||Jai Sri Gurudev||
Sri Adichunchanagiri Shikshana Trust®
BGS INSTITUTE OF TECHNOLOGY
(Affiliated to VTU Belagavi, Approved by AICTE, New Delhi)
BG Nagara – 571448 (Bellur Cross)
Nagamangala Taluk, Mandya District.

DIGITAL SIGNAL PROCESSING LABORATORY MANUAL

17ECL57

For

V Semester B.E. E&CE

2019-2020

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION ENGINEERING

Prepared by:

1. Mrs. Prabhavathi K, Asst. Prof
2. Mr. Jayanth dwijesh H P, Asst. Prof
3. Ms. Lakshmi D L, Asst. Prof

Approved by:

Head of Department

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

VISION:

To develop high quality engineers with technical knowledge, skills and ethics in the area of Electronics and Communication Engineering to meet industrial and societal needs.

MISSION:

1. To provide high quality technical education with up-to-date infrastructure and trained human resources to deliver the curriculum effectively in order to impart technical knowledge and skills.
2. To train the students with entrepreneurship qualities, multidisciplinary knowledge and latest skill sets as required for industry, competitive examinations, higher studies and research activities.
3. To mould the students into professionally-ethical and socially- responsible engineers of high character, team spirit and leadership qualities.

PROGRAM EDUCATIONAL OBJECTIVES (PEO's):

After 3 to 5 years of graduation, the graduates of Electronics and Communication Engineering will;

1. Engage in industrial, teaching or any technical profession and pursue higher studies and research.
2. Apply the knowledge of Mathematics, Science as well as Electronics and Communication Engineering to solve social engineering problems.
3. Understand, Analyze, Design and Create novel products and solutions.
4. Display professional and leadership qualities, communication skills, Team spirit, multidisciplinary traits and lifelong learning aptitude.

DIGITAL SIGNAL PROCESSING LAB SYLLABUS

Course Learning Objectives:

This course will enable students to

- Simulate discrete time signals and verification of sampling theorem.
- Compute the DFT for a discrete signal and verification of its properties using MATLAB.
- Find solution to the difference equations and computation of convolution and correlation along with the verification of properties.
- Compute and display the filtering operations and compare with the theoretical values.
- Implement the DSP computations on DSP hardware and verify the result.

Laboratory Experiments

1. Verification of sampling theorem.
2. Linear and circular convolution of two given sequences, Commutative, distributive and associative property of convolution.
3. Auto and cross correlation of two sequences and verification of their properties
4. Solving a given difference equation.
5. Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum (using DFT equation and verify it by built-in routine).
6. (i) Verification of DFT properties (like Linearity and Parseval's theorem, etc.)
   (ii) DFT computation of square pulse and Sinc function etc.
7. Design and implementation of FIR filter to meet given specifications (using different window techniques).
8. Design and implementation of IIR filter to meet given specifications.

Following Experiments to be done using DSP kit

9. Linear convolution of two sequences
10. Circular convolution of two sequences
11. N-point DFT of a given sequence
12. Impulse response of first order and second order system
13. Implementation of FIR filter

Beyond Syllabus:

1. Implementation of Decimation Process

Course Outcomes:

On the completion of this laboratory course, the students will be able to:

- Apply the fundamental DSP concepts of MATLAB to solve DSP problems.
- Analyze the properties of DFT to perform circular convolution and linear convolution.
- Analyze the concepts of Sampling theorem and correlation of signals.
- Design IIR and FIR filters for given specifications.
- Support practically with DSP kit, the design of filters.

INTRODUCTION

DIGITAL SIGNAL PROCESSING (DSP)

Digital signal processing (DSP) refers to various techniques for improving the accuracy and reliability of digital communications. The theory behind DSP is quite complex. Basically, DSP works by clarifying, or standardizing, the levels or states of a digital signal.

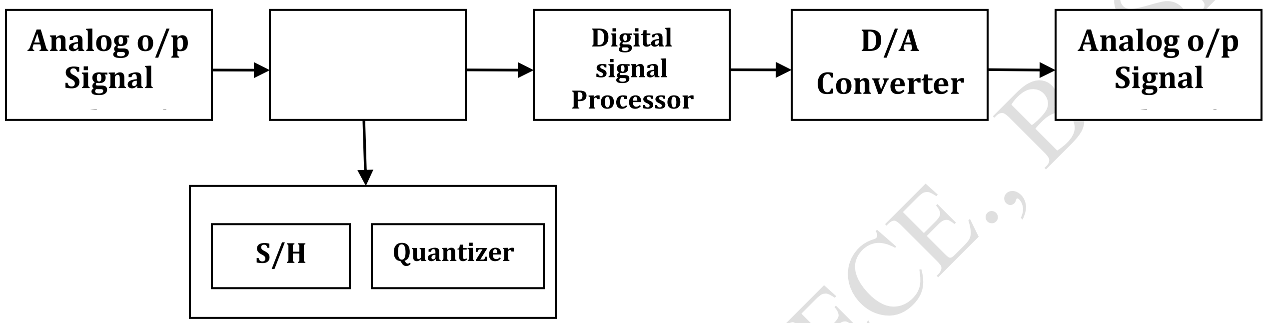


Figure: Basic Blocks of DSP Systems

As shown in the above figure:-

- Analog i/p signal is given to A/D converter.
- A/D converter internally contains sample/hold circuit and Quantizer.
- When the analog i/p signal is given to a sample/hold circuit, so analog i/p signal is sampled, based on the sample/hold circuit and each sample contains a time period of Ts.
- The sample/hold circuit o/p is a given to the Quantizer.
- The main role of a Quantizer is, it will places the amplitude of different signals in the same amplitude level.
- The o/p of the A/D converter is a digital signal
- The digital signal is given to the digital signal processor.
- The DSP is a special processor, which is used only for converting the continuous signal to the discrete signal.
- The DSP O/P is a given to the D/A converter. The D/A converter, converts the digital signal to analog signal.

Advantages of DSP

1. A digital programmable system allows flexibility in reconfiguring the digital signal processing operations by changing the program. In analog redesign of hardware is required.
2. In digital accuracy depends on word length, floating Vs fixed point arithmetic etc. In analog depends on components.
3. Can be stored on disk.
4. It is very difficult to perform precise mathematical operations on signals in analog form but these operations can be routinely implemented on a digital computer using software.
5. Cheaper to implement.
6. Small size.
7. Several filters need several boards in analog, whereas in digital same DSP processor is used for many filters.

Applications of DSP

1. Filtering.
2. Speech synthesis in which white noise (all frequency components present to the same level) is filtered on a selective frequency basis in order to get an audio signal.
3. Speech compression and expansion for use in radio voice communication.
4. Speech recognition.
5. Signal analysis.
6. Image processing: filtering, edge effects, and enhancement.
7. PCM used in telephone communication.
8. High speed MODEM data communication using pulse modulation systems such as FSK, QAM etc. MODEM transmits high speed (1200-19200 bits per second) over a band limited (3-4 KHz) analog telephone wire line.
9. Wave form generation.

INTRODUCTION TO MATLAB

MATLAB is a software package for high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include the following

- Math and computation
- Algorithm development¬ Data acquisition
- Modelling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.

MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.

The main features of MATLAB

1. Advance algorithm for high performance numerical computation ,especially in the Field matrix algebra.
2. A large collection of predefined mathematical functions and the ability to define one's own functions.
3. Two-and three dimensional graphics for plotting and displaying data .
4. A complete online help system.
5. Powerful, matrix or vector oriented high level programming language for individual applications.

6. Toolboxes available for solving advanced problems in several application areas.

BASIC MATLAB KEYWORDS OR FUNCTIONS

Clc: Clear Command Window.

Syntax: clc

Description: clc clears all the text from the Command Window, resulting in a clear screen. After running clc, you cannot use the scroll bar in the Command Window to see previously displayed text. You can, however, use the up-arrow key ↑ in the Command Window to recall statements from the command history.

Input: Request user input.

Syntax: x = input (prompt)

Description: x =input (prompt) displays the text in prompt and waits for the user to input a value and press the Return key. The user can enter expressions, like pi/4 or rand ,and can use variables in the workspace.

Disp: Display value of variable

Syntax: disp (X)

Description: disp (X) displays the value of variable X without printing the variable name. Another way to display a variable is to type its name, which displays a leading “X =” before the value. If a variable contains an empty array, disp returns without displaying anything.disp

Stem: Plot discrete sequence data

Syntax: stem(Y)

Stem(X, Y)

Description: Stem (Y) plots the data sequence, Y, as stems that extend from a baseline along the *x*-axis. The data values are indicated by circles terminating each .

Plot: 2-D line Plot

Syntax: plot(X, Y)

Plot(X, Y, Line Spec)

Description: plot (X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.

Subplot: Create axes in tiled positions

Syntax: subplot (m, n, p)
Subplot (m, n, p,'replace')

Description: subplot (m, n, p) divides the current figure into an m-by-n grid and creates axes in the position specified by p. MATLAB® numbers subplot positions by row. The first subplot is the first column of the first row; the second subplot is the second column of the first row, and so on. If axes exist in the specified position, then this command makes the axes the current axes.

Length: of largest array dimension

Syntax: L = Length (X)

Description = Length (X) returns the Length of the largest array dimension in X. For vectors, the Length is simply the number of elements. For arrays with more dimensions, the Length is `max (size(X))`. The Length of an empty array is zero.

Xlabel: Label x-axis

Syntax: Xlabel (txt)

Description: Xlabel (txt) labels the *x*-axis of the current axes or chart returned by the gca command. Reissuing the Xlabel command replaces the old label with the new label.

Ylabel: Label y-axis

Syntax: Ylabel (txt)

Description: Ylabel (txt) labels the *y*-axis of the current axes or chart returned by the gca command. Reissuing the Ylabel command causes the new label to replace the old label.

Title: Add Title

Syntax: Title (txt)

Description: Title (txt) adds the specified Title to the axes or chart returned by the gca command. Reissuing the Title command causes the new Title to replace the old Title.

Sin: Sine of argument in radians title

Syntax: Y = sin(X)

Description:Y = sin(X) returns the sine of the elements of X. The sin function operates element-wise on arrays. The function accepts both real and complex inputs. For real values of X in the interval [-Inf, Inf], sin returns real values in the interval [-1, 1]. For complex values of X, sin returns complex values. All angles are in radians.

Cos: Cosine of argument in radians

Syntax: Y = cos(X)

Description: Y = cos(X) returns the cosine for each element of X. The cos function operates element-wise on arrays. The function accepts both real and complex inputs. For purely real values or imaginary values of X, cos returns real values in the interval [-1 ,1]. For complex values of X, cos returns complex values. All angles are in radians.

Ramp: Generate constantly increasing or decreasing signal.

Description: The Ramp block generates a signal that starts at a specified time and value and changes by a specified rate. The block's Slope, Start time, and Initial output parameters determine the characteristics of the output signal. All must have the same dimensions after scalar expansion.

WRITE A MATLAB CODE TO GENERATE SINE WAVE

Continuous form:

```
clc;
f=input ('enter the input frequency');
t=0:0.01:1;
y=sin (2*pi*f*t);
disp ('the value of y=')
disp(y)
plot (t,y)
xlabel('time');
ylabel('amplitude');
title('sine wave');
```

OUTPUT:

```
Enter the input frequency1
The value of y=
  Columns 1 through 8

       0    0.0628    0.1253    0.1874    0.2487    0.3090    0.3681    0.4258

  Columns 9 through 16

    0.4818    0.5358    0.5878    0.6374    0.6845    0.7290    0.7705    0.8090

  Columns 17 through 24

    0.8443    0.8763    0.9048    0.9298    0.9511    0.9686    0.9823    0.9921

  Columns 25 through 32

    0.9980    1.0000    0.9980    0.9921    0.9823    0.9686    0.9511    0.9298

  Columns 33 through 40

    0.9048    0.8763    0.8443    0.8090    0.7705    0.7290    0.6845    0.6374

  Columns 41 through 48
```

0.5878 0.5358 0.4818 0.4258 0.3681 0.3090 0.2487 0.1874

Columns 49 through 56

0.1253 0.0628 0.0000 -0.0628 -0.1253 -0.1874 -0.2487 -0.3090

Columns 57 through 64

-0.3681 -0.4258 -0.4818 -0.5358 -0.5878 -0.6374 -0.6845 -0.7290

Columns 65 through 72

-0.7705 -0.8090 -0.8443 -0.8763 -0.9048 -0.9298 -0.9511 -0.9686

Columns 73 through 80

-0.9823 -0.9921 -0.9980 -1.0000 -0.9980 -0.9921 -0.9823 -0.9686

Columns 81 through 88

-0.9511 -0.9298 -0.9048 -0.8763 -0.8443 -0.8090 -0.7705 -0.7290

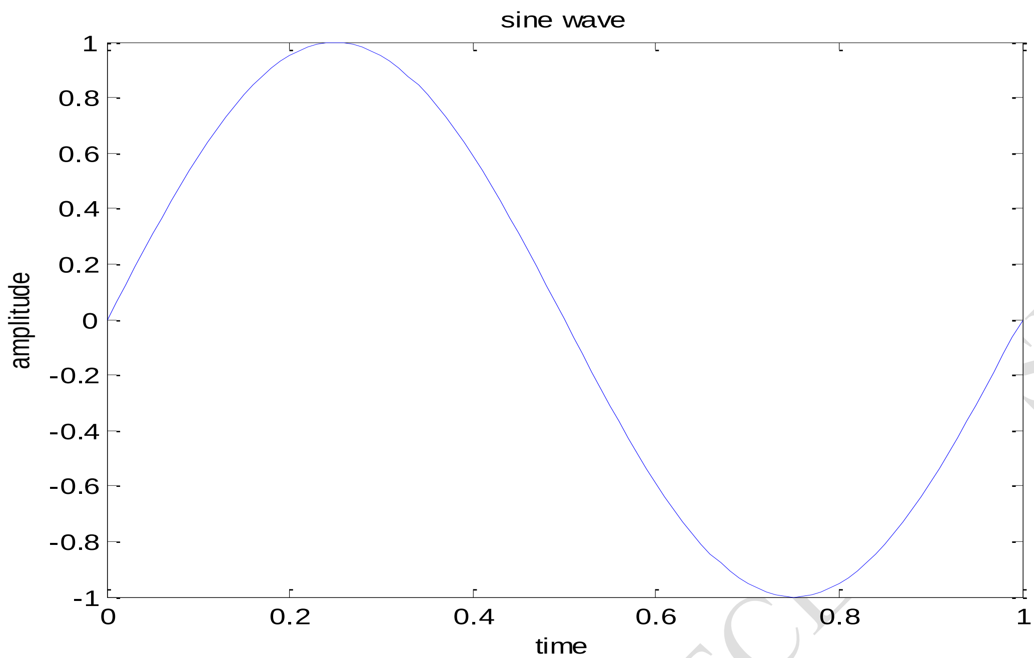
Columns 89 through 96

-0.6845 -0.6374 -0.5878 -0.5358 -0.4818 -0.4258 -0.3681 -0.3090

Columns 97 through 101

-0.2487 -0.1874 -0.1253 -0.0628 -0.0000

Graph:



Discrete form

```
clc;
f=input('enter the input frequency');
n=0:0.1:1;
y=sin(2*pi*f*n);
disp('the value of y=')
disp(y)
stem (n,y)
xlabel('time');
ylabel('amplitude');
title ('sine wave');
```

OUTPUT:

Enter the input frequency10

The value of y=

1.0e-14 *

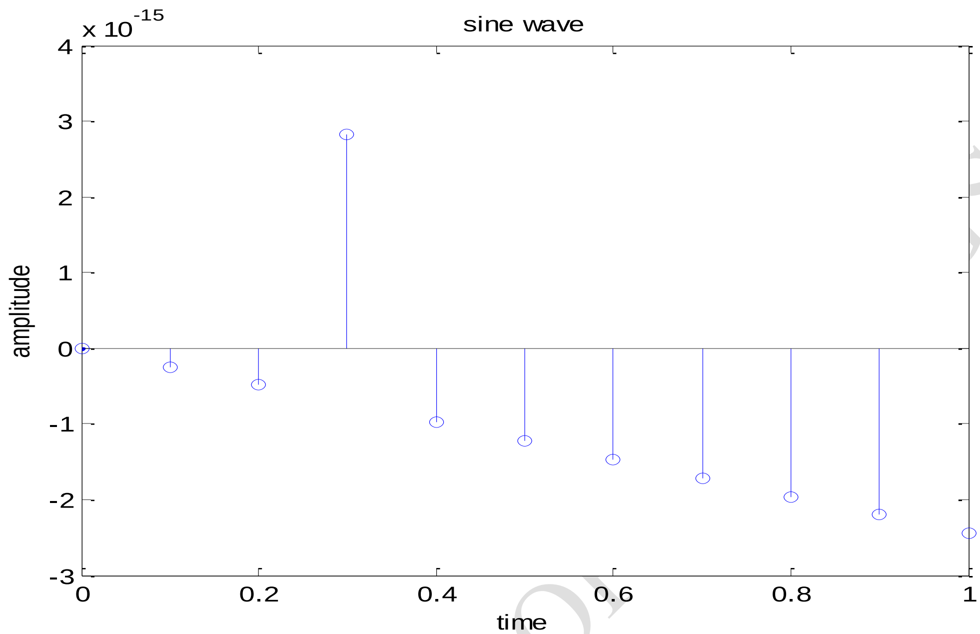
Columns 1 through 8

0 -0.0245 -0.0490 0.2818 -0.0980 -0.1225 -0.1470 -0.1715

Columns 9 through 11

-0.1959 -0.2204 -0.2449

Graph:



WRITE A MATLAB CODE TO GENERATE COSINE WAVE

Continuous form

```
clc;
f=input('enter the input frequency');
t=0:0.01:.1;
y=cos(2*pi*f*t);
disp('the value of y=')
disp(y)
plot(t,y)
xlabel('time');
ylabel('amplitude');
title('cosine wave');
```

OUTPUT:

Enter the input frequency1

The value of y=

```
Columns 1 through 8

  1.0000    0.9980    0.9921    0.9823    0.9686    0.9511    0.9298    0.9048

Columns 9 through 16

  0.8763    0.8443    0.8090    0.7705    0.7290    0.6845    0.6374    0.5878

Columns 17 through 24

  0.5358    0.4818    0.4258    0.3681    0.3090    0.2487    0.1874    0.1253

Columns 25 through 32

  0.0628    0.0000   -0.0628   -0.1253   -0.1874   -0.2487   -0.3090   -0.3681

Columns 33 through 40

 -0.4258   -0.4818   -0.5358   -0.5878   -0.6374   -0.6845   -0.7290   -0.7705

Columns 41 through 48

 -0.8090   -0.8443   -0.8763   -0.9048   -0.9298   -0.9511   -0.9686   -0.9823

Columns 49 through 56

 -0.9921   -0.9980   -1.0000   -0.9980   -0.9921   -0.9823   -0.9686   -0.9511

Columns 57 through 64

 -0.9298   -0.9048   -0.8763   -0.8443   -0.8090   -0.7705   -0.7290   -0.6845

Columns 65 through 72

 -0.6374   -0.5878   -0.5358   -0.4818   -0.4258   -0.3681   -0.3090   -0.2487

Columns 73 through 80

 -0.1874   -0.1253   -0.0628   -0.0000    0.0628    0.1253    0.1874    0.2487

Columns 81 through 88

  0.3090    0.3681    0.4258    0.4818    0.5358    0.5878    0.6374    0.6845
```

