
disp ('DFT Satisfied Linearity Properity'),
Else
disp ('DFT doesnot Satisfies linearity property')
End.
```

FH **output :**

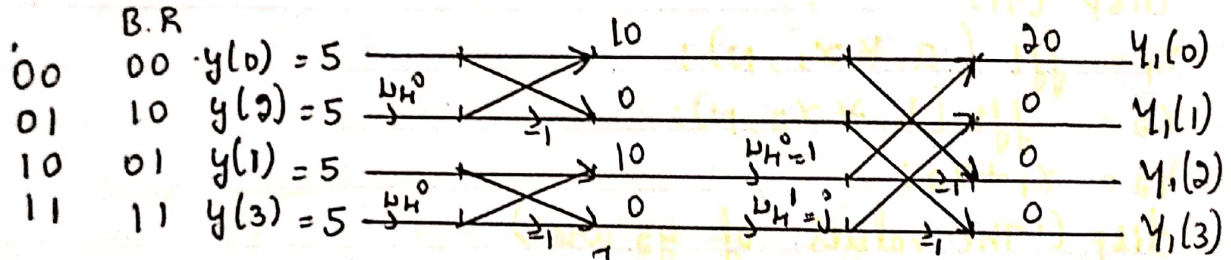
Enter the first input sequence = [1 2 3 4]
Enter the second input sequence = [4 3 2 1]
Enter the number of DFT points = 4.
The values of Y_1 are 20 0 0 0
The values of Y_2 are 20 0 0 0
DFT satisfied linearity property.

Verification:

LHS: $x_1(n) = \{1,2,3,4\}$ $x_2(n) = \{4,3,2,1\}$

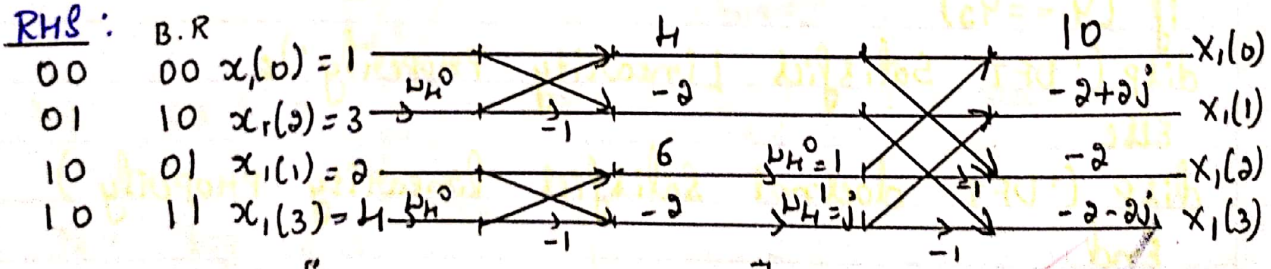
$ax_1(n) + bx_2(n) = [1\ 2\ 3\ 4] + [4\ 3\ 2\ 1]$

$y_1 = [5\ 5\ 5\ 5]$

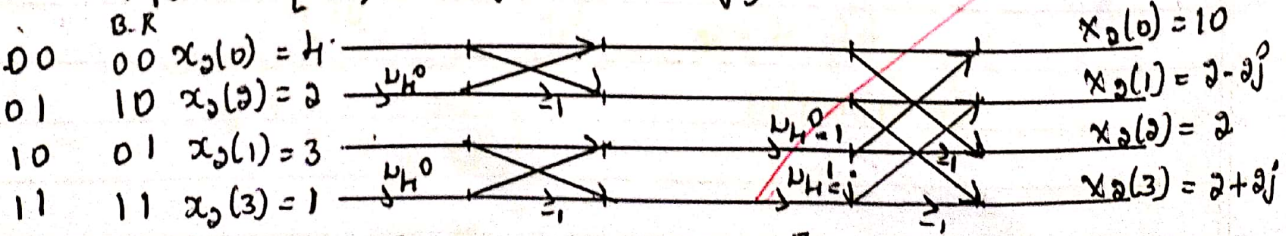


$Y_1(k) = [20\ 0\ 0\ 0]$

RHS:



$X_1(k) = [10, -2+2j, -2, -2-2j]$



$X_2(k) = [10, 2-2j, 2, 2+2j]$

$Y_2(k) = X_1(k) + X_2(k) = [10\ -2+2j\ -2\ -2-2j]\ [10\ 2-2j\ 2\ 2+2j]$

$Y_2(k) = [20\ 0\ 0\ 0]$

Name of Experiment .... Perseval's theorem.

Experiment No: ..... 7(b)

Date : ..11-10-2018..

Write a MATLAB code to Verify Perseval's theorem (or) Energy theorem

$$\sum_{n=0}^{N-1} |x(n)|^2 \xleftrightarrow{DFT} \frac{1}{N} \sum_{K=0}^{N-1} |X(K)|^2$$

```
clc;
x = input ('Enter the input sequence=');
N = input ('Enter the Number of DFT Points =');
Energy1 = sum (abs (x.^2));
disp ('the values of Energy1 are')
disp (Energy1)
X = fft (x, N);
Energy2 = 1/N * sum (abs (x.^2));
disp('the values of Energy 2 are')
disp (Energy 2)
if (Energy1 == Energy 2)
disp ('DFT satisfies perseval's theorem')
Else
disp ('DFT doesnot satisfies perseval's theorem')
End
```

12/12 11/10/18

Keyword :

Sum : Sum of array Elements

Syntax : B = Sum(A)

B = Sum (A, dim)

Description : B = Sum(A) returns sums along different dimensions of an array.

If A is a vector, Sum(A) returns the sum of the Elements

If A is a matrix, Sum (A) treats the Column of A al vectors, returning a row vector of the sums of Each Column.

Output :

Enter the input sequence = [1 2 3 4]

Enter the number of DFT Points = 4

The values of Energy1 are 30

The values of Energy2 are 30

Dft satisfied Perseval's theorem.

Name of Experiment FIR using BLACKMAN'S window Date : 25-10-2018

Experiment No. 8 (a) Page No. 50

Write a MATLAB code to design FIR filter using BLACKMAN'S window

```
clc;
fcut = input ('Enter the cutoff frequency=');
fs = input ('Enter the sampling frequency=');
N = input ('Enter the order of filter =');
Wc = (2* fcut) /fs;
L = N+1;
w = blackman (L);
b = fir1 (N, Wc, 'Low', w)
freqz(b)
title ('FIR filter using blackman');
```

Keyword:

fir1: window-based finite impulse response filter design

Syntax: b = fir1(n, Wc, 'ftype', window)

Description: b = fir1(n, Wc, 'ftype', window) accepts both 'ftype' and window parameter.

