NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY, BHOPAL

Session-

DEPARTMENT OF ELECTRONICS & COMMUNICAITON ENGINEERING

LAB MANUAL

OF

DIGITAL SIGNAL PROCESSING

EC-601

ABOUT MATLAB:

MATLAB is a 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 Math and computation, Algorithm development, Data acquisition, Modeling, simulation, and prototyping, Data analysis, exploration, and visualization, Scientific and engineering graphics, Application development, including graphical user interface building.

MATLAB is an interactive system whose basic data element is an **array**

that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or Fortran. The name MATLAB stands for **matrix laboratory**.

Starting MATLAB on Windows platforms, start MATLAB by double-clicking the MATLAB shortcut icon on your Windows desktop.

Quitting MATLAB: To end your MATLAB session, select File > Exit MATLAB in the desktop, or type quit in the Command Window. You can run a script file named finish.m each time MATLAB quits that, for example, executes functions to save the workspace.

➢ MATLAB DESKTOP:

On starting MATLAB the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables and applications associated with MATLAB. It contains:

(i) Command Window and Command History

Command Window: Use the Command Window to enter variables and to run functions and M-file scripts. Press the up arrow key to recall a statement previously typed. Edit the statement as needed and then press Enter to run it.

Command History: Statements entered in the Command Window are logged in the Command History. From the Command History, previously run statements can be viewed, copied and executed. M-file can be created from selected statements.

(ii) Current Directory Browser and Search Path: MATLAB file operations use the current directory and the search path as reference points. Any file required to run must either be in the current directory or on the search path.

(iii) Workspace Browser and Array Editor: The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory. The variables can be added to the workspace by using functions, running M-files, and loading saved workspaces. To delete variables from the workspace, select the variables and select Edit > Delete.

Array Editor: Double-click a variable in the Workspace browser, or use open var variable name, to see it in the Array Editor. Use the Array Editor to view and edit a visual representation of variables in the workspace.

(iv) Editor/Debugger: **Editor/Debugger is used to create and debug M-files, which are programs to run MATLAB functions.** The Editor/Debugger provides a graphical user interface for text editing, as well as for M-file debugging. To create or edit an M-file use File > New or File > Open, or use the edit function.

==========

➢ The important commands/ functions are as below :

1. **clc (Remove items from workspace, freeing up system memory)** clears all input and output from the Command Window display, giving "clean screen." After using clc, the scroll bar cannot be used to see the history of functions, but still the up arrow can be used to recall statements from the command history.
2. **close (Remove specified figure)**: close deletes the current figure or the specified figure(s). It optionally returns the status of the close operation.
3. **xlabel, ylabel, zlabel (Label x-, y-, and z-axis) :** Each axes graphics object can have one label for the x-, y-, and z-axis. The label appears beneath its respective axis in a two-dimensional plot and to the side or beneath the axis in a three-dimensional plot.

   xlabel('string') labels the x-axis of the current axes.

   ylabel(...) and zlabel(...) label the y-axis and z-axis, respectively, of the current axes.
4. **title(** Add title to current axes) : Each axes graphics object can have one title. The title is located at the top and in the center of the axes.

   title('string') outputs the string at the top and in the center of the current axes.
5. **figure** (create figure graphics object) : figure creates figure graphics objects. Figure objects are the individual windows on the screen in which MATLAB displays graphical output.
6. **subplot (Create axes in tiled positions):** subplot divides the current figure into rectangular panes that are numbered row wise. Each pane contains an axes object. Subsequent plots are output to the current pane.

   h = subplot(m,n,p) or subplot(mnp) breaks the figure window into an m-by-n matrix of small axes, selects the pth axes object for the current plot, and returns the axes handle. The axes are counted along the top row of the figure window, then the second row, etc. For example,

   subplot(2,1,1), plot(income)

   subplot(2,1,2), plot(outgo) plots income on the top half of the window and outgo on the bottom half.

7. **stem (Plot discrete sequence data)** : A two-dimensional stem plot displays data as lines extending from a baseline along the x-axis. A circle (the default) or other marker whose y-position represents the data value terminates each stem.

   stem(Y) Plots the data sequence Y as stems that extend from equally spaced and automatically generated values along the x-axis. When Y is a matrix, stem plots all elements in a row against the same x value.

   stem(X,Y) plots X versus the columns of Y. X and Y must be vectors or matrices of the same size. Additionally, X can be a row or a column vector and Y a matrix with length(X) rows.

8. **bar(Plot bar graph (vertical and horizontal))** : A bar graph displays the values in a vector or matrix as horizontal or vertical bars.

   bar(Y) draws one bar for each element in Y. If Y is a matrix, bar groups the bars produced by the elements in each row. The x-axis scale ranges from 1 up to length(Y) when Y is a vector, and 1 to size(Y,1), which is the number of rows, when Y is a matrix.

   barh(...) and h = barh(...) create horizontal bars. Y determines the bar length. The vector x is a vector defining the y-axis intervals for horizontal bars.

9. **plot (2-D line plot)** : **plot(Y)** Plots the columns of Y versus their index if Y is a real number. If Y is complex, plot(Y) is equivalent to plot(real(Y),imag(Y)). In all other uses of plot, the imaginary component is ignored.

   plot(X1,Y1,...) Plots all lines defined by Xn versus Yn pairs. If only Xn or Yn is a matrix, the vector is plotted versus the rows or columns of the matrix,depending on whether the vector's row or column dimension matches the matrix. If Xn is a scalar and Yn is a vector, disconnected line objects are created and plotted as discrete points vertically at Xn.

10. **input (Request user input)** : The response to the input prompt can be any MATLAB expression, which is evaluated using the variables in the current workspace.

    user_entry = input('prompt') Displays prompt as a prompt on the screen, waits for input from the keyboard, and returns the value entered in user_entry. user_entry = input('prompt', 's') returns the entered string as a text variable rather than as a variable name or numerical value.

11. **zeros (Create array of all zeros)** :

    B = zeros(n) Returns an n-by-n matrix of zeros. An error message appears if n is not a scalar.

12. **ones (Create array of all ones)** :

    Y = ones(n) Rreturns an n-by-n matrix of 1s. An error message appears if n is not a scalar.

13. **exp (Exponential) :**

    Y = exp(X) The exp function is an elementary function that operates element-wise on arrays. Its domain includes complex numbers.

    Y = exp(X) returns the exponential for each element of X.

14. disp (Display text or array) :

disp(X) Displays an array, without printing the array name. If X contains a text string, the string is displayed. Another way to display an array on the screen is to type its name, but this prints a leading "X=," which is not always desirable. Note that disp does not display empty arrays.

15. conv (Convolution and polynomial multiplication) :

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.

16. xcorr (Cross-correlation) :

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

17. filter (1-D digital filter) :

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.

18. poly (Polynomial with specified roots) :

r = roots(p) which returns a column vector whose elements are the roots of the polynomial specified by the coefficients row vector p. For vectors, roots and poly are inverse functions of each other, up to ordering, scaling, and round off error.

19. tf(Convert unconstrained MPC controller to linear transfer function) :

sys=tf(MPCobj) The tf function computes the transfer function of the linear controller ss(MPCobj) as an LTI system in tf form corresponding to the MPC controller when the constraints are not active. The purpose is to use the linear equivalent control in Control System Toolbox for sensitivity and other linear analysis.

20. freqz (Frequency response of filter) :

[h,w] = freqz(ha) returns the frequency response vector h and the corresponding frequency vector w for the adaptive filter ha. When ha is a vector of adaptive filters, freqz returns the matrix h. Each column of h corresponds to one filter in the vector ha.

21. abs (Absolute value and complex magnitude) :

abs(X) returns an array Y such that each element of Y is the absolute value of the corresponding element of X.

22. fft (Discrete Fourier transform) :

Y = fft(X) Y = fft(X) returns the discrete Fourier transform (DFT) of vector X, computed with a fast Fourier transform (FFT) algorithm.

23. mod (Modulus after division) :

M = mod(X,Y) returns X - n.*Y where n = floor(X./Y). If Y is not an integer and the quotient X./Y is within round off error of an integer, then n is that integer. The inputs X and Y must be real arrays of the same size, or real scalars.

24. **sqrt (Square root) :**

B = sqrt(X) returns the square root of each element of the array X. For the elements of X that are negative or complex, sqrt(X) produces complex results.