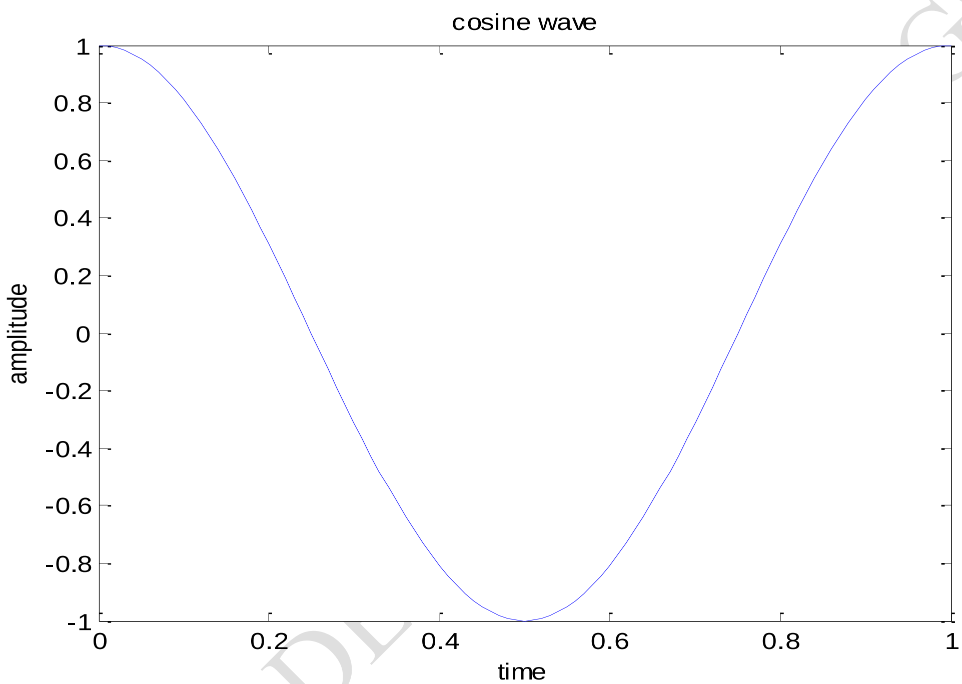
Columns 89 through 96

0.7290 0.7705 0.8090 0.8443 0.8763 0.9048 0.9298 0.9511

Columns 97 through 101

0.9686 0.9823 0.9921 0.9980 1.0000

Graph:



Discrete form

```
clc;
f=input('enter the input frequency');
n=0:0.01:.1;
y=cos(2*pi*f*n);
disp('the value of y=')
disp(y)
stem(n,y)
xlabel('time');
ylabel('amplitude');
title('cosine wave');
```

OUTPUT:

Enter the input frequency1

The value of y=

```
  Columns 1 through 8

    1.0000    0.9980    0.9921    0.9823    0.9686    0.9511    0.9298    0.9048

  Columns 9 through 16

    0.8763    0.8443    0.8090    0.7705    0.7290    0.6845    0.6374    0.5878

  Columns 17 through 24

    0.5358    0.4818    0.4258    0.3681    0.3090    0.2487    0.1874    0.1253

  Columns 25 through 32

    0.0628    0.0000   -0.0628   -0.1253   -0.1874   -0.2487   -0.3090   -0.3681

  Columns 33 through 40

   -0.4258   -0.4818   -0.5358   -0.5878   -0.6374   -0.6845   -0.7290   -0.7705

  Columns 41 through 48

   -0.8090   -0.8443   -0.8763   -0.9048   -0.9298   -0.9511   -0.9686   -0.9823

  Columns 49 through 56

   -0.9921   -0.9980   -1.0000   -0.9980   -0.9921   -0.9823   -0.9686   -0.9511

  Columns 57 through 64

   -0.9298   -0.9048   -0.8763   -0.8443   -0.8090   -0.7705   -0.7290   -0.6845

  Columns 65 through 72

   -0.6374   -0.5878   -0.5358   -0.4818   -0.4258   -0.3681   -0.3090   -0.2487

  Columns 73 through 80

   -0.1874   -0.1253   -0.0628   -0.0000    0.0628    0.1253    0.1874    0.2487
```

Columns 81 through 88

0.3090 0.3681 0.4258 0.4818 0.5358 0.5878 0.6374 0.6845

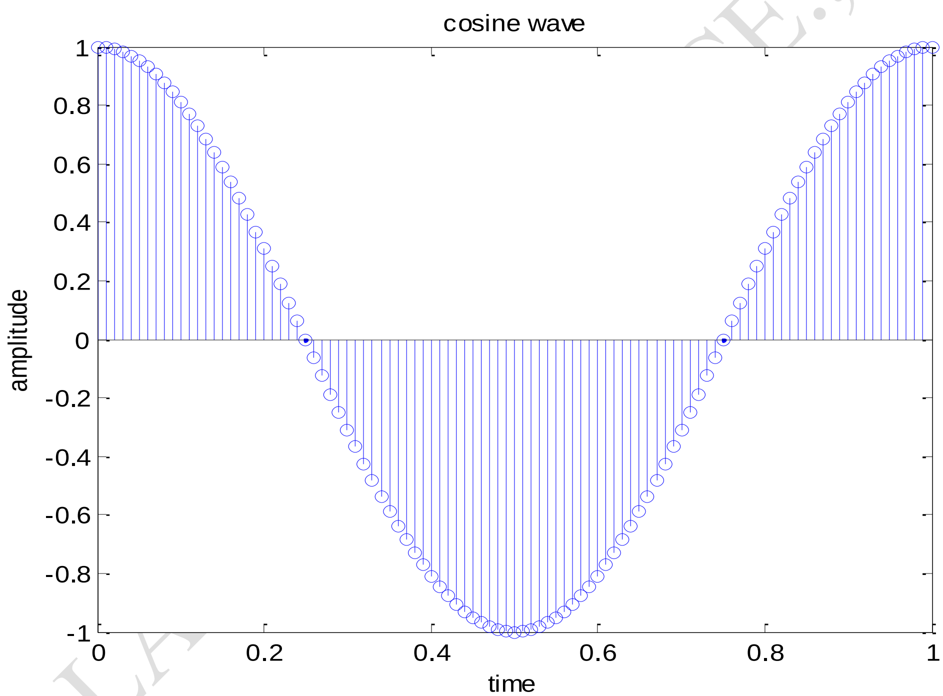
Columns 89 through 96

0.7290 0.7705 0.8090 0.8443 0.8763 0.9048 0.9298 0.9511

Columns 97 through 101

0.9686 0.9823 0.9921 0.9980 1.0000

Graph:



WIRTE A MATLAB CODE TO GENERATE RAMP WAVE

Continuous form

```
clc;
t=0:0.1:1;
y=t;
disp('the value of y=')
```

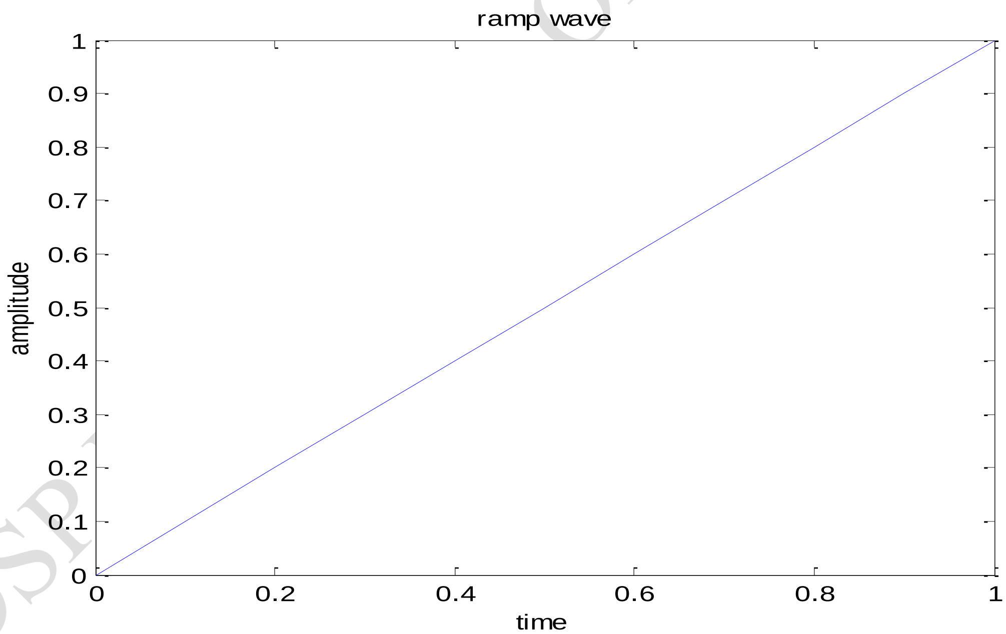
```
disp(y)
plot(t,y)
xlabel('time');
ylabel('amplitude');
title('ramp wave');
```

OUTPUT:

The value of y=

```
  Columns 1 through 8

    0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000

  Columns 9 through 11

    0.8000    0.9000    1.0000
```

Graph:



Discrete form

```
clc;
n=0:0.1:1;
```

```
y=n;
disp('the value of y=')
disp(y)
stem(n,y)
xlabel('time');
ylabel('amplitude');
title('ramp wave');
```

OUTPUT:

The value of y=

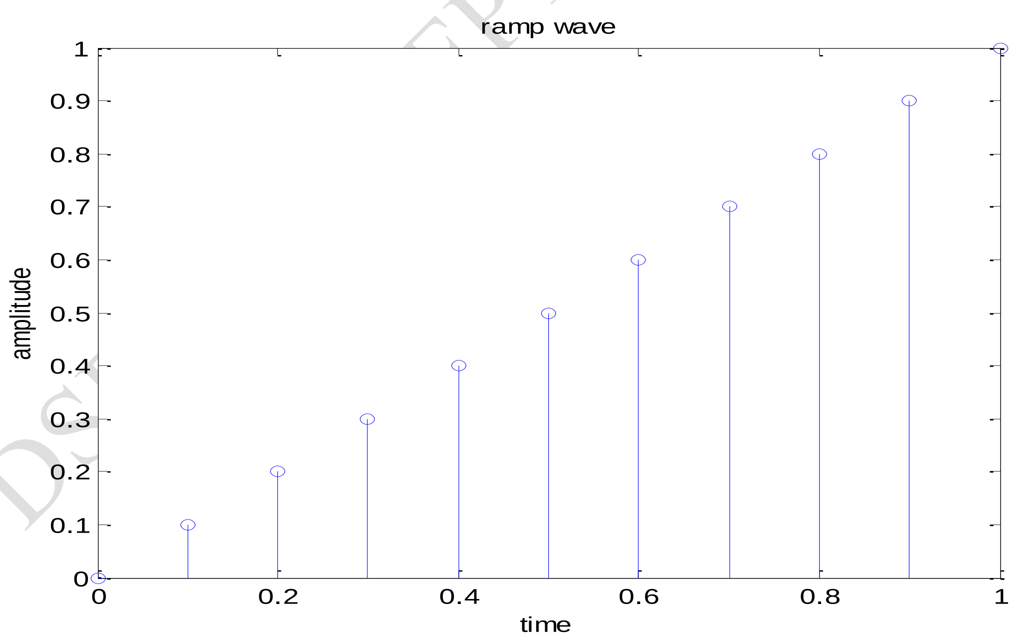
Columns 1 through 8

0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000

Columns 9 through 11

0.8000 0.9000 1.0000

Graph:



1- VERIFICATION OF SAMPLING THEOREM.

WRITE A MATLAB CODE TO VERIFY SAMPLING THEOREM

AIM: Verification of sampling theorem.

OBJECTIVE: To verify sampling theorem for Nyquist rate, under sampling and Over Sampling condition in both time and frequency domain using MATLAB.

THEORY:

Sample: is a piece of data taken from the whole data which is continuous in the time domain.

Sampling: Sampling is defined as, "The process of measuring the instantaneous values of continuous-time signal in a discrete form."

Sampling in dsp: In signal processing, sampling is the reduction of a continuous-time signal to a discrete-time signal. A common example is the conversion of a sound wave (a continuous signal) to a sequence of samples (a discrete-time signal). A sample is a value or set of values at a point in time and/or space.

Sampling theorem : A continuous time signal can be represented in its samples and can be recovered back when sampling frequency f_s is greater than or equal to the twice the highest frequency component of message signal. i. e.

$$fs \geq 2fm.$$

Nyquist Rate Sampling: The Nyquist rate is the minimum sampling rate required to avoid aliasing, equal to the highest modulating frequency contained within the signal. In otherwords, Nyquist rate is equal to two sided bandwidth of the signal (Upper and lower Sidebands) i.e., fs = 2W.To avoid aliasing, the sampling rate must exceed the Nyquist rate.

i.e. fs>fN, where fN =2W.

PROGRAM:

```
clc;
f=input('enter the input frequency=');
t=0:0.001:0.1;
y=cos(2*pi*f*t);
disp('the value of y are=')
disp(y)
subplot(3,1,1)
plot(t,y)
xlabel('time');
ylabel('amplitude');
title('cosine wave');
fs=input('enter the sampling frequency=');
ts=1/fs;
tn=0:ts:0.1;
ys=cos(2*pi*fs*tn);
disp('the value of ys are=')
disp(ys)
subplot(3,1,2)
plot(tn,ys)
xlabel('time');
ylabel('amplitude');
title('continuous wave');
subplot(3,1,3)
stem(tn,ys)
xlabel('time');
ylabel('amplitude');
title('discrete wave');
```

OUTPUT:

NYQUIST RATE

```
Enter the input frequency=100

The value of y are=

  Columns 1 through 9
    1.0000    0.8090    0.3090   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090    0.3090

  Columns 10 through 18
    0.8090    1.0000    0.8090    0.3090   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090

  Columns 19 through 27
    0.3090    0.8090    1.0000    0.8090    0.3090   -0.3090   -0.8090   -1.0000   -0.8090

  Columns 28 through 36
   -0.3090    0.3090    0.8090    1.0000    0.8090    0.3090   -0.3090   -0.8090   -1.0000

  Columns 37 through 45
   -0.8090   -0.3090    0.3090    0.8090    1.0000    0.8090    0.3090   -0.3090   -0.8090

  Columns 46 through 54
   -1.0000   -0.8090   -0.3090    0.3090    0.8090    1.0000    0.8090    0.3090   -0.3090

  Columns 55 through 63
   -0.8090   -1.0000   -0.8090   -0.3090    0.3090    0.8090    1.0000    0.8090    0.3090

  Columns 64 through 72
   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090    0.3090    0.8090    1.0000    0.8090

  Columns 73 through 81
    0.3090   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090    0.3090    0.8090    1.0000

  Columns 82 through 90
    0.8090    0.3090   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090    0.3090    0.8090

  Columns 91 through 99
    1.0000    0.8090    0.3090   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090    0.3090

  Columns 100 through 101
    0.8090    1.0000

Enter the sampling frequency=200
```

```
The value of ys are=

 Columns 1 through 9
   1.0000    0.3090   -0.8090   -0.8090    0.3090    1.0000    0.3090   -0.8090   -0.8090

 Columns 10 through 18
   0.3090    1.0000    0.3090   -0.8090   -0.8090    0.3090    1.0000    0.3090   -0.8090

 Columns 19 through 27
  -0.8090    0.3090    1.0000    0.3090   -0.8090   -0.8090    0.3090    1.0000    0.3090

 Columns 28 through 36
  -0.8090   -0.8090    0.3090    1.0000    0.3090   -0.8090   -0.8090    0.3090    1.0000

 Columns 37 through 45
   0.3090   -0.8090   -0.8090    0.3090    1.0000    0.3090   -0.8090   -0.8090    0.3090

 Columns 46 through 54
   1.0000    0.3090   -0.8090   -0.8090    0.3090    1.0000    0.3090   -0.8090   -0.8090

 Columns 55 through 63
   0.3090    1.0000    0.3090   -0.8090   -0.8090    0.3090    1.0000    0.3090   -0.8090

 Columns 64 through 72
  -0.8090    0.3090    1.0000    0.3090   -0.8090   -0.8090    0.3090    1.0000    0.3090

 Columns 73 through 81
  -0.8090   -0.8090    0.3090    1.0000    0.3090   -0.8090   -0.8090    0.3090    1.0000

 Columns 82 through 90
   0.3090   -0.8090   -0.8090    0.3090    1.0000    0.3090   -0.8090   -0.8090    0.3090

 Columns 91 through 99
   1.0000    0.3090   -0.8090   -0.8090    0.3090    1.0000    0.3090   -0.8090   -0.8090

 Columns 100 through 101

   0.3090    1.0000
```

UNDER SAMPLING

```
Enter the input frequency=100

The value of y are=

 Columns 1 through 9
   1.0000    0.8090    0.3090   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090    0.3090

 Columns 10 through 18
```

```
    0.8090    1.0000    0.8090    0.3090   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090

  Columns 19 through 27
    0.3090    0.8090    1.0000    0.8090    0.3090   -0.3090   -0.8090   -1.0000   -0.8090

  Columns 28 through 36
   -0.3090    0.3090    0.8090    1.0000    0.8090    0.3090   -0.3090   -0.8090   -1.0000

Columns 37 through 45
   -0.8090   -0.3090    0.3090    0.8090    1.0000    0.8090    0.3090   -0.3090   -0.8090

 Columns 46 through 54
   -1.0000   -0.8090   -0.3090    0.3090    0.8090    1.0000    0.8090    0.3090   -0.3090

  Columns 55 through 63
   -0.8090   -1.0000   -0.8090   -0.3090    0.3090    0.8090    1.0000    0.8090    0.3090

  Columns 64 through 72
   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090    0.3090    0.8090    1.0000    0.8090

  Columns 73 through 81
    0.3090   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090    0.3090    0.8090    1.0000

  Columns 82 through 90
    0.8090    0.3090   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090    0.3090    0.8090

  Columns 91 through 99
    1.0000    0.8090    0.3090   -0.3090   -0.8090   -1.0000   -0.8090   -0.3090    0.3090

  Columns 100 through 101
    0.8090    1.0000

Enter the sampling frequency=150

The values of ys are=

  Columns 1 through 9

    1.0000    0.5878   -0.3090   -0.9511   -0.8090   -0.0000    0.8090    0.9511    0.3090

  Columns 10 through 18
   -0.5878   -1.0000   -0.5878    0.3090    0.9511    0.8090    0.0000   -0.8090   -0.9511

  Columns 19 through 27
   -0.3090    0.5878    1.0000    0.5878   -0.3090   -0.9511   -0.8090    0.0000    0.8090
```

```
 Columns 28 through 36
  0.9511   0.3090  -0.5878  -1.0000  -0.5878   0.3090   0.9511   0.8090  -0.0000

 Columns 37 through 45
 -0.8090  -0.9511  -0.3090   0.5878   1.0000   0.5878  -0.3090  -0.9511  -0.8090

 Columns 46 through 54
 -0.0000   0.8090   0.9511   0.3090  -0.5878  -1.0000  -0.5878   0.3090   0.9511

 Columns 55 through 63
  0.8090  -0.0000  -0.8090  -0.9511  -0.3090   0.5878   1.0000   0.5878  -0.3090

 Columns 64 through 72
 -0.9511  -0.8090  -0.0000   0.8090   0.9511   0.3090  -0.5878  -1.0000  -0.5878

 Columns 73 through 81
  0.3090   0.9511   0.8090  -0.0000  -0.8090  -0.9511  -0.3090   0.5878   1.0000

 Columns 82 through 90
  0.5878  -0.3090  -0.9511  -0.8090  -0.0000   0.8090   0.9511   0.3090  -0.5878

 Columns 91 through 99
 -1.0000  -0.5878   0.3090   0.9511   0.8090   0.0000  -0.8090  -0.9511  -0.3090

 Columns 100 through 101
  0.5878   1.0000
```

