

18ECC204J DIGITAL SIGNAL PROCESSING LABORATORY

ACADEMIC YEAR: 2021-2022

NAME: Pushpal Das

REG.NO: RA1911004010565



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under SECTION 3 of the UGC Act, 1956)
S.R.M. NAGAR, KATTANKULATHUR – 603203.
KANCHEEPURAM DISTRICT
DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING

FACULTY OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY



BONAFIDE CERTIFICATE

Register No.:RA1911004010565

Certified to be the Bonafide record of work done by PUSHPAL DAS of 3rd YEAR ECE
B.Tech. Degree course in the Practical 18ECC204J – DIGITAL SIGNAL
PROCESSING in SRM Institute of Science and Technology, Kattankulathur during the
academic year 2021-2022

Mrs.D.VIJAYALAKSHMI
Staff In-Charge

Date:

Head of the Department

Submitted for University Examination held on 19/11/2021 at SRM Institute of
Science and Technology, Kattankulathur.

Date:

Internal Examiner I

Internal Examiner II

Name: Pushpal Das

Class: ECE- I

Reg.No: RA1911004010565

Branch: ECE

Exp. No.	Date of Performance	Title of Experiment	Marks Obtained (30)
1	16-07-2021	Generation of Basic Signals	27
2	26-07-2021	Generation of Continuous Time and Discrete Signal	23
3	02-08-2021	Sampling Theorem	25
4	09-08-2021	Generation of Basic Signals	27
5	16-08-2021	(a) Autocorrelation & cross correlation (b) Spectrum Analysis using DFT	27
6	24-08-2021	Efficient Computation of DFT and IDFT using FFT	22
7	31-08-2021	Design of digital FIR Low Pass, High Pass, Band Pass, Band Stop Filter using Rectangular Window	27
8	07-09-2021	Design of digital FIR Low Pass and High Pass Filter using Hanning and Hamming Window	26
9	15-09-2021	Design of digital FIR filter using Frequency Sampling Method	25
10	21-09-2021	(a) Design of Analog Butterworth Filter (b) Design of Analog Chebyshev Filter	22
11	27-09-2021	(a) Design of digital Butterworth filter using bilinear transformation (b) Design of digital butterworth filter using impulse invariance method	28
12	12-10-2021	To design Chebyshev filter using Bilinear and Impulse Invariance Method	27
13	12-10-2021	(a) Interpolation in time domain (b) Interpolation in frequency domain	27
14	12-10-2021	(a) Decimation in time domain (b) Decimation in frequency domain	27
15	12-10-2021	(a) Design of anti-aliasing filter (b) Design of anti-imaging filter	30

16	20-10-2021	Application of Sound Effect on a audio file	
----	------------	--	--

Laboratory Report Cover Sheet

SRM Institute of Science and Technology
College of Engineering and Technology
Department of Electronics and Communication Engineering

**18ECC204J DIGITAL SIGNAL
PROCESSING** Fifth Semester, 2021-22 (Odd semester)

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4th/I

Venue : TP916-DSP Lab

Title of Experiment : 1.Generation of Basic Signals .

Date of Conduction : 16 JULY 2021

Date of Submission : 21 JULY 2021

Particulars	Max. Marks	Marks Obtained
Pre lab and Post lab	10	
Lab Performance	10	
Simulation and results	10	
Total	30	

REPORT VERIFICATION

Staff Name : Mrs.D.VIJAYALAKSHMI , A.ANILETBALA

Signature :

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 wave ' ) ;
10 xlabel ( ' t ' ) ;
11 ylabel ( ' x ' ) ;
Simulation Output:
```

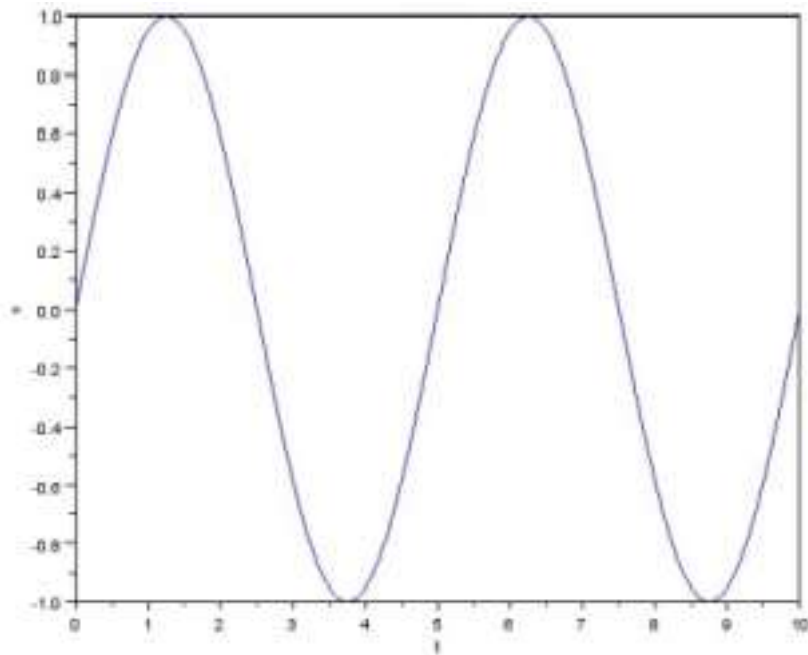


Figure 1.1: sinewave

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:

```

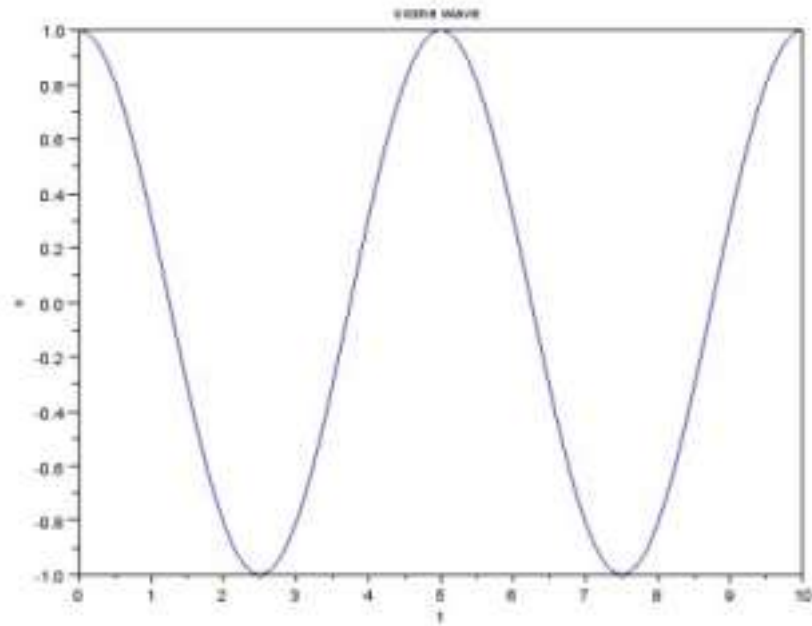


Figure 1.2: cosine wave

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 wave ' ) ;
12 xlabel ( ' t ' ) ;

```


13 ylabel (' y ') ;

Simulation Output:

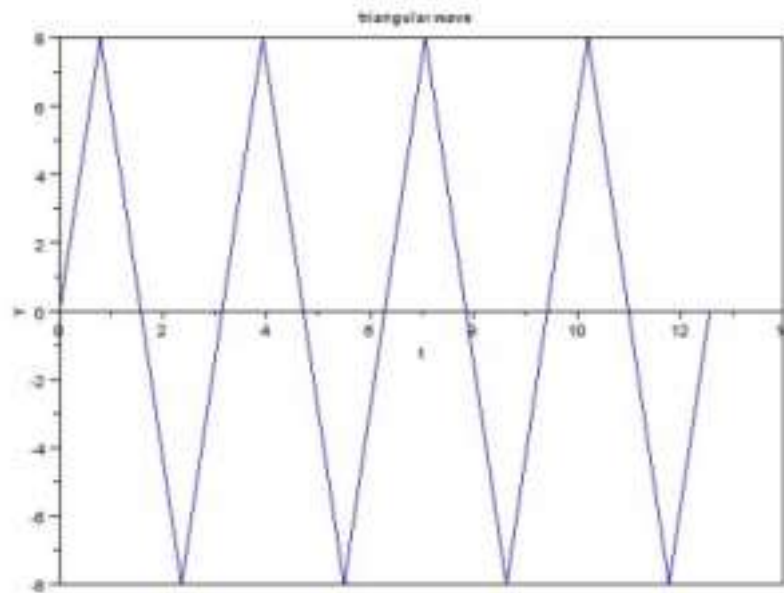


Figure 1.3: triangular wave

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:

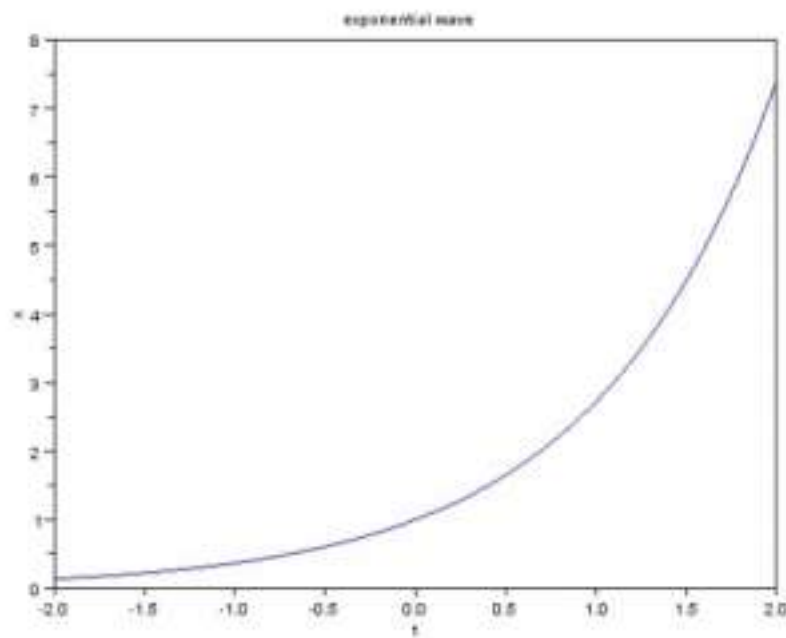


Figure 14: Exponential wave

1. b) Generation of Discrete Signals

Scilab code: Unit impulse signal

