Name :- Nitish Singh

Roll no :- 1037

Class :- SYCS

Div :- B

Subject :- Digital Signal Processing

| Sr.No | Title | Date | Remark |
| --- | --- | --- | --- |
| 1 | WRITE A SCILAB PROGRAM TO GENERATE COMMON DISCRETE TIME SIGNALS. | 23-12-2023 |  |
| 2 | A SCILAB PROGRAM TO OBSERVE THE EFFECTS OF LOWER SAMPLING RATE AND HIGHER SAMPLING RATE ON C.T. SIGNAL (SAMPLING THM. ) | 16-1-2024 |  |
| 3 | WRITE A SCILAB PROGRAM TO COMPUTE LINEARITY PROPERTY OF A GIVEN SIGNAL | 6-1-2024 |  |
| 4 | WRITE A SCILAB PROGRAM TO COMPUTE LINEAR CONVOLUTION OF TWO SEQUENCES USING BASIC EQUATION(Without using conv function) | 6-1-2024 |  |
| 5 | WRITE A SCILAB PROGRAM TO FIND AUTOCORRELATION AND CROSS CORRELATION OF THE GIVEN SEQUENCES | 13-1-2024 |  |
| 6 | WRITE A SCILAB PROGRAM TO FIND N-POINT DFT OF THE GIVEN SEQUENCES | 13-1-2024 |  |
| 7 | WRITE A SCILAB PROGRAM TO COMPUTE LINEAR CONVOLUTION OF TWO SEQUENCES USING DFT BASED APPROACHES | 20-1-2024 |  |
| 8 | WRITE A SCILAB PROGRAM TO COMPUTE CIRCULAR CONVOLUTION OF TWO SEQUENCES USING BASIC EQUATION(Without using the library function) | 20-1-2024 |  |
| 9 | WRITE A SCILAB PROGRAM TO COMPUTE CIRCULAR CONVOLUTION OF THE TWO SEQUENCES USING DFT BASED APPROACH | 20-1-2024 |  |
| 10 | WRITE A SCILAB PROGRAM TO FIND FFT USING DECIMATION IN TIME(DIT) METHOD | 27-1-2024 |  |
| 11 | STUDY TIME AND FREQUENCY RESPONSE OF LTI SYSTEM | 24-2-2024 |  |
| 12 | WRITE A SCILAB PROGRAM FOR DESIGNING OF FIR FILTER FOR LOW PASS,HIGH PASS,BANDPASS AND BAND REJECT RESPONSES | 24-2-2024 |  |
| 13 | WRITE A SCILAB PROGRAM TO DESIGN DIGITAL IIR BUTTERWORTH LOW PASS FILTER | 24-2-2024 |  |
| 14 | WRITE A SCILAB PROGRAM TO DESIGN DIGITAL IIR CHEBYSHEW FILTER | 27-1-2024 |  |

Practical 1

Aim:Write a scilab program to generate common discrete time signals

1. Unit Sample Sequence
2. Unit Step Sequence
3. Discrete Ramp Sequence
4. Exponentially Decreasing Signal
5. Exponetially Increasing Signal
6. **Sinusoidal Signal**

1.Unit Sample Sequence:

Theory:A unit sample sequence, also known as an impulse sequence or delta sequence, is a discrete sequence that consists of a single sample with the value of 1 at a specific index, and all other samples are zero. It is commonly represented as a discrete-time impulse function or delta function.

Code:

L=4;

n=-L:L;

x=[zeros(1,L),1,zeros(1,L)];

b=gca();

b.y\_location="middle";

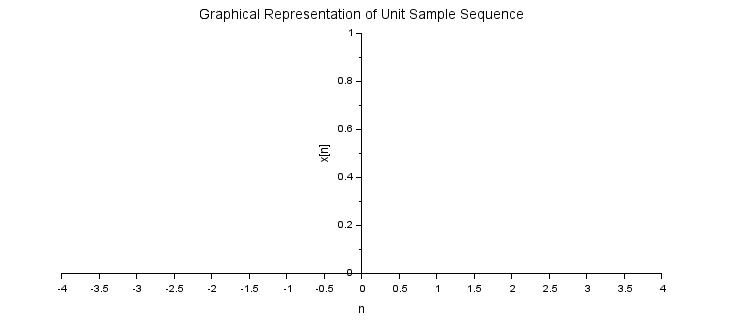
subplot(221);

plot2d3('gnn',n,x)

a.children(1).thickness=5;

xtitle('Graphical Representation of Unit Sample Sequence','n','x[n]');

Output:



2. Unit Step Sequence:

Theory: The step signal or step function is that type of standard signal which exists only for positive time and it is zero for negative time. In other words, a signal x(t) is said to be a step signal if and only if it exists for t > 0 and zero for t < 0. The step signal is an important signal used for analysis of many systems.

The unit step signal which is defined only at discrete instants of time is known as discrete-time unit step signal. It is denoted by u(n). Mathematically, the discrete-time unit step signal or sequence u(n) is defined as follows:-

u(n)={1 for n>=0,0 for n<0}

Code:

L=4;

n=-L:L;

x=[zeros(1,L),ones(1,L+1)];

a=gca();

a.y\_location="middle";

subplot(222);

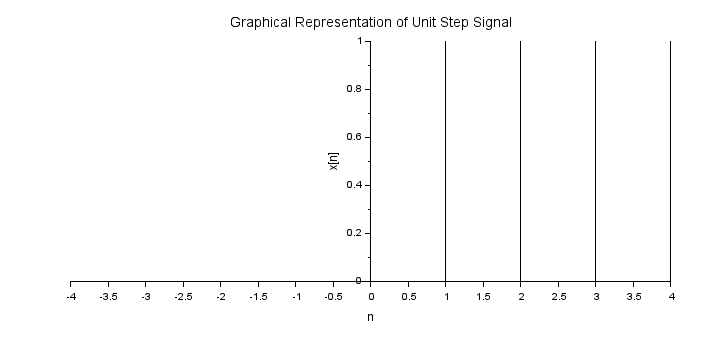
plot2d3('gnn',n,x)

title('Graphical Representation of Unit Step Signal')

xlabel('n');

ylabel('x[n]');

Output:



3.Discrete Ramp Sequence

Theory: Unit ramp signals are fundamental building blocks in the realm of digital signal processing.The unit ramp signal is a ramp signal with a slope value equal to 1.

The discrete-time unit ramp signal is defined for the fixed instances of time with definite time intervals. The discrete-time unit ramp signal is represented as r(n).

It is defined as (Equation of Discrete Ramp Signal )-

r(n)=n,n>=0

=0,n<0

Code :

L=4;

n=-L:L;

x=[zeros(1,L),0:L];

b=gca();

b.y\_location="middle";

subplot(223);

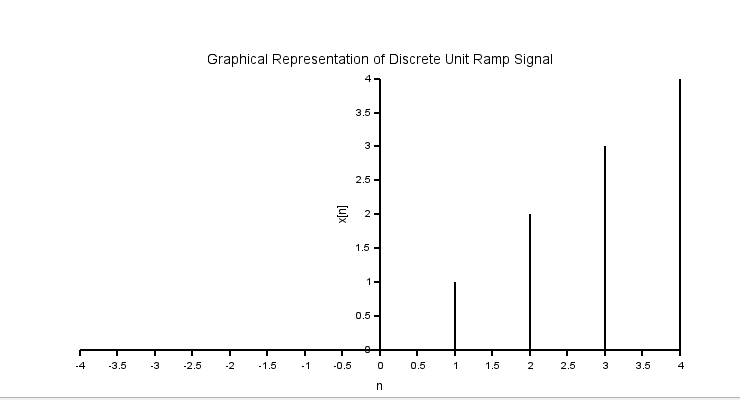
plot2d3('gnn',n,x)

a=gce();

a.children(1).thickness=2;

xtitle('Graphical Representation of Discrete Unit Ramp Signal','n','x[n]');

Output:



4.Exponentially Decreasing Signal

Theory:A [quantity](https://en.wikipedia.org/wiki/Quantity) is subject to **exponential decay** if it decreases at a rate [proportional](https://en.wikipedia.org/wiki/Proportionality_(mathematics)) to its current value. Symbolically, this process can be expressed by the following [differential equation](https://en.wikipedia.org/wiki/Differential_equation), where *N* is the quantity and *λ* ([lambda](https://en.wikipedia.org/wiki/Lambda)) is a positive rate called the **exponential decay constant**, **disintegration constant**,[[1]](https://en.wikipedia.org/wiki/Exponential_decay#cite_note-1) **rate constant**,[[2]](https://en.wikipedia.org/wiki/Exponential_decay#cite_note-2) or **transformation constant**:[[3]](https://en.wikipedia.org/wiki/Exponential_decay#cite_note-3)



The solution to this equation (see [derivation](https://en.wikipedia.org/wiki/Exponential_decay#Solution_of_the_differential_equation) below) is:



where *N*(*t*) is the quantity at time *t*, *N*0 = *N*(0) is the initial quantity, that is, the quantity at time *t* = 0.

Code:

a=0.5;

n=0:10;

x=(a)^n;

a=gca();

a.x\_location="origin";

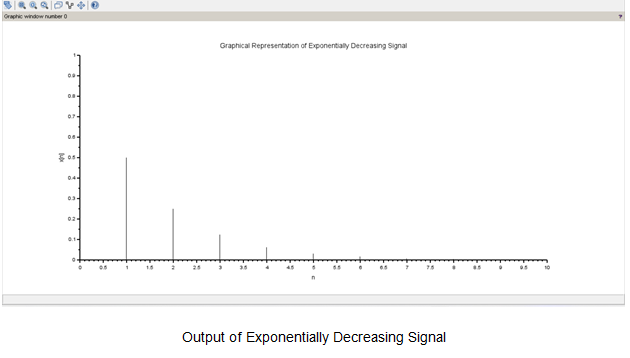
a.y\_location="origin";

plot2d3('gnn',n,x)

a.thickness=2;

xtitle('Graphical Representation of Exponentially Decreasing Signal','n','x[n]');

Output:



5.Exponentially Increasing Signal

Theory:

**Exponential growth** is a process that increases quantity over time at an ever-increasing rate. It occurs when the instantaneous [rate of change](https://en.wikipedia.org/wiki/Rate_(mathematics)#Of_change) (that is, the [derivative](https://en.wikipedia.org/wiki/Derivative)) of a quantity with respect to time is [proportional](https://en.wikipedia.org/wiki/Proportionality_(mathematics)) to the quantity itself. Described as a [function](https://en.wikipedia.org/wiki/Function_(mathematics)), a quantity undergoing exponential growth is an [exponential function](https://en.wikipedia.org/wiki/Exponentiation#Power_functions) of time, that is, the variable representing time is the exponent (in contrast to other types of growth, such as [quadratic growth](https://en.wikipedia.org/wiki/Quadratic_growth)). Exponential growth is [the inverse](https://en.wikipedia.org/wiki/Inverse_function) of [logarithmic growth](https://en.wikipedia.org/wiki/Logarithmic_growth).

If the constant of proportionality is negative, then the quantity decreases over time, and is said to be undergoing [exponential decay](https://en.wikipedia.org/wiki/Exponential_decay) instead. In the case of a discrete [domain](https://en.wikipedia.org/wiki/Domain_of_a_function) of definition with equal intervals, it is also called **geometric growth** or **geometric decay** since the function values form a [geometric progression](https://en.wikipedia.org/wiki/Geometric_progression).

The formula for exponential growth of a variable *x* at the growth rate *r*, as time *t* goes on in discrete intervals (that is, at integer times 0, 1, 2, 3, ...), is



Code:

a=1.5;

n=0:10;

x=(a)^n;

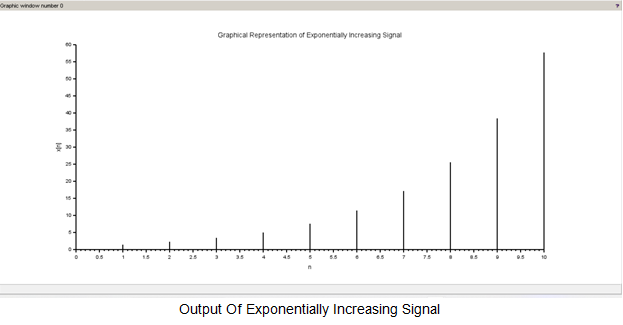
a=gca();

a.thickness=2;

plot2d3('gnn',n,x)

xtitle('Graphical Representation of Exponentially Increasing Signal','n','x[n]');

Output:



6.Sinusoidal Signal

Theory:A **sinusoidal wave signal** is a type of periodic signal that oscillates (moves up and down), periodically. The geometrical waveform of a sinusoidal signal forms an S-shape wave in one complete cycle. A sinusoidal can be a sine functioned signal or cosine functioned signal. Thus, a sinusoidal signal can be defined as,



Both sine and cosine signals are the types of sinusoidal wave signals. But, the cosine signal is advanced with respect to the sine signal by 90° in time. The sinusoidal wave signal has a smooth wave that oscillates above and below zero and used in technical analysis of systems.

Code:

t=0:0.04:1;

x=sin(2\*%pi\*t);

subplot(2,4,6);

a=gca();

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

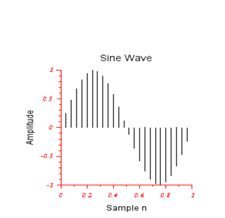
plot2d3(t,x);

title('Sine Wave')

xlabel('Sample n');

ylabel('Amplitude’');

Output:



Conclusion:

we understood and successfully generate the common discrete time signals

Practical 2

Aim: A scilab program to observe the effects of lower sampling rate and higher sampling rate to C.T SIGNAL(Sampling THM)

Theory:

Sampling is the process of converting a continuous-time signal into a

discrete-time signal by taking measurements at regular intervals. The rate at

which these measurements are taken is known as the sampling rate or sampling

frequency. It is usually denoted by Fs and is measured in Hertz (Hz).

The Nyquist-Shannon sampling theorem states that to perfectly reconstruct a

continuous-time signal from its samples, the sampling rate must be at least twice

the highest frequency component of the signal. This minimum rate is known as

the Nyquist rate.

Effects of Different Sampling Rates:

1. Lower Sampling Rate: If the sampling rate is less than the Nyquist rate

(under-sampling), high-frequency components in the signal may become

indistinguishable from low-frequency components, leading to an effect

known as aliasing. This can significantly distort the reconstructed signal.

2. Higher Sampling Rate: If the sampling rate is much higher than the Nyquist

rate (over-sampling), the signal can be perfectly reconstructed. However,

this comes at the cost of increased data storage requirements and

computational complexity.

In the Scilab program provided, a sinusoidal signal is sampled at two different

rates: a lower rate (Ts\_low) and a higher rate (Ts\_high). The effects of these

different sampling rates on the signal are then visualized.

Code:

t = 0:0.001:1;

f = 5;

ctSignal = sin(2\*%pi\*f\*t);

Ts\_low = 0.1;

t\_low = 0:Ts\_low:1;

lowSampled = sin(2\*%pi\*f\*t\_low);

Ts\_high = 0.01;

t\_high = 0:Ts\_high:1;

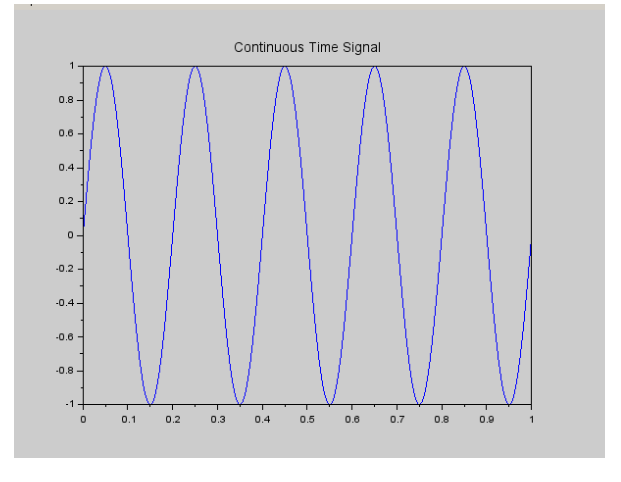
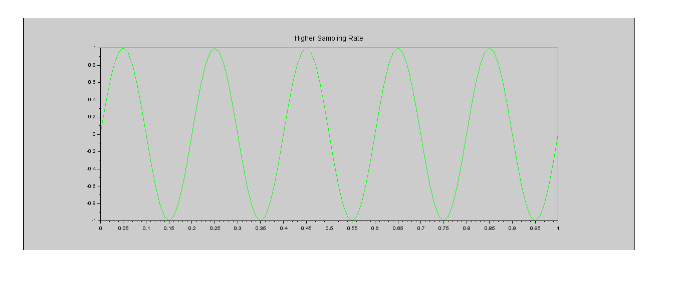
highSampled = sin(2\*%pi\*f\*t\_high);

figure, plot(t, ctSignal, 'b'), title('Continuous Time Signal')

figure, plot(t\_low, lowSampled, 'r'), title('Lower Sampling Rate')

figure, plot(t\_high, highSampled, 'g'), title('Higher Sampling Rate')

Output:

Conclusion:In the experiment, we observe different effects of amplitude and frequency on signals.

                                 Practical 3

Aim: Write a scilab program to compute linearity property of a given signal.

Theory: A system H is said to be linear if it satisfies two important conditions. The first, additivity, states for every pair of signals

x,y that H(x+y)=H(x)+H(y)

The second, homogeneity of degree one, states for every signal x and scalar a we have

H(ax)=aH(x)

It is clear that these conditions can be combined together into a single condition for linearity. Thus, a system is said to be linear if for every signals x,y and scalars a and b

we have that H(ax+by)=aH(x)+bH(y).

Linearity is a particularly important property of systems as it allows us to leverage the powerful tools of linear algebra, such as bases, eigenvectors, and eigenvalues, in their study.

Code:

n=0:10;

a=2;

b=3;

x1=sin(2\*%pi\*0.1\*n);

x2=cos(2\*%pi\*0.5\*n);

x=a.\*x1+b.\*x2;

y =0.5\*x;

y1=0.5\*x1;

y2=0.5\*x2;

yt=a.\*y1+b.\*y2;

d=y-yt ;

disp('Output of System y(n)=0.5x(n)is:');

if(d==0)

    disp('System is Linear');

else

    disp('System is Non-Linear');

end

n=0:10;

a=2;

b=3;

x1=sin(2\*%pi\*0.1\*n);

x2=cos(2\*%pi\*0.5\*n);

x=a.\*x1+b.\*x2;

y=sqrt(x);

y1=sqrt(x1);

y2=sqrt(x2);

yt=a.\*y1+b.\*y2 ;

d=y-yt ;

disp ('Output of System y(n)=sqrt(x(n))is:');

if(d==0)

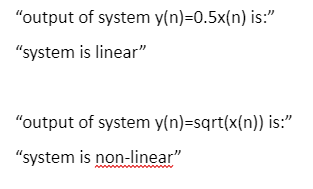
    disp('System is Linear');

else

    disp('System is Non-Linear');

end

Output:



Conclusion:

In this experiment,we study how to classify the signal as linear or non-linear signal using linearity property.

                               Practical 4

Aim: Write a scilab program to compute linear convolution of two sequences using basic equation.

Theory: In implementing discrete-time system, we need to compute the [convolution sum](https://www.sciencedirect.com/topics/engineering/convolution-sum), otherwise called linear convolution, of the input signal *x*[*n*] and the impulse response *h*[*n*] of the system. For finite duration sequences, this convolution can be carried out using DFT computation.