OVER SAMPLING

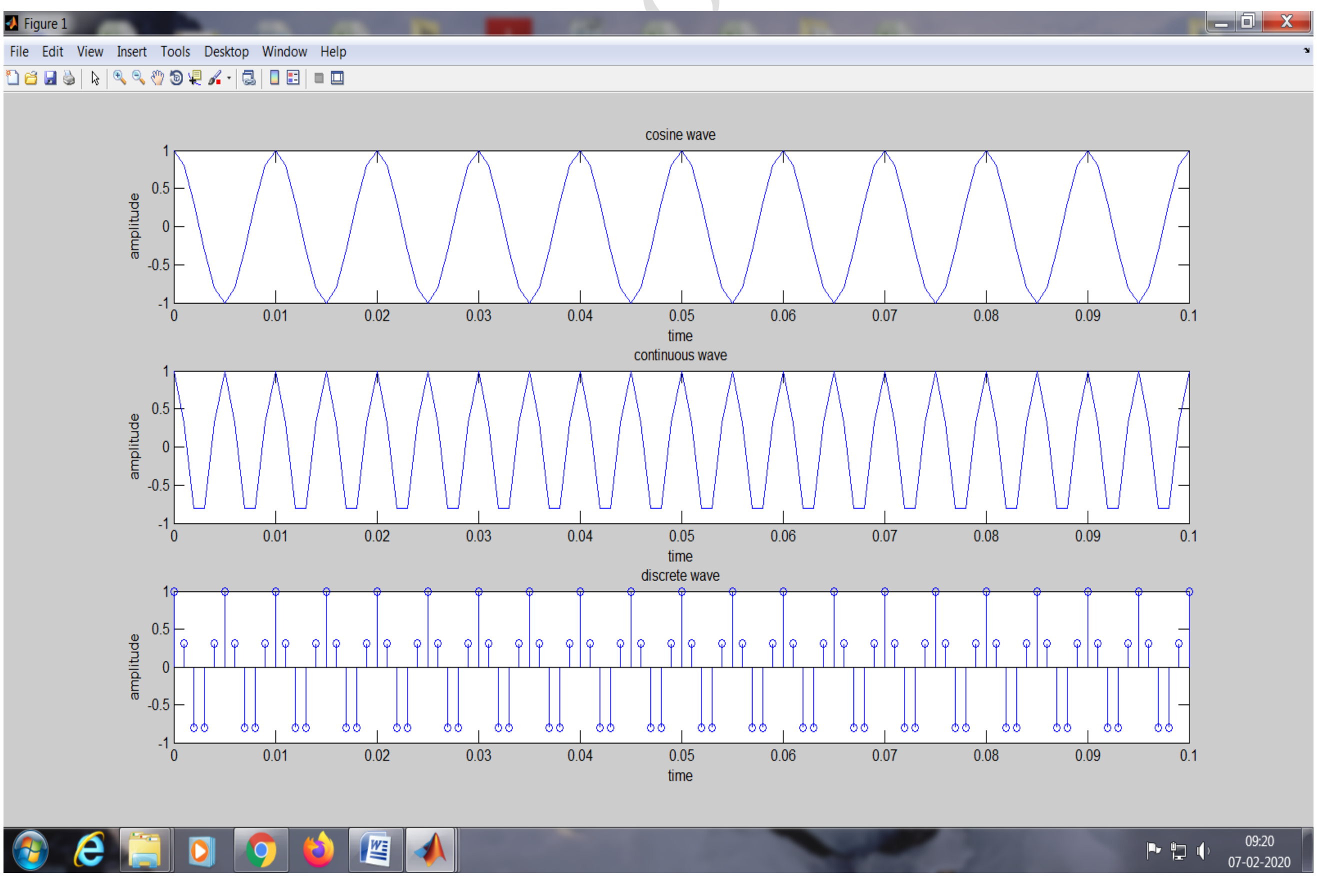
```
Enter the input frequency=100

The value of y are=
 Columns 1 through 9
  1.0000   0.8090   0.3090  -0.3090  -0.8090  -1.0000  -0.8090  -0.3090   0.3090

 Columns 10 through 18
  0.8090   1.0000   0.8090   0.3090  -0.3090  -0.8090  -1.0000  -0.8090  -0.3090

 Columns 19 through 27
  0.3090   0.8090   1.0000   0.8090   0.3090  -0.3090  -0.8090  -1.0000  -0.8090

 Columns 28 through 36
 -0.3090   0.3090   0.8090   1.0000   0.8090   0.3090  -0.3090  -0.8090  -1.0000

 Columns 37 through 45
 -0.8090  -0.3090   0.3090   0.8090   1.0000   0.8090   0.3090  -0.3090  -0.8090

 Columns 46 through 54
```

```
 -1.0000  -0.8090  -0.3090   0.3090   0.8090   1.0000   0.8090   0.3090  -0.3090

 Columns 55 through 63
 -0.8090  -1.0000  -0.8090  -0.3090   0.3090   0.8090   1.0000   0.8090   0.3090

 Columns 64 through 72
 -0.3090  -0.8090  -1.0000  -0.8090  -0.3090   0.3090   0.8090   1.0000   0.8090

 Columns 73 through 81
  0.3090  -0.3090  -0.8090  -1.0000  -0.8090  -0.3090   0.3090   0.8090   1.0000

 Columns 82 through 90
  0.8090   0.3090  -0.3090  -0.8090  -1.0000  -0.8090  -0.3090   0.3090   0.8090


 Columns 91 through 99
  1.0000   0.8090   0.3090  -0.3090  -0.8090  -1.0000  -0.8090  -0.3090   0.3090

 Columns 100 through 101
  0.8090   1.0000

Enter the sampling frequency=230

The value of ys are=

 Columns 1 through 9
  1.0000   0.1253  -0.9686  -0.3681   0.8763   0.5878  -0.7290  -0.7705   0.5358

 Columns 10 through 18
  0.9048  -0.3090  -0.9823   0.0628   0.9980   0.1874  -0.9511  -0.4258   0.8443

 Columns 19 through 27
  0.6374  -0.6845  -0.8090   0.4818   0.9298  -0.2487  -0.9921   0.0000   0.9921

 Columns 28 through 36
  0.2487  -0.9298  -0.4818   0.8090   0.6845  -0.6374  -0.8443   0.4258   0.9511

 Columns 37 through 45
 -0.1874  -0.9980  -0.0628   0.9823   0.3090  -0.9048  -0.5358   0.7705   0.7290

 Columns 46 through 54
 -0.5878  -0.8763   0.3681   0.9686  -0.1253  -1.0000  -0.1253   0.9686   0.3681

 Columns 55 through 63
 -0.8763  -0.5878   0.7290   0.7705  -0.5358  -0.9048   0.3090   0.9823  -0.0628

 Columns 64 through 72
```

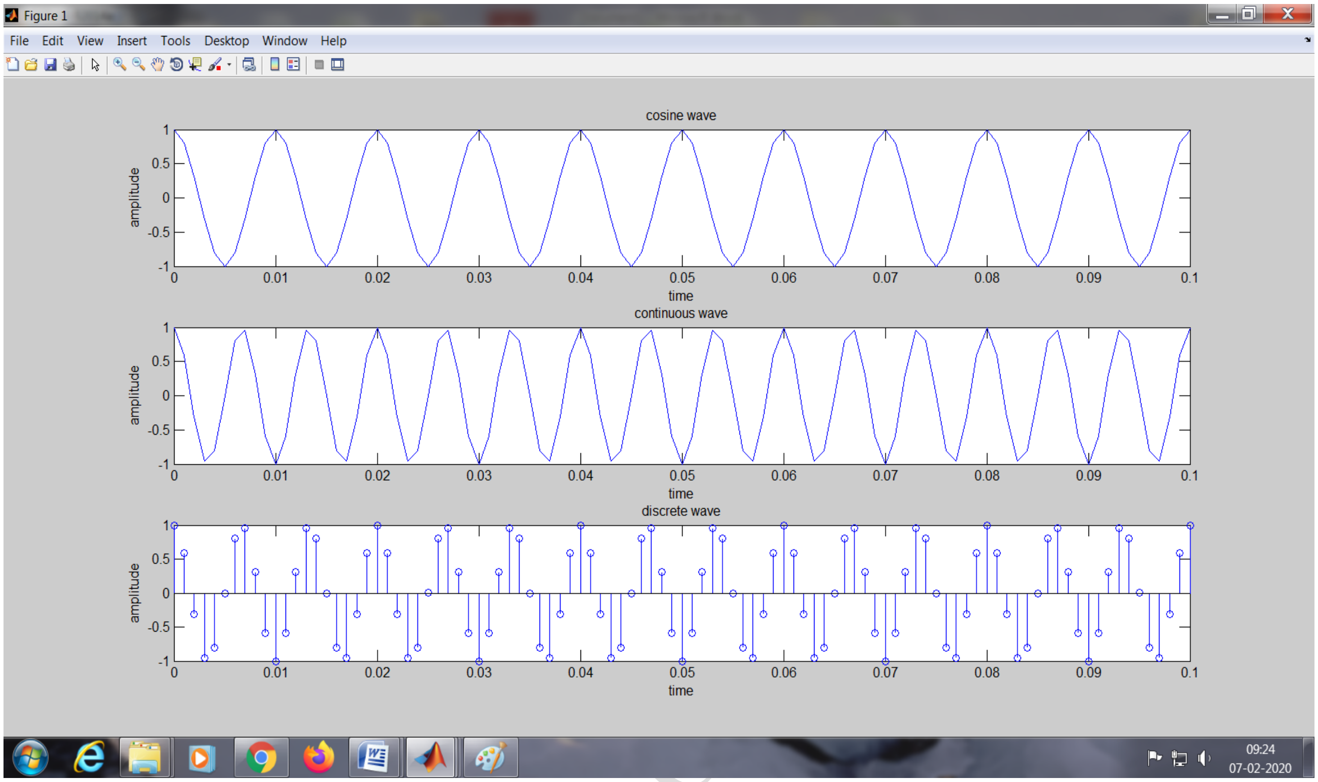
```
  -0.9980  -0.1874   0.9511   0.4258  -0.8443  -0.6374   0.6845   0.8090  -0.4818

 Columns 73 through 81
  -0.9298   0.2487   0.9921  -0.0000  -0.9921  -0.2487   0.9298   0.4818  -0.8090

 Columns 82 through 90
  -0.6845   0.6374   0.8443  -0.4258  -0.9511   0.1874   0.9980   0.0628  -0.9823

 Columns 91 through 99
  -0.3090   0.9048   0.5358  -0.7705  -0.7290   0.5878   0.8763  -0.3681  -0.9686

 Columns 100 through 101
   0.1253   1.0000
```


Graph:

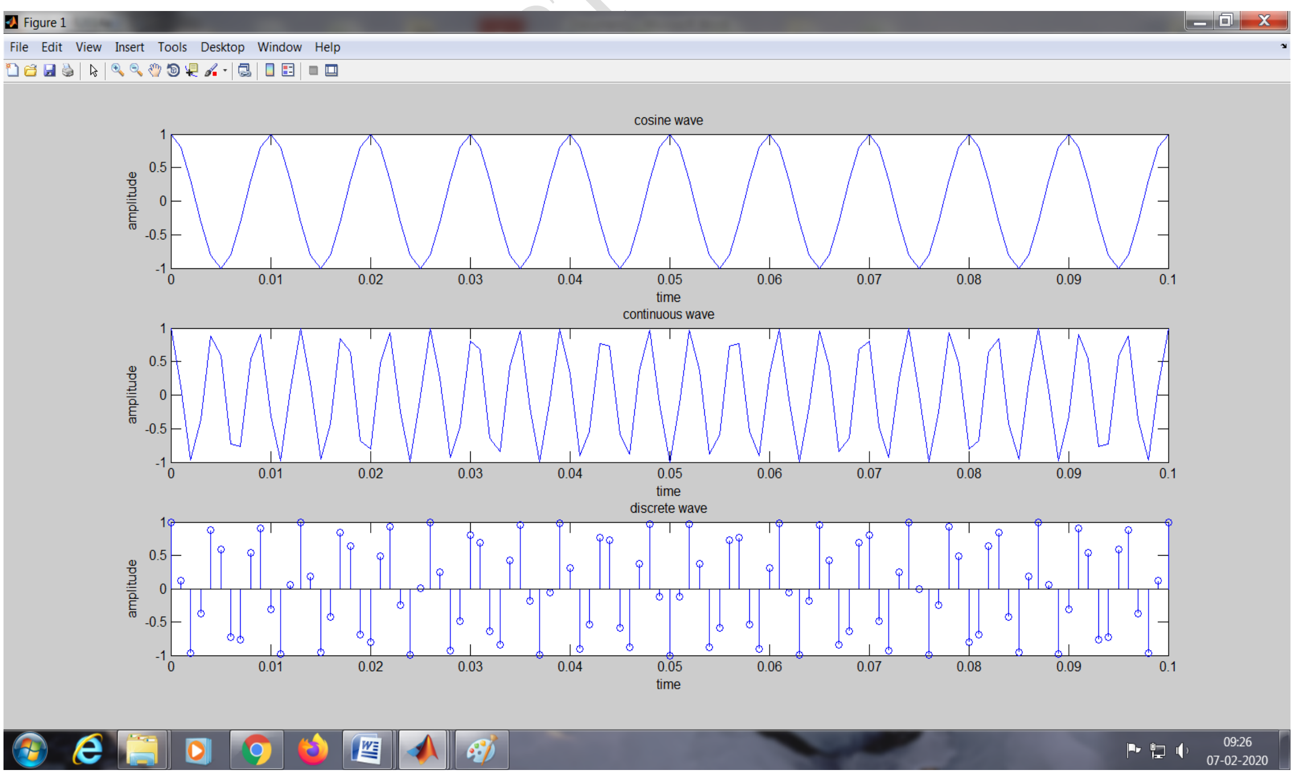
NYQUIST RATE



UNDER SAMPLING



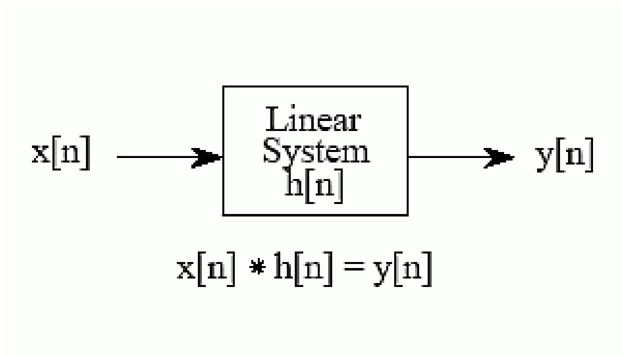
OVER SAMPLING



2- LINEAR AND CIRCULAR CONVOLUTION OF TWO GIVEN SEQUENCES, COMMUTATIVE, DISTRIBUTIVE AND ASSOCIATIVE PROPERTY OF CONVOLUTION.

Explanation:

Convolution: Convolution is a mathematical way of combining two signals to form a third signal. It is the single most important technique in Digital Signal Processing. Convolution is important because it relates the three signals of interest: the input signal, the output signal, and the impulse response.



Linear convolution: is the basic operation to calculate the output for any **linear** time invariant system given its input and its impulse response.

Circular convolution: is the same thing but considering that the support of the signal is periodic (as in a circle, hence the name).

Linear and circular convolutions: are fundamentally different operations. For two vectors, x and y, the circular convolution is equal to the inverse discrete Fourier transform (DFT) of the product of the vectors' DFTs. Knowing the conditions under which linear and circular convolution are equivalent allows you to use the DFT to efficiently compute linear convolutions.

The linear convolution of an N-point vector, x, and an L-point vector, y, has length $N + L - 1$.

For the circular convolution of x and y to be equivalent, you must pad the vectors with zeros to length at least $N + L - 1$ before you take the DFT. After you invert the product of the DFTs, retain only the first $N + L - 1$ element.

Comparison parameter	Linear convolution	Circular convolution
Shifting	Linear Shifting	Circular Shifting
Samples in the convolution result	N_1+N_2-1	Max (N_1, N_2)
Finding response of a filter	Possible	Possible with zero padding

WRITE A MATLAB CODE TO FIND THE LINEAR CONVOLUTION BETWEEN THE TWO SEQUENCE

Conv: Convolution and polynomial multiplication

Syntax: w = conv(u,v)

Description: w = conv(u,v) convolves vectors u and v. Algebraically, convolution is the same operation as multiplying the polynomials whose coefficients are the elements of u and v.