Blackmann: Blackmann window.

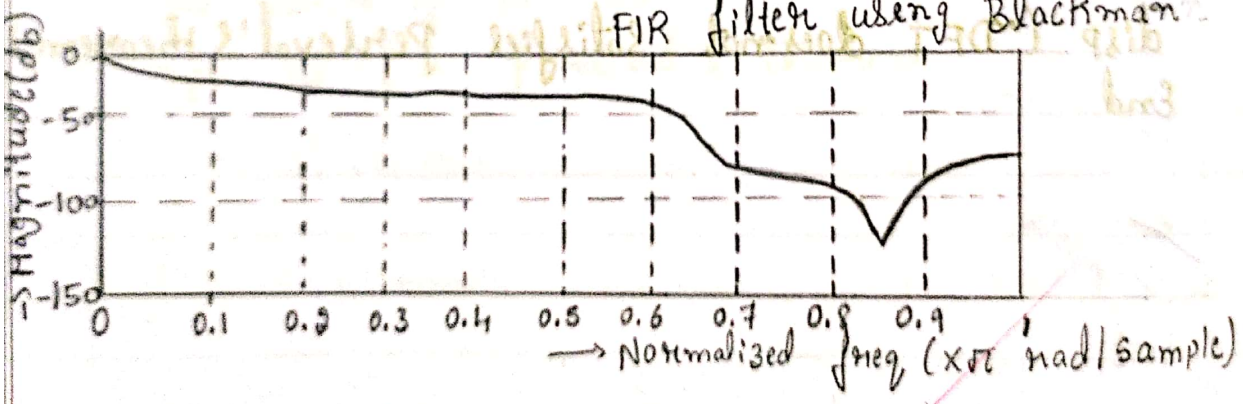
Syntax: w = blackmann(L)

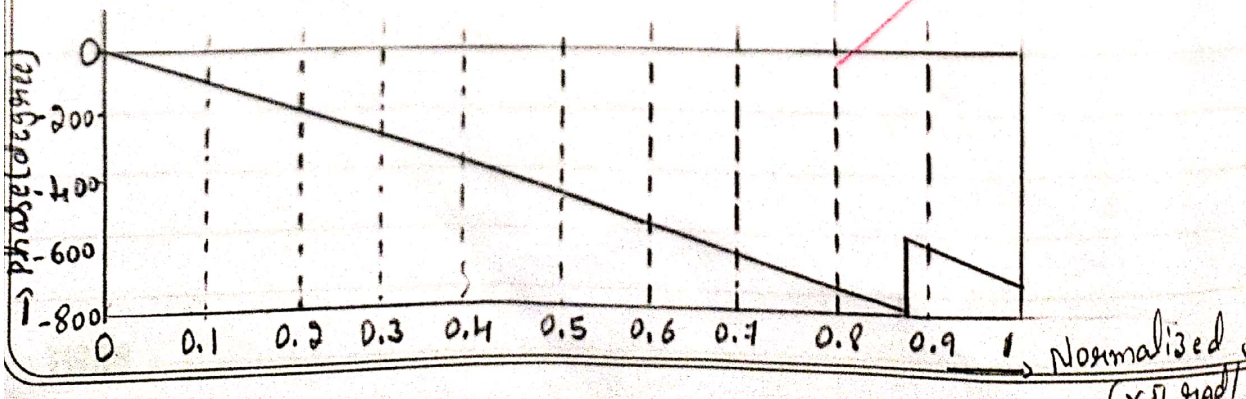
Description: w = blackmann(L) returns the L-point symmetric Blackmann window in the column vector w, where L is a positive integer.

Output:

Enter the cutoff freq = 100
Enter the sampling freq = 1000
Enter the order of filter = 10

b = -0.0000 0.0026 0.0283 0.1077 0.2218 0.2792 0.2218
0.1077 0.0283 0.0026 -0.0000





Name of Experiment FIR using Hamming window Date: 25-10-2018

Experiment No. 8(b) Page No. 51

Write a MATLAB code to design FIR filter using hamming window.

```
clc;
fcut = input ('Enter the cutoff frequency=');
fs = input ('Enter the sampling frequency=');
N = input ('Enter the order of the filter=');
wc = (2* fcut) /fs;
L=N+1;
w= hamming (L);
b= fir1 (N, wc, 'low', w)
freqz(b)
title ('FIR filter using hamming');
```

Keyword:

hamming : Hamming window

Syntax : W = hamming (L)

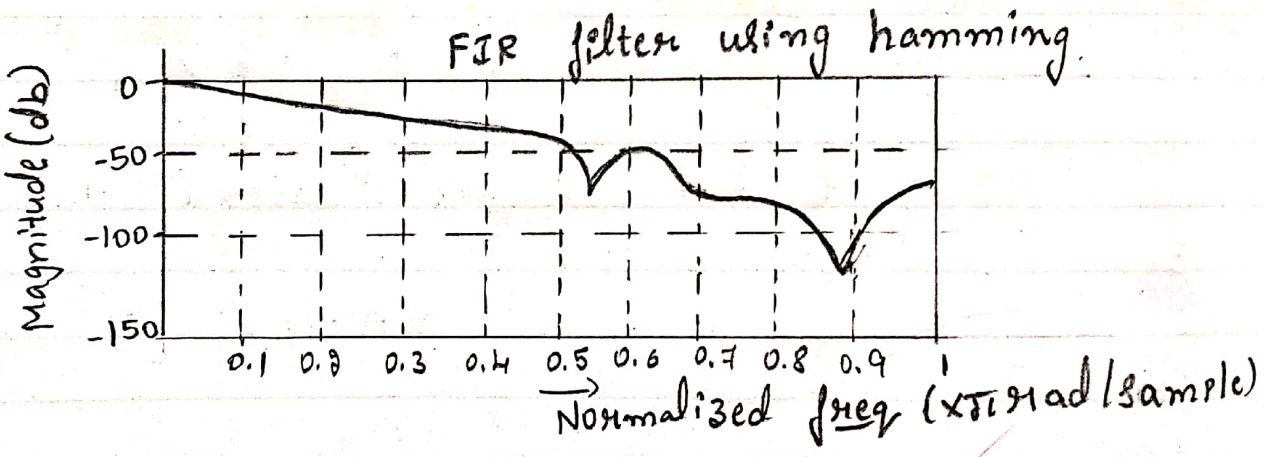
Description : W = hamming (L) returns an L-point symmetric Hamming window in the column vector W. L should be a positive integer. The coefficients of a Hamming window are computed from the following eqn.