25. **ceil (Round toward infinity) :**

B = ceil(A) rounds the elements of A to the nearest integers greater than or equal to A. For complex A, the imaginary and real parts are rounded independently.

26. **fir1(Window-based finite impulse response filter design) :**

b = fir1(n,Wn) returns row vector b containing the n+1 coefficients of an order n lowpass FIR filter. This is a Hamming-window based, linear-phase filter with normalized cutoff frequency Wn. The output filter coefficients, b, are ordered in descending powers of z.

27. **buttord (Butterworth filter order and cutoff frequency) :**

[n,Wn] = buttord(Wp,Ws,Rp,Rs) returns the lowest order, n, of the digital Butterworth 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, Wn, is also returned. Use the output arguments n and Wn in butter.

28. **fliplr (Flip matrix left to right) :**

B = fliplr(A) returns A with columns flipped in the left-right direction, that is, about a vertical axis.If A is a row vector, then fliplr(A) returns a vector of the same length with the order of its elements reversed. If A is a column vector, then fliplr(A) simply returns A.

29. **min (Smallest elements in array) :**

C = min(A) returns the smallest elements along different dimensions of an array. If A is a vector, min(A) returns the smallest element in A.If A is a matrix, min(A) treats the columns of A as vectors, returning a row vector containing the minimum element from each column. If A is a multidimensional array, min operates along the first nonsingleton dimension.

30. **max (Largest elements in array) :**

C = max(A) returns the largest elements along different dimensions of an array.If A is a vector, max(A) returns the largest element in A.If A is a matrix, max(A) treats the columns of A as vectors, returning a row vector containing the maximum element from each column. If A is a multidimensional array, max(A) treats the values along the first non-singleton dimension as vectors, returning the maximum value of each vector.

31. **find (Find indices and values of nonzero elements) :**

ind = find(X) locates all nonzero elements of array X, and returns the linear indices of those elements in vector ind. If X is a row vector, then ind is a row vector; otherwise, ind is a column vector. If X contains no nonzero elements or is an empty array, then ind is an empty array.

32. **residuez (z-transform partial-fraction expansion) :**

residuez converts a discrete time system, expressed as the ratio of two polynomials, to partial fraction expansion, or residue, form. It also converts the partial fraction expansion back to the original polynomial coefficients.

[r,p,k] = residuez(b,a) finds the residues, poles, and direct terms of a partial fraction expansion of the ratio of two polynomials, b(z) and a(z). Vectors b and a specify the coefficients of the polynomials of the discrete-time system b(z)/a(z) in descending powers of z.

33. **angle (Phase angle) :**

P = angle(Z) returns the phase angles, in radians, for each element of complex array Z. The angles lie between + π and – π.

34. **log (Natural logarithm) :**

Y = log(X) returns the natural logarithm of the elements of X. For complex or negative z, where z = x +y*i , the complex logarithm is returned.

.

==========

LIST OF EXPERIMENT

- To develop elementary signal function modules (m-files) for unit sample, unit step, exponential and unit ramp sequences.
- To write a MATLAB program to compute linear convolution of two given Sequences.
- To develop program for computing circular convolution.
- To develop program for discrete convolution and correlation.
- To develop program for finding the response of the LTI system by difference equation.
- To develop program for computing inverse Z-transform.
- To develop program for finding the magnitude and phase response of LTI system described by the system function h(z).
- To develop program for computing discrete Fast Fourier Transform (FFT).
- To develop program for designing of Butterworth Low Pass IIR filter.
- To develop program for designing of Chebyshev -I High Pass IIR filter.

EXPERIMENT NO: I

➢ **AIM:** To develop elementary signal function modules (m-files) for unit sample, unit step, exponential and unit ramp, sine wave and square wave sequences.

➢ **SOFTWARE USED:** MATLAB software.

➢ **THEORY:**

- Unit sample $\delta(n) = 1$ for $n=0$ elsewhere $\delta(n) = 0$
- Unit step of sequence $[u(n)- u(n)-N]$

  Where $u(n) = 1$ for $n \geq 0$ elsewhere $n \neq 0$
- Ramp signal $r(n) = n$ for n $n \geq 0$
- Exponential signal $x(n) = e^{an} u(n)$

➢ **PROCEDURE:**

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

➢ **PROGRAM :**

```
% program for generation of unit sample
clc;
clear all;
 close all;
t = -3:1:3;
y = [zeros(1,3),ones(1,1),zeros(1,3)];
subplot(2,2,1);stem(t , y);
ylabel('Amplitude------>');
xlabel('(a)n ------>');
title('Unit Impulse Signal');
% program for genration of unit step of sequence [u(n)- u(n)-N]
t = -4:1:4;
```

```
y1 = ones(1,9);
subplot(2,2,2);stem(t,y1);
ylabel('Amplitude------>');
xlabel('(b)n ------>');
title('Unit step');
% program for generation of ramp signal
n1 = input('Enter the value for end of the sequence ');   %n1 = <any value>7 %
x = 0:n1;
subplot(2,2,3);
stem(x,x);
ylabel('Amplitude------>');
xlabel('(c)n ------>');
title('Ramp sequence');
% program for generation of exponential signal
n2 = input ('Enter the length of exponential seqeuence ');   %n2 = <any value>7 %
t = 0:n2;
a = input('Enter the Amplitude');    %a=1%
y2 = exp(a*t);
subplot(2,2,4);stem(t,y2);
ylabel('Amplitude------>');
xlabel('(d)n ------>');
title('Exponential sequence');
disp('Unit impulse signal');y
disp('Unit step signal');y1
disp('Unit Ramp signal');x
disp('Exponential signal');x
```

==========

==========

➢ OUTPUT:

