

Lab 1: Generation of Basic Signals

Aim: To generate and obtain the output for the basic signals

Software Requirement: SCI Lab

Theory: Signals can be classified as continuous or discrete time. In the mathematical abstraction, the domain of a continuous-time signal is the set of real numbers (or some interval thereof), whereas the domain of a discrete-time (DT) signal is the set of integers (or other subset of real numbers). What these integers represent depends on the nature of the signal; most often it is time.

A continuous-time signal is any function which is defined at every time t in an interval, most commonly an infinite interval. A simple source for a discrete-time signal is the sampling of a continuous signal, approximating the signal by a sequence of its values at particular time instants. A signal, of which a sinusoid is only one example, is a sequence of numbers. A continuous-time signal is an infinite and uncountable set of numbers, as are the possible values each number can have between a start and end time, there are infinite possible values for time and instantaneous amplitude.

1. a) Generation of Continuous Signals

Scilab code: Sine wave

```
1 clc ;  
2 clf ;  
3 clear all;  
4 // Caption: Generation of sine wave  
5 f =0.2;  
6 t =0:0.1:10;  
7 x = sin (2* %pi * t * f ) ;  
8 plot (t ,x ) ;  
9 title ( ' s i n e w a v e ' ) ;  
10 xlabel ( ' t ' ) ;  
11 ylabel ( ' x ' ) ;
```

Simulation Output:

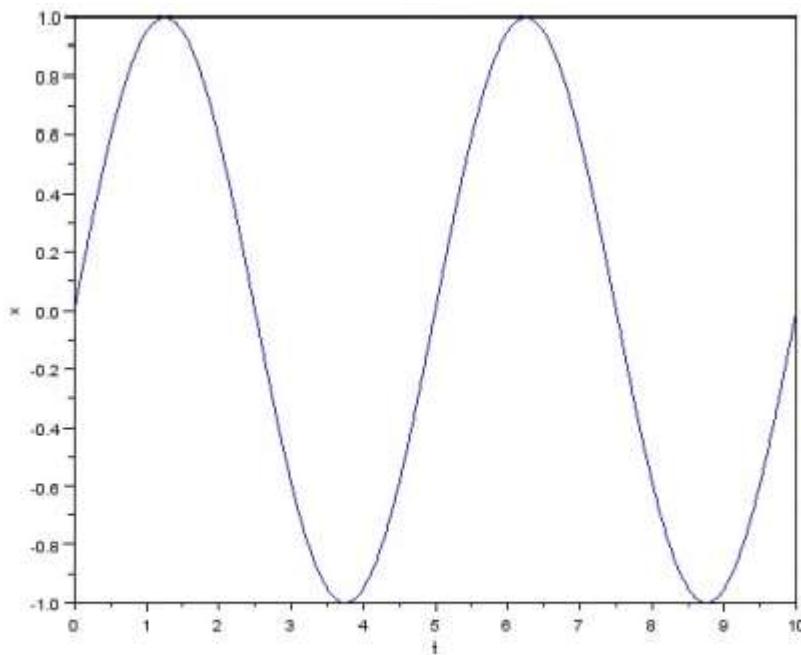


Figure 1.1: sinewave

Scilab code: Cosine wave

```
1 clc ;  
2 clf ;  
3 clear all;  
4 // Caption: Generation of cosine wave  
5 f =0.2;  
6 t =0:0.1:10;  
7 x = cos (2* %pi * t * f ) ;  
8 plot (t ,x ) ;  
9 title ( ' c o s i n e w a v e ' ) ;  
10 xlabel ( ' t ' ) ;  
11 ylabel ( ' x ' ) ;
```

Simulation Output:

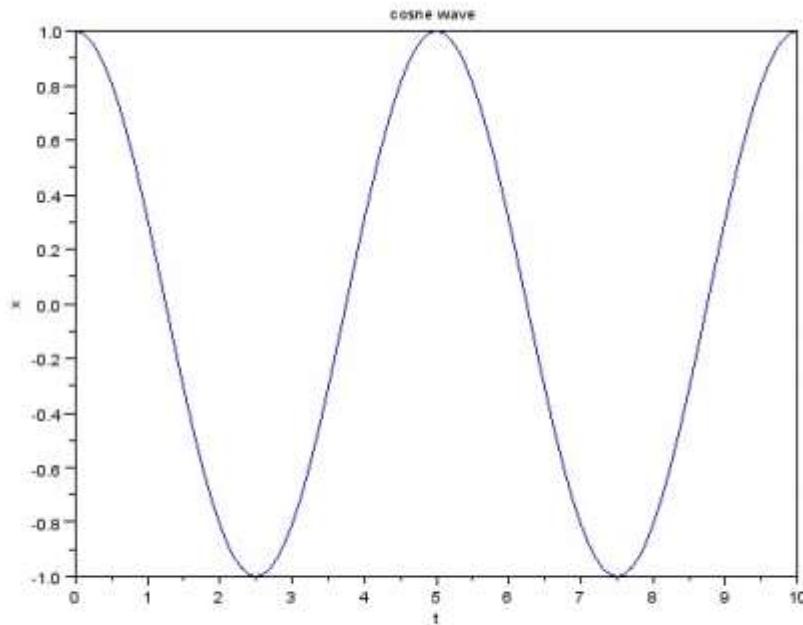


Figure 1.2: cosine wave

Scilab code: Triangular wave

```
1 clc ;
2 clf ;
3 clear all;
4 // Caption: Generation of Triangular wave
5 a =8;
6 t =0:( %pi /4 ):(4* %pi ) ;
7 y = a *sin (2* t ) ;
8 a = gca () ;
9 a . x_location =" mi d dl e "
10 plot (t ,y ) ;
11 title ( ' t r i a n g u l a r w a v e ' );
```

```
12 xlabel( 't' );
13 ylabel( 'y' );
```

Simulation Output:

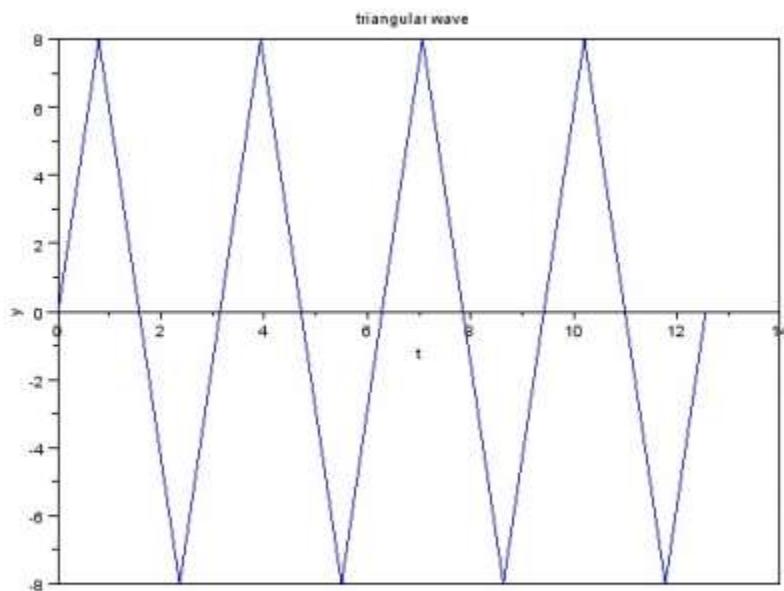


Figure 1.3: triangular wave

Scilab code: Exponential wave

```
1 clc ;
2 clf ;
3 clear all;
4 // Caption: Generation of Exponential wave
5 t = -2:0.1:2;
6 x = exp (t );
7 plot (t ,x );
8 title ( ' e x p o n e n t i a l w a v e ' );
9 xlabel ( ' t ' );
10 ylabel ( ' x ' );
```

Simulation Output:

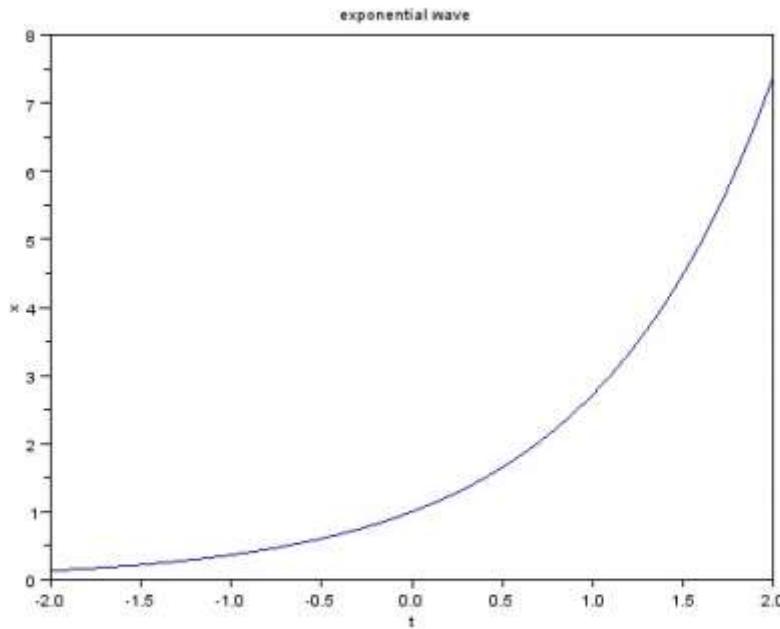


Figure 1.4 : Exponential wave

1. b) Generation of Discrete Signals

Scilab code: Unit impulse signal

```
1 clc ;
2 clf ;
3 clear all;
4 // u n i t l m p u l s e
5 L =5;
6 n = - L : L;
7 x =[ zeros (1 , L ) ,ones (1 ,1 ),zeros (1 , L )];
8 a = gca ();
9 a . y_location =" mi d dl e "
10 plot2d3 (n ,x );
11 title ( ' u n i t l m p u l s e ' );
```

Simulation Output:

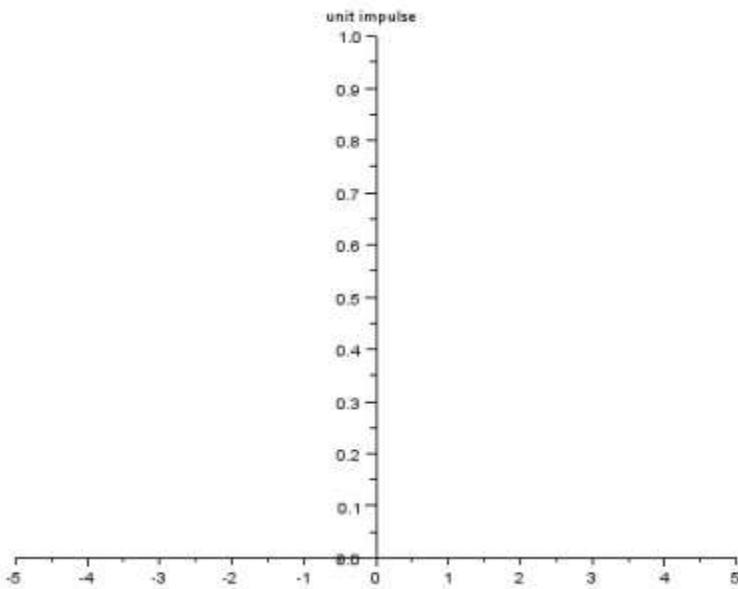


Figure 1.5: unit impulse signal

Scilab code: Unit step signal

```
1 clc ;
2 clf ;
3 clear all;
4 L =5;
5 n = - L : L;
6 x =[ zeros (1 , L ) ,ones (1 , L +1) ];
7 a = gca () ;
8 a . y_location =" mi d dl e ";
9 plot2d3 (n ,x ) ;
10 title ( ' u n i t s t e p ' );
11 xlabel ( ' n ' );
12 ylabel ( ' x ' );
```

Simulation Output:

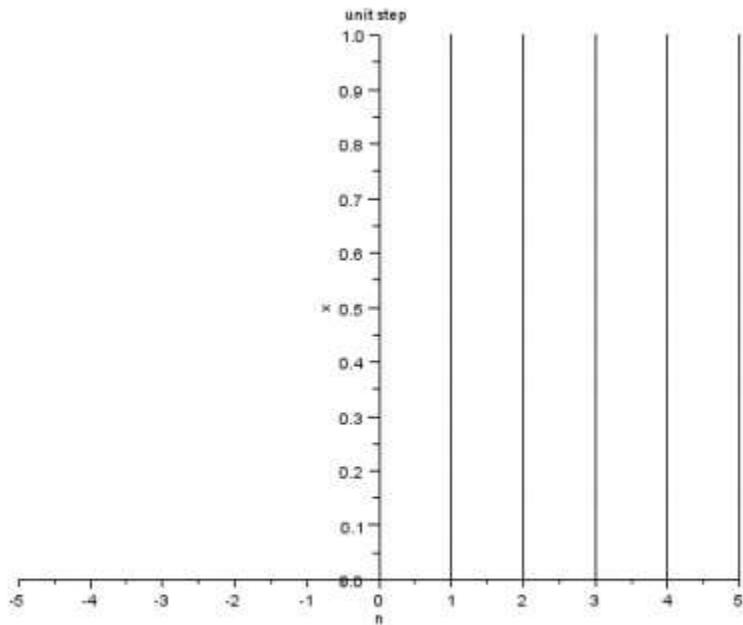


Figure 1.6 : unitstepsignal

Scilab code: Unit ramp signal

```
1 // u n i t r a m p  
2 clc ;  
3 clf ;  
4 clear all;  
5 L =5;  
6 n = - L : L;  
7 x =[ zeros (1 , L ) ,0: L ];  
8 a = gca () ;  
9 a . y_location = ' mi d dl e ' ;  
10 plot2d3 (n ,x ) ;  
11 xtitle ( ' u n i t r a m p s i g n a l ' ) ;  
12 xlabel ( '--->n ' ) ;  
13 ylabel ( '--->x ( n ) ' );
```

Simulation Output:

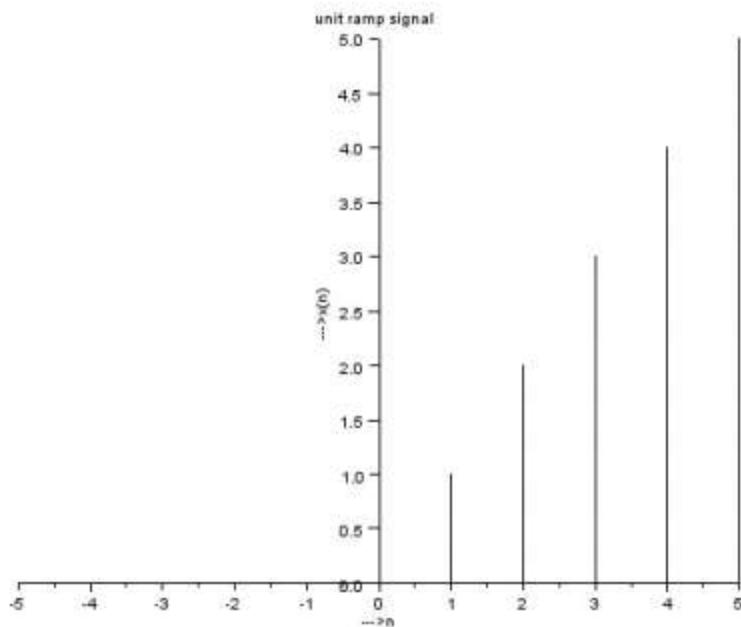


Figure 1.7 : unit ramp

Pre-lab questions:

1. What is continuous signal and discrete signal?
2. What are the properties of a signal?
3. How is a signal generated?
4. What is the difference between analog and digital signals?
5. Which signal is more reliable analog or digital?

Post-Lab questions:

1. Derive the code and show the output for signum function.
2. Derive the code and show the output for sinc function.
3. Derive the code and show the output for discrete exponential wave.

