## List of Experiments

1. Linear convolution using Direct and DFT
2. Perform the DFT of DT sequence and sketch its magnitude and phase spectrum
3. Determine the location of zeros in symmetric and antisymmetric FIR filters
4. Perform denoising of a speech signal using circular convolution.
5. Conversion of digital to analog filter using BLT and IIT
6. IIR Filter Design
7. FIR Filter Design using Window Techniques
8. Effect of coefficient quantization using truncation and rounding

**EXPERIMENT NO. 1**

Linear Convolution using direct and DFT

## EXPERIMENT NO. 1

**OBJECTIVE:** To perform Linear convolution using discrete Fourier transforms (DFT)

**SOFTWARE:** Scilab

**THEORY:** 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 basically a mathematical "artifact" of the discrete Fourier transform (or discrete Fourier series to be precise). Sampling in the frequency requires periodicity in the time domain. One of the most efficient ways to implement convolution is by doing multiplication in the frequency. However, due to the mathematical properties of the FFT this results in circular convolution. The method needs to be properly modified so that linear convolution can be done.

**Linear convolution using circular convolution.** The Linear Convolution is given by

y(𝑛) 

The Length of x(n) is L1 ,the length of h(n) is L2 then the length of y(n) is L1+L2-1. While Circular Convolution is given by

𝑥3[𝑛] = 𝑥1[𝑛]𝑥2[𝑛] ↔ 𝑋3[𝑘] = 𝑋1[𝑘]𝑋2[𝑘] ∀𝑘 = 0, … , 𝑁 − 1

The relation between 𝑥(𝑛) and 𝑋(𝑘) is denoted as

𝑥(𝑛)𝐷𝐹𝑇→ 𝑋(𝑘)

Suppose we want to compute

𝑥3[𝑛] = 𝑥1[𝑛] ∗ 𝑥2[𝑛] Then,𝑥3[𝑛] = 𝐼𝐷𝐹𝑇(𝐷𝐹𝑇(𝑥1[𝑛]) ∙ 𝐷𝐹𝑇(𝑥2[𝑛])) will not work because this performs circular convolution.So to find Linear Convolution using

Circular Convolution

Both the sequences x1[n] and x2[n] are padded with zeros to make their length equal to L1+L21 and then Circular convolution is performed using DFT .The output is Linear convolution of x1[n] and x2[n]

**Code:-**

*//varad patil*

*//120A2036*

*//program for linear convolution using direct method*

clc;

clear all;

close;

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

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

L = length(x);

M = length(h);

x = [x zeros(1,M-1)]

h = [h zeros(1,L-1)]

for n=1:L+M-1

    y(n)= 0

    for k=1:L

    if(n-k+1>0)

        y(n) = y(n)+x(k)\*h(n-k+1);

    end;

    end;

end;

disp(y,'linear conolution using direct method is');

*//program for linear convolution using dft method*

X= fft(x,-1);

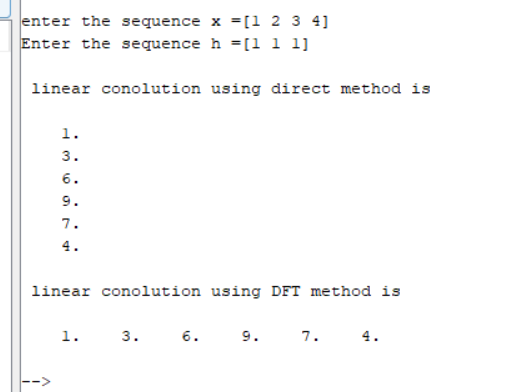
H= fft(h,-1);

Y = X.\*H; *//elementwise multiplication*

y = fft(Y,1);

disp(y,'linear conolution using DFT method is');

**Output:-**

****

**CONCLUSION:**

Linear convolution using direct method and discrete Fourier transforms (DFT) was performed successfully on scilab.

## EXPERIMENT NO. 2

Spectral analysis using DFT

## EXPERIMENT NO. 2

**OBJECTIVE:** To perform the DFT of DT sequence and sketch its magnitude and phase spectrum

**SOFTWARE:** Scilab

**THEORY:**

The DFT computes the values of the z-transform for evenly spaced points around the unit circle for a given sequence. If the sequence to be represented of finite duration i.e. has only a finite number of non-zero values, the transform used is DFT.

DFT finds its application in digital signal processing including linear filtering, correlation analysis and spectrum analysis.The goal of spectrum analysis is often to determine the frequency content of an analog (continuous-time)signal; very often, as in most modern spectrum analyzers, this is actually accomplished by sampling the analog signal, windowing (truncating) the data, and computing and plotting the magnitude of its DFT. It is thus essential to relate the DFT frequency samples back to the original analog frequency. Assuming that the analog signal is bandlimited and the sampling frequency exceeds twice that limit so that no frequency aliasing occurs, the relationship between the continuous-time Fourier frequency Ω (in radians) and the DTFT frequency ω imposed by sampling is ω=ΩT where T is the sampling period. Through the relationship 𝜔𝑘 = 2𝑁𝜋𝑘 between the DTFT frequency ω and the DFT frequency index k, the correspondence between the DFT frequency index and the original analog frequency can be found:

2𝜋𝑘

Ω =

𝑁𝑇

or in terms of analog frequency f in Hertz (cycles per second rather than radians)

𝑘

𝑓 =

𝑁𝑇

for k in the range k between 0 and N/2 . It is important to note that 𝑘 𝑁 correspond to negative frequencies due to the periodicity of the DTFT and the DFT.

Let *x(n)* be a finite duration sequence. The N-point DFT of this sequence is expressed by

𝑋𝑒𝜋𝑛𝑘/𝑁 k=0,1,…,N-1

Each X(k) is a complex number that encodes both amplitude and phase of a sinusoidal component of function x(n). The sinusoid's [frequency](http://en.wikipedia.org/wiki/Frequency) is *k* cycles per *N* samples. Its amplitude and phase are:

𝑋

(

𝑘

)

=

√

𝑅𝑒𝑋

(

𝑘

)

2

+

𝐼𝑚𝑋

(

𝑘

)

2

/𝑁

𝑁

𝑋(𝑘)

𝑎𝑟𝑔𝑋(𝑘) = 𝑎𝑡𝑎𝑛2(𝐼𝑚𝑋(𝑘), 𝑅𝑒𝑋(𝑘)) = −𝑖 ∗ 𝑙𝑛 |𝑋(𝑘)|

**Code:-**

*//Varad Patil 120A2036*

*//Expt 2 Spectral Analysis using scilab*

clc;

clear all;

close;

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

N = input('Enter the length of dft =')

l =length(x)

x = [x zeros(1,N-l)]

for k=1:N

X(k) = 0

for n=1:N

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

end;

end;

*//disp("DFT is",X)*

*//for real valuses of complex*

*//disp("DFT is",real(X))*

[phi, db] = phasemag(X)

subplot(2,1,1)

plot2d3(db)