Enter the value for end of the seqeuence = 6

Enter the length of exponential seqeuence = 4

Enter the Amplitude = 1

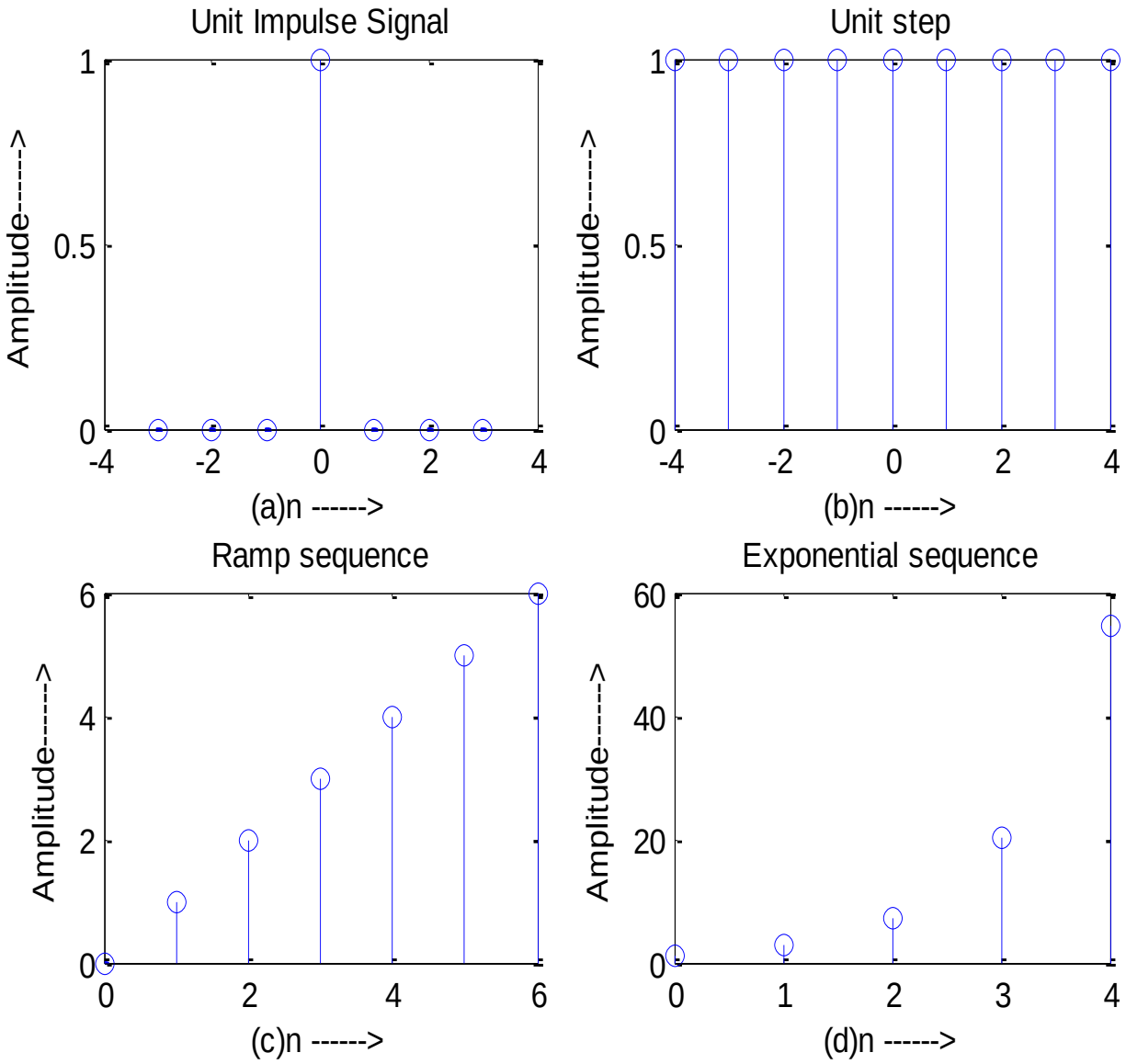
Unit impulse signal y = 0 0 0 1 0 0 0

Unit step signal y1 = 1 1 1 1 1 1 1 1 1

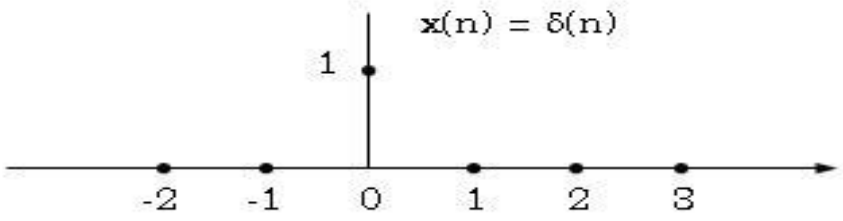
Unit Ramp signal x = 0 1 2 3 4 5 6

Exponential signal x = 0 1 2 3 4 5 6

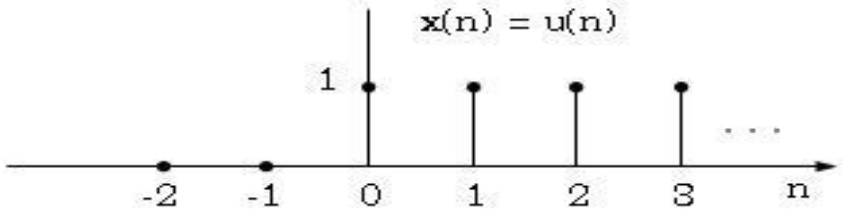
➢ SUMULATION RESULT:



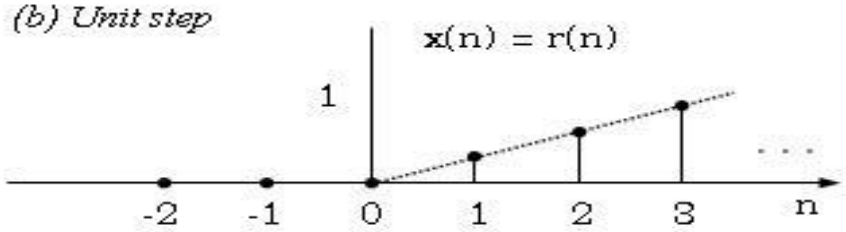
➢ VALIDATION OF RESULT:



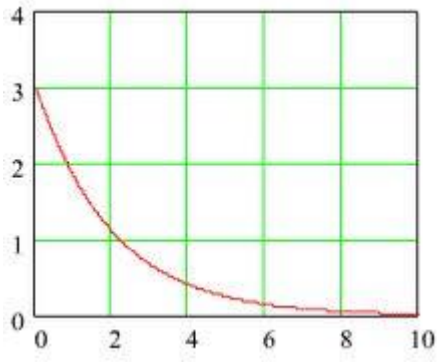
(a) Unit sample (Unit impulse)



(b) Unit step



(c) Unit ramp



(d)Exponential

➢ **CONCLUSION:** The simulation result of elementary signal function for unit sample, unit step, exponential and unit ramp sequences has been validated with the standard signal.

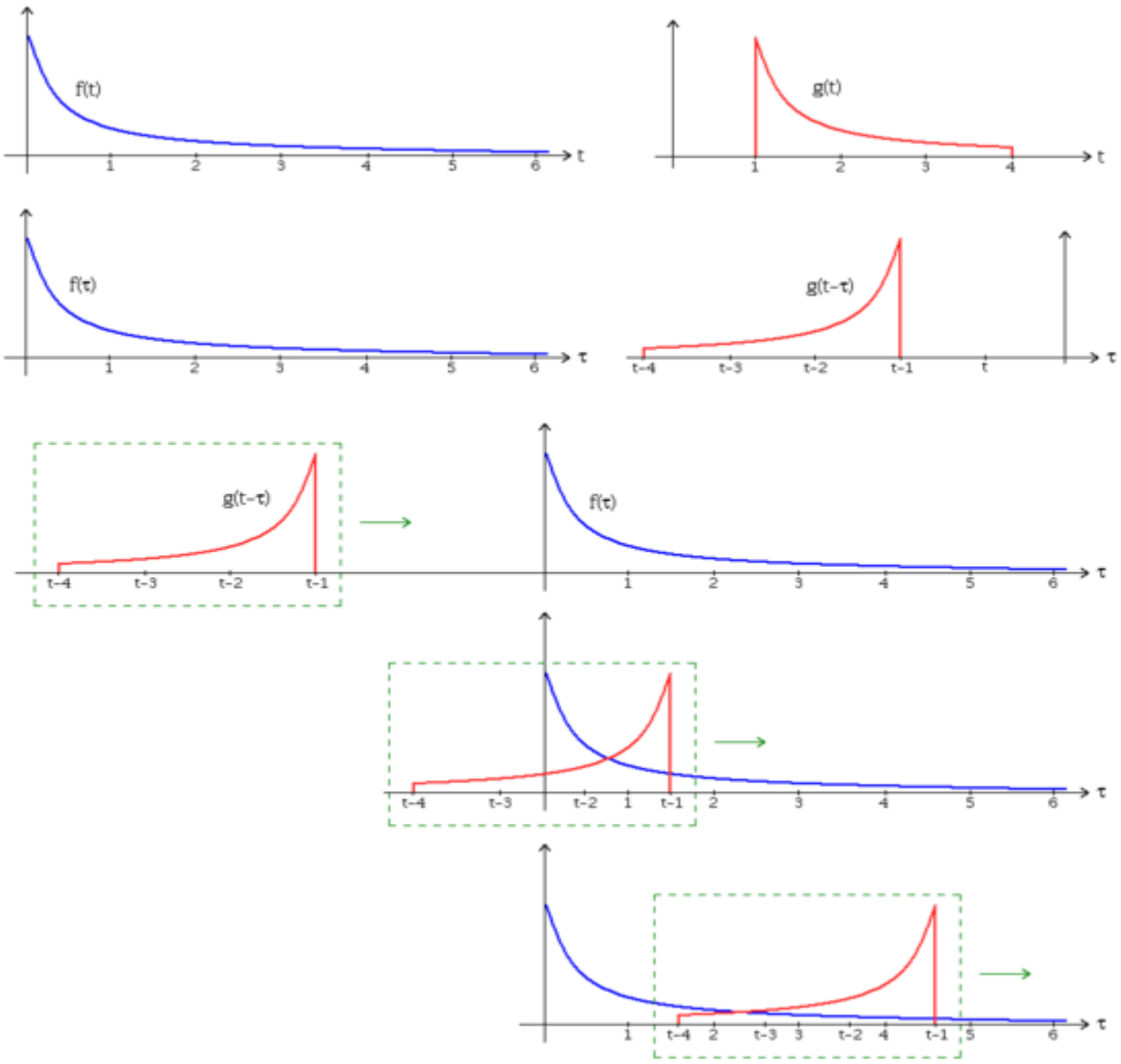
EXPERIMENT NO: II

- **AIM:** To write a MATLAB program to compute linear convolution of two given Sequences.