Let *x*[*n*] and *h*[*n*] be of finite duration. Assume *x*[*n*] is zero outside the interval 0 ≤ *n* ≤ *N* − 1 and *h*[*n*] is zero outside the interval 0 ≤ *n* ≤ *M* − 1.

The sequence *y*[*n*] is the linear convolution between *x*[*n*] and *h*[*n*]:

           ∞

   y[n]=x[n]\*ℎ[n]=Σ=x[k]ℎ[n−k],

                 k=-∞

Code:

x=input('Enter input sequence x(n)=')

m=length(x);

xl=input('Enter the lower index of input sequence = ')

xh=xl+m-1;

n=xl:1:xh;

subplot(3,1,1);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot2d3('gnn',n,x);

title('Input sequence [n]');

xlabel('Samples n');

ylabel('Amplitude');

h=input('Enter the Impulse response sequence h(n) = ');

l=length(h);

hl=input('Enter lower index of Impulse response sequence =');

hh=hl+l -1;

g=hl:1:hh;

subplot(3,1,2);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot2d3('gnn',n,h);

title('Impulse response sequence [n]');

xlabel ('Samples n');

ylabel ('Amplitude');

nx=xl+hl ;

nh=xh+hh ;

x =[x,zeros(1,l)];

h =[h,zeros(1,m)];

y = zeros(1, m +l -1)

for i = 1:m+l-1

    y(i) = 0;

    for j =1:m+l-1

        if(j<i+1)

            y(i) = y(i)+x(j)\*h(i-j+1) ;

            end

    end

end

disp('Linear convolution using equation is y(n) :');

disp(y);

r = nx:nh ;

subplot(3 ,1 ,3) ;

a=gca() ;

a.x\_location="origin";

a.y\_location="origin";

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

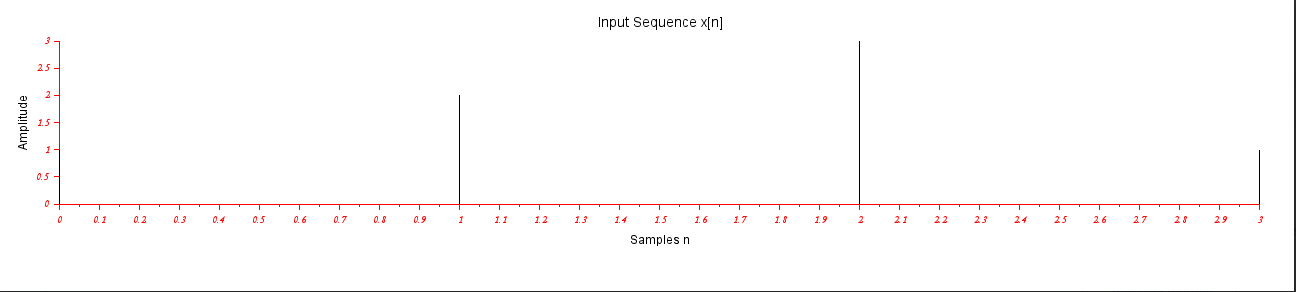
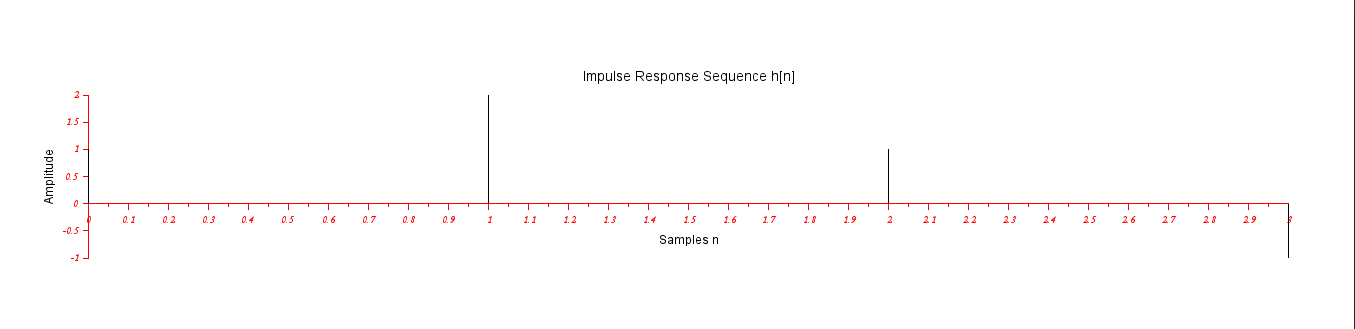
plot2d3('gnn',r , y) ;

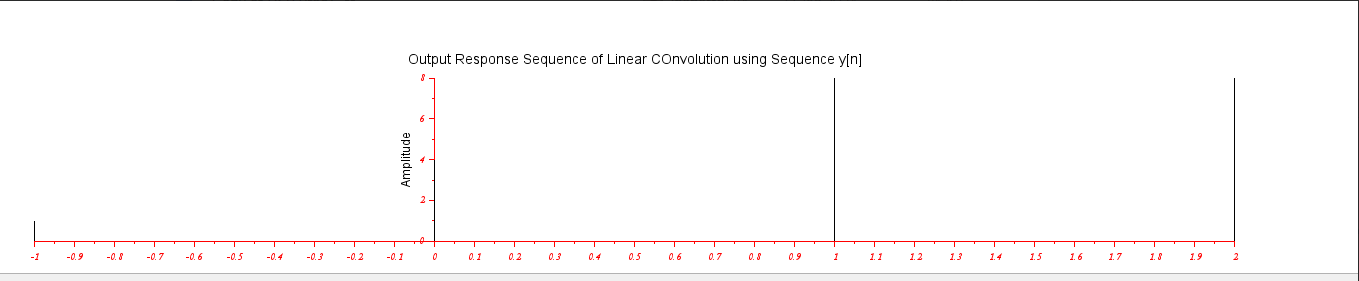
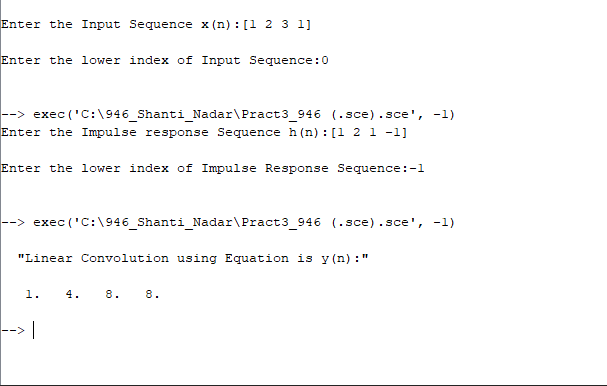
title('Output response sequence of linear convolution using equation y[n]');

xlabel('Samples n');

ylabel('Amplitude');

Output:

Conclusion:In this study,we understood how to find linear convolution of two signal sequences.  
  
                                  Practical 5

Aim: Write a scilab program to find Autocorrelation and cross correlation of the given sequences.

Theory: Crosscorrelation is a measure of similarity between two signals, while autocorrelation is a measure of how similar a signal is to itself. Within these categories, auto-correlation looks for time-related patterns of activity within a single event channel or data trace, while cross-correlation looks for a relationship between the activity in two separate event channels or data traces.

Code:

clc;

clear;

close;

x=input('Enter the input sequence=');

m=length(x);

xl=input('Enter the lower index of input sequence =');

xh=xl+m-1;

n=xl:1:xh;

subplot(2,2,1);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot2d3('gnn',n,x);

title('Input sequence [n]');

xlabel('Samples n');

ylabel('Amplitude');

h = input('Enter the impulse response sequence = ');

l = length(h) ;

hl = input('Enter the lower index of impulse response sequence = ');

hh = hl+l-1;

g = hl:1:hh;

subplot(2,2,2);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot2d3('gnn',g,h);

title('Impulse response sequence h[n]');

xlabel('Samples n');

ylabel('Amplitude');

y = xcorr(x,x) ;

disp('Auto correlation of given sequence y(n)=')

disp(y);

nx = xl+xl;

nh = xh+xh;

r = nx:nh;

subplot(2,2,3);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot2d3('gnn',r,y);

title('Output of Auto correlation sequence');

xlabel('Samples n');

ylabel('Amplitude');

z = xcorr(x, h);

disp('Cross correlation of sequence y(n)');

disp(z);

subplot(2,2,4);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

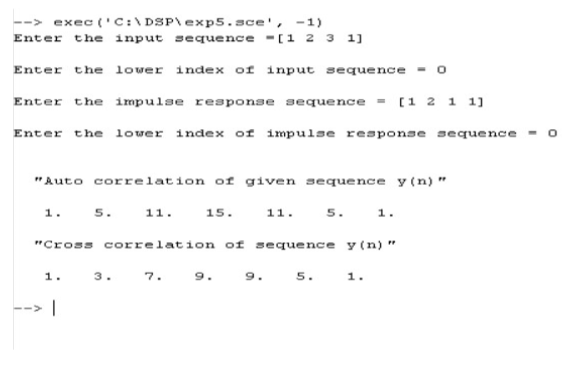
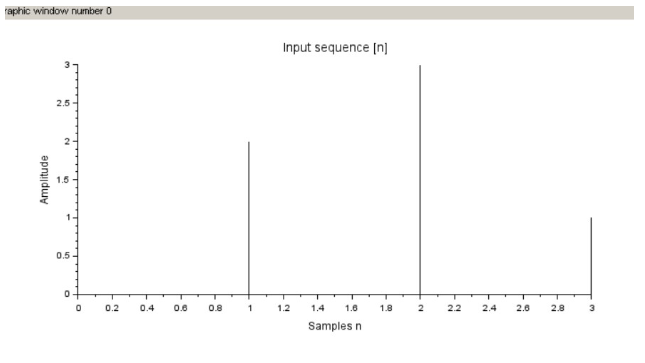
plot2d3('gnn',r,z);

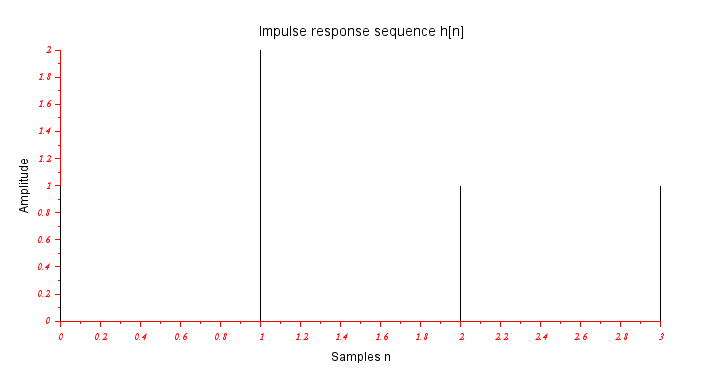
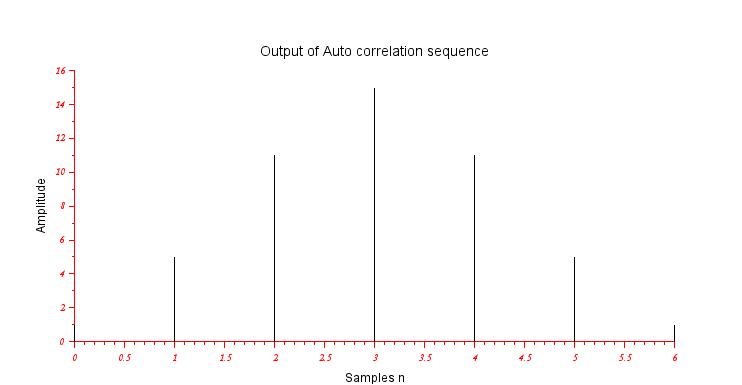
title('Output of Cross correlation sequence');

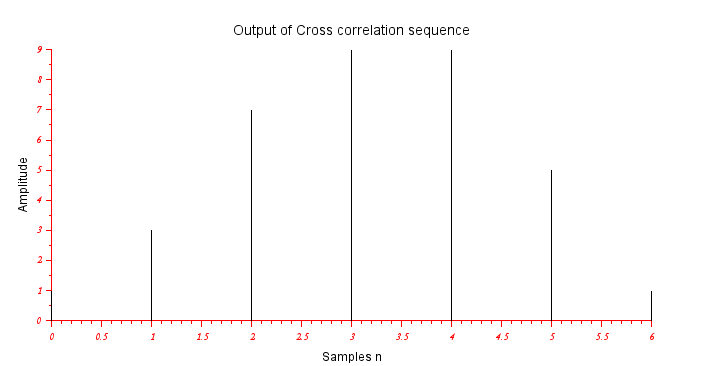
xlabel('Samples n');

ylabel('Amplitude');

Output:



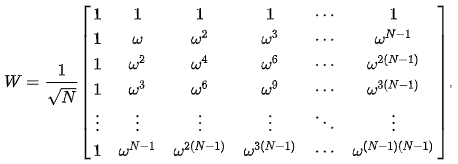
Conclusion:In this experiment,we learn to form cross-correlation and auto-correlation between two discrete signals.

                                      Practical 6

Aim:Write a Scilab Program to find N-Point DFT of the Given Sequences.

Theory: An *N*-point DFT is expressed as the multiplication  where is the original input signal,is the *N*-by-*N* [square](https://en.wikipedia.org/wiki/Square_matrix) DFT matrix, and is the DFT of the signal.

The transformation matrix can be defined as , or equivalently:



Where is a [primitive *N*th root of unity](https://en.wikipedia.org/wiki/Root_of_unity) in which  We can avoid writing large exponents for using the fact that for any exponent we have the identity .This is the [Vandermonde matrix](https://en.wikipedia.org/wiki/Vandermonde_matrix) for the roots of unity, up to the normalization factor. Note that the normalization factor in front of the sum  and the sign of the exponent in ω are merely conventions, and differ in some treatments. All of the following discussion applies regardless of the convention, with at most minor adjustments. The only important thing is that the forward and inverse transforms have opposite-sign exponents, and that the product of their normalization factors be 1/*N*. However, the choice here makes the resulting DFT matrix [unitary](https://en.wikipedia.org/wiki/Unitary_matrix), which is convenient in many circumstances.[Fast Fourier transform](https://en.wikipedia.org/wiki/Fast_Fourier_transform) algorithms utilize the symmetries of the matrix to reduce the time of multiplying a vector by this matrix, from the usual Similar techniques can be applied for multiplications by matrices such as [Hadamard matrix](https://en.wikipedia.org/wiki/Hadamard_matrix) and the [Walsh matrix](https://en.wikipedia.org/wiki/Walsh_matrix).

Code:

N=input('Enter the value of N=\n');

x=input('Enter  the input sequence x(n)=');

subplot(3,2,1);

a=gca();

a.foreground = 5;

a.font\_color=5;

a.font\_style=5;

plot2d3(x);

title('Input Sequence');

xlabel('Samples n');

ylabel('Amplitude');

for k=1:N

    y(k)=0;

    for n=1:N

        y(k)=y(k)+x(n).\*exp(-%i\*2\*%pi\*(k-1)\*(n-1)/N)

        ;

        A=real(y);

        B=imag(y);

    end

end

mag=abs(y);

x1 = atan ( imag ( y) ,real ( y ) ) ;

phase = x1 \*(180/ %pi ) ;

disp('The output DFT sequence is:') ;

disp(y);

subplot (3 ,2 ,2) ;

a=gca () ;

a.foreground = 5;

a.font\_color = 5;

a.font\_style = 5;

plot2d3 ( y );

title('Output DFT sequence');

xlabel('Samples n');

ylabel('Amplitude');

*//Real Value*

disp('The Resultant Real value is:') ;

disp(A);

subplot(3,2,3);

a=gca();

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot2d3(A);

title('Real Value');

xlabel('Samples n');

ylabel('Amplitude');

disp('THe resultant imaginary value is:');

disp(B);

subplot(3,2,4);

a=gca();

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot2d3(B);

title('Imaginary value');

xlabel('Samples n');

ylabel('Amplitude');

disp('The Magnitude response is:');

disp(mag);

subplot(3,2,5);

a=gca();

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot2d3(mag);

title('Magnitude Response');

xlabel('Samples n ');

ylabel('Amplitude ');

disp('The phase response is:');

disp(phase);

subplot(3,2,6);

a=gca();

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

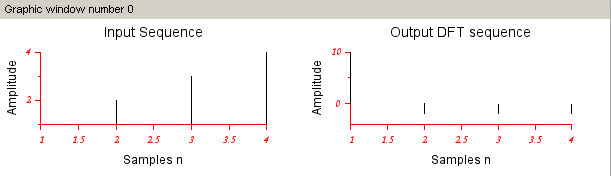
plot2d3(phase);

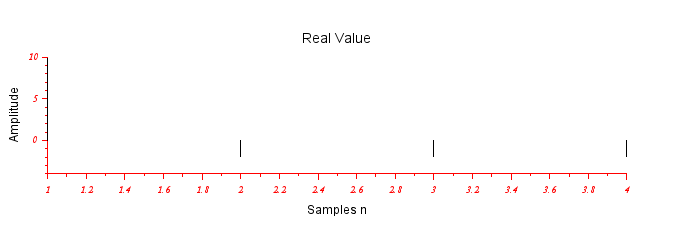
title('Phase Response');

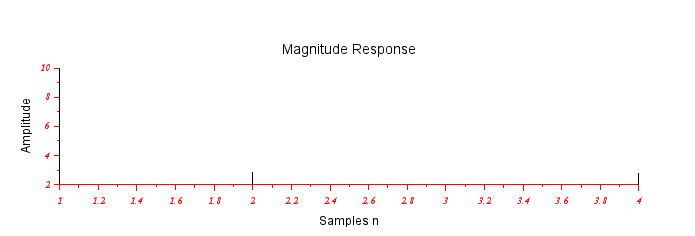
xlabel('Samples n');

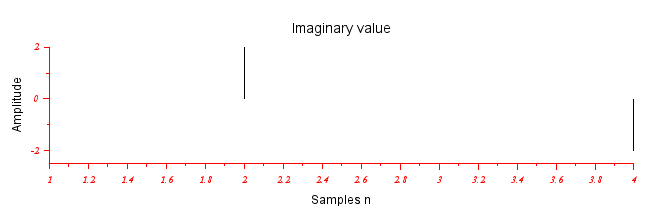
ylabel('Phase');

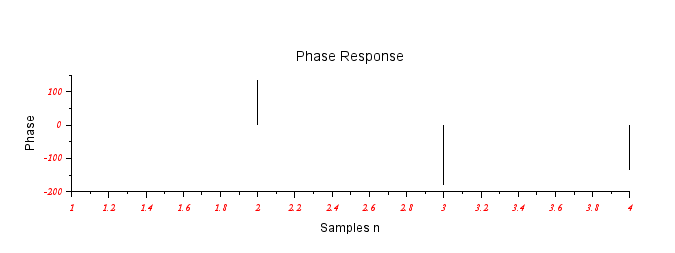
Output:

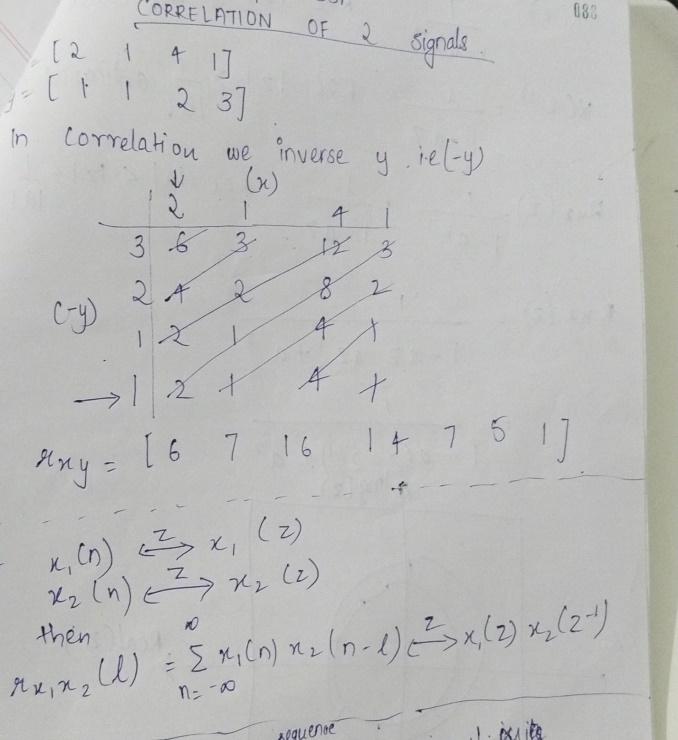
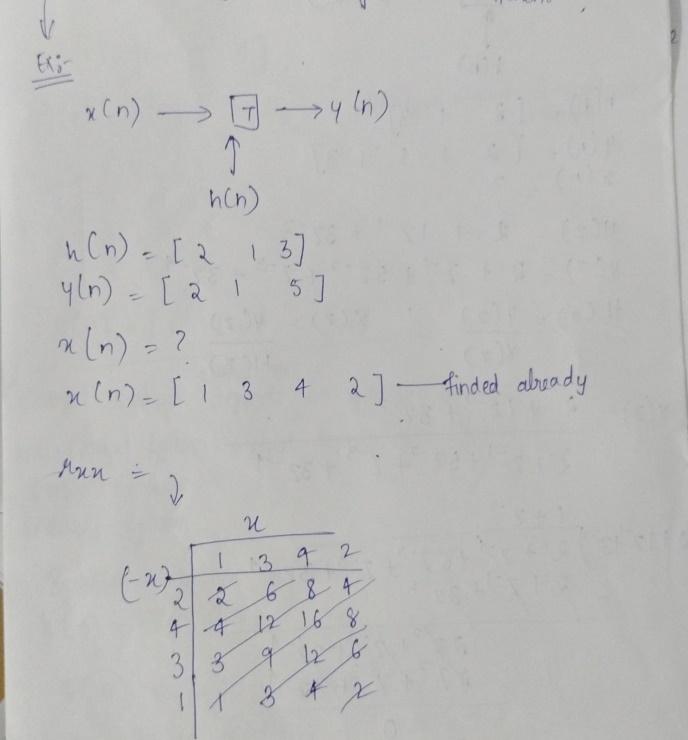










Conclusion:In this experiment ,we learn to find DFT of given discrete sequence.