Result:

Experiment 2:

Experiment 2: Generation of Continuous Time and Discrete Time Signal

Aim: To generate the continuous time and discrete time signal using SCI lab

Software Requirement: SCI Lab

Theory:

Signals are represented mathematically as functions of one or more independent variables. Here we focus attention on signals involving a single independent variable. For convenience, this will generally refer to the independent variable as time. There are two types of signals: continuous-time signals and discrete-time signals. Continuous-time signal: the variable of time is continuous. A speech signal as a function of time is a continuous-time signal. Discrete-time signal: the variable of time is discrete.

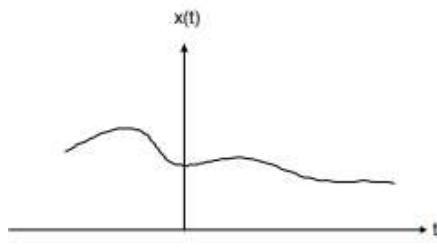


Fig.1. CT signal

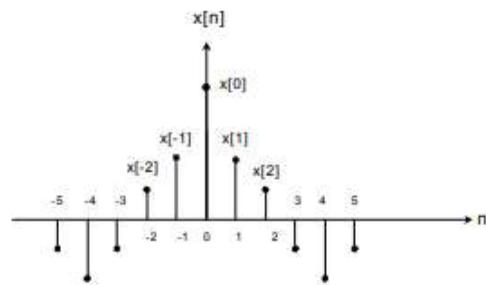


Fig.2 DT signal

To distinguish between continuous-time and discrete-time signals we use symbol t to denote the continuous variable and n to denote the discrete-time variable. And for continuous-time signals we will enclose the independent variable in parentheses (\cdot) , for discrete-time signals we will enclose the independent variable in bracket $[\cdot]$.

A discrete-time signal $x[n]$ may represent a phenomenon for which the independent variable is inherently discrete. A discrete-time signal $x[n]$ may represent successive samples of an underlying phenomenon for which the independent variable is continuous. For example, the processing of speech on a digital computer requires the use of a discrete time sequence representing the values of the continuous-time speech signal at discrete points of time.

The differences between continuous and discrete-time signals are as follows:

Table 1

S.No.	Continuous-time signal	Discrete-time signal
1	The continuous-time signal is an analog representation of a natural signal.	The discrete-time signal is a digital representation of a continuous-time signal.
2	The continuous-time signal can be converted into discrete-time signal by the Euler's method.	The discrete -time signal can be converted into continuous-time signal by the methods of zero-order hold or first-order hold.
3	The conversion of continuous to discrete-time signal is comparatively easy than the conversion of discrete to continuous-time signals.	The conversion of discrete to continuous-time signals is very complicated and it is done through a sample and hold process.
4	It is defined over a finite or infinite domain of sequence.	It is defined over a finite domain of sequence.
5	The value of the signal can be obtained at any arbitrary point of time.	The value of the signal can be obtained only at sampling instants of time.
6	The continuous-time signals are not used for the processing of digital signals.	The discrete-time signals are used for the processing of digital signals.
7	The continuous-time variable is denoted by a letter t .	The discrete-time variable is denoted by a letter n .
8	The independent variable encloses in the parenthesis (\bullet).	The independent variable encloses in the bracket [\bullet].

The differences between discrete and digital signals are as follows:

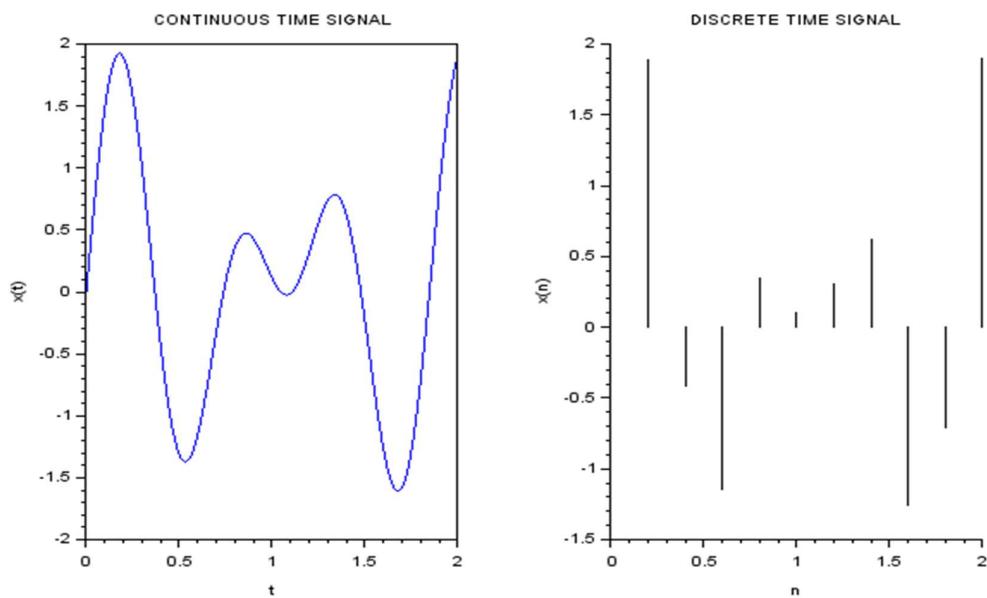
Table 2

S.No.	Discrete-time signal	Digital signal
1	The discrete-time signal is a digital representation of a continuous-time signal.	The digital signal is a form of discrete-time signal.
2	The discrete -time signal can be obtained from the continuous-time signal by the Euler's method.	The digital signal can be obtained by the process of sampling, quantization, and encoding of the discrete-time signal.
3	The discrete-time signal is a signal that has discrete in time and discrete in amplitude.	The digital signal is a signal that has discrete in amplitude and continuous in time.
4	The value of the signal can be obtained only at sampling instants of time.	The amplitude of the digital signal is either 1 or 0. That is, either OFF or ON.
5	The signals are sampled but not necessary to quantized in the discrete-time signals.	The signals are sampled and quantized in the digital signals.
6	All the discrete-time signals are digital signals.	All the digital signals are not discrete-time signals.

PROGRAM

```
// GENERATION OF CONTINUOUS TIME SIGNAL AND DISCRETE TIME SIGNAL
clear ;
clc ;
close ;
t =0:0.01:2;
x1 =sin (7* t ) +sin (10* t ) ;
subplot (1 ,2 ,1) ;
plot (t ,x1 ) ;
xlabel ( 't' ) ;
ylabel ( 'x(t)' ) ;
title ( 'CONTINUOUS TIME SIGNAL') ;
n =0:0.2:2;
x2 =sin (7* n ) +sin (10* n ) ;
subplot (1 ,2 ,2) ;
plot2d3 (n ,x2 ) ;
xlabel ( 'n') ;
ylabel ( 'x(n)' ) ;
title ( 'DISCRETE TIME SIGNAL') ;
```

SIMULATION RESULT



Pre-lab questions:

1. Define continuous time signal
2. How is discrete time signal generated?
3. Draw the graphical representation of continuous time signal and discrete time signal

Post-Lab questions:

1. Write any three difference between analog signal and digital signal
2. Write the code for generation of discrete signal for time interval $5\mu s$ and $7\mu s$ using subplot function
3. Give the advantages of digital signal over analog signal

Result:

EXPERIMENT 3:

EXPERIMENT 3a: SAMPLING THEOREM

Aim: To generate the sampled signal from the analog signal using SCI lab

Software Requirement: SCI Lab

Theory:

The real life signals that we encounter in our day to day basis are mostly analog signals. These signals are defined continuously in time and have infinite range of amplitude values. In order to process these signals to obtain meaningful information, they need to be converted to a format which is easily handled by computing resources like microprocessors, computers etc... The first step in this process is to convert the real-time signal into discrete-time signals. Discrete-time signals are defined only at a particular set of time instances. They can thus be represented as sequence of numbers with continuous range of values.

The process of converting an analog signal (denoted as $x(t)$) to a digital signal (denoted as $x(n)$) is called the analog-to-digital conversion (referred to as digitization), usually performed by an analog-to-digital converter (ADC). Here t is the continuous time variable and n is the sequence order. In many applications after the processing of the digital signal is performed, $x(n)$ needs to be converted back to analog signal $x(t)$ before it is applied to appropriate analog device. This reverse process is called digital-to-analog conversion and is typically performed using a digital-to-analog converter (DAC).

The typical block diagram of an ADC is shown in Fig. 1 below.

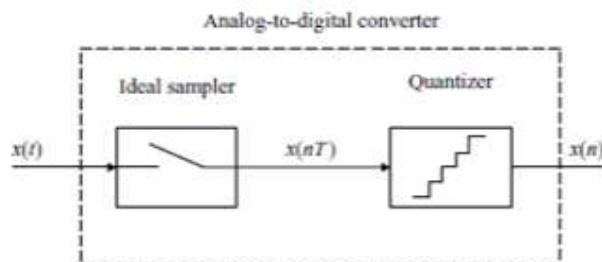


Fig:-1 Block diagram of an ADC

The process of digitization consists of first sampling (digitization in time) and quantization (digitization in amplitude). In this experiment we will study and understand the principle of sampling, while the principle of quantization will be studied in the next experiment. The sampling process depicts an analog signal as a sequence of values. The basic sampling function can be carried out with an ideal 'sample-and-hold' circuit which maintains the

sampled signal until next sample is taken. An ideal sampler can be considered as a switch that periodically opens and closes every T seconds. The sampling frequency (f_s in Hertz) is thus defined as

$$f_s = \frac{1}{T} \dots (1)$$

The sampled discrete time signal $x(nT)$, $n=0,1,2,\dots$ of the original continuous time signal $x(t)$ is shown in Fig. 2 below.

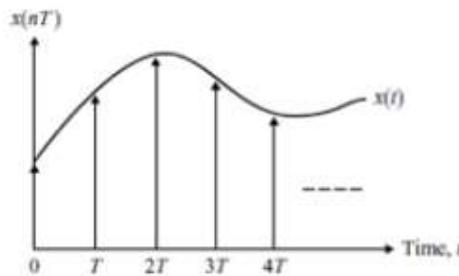


Fig:-2 Digitization of analog signal $x(t)$ into discrete-time signal $x(nT)$

In order to represent an analog signal $x(t)$ by a discrete-time signal $x(nT)$ accurately, so that the analog signal can be exactly reconstructed back from the discrete-time signal, the sampling frequency f_s must be at least twice the maximum frequency component (f_M) of the original analog signal. Thus we have,

$$f_s \geq 2f_m \dots (2)$$

The minimum sampling rate is called the Nyquist rate and the above Sampling Theorem is called the Shannon's Sampling Theorem. When an analog signal is sampled at f_s , frequency components higher than $f_s/2$ fold back into the frequency range $[0, f_s/2]$. This folded frequency components overlap with the original frequency components in the same range and leads to an undesired effect known as aliasing. In this case, the original analog signal cannot be recovered from the sample data.

Consider an analog signal of frequency 1Hz as shown in Fig. 3(a) below. The sampling frequency is 4Hz. The sampled signal is shown in Fig. 3(b), Note that an exact reconstruction of the missing samples is obtained so long as the Shannon's Sampling Theorem is satisfied.

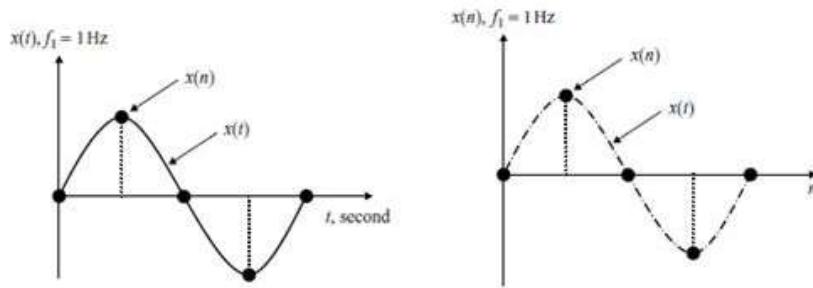


Fig:-3 Sampling of an analog signal $x(t)$ into discrete-time signal $x(nT)$ (a) and its exact reconstruction (b)

Now let's consider, the analog signal of frequency 5Hz as shown in Fig. 4(a) below. The sampling frequency is same as above, i.e. 4Hz. The sampled signal is shown in Fig. 4(b), Note that the reconstruction of the original analog signal is not possible since the sampling frequency does not satisfy Shannon's Sampling Theorem. In this case the reconstructed signal has a frequency of 1Hz. The signal of 5Hz is folded back as 1Hz, into the range determined by the sampling frequency leading to the problem of aliasing.

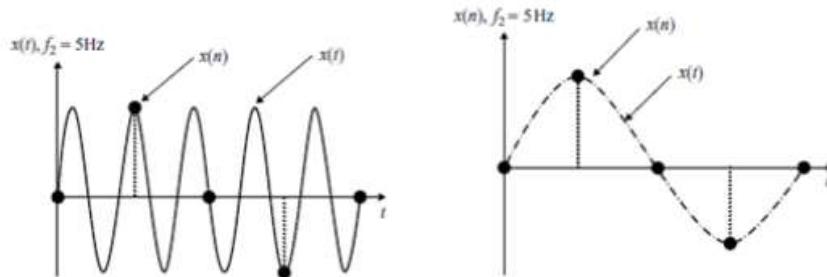


Fig:-4 Sampling of an analog signal $x(t)$ into discrete-time signal $x(nT)$ (a) and its inaccurate reconstruction (b)

PROGRAM

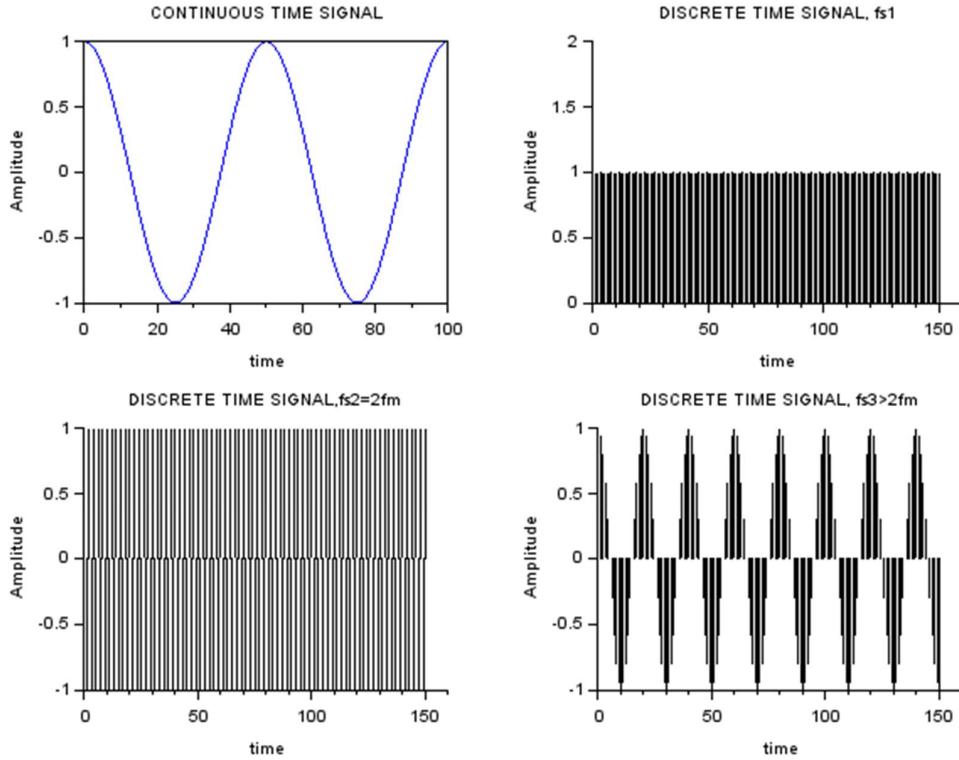
```
// PROGRAM FOR SAMPLING
// CONVERSION OF ANALOG SIGNAL TO DISCRETE SIGNAL
clc ;
clf ;
clear all;
t =0:0.01:100;
fm =0.02;
x = cos (2* %pi * fm * t );
subplot (2 ,2 ,1) ;
plot (t ,x ) ;
title ( 'CONTINUOUS TIME SIGNAL' ) ;
xlabel('time');
```

```

ylabel('Amplitude');
fs1 =0.002;
n =0:1:150;
x1 =cos (2* %pi * fm * n / fs1 ) ;
subplot (2 ,2 ,2) ;
plot2d3 (n , x1 ) ;
title ('DISCRETE TIME SIGNAL, fs1');
xlabel('time')
ylabel('Amplitude')
fs2 =0.04;
x2 =cos (2* %pi * fm * n / fs2 ) ;
subplot (2 ,2 ,3) ;
plot2d3 (n , x2 ) ;
title ('DISCRETE TIME SIGNAL,fs2=2fm' );
xlabel('time')
ylabel('Amplitude')
fs3 =0.4;
x3 =cos (2* %pi * fm * n / fs3 ) ;
subplot (2 ,2 ,4) ;
plot2d3 (n , x3 ) ;
title ('DISCRETE TIME SIGNAL, fs3>2fm' );
xlabel('time')
ylabel('Amplitude')