Equation: y (n) =x (n)*h (n)

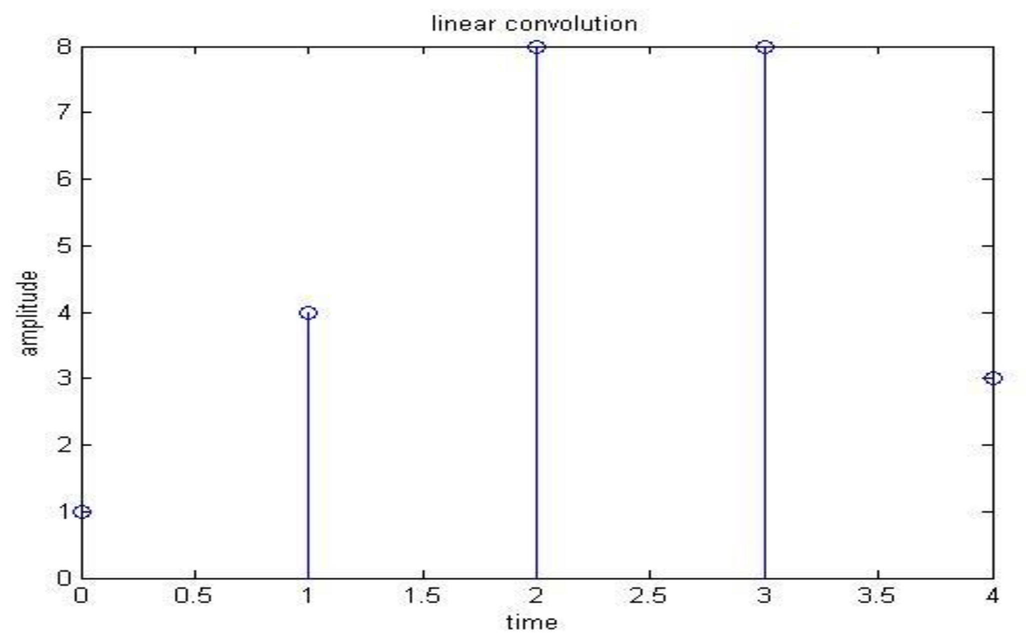
PROGRAM:

```
clc;
x=input ('enter the input sequence');
h=input ('enter the impulse response');
y=conv(x,h);
disp ('the value 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]
Enter the impulse response [1 2 1]
The values of y are=
   1    4    8    8    3
```

Graph:



WRITE A MATHLAB CODE TO VERIFY LINEAR CONVOLUTION IS COMMUTATIVE

Equation: x (n)*h (n) = h (n)*x (n)

PROGRAM:

```
clc;
x=input('enter the input sequence=');
h=input('enter the impulse response=');
y1=conv(x,h);
disp('the value of y1 are=')
disp(y1)
subplot(2,1,1)
n=0:length(y1)-1;
stem(n,y1)
xlabel('time');
ylabel('amplitude');
title('linear convolution1');
y2=conv(h,x);
disp('the value of y2 are=')
disp(y2)
subplot(2,1,2)
n=0:length(y2)-1;
stem(n,y2)
xlabel('time');
ylabel('amplitude');
title('linear convolution2');
if(y1==y2)
   disp('the linear convolution is commutative')
else
   disp('the linear convolution is not commutative')
end
```

OUTPUT:

Enter the input sequence= [1 2 3]

Enter the impulse response= [1 2 1]

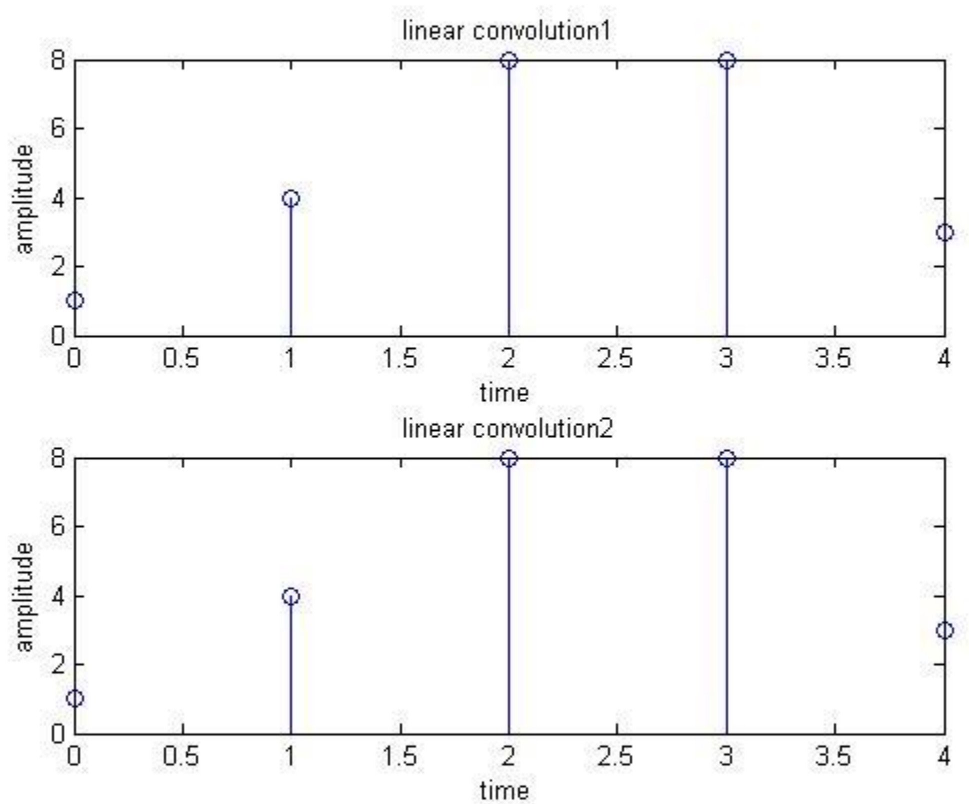
The values of y1 are=

    1   4   8   8   3

The values of y2 are=

    1   4   8   8   3

Graph:



WRITE A MATLAB CODE TO VERIFY LINEAR CONVOLUTION IS DISTRIBUTIVE

Equation: x (n)*[h1 (n) +h2 (n)] = x (n)*h1 (n) +x (n)*h2 (n)

PROGRAM:

```
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 value 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 value of y5 are=')
disp(y5)
subplot(2,1,2)
n=0:length(y5)-1;
stem(n,y5)
xlabel('time');
ylabel('amplitude');
title('linear convolution2');
if(y2==y5)
```

```
   disp('the linear convolution is distributive')

else
   disp('the linear convolution is not distributive')
end
```

OUTPUT:

Enter the input sequence=[1 2 3]

Enter the first impulse response=[1 2]

Enter the second impulse response=[1 1]

The value of y2 are=

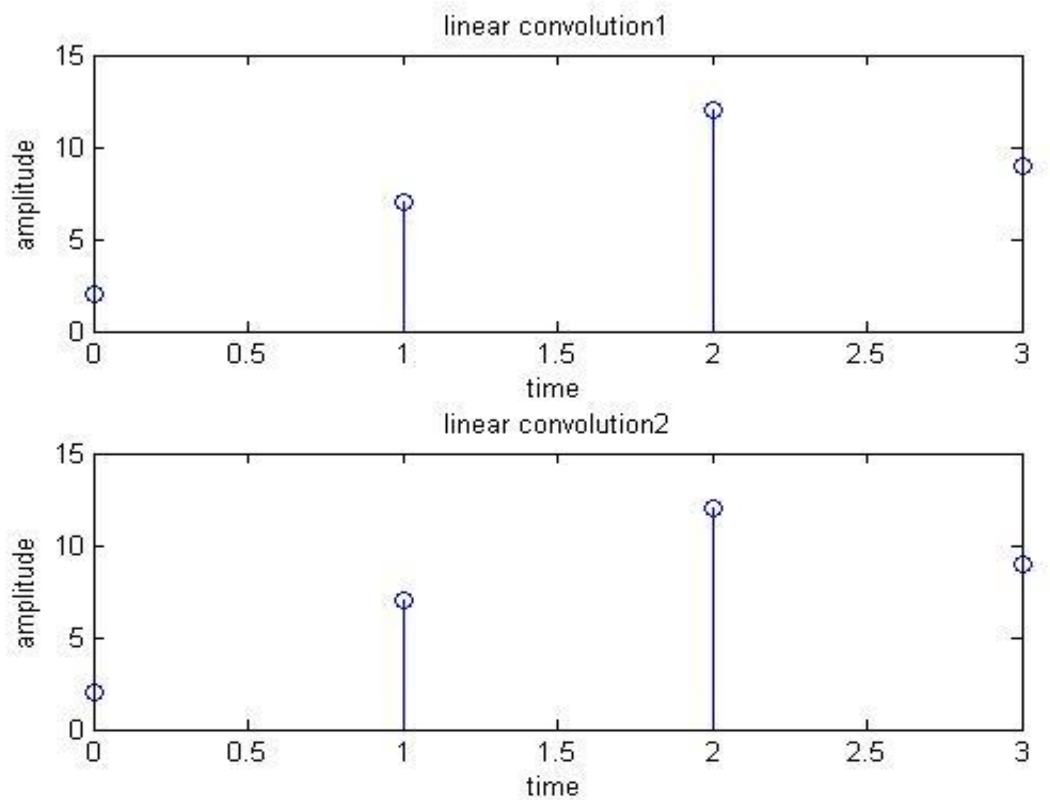
    2   7   12   9

The value of y5 are=

    2   7   12   9

The linear convolution is distributive

Graph:



WRITE A MATLAB CODE TO VERIFY LINEAR CONVOLUTION IS ASSOCIATIVE

Equation: x (n)*[h1 (n)*h2 (n)] = [x (n)*h1 (n)]*h2 (n)

PROGRAM:

```
clc;
x=input('enter the input sequence=');
h1=input('enter the first impulse response=');
h2=input('enter the second impulse response=');
y1=conv(h1,h2);
y2=conv(x,y1);
disp('the value 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(y3,h2);
disp('the value of y4 are=')
disp(y4)
subplot(2,1,2)
n=0:length(y4)-1;
stem(n,y4)
xlabel('time');
ylabel('amplitude');
title('linear convolution2');
if(y2==y4)
   disp('the linear convolution is associative')
else
```

```
    disp('the linear convolution is not associative')
end
```

OUTPUT:

Enter the input sequence= [1 2 3]

Enter the first impulse response= [1 2]

Enter the second impulse response= [1 1]

The value of y2 are=

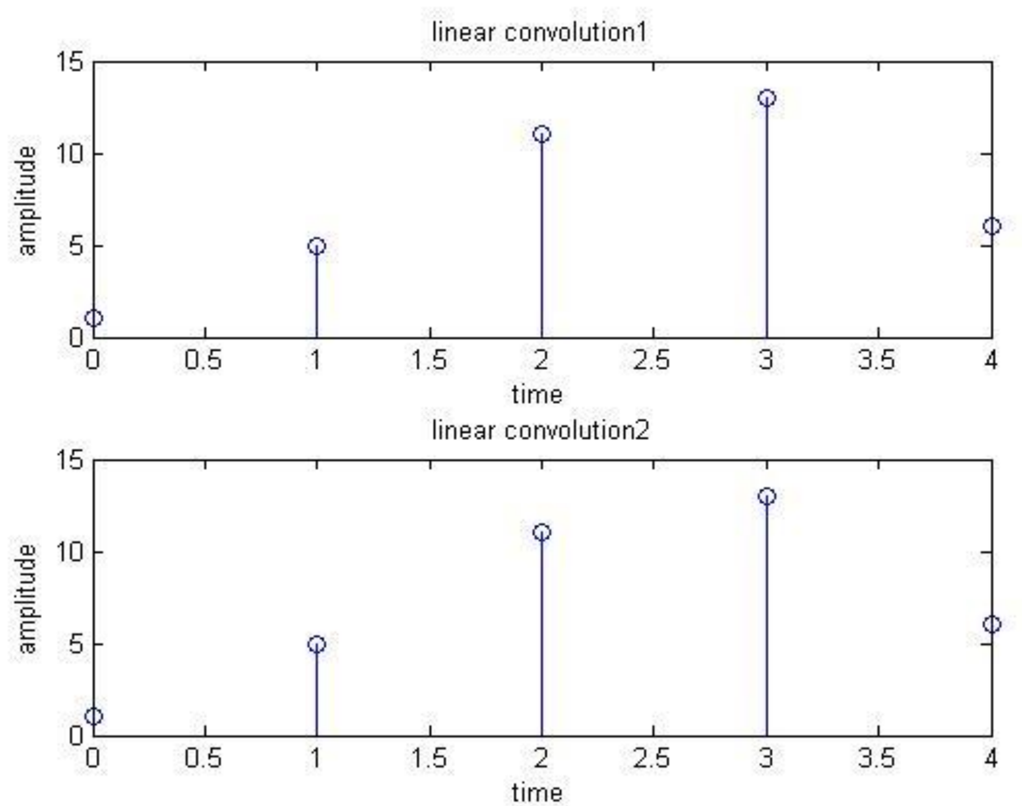
    1   5   11   13   6

The value of y4 are=

    1   5   11   13   6

The linear convolution is associative

Graph:



WRITE A MATLAB CODE TO FIND CIRCULAR CONVOLUTION BETWEEN TWO GIVEN SEQUENCES

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.

Equation: **y (n) =x (n) ⊕h (n)**

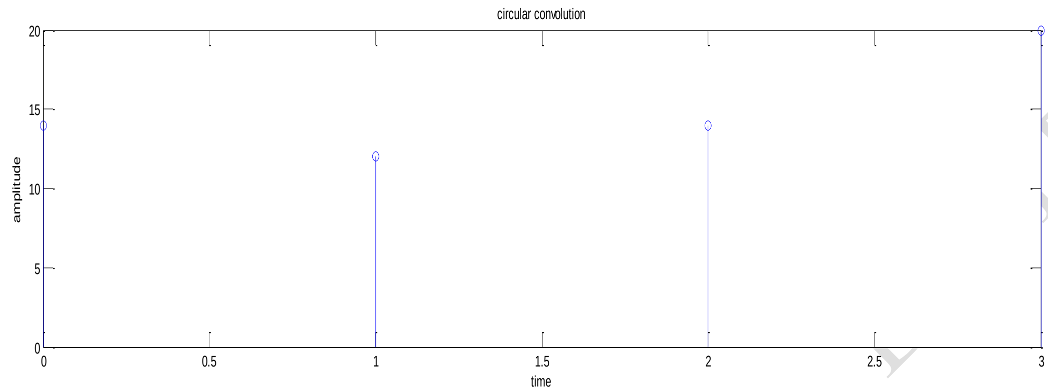
PROGRAM:

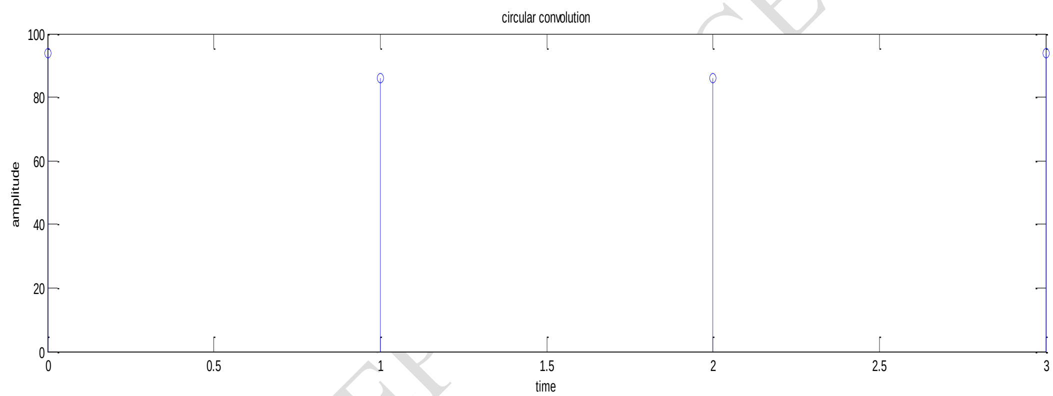
```
clc;
x=input ('enter the input sequence=');
h=input ('enter the input impulse responce=');
N=max (length(x), length (h));
y=cconv(x,h,N);
disp('the value of y are')
disp(y)
n=0: length(y)-1;
stem (n,y)
xlabel('time');
ylabel('amplitude');
title ('circular convolution');
```

OUTPUT:

```
Enter the input sequence= [1 2 3 4]
Enter the input impulse response= [3 2 1 0]
The values of y are
   14    12    14    20
```

Graph:


circular convolution
amplitude
time
0
5
10
15
20
0.5
1
1.5
2
2.5
3


circular convolution
amplitude
time
0
20
40
60
80
100
0.5
1
1.5
2
2.5
3

WRITE A MATLAB CODE TO VERIFY CIRCULAR CONVOLUTION IS COMMUTATIVE

Equation: x (n) ⊕ h (n) = h (n) ⊕ x (n)

PROGRAM:

```
clc;
x=input('enter the input sequence=');
h=input('enter the input impulse responce=');
N=max(length(x),length(h));
y1=cconv(x,h,N);
disp('the value of y1 are')
disp(y1)
subplot(3,1,1)
n=0:length(y1)-1;
stem(n,y1)
xlabel('time');
ylabel('amplitude');
title('circular convolution');
y2=cconv(h,x,N);
disp('the value of y2 are')
disp(y2)
subplot(2,1,2)
n=0:length(y2)-1;
stem(n,y2)
xlabel('time');
ylabel('amplitude');
title('circular convolution');
if(y1==y2)
disp('the circular convolution is commutative')
else
disp('the circular convolution is not commutative')
```

```
end
```

OUTPUT:

Enter the input sequence=[1 2 3 4]

Enter the input impulse responce=[3 2 1 0]

The value of y1 are

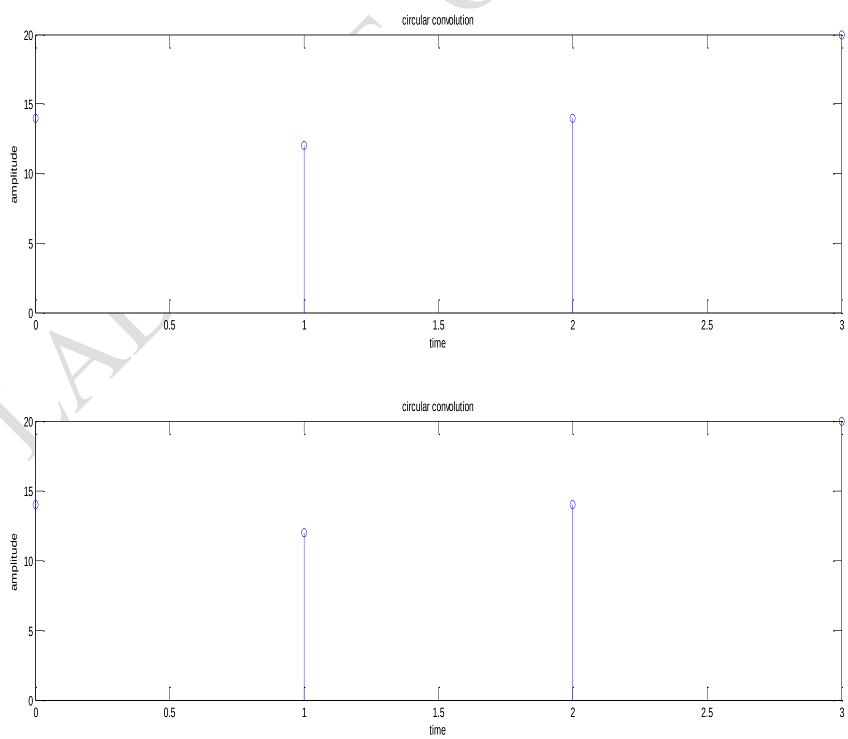
    14   12   14   20

The values of y2 are

    14   12   14   20

The circular convolution is commutative

Graph:



WRITE A MAT LAB CODE TO VERIFY CIRCULAR CONVOLUTION IS ASSOCIATIVE

Equation = x (n) ⊕[h1 (n) ⊕h2(n)]=[x(n) ⊕h1(n)] ⊕h2(n)

PROGRAM:

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

```
ylabel('amplitude');
title('circular convolution');
if(y2==y4)
disp('the circular convolution is associative')
else
disp('the circular convolution is not associative')
end
```

OUTPUT:

Enter the input sequence=[1 2 3 4]

Enter the input impulse response=[3 2 1]

Enter the second impulse response=[2 2 1 1]

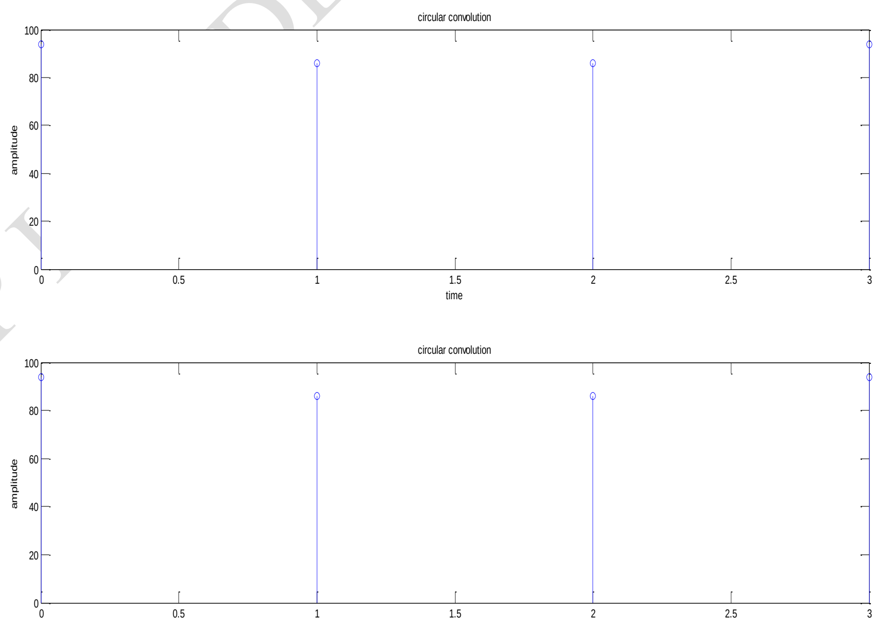
The values of y2 are

94 86 86 94

The value of y4 are
94 86 86 94

The circular convolution is associative

Graph:



WRITE A MAT LAB CODE TO VERIFY CIRCULAR CONVOLUTION IS DISTRIBUTIVE

Equation

X (n) ⊕[h1 (n) +h2(n)]=x(n) ⊕h1(n) + x(n) ⊕h2(n)

PROGRAM:

```
clc;
x=input ('enter the input sequence=');
h1=input ('enter the input impulse responce=');
h2=input ('enter the second impulse response=');
y1=h1+h2;
N1=max (length (x), length (y1));
Y2=cconv(x, y1, N1);
disp('the value of y2 are')
disp(y2)
subplot(2,1,1)
n=0:length(y2)-1;
stem(n,y2)
xlabel('time');
ylabel('amplitude');
title('circular convolution');
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 value of y4 are')
disp(y5)
subplot(2,1,2)
n=0:length(y5)-1;
```

```
stem(n,y5)
xlabel('time');
ylabel('amplitude');
title('circular convolution');
if(y2==y5)
disp('the circular convolution is distributive')
else
disp('the circular convolution is not distributive')
end
```

3- AUTO AND CROSS CORRELATION OF TWO SEQUENCES AND VERIFICATION OF THEIR PROPERTIES

THEORY:

Autocorrelation: also known as serial correlation is the correlation of a signal with a delayed copy of itself as a function of delay. Informally, it is the similarity between observations as a function of the time lag between them.