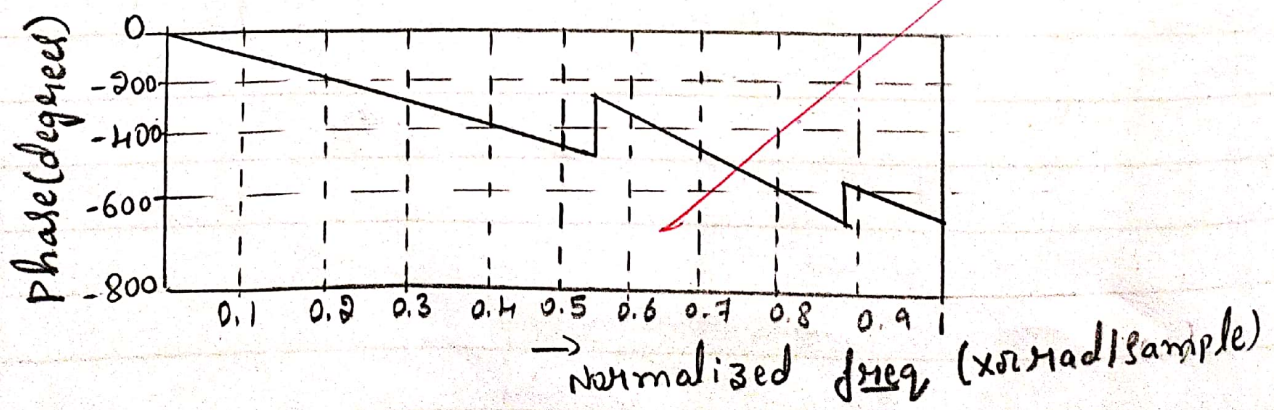
$$W(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N}\right), \quad 0 \le n \le N.$$

The window length is $L = N + 1$

Output :

```
enter the cutoff freq = 100
enter the sampling freq = 1000
enter the order of the filter = 10
b = 0.0000  0.0093  0.0476  0.1224  0.2022  0.2370  0.2022
            0.1224  0.0476  0.0093  0.0000
```





Name of Experiment FIR using Hann window. Date : 25-10-2018

Experiment No. 8 (c) Page No. 52

Write a MATLAB code to design FIR filter using Hanning window.

```
clc;
fcut = input ('Enter the cutoff frequency=');
fs = input ('Enter the sampling frequency=');
N = input ('Enter the order of the filter =');
wc = (2* fcut) / fs;
L = N+1;
w = hamm(L);
b = fir1 (N, wc, 'low', w)
freqz(b)
title ('FIR filter using hann');
```

Keyword:

Hann: Hann (Hanning) window.

Syntax: w = hann(L)

Description: w = hann(L) returns an L-point symmetric Hann window in the column vector w. L must be a positive integer. The co-efficients of a Hann window are completed from the following equation.

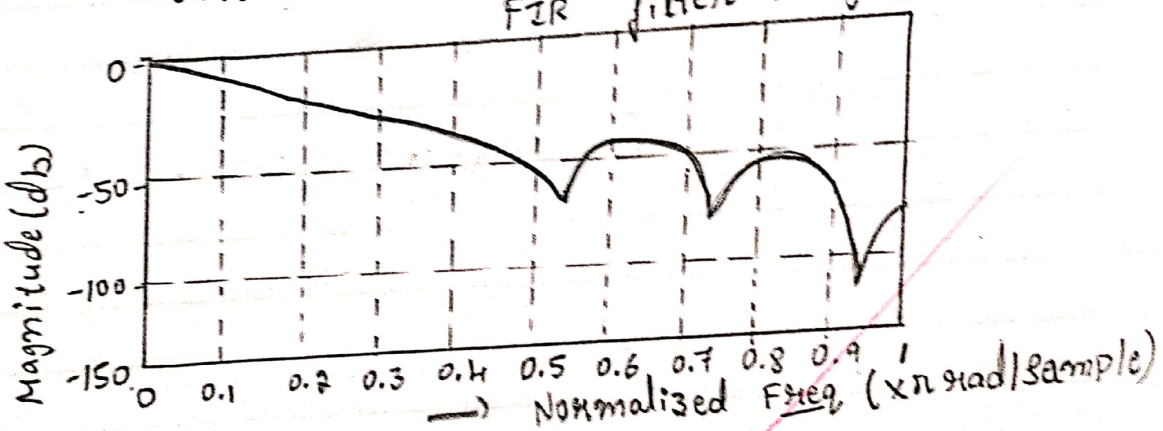
$$w(n) = 0.5\left(1 - \cos\left(2\pi \frac{n}{N}\right)\right), \quad 0 \le n \le N.$$

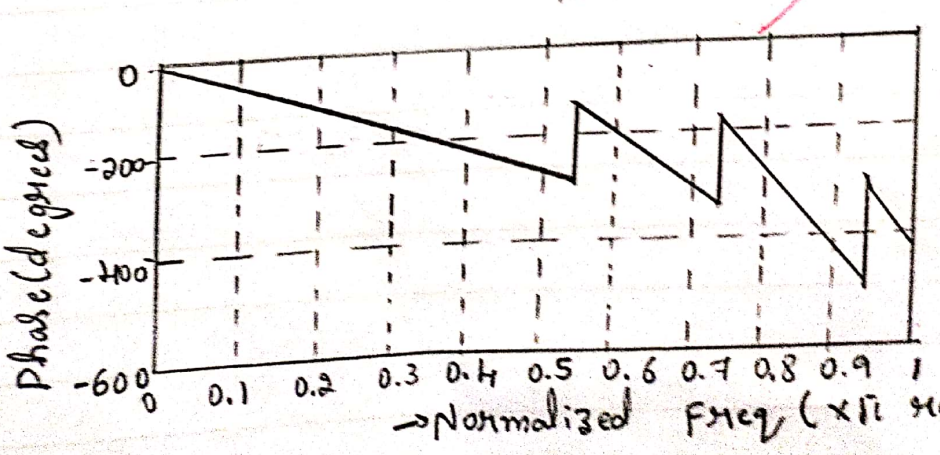
The window length is L = N+1.

Output:

Enter the cutoff frequency = 100
Enter the sampling frequency = 1000
Enter the order of filter = 10

b = 0 0.0055 0.0428 0.1215 0.02076 0.2453 0.2076
0.1215 0.0428 0.0055 0.000





Name of Experiment FIR using Kaiser window Date : 25-10-2018

Experiment No. 8(d) Page No. 53

Write a MATLAB code to design FIR filter using Kaiser window.