```

SIMULATION RESULTS



EXPERIMENT 3:

EXPERIMENT 3b: ALIASING AND ITS EFFECTS

Aim: To analyze the effects of aliasing frequencies for the various sampling frequencies of the discrete signal using SCI lab

Software Requirement: SCI Lab

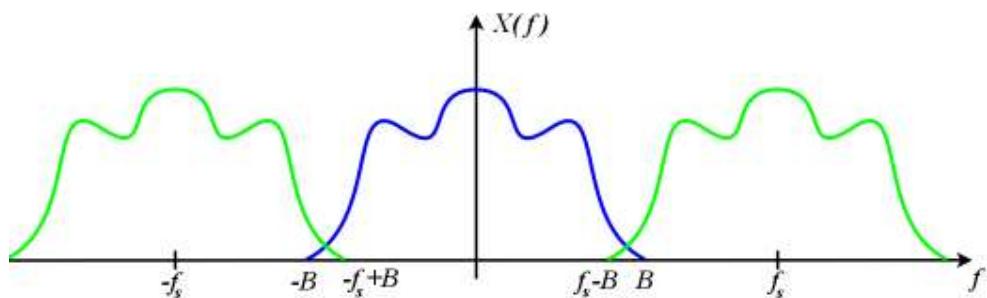
Theory:

Aliasing – from alias – is an effect that makes different signals indistinguishable when sampled. It also refers to the difference between a signal reconstructed from samples and the original continuous signal, when the resolution is too low. Basically, aliasing depends on the sampling rate and frequency content of the signal.

Sampling Rate and Nyquist Frequency Limit

- For a given highest frequency B , we get the lower bound on the sampling frequency : $2B$ or Nyquist rate. For instance : for a signal whose maximum frequency is 16 KHz, we need a 32 KHz sampling rate.
- For a given sampling rate, we get the upper bound for frequency components : $B < f_s/2$, or Nyquist frequency or F_{max} . For instance : for a signal whose sampling rate is 48 KHz, we can sample signals up to 24 KHz.

In practice, a signal can never be perfectly bandlimited. Even if an ideal reconstruction could be made, the reconstructed signal would not be exactly the original signal. The error that corresponds to the failure of bandlimitation is referred to as aliasing.



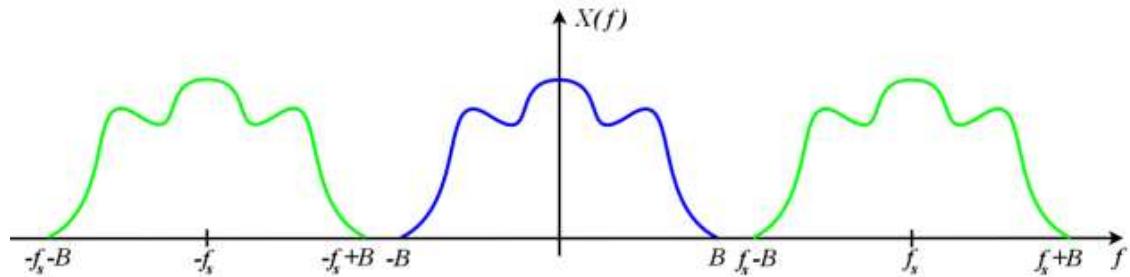
The blue sampled signal is insufficiently bandlimited. The overlapping edges of the green images are added and creating a spectrum

When a signal is sampled, its contents is reduced from real numbers to integer numbers. Values can be rounded to a superior or inferior value.

If a signal is sampled with a 32 KHz sampling rate, any frequency components above 16 KHz – Nyquist frequency, create an aliasing.

Any frequency component above $f_s/2$ is indistinguishable from a lower-frequency component, called an alias, associated with one of the copies.

The Fourier transform of the signal creates a symmetrical image. The energy above the Nyquist frequency is transferred below this frequency.



The blue signal is bandlimited and properly sampled. The images do not overlap.

PROGRAM 1:

```
// ALIASING AND ITS EFFECTS
function [F]=aliasfrequency(f, s, s1)
if (s>2*f) then
    disp ('Aliasing not occurred')
else
    disp ('Aliasing occurred')
end
F = f / s ;
for i =1:100
if ( abs ( F ) >0.5)
    F=F - i ;
end
end
fa = F * s1 ;
disp ( fa , 'Frequency of Reconstructed Signal is ')
endfunction

f=input('Enter the frequency ');
s=240; // sampling frequency
s1=s;
aliasfrequency (f, s )
s=140; // sampling frequency
s1=s;
aliasfrequency (f,s , s1 )
s=90; // sampling frequency
s1=s;
```

```
aliasfrequency(f,s ,s1 )
s =35;// sampling frequency
s1 = s;
aliasfrequency(f,s ,s1 )
```

SIMULATION RESULT

Enter the frequency: 100

"Aliasing not occurred"

"Frequency of Reconstructed Signal is 100."

"Aliasing occurred"

"Frequency of Reconstructed Signal is -40.
"

"Aliasing occurred"

"Frequency of Reconstructed Signal is 10.000000
"

"Aliasing occurred"

"Frequency of Reconstructed Signal is -5.000000"

Or

PROGRAM 2 :

```
//-----
//Aliasing - explore several different undersampling sins.
//Franz Hover MIT Mechanical Engineering
clear;
clf;
tfinal = .002 ;
fm = 20000*2*%pi;

fs = 7500*2*%pi; // change the sampling rate here [5500 6500 7500]

dtfi = 2*%pi/fm/20 ; //fine time scale

tfi = 0:dtfi:tfinal ;

dt = 2*%pi/fs ; //sampling time scale

t = 0:dt:tfinal ;

xfi = .9*sin(fm*tfi) ; //here are the signals

x = .9*sin(fm*t) ;

figure(1);clf;
//hold off;

//subplot(3,1,1);

plot(tfi,xfi);

//hold on;
//set(gca(),"auto_clear","on")
//mtlb_hold

plot(t,x,'ro-','LineWidth',2);
xlabel('time,seconds');
ylabel('Amplitude')

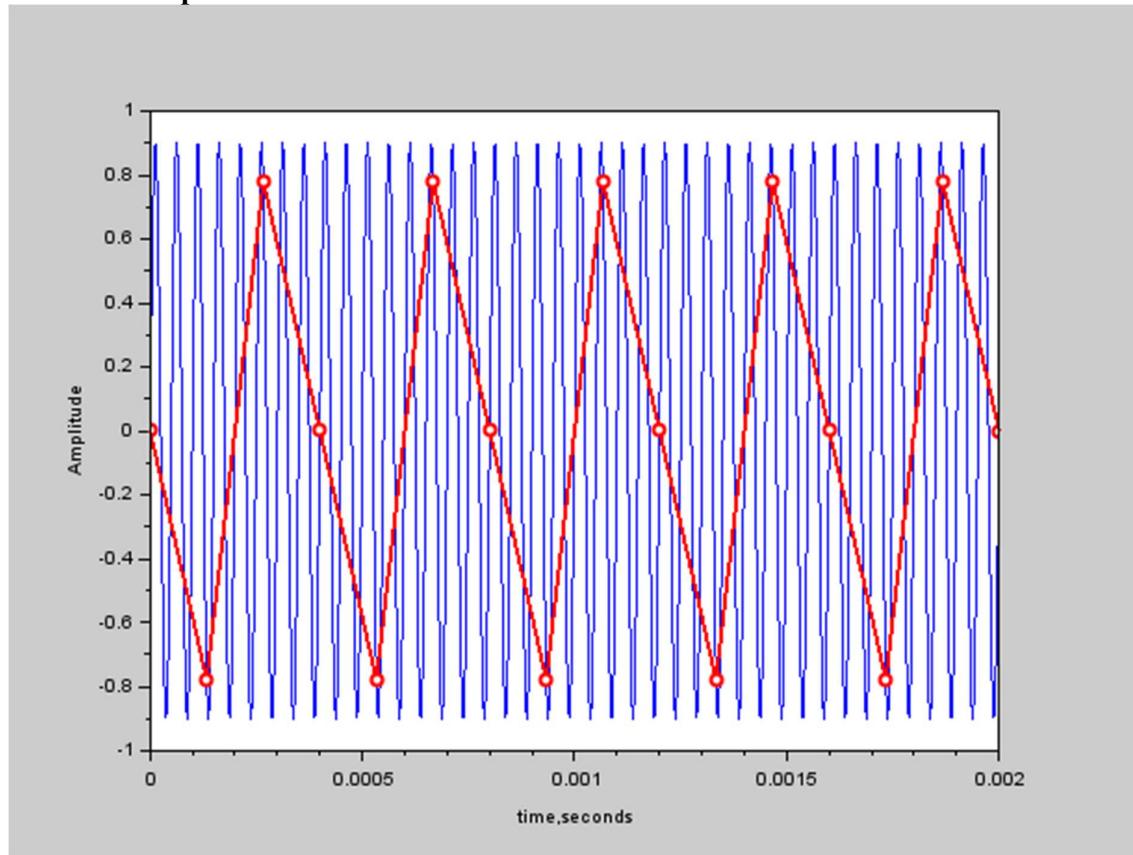
disp('Save the figure if necessary');
nfi = length(xfi);
fxfi = fft(xfi)/nfi;
wfi = [0:nfi-1]/(nfi-1)*2*%pi/dtfi ;
n = length(x);
fx = fft(x)/n ;
w = [0:n-1]/(n-1)*2*%pi/dt;
figure(2);clf;
//hold off;
plot(w/2/%pi,abs(fx),wfi/2/%pi,abs(fxfi));
//hold ;
plot(modulo(fm,fs)/2/%pi,0,'r*','LineWidth',2);
```

```
plot(modulo(-fm,fs)/2/%pi,0,'r*','LineWidth',2);
disp(sprintf('Sampling freq: %g Hz', fs/2/%pi));
disp(sprintf('Aliased frequencies: %g %g Hz.', ...
modulo(fm,fs)/2/%pi,modulo(-fm,fs)/2/%pi));
disp('Save figure if you want');
```

SIMULATION RESULT

"Sampling freq: 7500 Hz"

"Aliased frequencies: 5000 -5000 Hz."



Pre-lab questions:

1. Define Sampling rate
2. Why is signal to be sampled?
3. Define : sampling theorem
4. What is aliasing? When is aliasing occurred?
5. How to avoid aliasing?

Post-Lab questions:

1. If signal $x(t) = 5\sin(800\pi t)$ is sampled what is the minimum sampling rate required to avoid aliasing ? Determine the discrete time signal after sampled
2. Define Alias frequency with an example
3. What will be the graphical representation of discrete time signal of $x(t)$ for the following cases a) $f_s=f_m$ b) $f_s=2f_m$ c) $f_s < f_m$ d) $f_s > 2f_m$

Result:

EXPERIMENT 4

Linear Convolution and Circular Convolution

Aim: To obtain linear and circular convolution of two input sequence using scilab

A linear system has the property that the response to a linear combination of inputs is the same linear combination of the individual responses. The property of time invariance states that, in effect, the system is not sensitive to the time origin. More specifically, if the input is shifted in time by some amount, then the output is simply shifted by the same amount. The importance of linearity derives from the basic notion that for a linear system if the system inputs can be decomposed as a linear combination of some basic inputs and the system response is known for each of the basic inputs, then the response can be constructed as the same linear combination of the responses to each of the basic inputs. Signals (or functions) can be decomposed as a linear combination of basic signals in a wide variety of ways. For systems that are both linear and time-invariant, there are two particularly useful choices for these basic signals: delayed impulses and complex exponentials. The representation of both continuous time and discrete-time signals as a linear combination of delayed impulses and the consequences for representing linear, time-invariant systems. The resulting representation is referred to as convolution.

```
// Program for LINEAR CONVOLUTION
clc;
clf;
clear all;
x = input("Enter the first sequence: ");
h = input("Enter the second sequence: ");
disp(conv(x, h), "Convolution = ");
y = conv(x, h);
```

OUTPUT:

Enter the first sequence: [1 2 3 4 5 6 7 8]

Enter the second sequence: [5 4 5 2 1]

"Convolution = "

5. 14. 28. 44. 61. 78. 95. 112. 84. 60. 23. 8.

```

// program for circular convolution using concentric circle method

clc ;
clf ;
clear all;
g=input("enter the first sequence");
h=input("enter the second sequence");
N1=length (g);
N2=length(h);
N=max(N1,N2) ;
N3=N1-N2;
if(N3>=0)then
h =[h,zeros(1,N3)];
else
g =[g,zeros(1,- N3)];
end
for n=1:N
y(n)=0;
for i=1:N
j=n - i+1;
if(j<=0)
j= N + j;
end
y(n)=y(n)+g(i)*h(j);
end
end
disp(' sequence y =');
disp(y);
plot2d3(y);

Output
enter the first sequence[2 1 2 1]
enter the second sequence[ 1 2 3 4]

```

" sequence y ="

14. 16. 14. 16.

// Program for Circular convolution using DFT computation

clc ;

close ;

x1=[2 ,1 ,2 ,1];

x2=[1 ,2 ,3 ,4];

//DFT Computation

X1 =fft(x1,-1);

X2=fft(x2,-1);

X3=X1 .* X2 ;

//IDFT Computation

x3 =fft(X3 ,1);

// Display sequence [n] in command window

disp(x3);

Output

14. 16. 14. 16.

Pre lab

1. Define linear convolution

2. Define circular convolution

3. what is convolution property of DFT

Post lab

1.Find the linear convolution of two sequence manual and check your answer with scilab

$x(n)=\{3,-1,0,1,3,2,0,1,2,1\}$ $h(n)=\{1,1,1\}$

2.Find the linear convolution of two sequence manual and check your answer with scilab

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

3.Find the circular convolution of two sequence manual and check your answer with scilab

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

RESULT:

EXPERIMENT 5

Experiment 5a: Autocorrelation and cross correlation

5.1 Aim: To obtain correlation of two input sequence using scilab

- (i)Auto correlation
- (ii) cross correlation.

Autocorrelation is the convolution of a time series with its time-reversed self. Fourier transform of an autocorrelation is proportional to the Power Spectral Density of time series. The cross correlation of two sequences $x[n]$ and $y[n]=x[n-k]$ shows a peak at the value of k . Hence cross correlation is employed to compute the exact value of the delay k between the 2 signals. Used in radar and sonar applications, where the received signal reflected from the target is the delayed version of the transmitted signal (measure delay to determine the distance of the target)

//program for cross correlation

```
clc;
clear;
close;
x = input("Enter First Sequence: ");
h = input("Enter Second Sequence: ");
y = xcorr(x, h);
disp(y, "Cross correlation =");
```

Enter First Sequence: [1 2 3 4]

Enter Second Sequence: [3 4 5 6]

"Cross correlation = "

6. 17. 32. 50. 38. 25. 12.