```

1 clc ;
2 clf ;
3 clear all;
4 // u n i t I 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 =" m i d d l e "
10 plot2d3 (n ,x ) ;
11 title ( ' u n i t I m p u l s e ' ) ;
Simulation Output:

```

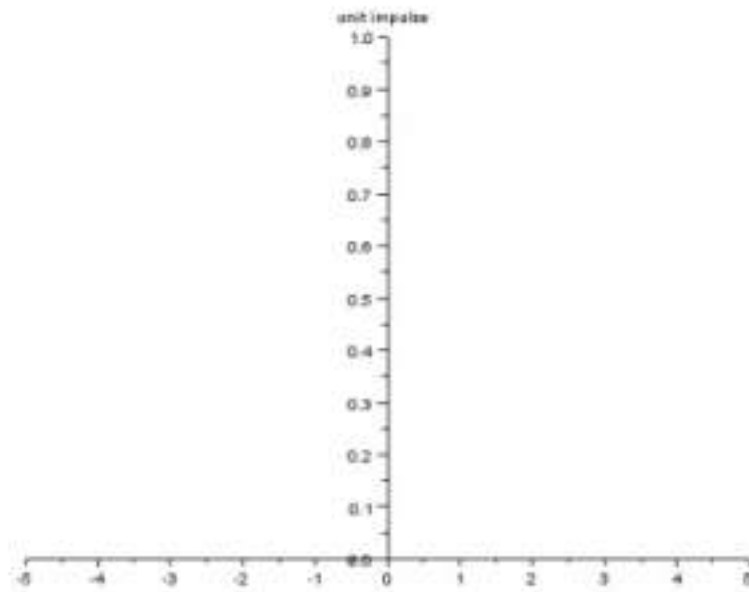


Figure 1.5: unit impulse signal

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:

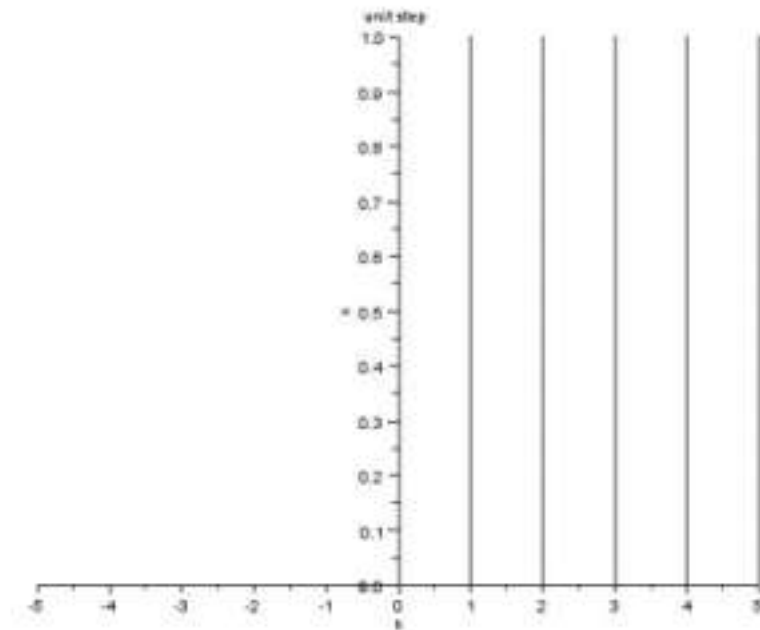


Figure 1.6 : unitstepsignal

Scilab code: Unit ramp signal