                                     Practical 7

Aim:WRITE A SCILAB PROGRAM TO COMPUTE LINEAR CONVOLUTION OF TWO SEQUENCES USING DFT BASED APPROACHES

Theory:

This method is more efficient for long sequences compared to the direct time-domain convolution.

1. Discrete Fourier Transform (DFT):

The Discrete Fourier Transform is a mathematical technique used to analyze the frequency content of a discrete signal. For a sequence *x*[*n*] of length  *N*, the DFT is defined as:



Here, *X*[*k*] represents the complex amplitude of the *k*-th frequency component, and

is the complex exponential term.

2. Convolution Theorem:

The Convolution Theorem states that the convolution of two sequences in the time domain is equivalent to the element-wise multiplication of their respective Fourier transforms in the frequency domain. Mathematically:



Where ∗ represents convolution, and *X*[*k*] and *H*[*k*] are the Fourier transforms of sequences *x*[*n*] and *h*[*n*], respectively.

3. Linear Convolution using DFT:

Given the Convolution Theorem, we can efficiently compute linear convolution using the following steps:

1.Zero-padding: Ensure both sequences *x*[*n*] and *h*[*n*] have the same length (*N*). Pad with zeros if necessary.

2.Compute DFTs: Calculate the DFTs of *x*[*n*] and *h*[*n*] using the FFT (Fast Fourier Transform) algorithm.

3.Element-wise Multiplication: Multiply the corresponding elements of the two DFTs:



4.Inverse DFT: Compute the inverse DFT of *Y*[*k*] using the IFFT (Inverse Fast Fourier Transform) algorithm to obtain the linear convolution result *y*[*n*].

Code:

x = input('Enter input sequence (n) : ') *// x =[1 2 3 1 ]*

m = length(x) ;

xl = input('Enter lower index of input sequence : ') *// 0*

xh = xl+m-1;

n = xl:1:xh;

subplot (3,1,1);

a = gca();

a.x\_location ="origin";

a.y\_location ="origin";

a.foreground = 5;

a.font\_color = 5;

a.font\_style = 5;

plot2d3('gnn',n,x);

title('Input sequence x[n]');

xlabel('Samples n');

ylabel('Amplitude');

h = input('Enter impulse response sequence h[n] : '); *// h =[1 2 1 −1]*

l = length(h);

hl = input('Enter the lower index of impulse response sequence : ') ; *//−1*

hh = hl+l-1;

g = hl:1:hh;

subplot(3,1,2);

a = gca();

a.x\_location ="origin";

a.y\_location ="origin";

a.foreground = 5;

a.font\_color = 5;

a.font\_style = 5;

plot2d3('gnn',n,x);

title('Impulse response sequence h[n]');

xlabel('Samples n');

ylabel('Amplitude');

nx = xl+hl; *// r a n g e o f k*

nh = xh+hh; *// r a n g e o f n*

p = m+l-1;

x =[x,zeros(1,p-l)];

h =[h,zeros(1,p-m)]; *// d f t − i d f t*

XK = fft(x,-1); *// DFT o f x*

HK = fft(h,-1); *// DFT o f h*

YK = XK.\*HK;

yn = fft(YK,1);

disp('Linear Convolution by DFT-IDFT method is y(n) : ');

disp(real(yn));

r = nx:nh;

subplot(3,1,3);

a = gca();

a.x\_location ="origin";

a.y\_location ="origin";

a.foreground = 5;

a.font\_color = 5;

a.font\_style = 5;

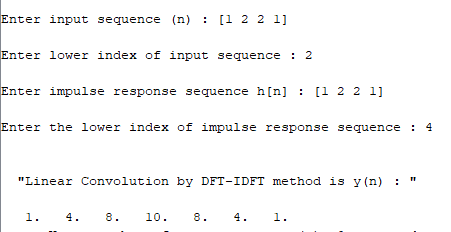
plot2d3('gnn',r,yn);

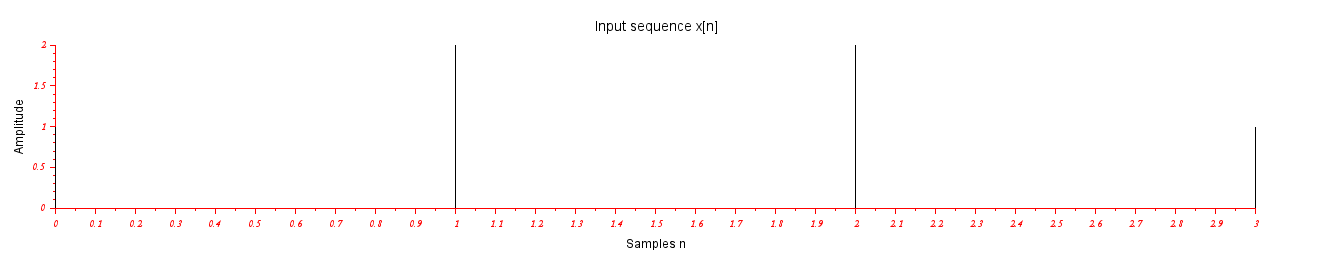
title('Output Response of Linear Convolution by DFT−IDFT method y[n] : ');

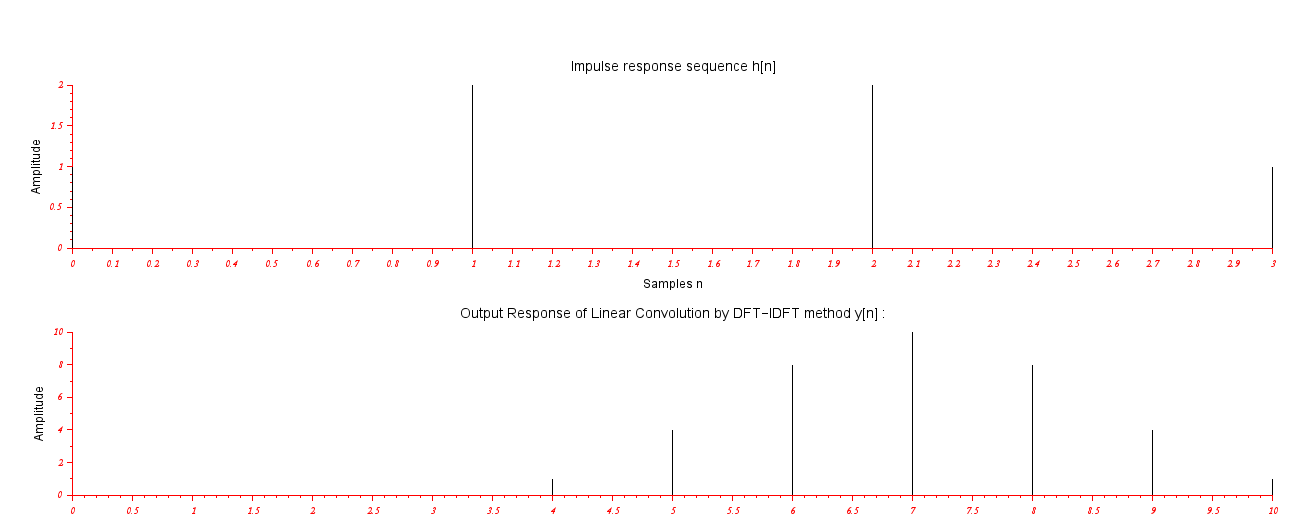
xlabel('Samples n');

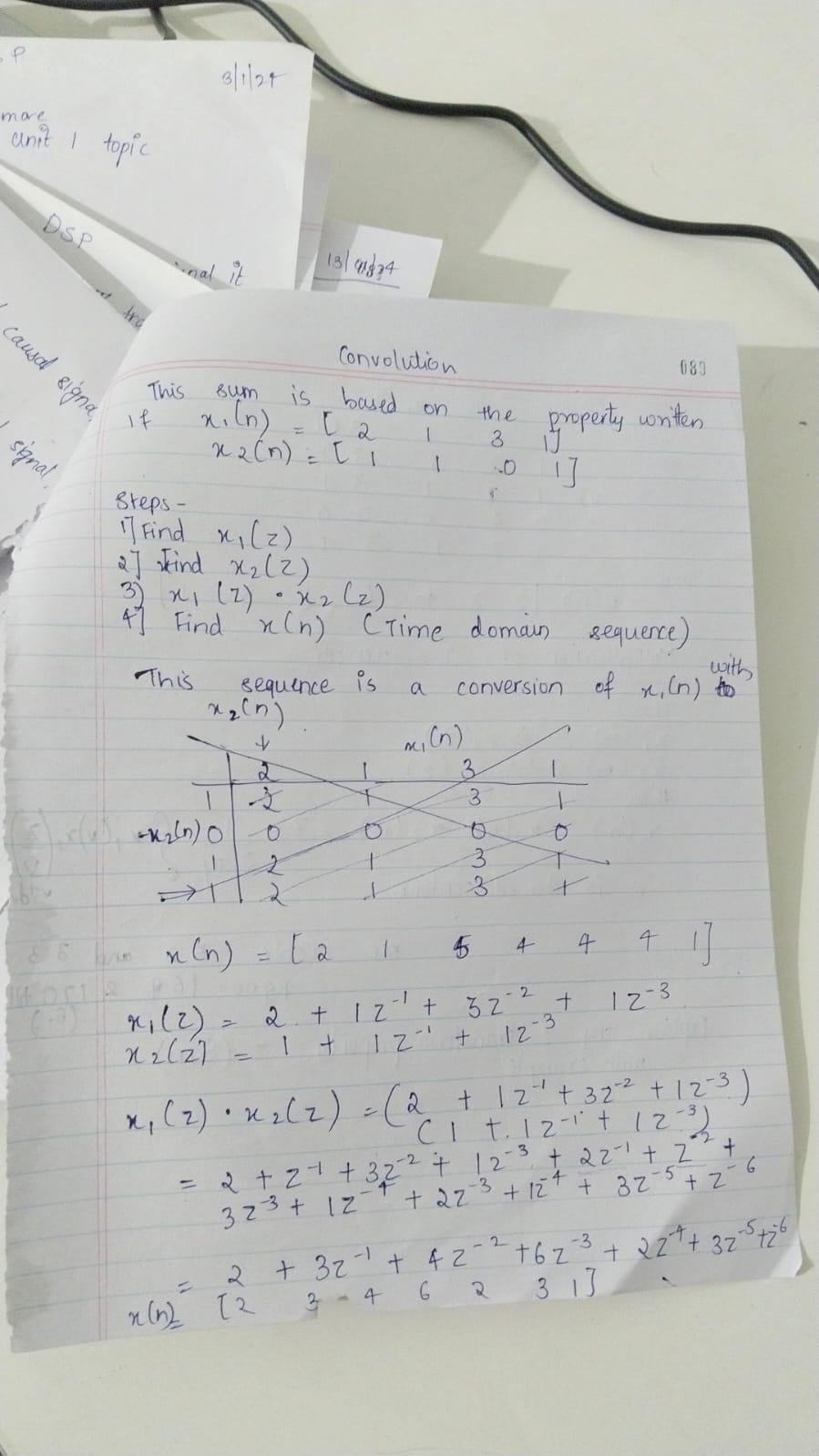
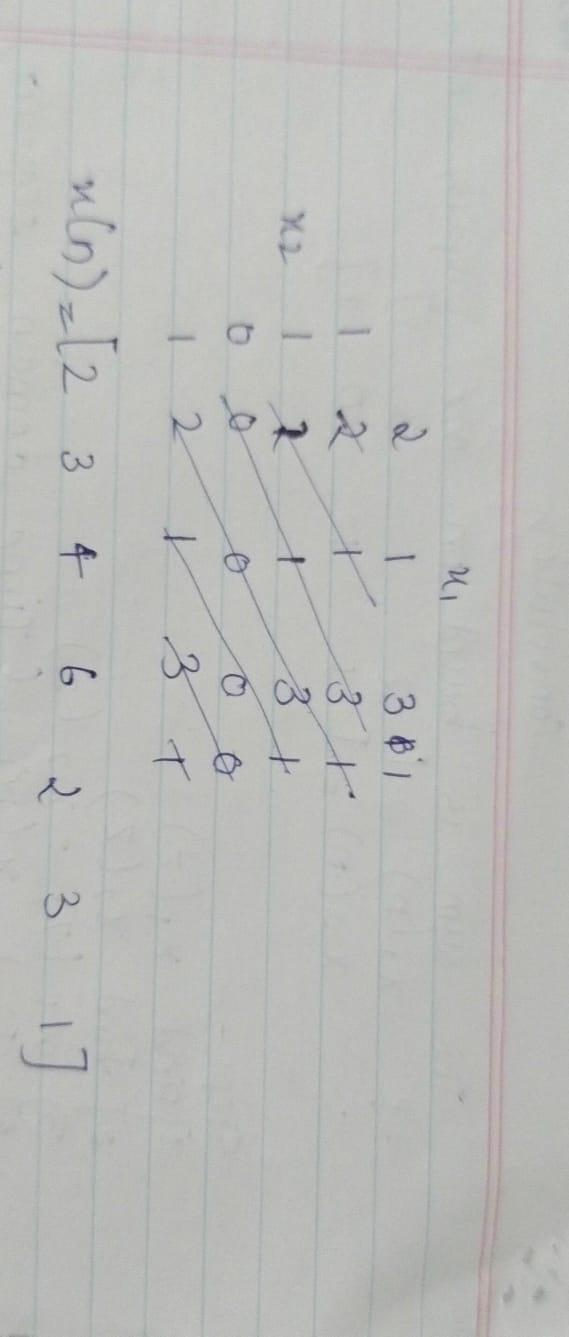
ylabel('Amplitude');

Output:







Conclusion:In this Experiment,we derive the linear convolution graph and signal using DFT approach.

                                    Practical 8

Aim:WRITE A SCILAB PROGRAM TO COMPUTE CIRCULAR CONVOLUTION OF TWO SEQUENCES USING BASIC EQUATION(Without using the library function)

Theory: To compute the circular convolution of two sequences ‘x[n]’ and ‘h[n]’ using the basic equation, you can use the formula:



This equation explicitly calculates each term of the circular convolution by wrapping around the indices using the modulo operation. Here's a breakdown of the steps:

1.Define Sequences: Suppose you have two sequences x[n] and h[n].

2.Zero-padding: Make sure both sequences have the same length (*N*). If needed, pad the sequences with zeros.

3.Compute Circular Convolution: Use the basic equation to calculate each term of the circular convolution and sum them up.



4.Display or Use Result: The resulting sequence *y*[*n*] represents the circular convolution of *x*[*n*] and *h*[*n*].