//program for Auto correlation

```
clc;
clear; close;
x = input("Enter First Sequence: ");
```

```
//h = input("Enter Second Sequence: ");
y = xcorr(x);
disp(y, "Auto correlation =");
```

out put:

Enter First Sequence: [1 2 3 4]

"Auto correlation

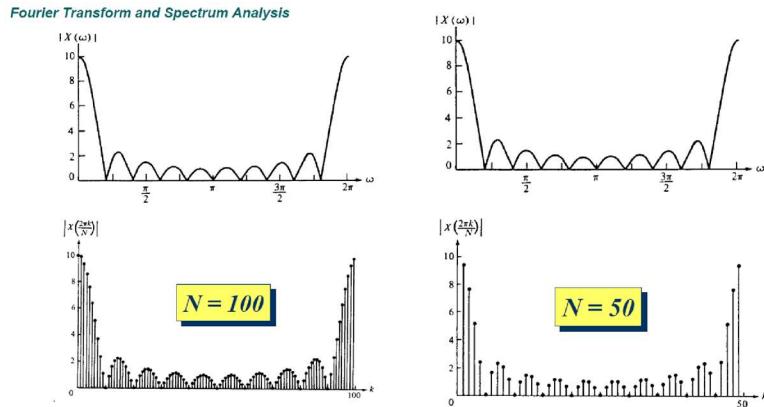
4. 11. 20. 30. 20. 11. 4.

Experiment 5b. Spectrum Analysis using DFT

Aim: To analyze the spectrum signal using DFT in scilab platform

Discrete Fourier Transform:

Spectrum of aperiodic discrete-time signals is periodic and continuous and it is difficult to handle by computer. Since the spectrum is periodic, there's no point to keep all periods – one period is enough. Computer cannot handle continuous data, we can only keep some samples of the spectrum. Interesting enough, such requirements lead to a very simple way to find the spectrum of signal is Discrete Fourier Transform. Discrete Fourier Transform (DFT) is exactly the output of the Fourier Transform of an aperiodic sequence at some particular frequencies.

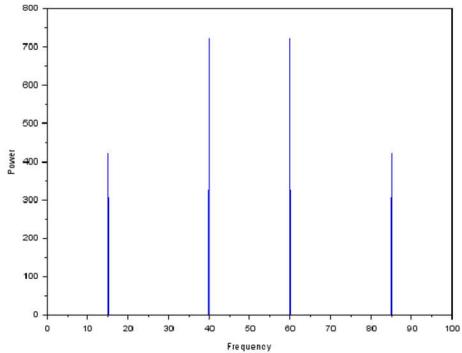


//Spectrum analysis using DFT

```
clc;
close;
clear;
fs=100;
t = 0:1/fs:10-1/fs;
x = ((1.3)*sin(2*%pi*15*t)+(1.7)*sin(2*%pi*40*(t-2)))
y=fft(x)
n = length(x);
```

```
f = (0:n-1)*(fs/n);
power = abs(y).^2/n;
plot(f,power)
xlabel('Frequency')
ylabel('Power')
```

OUTPUT:



Pre lab

1. Define auto correlation?
2. Define correlation
3. Give the properties of auto correlation.
4. Draw the spectrum for periodic and aperiodic signal.

Post lab

1. List the difference between Auto correlation and convolution?
2. List the difference between Auto correlation and cross correlation?

3. What is the length of the resultant sequence of auto correlation?
4. List few applications of correlation.

RESULT

Lab 6 : Efficient computation of DFT and IDFT using FFT

Aim: To obtain the efficient computation of DFT and IDFT using FFT

Software Requirement: SCI Lab

Theory:

DFT:

Discrete Fourier Transform (DFT) is used for performing frequency analysis of discrete time signals. DFT gives a discrete frequency domain representation whereas the other transforms are continuous in frequency domain. The N point DFT of discrete time signal $x[n]$ is given by the equation

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{\frac{-j2\pi kn}{N}} \text{ for } k=0,1,2,\dots,N-1$$

The inverse DFT allows us to recover the sequence $x[n]$ from the frequency samples

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)e^{\frac{-j2\pi kn}{N}} \text{ for } n=0,1,2,\dots,N-1$$

FFT:

A fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse. FFTs are of great importance to a wide variety of applications, from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers. Evaluating the sums of DFT directly would take $O(N^2)$ arithmetical operations. An FFT is an algorithm to compute the same result in only $O(N \log N)$ operations. In general, such algorithms depend upon the factorization of N, but there are FFTs with $O(N \log N)$ complexity for all N, even for prime N. Since the inverse DFT is the same as the DFT, but with the opposite sign in the exponent and a $1/N$ factor, any FFT algorithm can easily be adapted for it as well.

Algorithm:

- 1) Get the input sequence
- 2) Find the FFT of the input sequence using SciLAB function.
- 3) Find the IFFT of the input sequence using SciLAB function.
- 4) Display the above outputs.

Program:

```
// program for calculation of FFT of a signal
clc ;
clf ;
clear all;
N = input('Enter the value of N');
x = input ('enter input sequence');
y = fft(x);
A = real(y);
B = imag(y);
mag = abs(y);
x1 = atan(imag(y),real(y));
phase = x1 *(180/ %pi ) ;
disp ('the resultant FFT sequence is ' );
disp (y);
disp ('the magnitude response is ') ;
disp ( mag ) ;
disp ('the phase response is') ;
disp (phase) ;
z = ifft ( y ) ;
disp ('the resultant IFFT sequence is') ;
disp ( z );
subplot (3 ,2 ,1) ;
plot2d3 ( x );
title ( 'input sequence') ;
subplot (3 ,2 ,2) ;
plot2d3 ( A );
title ( 'FFT real sequence ' ) ;
subplot (3 ,2 ,3) ;
plot2d3 ( B );
title ('FFT imaginary sequence ' ) ;
subplot (3 ,2 ,4) ;
plot2d3 ( mag ) ;
title ( 'magnitude response ' ) ;
subplot (3 ,2 ,5) ;
plot2d3 ( phase ) ;
title ( 'phase response ') ;
subplot (3 ,2 ,6) ;
plot2d3 ( x );
```

```
title ( 'IFFT sequence ' ) ;
```

Results:

Enter the value of N 8

enter input sequence [1 2 3 4 5 6 7 8]

"the resultant FFT sequence is "

column 1 to 5
36. + 0.i -4. + 9.6568542i -4. + 4.i -4. + 1.6568542i -4. + 0.i
column 6 to 8
-4. - 1.6568542i -4. - 4.i -4. - 9.6568542i

"the magnitude response is "

column 1 to 7
36. 10.452504 5.6568542 4.3295688 4. 4.3295688 5.6568542
column 8
10.452504

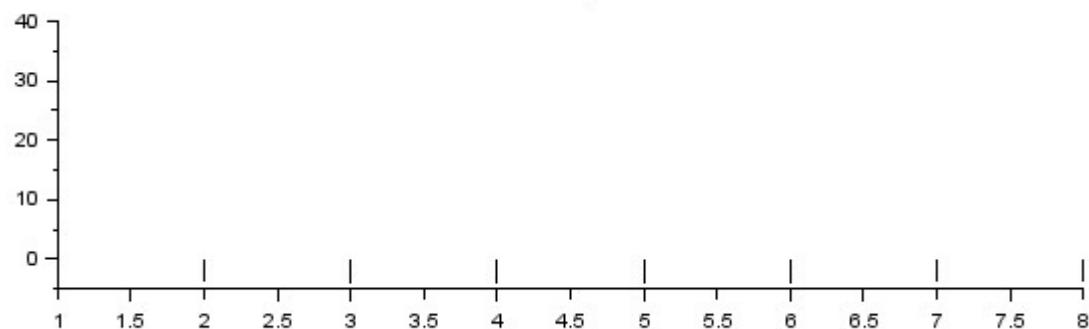
"the phase response is"

0. 112.5 135. 157.5 180. -157.5 -135. -112.5

"the resultant IFFT sequence is"

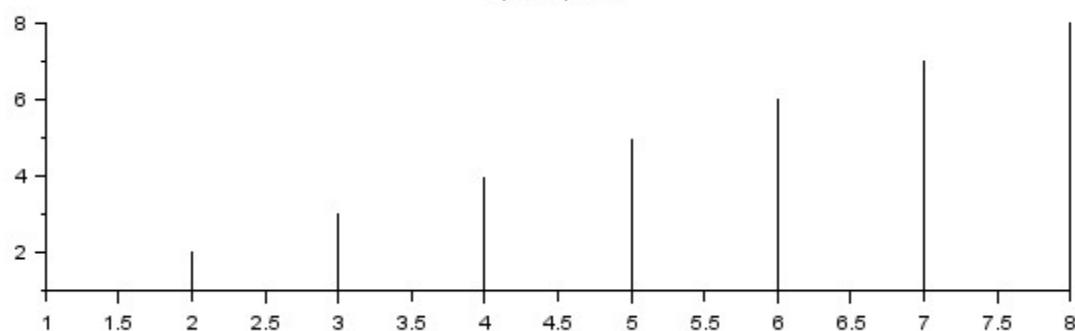
1. 2. 3. 4. 5. 6. 7. 8.

FFT real sequence

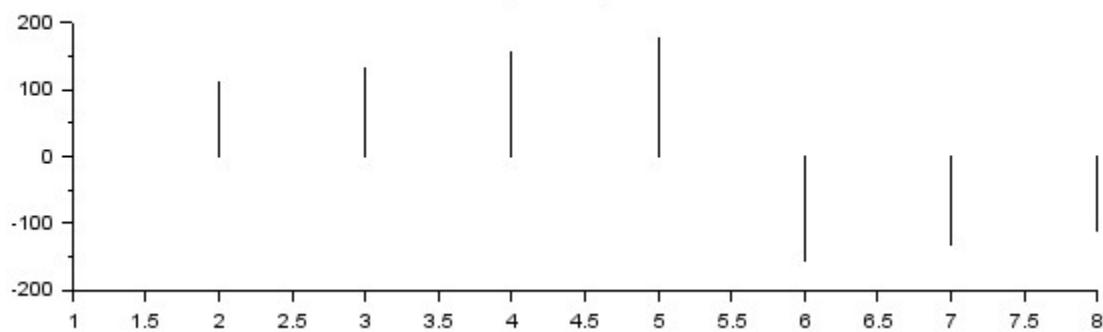


:

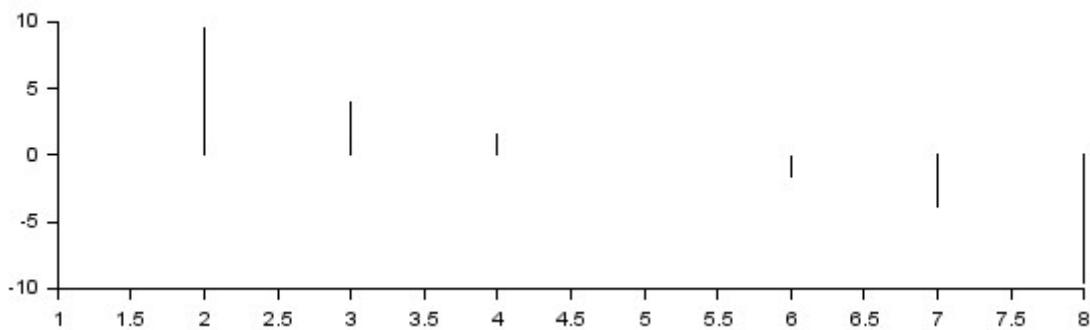
input sequence

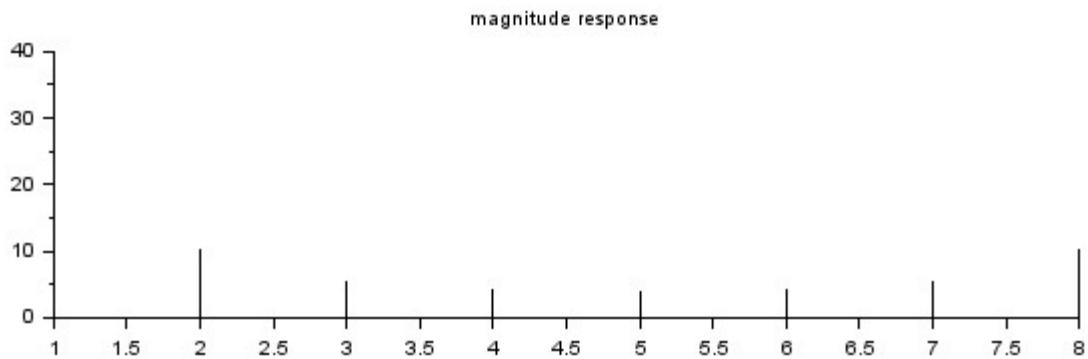


phase response



FFT imaginary sequence





Pre-lab questions:

1. Why do we need Fourier transform in DSP?
2. What is the need of FFT ?
3. What's the difference between FFT and DFT?
4. What is "decimation-in-time" versus "decimation-in-frequency"?

Post-Lab questions:

1. Compute the 8-point FFT of the sequence $x(n) = [1 \ 2 \ 1 \ 2 \ 3 \ 4 \ 3 \ 4]$ using DIT-FFT and DIF-FFT and verify the result using Sci Code.

Result:

Lab 7 : Design of digital FIR Low Pass ,High Pass , Band Pass, band Stop filter using rectangular window

Aim: Design and implementation of FIR Filter (LP /HP /BP /BS) to meet given specifications

Using Windowing technique

- a. Rectangular window

Software Requirement: SCI Lab

Theory:

FIR filters are digital filters with finite impulse response. They are also known as non-recursive digital filters as they do not have the feedback.

An FIR filter has two important advantages over an IIR design:

- Firstly, there is no feedback loop in the structure of an FIR filter. Due to not having a feedback loop, an FIR filter is inherently stable. Meanwhile, for an IIR filter, we need to check the stability.
- Secondly, an FIR filter can provide a linear-phase response. As a matter of fact, a linear-phase response is the main advantage of an FIR filter over an IIR design otherwise, for the same filtering specifications; an IIR filter will lead to a lower order.

FIR FILTER DESIGN

An FIR filter is designed by finding the coefficients and filter order that meet certain specifications, which can be in the time-domain (e.g. a matched filter) and/or the frequency domain (most common). Matched filters perform a cross-correlation between the input signal and a known pulse-shape. The FIR convolution is a cross-correlation between the input signal and a time-reversed copy of the impulse-response. Therefore, the matched-filter's impulse response is "designed" by sampling the known pulse-shape and using those samples in reverse order as the coefficients of the filter. When a particular frequency response is desired, several different design methods are common:

1. Window design method

WINDOW DESIGN METHOD

In the window design method, one first designs an ideal IIR filter and then truncates the infinite impulse response by multiplying it with a finite length window function. The result is a finite impulse response filter whose frequency response is modified from that of the IIR filter.

Algorithm:

- 1) Get the input analog cut off frequency.
- 2) Get the input sampling frequency.
- 3) Get the filter order.
- 4) Find the digital cut off frequency.
- 5) Normalize the digital cut off frequency.
- 6) Using built in filter function find the time domain filter coefficients, frequency domain filter response and Frequency grid for LPF,HPF,BPF and BSF.
- 7) Plot the magnitude response of the filter with respect to Normalized Digital Frequency and Analog Frequency.