Cross-correlation: In signal processing, cross-correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them. The cross-correlation is similar in nature to the convolution of two functions.

Xcorr: Cross-correlation

Syntax:

c = xcorr(x,y)

c = xcorr(x)

Description:

- xcorr estimates the cross-correlation sequence of a random process. Autocorrelation is handled as a special case.
- c = xcorr(x,y) returns the cross-correlation sequence in a length 2N – 1 vector, where x and y are vectors of length N and N > 1. If x and y are not the same length, the shorter vector is zero-padded to the length of the longer vector.
- c = xcorr(x) is the autocorrelation sequence for the vector x.

WRITE A MATLAB CODE TO PERFORM CROSS CORRELATION OF GIVEN TWO SEQUENCES

AIM: Cross-correlation of a given sequences

OBJECTIVE:

1- To implement Cross correlation of the given sequence using an inbuilt MATLAB Function "XCORR"
2- To verify the results theoretically.

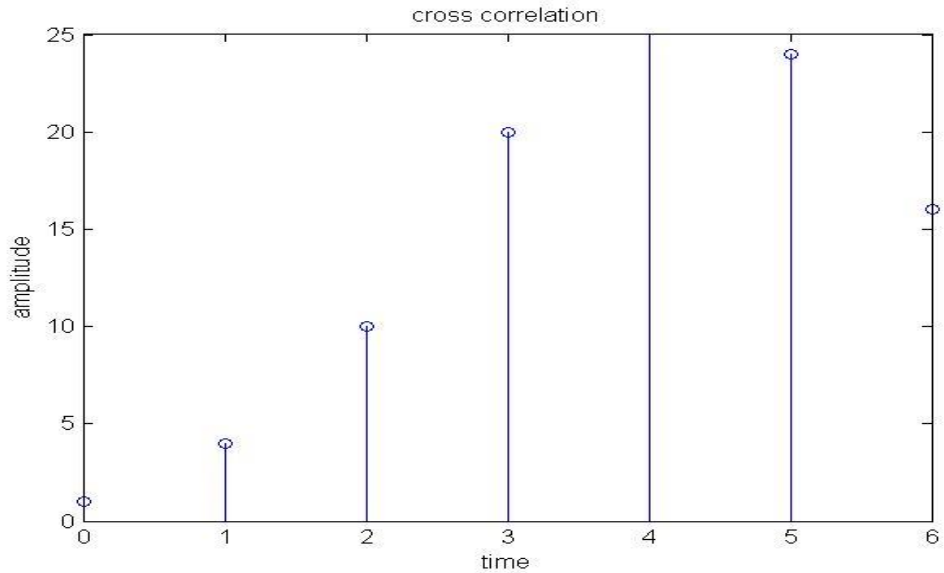
PROGRAM:

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

OUTPUT:

```
Enter the input sequence[1 2 3 4]
Enter the input sequence[4 3 2 1]
The value of y are
  1.0000   4.0000   10.0000   20.0000   25.0000   24.0000   16.0000
```

Graph:



WRITE A MATLAB CODE TO PERFORM AUTO CORRELATION OF GIVEN SEQUENCE

AIM: Autocorrelation of a given sequence

OBJECTIVE:

1- To find Autocorrelation of the given sequence using an inbuilt MATLAB function "XCORR".

2- To verify the results theoretically.

PROGRAM:

```
Clc;
a=input('enter the input sequence');
y=xcorr(a,a);
disp('the value of y are')
disp(y)
n=0:length(y)-1;
stem(n,y)
xlabel('time');
ylabel('amplitude');
title('auto correlation');
```

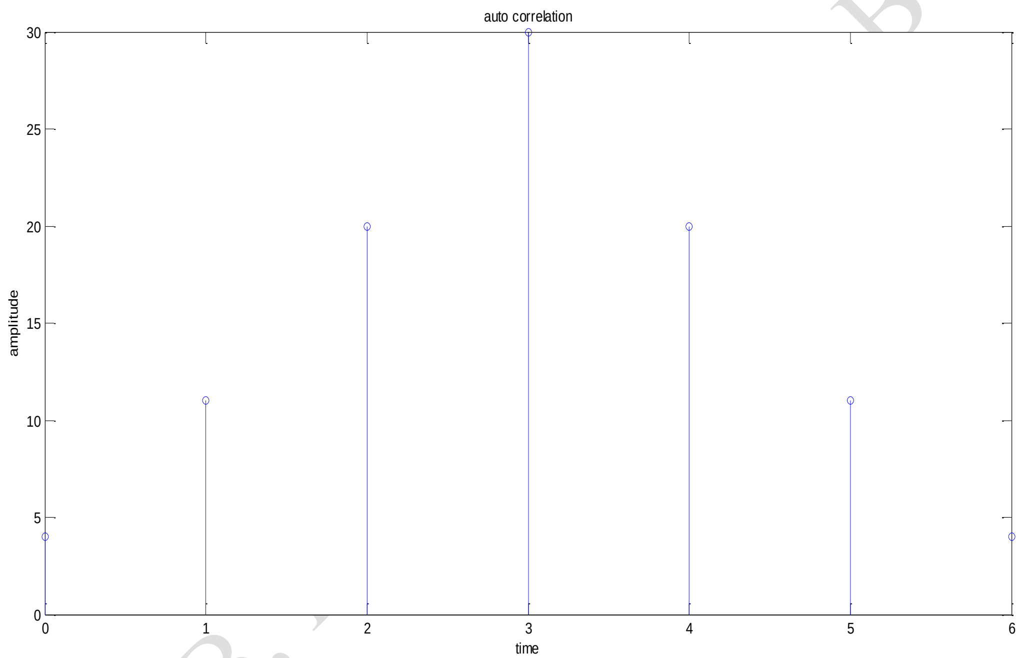
OUTPUT:

Enter the input sequence[1 2 3 4]

The value of y are

4.0000 11.0000 20.0000 30.0000 20.0000 11.0000 4.0000

Graph:



4-SOLVING A GIVEN DIFFERENCE EQUATION

AIM: To solve a given difference equation

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 coefficient vector b and denominator coefficient vector a. If a(1) is not equal to 1, filter normalizes the filter coefficients by a(1). If a(1) equals 0, filter returns an error.

WRITE A MATLAB CODE TO FIND STEP RESPONSE IN A GIVEN DIFFERENCE EQUATION

PROGRAM:

```
Clc
num=input('enter the numerator co-efficients=');
den=input('enter the denominator co-efficients=');
N=input('enter the number of samples=');
Step=ones(1,N)
y=filter(num,den,step);
disp('the value of y are=');
disp(y)
subplot(2,1,1)
n=0:length(step)-1;
stem(n, step)
xlabel('time');
ylabel('amplitude');
title('step function');
subplot(2,1,2)
n=0:length(y)-1;
```

```
step(n,y)
xlabel('time');
ylabel('amplitude');
title('step response');
```

WRITE A MATLAB CODE TO FIND IMPULSE RESPONSE IN A GIVEN DIFFERENCE EQUATION

PROGRAM:

```
Clc
num=input('enter the numerator co-efficients=');
den=input('enter the denominator co-efficients=');
N=input('enter the number of samples=');
Impulse=[ones(1,1), zeros(1,N-1)]
h=filter(num,den,impulse);
disp('the value of impulse function are=')
disp(h)
subplot(2,1,1)
n=0:length(impulse)-1;
stem(n,impulse)
xlabel('time')
ylabel('amplitude')
title('impulse function');
subplot(2,1,2)
n=0:length(h)-1;
stem(n,h)
xlabel('time');
ylabel('amplitude');
title('impulse response');
```

5- COMPUTATION OF N POINT DFT OF A GIVEN SEQUENCE AND TO PLOT MAGNITUDE AND PHASE SPECTRUM (USING DFT EQUATION AND VERIFY IT BY BUILT-IN ROUTINE)

AIM: To find Discrete Fourier Transform and Inverse Discrete Fourier Transform of given digital signal.

OBJECTIVE:

1. To find the N point DFT of a given sequence using the MATLAB inbuilt function "FFT" and to find Magnitude and Phase of DFT sequence using functions "ABS and ANGLE"
2. To verify the result theoretically.

THEORY:

Digital Signal Processing-Discrete Fourier Transform: As the name implies, the Discrete Fourier Transform (DFT) is purely discrete: discrete-time data sets are converted into a discrete-frequency representation. This is in contrast to the DTFT that uses discrete time, but converts to continuous frequency.

The Inverse Discrete Fourier Transform (IDFT): The inverse Fourier tranform maps the signal back from the frequency domain into the time domain. A time domain signal will usually consist of a set of real values, where each value has an associated time (e.g., the signal consists of a time series).

WRITE A MATLAB CODE TO FIND THE DFT FOR A GIVEN SEQUENCE AND PLOT MAGNITUDE AND SPECTRUM [USING BUILT IN ROUTINE].

DFT or FFT: Discrete Fourier transforms.

Syntax: x=fft (x, N)

Description: x=fft (x, N) returns the N-point DFT. If the length of x is less than x,N.

Angle: phase angle.

Syntax: p=angle(x)

Description: p=angle(x) returns the phase angles in routines for each elements of complex array X.

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 of the corresponding elements of x.

PROGRAM:

```
clc;
x=input('enter the input sequence=');
N=input('enter the number of dft points=');
X=fft(x,N);
disp('the value of X are=')
disp(X)
m=abs(X)
disp('the value of m 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 value of p are')
disp(p)
subplot(2,1,2)
n=0:length(p)-1;
stem(n,p)
xlabel('time');
ylabel('amplitude');
```

```
title('phase');
```

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 - 2.00

m =

10.0000 2.8284 2.0000 2.8284

The values of m are=

10.0000 2.8284 2.0000 2.8284

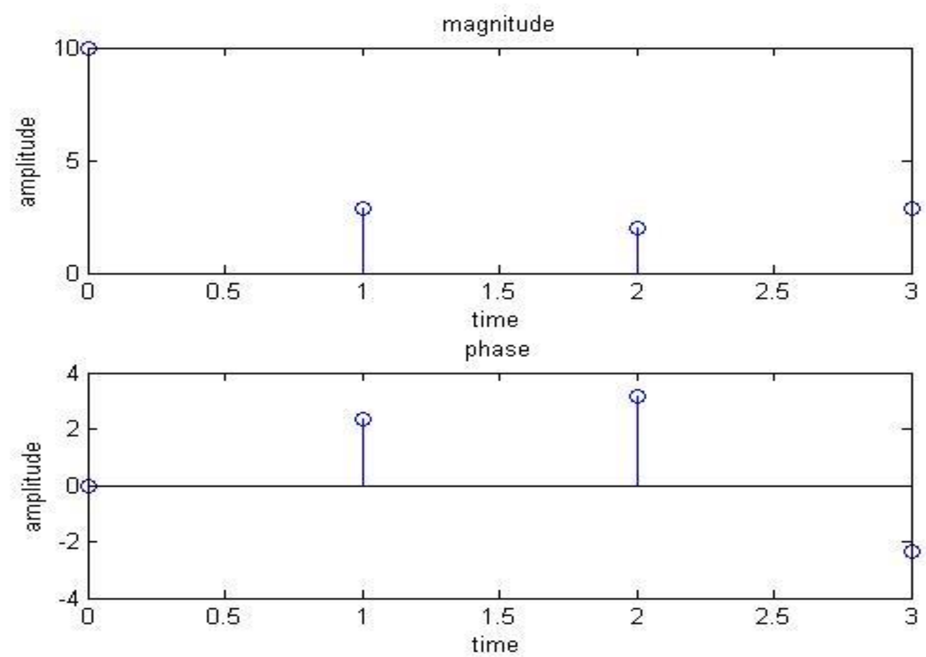
p =

0 2.3562 3.1416 -2.3562

The values of p are

0 2.3562 3.1416 -2.3562

Graph:



WRITE A MATLAB CODE TO FIND IDFT FOR THE GIVEN DFT SEQUENCE [USING BUILT IN ROUTINE].

Ifft: Inverse discrete Fourier transform.
Syntax: x=ifft(x,N)
Description: x=ifft(x,N) returns the N-point inverse DFT of vector x.

PROGRAM:

```
clc;
X=input('enter the dft sequence=');
N=input('enter the number of idft points=');
x=ifft(X,N);
disp('the value of x are=')
disp(x)
n=0:length(x)-1;
stem(n,x)
xlabel('time');
ylabel('amplitude');
```

title('input discrete sequence');

OUTPUT:

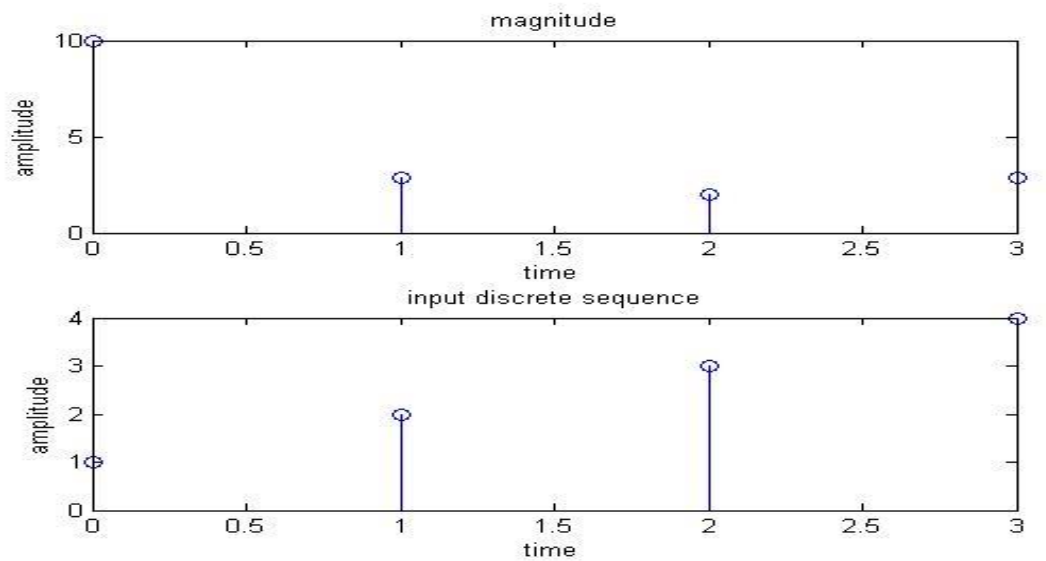
Enter the dft sequence=[10 -2+2j -2 -2-2j]

Enter the number of idft points=4

The values of x are=

1 2 3 4

Graph:



WRITE A MATLAB CODE TO FIND DFT FOR THE GIVEN SEQUENCE AND ALSO FIND MAGNITUDE AND PHASE USING DFT EQUATION

Zeros: zeros

Syntax: B=zeros (M, N)

Description: B=zeros (M, N) returns an M-by-N matrix of zeros.

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 elements of x.

Angle: 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. Converts back to the original complex Z.

DFT equation:

$$X(K) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N} \quad 0 \le K \le N-1$$

PROGRAM:

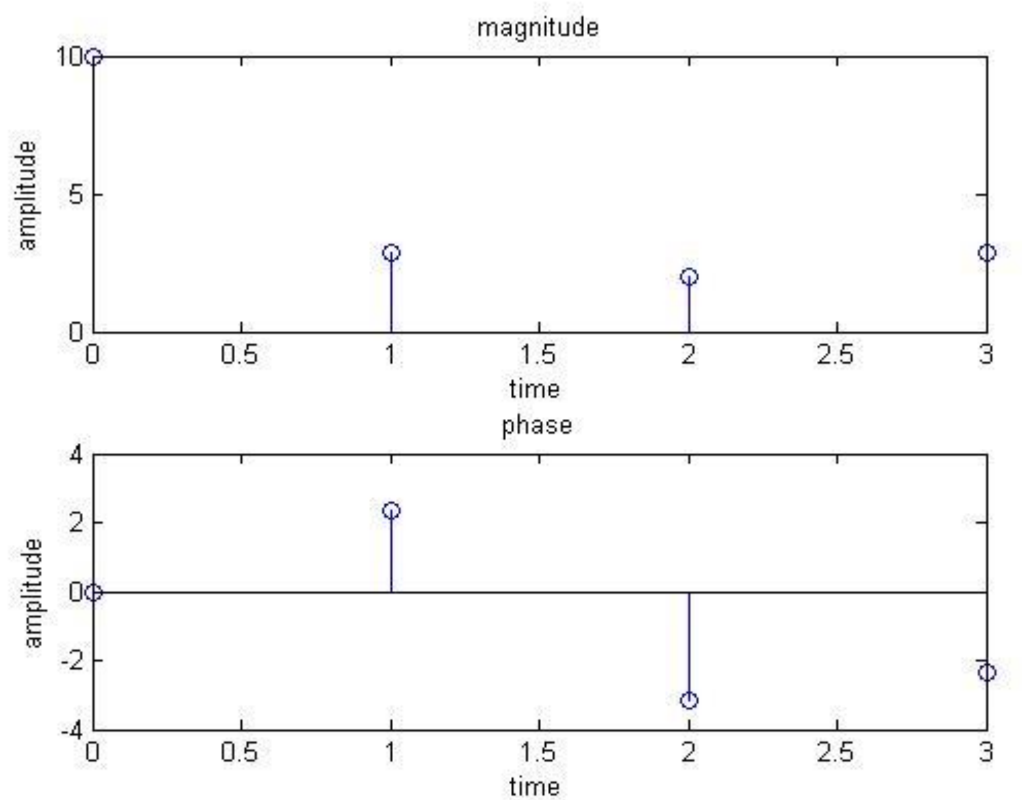
```
clc;
x=input('enter the dft sequence=');
N=input('enter the number of dft points=');
X=zeros(1,N);
for k=0:N-1
for n=0:N-1
X(k+1)=X(k+1)+x(n+1)*exp((-i)*2*pi*k*n/N);
end
end
disp('the value of X are=')
disp(X)
M=abs(X)
disp('the value of M are=')
disp(M)
subplot(2,1,1)
n=0:length(M)-1;
stem(n,M)
xlabel('time');
ylabel('amplitude');
title('magnitude');
p=angle(X)
disp('the value of p are=')
disp(p)
subplot(2,1,2)
n=0:length(p)-1;
stem(n,p)
xlabel('time');
```

```
ylabel('amplitude');
title('phase');
```