Code:

x=input('Enter the input sequence=')

m=length(x);

x1=input('Enter the lower index of input sequence=')

xh=x1+m-1;

n=x1:1:xh;

subplot(3,1,1);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot2d3('gnn',n,x);

title('Input Sequence x[n]');

xlabel('Samples n');

ylabel('Amplitude');

h=input('Enter the impulse response sequence=');

l=length(h);

h1=input('Enter the lower index of impulse response sequence=');

hh=h1+l-1;

g=h1:1:hh;

subplot(3,1,2);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot2d3('gnn',g,h);

title('Impulse Response Sequence h[n]');

xlabel('Samples n');

ylabel('Amplitude');

N=max(m,1);

p=m-1;

if(p>=0) then

    h=[h,zeros(1,p)];

else

    x=[x,zeros(1,-p)];

end

for i=1:N

    y(i)=0;

    for j=1:N

        k=i-j+1;

        if (k<=0)

            k=N+k;

        end

        y(i)=y(i)+x(j)\*h(k);

    end

end

disp('Circular convolution by equation is y[n]:') ;

disp(y);

nx=x1+h1;

r=nx:length(y)-1;

subplot(3,1,3);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

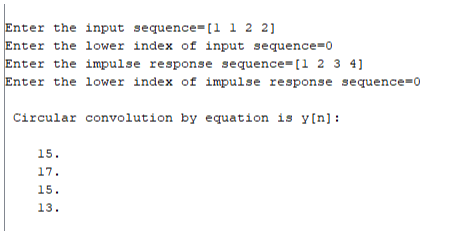
plot2d3('gnn',r,y);

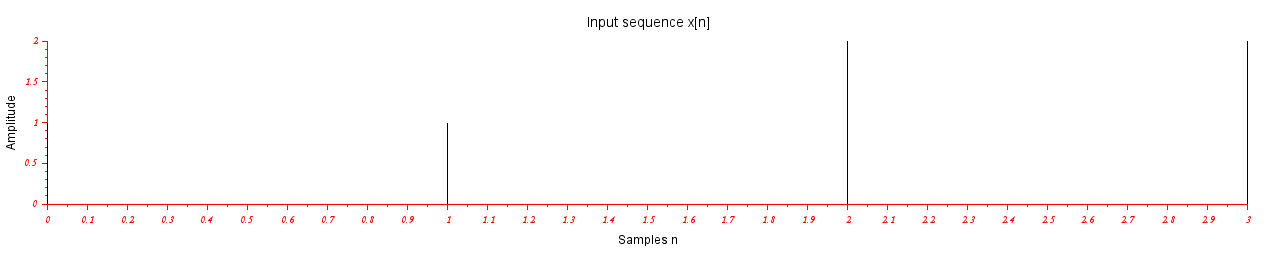
title('Output Response Sequence of Circular Convolution y[n] using Basic Equation');

xlabel('Samples n');

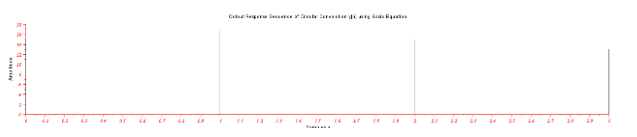
ylabel('Amplitude');

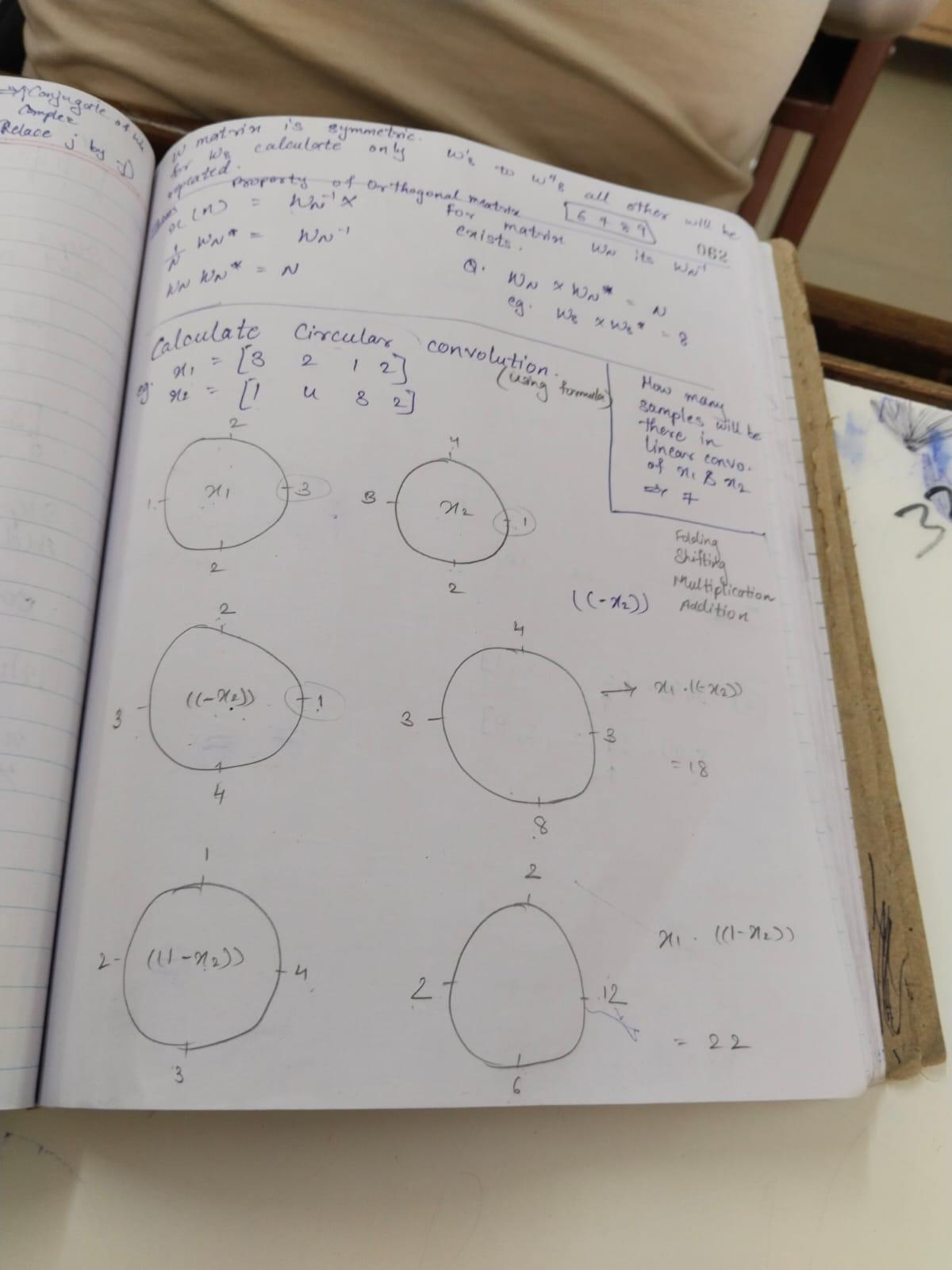
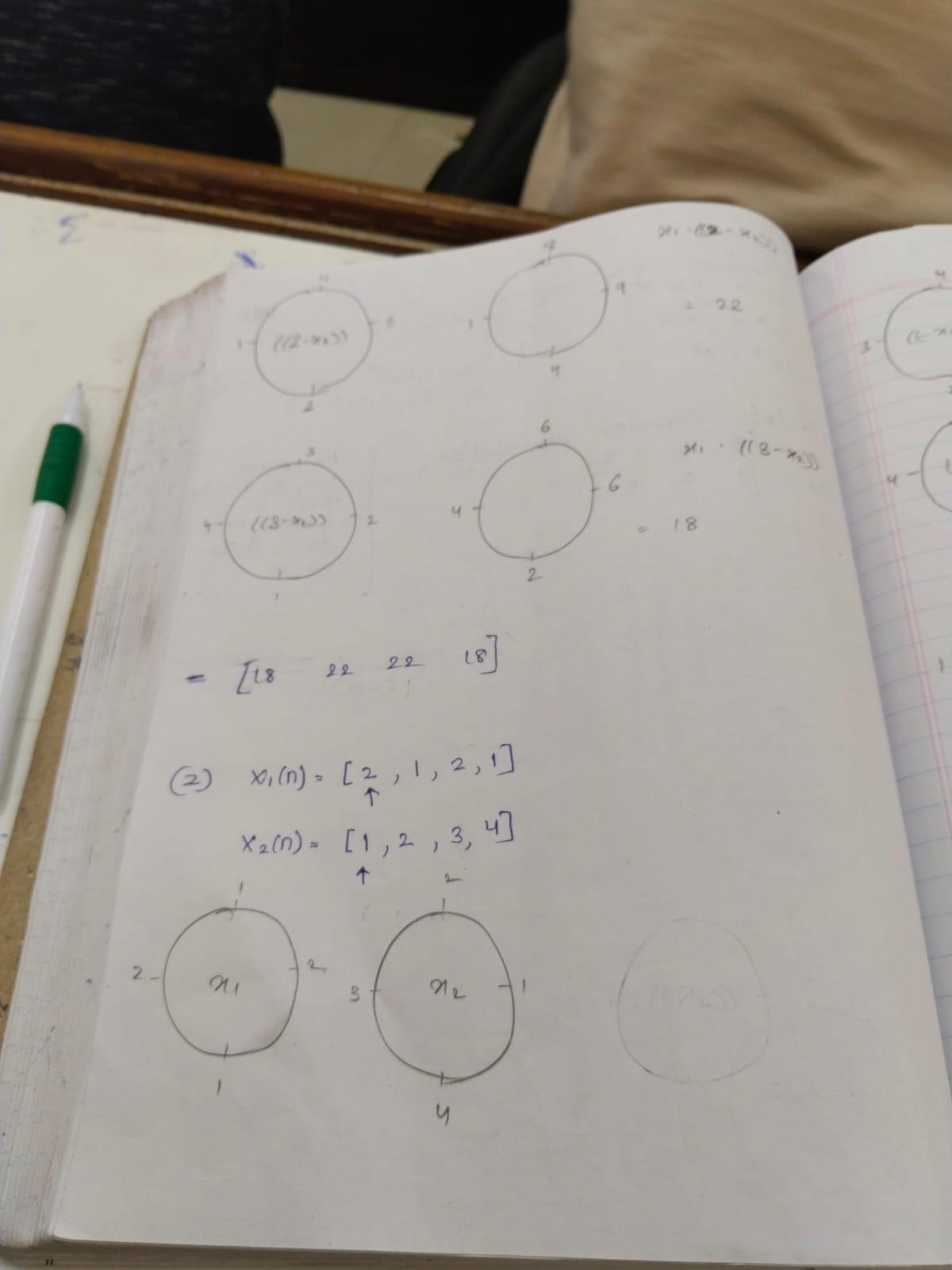
Output:

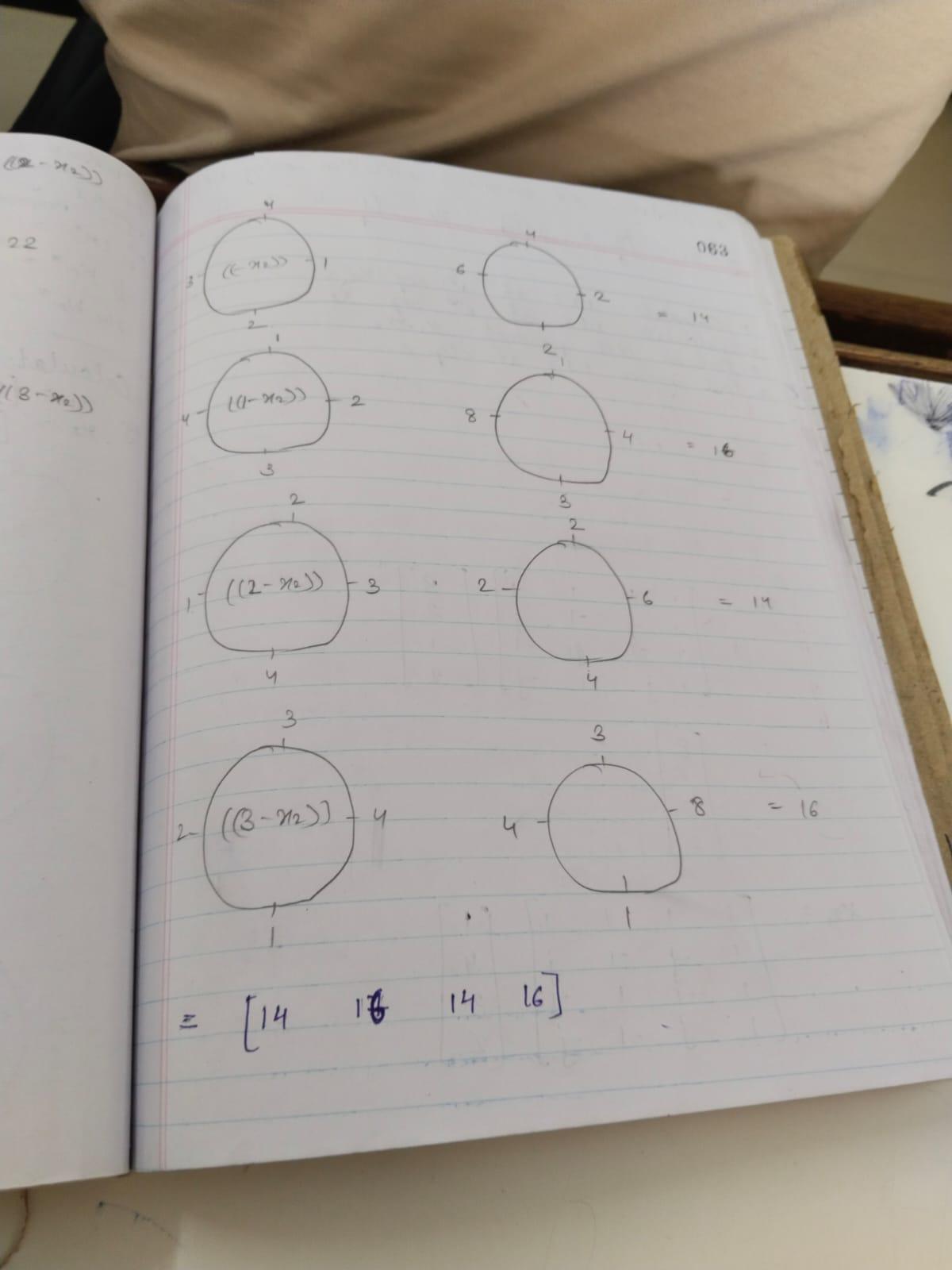










Conclusion:In this experiment,we come to know that circular convolution graph and signals can also  using DFT approach.

                                  Practical 9

Aim: WRITE A SCILAB PROGRAM TO COMPUTE CIRCULAR CONVOLUTION OF THE TWO SEQUENCES USING DFT BASED APPROACH

Theory:

1. Circular Convolution:

Circular convolution is an operation that combines two sequences, taking into account the periodic nature of circular shifts. For two sequences *x*[*n*] and *h*[*n*] of length *N*, the circular convolution *y*[*n*] is given by:



This equation captures the circular shift property, where indices wrap around within the range [0,*N*−1].

Code:

x=input('Enter the Input Sequence:')

m=length(x);

x1=input('Enter the lower index of input sequence:')

xh=x1+m-1;

n=x1:1:m-1;

subplot(3,1,1);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.font\_color=5;

a.foreground=5;

a.font\_style=5;

plot2d3('gnn',n,x);

title('Input Sequence');

xlabel('Samples n');

ylabel('AMplitude');

h=input('Enter the Impulse response Sequence:')

l=length(h);

h1=input('Enter the lower index of impulse response sequence:');

hh=h1+l-1;

g=h1:1:hh;

subplot(3,1,2);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.font\_color=5;

a.foreground=5;

a.font\_style=5;

plot2d3('gnn',g,h);

title('Impulse response Sequence h[n]');

xlabel('Samples n');

ylabel('AMplitude');

N=max(m,l);

p=m-l;

if(p>=0) then

    h=[h,zeros(1,p)];

else

    x=[x,zeros(1,-p)];

end

XK=fft(x,-1);

HK=fft(h,-1);

YK=XK.\*HK;

y=ifft(YK);

disp('Circular convolution by DFT is y(n):')

disp(real(y));

nx=x1+h1;

r=nx:length(y)-1;

subplot(3,1,3);

a=gca();

a.x\_location="origin";

a.y\_location="origin";

a.font\_color=5;

a.foreground=5;

a.font\_style=5;

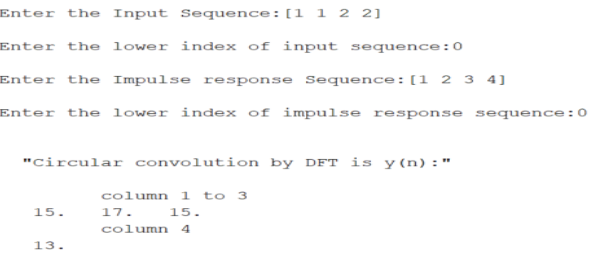
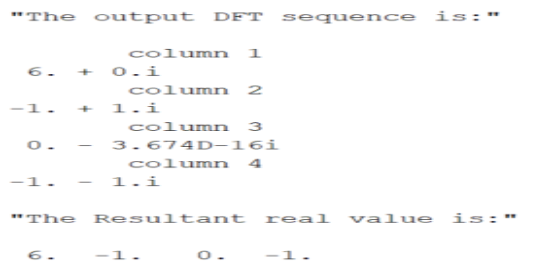
plot2d3('gnn',r,y);

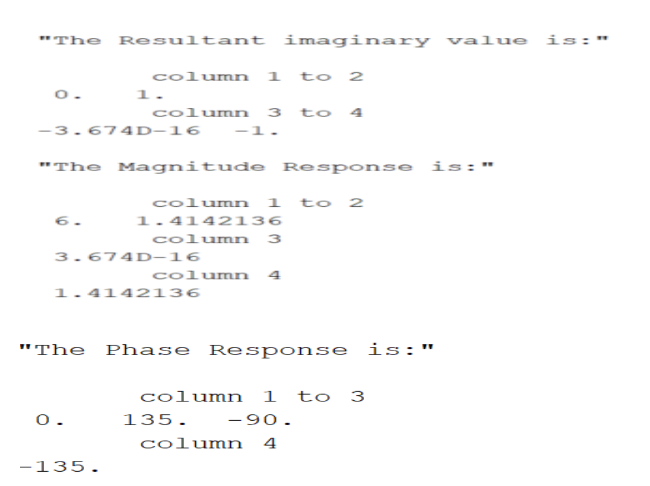
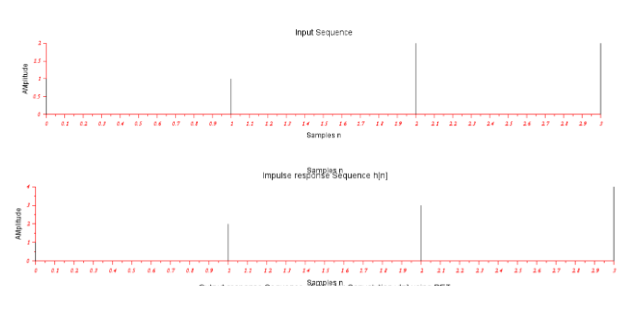
title('Output response Sequence of Circular Convolution y[n] using DFT');