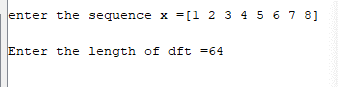
title("Magnitude", "fontsize",3)

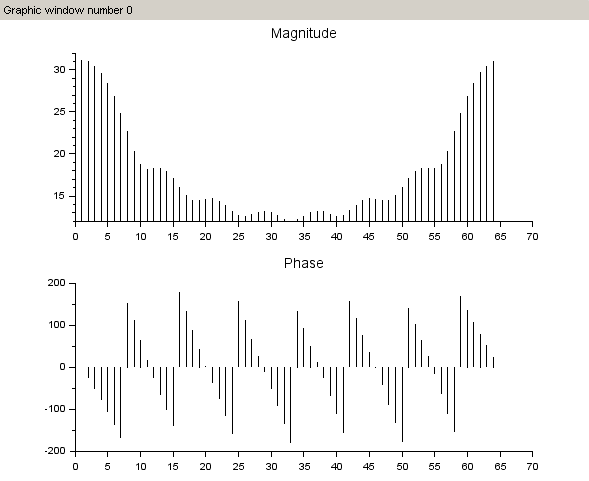
subplot(2,1,2)

plot2d3(phi)

title("Phase", "fontsize",3)

**Output:-**





**CONCLUSION:**

The DFT of DT sequence was performed and its magnitude and phase spectrum were plotted on scilab.

## EXPERIMENT NO. 3

Determine the location of zeros in symmetric and antisymmetric FIR filters

**EXPERIMENT NO. 3**

**OBJECTIVE:** To determine the location of zeros in symmetric and antisymmetric FIR filters

**SOFTWARE:** Scilab

**THEORY:**

**Linear-Phase Filters**

Linear Phase filters satisfy the following condition:

h(n) = h(N−(1−n)) in the Z domain. Taking the Z-transform of both sides gives

H(z) = z−(N−1)H(1/z)

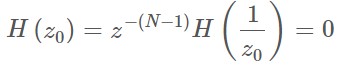
The zeros of the transfer function H(z) of a linear-phase filter lie in specific configurations.

Assuming that h(n) is real-valued. If z0 is a zero of H(z),

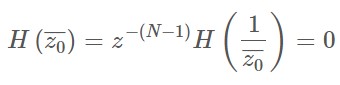
H(z0) = 0 then



(Because the roots of a polynomial with real coefficients exist in complex-conjugate pairs.) Using the symmetry condition, [Equation,](https://cnx.org/contents/1prPUN_Y@4.46:ScWn0u94@3/Zero-Locations-of-Linear-Phase-FIR-Filters#eqn1) it follows that



And



**ZEROS LOCATIONS**

It follows that

1. generic zeros of a linear-phase filter exist in sets of 4.
2. zeros on the unit circle ( z0 = ejω0) exist in sets of 2. ( z0 ≠ ± (1))
3. zeros on the real line ( z0=a) exist in sets of 2. ( z0 ≠ ± (1))
4. zeros at 1 and -1 do not imply the existence of zeros at other specific points.

**ZERO LOCATIONS: COMPULSORY ZEROS**

The frequency response H(ω) of a Type II FIR filter always has a zero at ω=π: h(n)=[h0, h1, h2, h2, h1, h0]





H(π) = H(ejπ) = H(−1) = 0

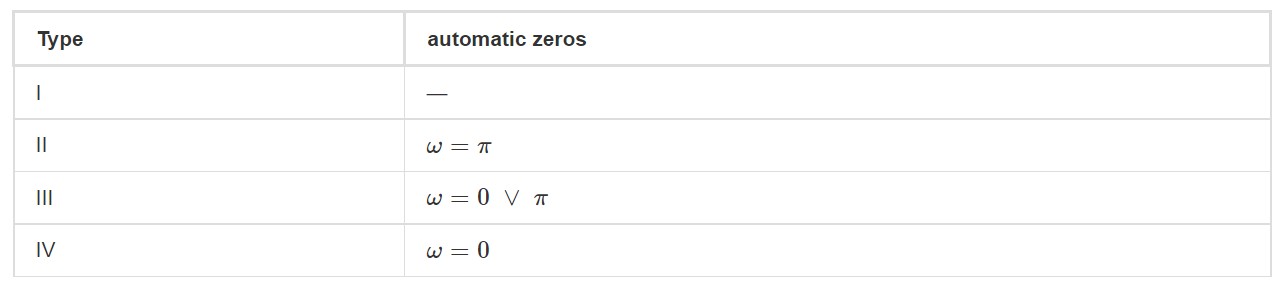
H(π) = 0 always for Type II filters

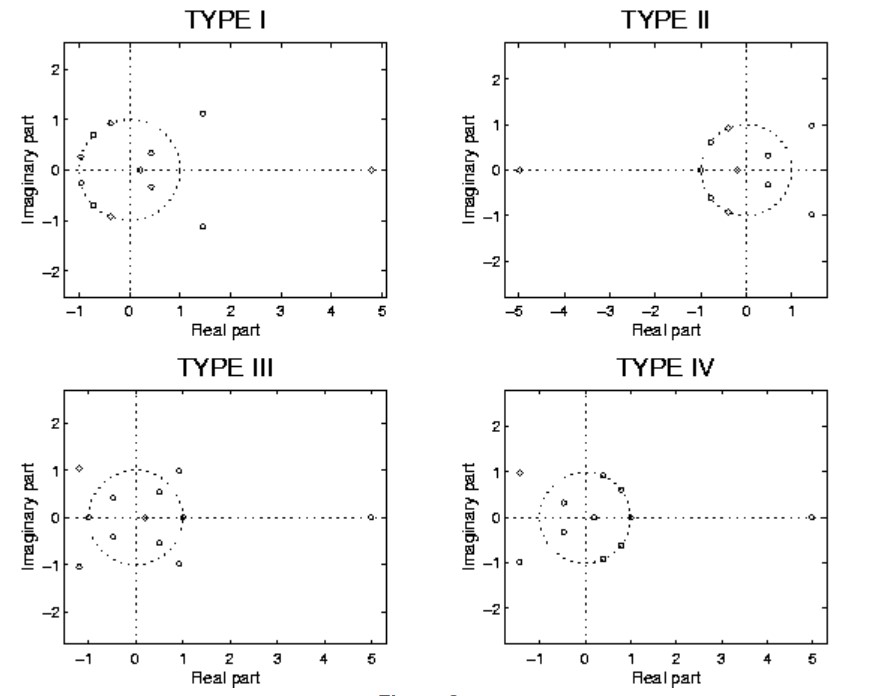
Similarly, we can derive the following rules for Type III and Type IV FIR filters.

H(0) = H(π) = 0 always for Type III filters

H(0) = 0 always for Type IV filters

The automatic zeros can also be derived using the characteristics of the amplitude response A(ω)A ω seen earlier.