OUTPUT:

```
Enter the dft 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

M =

   10.0000    2.8284    2.0000    2.8284

The value of M are=

   10.0000    2.8284    2.0000    2.8284

p =

        0    2.3562   -3.1416   -2.3562

The values of p are=

        0    2.3562   -3.1416   -2.3562
```

Graph:


magnitude
10
5
0
amplitude
0
0.5
1
1.5
2
2.5
3
time
phase
4
2
0
-2
-4
amplitude
0
0.5
1
1.5
2
2.5
3
time

WRITE A MATLAB CODE TO FIND IDFT FOR THE GIVEN SEQUENCE USING IDFT EQUATION

IDFT equation:

$$x(n) = 1/N \sum_{k=0}^{N-1} X(K)\, e^{j2\pi kn/N} \quad n=0, 1 \ldots N-1$$

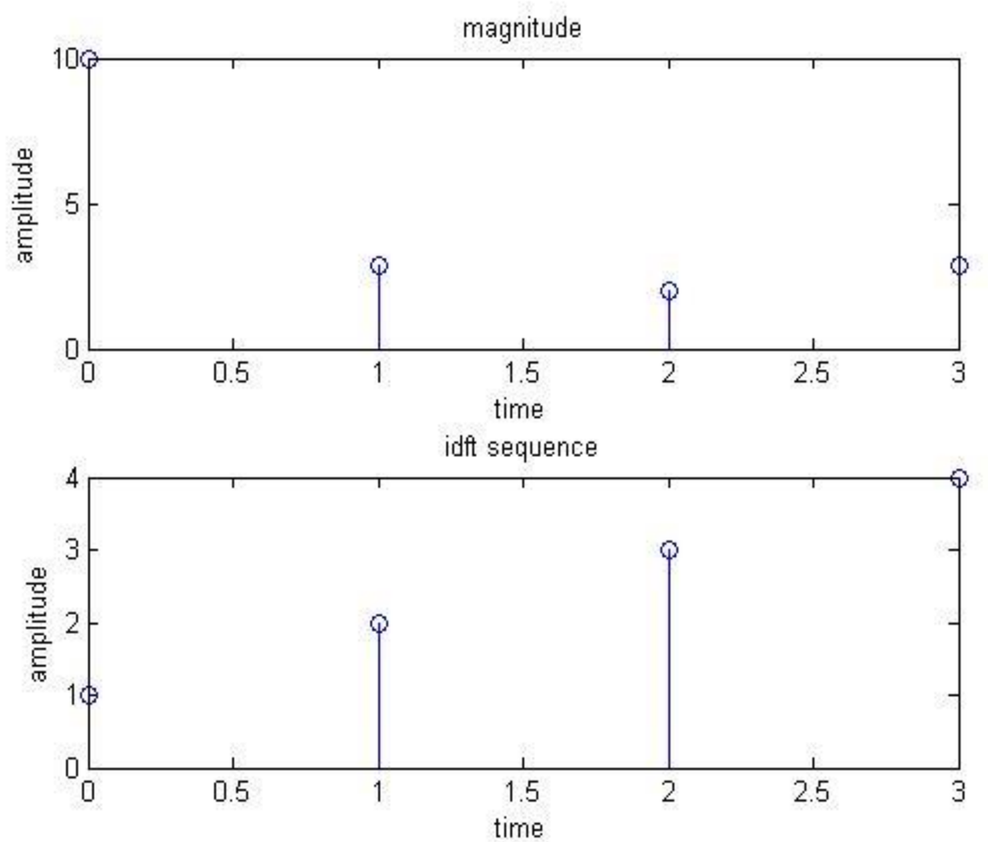
PROGRAM:

```
clc;
X=input('enter the dft sequence=');
N=input('enter the idft sequence=');
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 idft sequence is')
disp(xn)
n=0:length(xn)-1;
stem(n,xn)
xlabel('time');
ylabel('amplitude');
title('idft sequence');
```

OUTPUT:

```
Enter the dft sequence=[10 -2+2j -2 -2-2j]
Enter the idft sequence= [4]
The idft sequence is
   1.0000 + 0.0000i   2.0000 + 0.0000i   3.0000 - 0.0000i   4.0000 - 0.0000i
```

Graph:



6- (I) VERIFICATION OF DFT PROPERTIES (LIKE LINEARITY AND PARSEVALS THEOREM, ETC). (II) DFT COMPUTATION OF SQUARE PULSE AND SINC FUNCTION ETC.

WRITE A MATLAB CODE TO DETERMINE WHETHER THE GIVEN DFT SEQUENCE IS LINEAR OR NON LINEAR

SUM: Sum of array element

Syntax: B=sum (A)

Description: B=sum (A) returns sum along different dimension of an array

PROGRAM:

```
clc;
x1=input('enter the first input sequence');
x2=input('enter the second input sequence');
N=input('enter the dft point');
a=1;
b=1;
y=a*x1+b*x2;
y1=fft(y,N);
disp('the value of y1 are=' )
disp(y1)
x1=fft(a.*x1,N);
x2=fft(b.*x2,N);
y2=x1+x2;
disp('the value of y2 are=')
disp(y2)
if(y1==y2)
   disp('the dft is linear')
else
   disp('the dft is nonlinear')
end
```

OUTPUT:

Enter the first input sequence[1 2 3 4]

Enter the second input sequence[4 3 2 1]

Enter the dft point4

The value of y1 are=

    20   0   0   0

The value of y2 are=

    20   0   0   0

The dft is linear

WRITE A MATLAB CODE TO VERIFY PARSVELS THEOREM

Equation: $\sum_{k=0}^{N-1} |\mathbf{x(n)}|^2 = 1/N \sum_{k=0}^{N-1} |\mathbf{x(K)}|^2$

PROGRAM:

```
clc;
x=input('enter the  input sequence');
N=input('enter the number of  DFT point');
X=fft(x,N)
Energy 1 =sum(x.^2);
disp('the value of energy1 are')
disp (energy 1)
Energy2=1/N* sum (abs(x). ^2);
disp ('the value of energy 2 are')
disp(energy2)
If (energy1==energy2)
disp('DFT satisfies parsvel theorem")
else
```

```
disp('dft does not satisfies parsevel's theorem ')
end
```

7-DESIGN AND IMPLEMENTATION OF FIR FILTER TO MEET GIVEN SPECIFICATIONS (USING DIFFERENT WINDOW TECHNIQUES).

Hamming: Hamming window

Syntax:

w = hamming(L)

w = hamming(L,'sflag')

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 equation.

$$w(n) = 0.54 - 0.46\cos\left(2\pi\frac{n}{N}\right), 0 \leq n \leq N$$

The window length is L=N+1 .

Blackman: Blackman window

Syntax:

w = blackman(N)

w = blackman(N,SFLAG)

Description

w = blackman(N) returns the N-point symmetric Blackman window in the column vector w, where N is a positive integer.

Hann: Hann (Hanning) windowexpand all in page

Syntax:

w = hann(L)

w = hann(L,'sflag')

Description:

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

$$w(n) = 0.54\left(1 - \cos\left(2\pi\frac{n}{N}\right)\right), 0 \le n \le N$$

The window length is L=N+1.

Kaiser: Kaiser window

Syntax: w = kaiser(L,beta)

Description:

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

To obtain a Kaiser window that designs an FIR filter with sidelobe attenuation of α dB, use the following β.

$$\beta = \begin{cases} 0.1102(\alpha - 8.7), & \alpha > 50 \\ 0.5842(\alpha - 21)^{0.4} + 0.07886(\alpha - 21), & 50 \ge \alpha \ge 21 \\ 0, & \alpha < 21 \end{cases}$$

Increasing beta widens the main lobe and decreases the amplitude of the sidelobes (i.e., increases the attenuation).

Rectwin: Rectangular window

Syntax: w = rectwin(L)

Description:

w = rectwin(L) returns a rectangular window of length L in the column vector w. This function is provided for completeness; a rectangular window is equivalent to no window at all.

WRITE A MATLABCODE TO DESIGN A FIR FILTER USING BLACKMAN WINDOW APPROXIMATION

AIM: Design and implementation of FIR filter to meet given specifications (low pass filter using BLACKMAN window).

OBJECTIVE:

1. To design the FIR filter by Blackman window using the inbuilt MATLAB function "FIR1 and Blackman".
2. To verify the result by theoretical calculations.

PROGRAM:

```
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=blackman(L);
b=fir1(N,wc,'low',w)
freqz(b)
title('FIR filter using blackman');
```

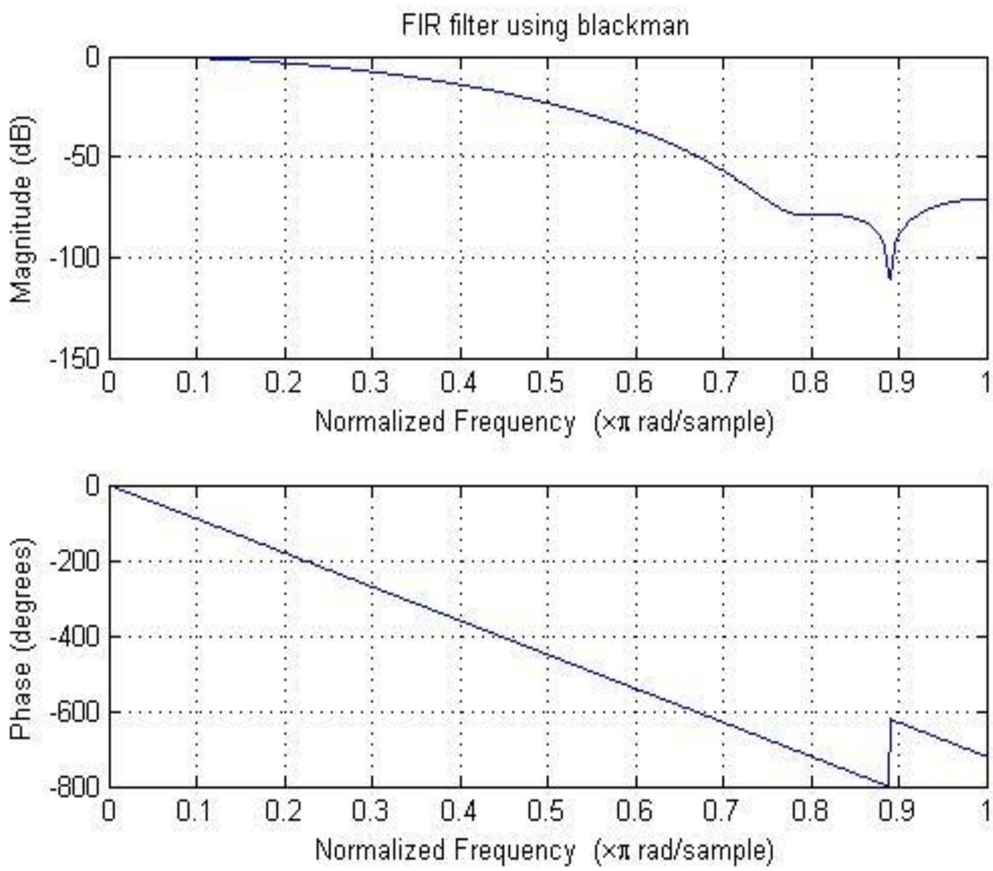
OUTPUT:

Enter the cutoff frequency=100

Enter the sampling frequency=1000

```
Enter the order of the filter=10
b =

  Columns 1 through 9

       0    0.0026    0.0283    0.1077    0.2218    0.2792    0.2218    0.1077    0.0283

  Columns 10 through 11

    0.0026         0
```


Graph:



WRITE A MATLOB CODE TO DESIGN AFIR FILTER USING HAMMING WINDOW APPROXIMATION

AIM: Design and implementation of FIR filter to meet given specifications (low pass filter using hamming window).

OBJECTIVE:

1. To design the FIR filter by Hamming window using the inbuilt MATLAB function "FIR1 and hamming".
2. To verify the result by theoretical calculations.

PROGRAM:

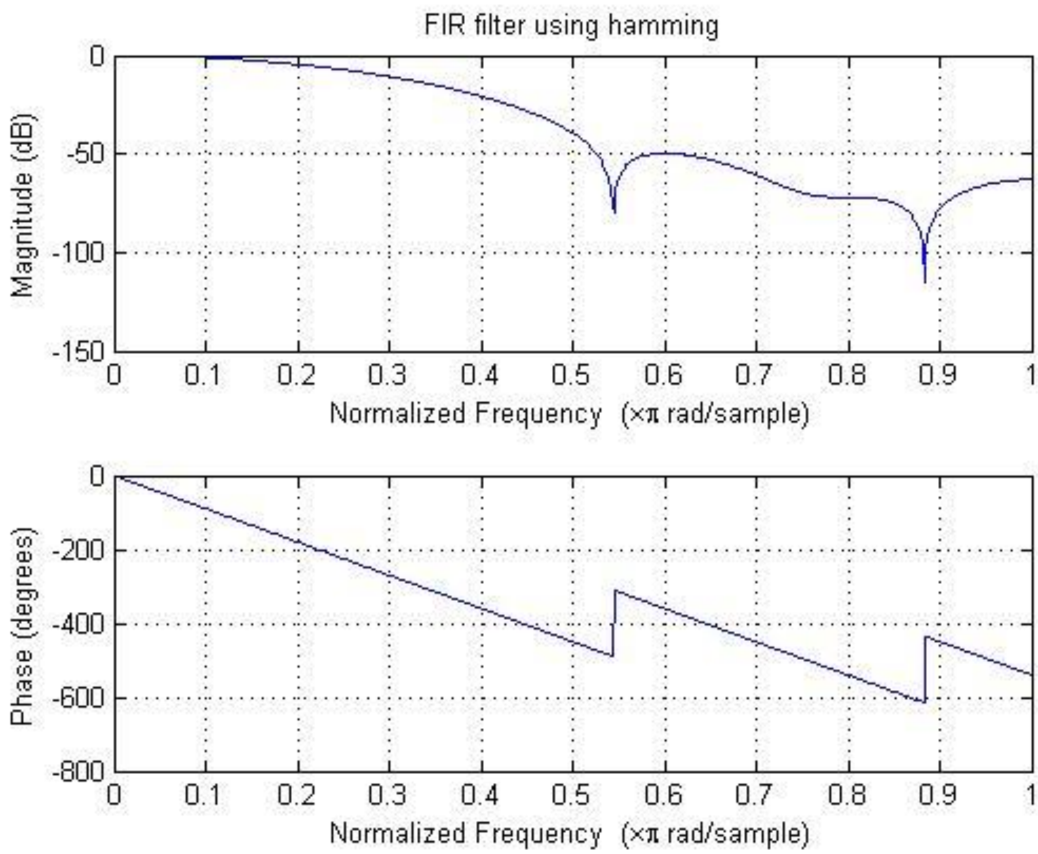
```
clc;
fcut=input('enter the cutoff frequency=');
fs=input('enter the sampling frequency=');
N=input('enter the order of the filter=');
wc=2*fcut/fs;
L=N+1;
w=hamming(L);
b=fir1(N,wc,'low',w)
freqz(b)
title('FIR filter using hamming');
```

OUTPUT:

```
Enter the cutoff frequency=100
Enter the sampling frequency=1000
Enter the order of the filter=10
b =
 Columns 1 through 9

  0.0000   0.0093   0.0476   0.1224   0.2022   0.2370   0.2022   0.1224   0.0476

 Columns 10 through 11
```

0.0093 0.0000

Graph:



WRITE A MAT LAB CODE TO DESIGN A FIR FILTER USING HANNING WINDOW APPROXIMATION

AIM: Design and implementation of FIR filter to meet given specifications (low pass filter using hanning window).

OBJECTIVE:

1. To design the FIR filter by hanning window using the inbuilt MATLAB function "FIR1 and hanning".
2. To verify the result by theoretical calculations.

PROGRAM:

```
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=hanning(L);
b=fir1(N,wc,'low',w)
freqz(b)
title('FIR filter using hanning');
```

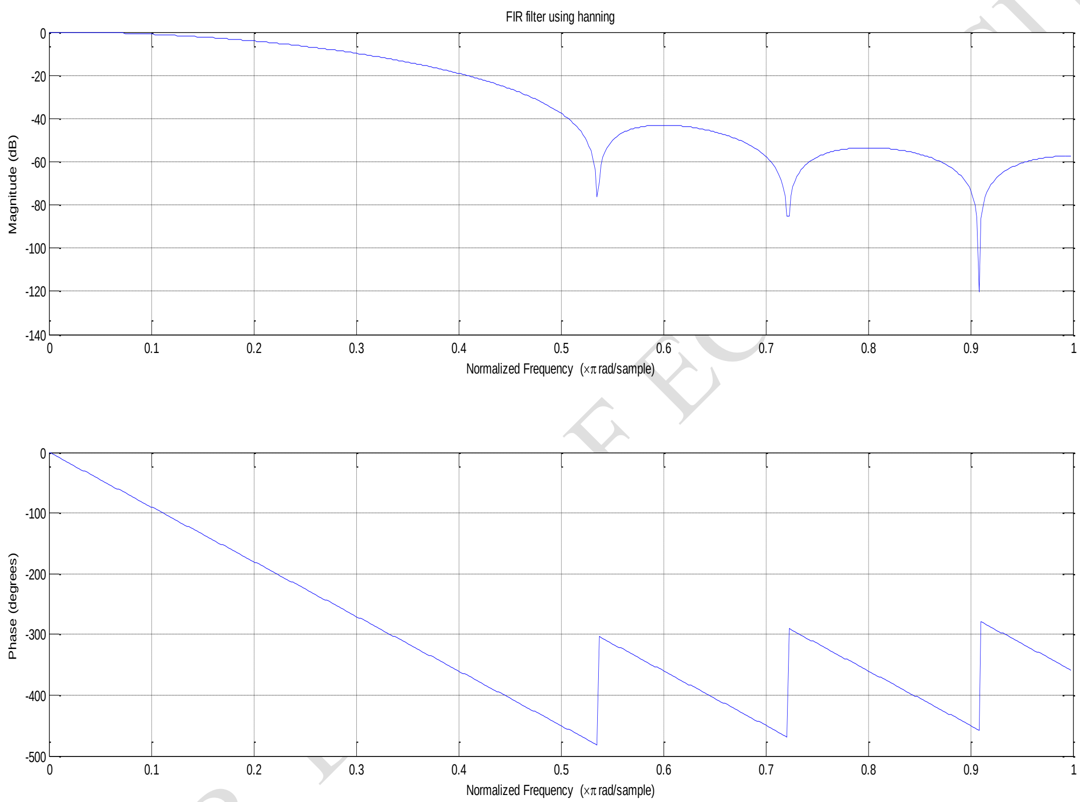
OUTPUT:

```
Enter the cutoff frequency=100hz

Enter the sampling frequency=1000hz

Enter the order of the filter=10

b =

 Columns 1 through 9

   0.0000    0.0130    0.0560    0.1261    0.1939    0.2221    0.1939    0.1261    0.0560
```

Columns 10 through 11

0.0130 0.0000

Graph:



WRITE A MATLOB CODE TO DESIGN AFIR FILTER USING KAISER WINDOW APPROXIMATION

AIM: Design and implementation of FIR filter to meet given specifications (low pass filter using Kaiser Window).