```
clc;
fcut = input ('Enter the cutoff frequency =');
fs = input ('Enter the sampling frequency =');
N = input ('Enter the order of the filter =');
wc = (2* fcut) / fs;
L = N+1;
w = kaiser (L);
b = fir1 (N, wc, 'low', w)
freqz(b)
title ('FIR filter using Kaiser');
```

Keyword:

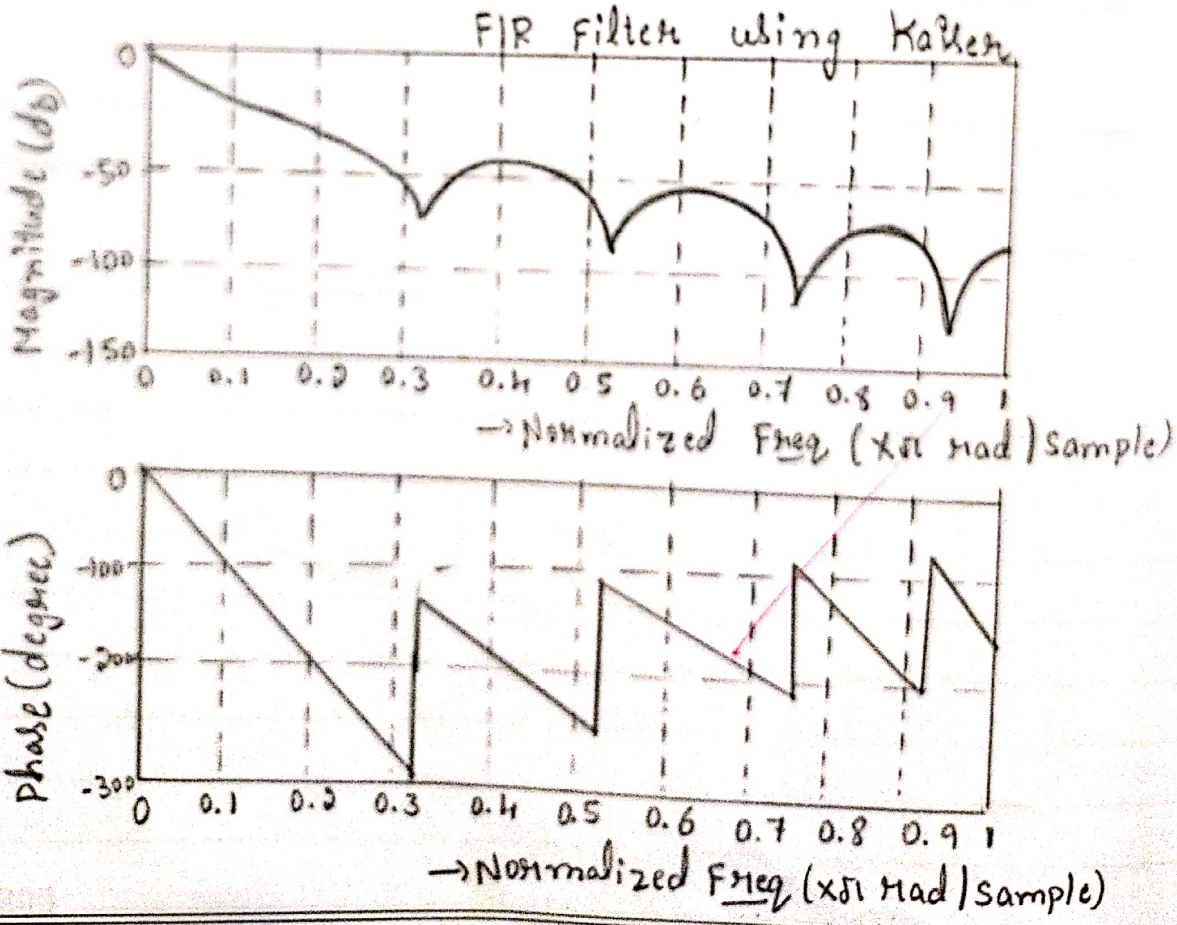
kaiser : Kaiser window

Syntax : w = kaiser (L, beta)

Description: w = kaiser (L, beta) returns an L-point Kaiser (Io Io-Sinh) window in the column vector w. beta is the kaiser window β parameter that affects the sidelobe attenuation of the fourier transform of the window. The default value for beta is 0.5.

output: Enter the cutoff frequency = 100
Enter the sampling frequency = 1000
Enter the order of the filter = 10

b = 0.0000 0.0388 0.0851 0.1292 0.1608 0.1723 0.1608
0.1292 0.0851 0.0388 0.0000



Name of Experiment FIR using rectangular windows Date : 25-10-2018

Experiment No. 8 (e)

Write a MATLAB code to design FIR filter using rectangular windows.

```
clc;
fcut = input ('Enter the cutoff frequency =');
fs = input ('Enter the sampling frequency =');
N = input ('Enter the order of the filter =');
Wc = (2* fcut) /fs;
L = N+1;
W = rectwin (L);
b = fir1 (N, Wc, 'LOW', W)
freqz(b)
title ('FIR filter using rectwin');
```

10/10 25/10/18

Keyword:

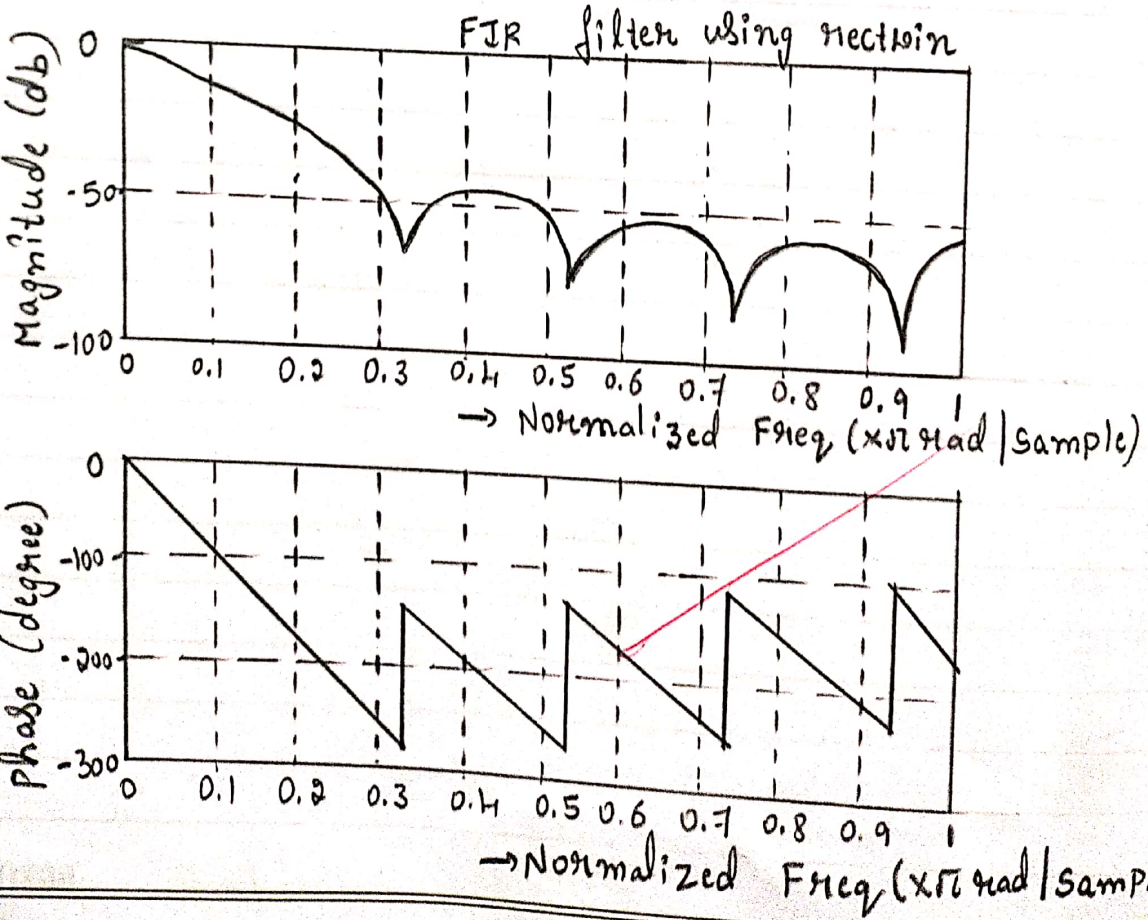
Rectwin: Rectangular window

Syntax: W = rectwin(L)

Description: W = rectwin(L) returns a rectangular window length L in the column vector W. This function is provided for completness; a rectangular window is equivalent to no window at all.