```

1 // u n i t ramp
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 = ' m i d d l e ' ;
10 plot2d3 (n ,x ) ;
11 xtitle ( ' u n i t ramp s i g n a l ' ) ;
12 xlabel ( ' --->n ' ) ;
13 ylabel ( ' --->x ( n ) ' ) ;

```

Simulation Output:

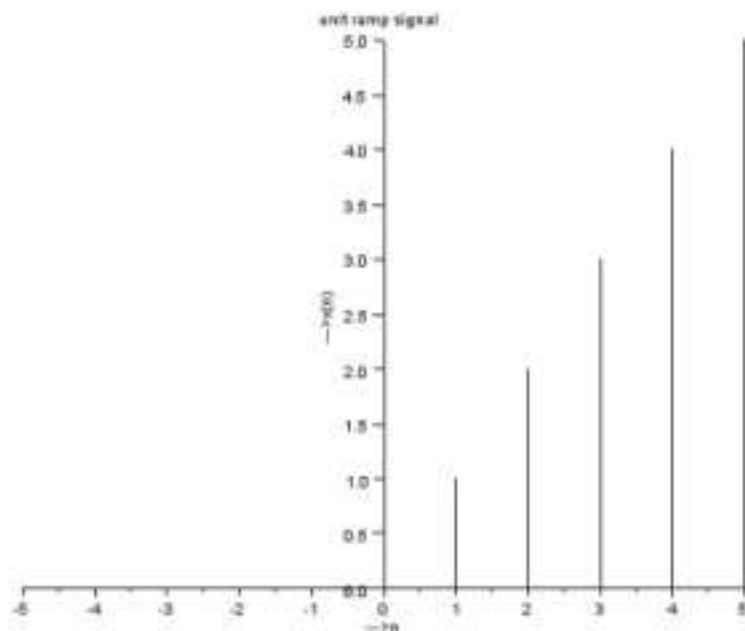
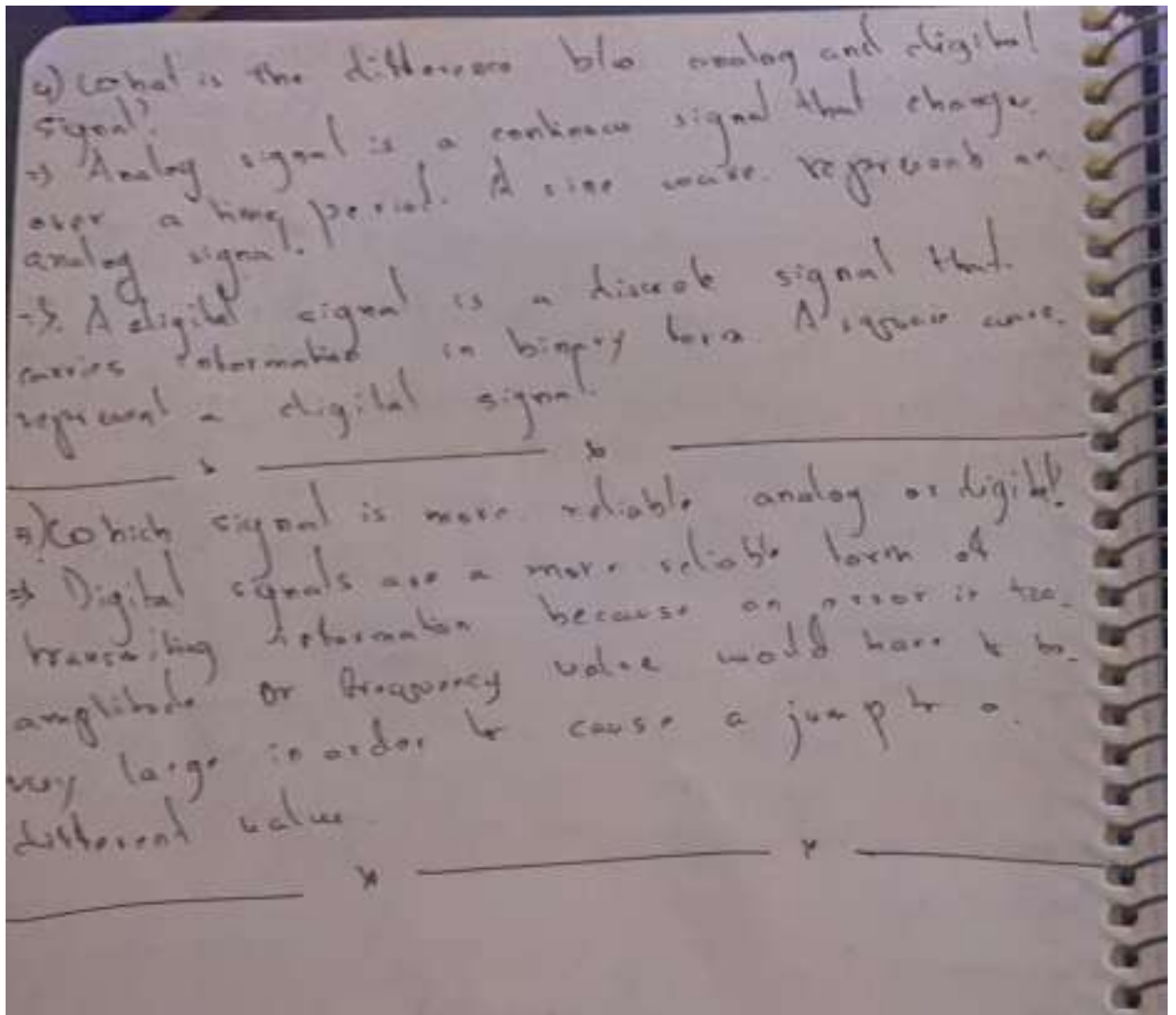


Figure1.7 : unit ramp

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.

Scilab Output:-

Result: Thus we generate and obtain the output for the basic signals

SRM Institute of Science and Technology
Faculty of Engineering and Technology
Department of Electronics and Communication Engineering

**18ECC204J DIGITAL SIGNAL
PROCESSING** Fifth Semester, 2020-21 (Odd semester)

Name :Pushpal Das

Register No. :RA1911004010565

Day / Session : 4/FN

Venue : Online (G Meet)

**Title of Experiment : Generation of Continuous Time and
Discrete Time Signal**

Date of Conduction :26 JULY 2021

Date of Submission : 01 AUG 2021

Particulars Max. Marks	Marks Obtained
Pre lab and Post lab 1 0	
Lab Performance 1 0	
Simulation and results 1 0	
Total 30	

REPORT VERIFICATION

Staff Name : Mrs. Vijayalakshmi D

Signature :

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

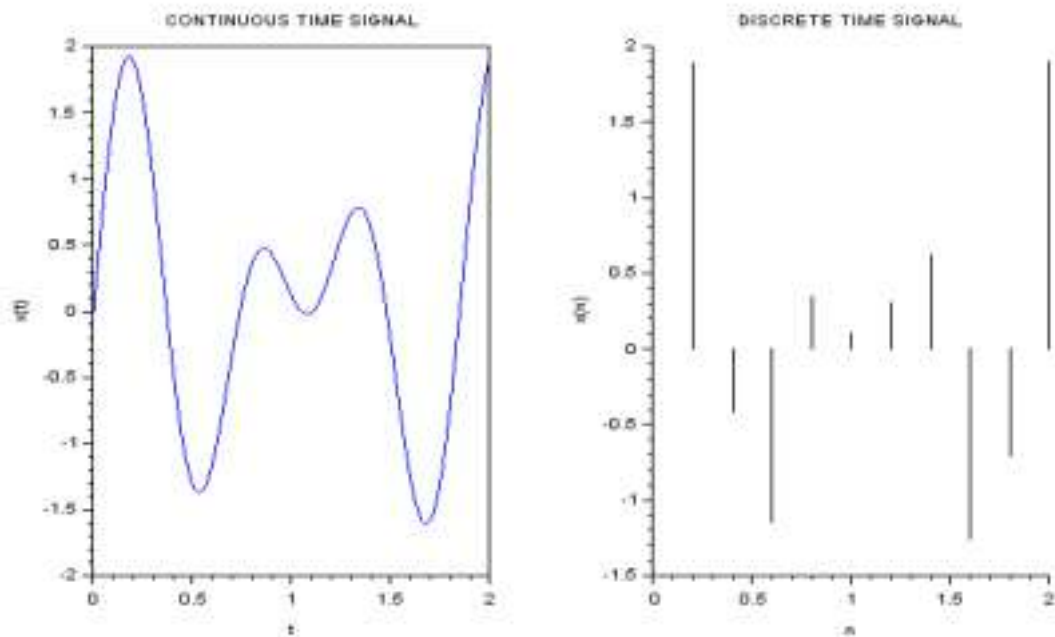
S.No.	Discrete-time signal	Digital signal
1	The discrete-time signal is a digital representation of a continuous-time signal.	The digital signal is a form of discrete-time signal.
2	The discrete -time signal can be obtained from the continuous-time signal by the Euler's method.	The digital signal can 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

Experiment-2

RA1911064010565

Pushpal Das

Pre lab

1) Define continuous time signal.

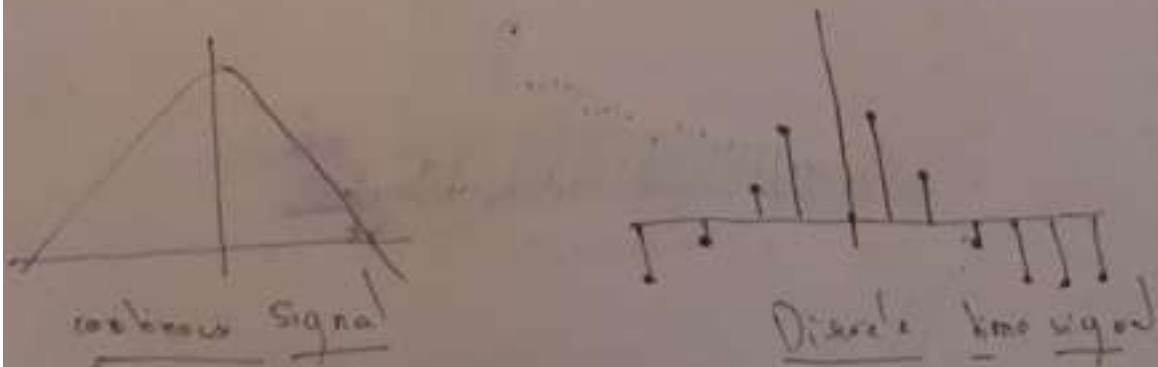
→ A continuous-time signal is a function $s(t)$ that is defined for all time t contained in some interval on the real line.

2) CT signals are often called analog signals.

3) How is discrete time signal generated?

→ Discrete time signal, used in digital signal processing, can be obtained by sampling and quantization of continuous signals. Continuous signal may also be defined over an independent variable other than time.

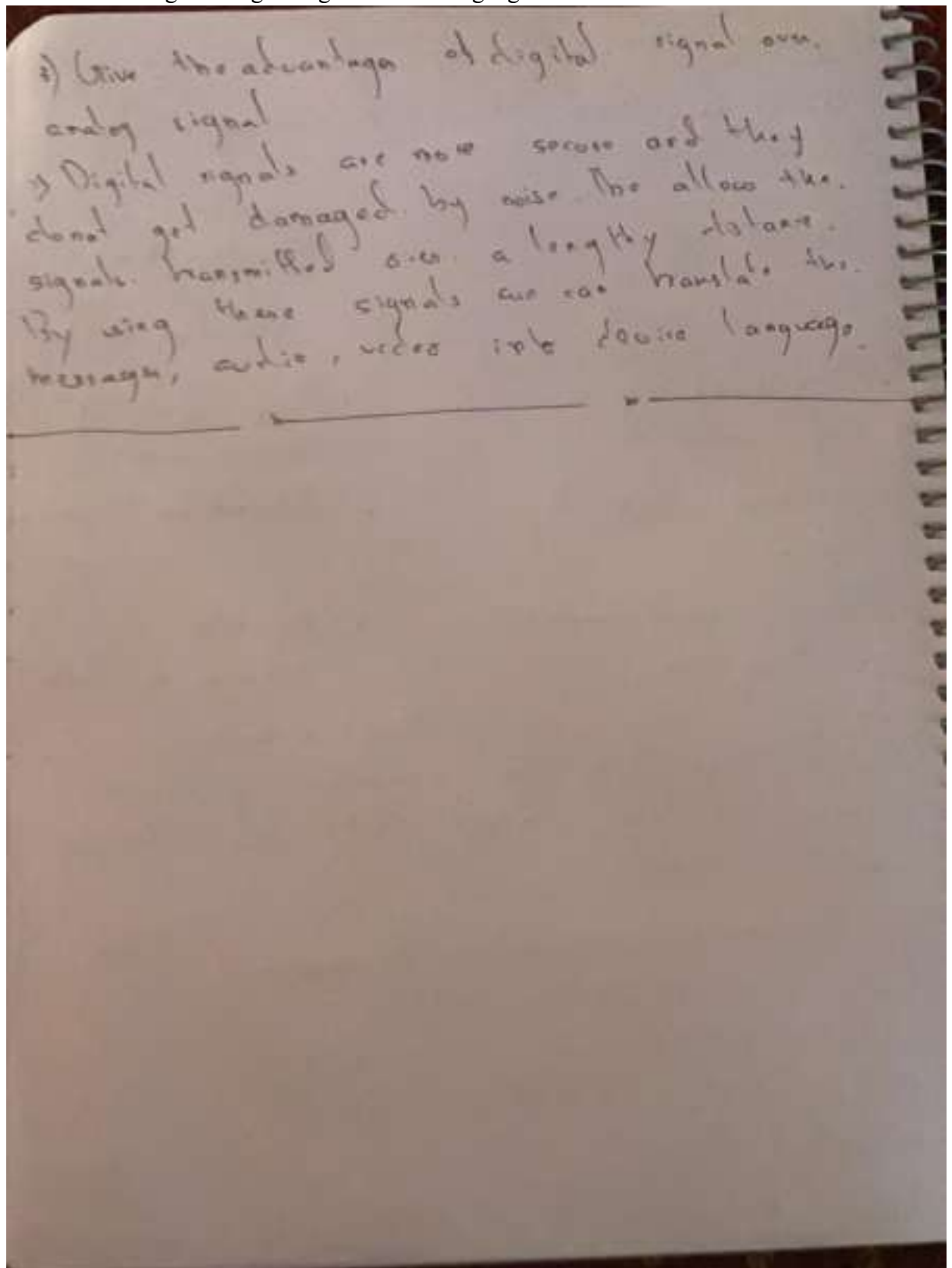
4) Draw the graphical representation of continuous time signal and discrete time signal.



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: Thus, we generated the continuous time signal and discrete time signal using SCI lab was implemented Successfully.

SRM Institute of Science and Technology
Faculty of Engineering and Technology
Department of Electronics and Communication Engineering

**18ECC204J DIGITAL SIGNAL
PROCESSING Fifth Semester, 2020-21 (Odd semester)**

Name : Pushpal Das

Register No. : RA191100401565

Day / Session : 4/3&4

Venue : Online

Title of Experiment : a) Sampling Theorem b) Aliasing and its Effects

Date of Conduction : 26/7/21

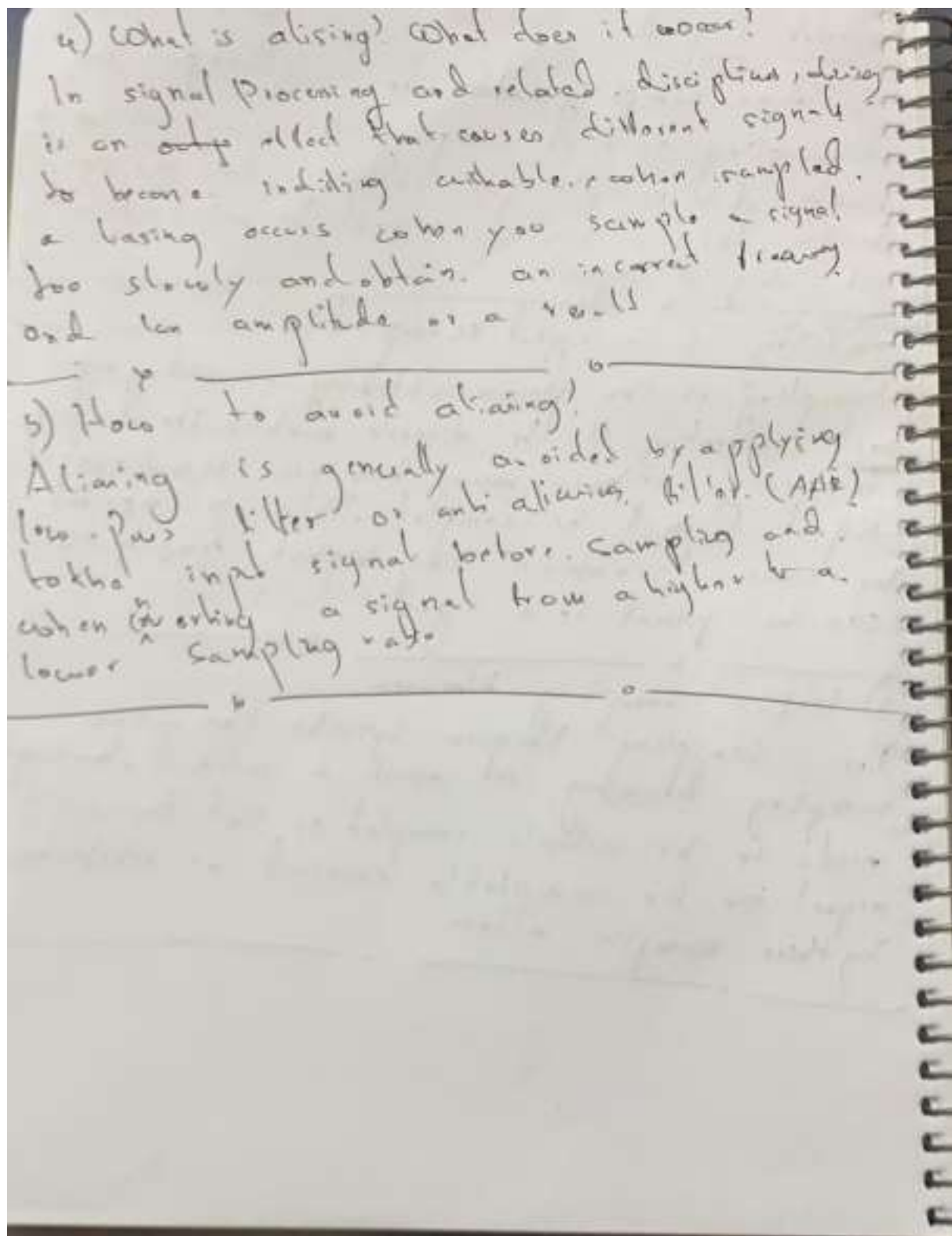
Date of Submission : 15/8/21

Particulars Max. Marks	Marks Obtained
Pre lab and Post lab 1 0	
Lab Performance 1 0	
Simulation and results 1 0	
Total 30	

REPORT VERIFICATION

Staff Name : Dr. C.Vimala

Signature :
PRELAB-



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.

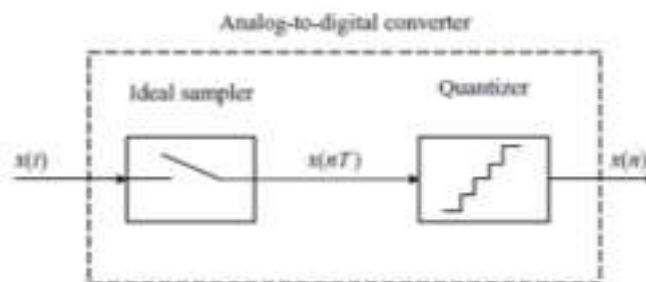


Fig:-1 Block diagram of an ADC

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.

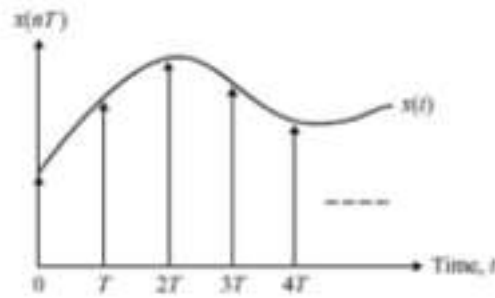


Fig:-2 Digitization of analog signal $x(t)$ into discrete-time signal $x(nT)$

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.

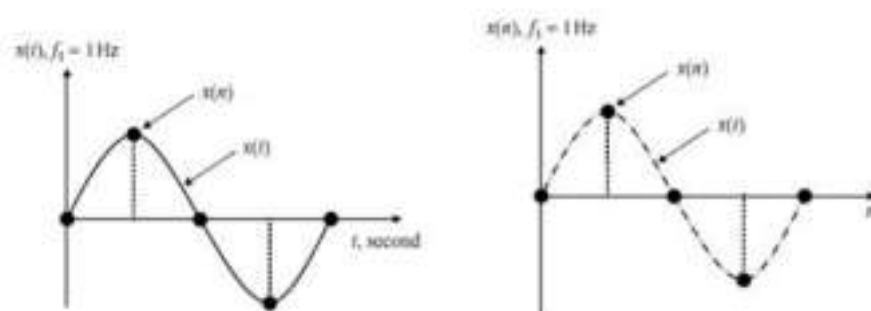


Fig:-3 Sampling of an analog signal $x(t)$ into discrete-time signal $x(nT)$ (a) and its exact reconstruction (b)

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.

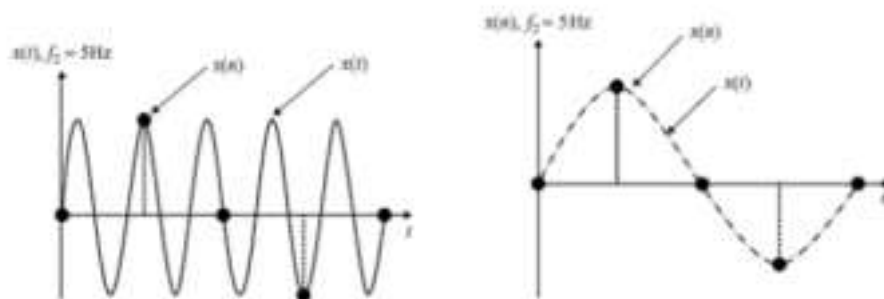


Fig:-4 Sampling of an analog signal $x(t)$ into discrete-time signal $x(nT)$ (a) and its inaccurate reconstruction (b)

PROGRAM

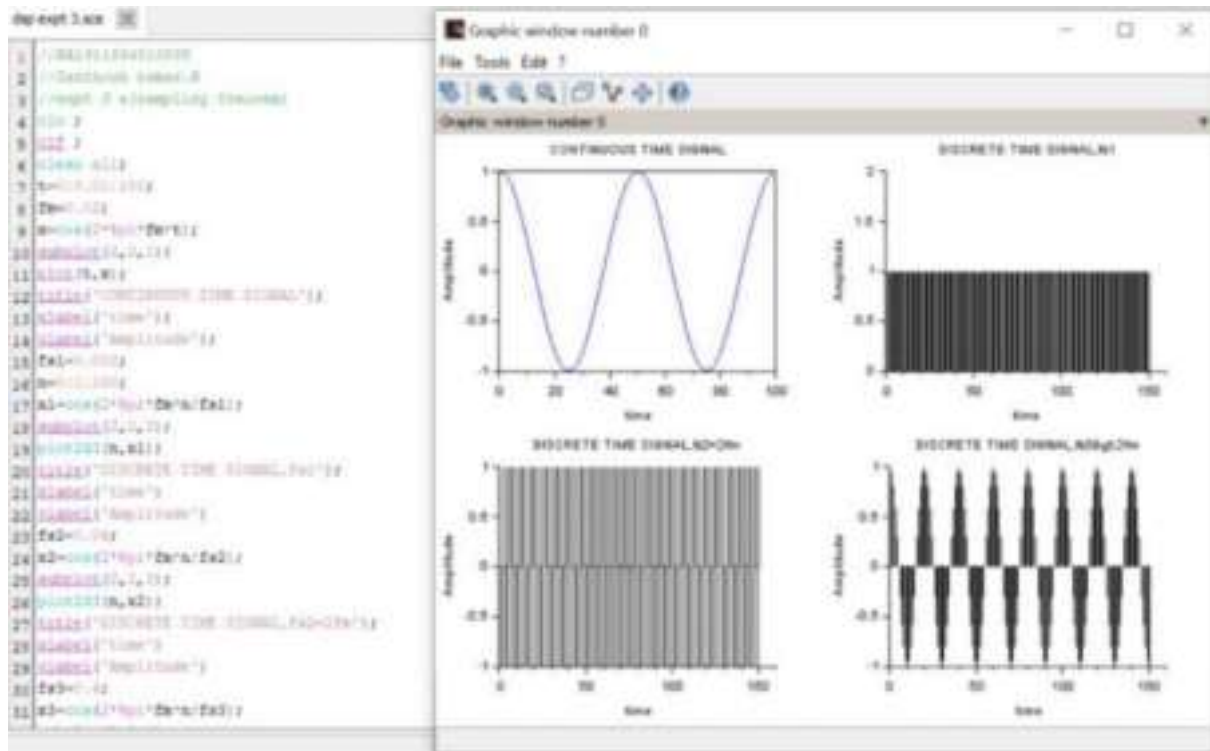
```
// PROGRAM FOR SAMPLING
// CONVERSION OF ANALOG SIGNAL TO DISCRETE SIGNAL
clc ;
clf ;
clear all;
t = 0:0.01:100;
fm = 0.02;
x = cos (2* %pi * fm * t);
subplot (2 ,2 ,1) ;
plot (t ,x ) ;
title ( 'CONTINUOUS TIME SIGNAL' ) ;
xlabel('time');
ylabel('Amplitude');
fs1 = 0.002;
n = 0:1:150;
x1 = cos (2* %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')
```

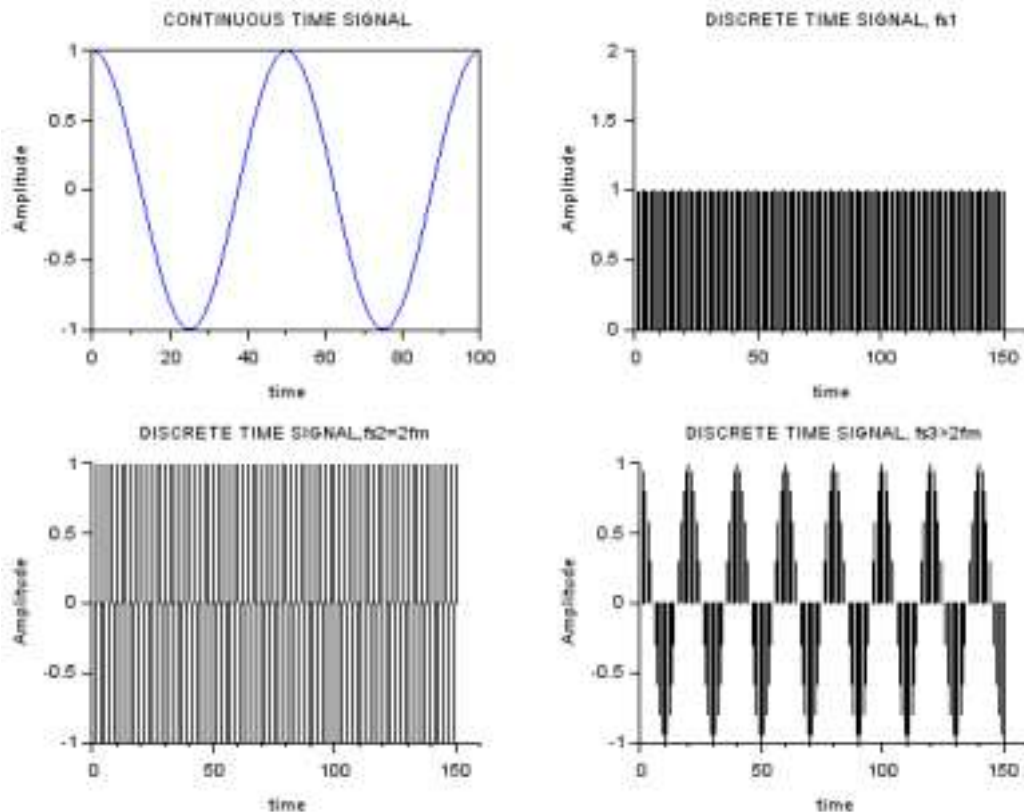
```