OBJECTIVE:

1. To design the FIR filter by Kaiser Window using the inbuilt MATLAB function "FIR1 and Kaiser".
2. To verify the result by theoretical calculations.

PROGRAM:

```
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');
```

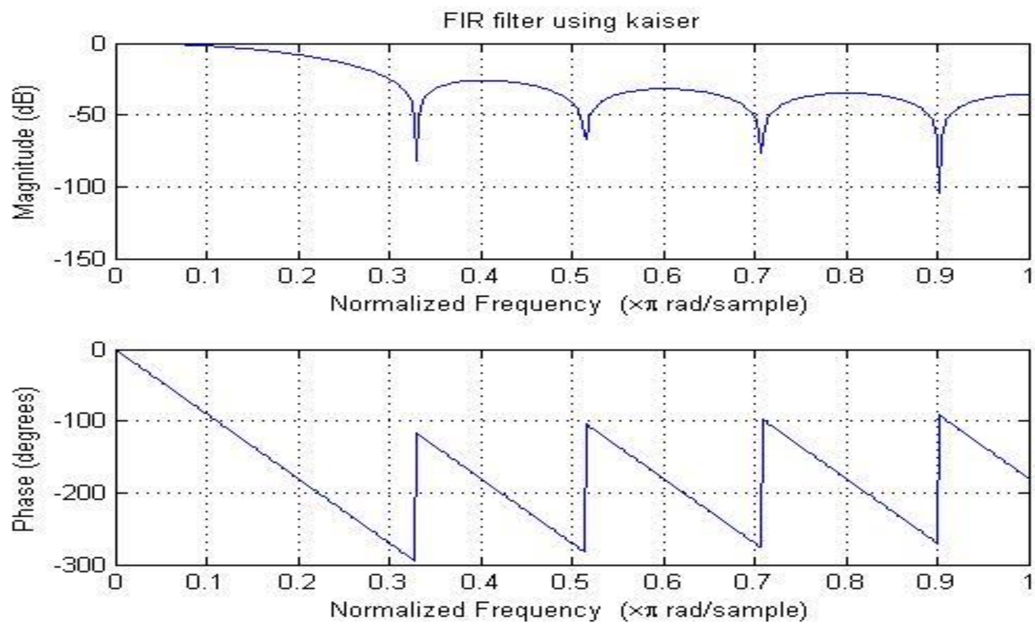
OUTPUT:

```
Enter the cutoff frequency=100

Enter the sampling frequency=1000

Enter the order of the filter=10

b =

  Columns 1 through 9

    0.0000    0.0388    0.0851    0.1292    0.1608    0.1723    0.1608    0.1292    0.0851
```

Columns 10 through 11

0.0388   0.0000

Graph:



WRITE A MATLOB CODE TO DESIGN AFIR FILTER USING RECTWIN WINDOW APPROXIMATION

AIM: Design and implementation of FIR filter to meet given specifications (low pass filter using rectwin window).

OBJECTIVE:

1. To design the FIR filter by Rectwin window using the inbuilt MATLAB function "FIR1 and rectwin".
2. To verify the result by theoretical calculations.

PROGRAM:

```
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');
```

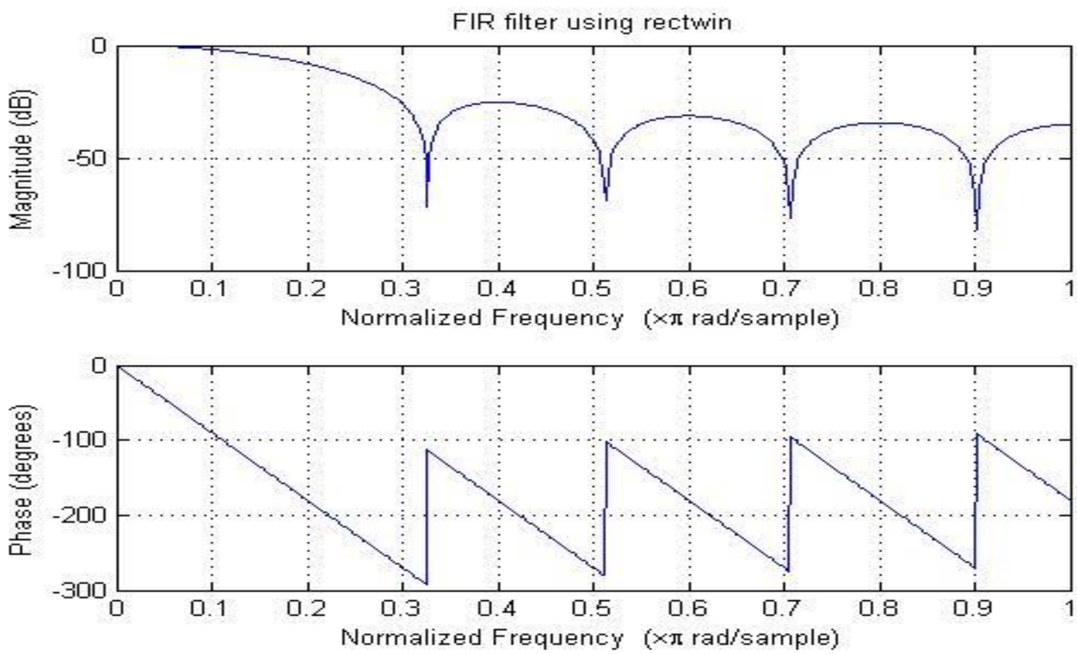
OUTPUT:

```
Enter the cutoff frequency=100

Enter the sampling frequency=1000

Enter the order of the filter=10

b =

 Columns 1 through 9

   0.0000    0.0399    0.0861    0.1291    0.1596    0.1706    0.1596    0.1291    0.0861
```

Columns 10 through 11

0.0399 0.0000

Graph:



8- DESIGN AND IMPLEMENTATION OF IIR FILTER TO MEET GIVEN SPECIFICATIONS.

AIM: Design and implementation of IIR filter to meet given specifications.

OBJECTIVE:

1- To design the BUTTERWORTH filter to meet the given specification using the MATLAB functions BUTTORD and BUTTER
2- Bilinear transformation for analog-to-digital filter conversion using function "BILINEAR"
3- To design the CHEBYSHEV filter to meet the given specification using the MATLAB function cheb1ord and cheby1
4- To verify the result by theoretical calculations.

Buttord: Butterworth filter order and cutoff 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.

Butter: Butterworth IIR filter design using specification object

Syntax:

hd = design(d,'butter')

hd = design(d,'butter',designoption,value...)

Description:

hd = design(d,'butter') designs a Butterworth IIR digital filter using the specifications supplied in the object d.

hd = design(d,'butter',designoption,value...) returns a Butterworth IIR filter where you specify a design option and value.

cheby1: Chebyshev Type I filter using specification object

Syntax:

hd = design(d,'cheby1')

hd = design(d,'cheby1',designoption,value,designoption, value,...)

Description:

hd = design(d,'cheby1') designs a type I 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. Use designmethods with the filter specification object to determine if a Chebyshev type I filter design is possible.

hd = design(d,'cheby1',designoption,value,designoption, value,...) returns a type I Chebyshev IIR filter where you specify design options as input arguments.

cheb1ord: Chebyshev Type I 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 I filter required to meet a set of filter design specifications.

Digital Domain [n,Wp] = cheb1ord(Wp,Ws,Rp,Rs) returns the lowest order n of the Chebyshev Type I filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. The scalar (or vector) of corresponding cutoff frequencies Wp, is also returned. Use the output arguments n and Wp with the cheby1 function.

Freqs: Frequency response of analog filters

Syntax:

h = freqs(b,a,w)

[h,w] = freqs(b,a,n)

Description:

freqs returns the complex frequency response H(jω) (Laplace transform) of an analog filter

$$H(s) = \frac{B(s)}{A(s)} = \frac{b(1)s^n + b(2)s^{n-1} + \cdots \dots . + b(n+1)}{a(1)s^m + a(2)s^{m-1} + \cdots \dots . + a(m+1)}$$

Given the numerator and denominator coefficients in vectors b and a.

h = freqs(b,a,w) returns the complex frequency response of the analog filter specified by coefficient 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.

[h,w] = freqs(b,a,n) uses n frequency points to compute the frequency response, h, where n is a real, scalar value. The frequency vector w is auto-generated and has length n. If you omit n as an input, 200 frequency points are used. If you do not need the generated frequency vector returned, you can use the form h = freqs (b,a,n) to return only the frequency response, h.

Freqz: Frequency response of filter

Syntax:

[h,w] = freqz(hfilt)

[h,w] = freqz(hfilt,n)

freqz(hfilt)

[h,w] = freqz(hs)

[h,w] = freqz(hs,n)

[h,w] = freqz(hs,Name,Value)

freqz(hs)

Description:

freqz returns the frequency response based on the current filter coefficients. This section describes common freqz operation with adaptive filters, discrete-time filters, multirate filters, and filter System objects.

[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.

Bilinear: Bilinear transformation method for analog-to-digital filter conversion

Syntax:

[zd,pd,kd] = bilinear(z,p,k,fs)

[zd,pd,kd] = bilinear(z,p,k,fs,fp)

[numd,dend] = bilinear(num,den,fs)

[numd,dend] = bilinear(num,den,fs,fp)

[Ad,Bd,Cd,Dd] = bilinear(A,B,C,D,fs)

[Ad,Bd,Cd,Dd] = bilinear(A,B,C,D,fs,fp)

Description:

The bilinear transformation is a mathematical mapping of variables. In digital filtering, it is a standard method of mapping the s or analog plane into the z or digital plane. It transforms analog filters, designed using classical filter design techniques, into their discrete equivalents.

The bilinear transformation maps the s-plane into the z-plane by

$$H(z) = H(s)\big|_{s=2f_z\frac{z-1}{z+1}}$$

This transformation maps the jΩ axis (from Ω = −∞ to +∞) repeatedly around the unit circle (ejw, from ω = −π to π) by

$$w = 2\tan^{-1}\frac{\Omega}{2f_s}$$

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 numerator and denominator coefficients bz and az, respectively, whose impulse response is equal to the impulse response of the analog filter with coefficients b and a, scaled by 1/fs. If you leave out the argument fs, or specify fs as the empty vector , it takes the default value of 1 Hz.

WRITE A MATLAB CODE TO DESIGN ANALOG BUTTERWORTH LOW PASS FILTERS.

PROGRAM:

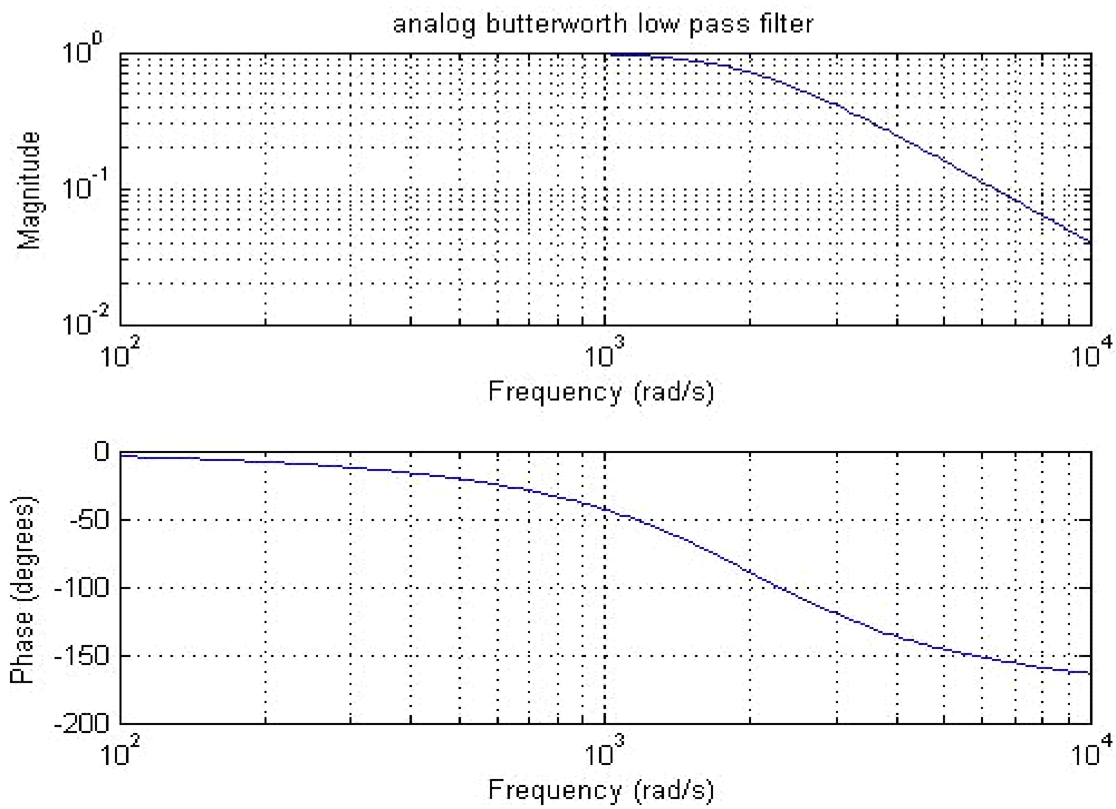
```
clc;
fp=input('enter the passband edge frequency=');
fs=input('enter the stopband edge frequency=');
ap=input('enter the passband attenuation=');
as=input('enter the stopband  attenuation=');
wp=2*pi*fp;
ws=2*pi*fs;
[n,wn]=buttord(wp,ws,ap,as,'s')
[z,p]=butter(n,wn,'low','s')
disp('the order of lowpass filter is')
disp(n)
printsys(z,p)
freqs(z,p)clc;
title('analog butterworth low pass filter');
```

OUTPUT:

```
Enter the pass band edge frequency=250
Enter the stop band edge frequency=750
Enter the pass band attenuation=3
Enter the stop band attenuation=15
n =
    2
wn =
  2.0032e+03
z =
  1.0e+06 *
```

```
        0         0    4.0129

p =

   1.0e+06 *

    0.0000    0.0028    4.0129

The order of low pass filter is
     2

Num/den =

             4012915.1817
   --------------------------------
   s^2 + 2832.9897 s + 4012915.1817
```


Graph:



WRITE A MAT LAB CODE TO DESIGN ANALOG BUTTERWORTH HIGH PASS FILTERS.

PROGRAM:

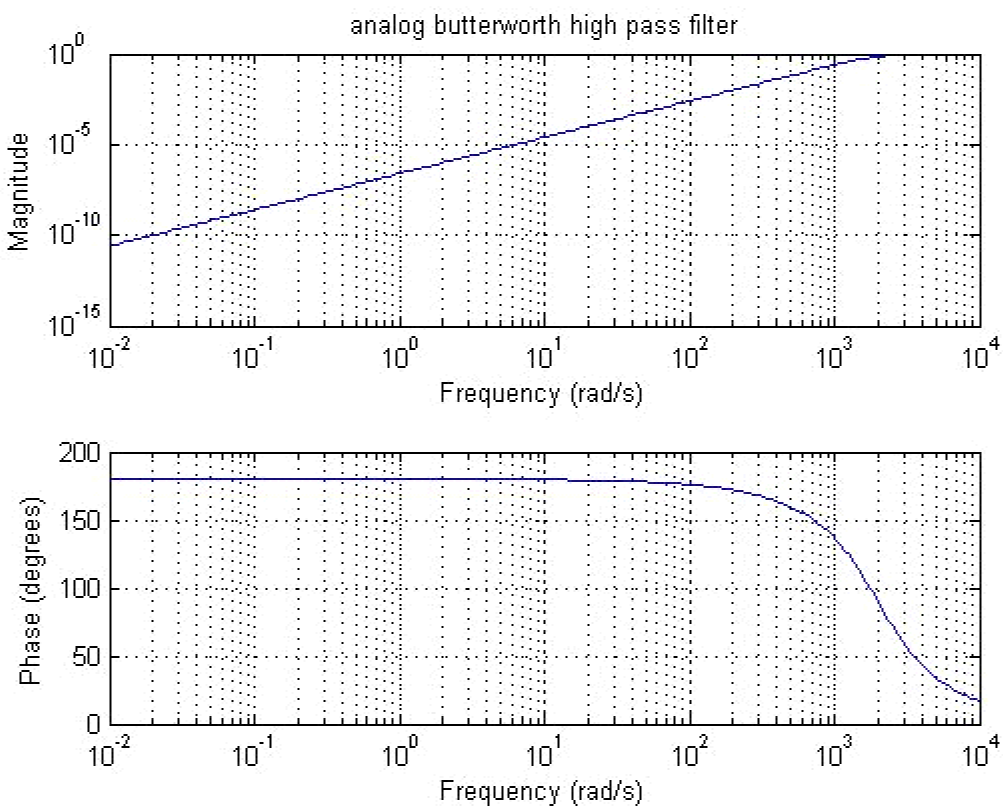
```
clc;
fp=input('enter the passband edge frequency=');
fs=input('enter the stopband edge frequency=');
ap=input('enter the passband attenuation=');
as=input('enter the stopband  attenuation=');
wp=2*pi*fp;
ws=2*pi*fs;
[n,wn]=buttord(wp,ws,ap,as,'s')
[z,p]=butter(n,wn,'high','s')
disp('the order of highpass filter is')
disp(n)
printsys(z,p)
freqs(z,p)
title('analog butterworth high pass filter');
```

OUTPUT:

```
Enter the pass band edge frequency=250
Enter the stop band edge frequency=750
Enter the pass band attenuation=3
Enter the stop band attenuation=15
n =
    2

wn =

  2.0032e+03

z =
```

```
    1     0     0

p =

   1.0e+06 *

    0.0000    0.0028    4.0129

the order of highpass filter is

    2

Num/den =

              S^2

   ---------------------------------

   s^2 + 2832.9897 s + 4012915.1817
```

Graph:



WRITE A MATLAB CODE TO DESIGN ANALOG CHEBYSBES LOWPASS FILTER.