output: Enter the cutoff frequency = 100
Enter the sampling frequency = 1000
Enter the order of the filter = 10

b = 0.0000 0.0399 0.0861 0.1291 0.1596 0.1706 0.1596
0.1291 0.0861 0.0399 0.0000



Name of Experiment ..... auto correlation ..... Date : 25/10/18

Experiment No. ..... 9(a) ..... Page No. 55

Write MATLAB code to find auto correlation between two sequences.

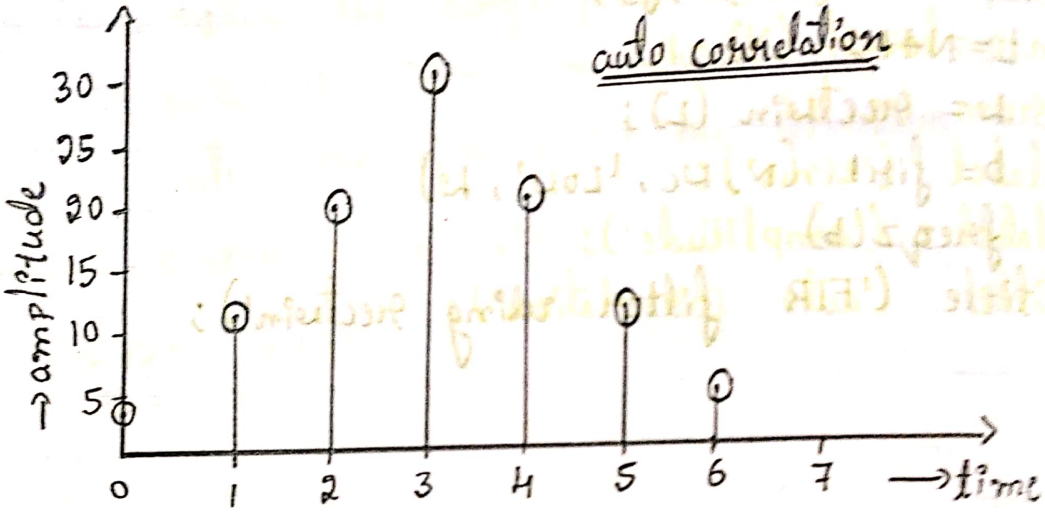
```
clc;
a = input ('Enter the sequence =');
Y = xcorr (a, a);
disp ('the values of Y are =')
disp (Y)
n = 0 : length (Y) - 1;
stem (n, Y)
xlabel ('time');
ylabel ('amplitude');
title ('auto correlation');
```

Output :

Enter the sequence = [1 2 3 4]

a = 1 2 3 4

The values of y are 4.0000 11.0000 20.0000 30.0000
20.0000 11.0000 4.0000



Name of Experiment ..... Cross correlation ..... Date : 25/10/18

Experiment No. ..... 9 (b) ..... Page No. 56

Write a MATLAB code to find cross correlation between two sequences.

```
clc;
a=input ('Enter the first sequence =');
b=input ('Enter the second sequence =');
y=xcorr(a,b);
disp ('The value of y are =')
disp(y)
n=0 : length (y)-1;
stem(n,y)
xlabel ('time');
ylabel ('amplitude');
title ('cross correlation');
```

Good

12/12

Keyword :

xcorr : Cross correlation

Syntax : C = xcorr (x, y)

Description: C = corr (x, y) returns the cross correlation sequence in a length $2^* N-1$ vector, where x & y are length N vectors (N>1). If x and y are the not same length, the shorter vector is zero-padded to the length of the longer vector.

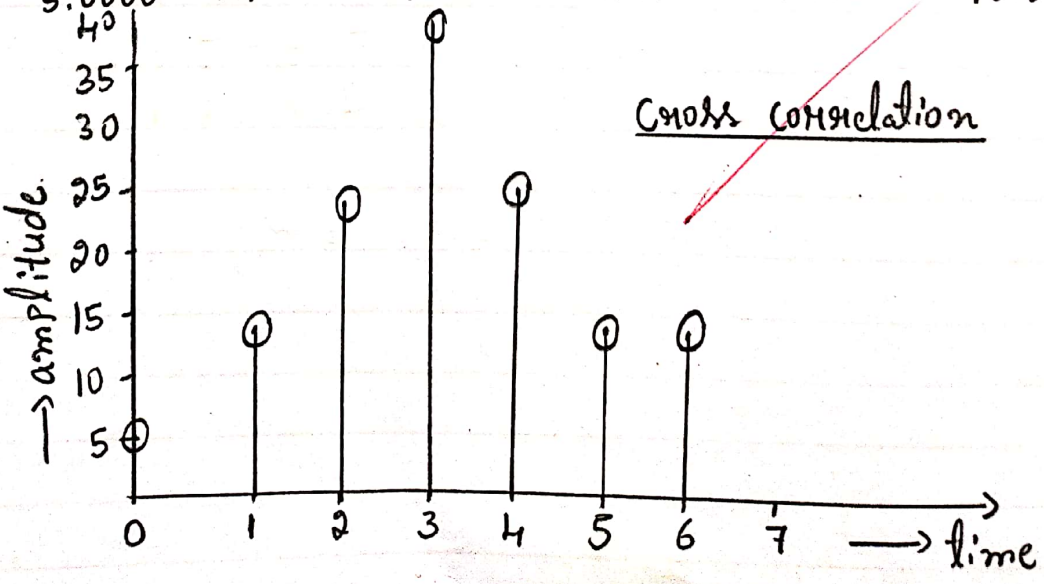
C = xcorr(x) is autocorrelation sequence for the vector x. If x is an N-by-P matrix, C is a matrix with 2N-1 rows whose P^2 column contain the cross-correlation sequences for all combination of the column of x.