Program(a)Low pass filter

```
// To Design a Low Pass FIR Filter
// Filter Length =5 , Order = 4
//Window = Rectangular Window
clc ;
clear ;
xdel(winsid());
fc=input('Enter Analog cutoff freq.in Hz=')
fs=input('Enter Analog sampling freq.in Hz=')
M=input('Enter order of filter =')
w=(2*pi)*(fc/fs);
disp(w,'Digital cut off frequency in radians,cycles/samples');
wc=w/%pi;
```

```

disp(wc,'Normalized digital cut off frequency in cycles/samples');
[wft,wfm,fr]=wfir('lp',M+1,[wc/2,0],'re',[0,0]);
disp(wft,'Impulse Response of LPF FIR Filter:h[n]=');
// Plotting the Magnitude Response of LPF FIR Filter
subplot(2,1,1);
plot(2*fr,wfm);
xlabel('Normalized Digital Frequency w--->')
ylabel('Magnitude | H(w) | =')
title('Magnitude Response of FIR LPF')
xgrid(1)
subplot(2,1,2)
plot(fr*fs,wfm)
xlabel('Analog Frequency in Hz f--->')
ylabel('Magnitude | H(w) | =')
title('Magnitude Response of FIR LPF')
xgrid(1)

```

Results

Enter Analog cutoff freq.in Hz=250

Enter Analog sampling freq.in Hz=2000

Enter order of filter =4

0.7853982

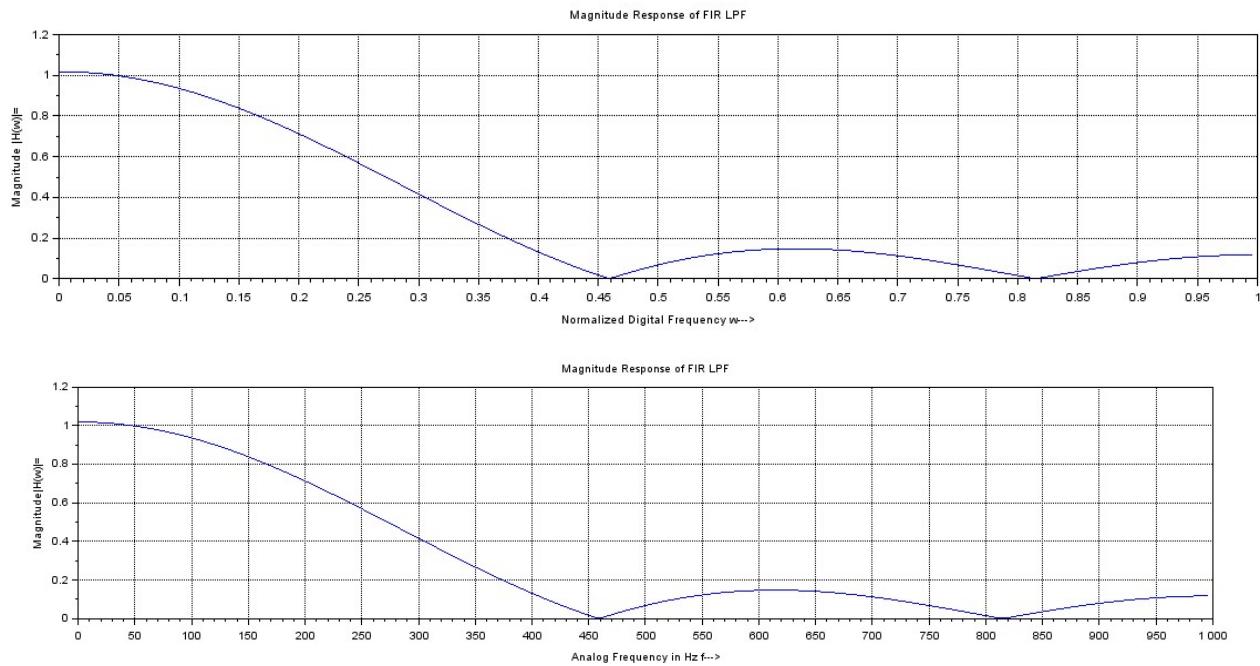
"Digital cut off frequency in radians,cycles/samples"

0.25

"Normalized digital cut off frequency in cycles/samples"

0.1591549 0.2250791 0.25 0.2250791 0.1591549

"Impulse Response of LPF FIR Filter:h[n] ="



Program:(b)High pass filter

```
// To Design an High Pass FIR Filter
// Filter Length =5 , Order = 4
//Window = Rectangular Window
clc;
clear;
xdel(winsid());
fc=input('Enter Analog cut off freq in Hz=')// 250
fs=input('Enter Analog sampling freq in Hz=')//2000
M=input('Enter order of filter ='//4
w=(2*%pi)*(fc/fs);
disp(w,'Digital cut off frequency in radians cycles/samples');
wc=w/%pi;
disp(wc,'Normalized digital cut off frequency in cycles/samples');
[wft,wfm,fr]=wfir('hp',M+1,[wc/2,0],'re',[0,0]);
disp(wft,'Impulse Response of HPF FIR Filter:h[n]=');
// Plotting the Magnitude Response of HPF FIR Filter
subplot(2,1,1)
plot(2*fr,wfm)
```

```

xlabel('Normalized Digital Frequency w-->')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of FIR HPF')
xgrid(1)
subplot(2,1,2)
plot(fr*fs,wfm)
xlabel('Analog Frequency in Hz f-->')
ylabel ('Magnitude |H(w)|=')
title('Magnitude Response of FIR HPF')
xgrid(1)

```

Results:

Enter Analog cut off freq in Hz=250

Enter Analog sampling freq in Hz=2000

Enter order of filter =4

0.7853982

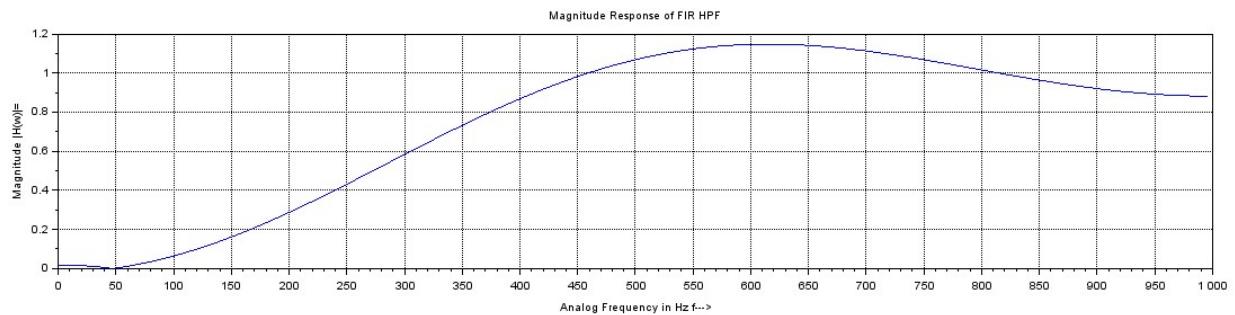
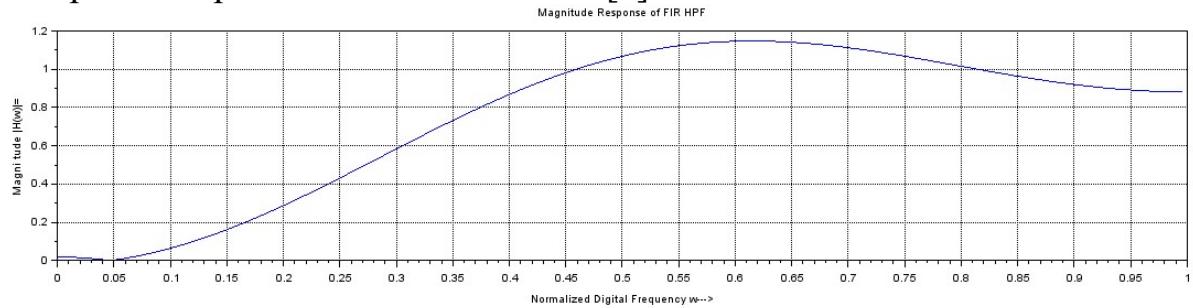
"Digital cut off frequency in radians cycles/samples"

0.25

"Normalized digital cut off frequency in cycles/samples"

-0.1591549 -0.2250791 0.75 -0.2250791 -0.1591549

"Impulse Response of HPF FIR Filter:h[n]+"



Program:(c)Band pass filter

```
// To Design an Band Pass FIR Filter
// Filter Length =5 , Order = 4
//Window = Rectangular Window
clc ;
clear;
xdel( winsid());
fc1=input('Enter Analog lower cut off freq.in Hz=') // 250
fc2=input('Enter Analog higher cut off freq. in Hz=') // 600
fs=input('Enter Analog sampling freq.in Hz=')//2000
M=input('Enter order of filter =')//4
w1=(2*%pi)*(fc1/fs);
w2=(2*%pi)*(fc2/fs);
disp(w1,'Digital lower cut off frequency in radians cycles/samples');
disp(w2,'Digital higher cut off frequency in radians cycles/samples');
wc1=w1/%pi;
wc2= w2/%pi;
disp(wc1,'Normalized digital lower cut off frequency in cycles/ samples');
disp(wc2,'Normalized digital higher cut off frequency in cycles / samples');
[wft,wfm,fr]=wfir('bp' ,M+1,[wc1/2,wc2/2],'re',[0,0]);
disp(wft,'Impulse Response of BPF FIR Filter : h[ n]= ');
// Plotting the Magnitude Response of HPF FIR Filter
subplot(2,1,1)
plot(2*fr,wfm)
xlabel('Normalized Digital Frequency w--->')
ylabel('Magnitude | H(w) |=')
title('Magnitude Response of FIR BPF')
xgrid(1)
subplot(2,1,2)
plot(fr*fs,wfm)
xlabel('Analog Frequency in Hz f --->')
ylabel('Magnitude | H(w) | = ')
title ('Magnitude Response of FIR BPF')
xgrid(1)
```

Results:

Enter Analog lower cut off freq.in Hz=250

Enter Analog higher cut off freq. in Hz=600

Enter Analog sampling freq.in Hz=2000

Enter order of filter =4

0.7853982

"Digital lower cut off frequency in radians cycles/samples"

1.8849556

"Digital higher cut off frequency in radians cycles/samples"

0.25

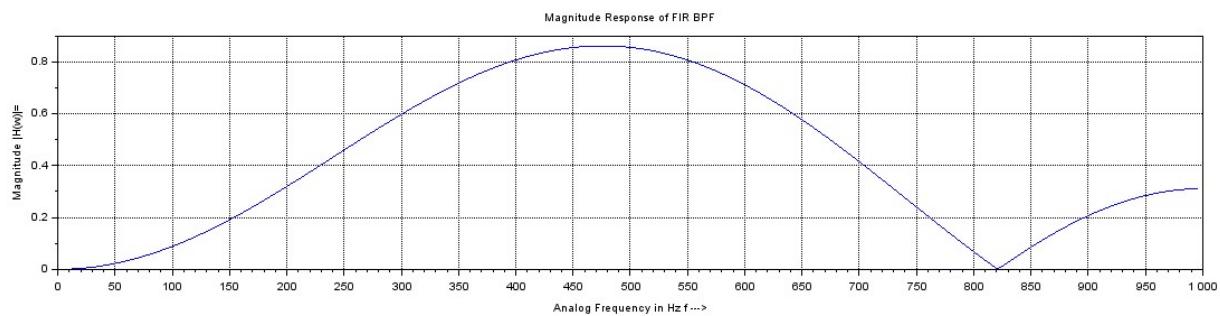
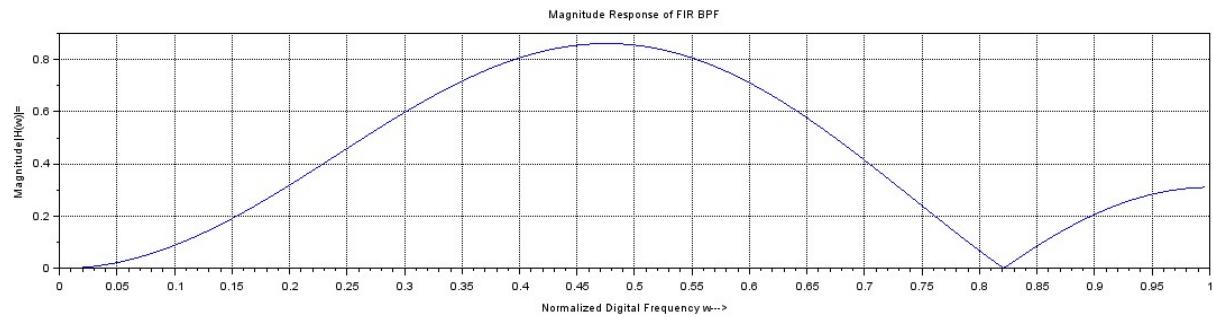
"Normalized digital lower cut off frequency in cycles/ samples"

0.6

"Normalized digital higher cut off frequency in cycles / samples"

-0.2527039 0.0776516 0.35 0.0776516 -0.2527039

"Impulse Response of BPF FIR Filter : $h[n] =$ "



Program:(d)Band stop filter

```
// Filter Length =5 , Order = 4
//Window = Rectangular Window
clc ;
clear ;
xdel ( winsid ());
fc1 = input ('Enter Analog lower cut off freq in Hz=') // 250
fc2 = input ('Enter Analog higher cut off freq in Hz=') // 600
fs = input ('Enter Analog sampling freq in Hz=') //2000
M = input ('Enter order of filter =') // 4
w1 = (2* %pi )*( fc1/fs);
w2 = (2* %pi )*( fc2/fs);
disp (w1,'Digital lower cut off frequency in radians cycles/samples' );
disp(w2 , 'Digital higher cut off frequency in radians.cycles/samples ' );
wc1 = w1/%pi;
wc2 = w2/%pi;
disp (wc1 , 'Normalized digital lower cut off frequency in cycles / samples
' );
disp (wc2 , 'Normalized digital higher cut off frequency in cycles / samples
' );
[wft,wfm,fr]= wfir('sb',M+1,[wc1/2,wc2/2],'re',[0,0]);
disp (wft , 'Impulse Response of BSF FIR Filter : h [ n]= ' );
// Plotting the Magnitude Response of HPF FIR Filter
subplot (2 ,1 ,1)
plot (2*fr , wfm )
xlabel ( ' Normalized Digital Frequency w--->' )
ylabel ( 'Magnitude |H(w)|= ' )
title ( 'Magnitude Response of FIR BSF ' )
xgrid (1)
subplot (2 ,1 ,2)
plot (fr*fs , wfm )
xlabel ( ' Analog Frequency in Hz f --->')
ylabel ('Magnitude |H(w) |= ')
title ('Magnitude Response of FIR BSF')
xgrid (1)
```

Results:

Enter Analog lower cut off freq in Hz=250

Enter Analog higher cut off freq in Hz=600

Enter Analog sampling freq in Hz=2000

Enter order of filter =4

0.7853982

"Digital lower cut off frequency in radians cycles/samples"

1.8849556

"Digital higher cut off frequency in radians.cycles/samples "

0.25

"Normalized digital lower cut off frequency in cycles / samples "

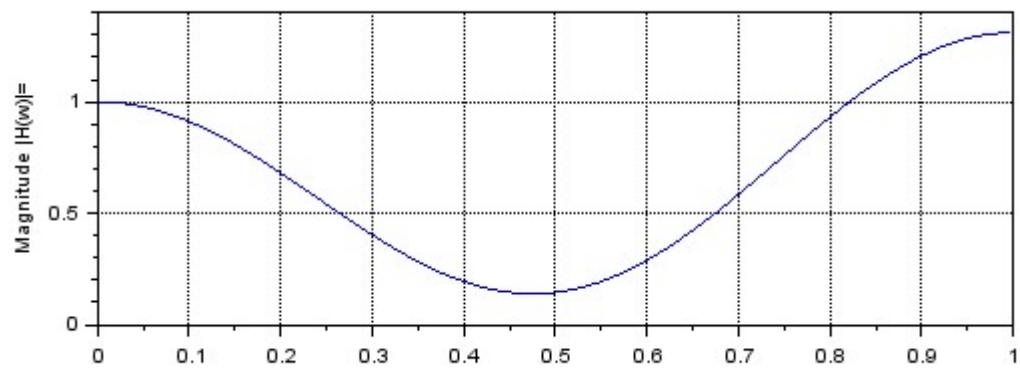
0.6

"Normalized digital higher cut off frequency in cycles / samples "

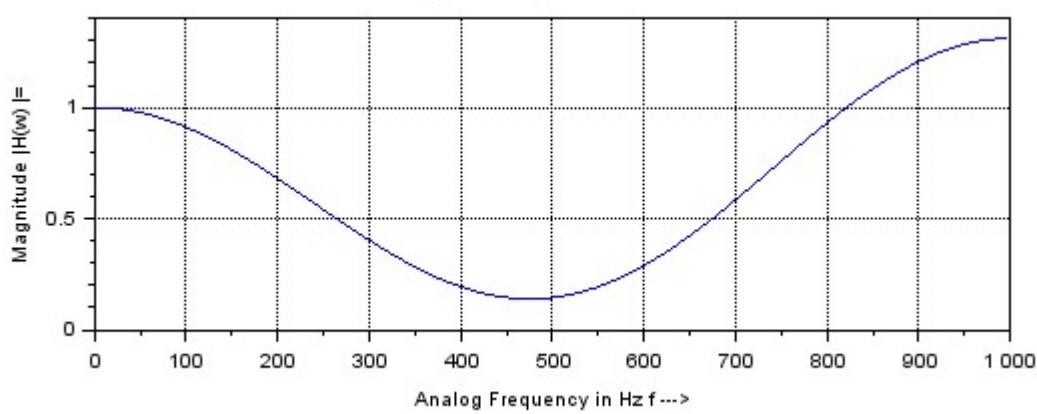
0.2527039 -0.0776516 0.65 -0.0776516 0.2527039

"Impulse Response of BSF FIR Filter : h [n]= "

Magnitude Response of FIR BSF



Magnitude Response of FIR BSF



8. Design of digital FIR Low Pass and High Pass filter using Hanning and Hamming window

Theory

Finite Impulse Response (FIR) Filter

FIR filters are digital filters with finite impulse response. They are also known as non-recursive digital filters as they do not have the feedback (a recursive part of a filter), even though recursive algorithms can be used for FIR filter realization. Hence it is an all zero filter. Therefore, input and output difference equation for FIR filter is given by

$$y(n) = b_0x(n) + b_1x(n-1) + b_2x(n-2) + \dots + b_{M-1}x(n-N+1)$$

Where $b_0, b_1, b_2 \dots b(M-1)$ are filter coefficients. FIR filters are particularly useful for applications where exact linear phase response is required. The FIR filter is generally implemented in a non-recursive way which guarantees a stable filter.