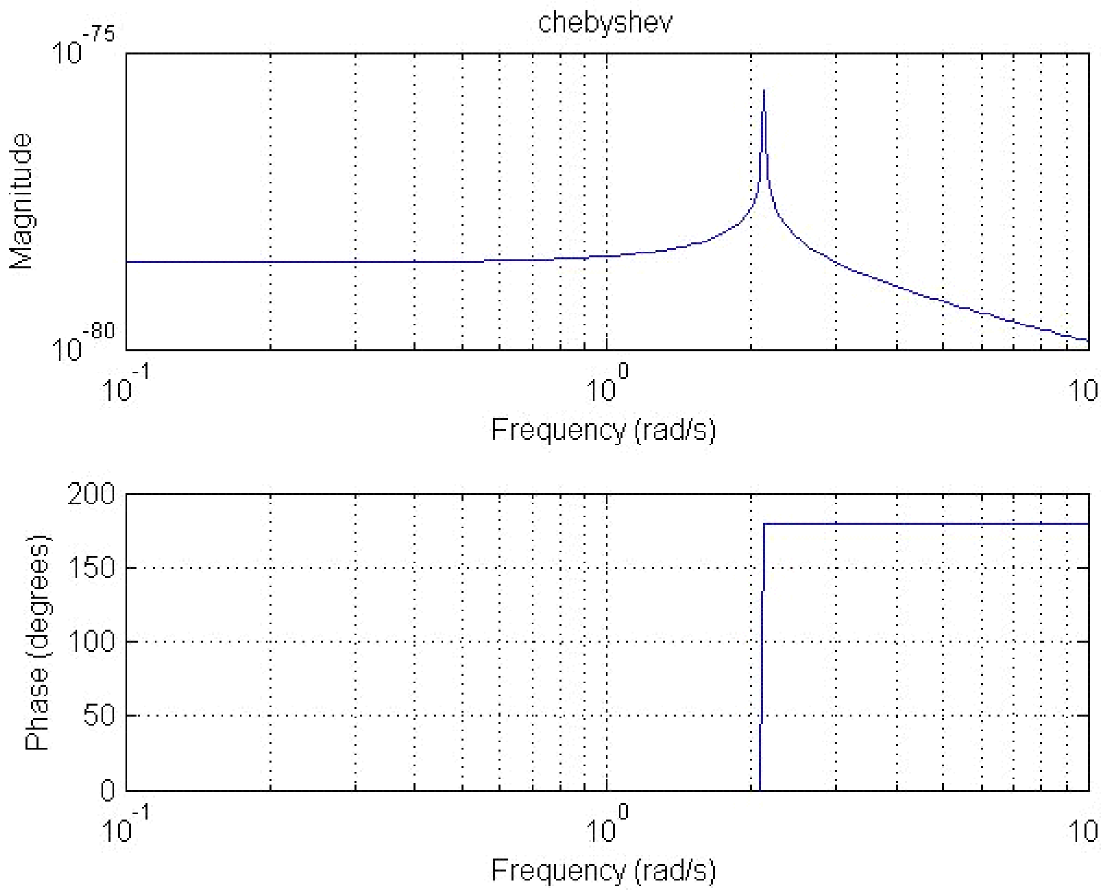
PROGRAM:

```
clc;
fp=input('enter the passband edge frequency =');
fs=input('enter the stopband edge frequency =');
Ap=input('enter the passsband attenuation =');
As=input('enter the stopband attenuation');
wp=2*pi*fp;
ws=2*pi*fs;
[n,wp]=cheb1ord(wp,ws,Ap,As,'s');
[b,a]=cheby1(n,wp,Ap,'low','s');
disp('the order of lowpass filter is')
disp(n)
printsys(b,a)
freqs(b,a)
title('chebyshev');
```

OUTPUT:

```
Enter the pass band edge frequency =250
Enter the stop band edge frequency =750
Enter the pass band attenuation =3
Enter the stop band attenuation15
The order of low pass filter is
   2
Num/den =

  1.2984e-78

  ----------

   s^2 + 4.5
```

Graph:


chebyshev
Magnitude
10^{-75}
10^{-80}
10^{-1}
10^{0}
10^{1}
Frequency (rad/s)
Phase (degrees)
200
150
100
50
0
10^{-1}
10^{0}
10^{1}
Frequency (rad/s)

WRITE A MATLAB CODE TO DESIGN A ANALOG CHEBYSBES HIGHPASS FILTER

PROGRAM:

```
clc;
fp=input('enter the passband edge frequency =');
fs=input('enter the stopband edge frequency =');
Ap=input('enter the passsband attenuation =');
As=input('enter the stopband attenuation');
wp=2*pi*fp;
ws=2*pi*fs;
[n,wp]=cheb1ord(wp,ws,Ap,As,'s');
[b,a]=cheby1(n,wp,Ap,'high','s');
disp('the order of highpass filter is')
disp(n)
printsys(b,a)
freqs(b,a)
title('chebyshev');
```

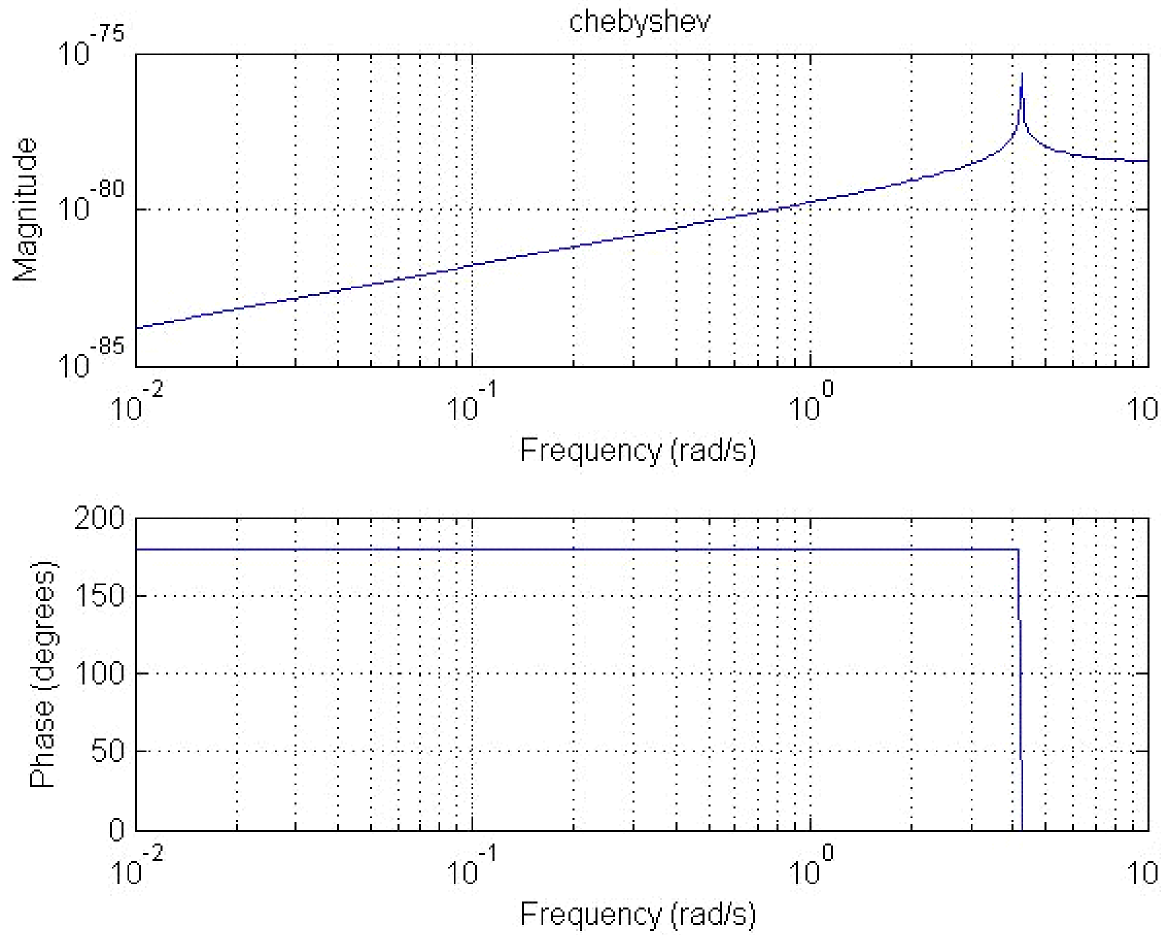
OUTPUT:

```
Enter the passband edge frequency =250

Enter the stopband edge frequency =750

Enter the passsband attenuation =3

Enter the stopband attenuation15

The order of lowpass filter is

    2

num/den =

  2.8853e-79 s^2
```

----

s^2 + 18

Graph:


chebyshev
Magnitude
10^-75
10^-80
10^-85
10^-2
10^-1
10^0
10^1
Frequency (rad/s)
Phase (degrees)
200
150
100
50
0
10^-2
10^-1
10^0
10^1
Frequency (rad/s)

WRITE A MATLAB CODE TO DESIGN LOWPASS FILTER USING BILINEAR TRANSFORMATION

PROGRAM:

```
clc;
fs=input('enter the sampling frequency=');
pf=input('enter the passband edge frequency=');
sf=input('enter the stopband frequency=');
ap=input('enter the passband attenuation=');
as=input('enter the stopband attenuation=');
pf1=2*pi*(pf/fs);
sf1=2*pi*(sf/fs);
wp=2*tan(pf1/2);
ws=2*tan(sf1/2);
[n,wn]=buttord(wp,ws,ap,as,'s');
[b,a]=butter(n,wn,'low','s');
disp('the order of the 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('lowpass filter');
```


OUTPUT:

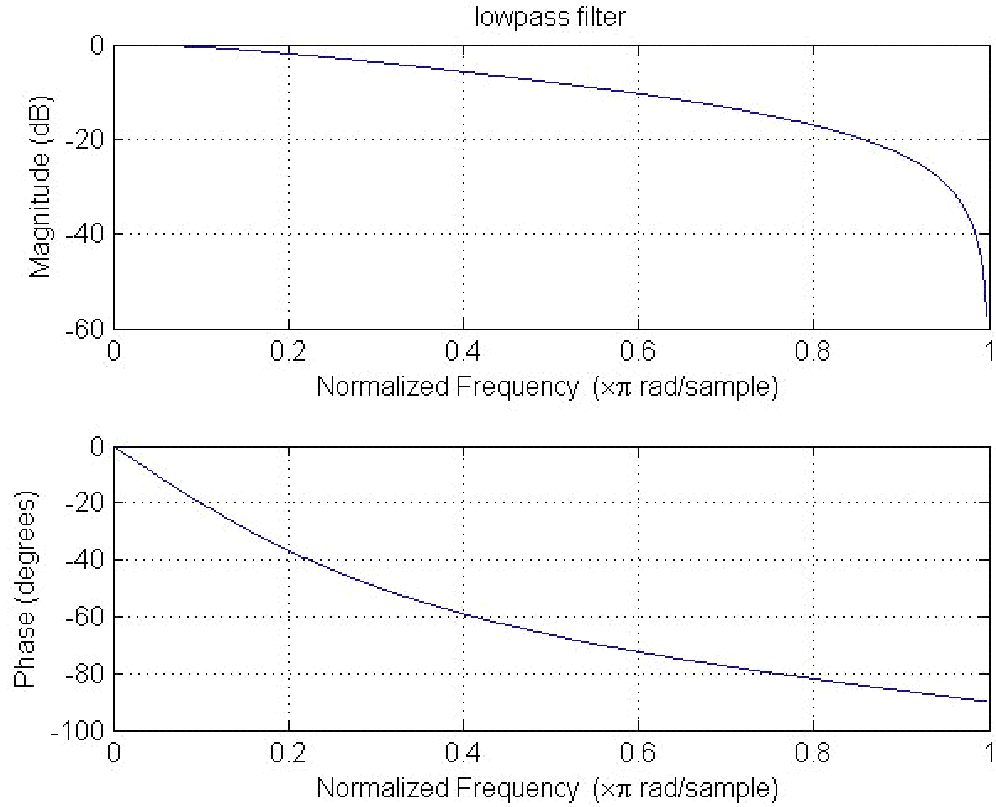
Enter the sampling frequency=2000

Enter the passband edge frequency=250

Enter the stopband frequency=750

```
Enter the passband attenuation=3
Enter the stopband attenuation=15
The order of the filter is
     1
System function in analog
Num/den = 0.87254
                      -----------
                    s + 0.87254
z =
    0.3038    0.3038
p =
    1.0000   -0.3925

System function in digital
num/den =   0.30375 z + 0.30375
                           ------------------
                                  z - 0.3925
```

Graph:



WRITE A MATLAB CODE TO DESIGN DIGITAL BUTTER WORD LOWPASS FILTER USING IMPULSE INVARIENT TRANSFORMATION

PROGRAM:

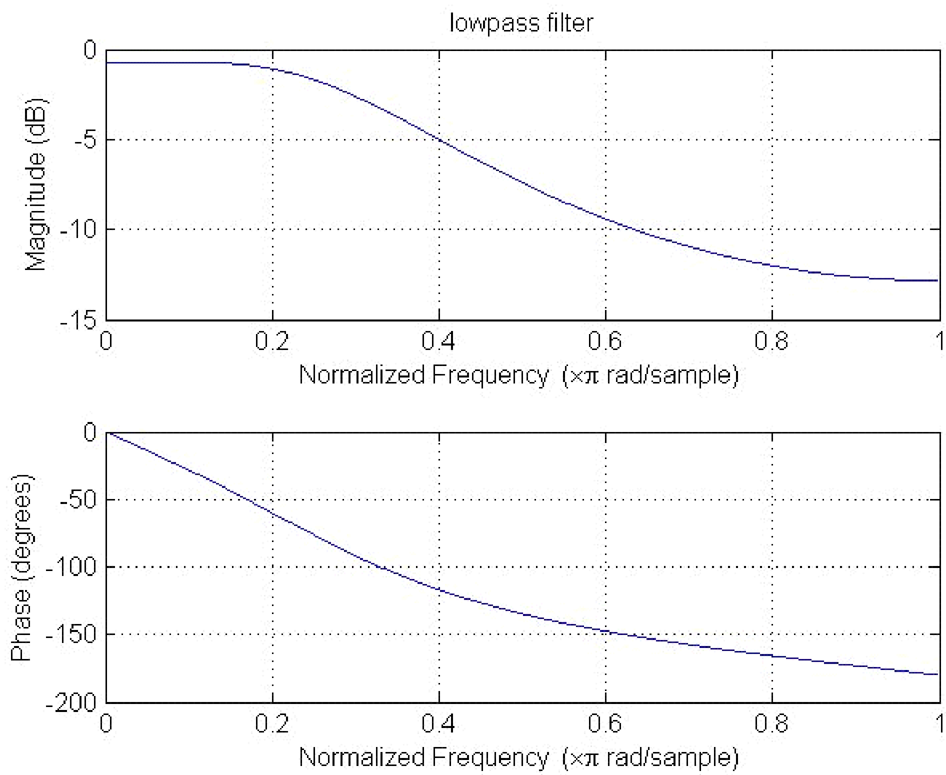
```
clc;
fs=input('enter the sampling frequency=');
pf=input('enter the passband edge frequency=');
sf=input('enter the stopband frequency=');
ap=input('enter the passband attenuation=');
as=input('enter the stopband attenuation=');
wp=2*pi*(pf/fs);
ws=2*pi*(sf/fs);
[n,wn]=buttord(wp,ws,ap,as,'s');
[b,a]=butter(n,wn,'low','s');
disp('the order of the filter is')
```

```
disp(n)
disp('system function is analog')
printsys(b,a,'s');
fs=1;
[z,p]=impinvar(b,a,fs);
disp('system function is digital')
printsys(z,p,'z');
freqz(z,p)
title('lowpass filter');
```

OUTPUT:

```
Enter the sampling frequency=2000
Enter the passband edge frequency=250
Enter the stopband frequency=750
Enter the passband attenuation=3
Enter the stopband attenuation=15
The order of the filter is
    2
System function is analog
Num/den =
        1.0032
 ----------------------
   s^2 + 1.4165 s + 1.0032
system function is digital

num/den =

         0.45381 z
   ------------------------
   z^2 - 0.74812 z + 0.24256
```

Graph:


lowpass filter
Magnitude (dB)
0
-5
-10
-15
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-50
-100
-150
-200
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)

HARDWARE PROGRAMS

C-PROGRAM TO PERFORM LINEAR CONVOLUTION BETWEEN TWO SEQUENCES

PROGRAM:

```
#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]);
}
```

C-PROGRAM TO PERFORM CIRCULAR CONVOLUTION BETWEEN TWO SEQUENCES

PROGRAM:

```
#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]);
}
}
```

C-PROGRAM TO GET IMPULSE RESPONSE FOR THE GIVEN FILTER COEFFICIENTS

PROGRAM:

```
#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]);
}
}
```

C-PROGRAM TO PERFORM N-POINT DFT BETWEEN TWO SEQUENCES

PROGRAM:

```
#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*)0x0000000;
for(k=0;k<2*N;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]);
}
}
```

Beyond Syllabus:

IMPLEMENTATION OF DECIMATION PROCESS

PROGRAM:

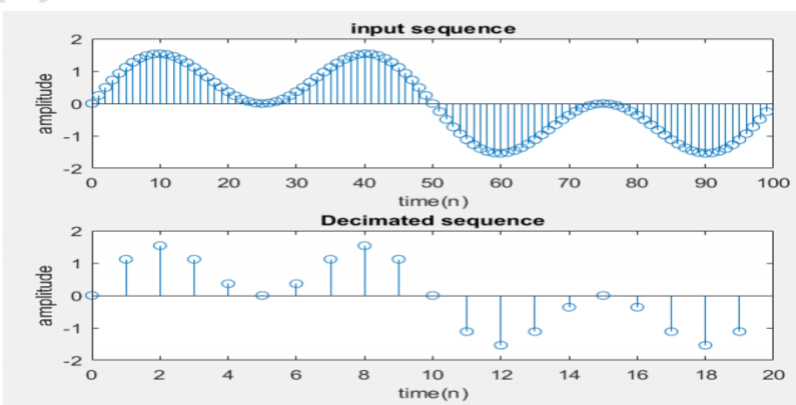
```
Clc;
Clear all;
Close all;
D=input('enter the downsampling factor');
L=input('enter the length of the input signal');
f1=input('enter the frequency of first sinusodal');
f2=input('enter the frequency of second sinusodal');
n=0:L-1;
x=sin(2*pi*f1*n)+sin(2*pi*f2*n);
y=decimate(x,D,'fir');
subplot(2,1,1);
stem(n,x(1:L));
title('input sequence');
xlabel('time(n)');
 ylabel('amplitude');
subplot(2,1,2) m=0:(L/D)-1;
stem(m,y(1:L/D));
 title('Decimated sequence');
 xlabel('time(n)');
ylabel('amplitude');
```

Expected Result:

INPUT:

enter the downsampling factor = 5
enter the length of the input signal = 100
enter the frequency of first sinusoidal = 0.01
enter the frequency of second sinusoidal = 0.03

Output Waveforms:



DSP Lab - Sample viva questions

1. What is the requirement for convolution?
2. What is the difference between convolution & correlation?
3. What is meant by impulse response?
4. Is it possible to represent any discrete time signal in terms of impulses? If yes, represent by using example.
5. Draw the h(2n-k) & h(n-2k) for the following sequence h(n) = { 4 3 2 1} assume (i) k= 3 (ii) k =5.
6. Write the expressions for LTI system convolution formula & causal LTI system convolution formula.
7. What us the length of linear convolution if length of input & impulse responses are N_1 & N_2 respectively?
8. What is the difference between continuous and discrete convolution?
9. What are the advantages of FIR as compared to IIR?
10. What do you understand by linear phase response?
11. To design all types of filters what would be the expected impulse response?
12. What are the properties of FIR filter?.
13. How the zeros in FIR filter is located?
14. What are the desirable characteristics of the window?
15. What are the specifications required to design filter?
16. What is meant by IIR filter?
17. What is the difference between recursive & non-recursive systems?
18. Write the difference equation for IIR system.
19. What are the mapping techniques in IIR filter design? Discuss the advantage & disadvantages of them.
20. What are IIR analog filters? What are the advantages & disadvantages of them?
21. What is the disadvantage in impulse invariance method?
22. Explain the pole mapping procedure of Impulse invariant & bilinear transformation method.
23. What is the difference between type I & type II T chebyshev filters?.
24. Where the poles are located for Butter worth & T chebyshev filters?

25. What is the difference between continuous time & discrete time Fourier transform?
26. What is the difference between discrete Time Fourier Transform (DTFT)& DFT?
27. What is the difference between Z transform & DFT?
28. State convolution property of the DFT? Where we could use the convolution property?
29. State Parseval's theorem.?
30. State correlation property of the DFT.?
31. What is the difference between radix 2 & radix4 FFT algorithms?
32. What is the difference between decimation in time (DIT FFT) & Decimation in frequency (DIF FFT) algorithms?
33. What is meant by 'in-place' computation in DIF & DIF algorithms?
34. Which properties are used in FFT to reduce no of computations?