- **SOFTWARE USED:** MATLAB software version 7.5.0.

- **THEORY:** Convolution describes the output (in terms of the input) of an important class of operations known as linear time-invariant (LTI).

- **EXERCISE:**



➢ PROCEDURE:

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

➢ PROGRAM:

```
clc;
clear all;
close;
disp('enter the length of the first sequence m=');
m=input('');
disp('enter the length of first sequence x[m]=');
for i=1:m
 x(i)=input('');
end
disp('enter the length of the second sequence n=');
n=input('');
disp('enter the length of second sequence h[n]=');
for j=1:n
h(j)=input('');
end
y=conv(x,h);
figure;
subplot(3,1,1);
stem(x);
ylabel ('amplitude---->');
xlabel('n---->');
title('x(n) Vs n');
subplot(3,1,2);
stem(h);
ylabel('amplitude---->');
xlabel('n---->');
title('h(n) Vs n');
```

```
subplot(3,1,3);
stem(y);
ylabel('amplitude---->');
xlabel('n---->');
title('y(n) Vs n');
disp('linear convolution of x[m] and h[n] is y');
```

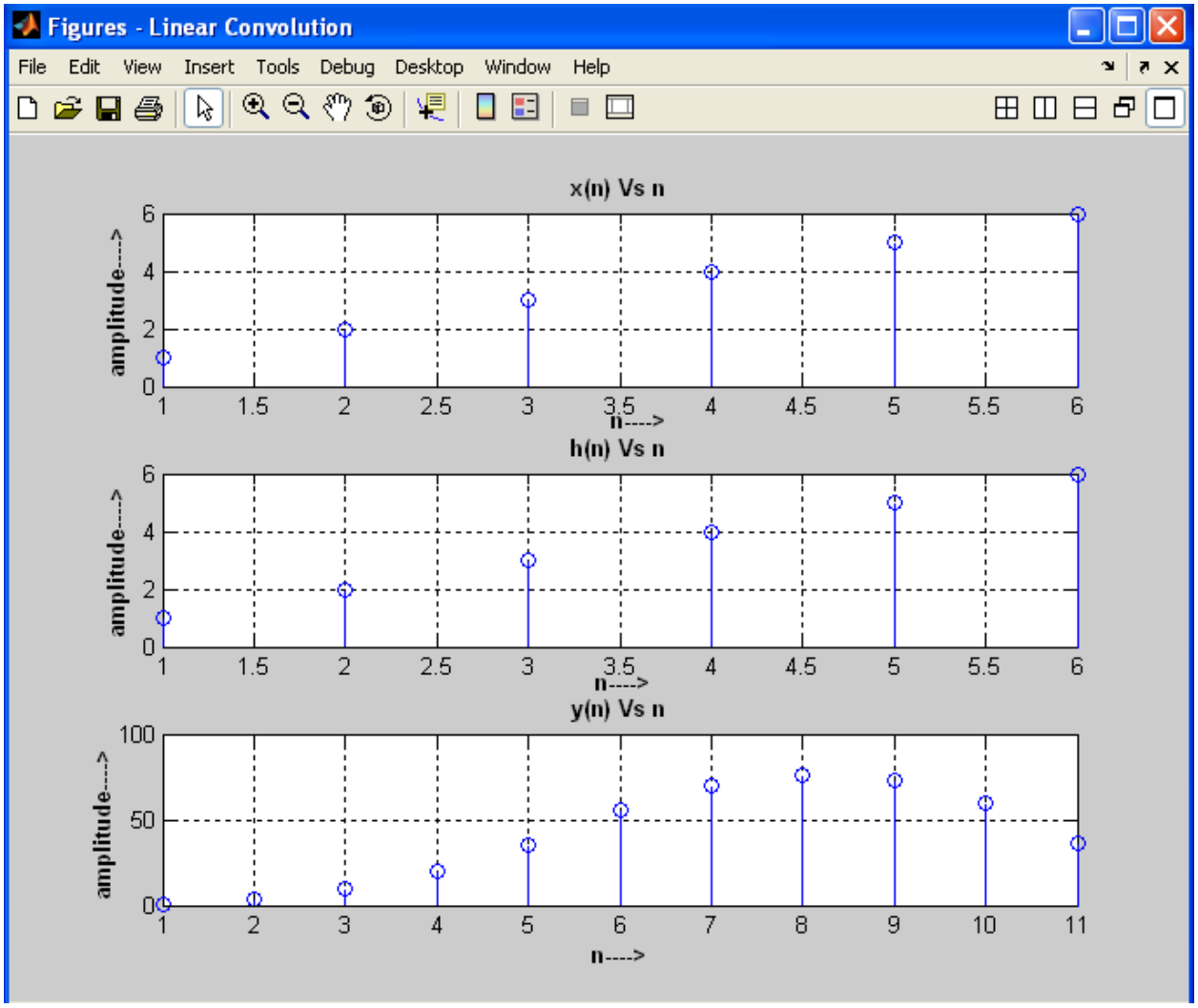
- **INPUT:**

```
Enter the length of the first sequence m=
6
Enter the length of first sequence x[m]=
1
2
3
4
5
6
Enter the length of the second sequence n=
6
Enter the length of second sequence h[n]=
1
2
3
4
5
6
```

- **OUTPUT:**

```
Linear convolution of x[m] and h[n] is y=
1 4 10 20 35 56 70 76 73 60 36
```

➢ **SUMULATION RESULT:**



➢ **VALIDATION OF RESULT:** It is done by using the following formulae:

$$\mathbf{Y(n)} = \sum_{n=-\infty}^{\infty} x(k).y(n-k)$$

Y(n) = {1, 4, 10, 20, 35, 56, 70, 76, 73, 60, 36 }

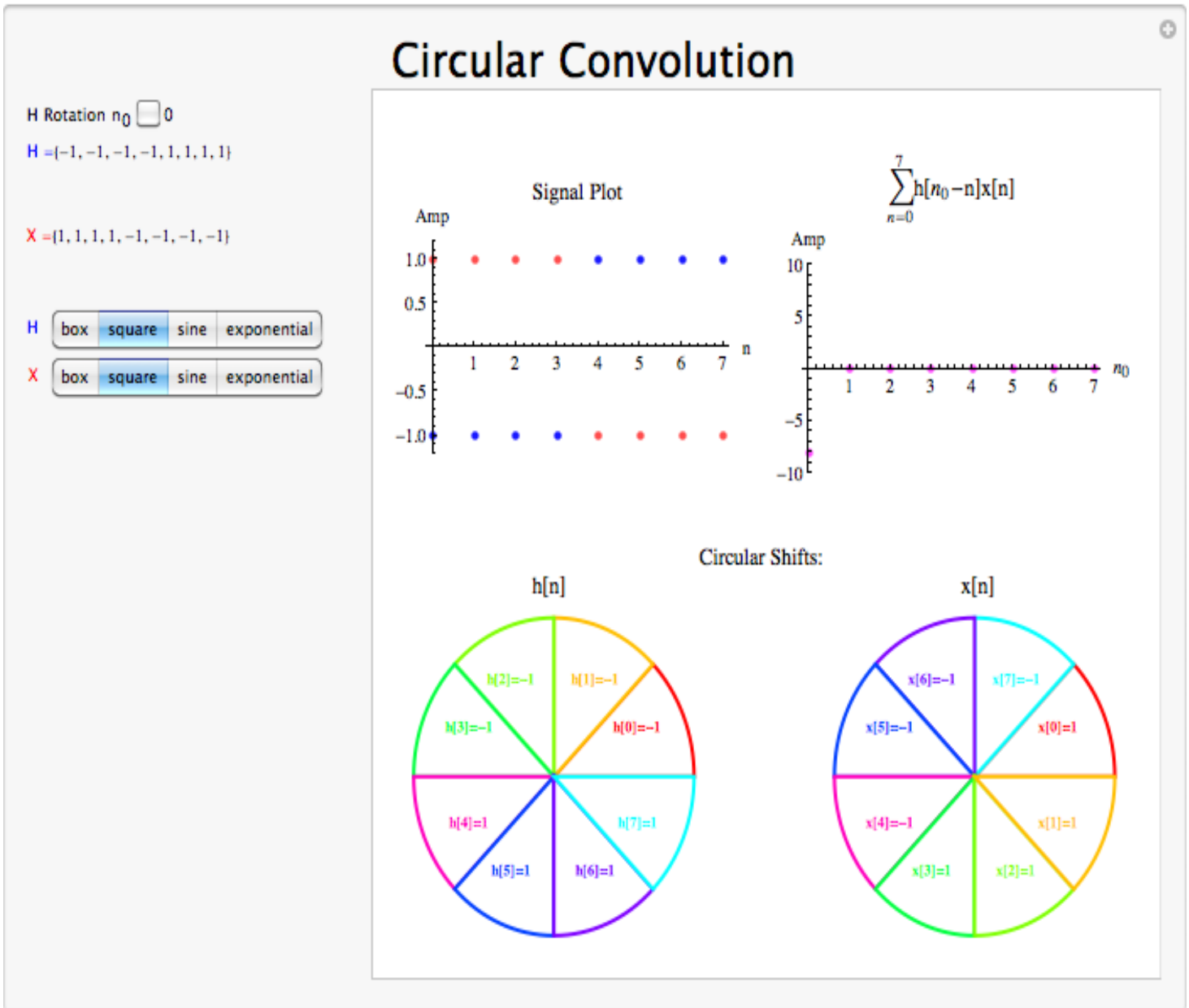
➢ **CONCLUSION:** The simulation result of linear convolution has been validated with the manually calculated result of two given Sequences.