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.

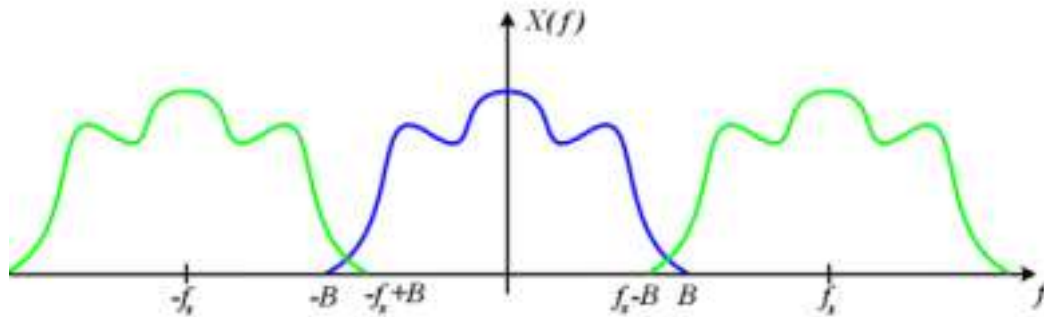
Sampling Rate and Nyquist Frequency Limit

- 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.



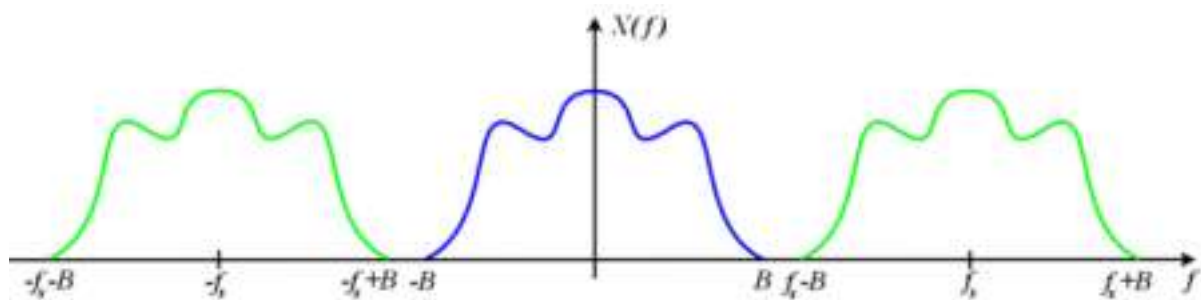
The blue sampled signal is insufficiently bandlimited. The overlapping edges of the green images are added and creating a spectrum

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, sl)
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 -5.0000000"

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/dt;
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"