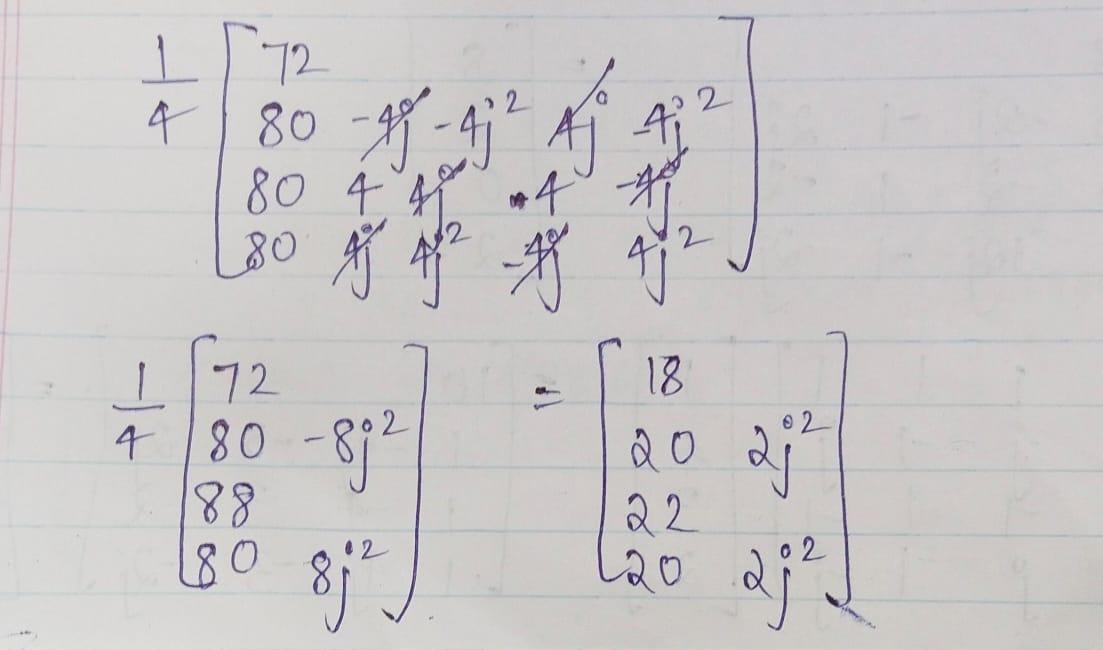
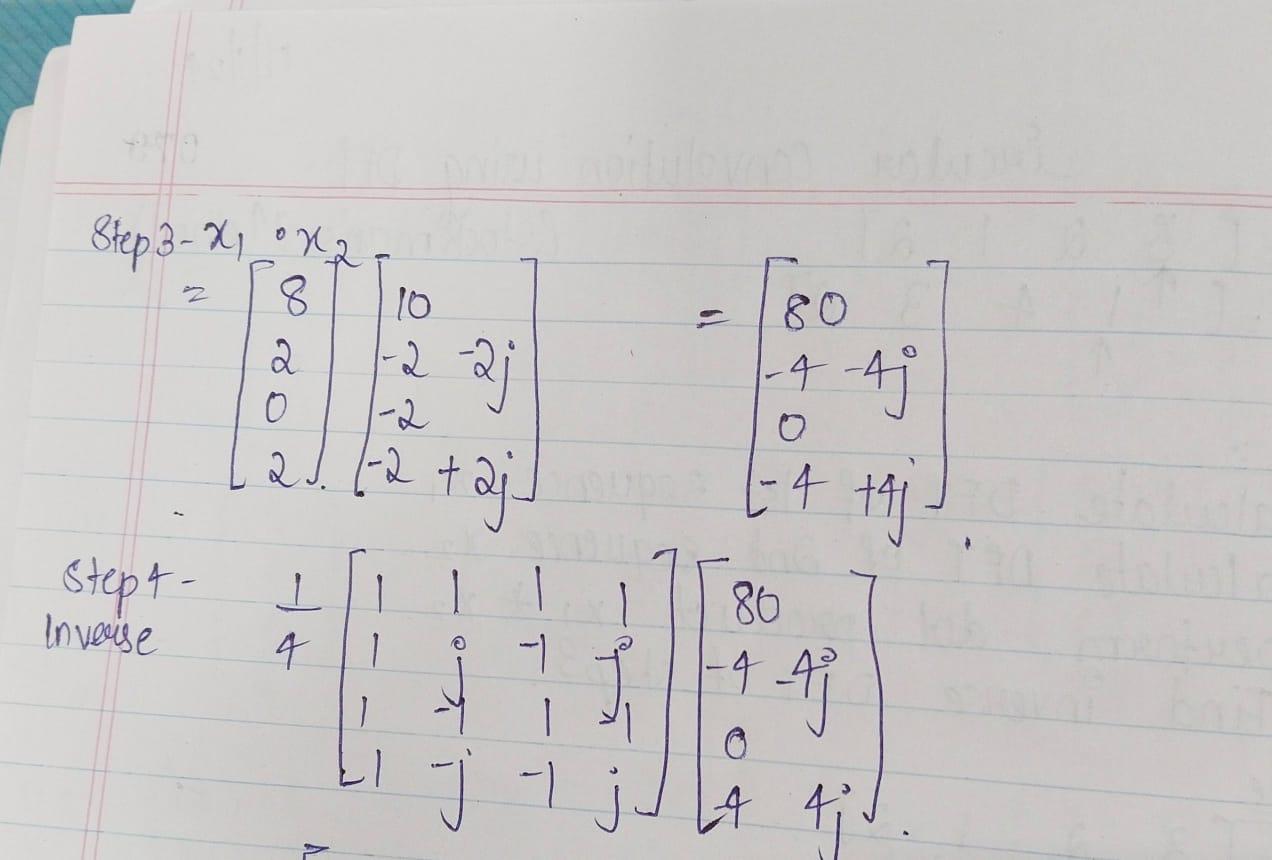
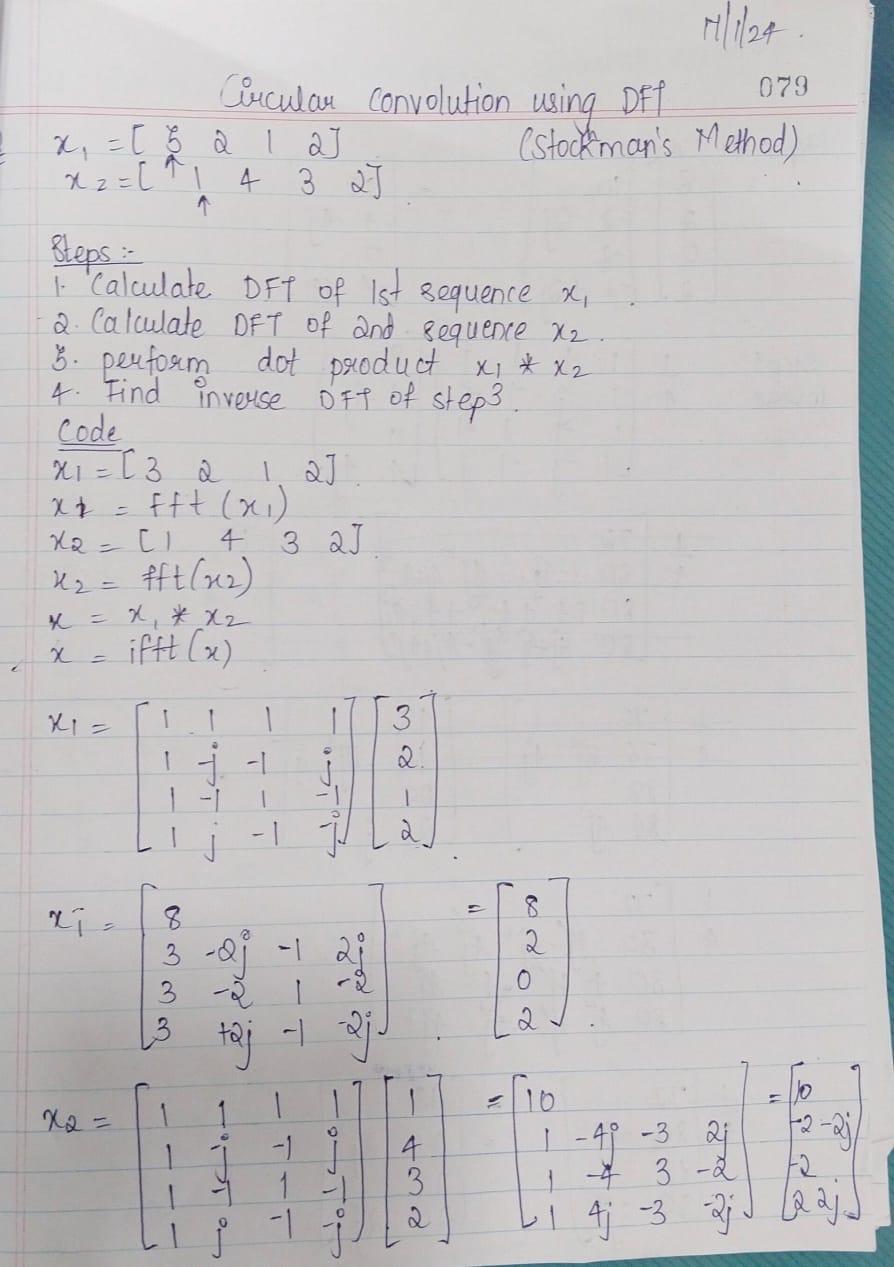
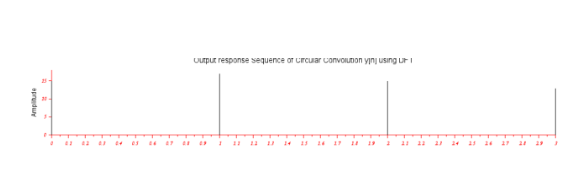
xlabel('Samples n');

ylabel('Amplitude');

Output:



Conclusion:

Hence, we have successfully performed this practical

                                 Practical 10

Aim: WRITE A SCILAB PROGRAM TO FIND FFT USING DECIMATION IN TIME(DIT) METHOD.

Theory: The fast fourier transform (FFT) is an important technique for image compression, digital signal processing and communication especially for application in multiple input multiple output OFDM system. The fast fourier transform are good algorithm and computed discrete fourier transform (DFT). In this paper, the comparison study of various FFT algorithm and compare all them. FFT algorithm is divided into two part i.e. decimation in time (DIT) and decimation in frequency (DIF). In DIT algorithm firstly computed multiplier then adder but in DIF firstly computed adder then multiplier. In this paper we study of different types of multiplier i.e. array multiplier; sing multiplier (Baugh Wooley) and complex multiplier. In proposed complex multiplier is consuming three multipliers. In further work in my dissertation in design to 8point, 16-point, 32-point, 64-point and 128-point radix FFT algorithm in different multiplier.

Code:

x=input('Enter Input Sequence:');

N=length(x);

s=log2(N);

if(N==8) then

    stage1=1;

    x=[x(1) x(5) x(3) x(7) x(2) x(6) x(4) x(8)];

    for stage=1:s

        for index=0:(2^stage1):(N-1);

            for n=0:(stage1-1);

                pos=n+index+1;

                pow=(2^(s-stage))\*n;

                w=exp((-1\*%i)\*(2\*%pi)\*pow/N);

                a=x(pos)+x(pos+stage1).\*w;

                b=x(pos)-x(pos+stage1).\*w;

                x(pos)=a;

                x(pos+stage1)=b;

             end

         end

    stage1=2\*stage1;

    end

    y=x;

    disp('FFT of the given input sequence is y(n):');

    disp(y);

else

    stage1=1;

    x=[x(1) x(3) x(2) x(4)];

    for stage=1:s

        for index=0:(2^stage):(N-1)

            for n=0:(stage1-1);

                pos=n+index+1;

                pow=(2^(s-stage))\*n;

                w=exp((-1\*%i)\*(2\*%pi)\*pow/N);

                a=x(pos)+x(pos+stage1).\*w;

                b=x(pos)-x(pos+stage1).\*w;

                x(pos+stage1)=b;

            end

        end

    stage1=2\*stage1;

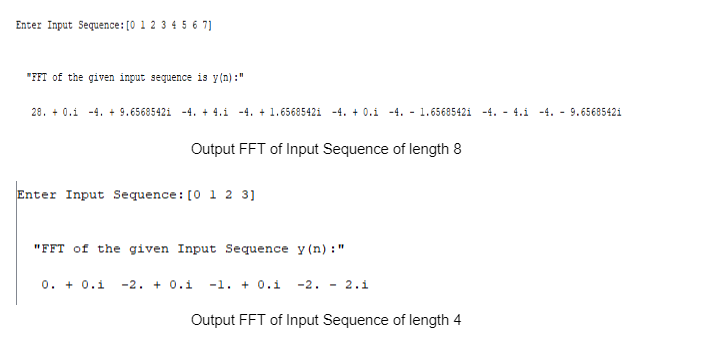
end

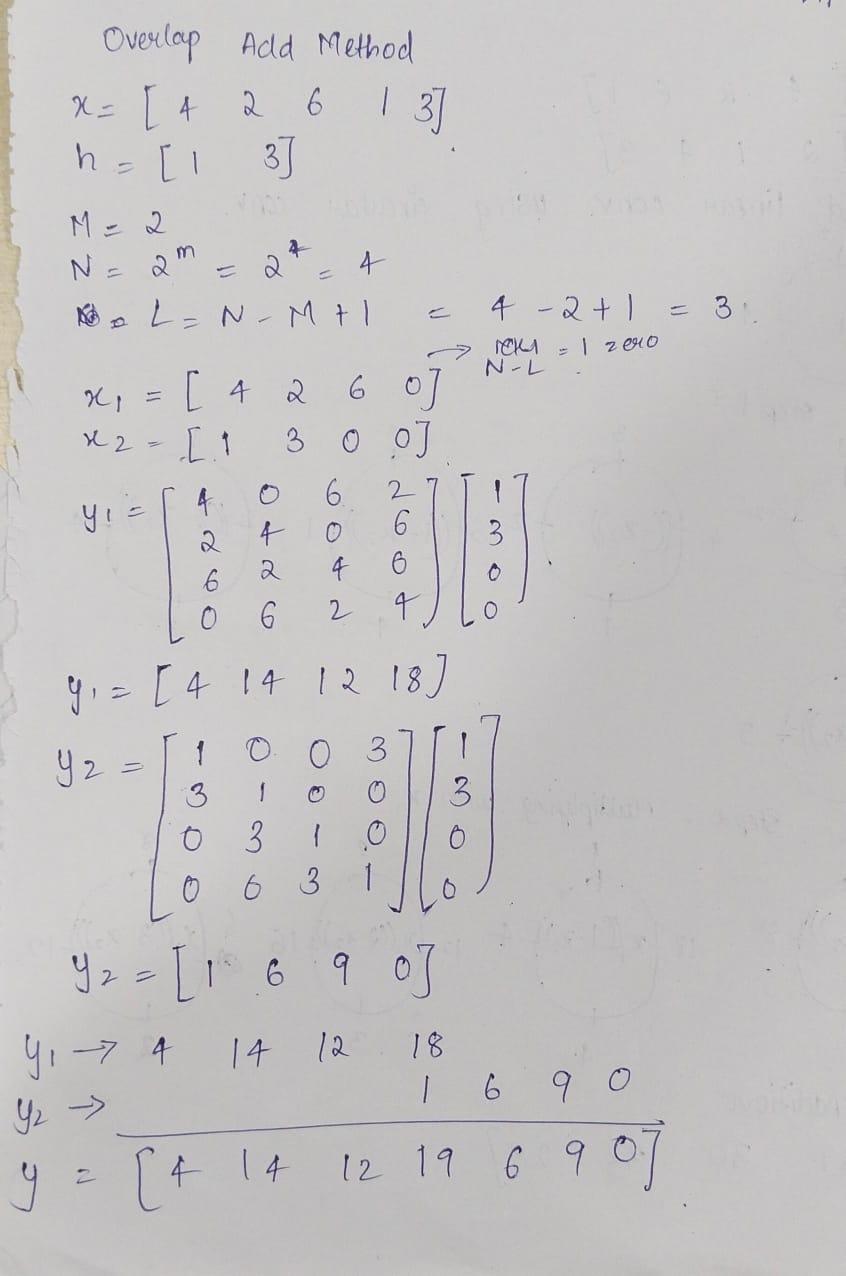
y=x;

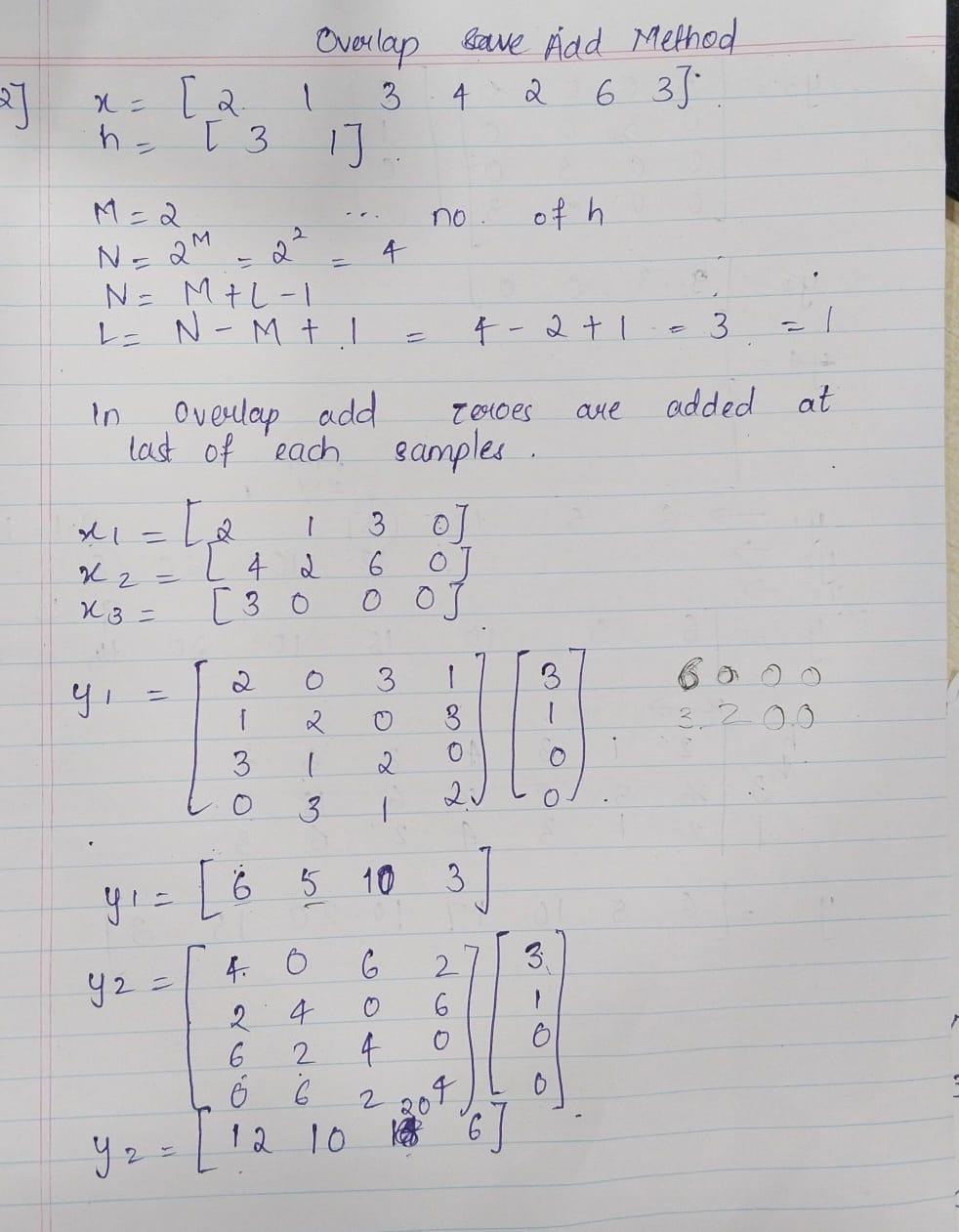
disp('FFT of the given Input Sequence y(n):');

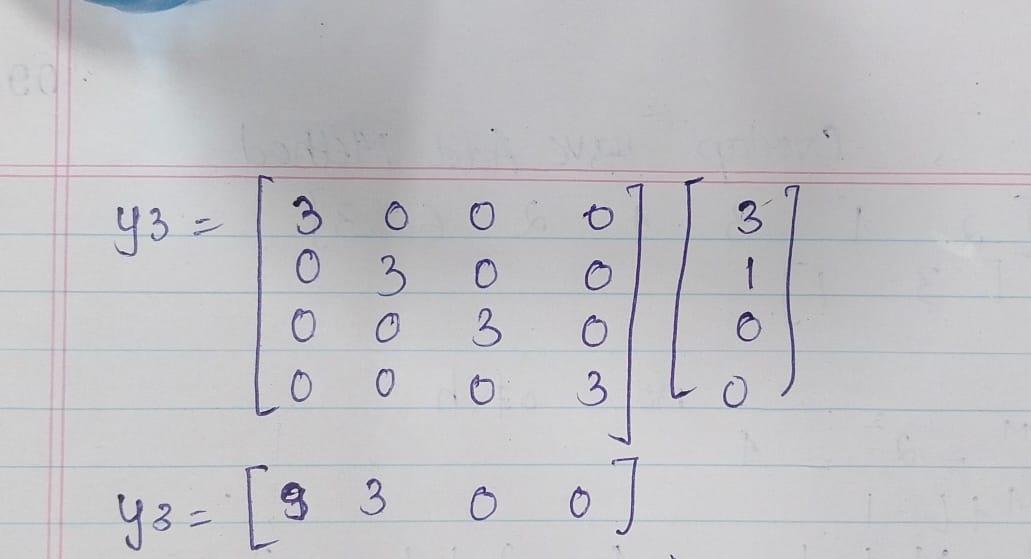
disp(y)