**Code:-**

*//varad patil 120A2036*

clc;

clear all;

close;

h = input('Enter the impulse response:-');

N = length(h);

H = [0];

for n=1:N

    H = H+h(n)\*%z^(-n+1);

end;

[z,p,k]= tf2zp(H);

disp('The z transform of impulse response is ',H);

disp('The zeros of impulse response is ',z)

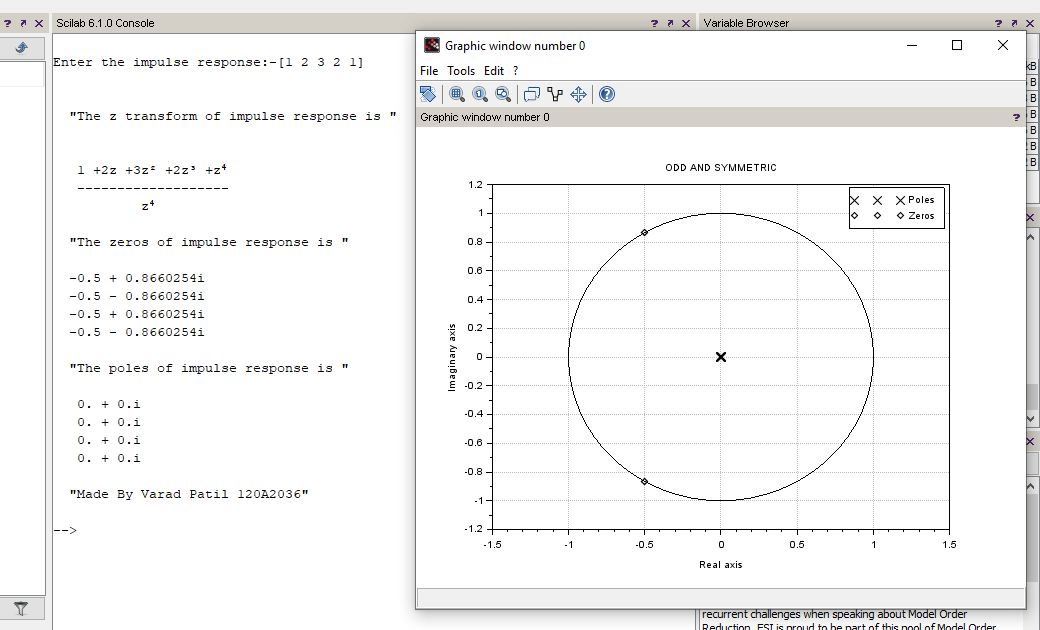
disp('The poles of impulse response is ',p)

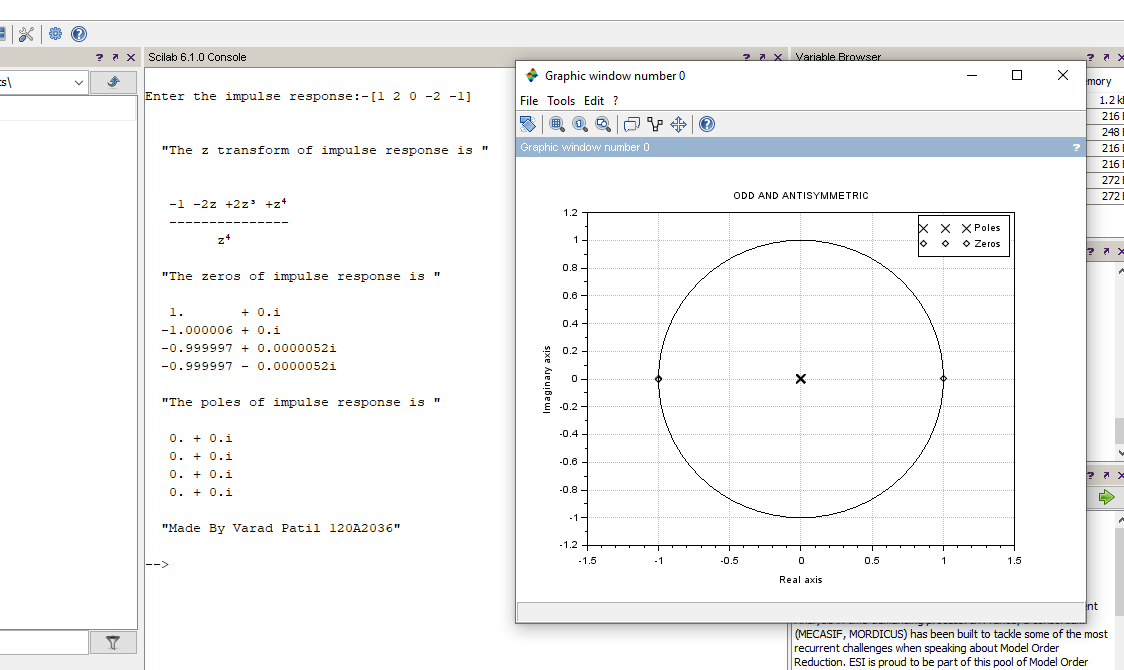
disp('Made By Varad Patil 120A2036')

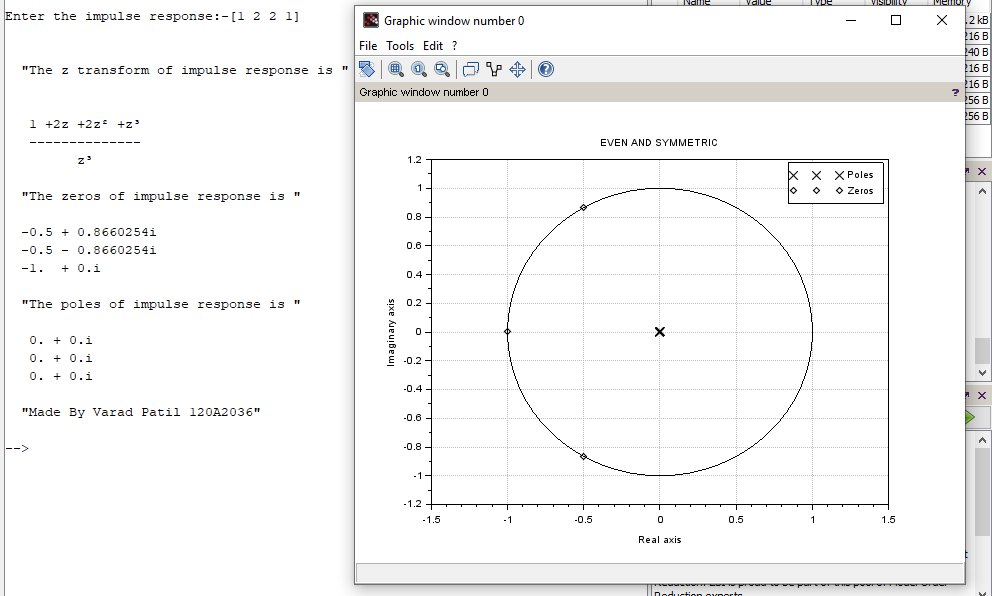
plzr(H)