FIR filters can be designed using different methods, but most of them are based on ideal filter approximation. The objective is not to achieve ideal characteristics, as it is impossible anyway, but to achieve sufficiently good characteristics of a filter. The transfer function of FIR filter approaches the ideal as the filter order increases, thus increasing the complexity and amount of time needed for processing input samples of a signal being filtered.

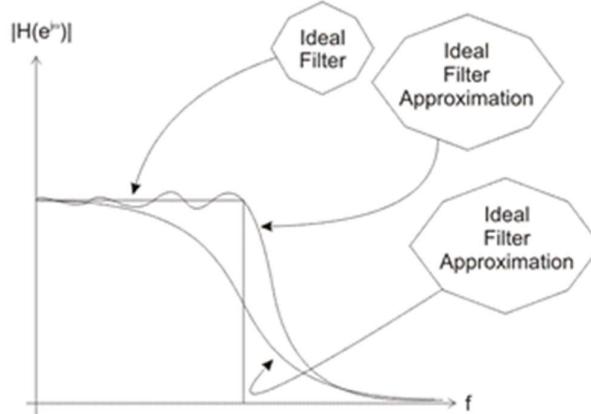


Fig-1

FIR filters can have linear phase characteristic, which is not like IIR filters. Obviously, in such cases when it is necessary to have a linear phase characteristic, FIR filters are the only option available. If the linear phase characteristic is not necessary, as is the case with processing speech signals, FIR filters are not good solution at all.

One of the drawbacks of FIR filters is a high order of designed filter. The order of FIR filter is remarkably higher compared to an IIR filter with the same frequency response. This is the reason why it is so important to use FIR filters only when the linear phase characteristic is very important.

A number of delay lines contained in a filter, i.e. a number of input samples that should be saved for the purpose of computing the output sample, determines the order of a filter. For example, if the filter is assumed to be of order 10, it means that it is necessary to save 10 input samples preceding the current sample. All eleven samples will affect the output sample of FIR filter

The transform function of a typical FIR filter can be expressed as a polynomial of a complex variable z^{-1} . All the poles of the transfer function are located at the origin. For this reason, FIR filters are guaranteed to be stable, whereas IIR filters have potential to become unstable

Basic concepts and FIR filter specification

Most FIR filter design methods are based on ideal filter approximation. The resulting filter approximates the ideal characteristic as the filter order increases, thus making the filter and its implementation more complex.

Figure 2.a and 2.b illustrates a low-pass digital filter specification. The word specification actually refers to the frequency response specification

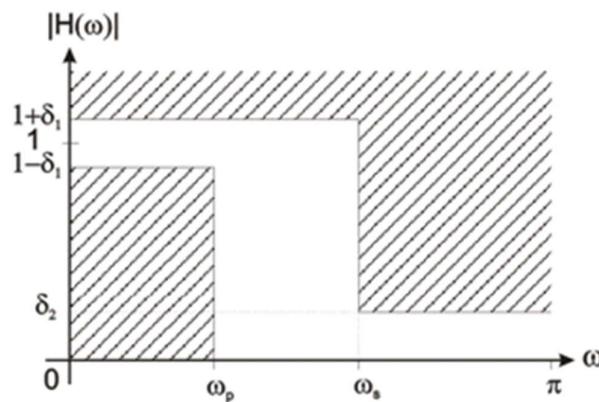


Figure 2.a: Low-pass digital filter specification

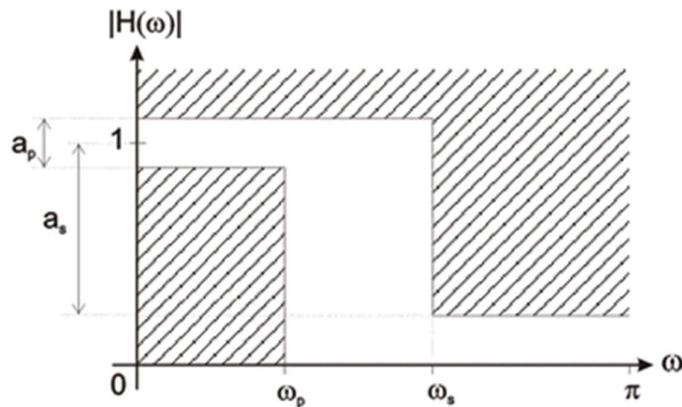


Figure 2.b: Low-pass digital filter specification

ω_p -normalized cut-off frequency in the passband;

ω_s - normalized cut-off frequency in the stopband;

δ_1 - maximum ripples in the passband

δ_2 . minimum attenuation in the stopband [dB]

a_p . maximum ripples in the passband; and

a_s . minimum attenuation in the stopband [dB].

$$a_p = 20 \log_{10}(1 + \delta_1 / 1 - \delta_1)$$

$$a_s = -20 \log_{10} \delta_1$$

Frequency normalization can be expressed as follows:

$$\omega = (2\pi f / f_s)$$

where:

f_s . is a sampling frequency;

f - is a frequency to normalize; and

ω - is normalized frequency.

Specifications for high-pass, band-pass and band-stop filters are defined almost the same way as those for low-pass filters. Figure 3.a and 3.b illustrates a high-pass filter specification

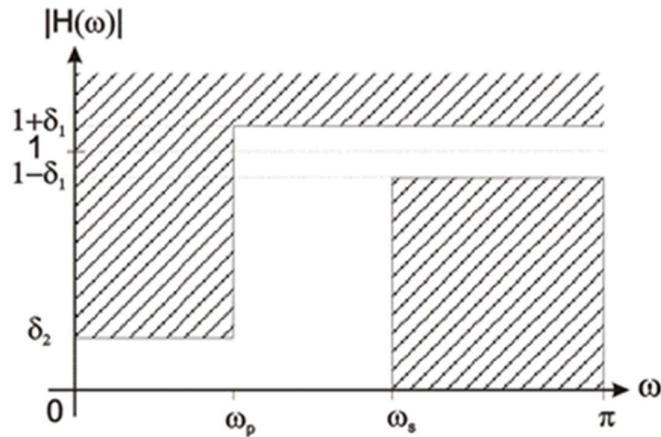


Figure 3.b: High-pass digital filter specification

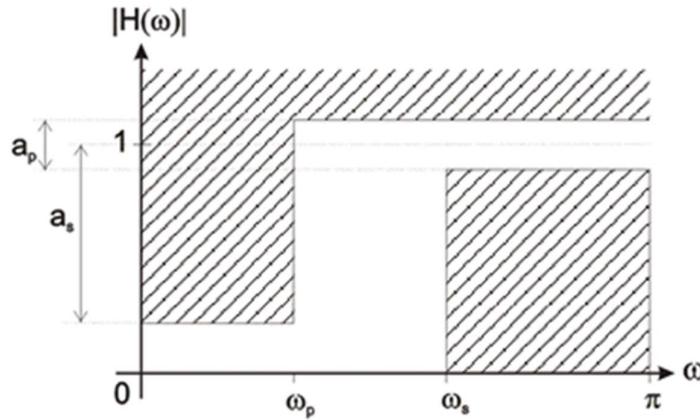


Figure 3.b: High-pass digital filter specification

Comparing these two figures 2.a, 2.b and 3.a, 3.b, it is obvious that low-pass and high-pass filters have similar specifications. The same values are defined in both cases with the difference that in the latter case the pass band is substituted by the stop band and vice versa.

Finite impulse response (FIR) filter design methods

The filter design process starts with specifications and requirements of the desirable FIR filter. Which method is to be used in the filter design process depends on the filter specifications and implementation?

There are essentially three well-known methods for FIR filter design namely:

1. The window method.
2. The frequency sampling technique.
3. Optimal filter design methods

Each of the given methods has its advantages and disadvantages. Thus, it is very important to carefully choose the right method for FIR filter design. Due to its simplicity and efficiency, the window method is most commonly used method for designing filters. The sampling frequency method is easy to use, but filters designed this way have small attenuation in the stop band.

Out of these three techniques we are going to discuss about window method only

Ideal filter approximation

The ideal filter frequency response is used when designing FIR filters using window functions. The objective is to compute the ideal filter samples. FIR filters have finite impulse response, which means the ideal filter frequency sampling must be performed in a finite number of points. As the ideal filter frequency response is infinite, it is easy to produce sampling errors. The error is less as the filter order increases. Figure 4.a and 4.b illustrates the transfer functions of two standard ideal filters

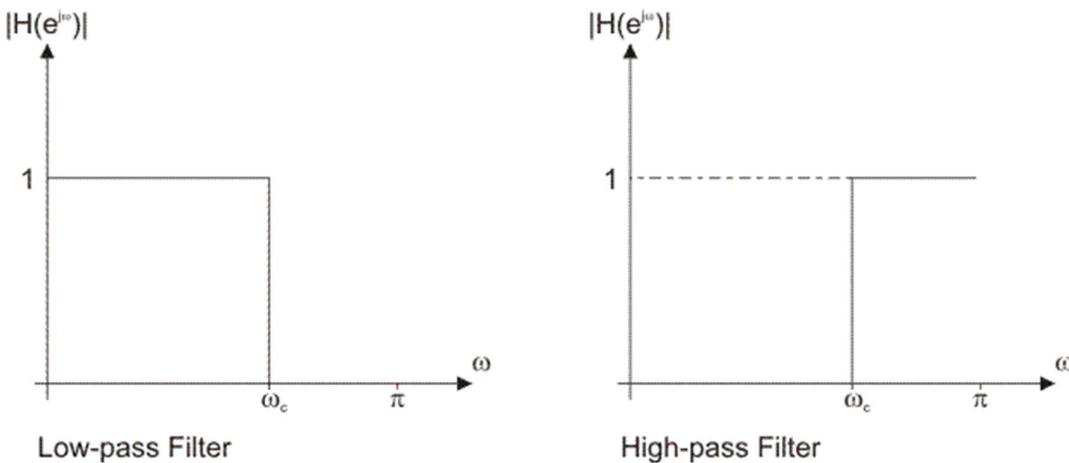


Figure 4.a and 4.b: Transfer functions of two standard ideal filters

The ideal filter frequency response can be computed via inverse Fourier transform. The two standard ideal filters frequency responses are contained in the table 1 below.

| Type of filter | Frequency response $h_d[n]$ |
|------------------|---|
| low-pass filter | $h_d[n] = \begin{cases} \frac{\sin[\omega_c(n - M)]}{\pi(n - M)}; & n \neq M \\ \frac{\omega_c}{\pi}; & n = M \end{cases}$ |
| high-pass filter | $h_d[n] = \begin{cases} 1 - \frac{\omega_c}{\pi}; & n \neq M \\ -\frac{\sin(\omega_c(n - M))}{\pi(n - M)}; & n = M \end{cases}$ |

Table 1: The frequency responses of two standard ideal filters

The value of variable n ranges between 0 and N , where N is the filter order. A constant M can be expressed as $M = N / 2$. Equivalently, N can be expressed as $N = 2M$

The constant M is an integer if the filter order N is even, which is not the case with odd order filters. If M is an integer (even filter order), the ideal filter frequency response is symmetric about its M th sample which is found via expression shown in the table 1 above. If M is not an integer, the ideal filter frequency response is still symmetric, but not about some frequency response sample

Since the variable n ranges between 0 and N , the ideal filter frequency response has $N+1$ samples

If it is needed to find frequency response of a non-standard ideal filter, the expression for inverse Fourier transform must be used:

$$h_d[n] = \frac{1}{\pi} \int_0^\pi e^{j\omega(n-M)} d\omega$$

Non-standard filters are rarely used. However, if there is a need to use some of them, the integral above must be computed via various numerical methods

FIR filter design using window functions

The FIR filter design process via window functions can be split into several steps:

1. Defining filter specifications;
2. Specifying a window function according to the filter specifications;
3. Computing the filter order required for a given set of specifications;
4. Computing the window function coefficients;
5. Computing the ideal filter coefficients according to the filter order;
6. Computing FIR filter coefficients according to the obtained window function and ideal filter coefficients
7. If the resulting filter has too wide or too narrow transition region, it is necessary to change the filter order by increasing or decreasing it according to needs, and after that steps 4, 5 and 6 are iterated as many times as needed

The final objective of defining filter specifications is to find the desired normalized frequencies, transition width and stopband attenuation. The window function and filter order are both specified according to these parameters. Accordingly, the selected window function must satisfy the given specifications.

After this step, that is, when the window function is known, we can compute the filter order required for a given set of specifications.

When both the window function and filter order are known, it is possible to calculate the window function coefficients $w[n]$ using the formula for the specified window function

After estimating the window function coefficients, it is necessary to find the ideal filter frequency samples. The final objective of this step is to obtain the coefficients $h_d[n]$. Two sequences $w[n]$ and $h_d[n]$ have the same number of elements.

The next step is to compute the frequency response of designed filter $h[n]$ using the following expression:

$$h[n] = w[n] \cdot h_d[n]$$

Lastly, the transfer function of designed filter will be found by transforming impulse response via Fourier transform:

$$H(e^{jw}) = \sum_{n=0}^N h(n) \cdot e^{-jnw}$$

If the transition region of designed filter is wider than needed, it is necessary to increase the filter order, reestimate the window function coefficients and ideal filter frequency samples, multiply them in order to obtain the frequency response of designed filter and reestimate the transfer function as well. If the transition region is narrower than needed, the filter order can be decreased for the purpose of optimizing hardware and/or software resources. It is also necessary to reestimate the filter frequency coefficients after that. For the sake of precise estimates, the filter order should be decreased or increased by 1.

Window functions

The window method is most commonly used method for designing FIR filters. The simplicity of design process makes this method very popular.

A window is a finite array consisting of coefficients selected to satisfy the desirable requirements. This chapter provides a few methods for estimating coefficients and basic characteristics of the window itself as well as the result filters designed using these coefficients. The point is to find these coefficients denoted by $w[n]$.

When designing digital FIR filters using window functions it is necessary to specify:

A window function to be used; and

The filter order according to the required specifications (selectivity and stop band attenuation).

These two requirements are interrelated. Each function is a kind of compromise between the two following requirements:

The higher the selectivity, i.e. the narrower the transition region; and

The higher suppression of undesirable spectrum, i.e. the higher the stop band attenuation.