EXPERIMENT NO: III

- **AIM:** To develop program for computing circular convolution.

- **SOFTWARE USED:** MATLAB software version R2022A

- **THEORY:** Convolution describes the output (in terms of the input) of an important class of operations known as linear time-invariant (LTI). Circular convolution is an another way to convolute two given sequence in circular form.



➢ PROCEDURE:

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

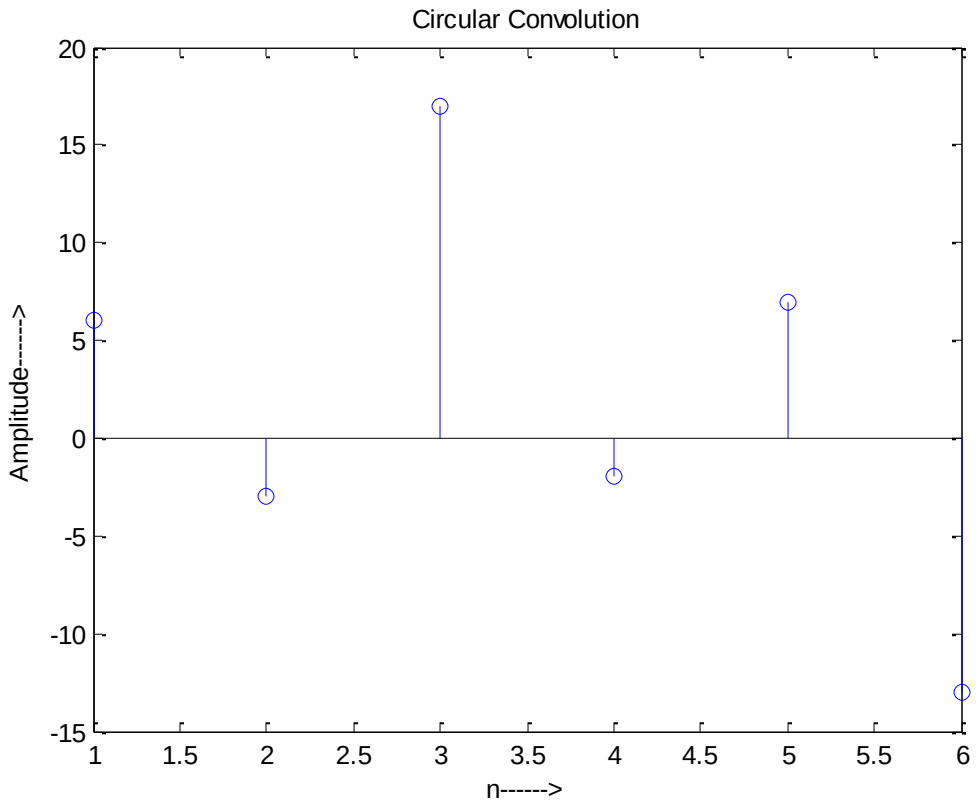
➢ PROGRAM:

```
%prog for computing circular convolution of sequence
clc;
clear all;
close all;
x=[1 -3 4 2 0 -2];
y=[3 0 1 -1 2 1];
n=4;
z=cconv(x,y,n);
subplot(3,1,1);
stem(x);
ylabel('amplitude');
xlabel('time');
subplot(3,1,2);
stem(y);
ylabel('amplitude');
xlabel('time');
subplot(3,1,3);
stem(z);
ylabel('amplitude');
xlabel('time');
disp('The result of Y');
```

➢ OUTPUT:

The resultant Signal is y = { 6 , -3, 17, -2, 7, -13 }

➢ **SIMULATION RESULT:**



➢ **VALIDATION OF RESULT:** It is done by using the following formulae:

$$\mathbf{Y(n)} = \sum_{n=0}^{N-1} x(n) \ . \ h\big(m-n, (mod\ N)\big), m = 0, 1, \ldots \ldots N-1$$

Y (n) = { 6 -3 17 -2 7 -13 }

➢ **CONCLUSION:** The simulation result of Circular convolution has been validated with the manually calculated result of two given Sequences.

EXPERIMENT NO: IV

- **AIM:** To develop program for discrete cross correlation.

- **SOFTWARE USED:** MATLAB software

- **THEORY:** Correlation gives a measure of similarity between two data sequences. In this process, two signal sequences are compared and the degree to which the two signals are similar is computed.

- **PROCEDURE:**

  - Open MATLAB
  - Open new M-file
  - Type the program
  - Save in current directory
  - Compile and Run the program
  - For the output see command window\ Figure window

- **PROGRAM:**

```
clc;
clear all;
close all;
x=[1 2 3 4];
y=[4 3 2 1];
z=xcorr(x,y);
subplot(3,1,1);
stem(x);
ylabel('amplitude');
xlabel('time');
subplot(3,1,2);
stem(y);
ylabel('amplitude');
xlabel('time');
subplot(3,1,3);
stem(z);
ylabel('amplitude');
xlabel('time');
disp('The result of z');
```

➢ OUTPUT:

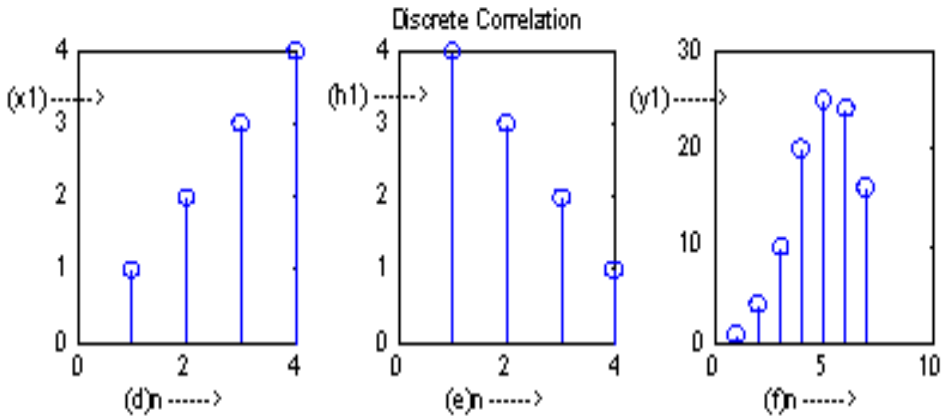
Correlation :

Enter the 1st sequence : [1 2 3 4]

Enter the 2nd sequence : [4 3 2 1]

Signal is : y1 = 1.0000 4.0000 10.0000 20.0000 25.0000 24.0000 16.000

➢ SIMULATION RESULT:



➢ **VALIDATION OF RESULT:** It is done by using the following formulae:

$$r(l) = \sum_{n=-\infty}^{\infty} x1(n) \; . \; x2(n-l)$$

$$\mathbf{R(z) = X_1(z) \; . \; X_2(z^{-1})}$$

$$\mathbf{r(l) = Y(n) = \{ 6 \quad -3 \quad 17 \quad -2 \quad 7 \quad -13 \}}$$

➢ **CONCLUSION:** The simulation result of cross correlation has been validated with the manually calculated result of two given Sequences.