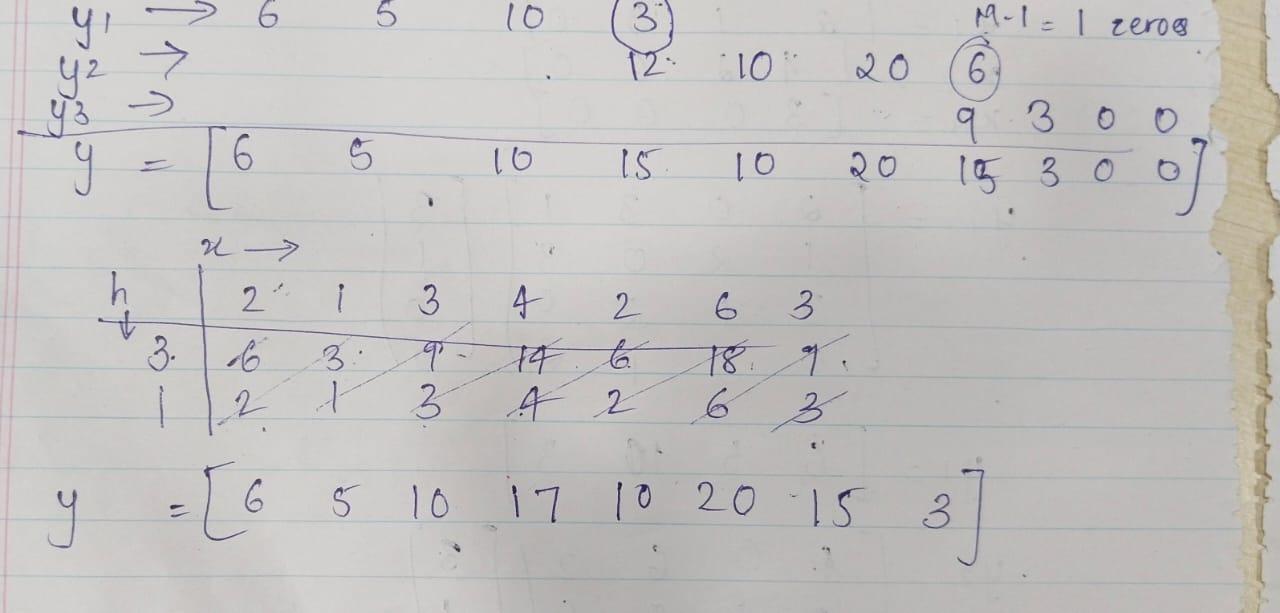
Output:











Conclusion:Hence,we found FFT using DIT Method.

                              Practical 11

Aim: STUDY TIME AND FREQUENCY RESPONSE OF LTI SYSTEM.

Theory: LTI stands for Linear Time-Invariant system, a fundamental concept in signal processing and control theory.

1.Time Response:

-The time response of an LTI system describes how the system behaves over time when subjected to different inputs.

-LTI systems exhibit properties of linearity and time invariance, which means their response to a sum of inputs is the sum of their responses to individual inputs, and their behavior doesn't change over time.

-The time response of an LTI system is often characterized by its impulse response. When the input to the system is an impulse (a very short signal), the output of the system is its impulse response.

-By convolving the input signal with the impulse response, you can find the system's output for any input signal, thanks to the properties of linearity and time invariance.

2.Frequency Response:

-The frequency response of an LTI system describes how the system responds to different frequencies present in the input signal.

-It's often represented using the system's transfer function, which is the Laplace transform of its impulse response (for continuous-time systems) or the Z-transform of its impulse response (for discrete-time systems).

-The transfer function gives you insight into how the system amplifies or attenuates different frequencies. It's often expressed in terms of magnitude and phase response.

-The magnitude response tells you how the system's gain changes with frequency, while the phase response tells you how the system shifts the phase of the input signal at different frequencies.

In summary, the time response of an LTI system describes its behavior over time, often characterized by its impulse response, while the frequency response describes how the system responds to different frequencies present in the input signal, often represented by its transfer function in the frequency domain. Both the time and frequency responses are essential for understanding and analyzing the behavior of LTI systems in various applications.

Code:

n = input ( 'Enter number of points for frequency response n= ' ) ;

w =0:2\* %pi / n :2\* %pi ;

h = zeros (1 , length ( w ) ) ;

for x =1: length (w )

h ( x ) =1/(1 -0.9\* exp( - %i \* w ( x ) )) ;

end

h1 =abs( h ) ; *// magnitude of transfer function*

h2 =atan ( imag ( h) ,real ( h ) ) ; *// phase of the transfer function*

*// plot the magnitude spectrum*

subplot (2 ,1 ,1) ;

plot (w , h1 ) ;

xlabel ( ' frequency w' ) ;

ylabel ( ' amplitude' ) ;

title ( ' magnitude response of system H(w) ' ) ;

*// plotting the phase spectrum*

subplot (2 ,1 ,2) ;

plot (w , h2 ) ;

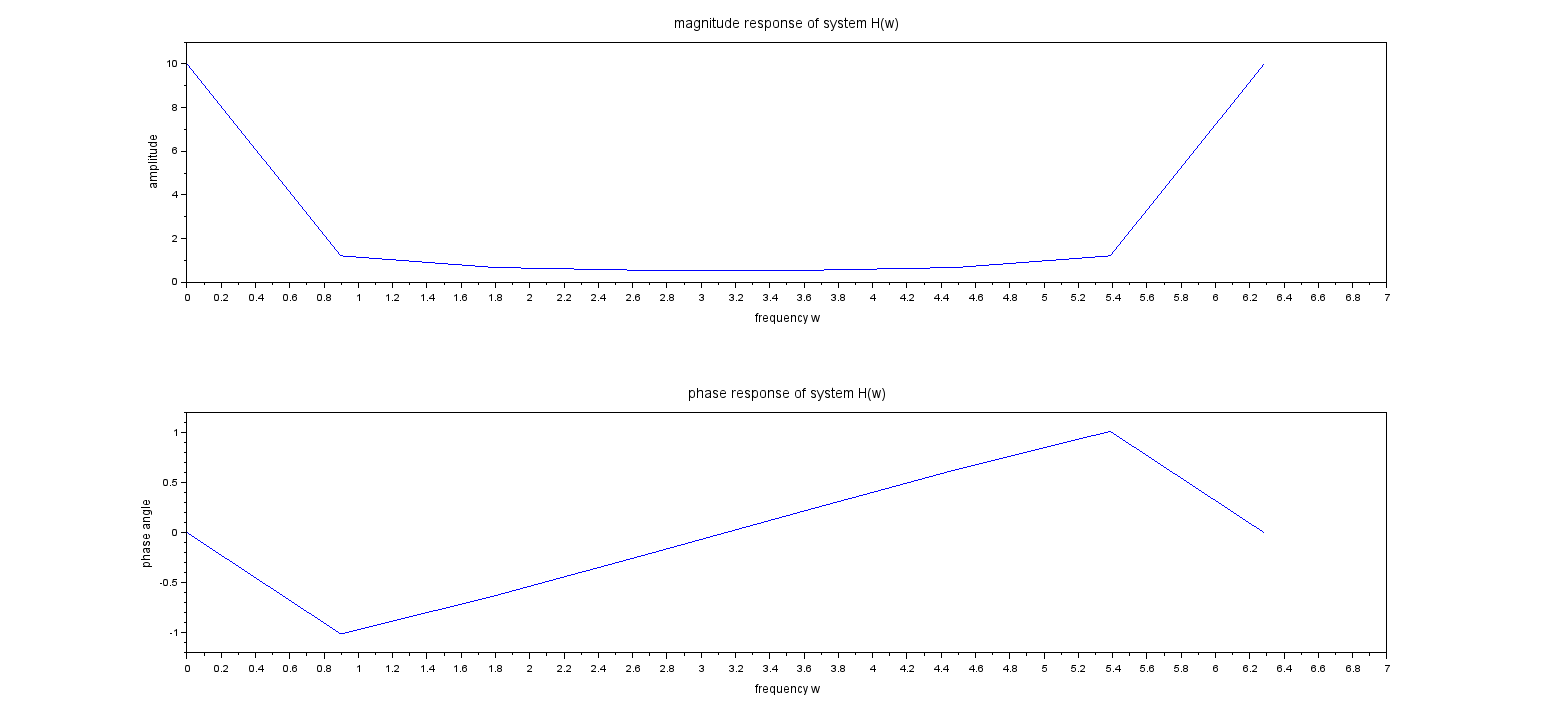
xlabel ( ' frequency w' ) ;

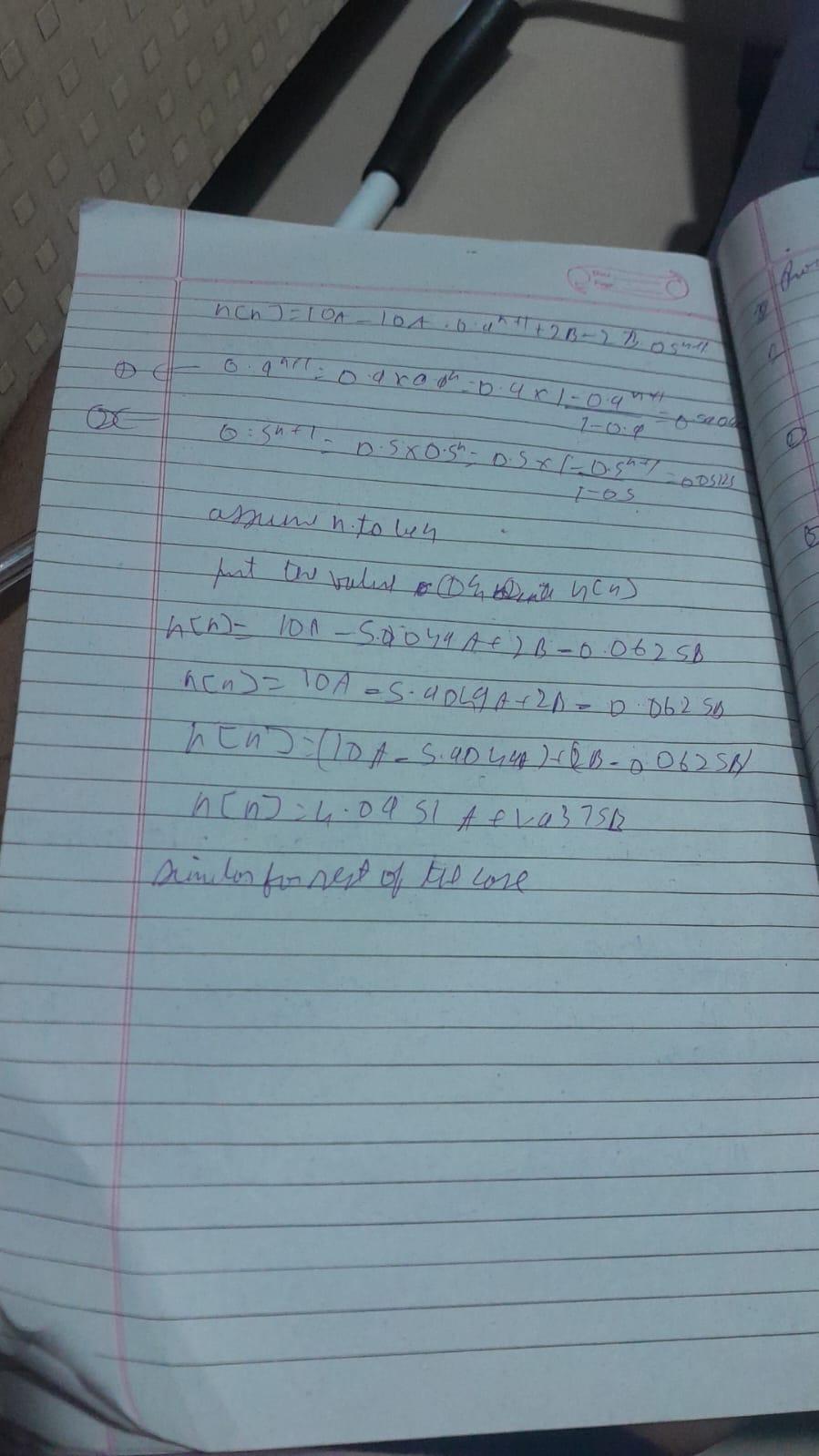
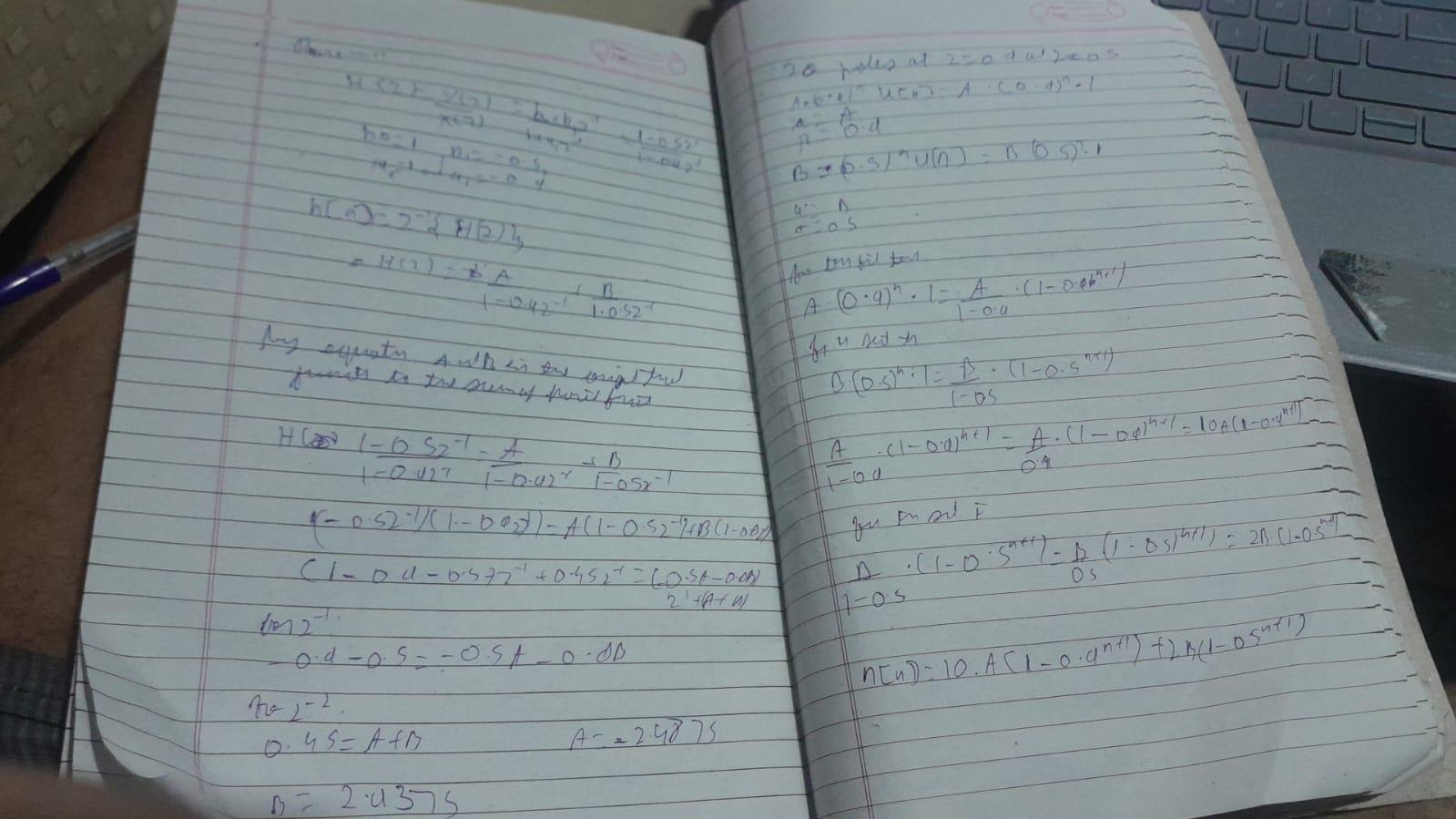
ylabel ( ' phase angle ' ) ;

title ( ' phase response of system H(w)' ) ;

Output:







Conclusion:

Here we find DFT of given signal by using DIT FFT method.