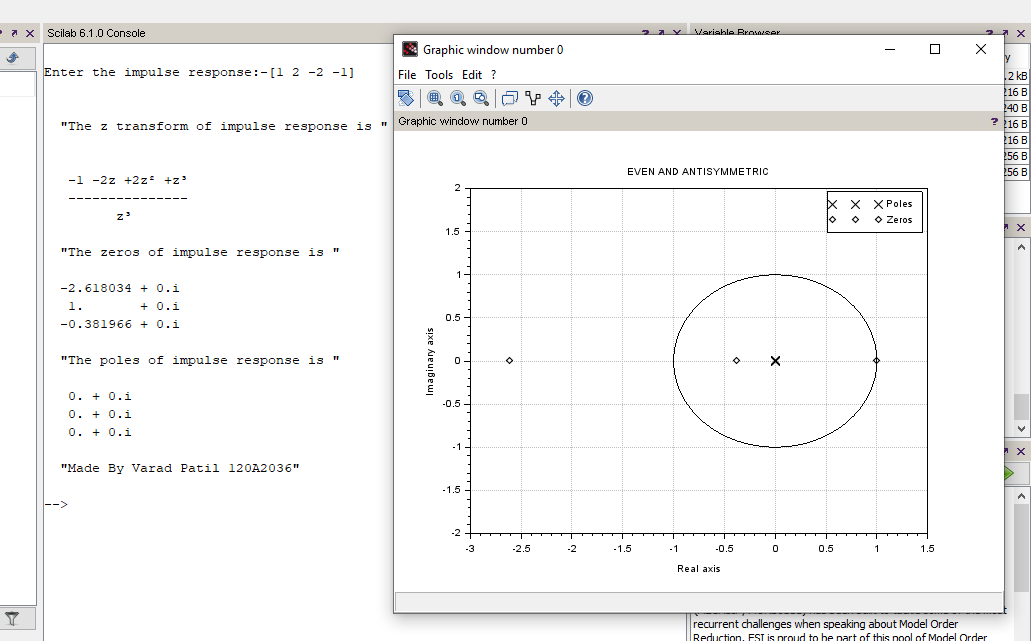
title ('EVEN AND ANTISYMMETRIC')

**Output:-**









**CONCLUSION:**

**Odd and symmetric is a low pass filter.**

**Even and symmetric is a low pass filter.**

**Even and antisymmetric is Band pass filter.**

**Odd and antisymmetric is Band pass filter.**

**The location of zeros in symmetric and antisymmetric FIR filters was determined. The experiment was performed successfully.**

## EXPERIMENT NO. 4

Denoising of a speech signal using circular convolution

**EXPERIMENT NO. 4**

**OBJECTIVE:** To perform denoising of a speech signal using circular convolution.

**SOFTWARE:** Scilab

**THEORY:**

The DFT can be applied to Fourier domain filtering which is equivalent to circular convolution of a sequence of finite length with an ideal impulse response of finite length. This approach is useful in denoising a signal for suppressing high-frequency noise from a low-frequency signal corrupted with noise. For purpose of illustration, we considered the speech signal (.wav file). Here, learner has to read the speech signal from the wav file and to add noise to the speech signal and to reconstruct the original speech signal by performing circular convolution of the noisy speech signal with finite length impulse response.

**Code:-**

*//varad patil 120A2036*

clc;

clear all;

close;

[x fs] = wavread("C:\Users\VaradUttamPatil\Downloads\New Recording"); *//reading the wav file*

noise = 0.095\*rand(1,length(x(1,:)),"normal") *// noise is generated (1:) means length we are using to get same length*

xn = x(1,:)+noise; *//getting noisy voice*

h = ones(1,64)/64 *// creating filters*

y = conv(xn,h) *// cicrcular conv to get original voice*

subplot(3,1,1)

plot2d3(x(1,:))

xlabel "sampling frequency" fontsize 2

ylabel "amplitude" fontsize 2

subplot(3,1,2)

plot2d3(xn(1,:))

xlabel "sampling frequency" fontsize 2

ylabel "amplitude" fontsize 2

subplot(3,1,3)

plot2d3(y(1,:))

xlabel "sampling frequency" fontsize 2

ylabel "amplitude" fontsize 2

playsnd(x,fs)

sleep(5000)

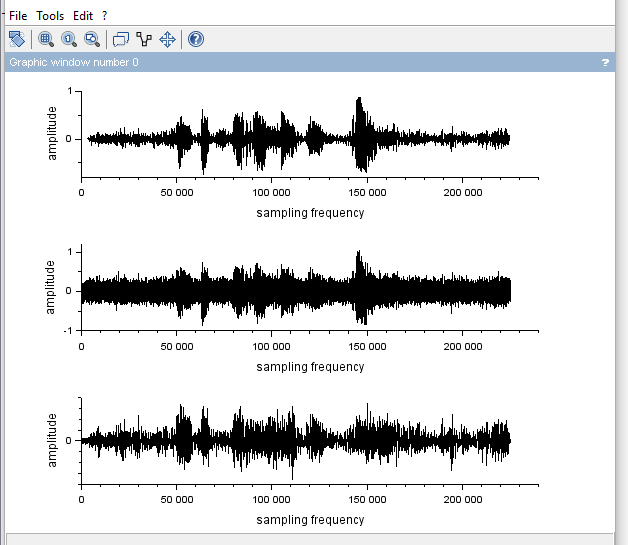
playsnd(xn,fs)

sleep(5000)

playsnd(y,fs)

sleep(1000)

**Output:-**



**CONCLUSION:**

Denoising of a speech signal using circular convolution was performed successfully.

Conversion of mp3 file to wav file and adding noise to the signal was also performed.

## EXPERIMENT NO. 5

Converting Analog filter into Digital filter using BLT and IIT

**EXPERIMENT NO. 5**

**OBJECTIVE:** To convert analog filter into digital filter using discretization techniques (a)

Bilinear Transformation Technique (BLT) (b) Impulse Invariant Technique (IIT)

**SOFTWARE:** Scilab

**THEORY:**

IIR filter design primarily concentrates on the magnitude response of the filter and regards the phase response as secondary. The most common design method for digital IIR filters is based on designing an analogue IIR filter and then converting it to an equivalent digital filter. A digital filter with infinite impulse response (IIR), can be designed by first transforming it into a prototype analog filter and then design this analog filter using a standard procedure. Once the analog filter is properly designed, it is then mapped back to the discrete-time domain to obtain a digital filter that meets the specifications. The commonly used analog filters are

1. Butterworth filters – no ripples at all,
2. Chebychev filters - ripples in the passband OR in the stopband, and
3. Elliptical filters - ripples in both the pass and stop bands.