EXPERIMENT NO: V

➢ **AIM:** To develop program for finding the response of the LTI system by difference equation.

➢ **APPARATUS:** PC having MATLAB software.

➢ **PROCEDURE:**

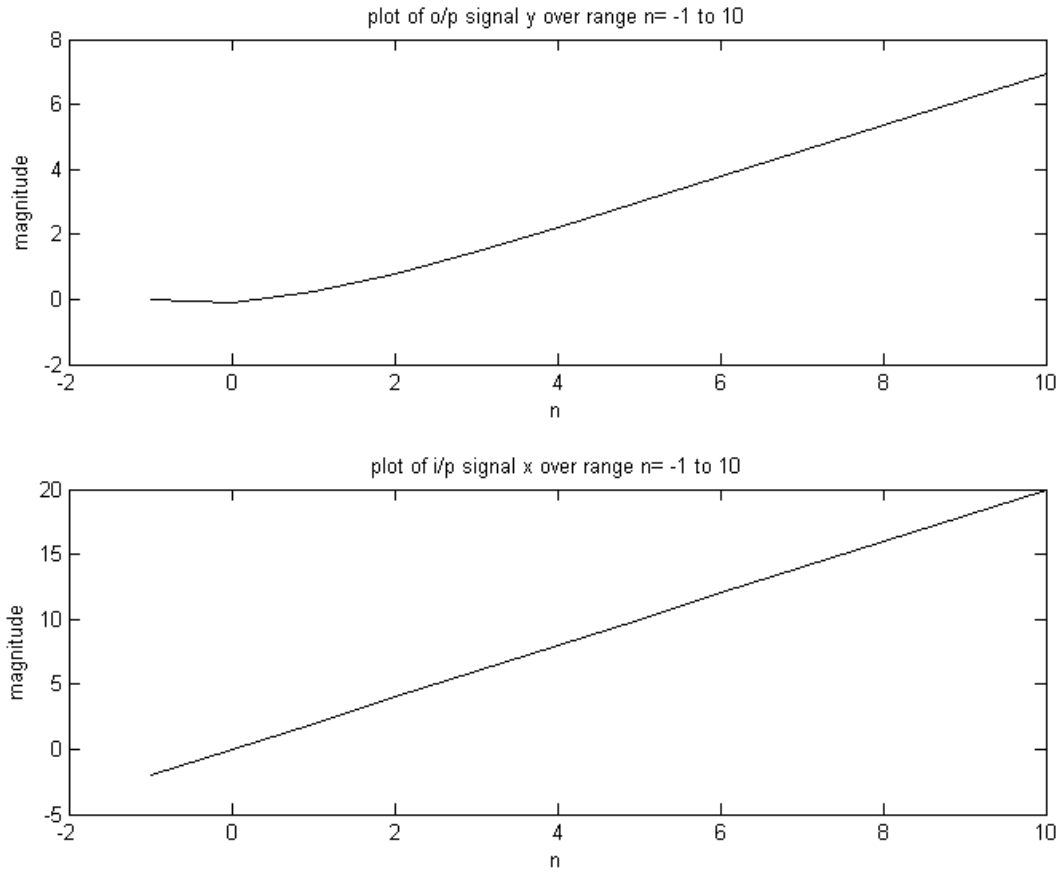
- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

➢ **PROGRAM:**

```
% prog for finding the response of LTI system by difference equation 8y(n) -2y(n-1)-y(n-2)=x(n)+x(n-1) plot impulse response h(n) & input x(n) If x(n)=2n.u(n)    & y(-1)=0 ,y(0)=1 ,the range is n=-1 to 10

clc;

clear all;

close all;

n = -1:10;

a = [8 -2 -1];   % left hand side of difference equation

b = [1 1 0]; % right hand side of difference equation

yi = [1 0];

xi = 0;

x=2*n;

zi = filtic(b,a,yi,xi);

y = filter(b,a,2*n,zi);

subplot(2,1,1);

plot(n,y);

xlabel('--------n----------->');

ylabel('-------Values------->');

title(' Plot of output signal y over the range from n=-1 to n=10');

subplot(2,1,2);
```

```
plot(n,x);
xlabel('--------n----------->');
ylabel('-------Values------->');
title(' Plot of input signal x over the range from n=-1 to n=10');
```

- **OUTPUT :**



EXPERIMENT NO: VI

➢ **AIM:** To develop the program for computing Z transform & inverse Z-transform.

➢ **APPARATUS:** PC having MATLAB software.

➢ **PROCEDURE:**

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

➢ **PROGRAM:**

%prog for computing the Z- transform of $x(n)=(1/4)^n u(n)$

```
clc;
clear all;
close all;
syms z n;
ztrans(1/4)^n
```

%prog for computing the inverse Z-transform of X(z) =2z /(2z-1)

```
clc;
clear all;
close all;
syms z n;
iztrans(2*z / (2*z-1))
```

➢ **OUTPUT :** $x(n)=(1/4)^n u(n)$ Thus, X(z) = 4*z /4*z – 1

X(z) =2z /(2z-1) Thus, $x(n) = (1/2)^n$

EXPERIMENT NO: VII

- **AIM:** To develop program for finding the magnitude and phase response of LTI system described by the system function h(z).

- **APPARATUS:** PC having MATLAB software.

- **PROCEDURE:**

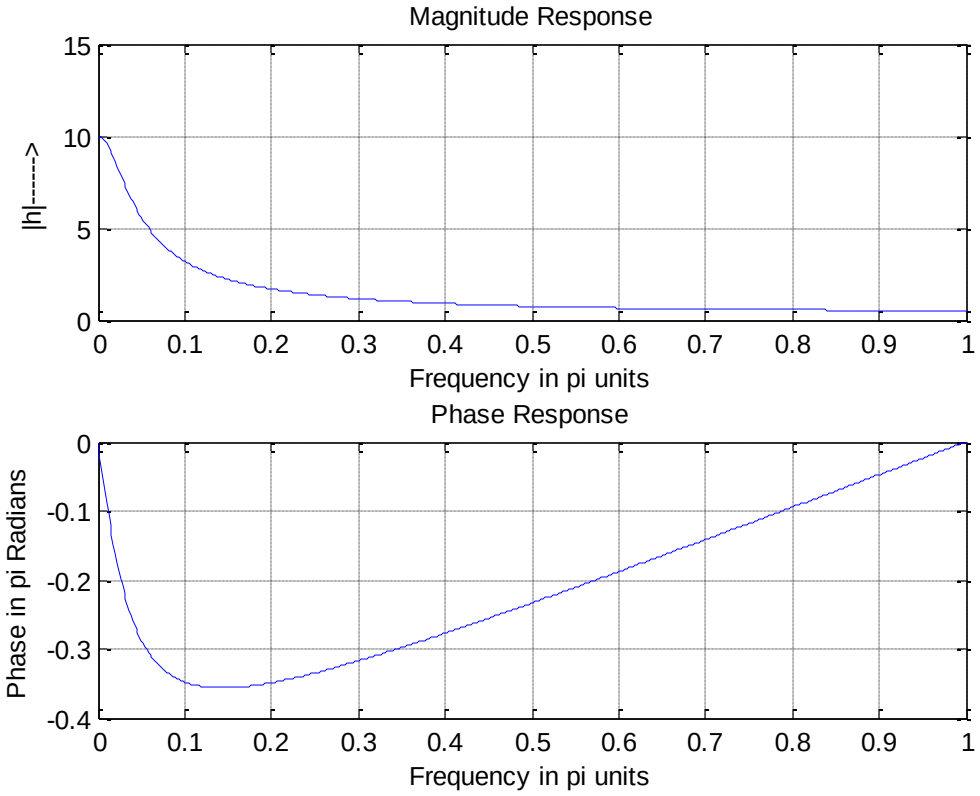
  - Open MATLAB
  - Open new M-file
  - Type the program
  - Save in current directory
  - Compile and Run the program
  - For the output see command window\ Figure window

- **PROGRAM:**

```
%prog for finding the magnitude and phase response of LTI system by
%h(n) =(0.9)^n*u(n)
w =[0:1:500]*pi/500;    %[0,pi] axis divided into 501 points
h =exp(j*w)./(exp(j*w)-0.9*ones(1,501));
magh =abs(h); angh =angle(h);
subplot(2,1,1);plot(w/pi,magh);grid;
ylabel('|h|------>');
xlabel('(Frequency in pi units');
title('Magnitude Response');
subplot(2,1,2);plot(w/pi,angh/pi);grid;
ylabel('Phase in pi Radians');
xlabel('(Frequency in pi units');
title('Phase Response');
```

- **OUTPUT:**

- **GRAPH:**


Magnitude Response
|h|----->
15
10
5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Frequency in pi units
Phase Response
Phase in pi Radians
0
-0.1
-0.2
-0.3
-0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Frequency in pi units

EXPERIMENT NO: VIII

➢ **AIM:** To develop program for computing discrete Fast Fourier Transform (FFT) .