Table 2 below contains all window functions mentioned in this chapter and briefly compares their selectivity and stop band attenuation

These two requirements are interrelated. Each function is a kind of compromise between the two following requirements:

The higher the selectivity, i.e. the narrower the transition region; and

The higher suppression of undesirable spectrum, i.e. the higher the stop band attenuation.

Table 2 below contains all window functions mentioned in this chapter and briefly compares their selectivity and stop band attenuation.

Special attention should be paid to the fact that minimum attenuation of window function and that of the filter designed using that function are different in most cases. The difference, i.e. additional attenuation occurs under the process of designing a filter using window functions. This affects the stop band attenuation to become additionally higher, which is very desirable.

However, a drawback of this method is that the minimum stop band attenuation is fixed for each function. The following concepts such as the main lobe, main lobe width, side lobes, transition region, minimum stopband attenuation of window function and minimum stopband attenuation of designed filter are described in more detail in Figure 5

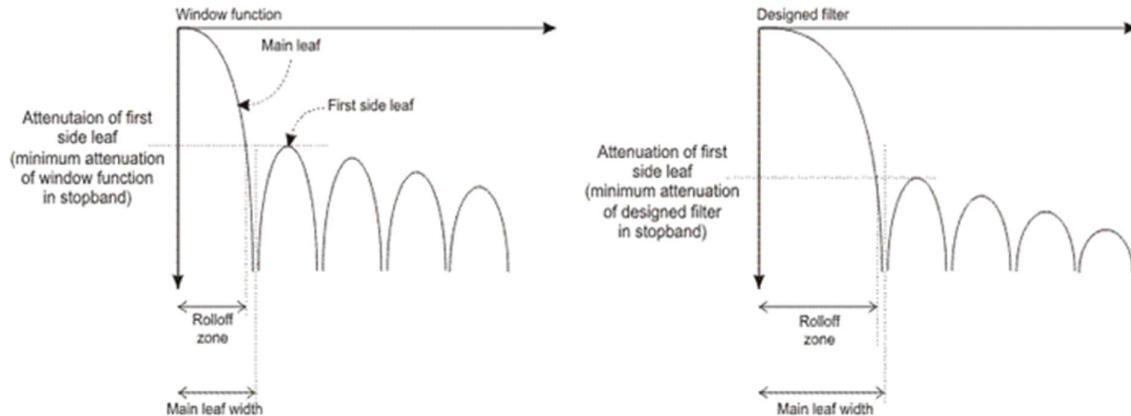
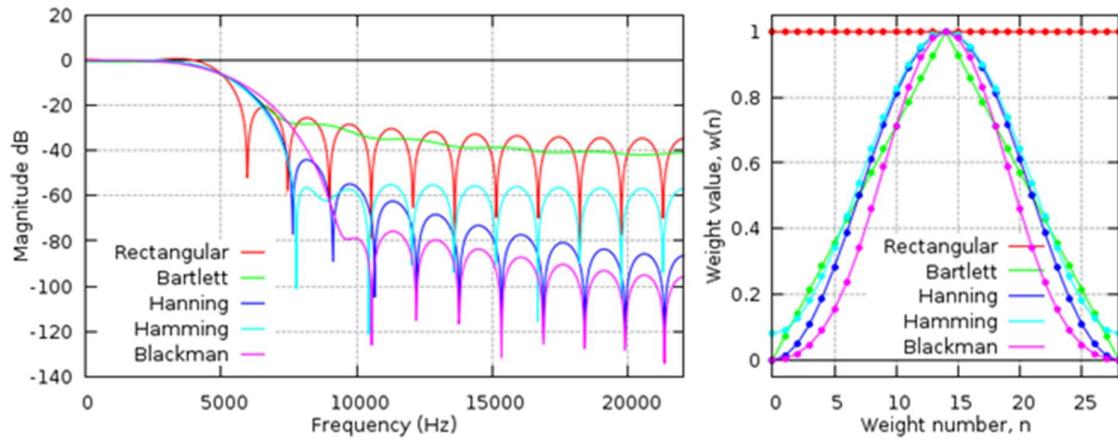


Figure 5: Main lobe, main lobe width, side lobes, transition region

As can be seen in the table 2 above, the stopband attenuation of these windows is not adjustable. It is only possible to affect the transition region by increasing the filter order. For this reason, it is preferable to start design process by specifying the appropriate window function on the basis of the stopband attenuation. It is most preferable to specify a window with the least stopband attenuation that satisfies the given requirements. This enables the designed filter to have the narrowest transition region

| Name of Window | Time-domain sequence of window $w(n)$, $0 \leq n \leq N-1$ |
|--|---|
| Rectangular Window | 1 |
| Barlett (triangular) window | $1 - \frac{2 n - \frac{N-1}{2} }{N-1}$ |
| Blackman Window | $0.42 - 0.5\cos\frac{2\pi n}{N-1} + 0.08\cos\frac{4\pi n}{N-1}$ |
| Hamming Window | $0.54 - 0.46\cos\frac{2\pi n}{N-1}$ |
| Hanning Window | $\frac{1}{2}(1 - \cos\frac{2\pi n}{N-1})$ |
| Kaiser Window | $\frac{I_0[\alpha\sqrt{\left(\frac{N-1}{2}\right)^2 - \left(n - \frac{N-1}{2}\right)^2}]}{I_0[\alpha\left(\frac{N-1}{2}\right)]}$ |
| <i>I₀(x) is the zeroth order Bessel function.</i> | |

Frequency Response and Weight Values of different windows types



Design an ideal high pass filter with a frequency response

$$H_d e^{(j\omega)} = 1 \text{ for } \pi/4 \leq |\omega| \leq \pi$$

$= 0$ for $|\omega| \leq \pi/4$. Find the values of $h(n)$ for $N= 11$. Find $H(z)$. Plot the magnitude response using Hanning window and Hamming window.

Formula for Hanning window

The rectangular window (no window function):

$$w_R(n) = 1 \quad 0 \leq n \leq N - 1$$

The triangular window:

$$w_{tri}(n) = 1 - \frac{|2n - N + 1|}{N - 1}, \quad 0 \leq n \leq N - 1$$

The Hamming window:

$$w_{hm}(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N - 1}\right), \quad 0 \leq n \leq N - 1$$

The Hanning window:

$$w_{hn}(n) = 0.5 - 0.5 \cos\left(\frac{2\pi n}{N - 1}\right), \quad 0 \leq n \leq N - 1$$

```
clear ;
```

```
clc ;
```

```
close ;
```

```
N=11;
```

```
U=6;
```

```

h_hann = window( 'hn' ,N ) ;

for n = -5+ U :1:5+ U

if n ==6

hd ( n)=0.75;

else

hd ( n)=( sin ( %pi *( n - U ) ) -sin( %pi *( n - U ) /4) )/( %pi *( n - U )) ;

end

h ( n)=h_hann ( n ) * hd ( n ) ;

end

[ hzm , fr ]= frmag ( h ,256) ;

hzm_dB = 20* log10 ( hzm ) ./ max ( hzm );

figure

plot (2* fr , hzm_dB )

a = gca ();

xlabel ('Frequency w*pi');

ylabel ( 'Magnitude in dB') ;

title ( 'Frequency Response of FIR HPF with N =11 Using Hanning Window') ;

xgrid (2);

```

- ✓ In the below figure the hanning window coefficients are computed in *h_hann(-5), h_hann(-4), h_hann(-3), h_hann(-2), h_hann(-1), h_hann(0), h_hann(1), h_hann(2), h_hann(3), h_hann(4), h_hann(5)* as a total of 11 values (N=11).
- ✓ And then the $h_d(n)$ followed by $h(n)$ (Multiplying *h_hann(n) * h_d(n)*).

```

--> N =11;

--> U =6;

--> h_hann = window ( 'hn',N )
h_hann =
0. 0.0954915 0.3454915 0.6545085 0.9045085 1. 0.9045085 0.6545085 0.3454915 0.0954915 0.

--> for n = -5+ U :1:5+ U
> if n ==6
> hd ( n)=0.75;
> else
> hd ( n)=( sin ( %pi *( n - U ) ) -sin( %pi *( n - U ) /4) )/( %pi *( n - U )) ;
> end
> h ( n)=h_hann ( n ) * hd ( n ) ;
> end

--> [ hzm , fr ]= frmag ( h ,256) ;

-->

```

- ✓ “frmag” is used to find all the 256 points for the “ ω ” i.e., $h'e^{j\omega}$
- ✓ For Example, the value “0.0826497” corresponds to $h'e^{j\omega}$ where $\omega=0^\circ$ likewise its calculated for all 256 points of “ ω ”.
- ✓ Then its is converted in to dB values by “hzm_dB”.
- ✓ Finally, the graph is plotted between “ ω ” and “hzm_dB”.

```
--> [ hzm , fr ]= frmag ( h ,256)
hzm =
column 1 to 15

0.0826497  0.0827782  0.0831637  0.0838059  0.0847044  0.0858585  0.0872675  0.0889303  0.0908457  0.0930125  0.0954291  0.0980939  0.101005  0.1041603  0.1075576

column 16 to 30

0.1111947  0.115069  0.1191777  0.1235181  0.1280872  0.1328818  0.1378986  0.1431342  0.1485849  0.1542471  0.1601169  0.1661903  0.1724632  0.1789313  0.1855903

column 31 to 45

0.1924357  0.1994628  0.206667  0.2140435  0.2215874  0.2292936  0.2371571  0.2451728  0.2533353  0.2616393  0.2700795  0.2786504  0.2873464  0.296162  0.3050916

column 46 to 60

0.3141295  0.3232701  0.3325074  0.3418359  0.3512496  0.3607429  0.3703098  0.3799446  0.3896414  0.3993944  0.4091977  0.4190456  0.4289322  0.4388518  0.4487987

column 61 to 75

0.4587672  0.4687515  0.4787461  0.4887454  0.4987438  0.5087359  0.5187162  0.5286795  0.5386205  0.5485338  0.5584146  0.5682575  0.5780578  0.5878106  0.5975111

column 76 to 90

0.6071545  0.6167364  0.6262522  0.6356976  0.6450683  0.6543602  0.6635692  0.6726914  0.6817231  0.6906605  0.6995001  0.7082384  0.7168723  0.7253984  0.7338138

column 91 to 105
```

```

-> hzm_dB = 20* log10 ( hzm ) ./ max ( hzm )
hzm_dB =

```

column 1 to 15

```

-21.52367 -21.510254 -21.470141 -21.403728 -21.311665 -21.19483 -21.05431 -20.891363 -20.707389 -20.503893 -20.282454 -20.044686 -19.792216 -19.526651 -19.249561

```

column 16 to 30

```

-18.962457 -18.666782 -18.363896 -18.05507 -17.741495 -17.424229 -17.104296 -16.782592 -16.459935 -16.137061 -15.814627 -15.493221 -15.173361 -14.855505 -14.540055

```

column 31 to 45

```

-14.22736 -13.917725 -13.611411 -13.308643 -13.009612 -12.714476 -12.42337 -12.136402 -11.853661 -11.575216 -11.301118 -11.031406 -10.766104 -10.505226 -10.248774

```

column 46 to 60

```

-9.9967445 -9.7491237 -9.5058924 -9.2670258 -9.0324938 -8.8022625 -8.5762942 -8.3545483 -8.1369816 -7.9235489 -7.7142031 -7.5088959 -7.3075777 -7.1101982 -6.9167064

```

column 61 to 75

```

-6.7270508 -6.5411797 -6.3590412 -6.1805835 -6.005755 -5.8345039 -5.6667791 -5.5025298 -5.3417053 -5.1842557 -5.0301315 -4.8792836 -4.7316637 -4.5872238 -4.4459167

```

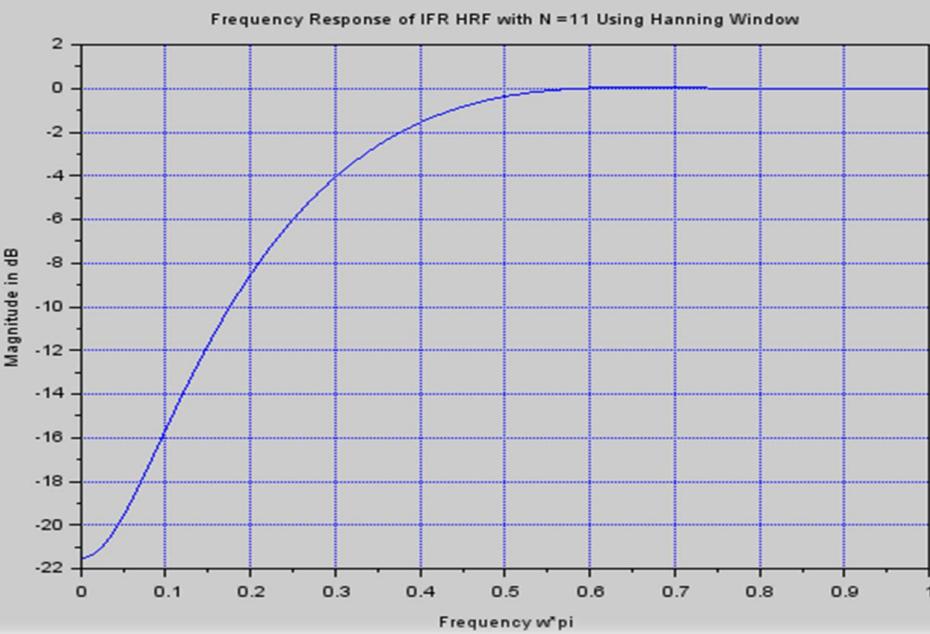


Fig. 1: Magnitude response of HPF using Hanning Window

Similarly Using Hamming Window we can obtain the HPF response

```

clear ;

clc ;
close ;

N=11;
U =6;

h_hamm = window ( 'hm' ,N ) ;

for n = -5+ U :1:5+ U

if n ==6

hd ( n)=0.75;

else

hd ( n)=( sin ( %pi *( n - U ) ) -sin( %pi *( n - U ) /4) )/( %pi *( n - U )) ;

end

h ( n)=h_hamm ( n ) * hd ( n ) ;

end

[ hzm , fr ]= frmag ( h ,256) ;

hzm_dB = 20* log10 ( hzm ) ./ max ( hzm );

figure

plot (2* fr , hzm_dB )

a = gca () ;

xlabel ( 'Frequency w*pi') ;

ylabel ( 'Magnitude in dB') ;

title ('Frequency Response of FIR HRF with N=11 Using Hamming Window') ;

xgrid (2) ;

```

Design an LPF filter with

$$H_d e^{j\omega} = e^{-j3w} \text{ for } -\pi/4 \leq \omega \leq \pi$$

= 0 for $\pi/4 \leq |\omega| \leq \pi$. Find the values of $h(n)$ for $N=7$. Find $H(z)$. Plot the magnitude response using Hamming window and Hanning window techniques.