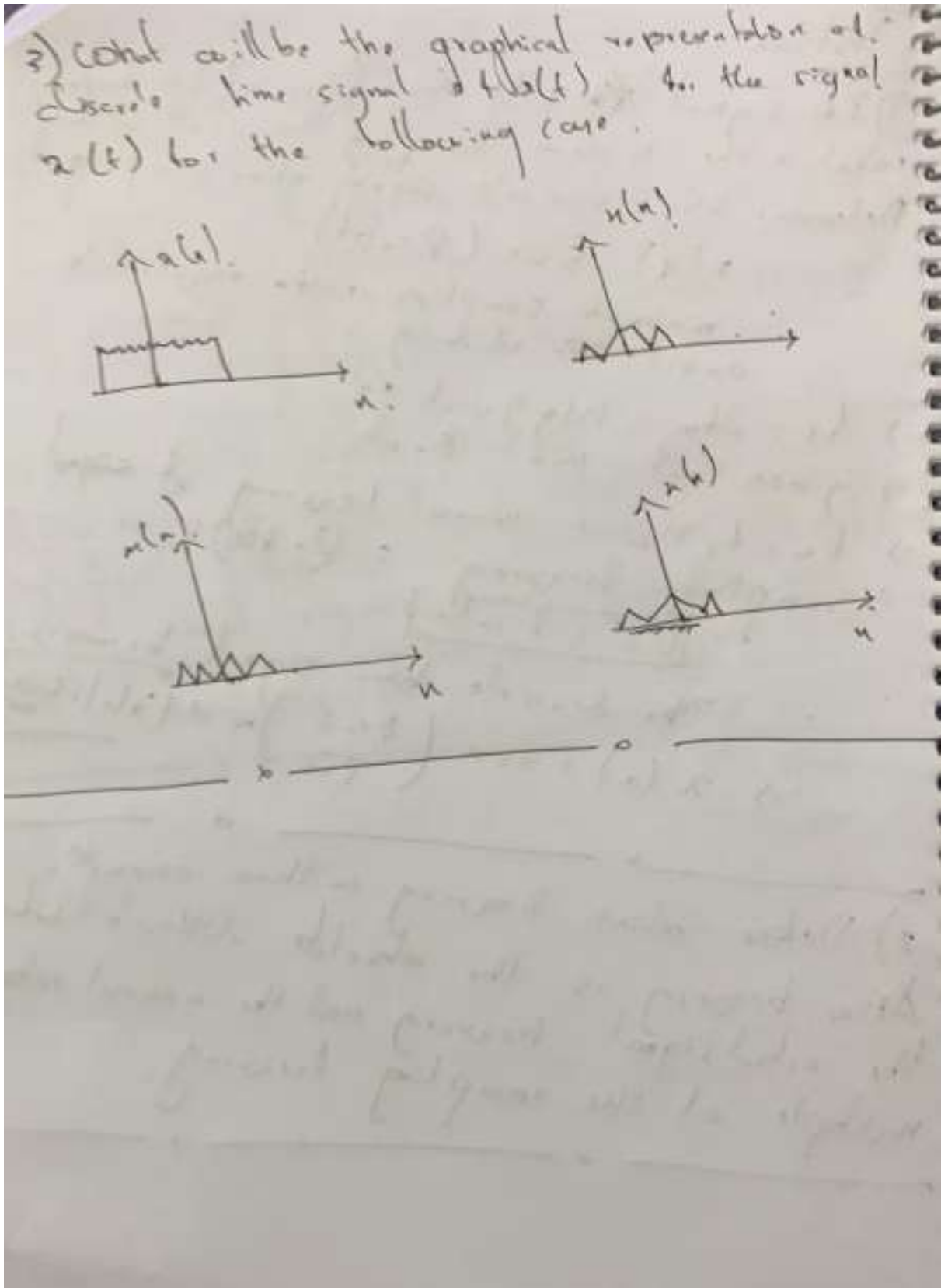
"Aliased frequencies: 5000 -5000 Hz."

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: Thus, the sampling theorem as well as the aliasing techniques are designed, simulated and verified using scilab

Laboratory Report Cover Sheet

SRM Institute of Science and Technology College of Engineering and Technology Department of Electronics and Communication Engineering
18ECC204J DIGITAL SIGNAL PROCESSING Fifth Semester, 2021-22 (Odd semester)

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4TH/FN

Venue : Online (G-meet)

Title of Experiment : Linear Convolution and Circular Convolution

Date of Conduction : 09 AUG 2021

Date of Submission : 23 AUG 2021

Particulars	Max. Marks	Marks Obtained
Pre lab and Post lab	10	
Lab Performance	10	
Simulation and results	10	
Total	30	

REPORT VERIFICATION

**Staff Name : Mrs.D.Vijayalakshmi
Mrs.A.Aniletbala**

Signature :

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:

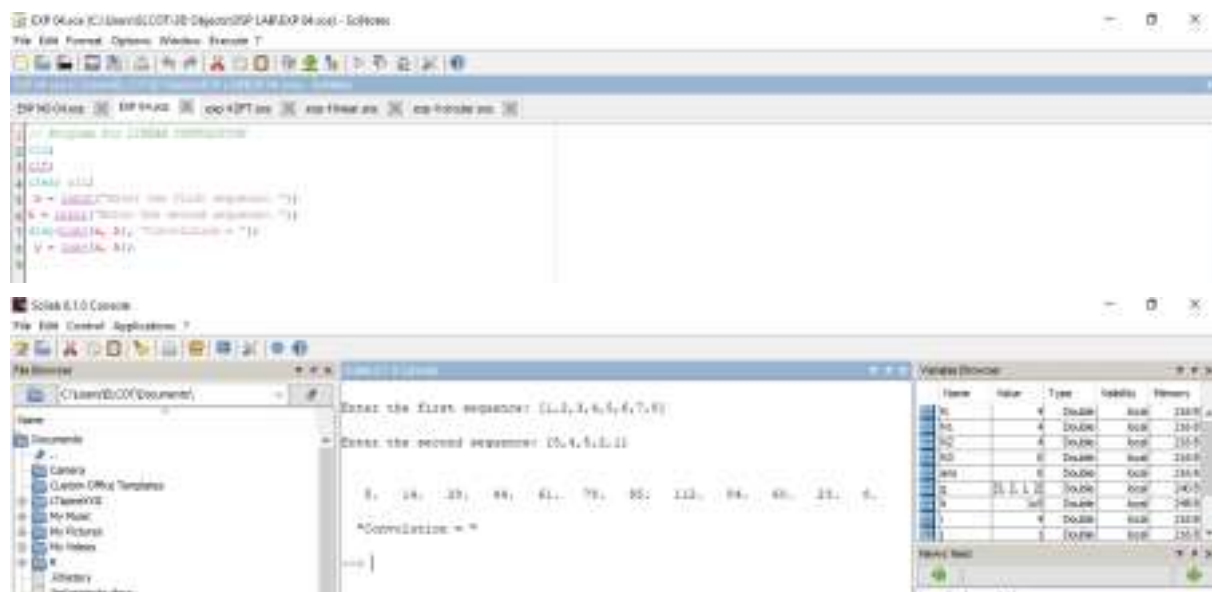
Enter the first sequence: [1 2 3 4 5 6 7 8]

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

"Convolution = "

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

CODE & SIMULATION:-



// 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:

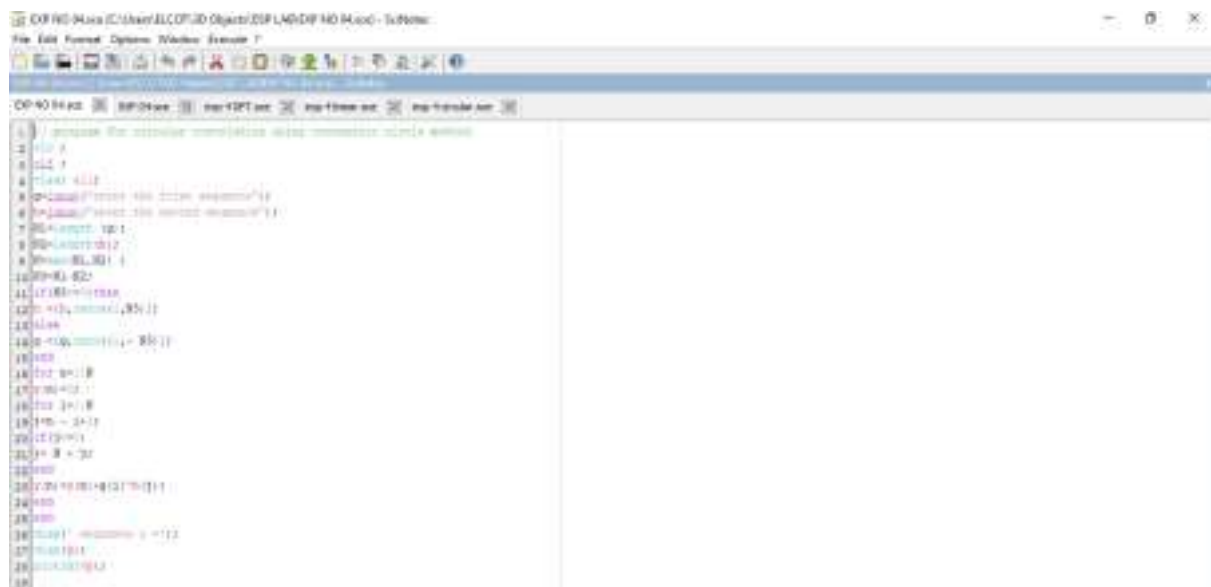
enter the first sequence[2 1 2 1]

enter the second sequence[1 2 3 4]

" sequence y ="

14. 16. 14. 16.

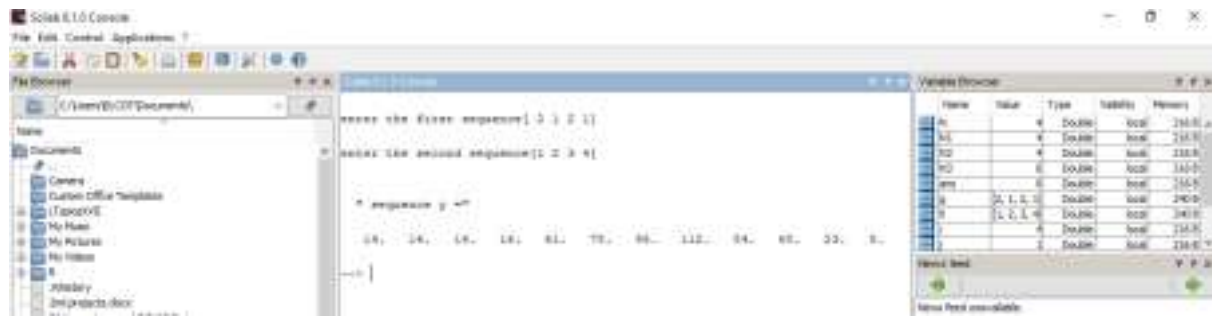
CODE & SIMULATION:-



```

1 // generate the discrete correlation using cross-correlation function
2
3 % Define the first sequence g
4 g = [2 1 2 1];
5
6 % Define the second sequence h
7 h = [1 2 3 4];
8
9 % Calculate the cross-correlation sequence y
10 y = conv(g, h, 'full');
11
12 % Display the result
13 disp(' sequence y =');
14 disp(y);
15
16 % Plot the sequence y
17 plot2d3(y);
18
19 % End of script
20

```



// Program for Circular convolution using DFT computation

clc ;

close ;

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

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

//DFT Compu tation

X1 =fft(x1,-1);

X2=fft(x2,-1);

X3=X1 .* X2 ;

//IDFT Computation

x3 =fft(X3 ,1);

// Display sequence x3 [n] i n command window

disp(x3);

Output

14. 16. 14. 16.