output :

Enter the first sequence = [1 2 3 4]
Enter the second sequence = [3 1 4 5]

The values of y are =

5.0000 14.0000 24.0000 37.0000 25.0000 13.0000 12.0000



Name of Experiment ..... Linear Convolution ..... Date : ..... 15/11/18 .....

Experiment No. ..... 1 .....

DSP LAB

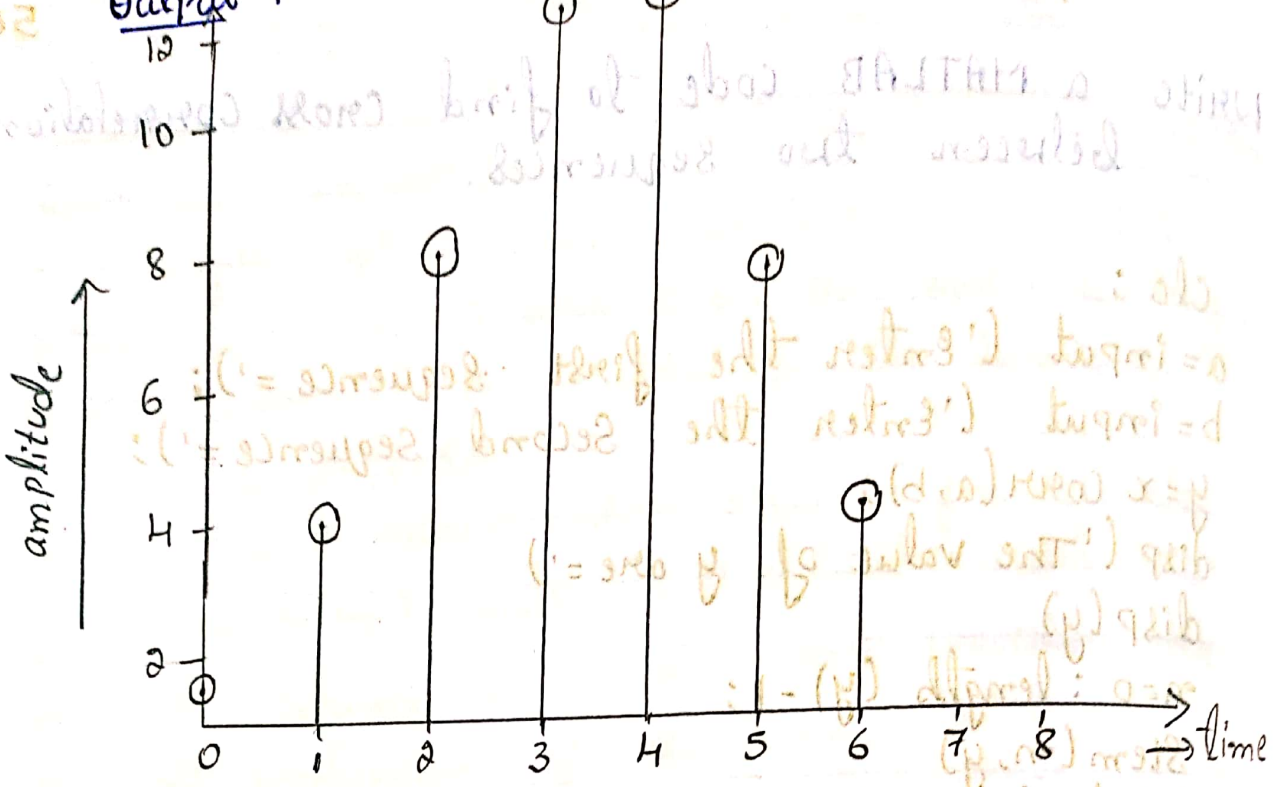
Part B

LINEAR CONVOLUTION

/*C-CODE to perform linear convolution between the two sequences*/

```
#include<stdio.h>
void main( )
{
   int m=4;
   int n=4;
   int i=0,j;
   int x[10]={1,2,1,1};
   int h[10]={1,2,3,4};
   int *y=(int *)0x0000100;
   for(i=0;i<m+n-1;i++)
   {
      y[i]=0;
      for(j=0;j<=i;j++)
      y[i]+=x[j]*h[i-j];
   }
  for(i=0;i<m+n-1;i++)
printf("%d\n", y[i]);
}
```

Output :



Verification :-

$x(n) = \{1, 2, 1, 1\}$ $h(n) = \{1, 2, 3, 4\}$

	1	2	1	1
1	1	2	1	1
2	2	4	2	2
3	3	6	3	3
4	4	8	4	4

$y(n) = \{1, 4, 8, 13, 13, 7, 4\}$ //

Name of Experiment ....Circular convolution..... Date : ....15/11/18.....