Scilab code for both Hanning and Hamming window Techniques

```
clear all;
clc ;
close ;
N =7;
alpha =3;
U =1;
h_hann = window ( ' hn ' ,N ) ; // (h_hamm = window ( 'hm' ,N ) ; ) for Hamming window
for n =0+ U :1:6+ U
if n ==4
hd ( n ) =0.25;
else
hd ( n ) =( sin ( %pi *( n -U - alpha ) /4 ) ) /(%pi *( n -U - alpha ) );
end
h ( n ) =hd ( n ) * h_hann ( n ) ; //(h ( n ) =h_hamm ( n ) * hd ( n ) ; ) For Hamming window
end
[ hzm , fr ]= frmag ( h ,256) ;
hzm_dB = 20* log10 ( hzm ) ./ max ( hzm );
figure
plot (2* fr , hzm_dB )
a = gca () ;
xlabel ( 'Frequency w * pi' ) ;
ylabel ('Mangnitude in dB');
```

```
title ('Frequency response of given LPF with N=7') ;
```

```
xgrid (2) ;
```

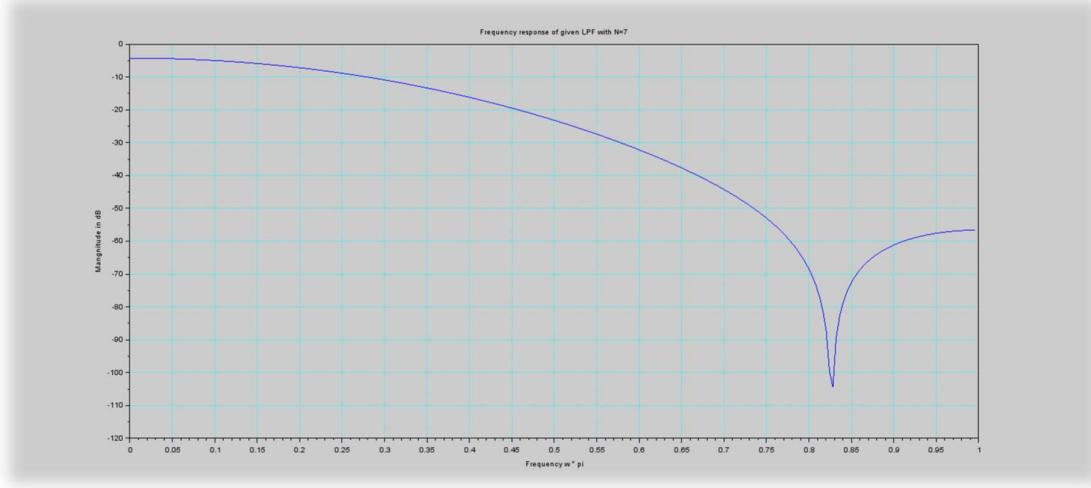


Fig.2: Magnitude response of LPF filter using Hamming Window

Pre-lab: Design of digital FIR Band Pass Filter using Hanning and Hamming window

Post-lab: Design of digital FIR Band Stop filter using Hanning and Hamming window

Lab 9: Design of digital FIR filter using frequency sampling method

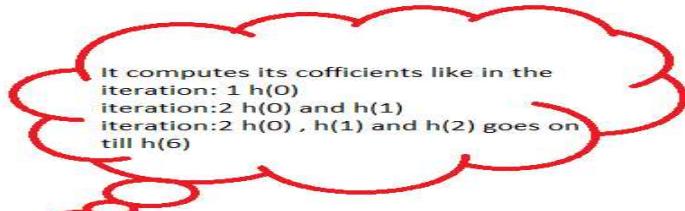
Problem. 1: Determine the filter coefficients $h(n)$ obtained by sampling

$$H_d e^{j\omega} = e^{-j(N-1)\omega/2} \text{ for } 0 \leq |\omega| \leq \pi/2$$

$$= 0 \text{ for } \pi/2 \leq |\omega| \leq \pi. \text{ For } N=7.$$

```
clear ;
clc ;
close ;
N=7;
U=1; // Zero Adjust
for n=0+U :1: N-1+U
h ( n ) =(1+2* cos (2* %pi *( n -U -3) /7) ) / N
end
disp (h , " Filter Coefficients , h(n)="" )
```

```
--> clear ;
--> N =7;
--> U =1; // Z e ro A dj u s t
--> for n =0+ U :1: N -1+ U
> h ( n ) =(1+2* cos (2* %pi *( n -U -3) /7) ) / N
h =
-0.1145625
h =
-0.1145625
0.0792797
h =
-0.1145625
0.0792797
0.3209971
h =
-0.1145625
0.0792797
0.3209971
0.4285714
```



It computes its coefficients like in the
iteration: 1 h(0)
iteration:2 h(0) and h(1)
iteration:2 h(0), h(1) and h(2) goes on
till h(6)

```
--> disp (h , "Filter Coefficients, h(n)="" )
```

```
-0.1145625
0.0792797
0.3209971
0.4285714
0.3209971
0.0792797
-0.1145625
```


BY using "disp" comment we get all the
7 coefficients alone shown in the
console.

```
"Filter Coefficients, h(n)=""
```

Pre-lab Question

1. Find the values of $h(n)$ for $N=11$. (For problem.1)

Post-lab question

1. What is the general formula to find the values of $H(k)$, $\theta(k)$ and $h(n)$ using frequency sampling method?
2. List the steps in defining the “ k ” values of frequency sampling method.

Experiment 10(a) Design of Analog Butterworth filter

Aim: To Design and analyze the function of an analog Butterworth filter

Software Requirement: SCI Lab

Theory: The signal processing filter which is having a flat frequency response in the passband can be termed as Butterworth filter and is also called as a maximally flat magnitude filter. In 1930 physicist and the British engineer Stephen Butterworth described about a Butterworth filter in his “on the theory of filter amplifiers” paper for the first time. Hence, this type of filter named as Butterworth filter. There are various types of Butterworth filters such as low pass Butterworth filter and digital Butterworth filter. The frequency response of the nth order Butterworth filter is given as

$$H_{(j\omega)} = \frac{1}{\sqrt{1 + \epsilon^2 \left(\frac{\omega}{\omega_p}\right)^{2n}}}$$

Where ‘n’ indicates the filter order, ‘ ω ’ = $2\pi f$, Epsilon ϵ is maximum pass band gain, (Amax).

Algorithm:

Step 1: For the given Cutoff frequency, band ripple and stopband ripple find the order of the filter

Step 2: Round the order of the filter to the next higher order integer

Step 3: Determine the magnitude and phase response of the analog Butterworth filter to Plot

Program:

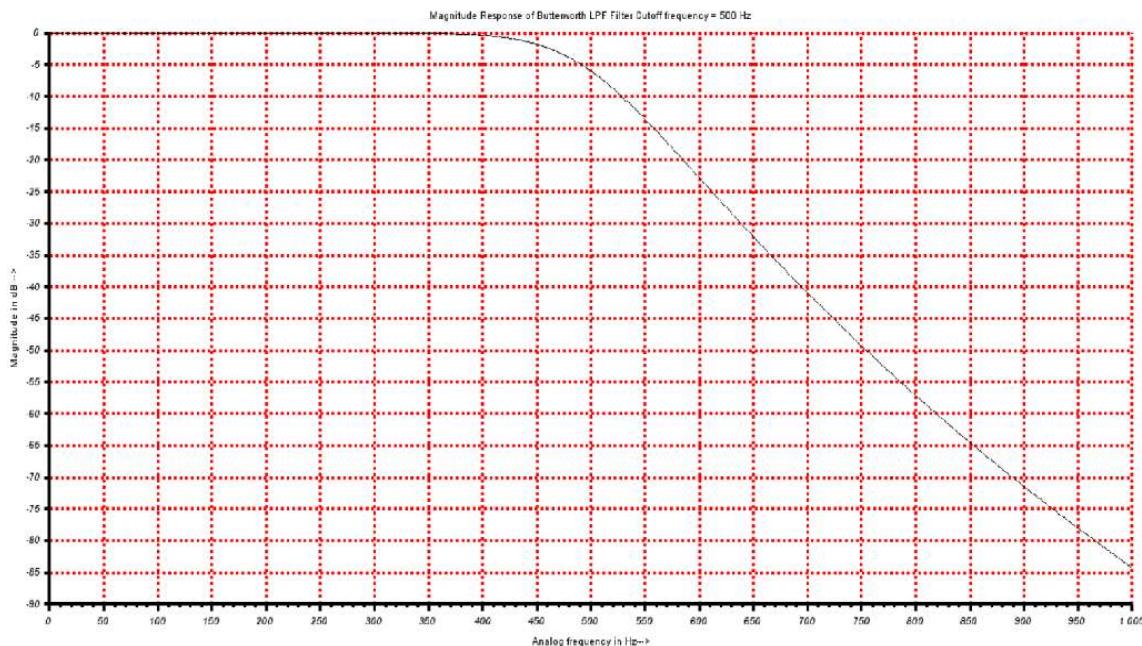
```
// To Design an Analog Butterworth Filter
//For the given cutoff frequency and filter order
//Wc = 500 Hz
omegap = 500; //pass band edge frequency
omegas = 1000;//stop band edge frequency
delta1_in_dB = -3;//PassBand Ripple in dB
delta2_in_dB = -40;//StopBand Ripple in dB
delta1 = 10^(delta1_in_dB/20)
```

```

delta2 = 10^(delta2_in_dB/20)
//Calculation of filter order
N = log10((1/(delta2^2))-1)/(2*log10(omegas/omegap))
N = ceil(N) //Rounding off nearest integer
omegac = omegap;
h=buttmag(N,omegac,1:1000); //Analog Butterworth filter magnitude response
mag=20*log10(h); //Magnitude Response in dB
plot2d((1:1000),mag,[0,-180,1000,20]);
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(5)
xtitle('Magnitude Response of Butterworth LPF Filter Cutoff frequency = 500 Hz','Analog frequency in Hz-->','Magnitude in dB -->');

```

Simulation Output:



Experiment 10(b) Design of Analog Chebyshev filter

Aim: To Design and analyze the function of an analog Chebyshev filter

Software Requirement: SCI Lab

Theory: Chebyshev filters are used for distinct frequencies of one band from another. They cannot match the windows-sink filter's performance and they are suitable for many applications. The main feature of Chebyshev filter is their speed, normally faster than the windowed-sinc. Because these filters are carried out by recursion rather than convolution. The designing of the Chebyshev and Windowed-Sinc filters depends on a mathematical technique called as the Z-transform. Types of Chebyshev Filters

Chebyshev filters are classified into two types, namely type-I Chebyshev filter and type-II Chebyshev filter.

Type-I Chebyshev Filters

This type of filter is the basic type of Chebyshev filter. The amplitude or the gain response is an angular frequency function of the nth order of the LPF (low pass filter) is equal to the total value of the transfer function $H_n(j\omega)$

$$G_n(\omega) = |H_n(j\omega)| = 1/\sqrt{1 + \epsilon^2 T_n^2(\omega/\omega_0)}$$

Where, ϵ = ripple factor

ω_0 = cutoff frequency

T_n = Chebyshev polynomial of the nth order

$$\begin{aligned} s_{pm}^{\pm} &= \pm \sinh \left(\frac{1}{n} \operatorname{arsinh} \left(\frac{1}{\epsilon} \right) \right) \sin(\theta_m) \\ &\quad + j \cosh \left(\frac{1}{n} \operatorname{arsinh} \left(\frac{1}{\epsilon} \right) \right) \cos(\theta_m) \end{aligned}$$

Here $m = 1, 2, 3, \dots, n$

$$\theta_m = \frac{\pi}{2} \frac{2m-1}{n}.$$

The above equation produces the poles of the gain G. For each pole, there is the complex conjugate, & for each and every pair of conjugate there are two more negatives of the pair. The TF should be stable, transfer function (TF) is given by

$$H(s) = \frac{1}{2^{n-1} \epsilon} \prod_{m=1}^n \frac{1}{(s - s_{pm}^-)}$$

Algorithm:

Step 1: Convert the given analog frequencies to digital domain

Step 2: Prewarp the frequency components and find the prewarping frequency

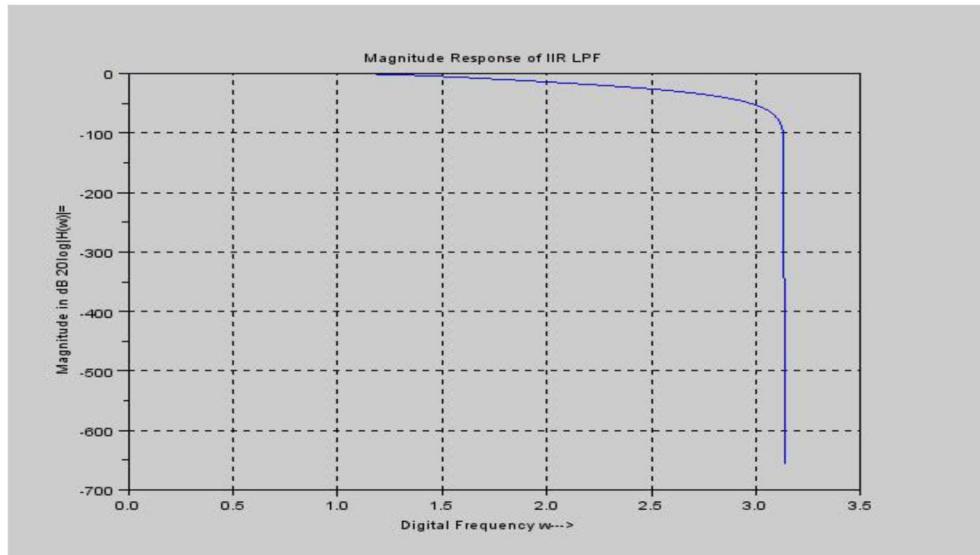
Step 3: Determine the order of the filter and round to next higher order integer

Step 4: Convert the analog transfer function to digital by using Bilinear transformation technique and plot the magnitude and frequency response

Program:

```
// Capt ion : To o b t a i n D i g i t a l I R Chebyshev low pa s s
f i l t e r
// Fr equency r e s p o n s e
clc ;
clear ;
close ;
fp= input (' Enter the pass band edge (Hz ) = ' );
fs= input (' Enter the stop band edge (Hz ) = ' );
kp= input (' Enter the pass band attenuation (dB) = ' );
ks= input (' Enter the stop band attenuation (dB) = ' );
Fs= input (' Enter the sampling rate samples / s e c = ' );
d1 = 10^( kp /20 ) ;
d2 = 10^( ks /20 ) ;
d = sqrt ((1/( d2 ^2)) -1);
E = sqrt ((1/( d1 ^2)) -1);
// D i g i t a l f i l t e r s p e c i f i c a t i o n s ( rad / s a m p l e s )
wp =2* %pi *fp *1/ Fs;
ws =2* %pi *fs *1/ Fs;
// Pre warping
op =2* Fs* tan (wp /2);
os =2* Fs* tan (ws /2);
N = acosh (d/E)/ acosh (os/op);
oc = op /(( E ^2) ^1/(2* N)));
N = ceil (N); // rounded to n e a r e s t i n t e g e r
disp (N, ' I IR F i l t e r o r d e r N =' );
disp (oc , ' Cu t o ff Fr equency i n rad / s e c o n d s OC = ' )
[pols ,gn] = zpch1 (N,E,op);
HS = poly (gn , ' s ', ' c o e f f ')/ real ( poly (pols , ' s '));
z = poly(0,'z')
Hz = horner (HS ,(2* Fs *(z -1) /(z +1) ))
num = coeff (Hz (2) )
den = coeff (Hz (3) )
Hz (2)= Hz (2) ./ den (3) ;
Hz (3)= Hz (3) ./ den (3) ;
disp (Hz , ' Tr a n s f e r f u n c t i o n o f D i g i t a l I R Chebyshev LPF H(Z)= ' )
[Hw ,w] = frmag (Hz ,256) ;
figure (1)
plot (2*w*%pi ,20* log10 ( abs (Hw)));
xlabel ( ' D i g i t a l Fr equency W>' )
ylabel ( 'M a g n i t u d e i n d B 20 l o g jH(w) j= ' )
title ( 'M a g n i t u d e Re s p o n s e o f I IR LPF ' )
xgrid (1)
```

Simulation Output



Prelab Questions:

1. List the differences between butterworth and Chebyshev filter
2. Define the term Prewarping and mention its importance
3. To design a Discrete time Low pass filter the specifications are
Passband $F_p = 4$ KHz with 0.8 dB ripple
Stopband $F_p = 4.5$ KHz with 50 dB attenuation
Sampling frequency $F_s = 22$ KHz
(i) Determine the passband and stopband frequencies (ii) Determine the maximum and minimum values of $|H(\omega)|$ in the pass band and stopband

Postlab Questions:

1. List the properties of Chebyshev filter
2. In your CD the data is sampled at 44.1kHz, and we want to have a good sound quality upto 21kHz. If you had to use an analog butterworth filter as reconstruction filter, What would be the order of the filter?
3. Give the Normalized Low pass Butterworth Denominator polynomials for N=8,9,10.