CODE & SIMULATION:-

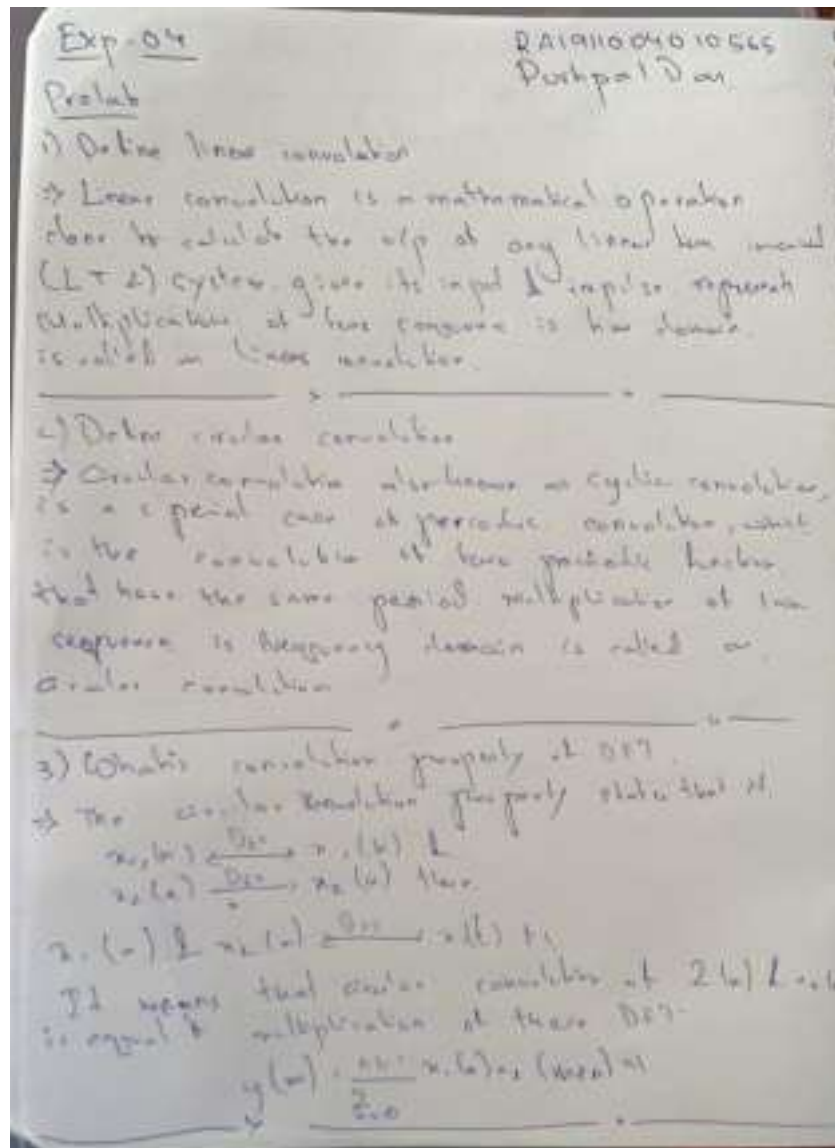




Pre lab

1. Define linear convolution
2. Define circular convolution
3. what is convolution property of DFT

PRE LAB:-



Post lab

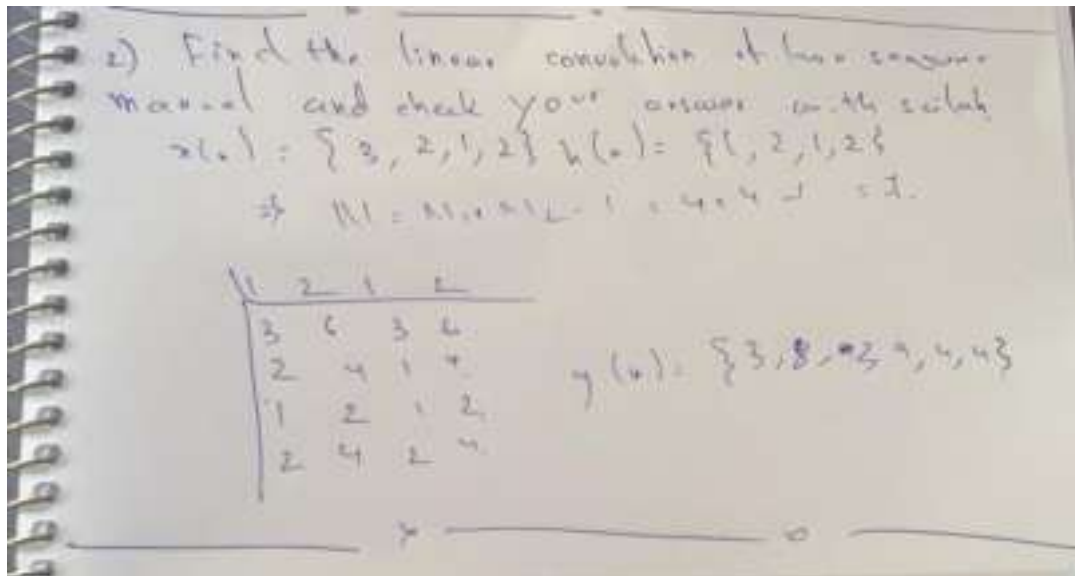
1. Find the linear convolution of two sequence manual and check your answer with scilab
 $x(n) = \{3, -1, 0, 1, 3, 2, 0, 1, 2, 1\}$ $h(n) = \{1, 1, 1\}$

```
Scilab 6.1.0 Console
Enter the first sequence: [3, -1, 0, 1, 3, 2, 0, 1, 2, 1]
Enter the second sequence: [1, 1, 1]

3.  2.  2.  0.  4.  6.  5.  3.  3.  4.  3.  1.

"Convolution = "
```

2. Find the linear convolution of two sequence manual and check your answer with scilab
 $x(n) = \{3, 2, 1, 2\}$ $h(n) = \{1, 2, 1, 2\}$



```
EXP NO 04.sce  X  EXP 04.sce  X  exp 4 DFT.sce  X  exp 4 linear.sce  X

1 |clc;
2 |clf;
3 |clear all;
4 |x = input("Enter the first sequence: ");
5 |h = input("Enter the second sequence: ");
6 |disp(conv(x, h), "Convolution = ");
7 |y = conv(x, h);

Scilab 8.1.0 Console

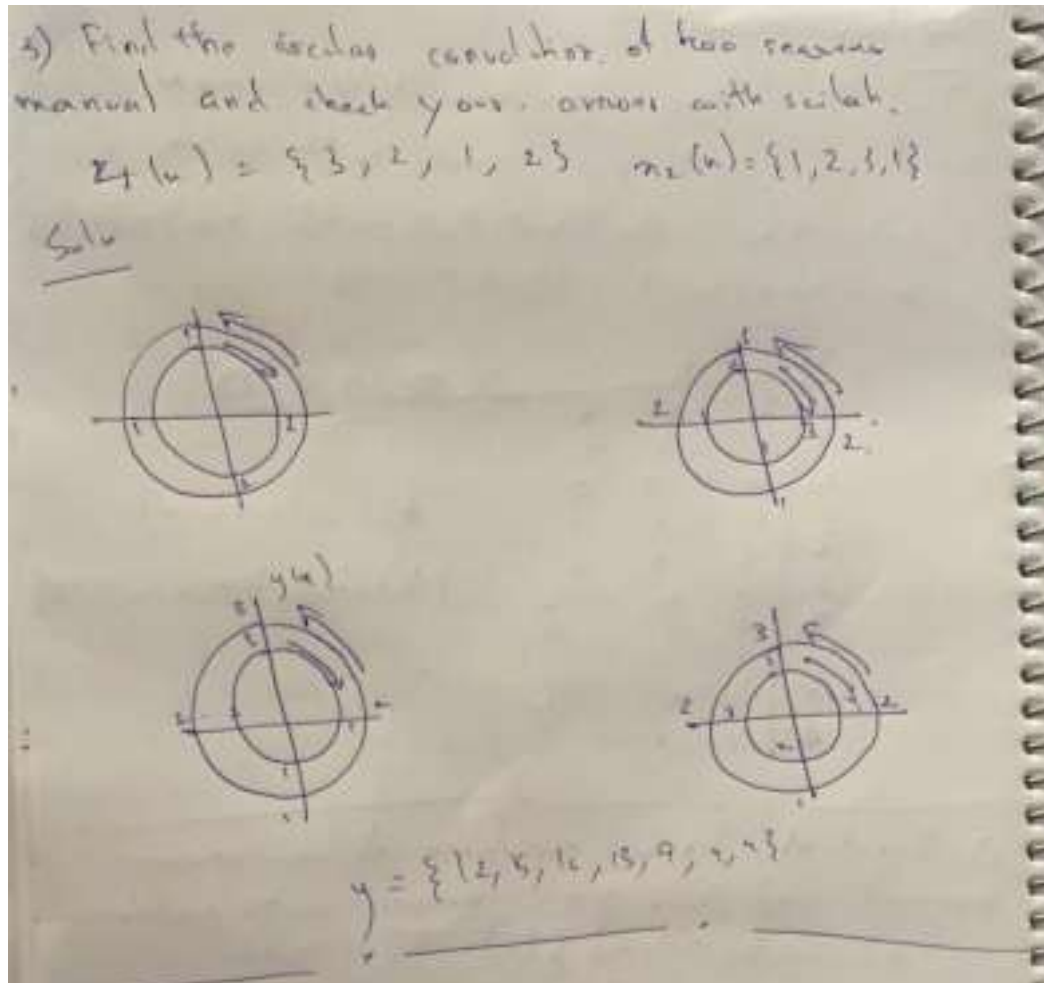
Enter the first sequence: [3,2,1,2]

Enter the second sequence: [1,2,1,2]

3.  8.  9.  12.  9.  4.  4.

"Convolution = "
```

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:

The linear convolution and circular convolution of two sequences using SCILAB was performed successfully.

Laboratory Report Cover Sheet

SRM Institute of Science and Technology
Faculty of Engineering and Technology
Department of Electronics and Communication Engineering

**18ECC204J DIGITAL SIGNAL
PROCESSING Fifth Semester, 2021-22 (Odd
semester)**

Name : Pushpal Das

Register No : RA1911004010565

Day / Session : 4Th /FN

Venue : ONLINE

**Title of Experiment : a) AUTOCORRELATION AND CROSS CORRELATION
b) SPECTRUM ANALYSIS USING DFT**

Date of Conduction : 09/08/2021

Date of Submission : 23/08/2021

| Particulars | Max. Marks | Marks
Obtained |
|------------------------|-------------------|---------------------------|
| Pre lab and Post lab | 1
0 | |
| Lab Performance | 1
0 | |
| Simulation and results | 1
0 | |
| Total | 30 | |

REPORT VERIFICATION

Staff Name : Dr.C.VIMALA

Signature :

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
```

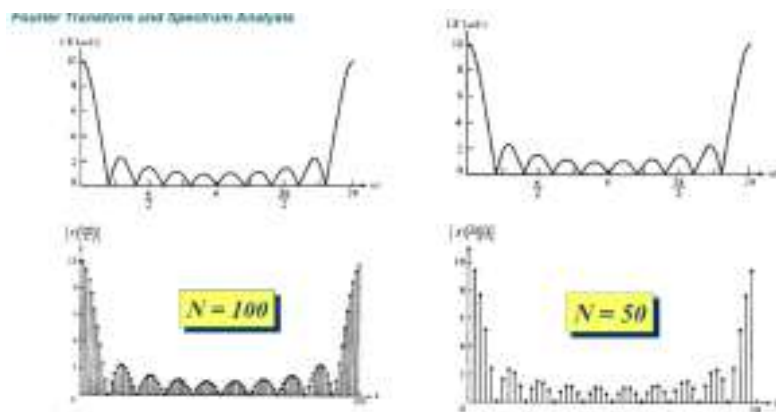
```
4. 11. 20. 30. 20. 11. 4.
```

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.

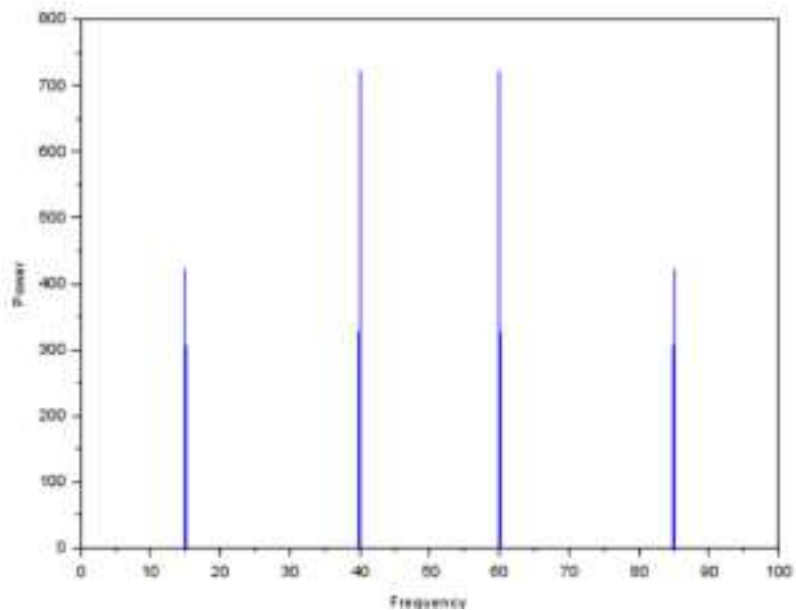


//Spectrum analysis using DFT

```
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.