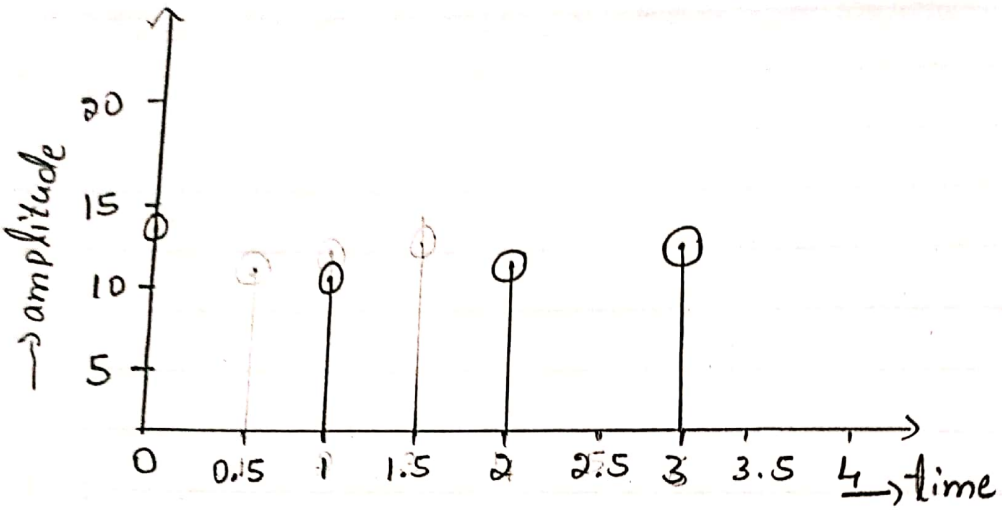
Experiment No. .....2..... Page No. 58

CIRCULAR CONVOLUTION

/*C-CODE to perform Circular convolution between the two sequences*/

```
#include<stdio.h>
#include<math.h>
void main()
{
   float x[4]={1,2,3,4};
   float h[4]={1,2,1,1};
   float *y=(float *) 0x0000100;
   int N=4;
   int k,n,i;
   for(n=0;n<N;n++)
   {
      y[n]=0;
      for(k=0;k<N;k++)
      {
         i=(n-k)%N;
         if(i<0)
         i=i+N;
         y[n]=y[n]+h[k]*x[i];
      }
      printf("%f\t ",y[n]);
   }
}
```

output :



Verification :

$x(n) = \{1, 2, 3, 4\}$ $\quad h(n) = \{1, 2, 1, 1\}$

$$x(n) \circledast_N h(n) = \begin{bmatrix} 1 & 4 & 3 & 2 \\ 2 & 1 & 4 & 3 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1+8+3+2 \\ 2+2+4+3 \\ 3+4+1+4 \\ 4+6+2+1 \end{bmatrix} = \begin{bmatrix} 14 \\ 11 \\ 12 \\ 13 \end{bmatrix}$$

Name of Experiment ..Impulse Response..... Date : ..15/11/18.....

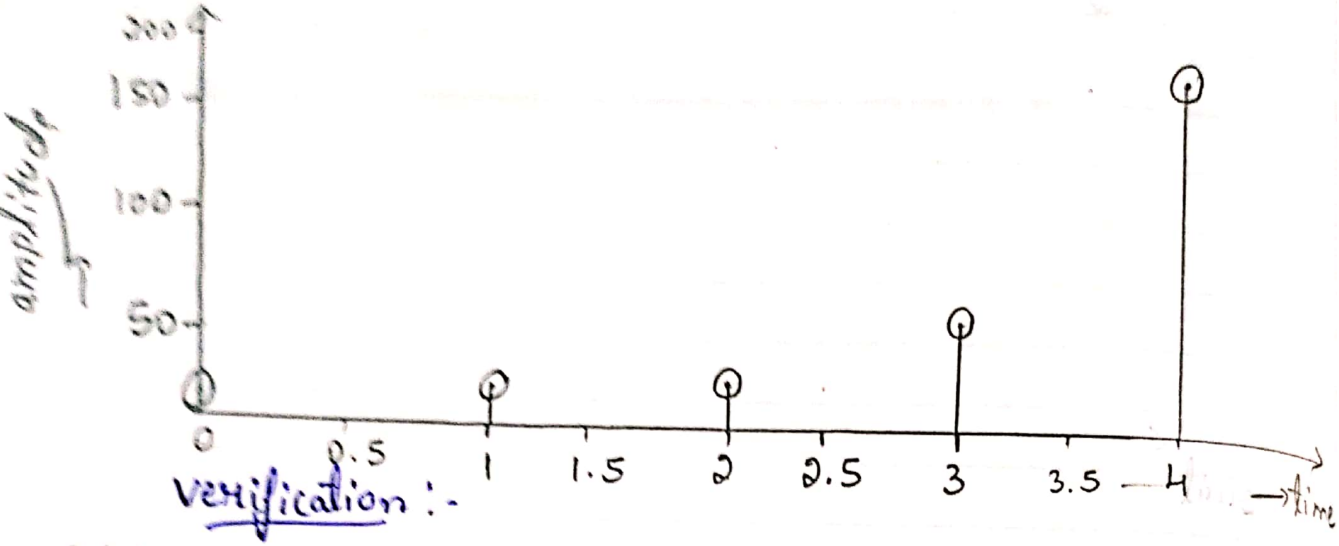
Experiment No. .....3..... Page No. 59

IMPULSE RESPONSE

/*C-CODE to obtain impulse response for the given filter Coefficients*/

```
#include<stdio.h>
float x[50],y[50];
void main( )
{
    float a0,a1,a2,b0,b1,b2;
    int i,j,N=5;
    a0=1;
    a1=-3.5;
    a2=1.5;
    b0=3;
    b1=-4;
    b2=0;
    x[0]=1;
    for(i=1;i<N;i++)
    x[i]=0;
    for(j=0;j<N;j++)
    {
        y[j]=b0*x[j]-a0*y[j];
        if(j>0)
        y[j]=y[j]+b1*x[j-1]-a1*y[j-1];
        if((j-1)>0)
        y[j]=y[j]+b2*x[j-2]-a2*y[j-2];
        printf("%f\t ",y[j]);
    }
}
```

output :



verification :-

$$H(z) = \frac{3-4z^{-1}}{1-3.5z^{-1}+1.5z^{-2}} \qquad N=5$$

$$H(z) = \frac{Y(z)}{X(z)} \Longleftrightarrow \frac{3-4z^{-1}}{1-3.5z^{-1}+1.5z^{-2}}$$

$$Y(z) - 3.5z^{-1}Y(z) + 1.5z^{-2}Y(z) = 3X(z) - 4z^{-1}X(z)$$