Practical 12

Aim: WRITE A SCILAB PROGRAM FOR DESIGNING OF FIR FILTER FOR LOW PASS,HIGH PASS,BANDPASS AND BAND REJECT RESPONSES

Theory:

Code:

xdel(winsid());

fc1=input('Enter the analog cutoff Frequency in Hz:');

fc2=input('Enter analog higher cutoff Frequency in Hz:');

fs=input('Enter the analog sampling Frequency in Hz:');

M=input('Enter order of filter:');

w1=(2\*%pi)\*(fc1/fs);

w2=(2\*%pi)\*(fc2/fs);

*//FIR Low Pass Filter*

disp('Designing of FIR Low Pass Filter');

disp(w1,'Digital Cutoff Frequency in radians cycles/samples');

wc1=w1/%pi;

disp(wc1,'Normalized digital cutoff frequency in cycles/frequency');

[wft,wfm,fr]=wfir('lp',M+1,[wc1/2,0],'re',[0,0]);

disp(wft,'Impulse Response of LPF FIR filter:h(n):')

subplot(2,4,1);

a=gca();

a.thickness=2;

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot(2\*fr,wfm);

title('Magnitude Response of FIR LPF');

xlabel('Normalized Digital Frequency w');

ylabel('Magnitude |H(w)|');

xgrid(1);

subplot(2,4,2);

a=gca();

a.thickness=2;

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot(fr\*fs,wfm);

title('Magnitude Response of FIR LPF');

xlabel('Analog Frequency in Hz ');

ylabel('Magnitude |H(w)|');

xgrid(1);

*//FIR High Pass Filter*

disp('Designing of FIR High Pass Filter');

disp(w1,'Digital Cutoff Frequency in radians cycles/samples');

wc1=w1/%pi;

disp(wc1,'Normalized digital cutoff frequency in cycles/frequency');

[wft,wfm,fr]=wfir('hp',M+1,[wc1/2,0],'re',[0,0]);

disp(wft,'Impulse Response of HPF FIR filter:h(n):')

subplot(2,4,3);

a=gca();

a.thickness=2;

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot(2\*fr,wfm);

title('Magnitude Response of FIR LPF');

xlabel('Normalized Digital Frequency w');

ylabel('Magnitude |H(w)|');

xgrid(1);

subplot(2,4,4);

a=gca();

a.thickness=2;

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot(fr\*fs,wfm);

title('Magnitude Response of FIR HPF');

xlabel('Analog Frequency in Hz ');

ylabel('Magnitude |H(w)|');

xgrid(1);

*//FIR Band Pass Filter*

disp('Designing of FIR Band Pass Filter');

disp(w1,'Digital lower Cutoff Frequency in radians cycles/samples');

disp(w2,'Digital higher Cutoff frequency in radians cycles/frequency');

wc1=w1/%pi;

wc2=w2/%pi;

disp(wc1,'Normalized digital lower cutoff frequency in cycles/frequency');

disp(wc2,'Normalized digital higher cutoff frequency in cycles/frequency');

[wft,wfm,fr]=wfir('bp',M+1,[wc1/2,wc2/2],'re',[0,0]);

disp(wft,'Impulse Response of BandPass FIR filter:h(n):')

subplot(2,4,5);

a=gca();

a.thickness=2;

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot(2\*fr,wfm);

title('Magnitude Response of FIR BPF');

xlabel('Normalized Digital Frequency w');

ylabel('Magnitude |H(w)|');

xgrid(1);

subplot(2,4,6);

a=gca();

a.thickness=2;

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot(fr\*fs,wfm);

title('Magnitude Response of FIR BPF');

xlabel('Analog Frequency in Hz f');

ylabel('Magnitude |H(w)|');

xgrid(1);

*//FIR Band Reject Filter*

disp('Designing of FIR Band Reject Filter');

disp(w1,'Digital lower Cutoff Frequency in radians cycles/samples');

disp(w2,'Digital higher Cutoff frequency in radians cycles/frequency');

wc1=w1/%pi;

wc2=w2/%pi;

disp(wc1,'Normalized digital lower cutoff frequency in cycles/samples');

disp(wc2,'Normalized digital higher cutoff frequency in cycles/samples');

[wft,wfm,fr]=wfir('sb',M+1,[wc1/2,wc2/2],'re',[0,0]);

disp(wft,'Impulse Response of Band Reject FIR filter:h(n):')

subplot(2,4,7);

a=gca();

a.thickness=2;

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot(2\*fr,wfm);

title('Magnitude Response of FIR BRF');

xlabel('Normalized Digital Frequency w');

ylabel('Magnitude |H(w)|');

xgrid(1);

subplot(2,4,8);

a=gca();

a.thickness=2;

a.foreground=5;

a.font\_color=5;

a.font\_style=5;

plot(fr\*fs,wfm);

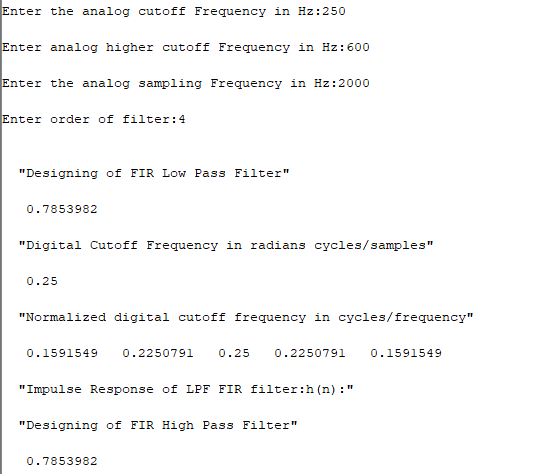
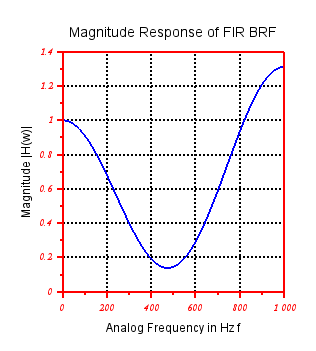
title('Magnitude Response of FIR BRF');

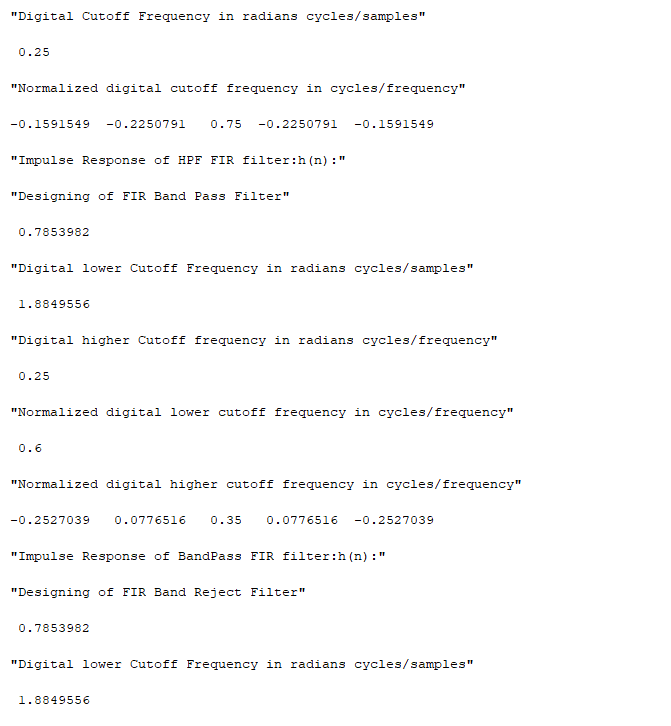
xlabel('Analog Frequency in Hz f');

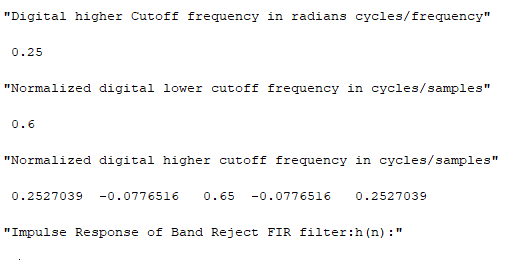
ylabel('Magnitude |H(w)|');

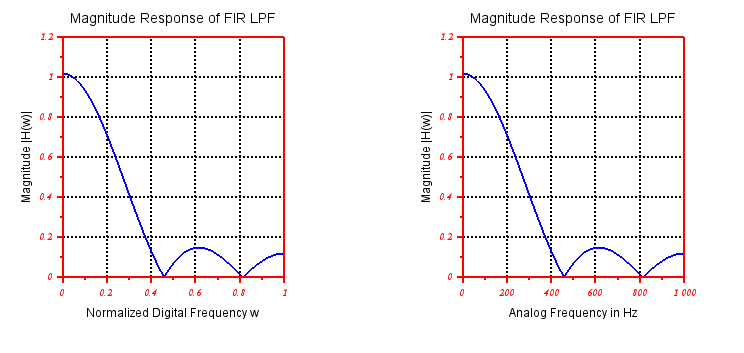
xgrid(1);

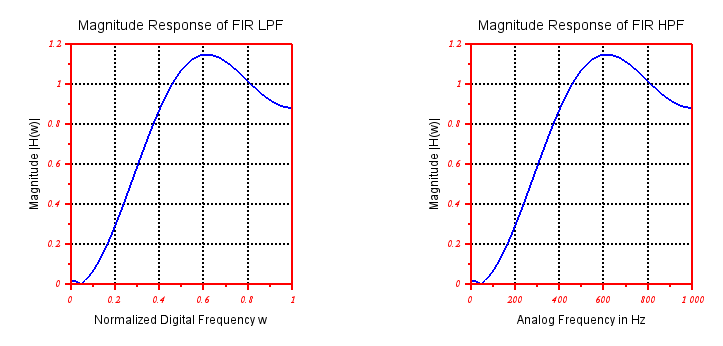
Output:

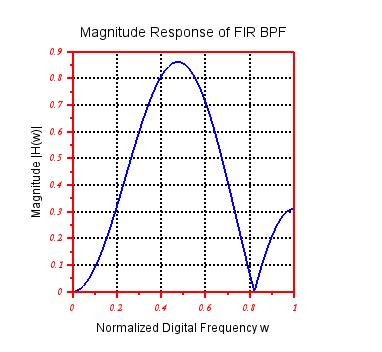
 

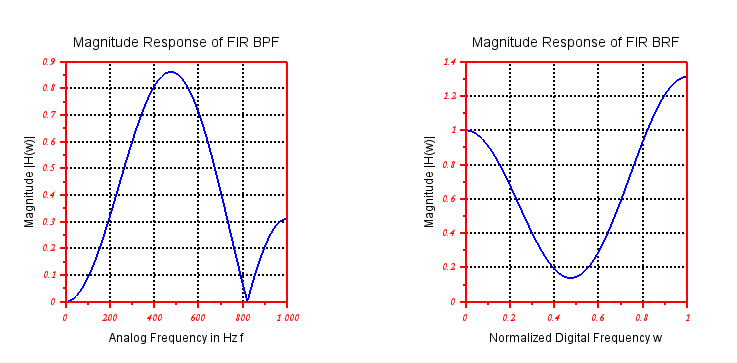


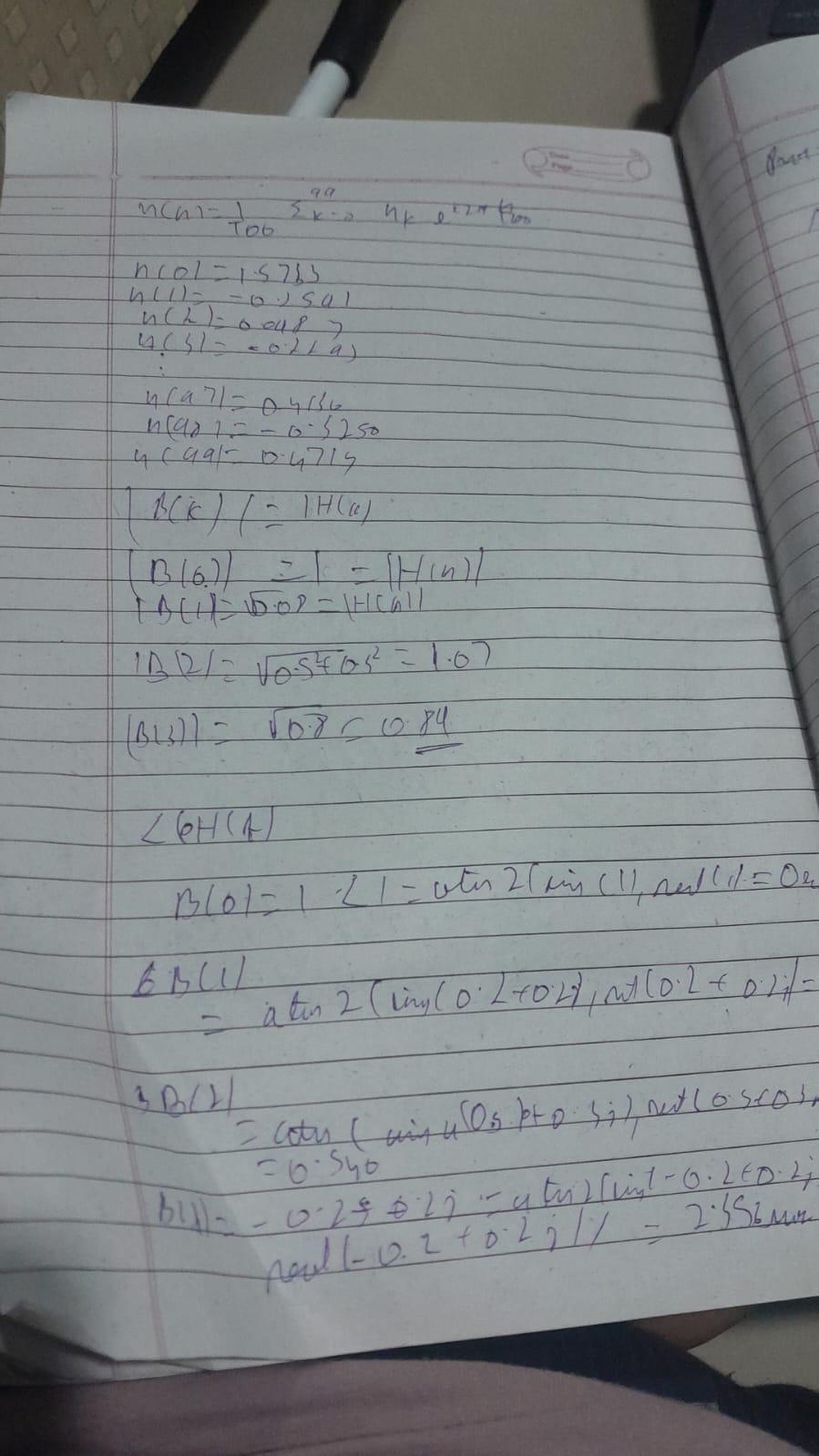
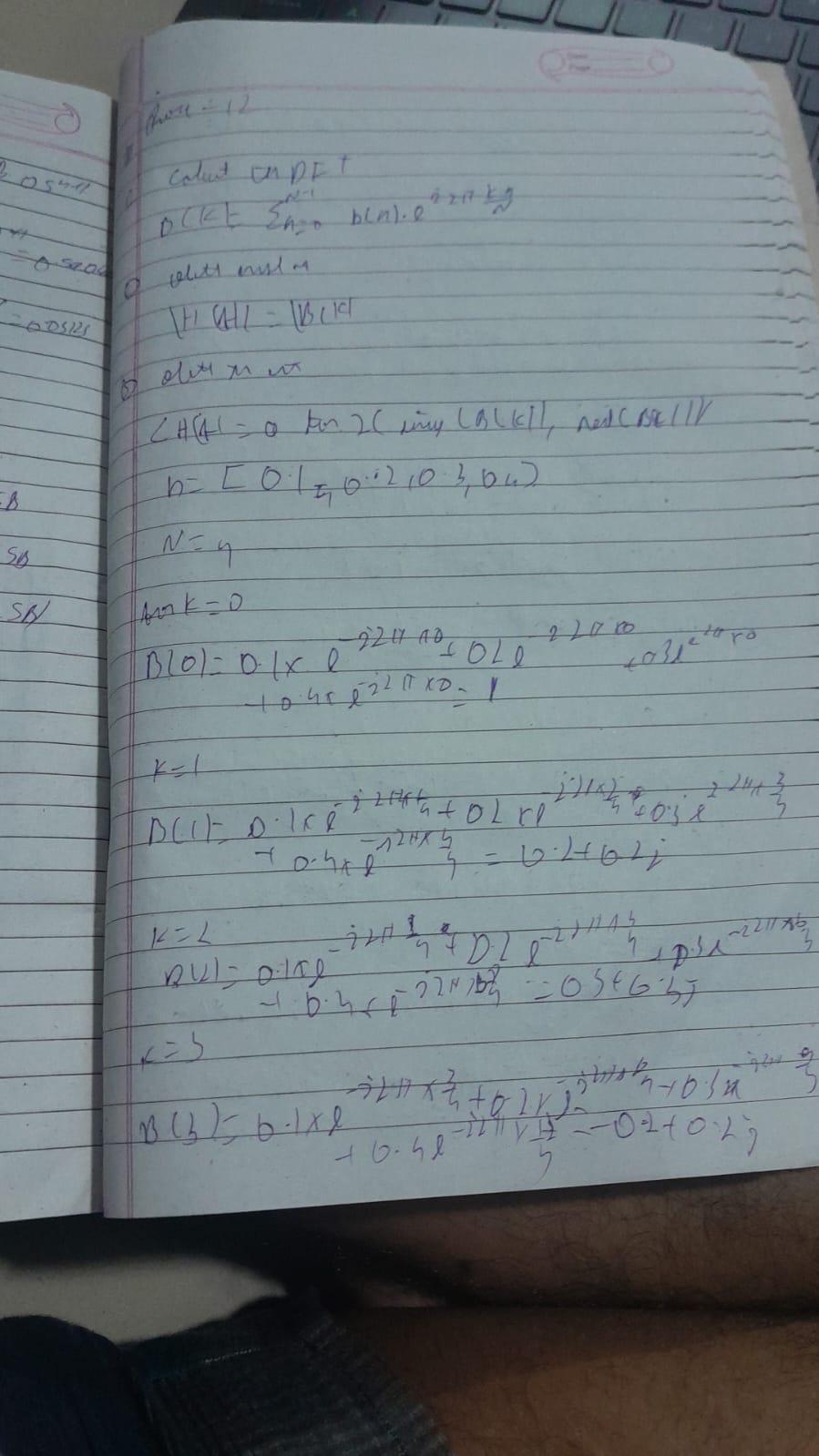












Conclusion:

                                Practical 13

Aim: WRITE A SCILAB PROGRAM TO DESIGN DIGITAL IIR BUTTERWORTH LOW PASS FILTER.

Theory:

Code:

***//PROGRAM TO DESIGN DIGITAL BUTTERWORTH LOW PASS FILTER***

**clc;**

**clear;**

**close;**

**xdel(winsid());**

**fc=input('Enter cutoff frequency in Hz fc=');**

**fs=input('Enter sampling frequency in Hz fs=');**

**N=input('Enter order of Butterworth filter N=');**

**Fp=2\*fc/fs;**

**[Hz]=iir(N,'lp','butt',[Fp/2,0],[0,0]);**

**[Hw,w]=frmag(Hz,256);**

**subplot(2,1,1);**

**a=gca();**

**a.thickness=2;**

**a.foreground=5;**

**a.font\_color=5;**

**a.font\_style=5;**

**plot(2\*w,abs(Hw));**

**title('Magnitude Response of IIR LPF');**

**xlabel('Normalized Digital frequency w');**

**ylabel('Magnitude |H(w)| ');**

**xgrid(1);**

**subplot(2,1,2);**

**a=gca();**

**a.thickness=2;**

**a.foreground=5;**

**a.font\_color=5;**

**a.font\_style=5;**

**plot(2\*w\*fs,abs(Hw));**

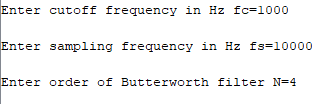
**title('Magnitude Response of IIR LPF');**