Pre lab answers:-

Exp-05

RA19A1004010545

Pooja Patil

Probab

1) Define auto correlation.

→ Auto correlation is a mathematical representation of the degree of similarity between a given time series and a lagged version of itself over successive time intervals.

2) Define correlation.

→ Correlation is defined as the observed association between two variables. A correlation exists between two variables when one of them is related to the other in some way. A scatterplot is the best place to check.

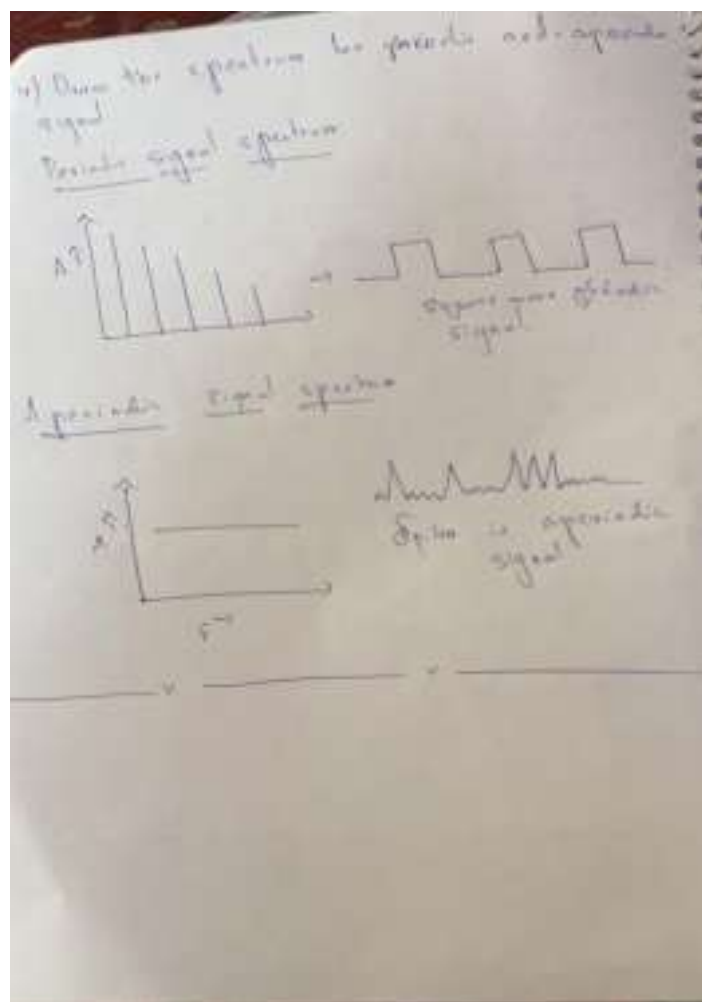
3) Give the properties of auto correlation.

→ The auto correlation is the symmetric measure.

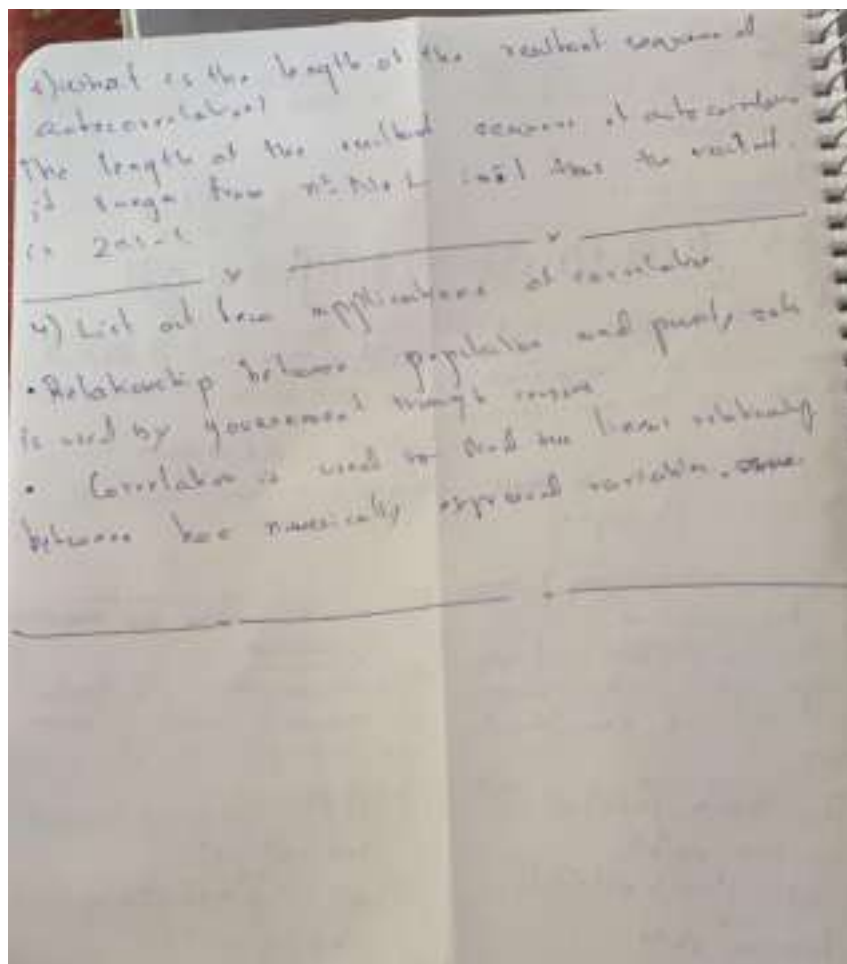
$$r_x(k) = r_x(-k)$$

→ Another property is that.

$$r_x(0) = 1 \text{ or } |r_x(k)| \leq 1$$



Post lab answers :-



Simulation results:-

a) CROSS CORRELATION :

```

EXP_5_DSP.sce
1 //RA1911004010590 - MOHANAPPRIYA K
2 //program for cross correlation
3 clc;
4 clear;
5 close;
6 x = input("Enter First Sequence:");
7 h = input("Enter Second Sequence:");
8 y = xcorr(x, h);
9 disp(y, "Cross correlation:");
10

```

AUTO CORRELATION :

Auto correlation.sce

```
1 //RA1911004010590 - MOHANAPPRIYA K
2 //program for Auto correlation
3 clc;
4 clear; close;
5 x = input("Enter First Sequence: ");
6 //h = input("Enter Second Sequence: ");
7 y = xcorr(x);
8 disp(y, "Auto correlation = ");
9
```

Scilab 5.1.0 Console

Enter First Sequence:[1 2 3 4]

Enter Second Sequence:[3 4 5 6]

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

"Cross correlation ="

--> |

Scilab 5.1.0 Console

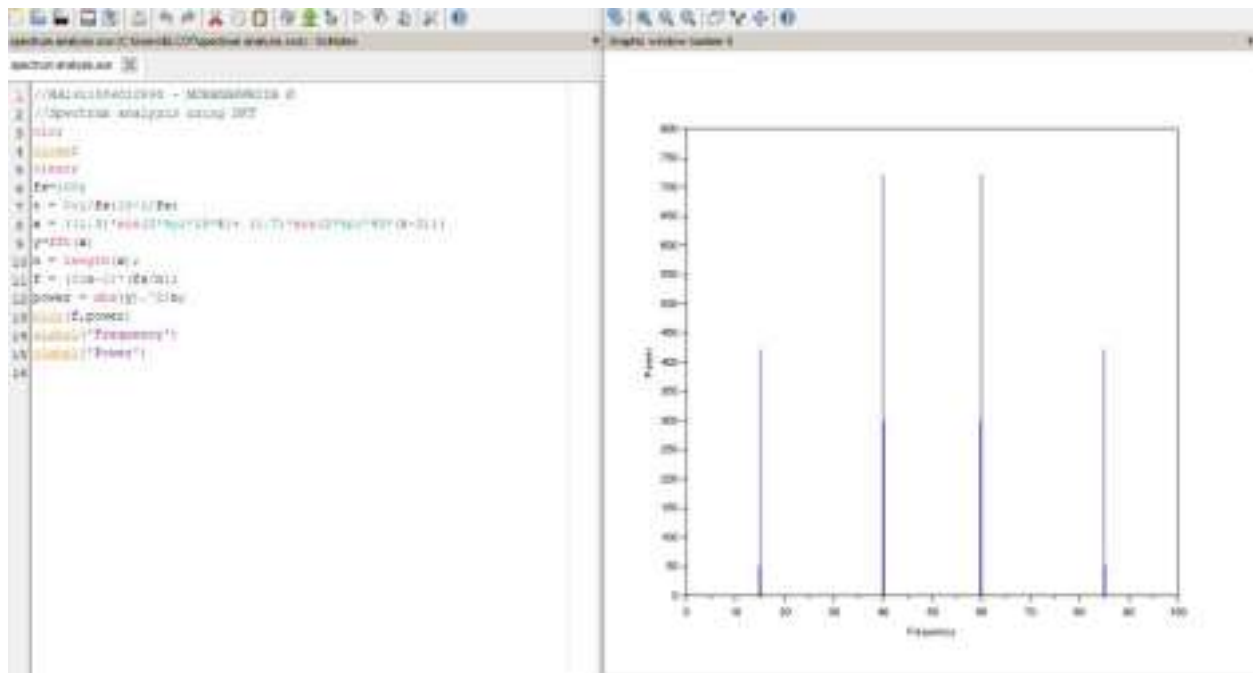
Enter First Sequence: [1 2 3 4]

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

"Auto correlation ="

--> |

b) SPECTRUM ANALYSIS :



RESULT :-

Thus, correlation and spectrum analysis using DFT are verified using scilab.