By I-ZT

$$y(n) - 3.5y(n-1) + 1.5y(n-2) = 3x(n) - 4x(n-1)$$

$$x(n) = \delta(n)$$

$$y(n) = 3\delta(n) - 4\delta(n-1) + 3.5y(n-1) - 1.5y(n-2)$$

N=5, 0, 1, 2, 3, 4 Assume initial values is zero

@ N=0, $y(0) = 3\delta(0) - 4\delta(0-1) + 3.5y(0-1) - 1.5y(0-2) \Rightarrow \boxed{y(0)=3}$

@ N=1, $y(1) = 3\delta(1) - 4\delta(1-1) + 3.5y(1-1) - 1.5y(1-2)$

$y(1) = -4 + 3.5(3) \Rightarrow \boxed{y(1) = 6.5}$

@ N=2, $y(2) = 3\delta(2) - 4\delta(2-1) + 3.5y(2-1) - 1.5y(2-2)$

$y(2) = 3.5(-0.5) - 1.5(3) \Rightarrow \boxed{18.25 = y(2)}$

@ N=3, $y(3) = 3\delta(3) - 4\delta(3-1) + 3.5y(3-1) - 1.5y(3-2)$

$y(3) = 3.5[18.25] - 1.5(6.5) \Rightarrow \boxed{54.125 = y(3)}$

@ N=4, $y(4) = 3\delta(4) - 4\delta(4-1) + 3.5y(4-1) - 1.5y(4-2) \Rightarrow \boxed{y(4) = 162.0625}$

$y(n) = \{3, 6.5, 18.25, 54.125, 162.0625\}$ //

Name of Experiment N-Point DFT Date : 15/11/18

Experiment No. 5 Page No. 60

N-POINT DFT

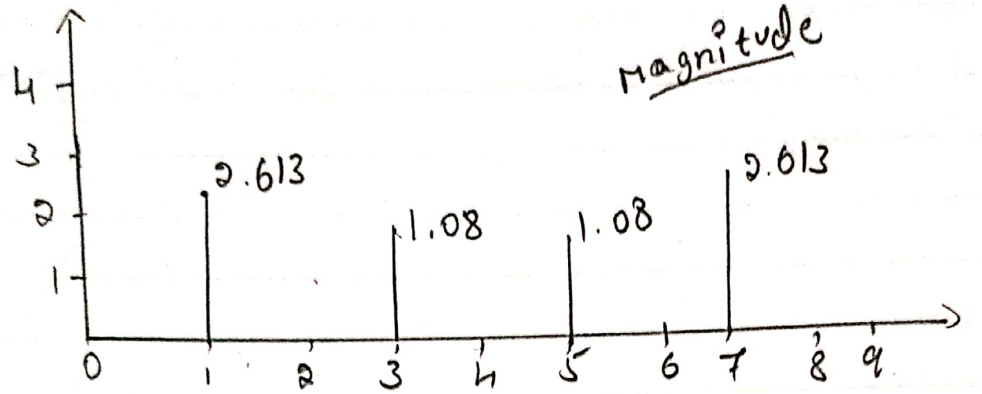
```
/*C-CODE to perform N-point DFT between the two sequence*/
#include<stdio.h>
#include<math.h>
void main()
{
    float x[8]={1,1,1,1,0,0,0,0};
    float w;
    int n,k,k1,N=8,xlen=8;
    float *y=(float *)0x0000100;
    float *y1=(float *)0x0000300;
    for(k=0;k<2*N;k=k+2)
    {
        y[k]=0;
        y1[k+1]=0;
        k1=k/2;
        for(n=0;n<xlen;n++)
        {
            w=-2*3.14*k1*n/N;
            y[k]=y[k]+x[n]*cos(w);
            y1[k+1]=y1[k+1]+x[n]*sin(w);
        }
        Printf("%f+j%f\n", y[k],y1[k+1]);
    }
}
```

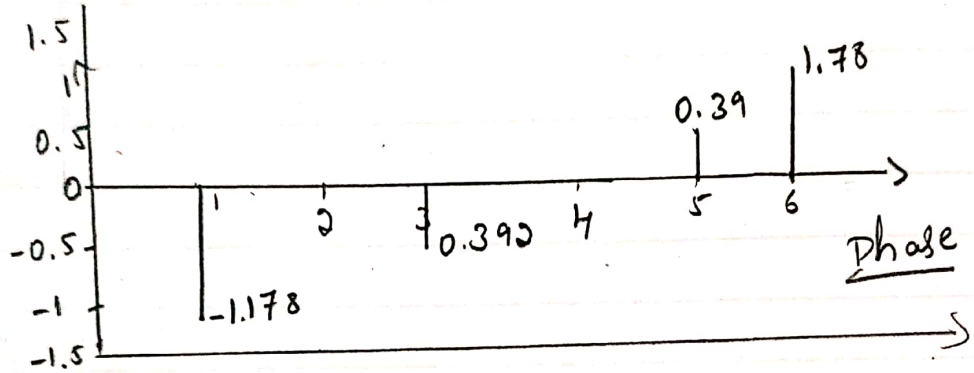
12/12

27/11/18

BGSIT

output :

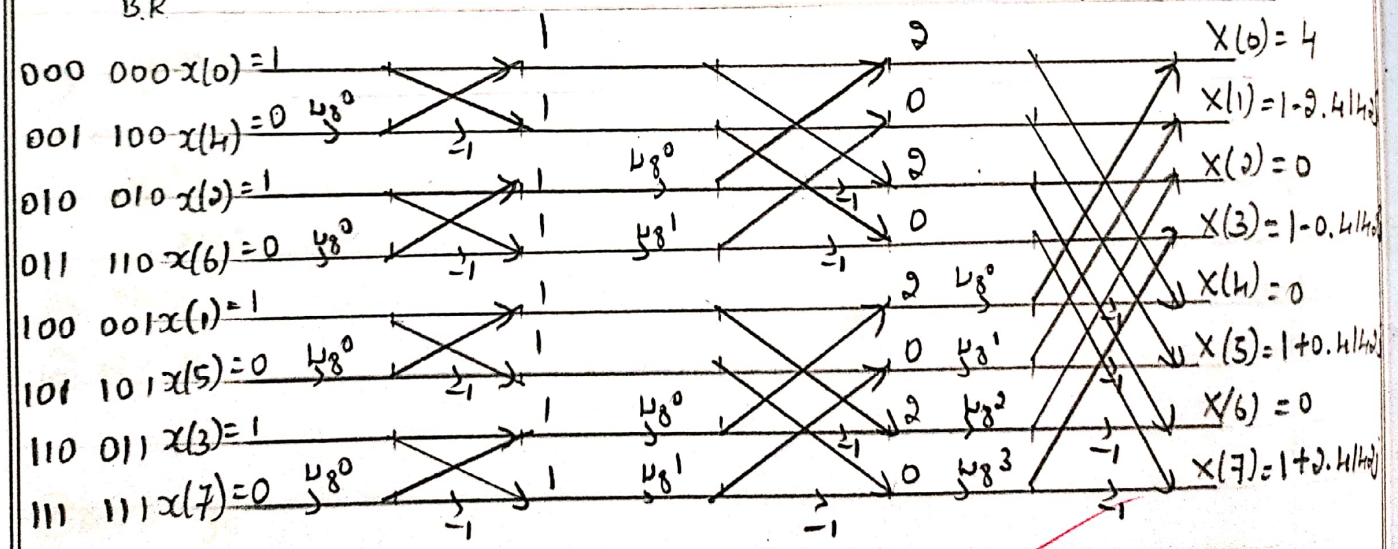




Verification :-

$x(n) = \{1, 1, 1, 1, 0, 0, 0, 0\}$

B.R



$X(k) = \{4, 1-2.414j, 0, 1-0.414j, 0, 1+0.414j, 0, 1+2.414j\}$