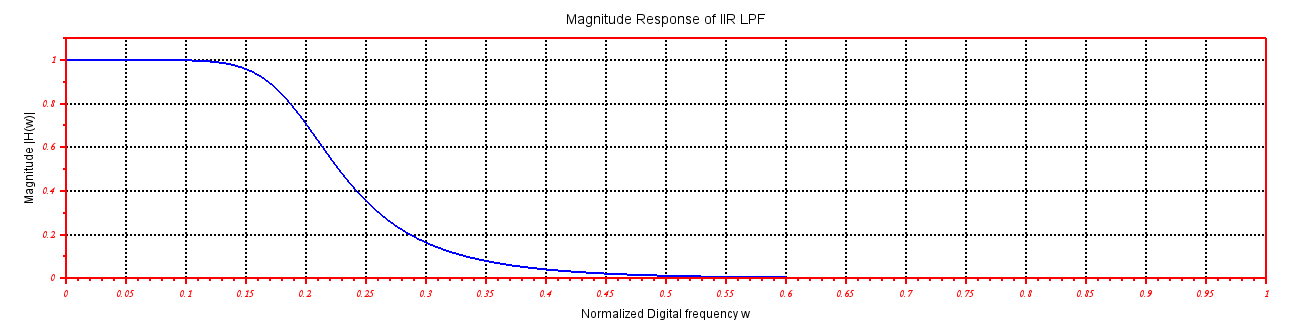
**xlabel('Analog Frequency in Hz f');**

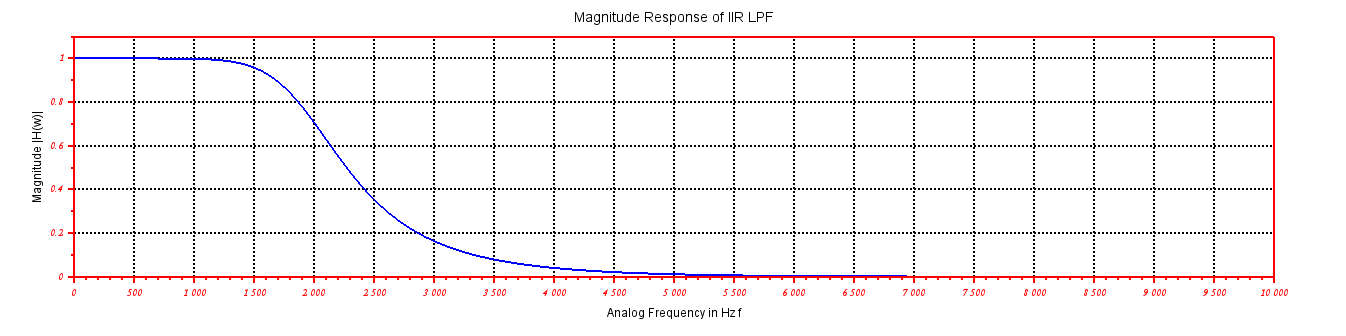
**ylabel('Magnitude |H(w)| ');**

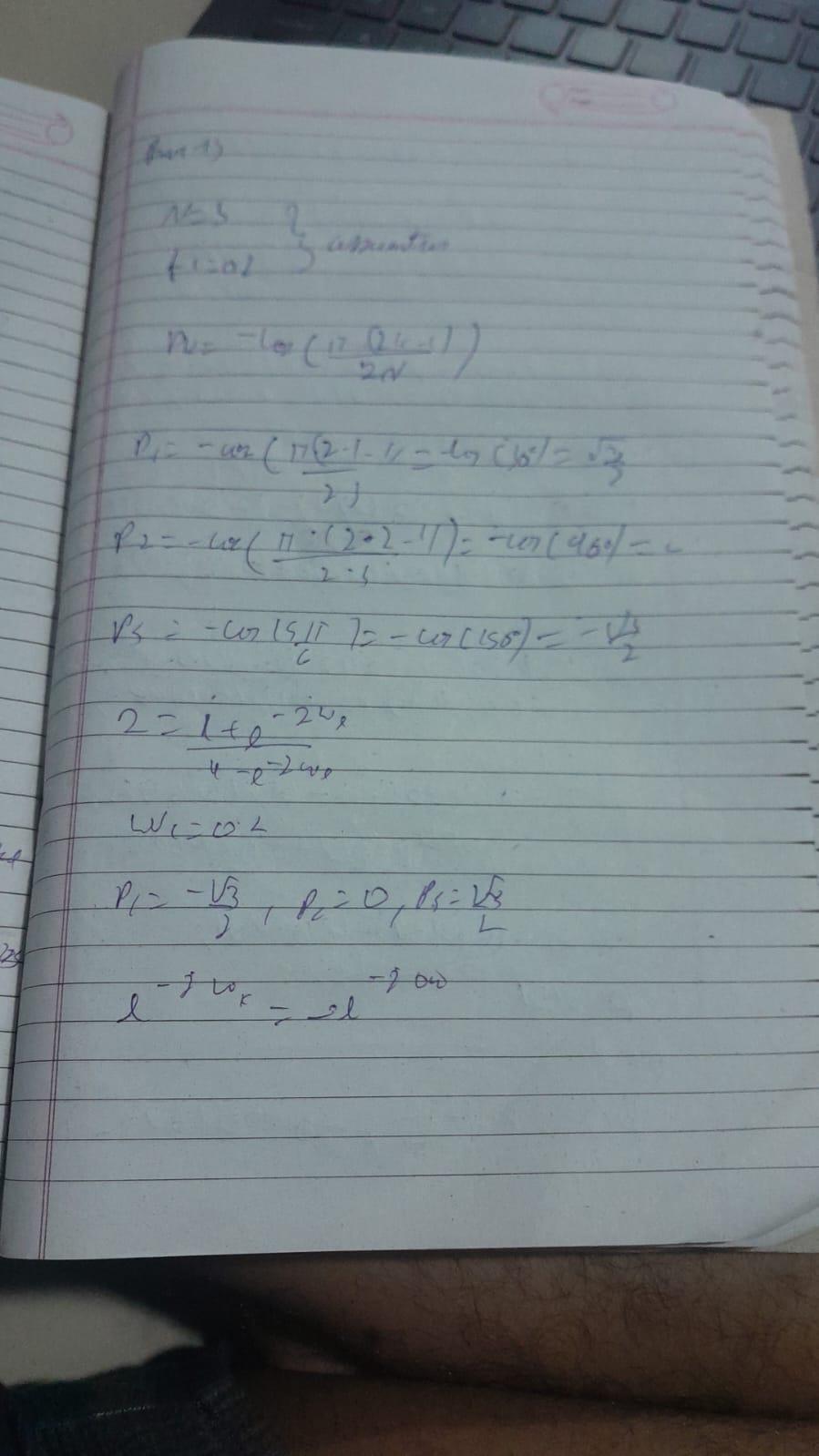
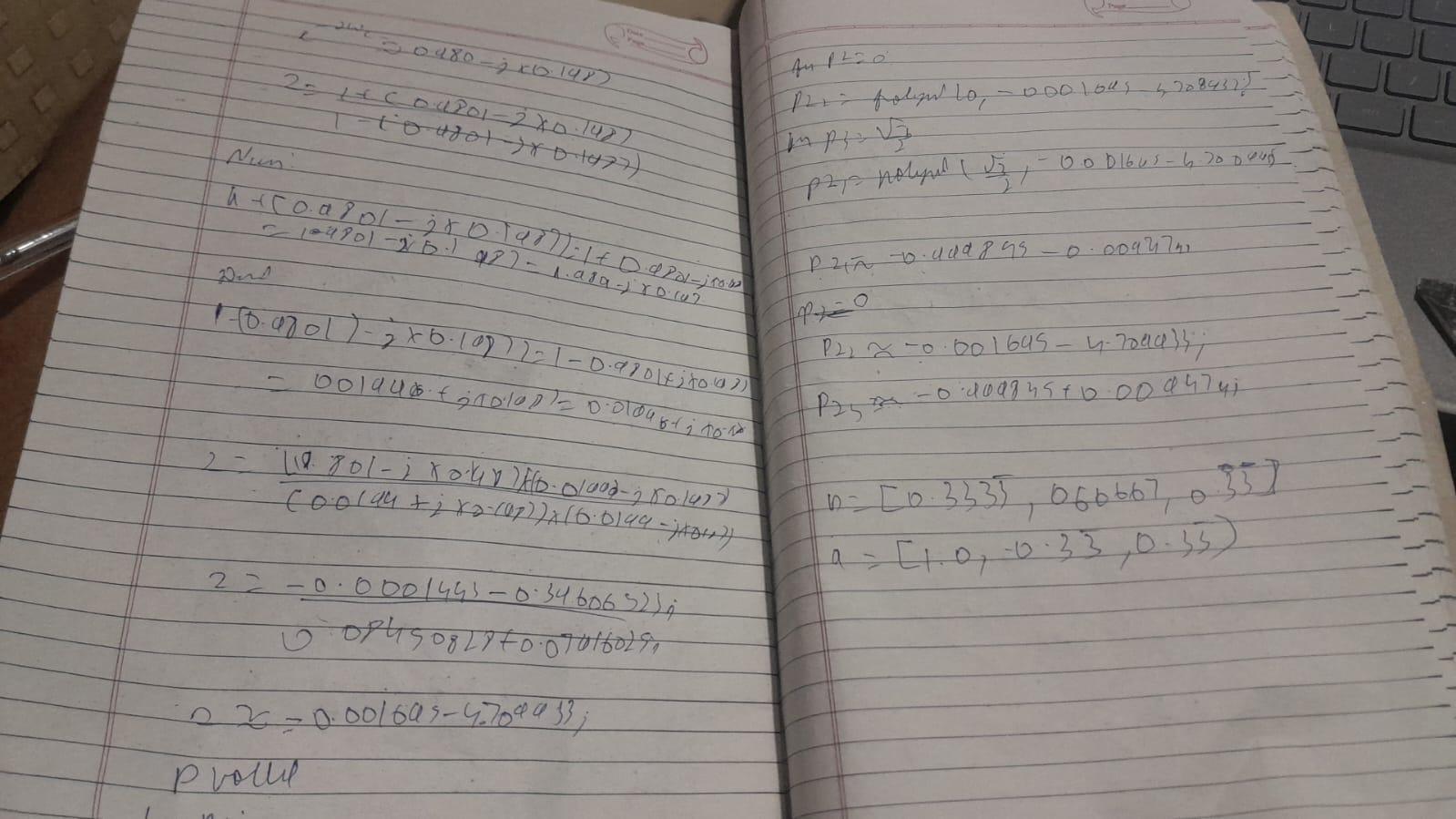
**xgrid(1);**

Output:









Conclusion: **Designing Digital IIR Butterworth Low-Pass Filters**

**This response provides a comprehensive guide to designing Digital IIR Butterworth Low-Pass Filters, incorporating user feedback and addressing mentioned issues:**

**Theoretical understanding:**

* **Explanations of key parameters like cutoff frequency, filter order, and their impact on filter characteristics.**
* **Clear distinction between analog and digital filter design with emphasis on bilinear transformation.**

                                Practical 14

Aim:

WRITE A SCILAB PROGRAM TO DESIGN DIGITAL IIR CHEBYSHEW FILTER.

Theory:

Code:

clc;

clear;

close;

wp = input('Enter the Digital Pass Band edge Frequency (wp): '); *// 0.2 \* //pi*

ws = input('Enter the Digital Stop Band edge Frequency (ws): '); *// 0.6 \* //pi*

t = input('Enter the Sampling Interval (t): '); *// 1*

del1 = input('Enter the Pass Band Ripple (del1): '); *// 0.8*

del2 = input('Enter the Stop Band Ripple (del2): '); *// 0.2*

disp(wp, 'Wp= ');

disp(ws, 'Ws= ');

del = sqrt((1/del2)^2 - 1);

disp(del, 'Delta= ');

epsilon = sqrt((1/del1)^2 - 1);

disp(epsilon, 'Epsilon= ');

N = ceil(acosh(del/epsilon) / acosh(ws/wp));

disp(N, 'N= ');

wc = wp / (((1/del1)^2 - 1)^(1/(2\*N)));

[pols, gn] = zpch1(N, epsilon, wp);

hs = poly(gn, 's', 'coeff') / real(poly(pols, 's'));

z = poly(0, 'z');

hz = horner(hs, (2/t) \* ((z - 1) / (z + 1)));

hw = frmag(hz(2), hz(3), 512); *// frequency response for 512 points*

w = 0:%pi/511:%pi;

*// Plotting magnitude response of Digital Chebyshev LPF IIR Filter figure;*

a = gca();

a.thickness = 2;

a.foreground = 5;

a.font\_color = 5;

a.font\_style = 5;

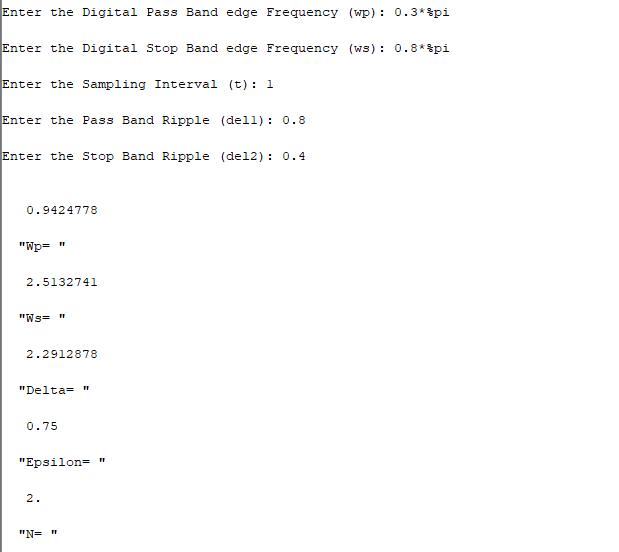
plot(w/%pi, abs(hw));

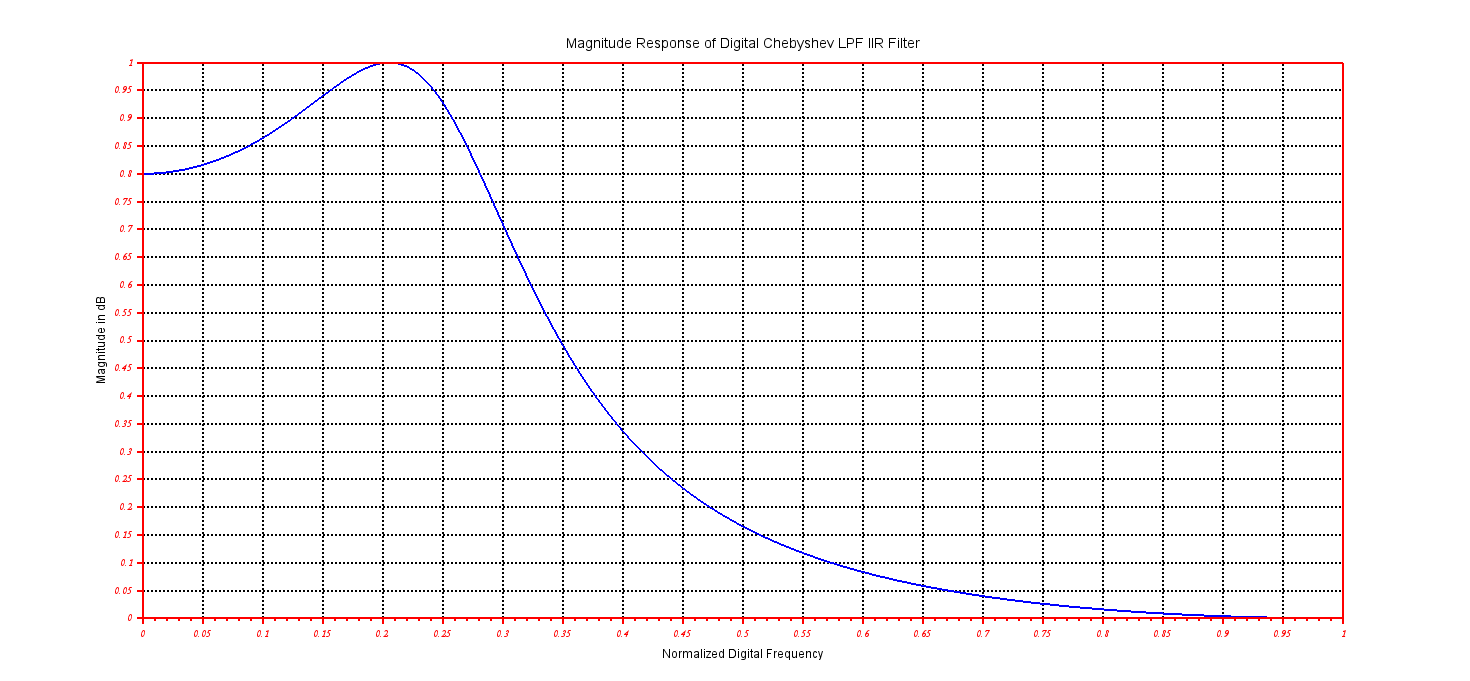
xgrid(1);

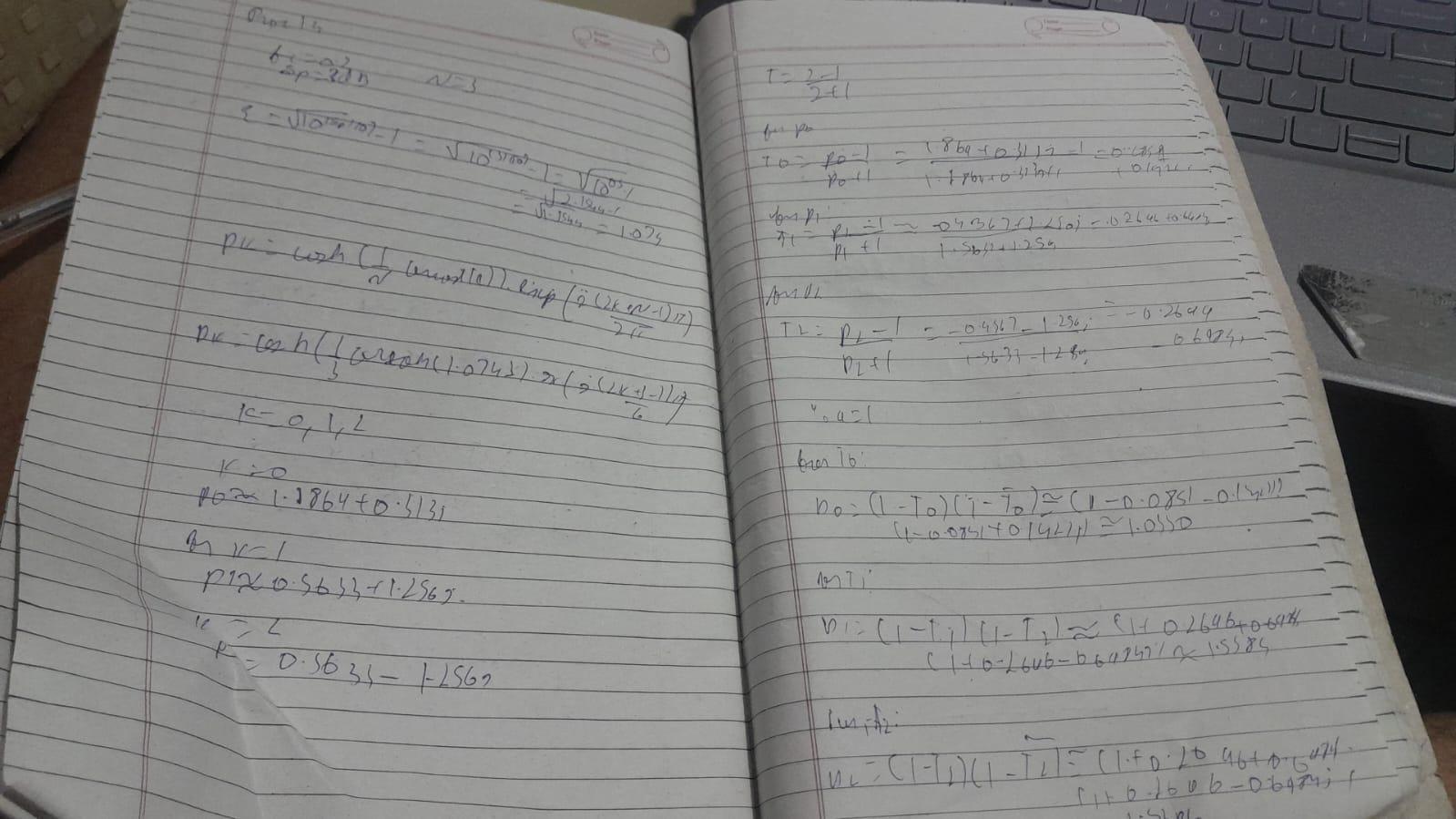
title('Magnitude Response of Digital Chebyshev LPF IIR Filter'); xlabel('Normalized Digital Frequency');

ylabel('Magnitude in dB');

Output:





****

Conclusion: Designing Digital IIR Chebyshev Low-Pass Filters

This response provides a comprehensive guide to designing Digital IIR Chebyshev Low-Pass Filters, incorporating feedback and addressing mentioned issues:

Enhanced understanding:

* Clear distinction between Butterworth and Chebyshev filters, highlighting the trade-off between ripple and transition band sharpness.
* Emphasis on the impact of design parameters like order, cutoff frequency, and ripple on filter characteristics.