➢ **APPARATUS:** PC having MATLAB software.

➢ **PROCEDURE:**

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

➢ **PROGRAM:**

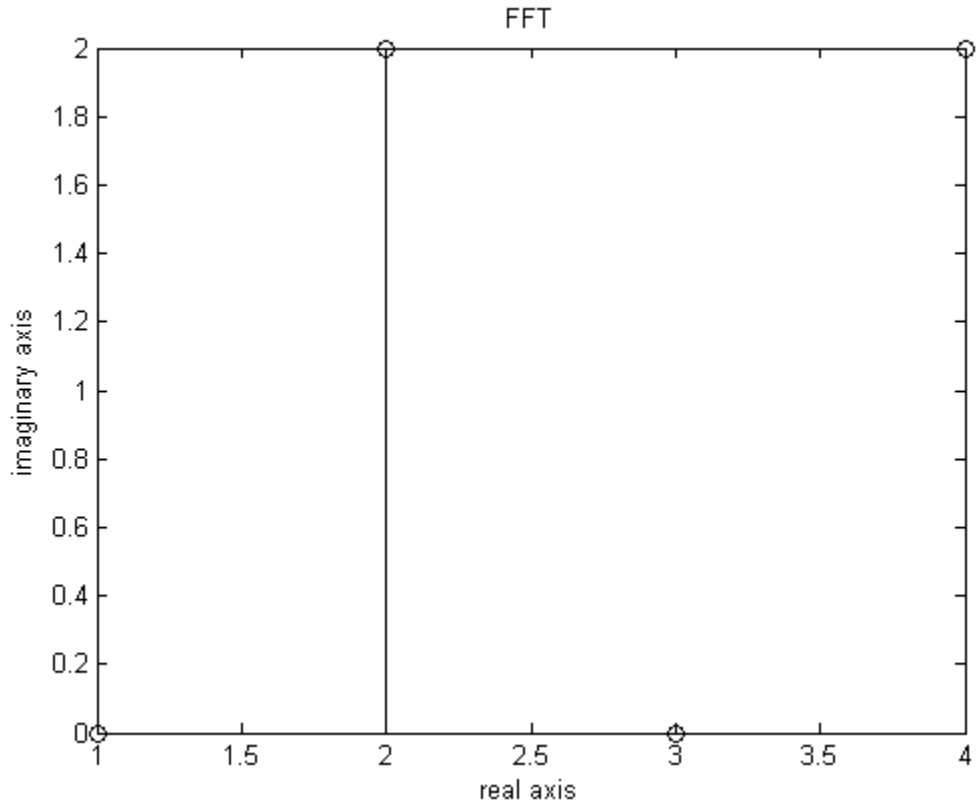
```
%prog for computing Fast Fourier Transform
clc;
clear all;
close all;
x=input('enter the sequence');
n=input('enter the length of the fft');
X=fft(x,n);
stem(X);
ylabel('imaginary axis');
xlabel('real axis');
title('FFT');
```

➢ **Input :**

enter the sequence [1 0 -1 0]

enter the length of the fft [4]

- **Graph:**


FFT
imaginary axis
real axis
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
1.5
2
2.5
3
3.5
4

EXPERIMENT NO: IX

➢ **AIM:** To develop program for designing FIR filter.

➢ **APPARATUS:** PC having MATLAB software.

➢ **PROCEDURE:**

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

➢ **PROGRAM:**

```
%prog for designing of FIR low pass
% filters using rectangular window
clc;clear all;close all;
rp =input('Enter the pass band ripple ');
rs =input('Enter the stop band ripple ');
fp =input('Enter the pass band freq ');
fs =input('Enter the stop band freq ');
f =input('Enter the sampling freq ');
wp = 2*fp/f;
ws = 2*fs/f;
num = -20*log10(sqrt(rp*rs))-13;
dem = 14.6*(fs-fp)/f;
n = ceil(num/dem);
n1 = n+1;
if (rem(n,2)~=0)
    n1 =n;
    n = n-1;
end
```

```
y = boxcar(n1);
% low pass filter
b = fir1(n,wp,y);
[h,o] = freqz(b,1,256);
m = 20*log(abs(h));
subplot(2,2,1);plot(o/pi,m);
ylabel('Gain in dB ------>');
xlabel('(a)   Normalised freq ---->');
```

Output :

Enter the pass band ripple 0.05

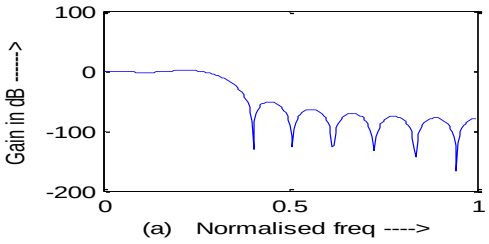
Enter the stop band ripple 0.04

Enter the pass band freq 1500

Enter the stop band freq 2000

Enter the sampling freq 9000

Graph:



EXPERIMENT NO: IX

➢ **AIM:** To develop program for designing of Butterworth Low Pass IIR filter.

➢ **APPARATUS:** PC having MATLAB software.

➢ **PROCEDURE:**

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

➢ **PROGRAM:**

```
% prog for designing of IIR low pass
clc;
clear all;
close all;
rp=input('enter the PB ripple');
rs=input('enter the SB ripple');
wp=input('enter the PB frequency');
ws=input('enter the SB frequency');
fs=input('enter the sampling frequency');
w1=2*wp/fs;
w2=2*ws/fs;
[n,wn]=buttord(w1,w2,rp,rs);
[b,a]=butter(n,wn);
w=0:0.01:pi;
[h,om]=freqz(b,a,w);
m=20*log(abs(h));
an=angle(h);
subplot(2,1,1);
plot(om/pi,m);
ylabel('gain in dB');
xlabel('(a) normalised frequency');
```

```
title('Butterworth digital IIR LPF');
subplot(2,1,2);
plot(om/pi,an);
ylabel('phase in radian');
xlabel('(b) normalised frequency');
```

Input :

enter the PB ripple.5

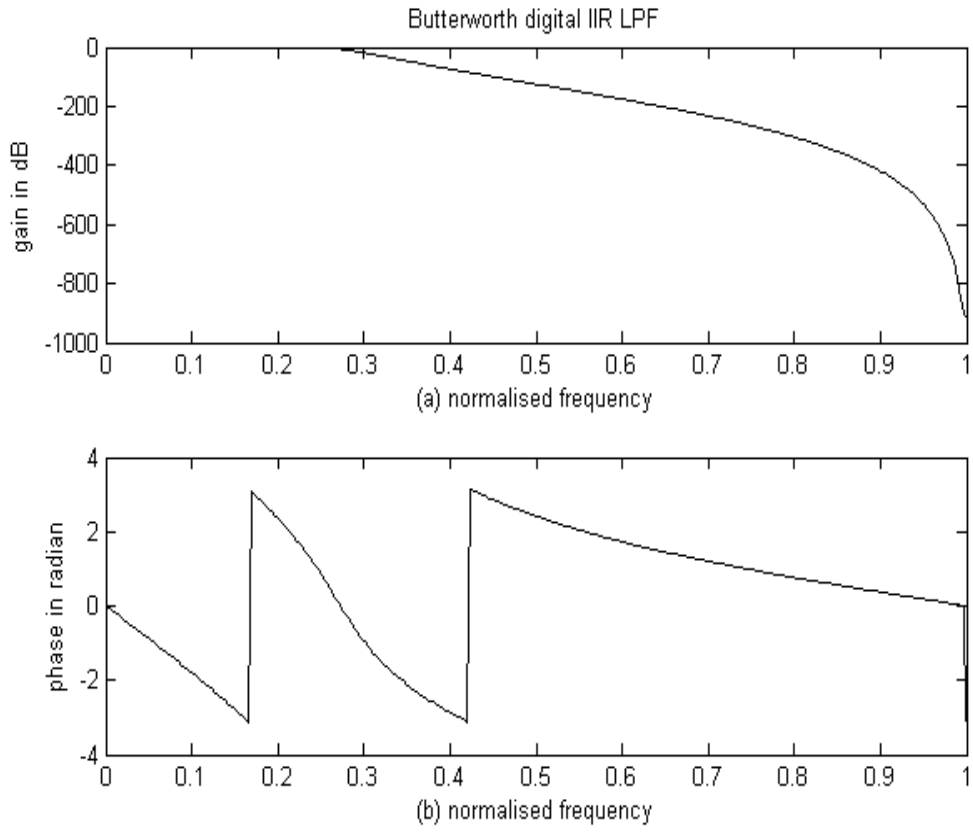
enter the SB ripple50

enter the PB frequency1200

enter the SB frequency2400

enter the sampling frequency10000

Graph :



\

EXPERIMENT NO: X

➢ **AIM:** To develop program for designing of Chebyshev -I High Pass IIR filter.