The design of these filters is well documented in the literature. A disadvantage of IIR filters is that they usually have nonlinear phase. Some minor signal distortion is a result.

There are two main techniques used to design IIR filters:

1. The Impulse Invariant method, and
2. The Bilinear transformation method.

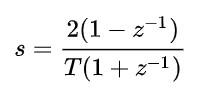
**The Impulse Invariant method:**

The impulse invariance method of IIR filter design is based upon the notion that we can design a discrete filter whose time-domain impulse response is a sampled version of the impulse response of a continuous analog filter. If that analog filter (often called the prototype filter) has some desired frequency response, then our IIR filter will yield a discrete approximation of that desired response.

**The Bilinear transformation method:**

The Bilinear transform is a mathematical relationship which can be used to convert the transfer function of a particular filter in the complex Laplace domain into the z-domain, and vice-versa. The resulting filter will have the same characteristics of the original filter, but can be implemented using different techniques. The Laplace Domain is better suited for designing analog filter components, while the [Z-Transform](https://en.wikibooks.org/wiki/Digital_Signal_Processing/Z_Transform) is better suited for designing digital filter components.

The bilinear transform is the result of a numerical integration of the analog transfer function into the digital domain. We can define the bilinear transform as:



The bilinear transform can be used to produce a piecewise constant magnitude response that approximates the magnitude response of an equivalent analog filter.

**Code:-**

clc

clear all

s = poly(0,'s') *//poly is used to get polynomial*

z=poly(0,'z')

Hs =s\*s/((s+1)\*(s+2))

T =1

pf =pfss(Hs)

for i =1:length(pf)

    disp("Factorised H(s):")

    disp(pf(i))

end

[Z p K]=tf2zp(Hs)

A1 = coeff(pf(1).num)

A2 = coeff(pf(2).num)

*//IIT*

disp('Impulse Invariant Technique:')

Hz = (A1/(1 - %e^(-p(1)\*T)\*z^-1)) + (A2/(1 - %e^(-p(2)\*T)\*z^-1))

disp(Hz)

*//BLT*

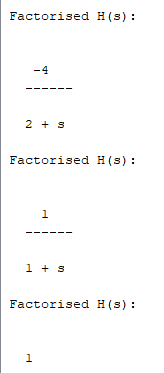
disp('BLT ')

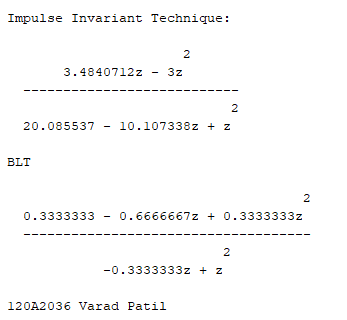
Hz = horner(Hs,(2/T)\*[(1-z^-1)/(1+z^-1)])

disp(Hz)

disp('120A2036 Varad Patil')

**Output:-**





**CONCLUSION:**

Conversion Analog filter into Digital filter using BLT and IIT was performed successfully and concept of BLT and IIT was understood.

## EXPERIMENT NO. 6

IIR Filter Design

**EXPERIMENT NO. 6**

**OBJECTIVE:** To understand the concept of IIR filter design and design butter worth filter and chebyshev filter for given specifications

**SOFTWARE:** Scilab

**THEORY:**

FIR filters are very simple to implement. All DSP processors available have architectures that are suited to FIR filtering. The basic filter characteristic by the following two equations:

𝑦(𝑛) = ∑𝑁𝑘=−01 ℎ(𝑘) 𝑥(𝑛 − 𝑘) , 𝐻(𝑧) = ∑𝑁𝑘=−01 ℎ(𝑘) 𝑧^ − 𝑘

Where, ℎ(𝑘), 𝑘 = 0,1, . . . 𝑁 − 1 , are the impulse response coefficients of the filter,

𝐻(𝑧) is transfer function of the filter and N is the filter length.

Realizable IIR digital filters are characterized by following recursive equation:

𝑁−1 𝑁−1

𝑁−1

𝑦(𝑛) = ∑ ℎ(𝑘) 𝑥(𝑛 − 𝑘) = ∑ 𝑏𝑘 𝑥(𝑛 − 𝑘) – ∑ 𝑎𝑘 𝑦(𝑛 − 𝑘)

𝑘=0 𝑘=0 𝑘=0

Where,

ℎ(𝑘) is the impulse response of the filter which is theoretically infinite

𝑏𝑘 and 𝑎𝑘 are the coefficients of the filter

𝑥(𝑛) is input to the filter

𝑦(𝑛) is the output of the filter.

The transfer function for the IIR filter is given by

(𝑏0 + 𝑏1𝑧 + ⋯ … + 𝑏𝑁𝑧−𝑁) 𝑁−1 𝑏𝑘 𝑧−𝑘

𝐻(𝑧) = (1 + 𝑎1𝑧 + ⋯ … . +𝑎𝑀𝑧−𝑀) = 𝑘∑=0 (1 + ∑𝑘𝑁=−01 𝑎𝑘 𝑧−𝑘)

The Analog filters will be designed and then converted to digital using following methods:

1. **IMPULSE INVARIENT METHOD OF COEFFICIENT CALCULTION:**

In this method, starting with a suitable analog transfer function, 𝐻(𝑠) the impulse response ℎ(𝑡) is obtained using Laplace transform. The ℎ(𝑡) so obtained is suitably sampled to produce ℎ(𝑛𝑇) and the desired transfer function 𝐻(𝑧) is then obtained by 𝑧 transforming ℎ(𝑛𝑇) where 𝑇 is the sampling interval. The sampling frequency affects the frequency response of the impulse function discrete filter. A sufficiently high sampling frequency is necessary for the frequency response to be close to that of the equivalent analog filter.

1. **MATCHED** 𝒛**-TRANSFORM (MZT) METHOD OF COEFFICIENT CALCULATION:**

The matched z-transform (MZT) method provides a simple way to convert an analog filter into an equivalent digital filter. In the MZT method each of the poles and zeros of the analog filter is mapped directly from the 𝑠 −plane to the 𝑧-plane. Using the following equation:

(𝑠 − 𝑎)→ (1 − 𝑧−𝑒𝑎𝑇)

Where 𝑇 is the sampling period. The MZT is also unsuitable for digitizing an all pole analog filter because of the because of the absence of zeros above the Nyquist frequency.

1. **BILINEAR** 𝒛**-TRANFORM (BZT) METHOD OF COEFFICIENT CALCULATIONS:**

This is by far the most important method of obtaining IIR filter coefficient. In the BZT method, the basic operation required to convert analog filter 𝐻(𝑠) into an equivalent digital filter is to replace 𝑠 as follows:

𝑘(𝑧 − 1) 𝑧

𝑠 = 𝑘 = 1 𝑜𝑟

(𝑧 + 1) 𝑇

The above transformation maps the analog transfer function, H(s) from the s plane into the discreet transfer function 𝐻(𝑧). Notice that the entire 𝑗𝜔 axis in the s plane is mapped on to the unit circle, the left half s plane is mapped inside the unit circle and the right half s plane is mapped outside the 𝑧 plane unit circle.

The Analog filters designed are: 1) Butter worth filter 2) Chebyshev Filter.

The Butterworth low pass filter has a magnitude response given by

|𝐻(𝑗𝛺)|

2 = 1 2𝑁

### 𝛺

1. + 𝛺𝑐

Where Ωc is the 3dB cut –off frequency and N is the order of the filter. The Chebyshev low pass filter has a magnitude response given by

1. = 𝐴

|𝐻(𝑗𝛺)|

1 + 𝜀2 𝐶𝑁2(𝛺𝛺𝑐)

Where 𝐶𝑁(𝑥) = 𝐶𝑜𝑠(𝑁 𝐶𝑜𝑠−1x), for |x|≤1

= 𝐶𝑜𝑠ℎ(𝑁𝐶𝑜𝑠ℎ−1x), for |x| >1

**Code:-**

*//varad Patil 120A2036*

clc

clear all

n = 5

ftype = 'lp'

fdesign = 'butt'

frq = [0.3 0]

ripple =[0.03 0]

subplot(2,1,1)

Hz = iir(n,ftype,fdesign,frq,ripple)

[Hz,w] = frmag(Hz,256)

xlabel('Frequency')

ylabel('gain')

title('Butterworth Low Pass Filter')

plot(w,Hz,'r')

subplot(2,1,2)

fdesign = 'cheb1'

Hz = iir(n,ftype,fdesign,frq,ripple)

[Hz,w] = frmag(Hz,256)

plot(w,Hz,'b')

xlabel('Frequency')

ylabel('gain')

title('Chebshev low Pass Filter')

*//varad Patil 120A2036*

clc

clear all

n = 5

ftype = 'hp'

fdesign = 'butt'

frq = [0.3 0]

ripple =[0.03 0]

subplot(2,1,1)

Hz = iir(n,ftype,fdesign,frq,ripple)

[Hz,w] = frmag(Hz,256)

xlabel('Frequency')

ylabel('gain')

title('Butterworth High Pass Filter')

plot(w,Hz,'r')

subplot(2,1,2)

fdesign = 'cheb1'

Hz = iir(n,ftype,fdesign,frq,ripple)

[Hz,w] = frmag(Hz,256)

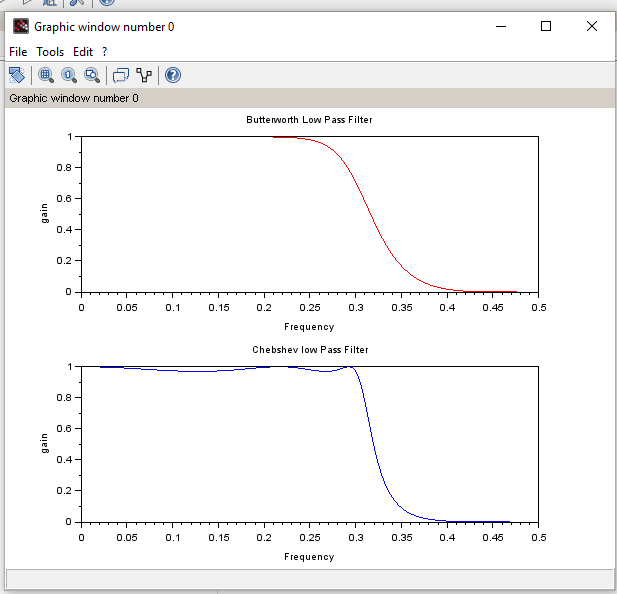
plot(w,Hz,'b')

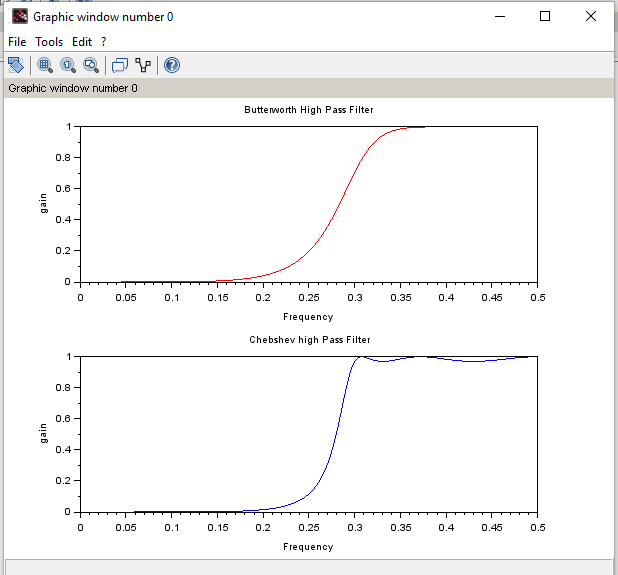
xlabel('Frequency')

ylabel('gain')

title('Chebshev high Pass Filter')

**Output:-**





**CONCLUSION:**

The concept of IIR filter design and designing of butter worth filter and chebyshev filter for given specifications was performed successfully.

## EXPERIMENT NO. 7

FIR Filter Design Using Window Techniques

## EXPERIMENT NO. 7

**OBJECTIVE:** To design FIR filter for converting discontinuities in frequency response into transition bands between values on either side of the discontinuity using window technique.

**SOFTWARE:** Scilab

**THEORY:**

In general, an FIR system is described by difference equation

𝑦(𝑛) = ∑𝑀𝑘=−01  𝑏𝑘𝑥(𝑛 − 𝑘)

Or equivalently, by the system function

𝐻(𝑧) = ∑𝑀𝑘=−01 𝑏𝑘 𝑧−𝑘

Furthermore, the unit sample response of the FIR system is identical to the coefficient {𝑏𝑘} , that is

𝐻(𝑛) = 𝑏𝑛 0 < 𝑛 < 𝑀 − 1

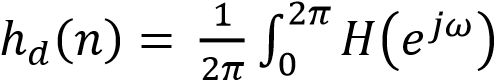
= 0 𝑜𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒

The length of the FIR filter is selected as **M** to conform to the established notation in the technical literature.

**WINDOW TECHNIQUES**:

The desired frequency response of any digital filter is periodic in frequency and can be expanded in a Fourier series, i.e.

𝐻𝑑 𝑒−𝑗𝜔𝑛

Where,  𝑒𝑗𝜔𝑛 𝑑𝜔

The major effects of windowing are that the discontinuities in the 𝐻(𝑒𝑗𝜔 ) are converted into transition bands between values on either sides of the discontinuity. The width of the transition bands between values on either sides of the main lobe of 𝜔(𝑒𝑗𝜔) a secondary effect of windowing is that the ripples from the side lobes of 𝜔 (𝑒𝑗𝜔)produces approximation errors for all 𝜔. Based on the above discussion, the desirable characteristic can be as below:

1. The Fourier Transform of the window function 𝜔(𝑒𝑗𝜔) should have a small width of main lobe containing as much of the total energy as possible.
2. The Fourier Transform of the window function [𝜔(𝑒𝑗𝜔)] should have side lobes that decrease in energy rapidly 𝜔 tends to 𝜋. Some of the most frequently used windows functions are described in the following sections.