➢ **APPARATUS:** PC having MATLAB software.

➢ **PROCEDURE:**

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

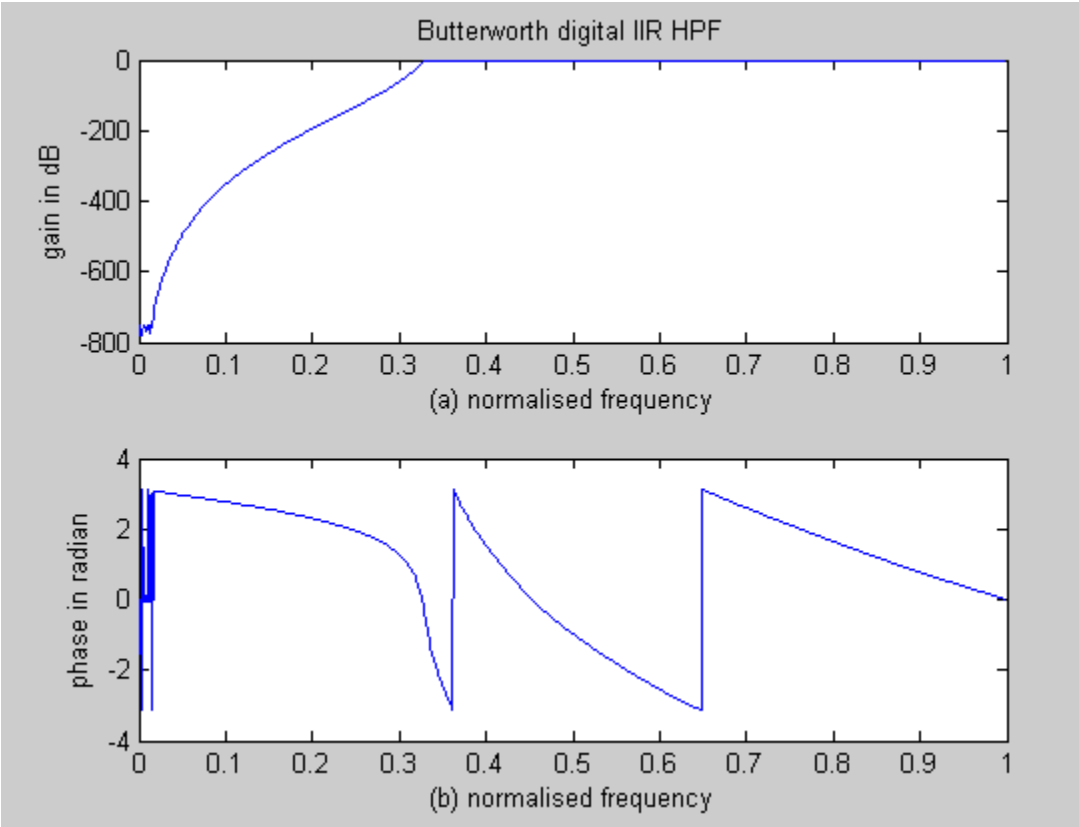
➢ **PROGRAM:**

```
% program for the design of Chebysev type-1 high pass digital filter
clc;
clear all;
close all;
rp=input('enter the PB ripple');
rs=input('enter the SB ripple');
wp=input('enter the PB frequency');
ws=input('enter the SB frequency');
fs=input('enter the sampling frequency');
w1=2*wp/fs;
w2=2*ws/fs;
[n,wn]=cheb1ord(w1,w2,rp,rs);
[b,a]=cheby1(n,rp,wn,'high');
w=0:0.01/pi:pi;
[h,om]=freqz(b,a,w);
m=20*log(abs(h));
an=angle(h);
subplot(2,1,1);
plot(om/pi,m);
ylabel('gain in dB');
xlabel('(a) normalised frequency');
```

```
title('Butterworth digital IIR HPF');
subplot(2,1,2);
plot(om/pi,an);
ylabel('phase in radian');
xlabel('(b) normalised frequency');
```

- **Input :** enter the PB ripple0.3
  enter the SB ripple60
  enter the PB frequency1500
  enter the SB frequency2000
  enter the sampling frequency9000

- **Graph :**



fft

```
x=input('enter the sequence');
n=input('enter the length of the fft');
y=fft(x,n);
stem(y);
```

write in command window

```
x=input('enter the sequence');
n=input('enter the length of the fft');
y=fft(x,n)
```

input :

enter the sequence[1 2 3 4]

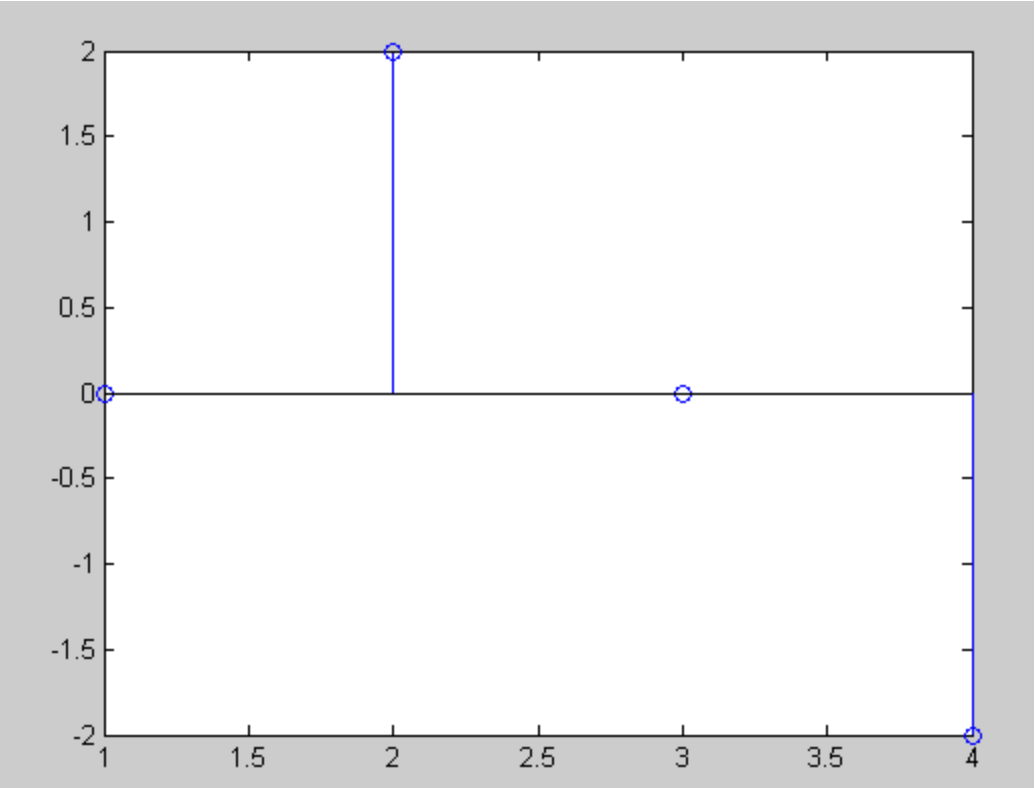
enter the length of the fft 4

output :

y =

10.0000 -2.0000 + 2.0000i -2.0000 -2.0000 - 2.0000i

Graph shows the imaginary values


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
1
1.5
2
2.5
3
3.5
4