**RECTANGULAR WINDOW FUNCTION:**

The weighting function for the rectangular window is given by,

1, 

𝑊𝑅(𝑛) = {0, 𝑜𝑡𝑒𝑟𝑤𝑖𝑠𝑒

The spectrum can be obtained by taking Fourier Transform of 𝑊𝑅(𝑛)

**HAMMING WINDOW FUNCTION:**

The Hamming window function is expressed by,

2𝜋𝑛

𝑊𝐻(𝑛) = {0.54 − 0.46 cos (𝑀 − 1) , 

0 , 𝑜𝑡𝑒𝑟𝑤𝑖𝑠𝑒

**HANNING WINDOW FUNCTION:**

The window function of a Hanning window is given by,

2𝜋𝑛

𝑊𝐻𝑎𝑛𝑛(𝑛) = {0.5 − 0.5 cos (𝑀 − 1) , 

0 , 𝑜𝑡𝑒𝑟𝑤𝑖𝑠𝑒

**BLACKMAN WINDOW FUNCTION:**

The window function of a Blackman window is expressed by,

2𝜋𝑛 4𝜋𝑛

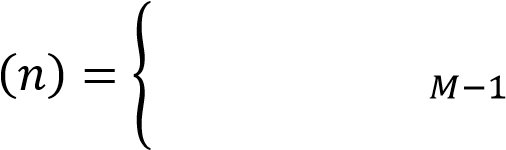
𝑊𝐵(𝑛) = {0.42 − 0.5 cos (𝑀 − 1) + 0.08 𝑐𝑜𝑠 (𝑀 − 1) , 

0 , 𝑜𝑡𝑒𝑟𝑤𝑖𝑠𝑒

**BARTLETT WINDOW FUNCTION:**

The window function of a non causal Bartlett window is expressed by,

|

𝑊𝑏𝑎𝑟𝑡 ,  𝑀 

 , 𝑜𝑡𝑒𝑟𝑤𝑖𝑠𝑒

**KAISER WINDOW:**

The Kaiser Window function of length M+1 is given by,

𝐼𝑜



𝜔𝑘(𝑛) = {𝐼𝑜

0

Where 𝛼 an independent variable determined by Kaiser and the parameter 𝛽 is expressed by,

2𝑛 2 0.5

𝛽 = 𝛼 [1 − ( ) ] 𝑀 − 1

**ADVANTAGES AND DISADVANTAGES OF THE WINDOW TECHNIQUE:**

* An important advantage of the window method is its simplicity: it is simple to apply and simple to understand. It involves a minimum amount of computational effort, even for the more complicated Kaiser window.
* The major disadvantage is its lacks flexibility. Both the peak pass band and stop band ripples are approximately equal, so that designer may end up with either too small a pass band ripple or too large a stop band attenuation.
* In some applications, the expression for the desired filter response, 𝐻𝐷(𝜔) will be too complicated for ℎ𝐷(𝑛) may be obtained via the frequency sampling method before the window function is applied.
* Because of the effect of convolution of the spectrum of the window function and desired response, pass band and stop band edge frequencies cannot be precisely specified.

**Code:-**

*//varad patil 120A2036*

clc;

clear all;

close;

wc = input('Enter the cutoff frequency:- ');

N = input('Enter the length:- ');

a = (N-1)/2;

for n = 0:N-1

    if n == a then

        hd(n+1) = wc/%pi;

    else

        hd(n+1) = (wc/%pi)\*((sin(wc\*(n-a)))/(wc\*(n-a)));

    end;

end;

for n = 0:N-1

    Wr(n+1) = 1;  *// rectangular window*

    Whamm(n+1) = 0.54 - (0.46\*cos(2\*%pi\*n/N-1)); *// hamming window*

    Whann(n+1) = 0.5 - (0.5\*cos(2\*%pi\*n/N-1)); *// hanning window*

end;

for n = 1:N

    h1(n) = hd(n)\*Wr(n);

    h2(n) = hd(n)\*Whamm(n);

    h3(n) = hd(n)\*Whann(n);

end;

[H1, w] = frmag(h1,256)

[H2, w] = frmag(h2,256);

[H3, w] = frmag(h3,256);

subplot(3,1,1)

plot(w,H1)

title('Rectangular window')

subplot(3,1,2)

plot(w,H2)

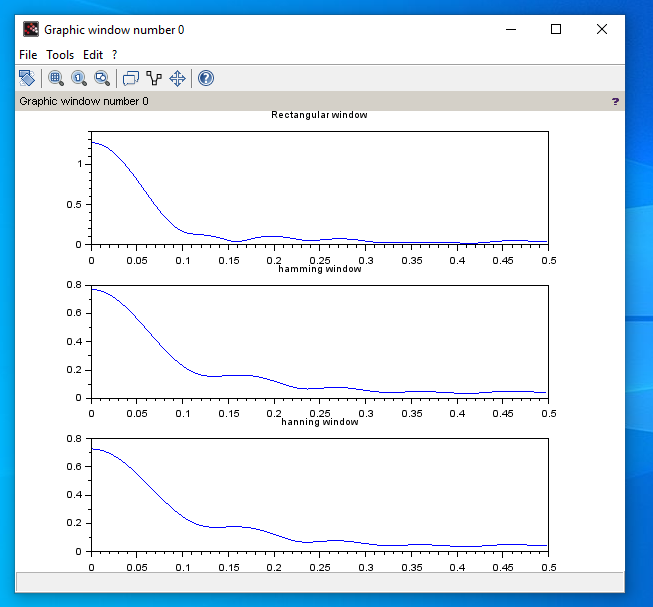
title('hamming window')

subplot(3,1,3)

plot(w,H3)

title('hanning window')

**Output:-**

****

**CONCLUSION:**

Designing of FIR filter for converting discontinuities in frequency response into transition bands between values on either side of the discontinuity using window technique was performed successfully. We understood the concepts of windowing technique and its different types.

## EXPERIMENT NO. 8

### Effect of coefficient quantization using truncation and rounding

### **EXPERIMENT NO. 8**

**OBJECTIVE:** To study the effect of coefficient quantization using truncation and rounding

**SOFTWARE:** Scilab

**THEORY:**

There are many digital filter structures for both FIR and IIR systems, including Direct Form I/II, transposed Direct Form I/II, Cascade Form, and Parallel Form. In addition, there are many algorithms commonly used to generate the numerator and denominator coefficients, given a design specification. For IIR filter designs, elliptic filters are commonplace, as are Butterworth and Chebyshev designs. Such algorithms can be implemented in scilab, where the set of coefficients is computed in double-precision, which we will consider to be essentially infinite precision. Unfortunately, many real-world implementations of digital filters require hardware that does not support the use of full double-precision floating point. For applications on custom hardware, a double-precision floating-point unit is costly in terms of design-time, chip space, and power consumption. Alternatives include going to fixed-point implementations or to floating-point implementations with smaller mantissa and/or exponent sizes. By dropping the precision in the implemented digital filter, the double-precision designed coefficients will need to be quantized. This coefficient quantization fundamentally changes the filter response, and care will need to be taken to ensure that the filter’s performance still adequately meets its design criteria. Much analysis has been done on the effects of coefficient quantization, including the effect on frequency response and the effects on the system output. Here we are studying the effect of quantization on the frequency response of digital filters.

**Code:-**

*//varad patil*

*//120A2036*

clear;

clc

z=poly(0,'z')

function **beq**=truncround(**b**, **n**, **flag**)

l=0;

d=abs(**b**);

while fix(d)>0

l=l+1

d=d/2

*//d=abs(b)/(2^l);*

end

if **flag**==1

**beq**=fix(abs(**b**)\*2^(**n**-l));

end

if **flag**==2

**beq**=fix(abs(**b**)\*2^(**n**-l)+0.5);

end

**beq**=sign(**b**).\***beq**.\*2^-(**n**-l);

endfunction

[Hz1]=iir(3,'lp','cheb1',[0.3 0],[0.08 0.02])

disp("Transfer Function: ", Hz1)

b = coeff(Hz1.num)

a = coeff(Hz1.den)

disp("Numerator coefficients: ", b)

disp("Numerator coefficients: ", a)

[Hw fr]=frmag(Hz1,100)

disp(' ')

flag=input('enter 1 for truncation, 2 for rounding=');

bq=truncround(b,6,flag);

aq=truncround(a,6,flag);

disp("Numerator coefficients: ", bq)

disp("Numerator coefficients: ", aq)

H1=0

H2=0

for i = 1:length(bq)

H1 = H1 + bq(i)\*(z^(i+1))

end

for i = 1:length(aq)

H2 = H2 + aq(i)\*(z^(i+1))

end

Hz2 = H1./H2

[Hwq fr]=frmag(Hz2,100)

figure(1)

plot(fr,20\*log10(abs(Hw)));

plot(fr,20\*log10(abs(Hwq)),'--');

xlabel('frequency');

ylabel('Gain,dB');

if flag==1

legend('without quantization','with truncation');

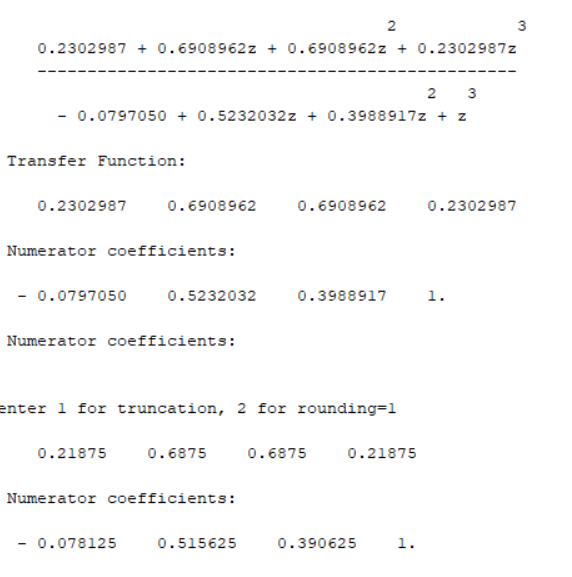
end

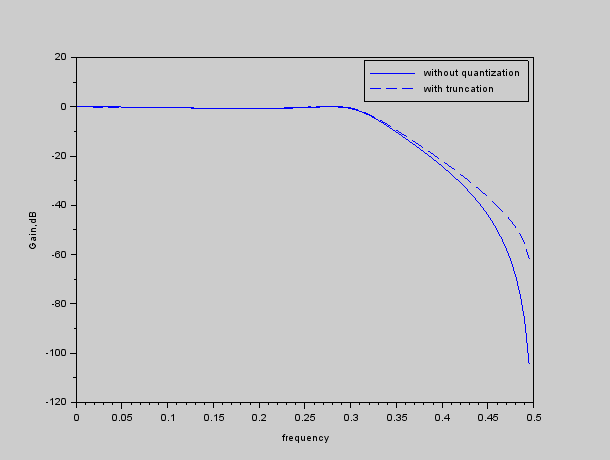
if flag==2

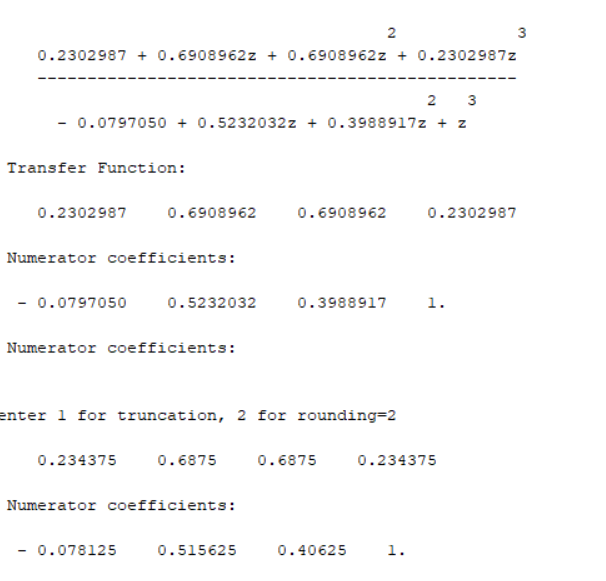
legend('without quantization','with rounding');

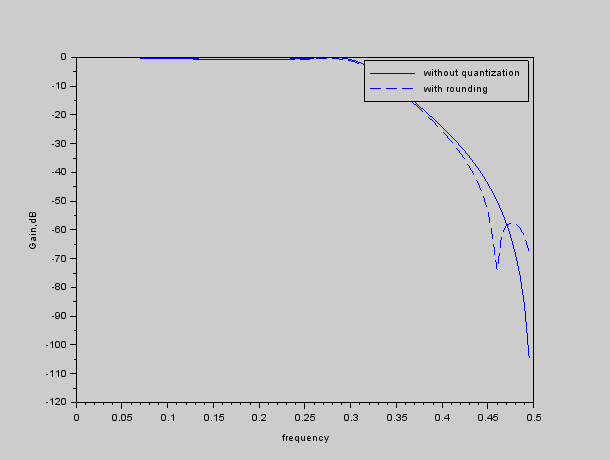
end

**Output:-**

****

****

****

****

**CONCLUSION:**

The effect of coefficient quantization using truncation and rounding was performed successfully.

And the concept of rounding and truncation was understand.