SRM
Institute
of
Science
and Technology
Faculty of Engineering and Technology
Department of Electronics and Communication Engineering

18ECC204J DIGITAL SIGNAL PROCESSING
Fifth Semester, 2020-21 (Odd semester)

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4th / FN

Venue : Online(G meet)

Title of Experiment : 6.Efficient computation of DFT and IDFT using FFT

Date of Conduction : 24/08/2021

Date of Submission : 20/09/2021

| Particulars | Max. Marks | Marks
Obtained |
|------------------------|------------|-------------------|
| Pre lab and Post lab | 10 | |
| Lab Performance | 10 | |
| Simulation and results | 10 | |
| Total | 30 | |

REPORT VERIFICATION

Staff Name : Mrs.D.VIJAYALAKSHMI
Mrs.A.ANILETBALA

Signature :

Lab 6: Efficient computation of DFT and IDFT using FFT

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

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

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

samples $X(k)$

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

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, Reg no and name' ) ; 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

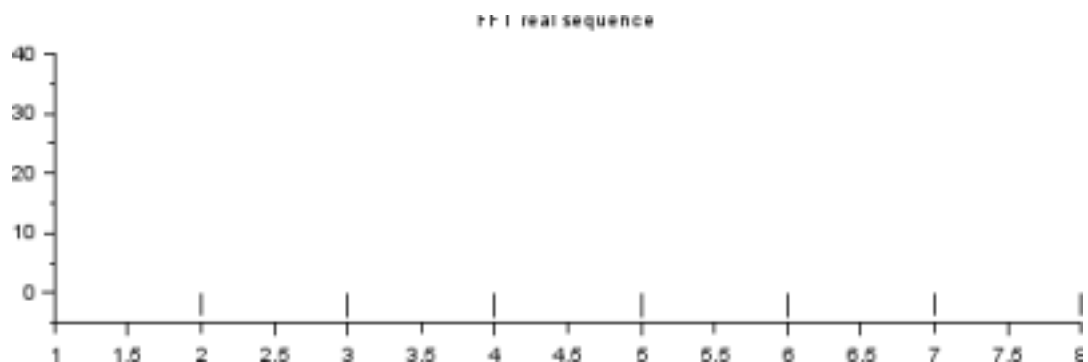
10.452504

"the phase response is"

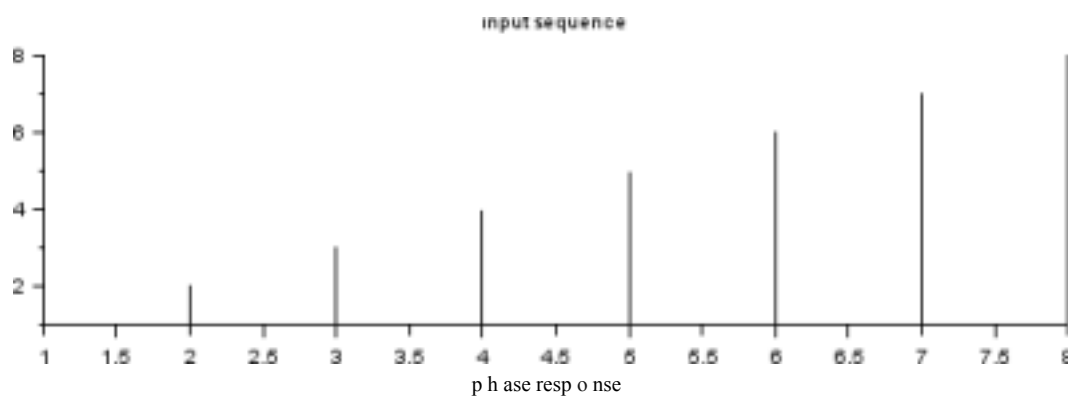
0. 112.5 135. 157.5 180. -157.5 -135. -112.5

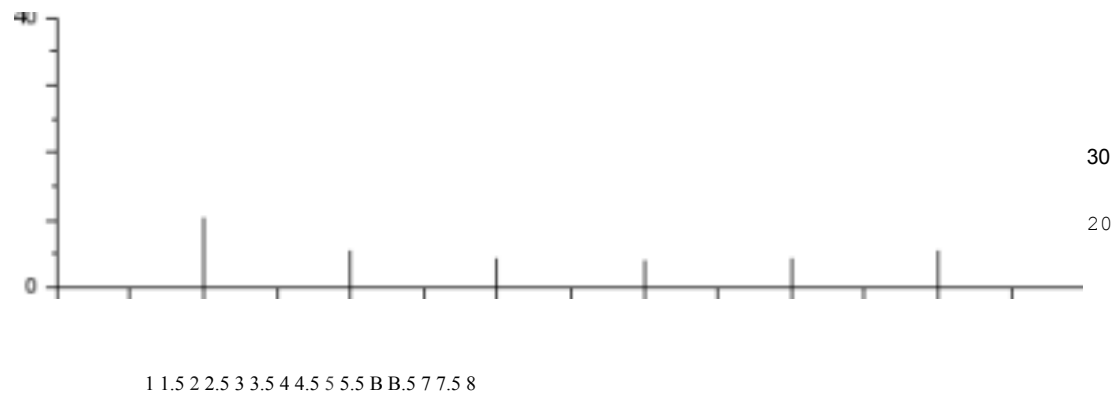
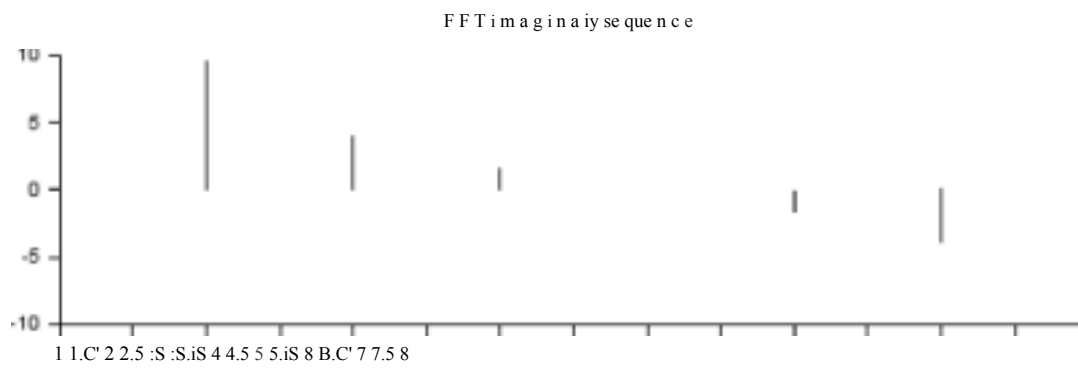
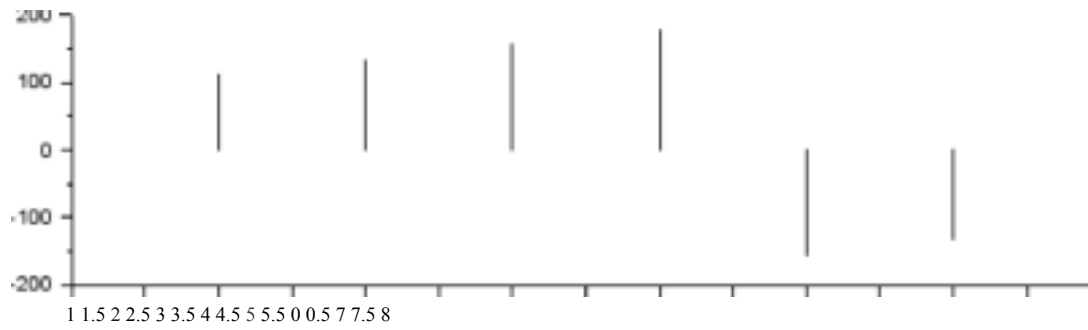
"the resultant IFFT sequence is"

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



:



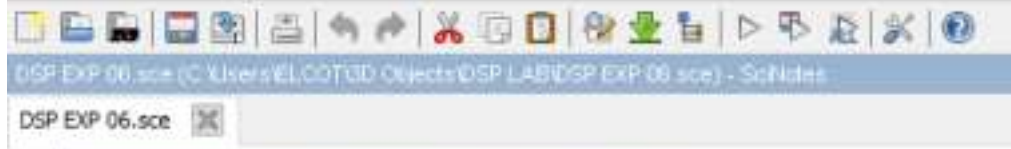


1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8

SCI LAB CODE:-

DSP EXP 06.sce (C:\Users\ELCOT\3D Objects\DSP LAB\DSP EXP 06.sce) - SciNotes

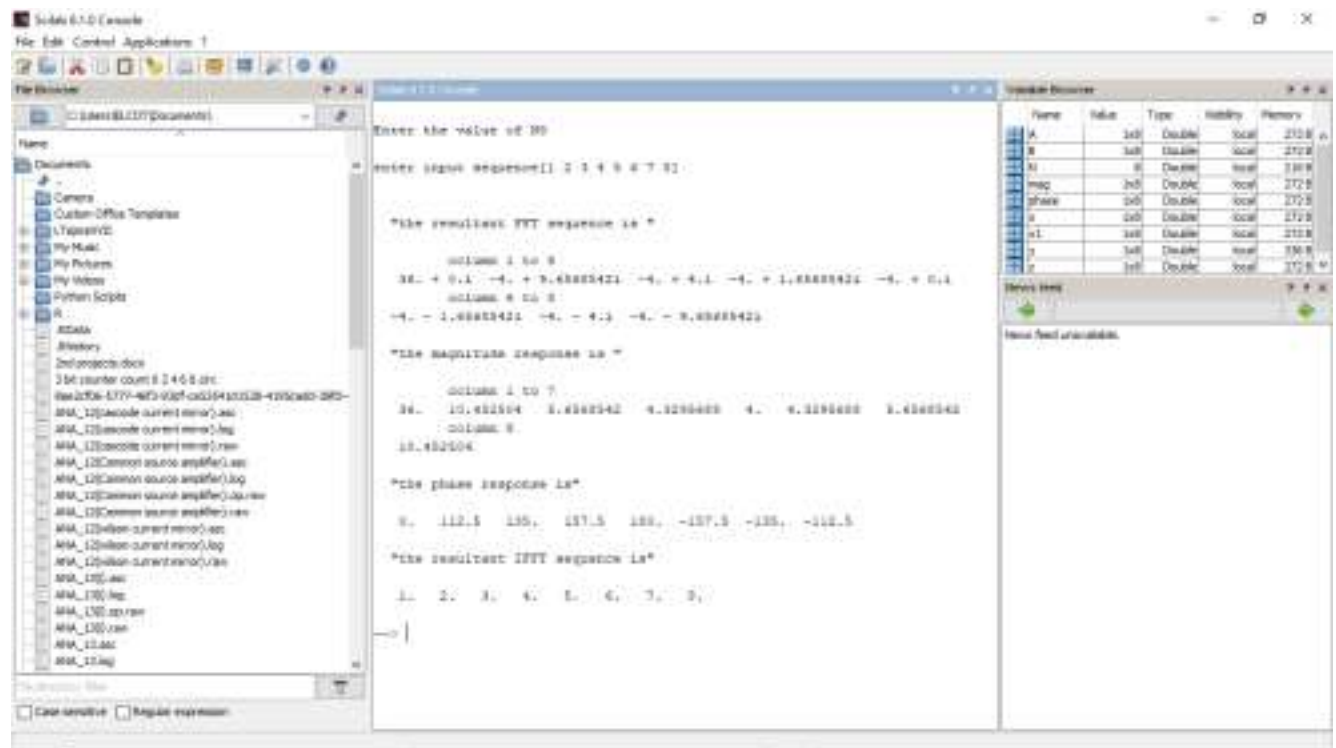
File Edit Format Options Window Execute ?



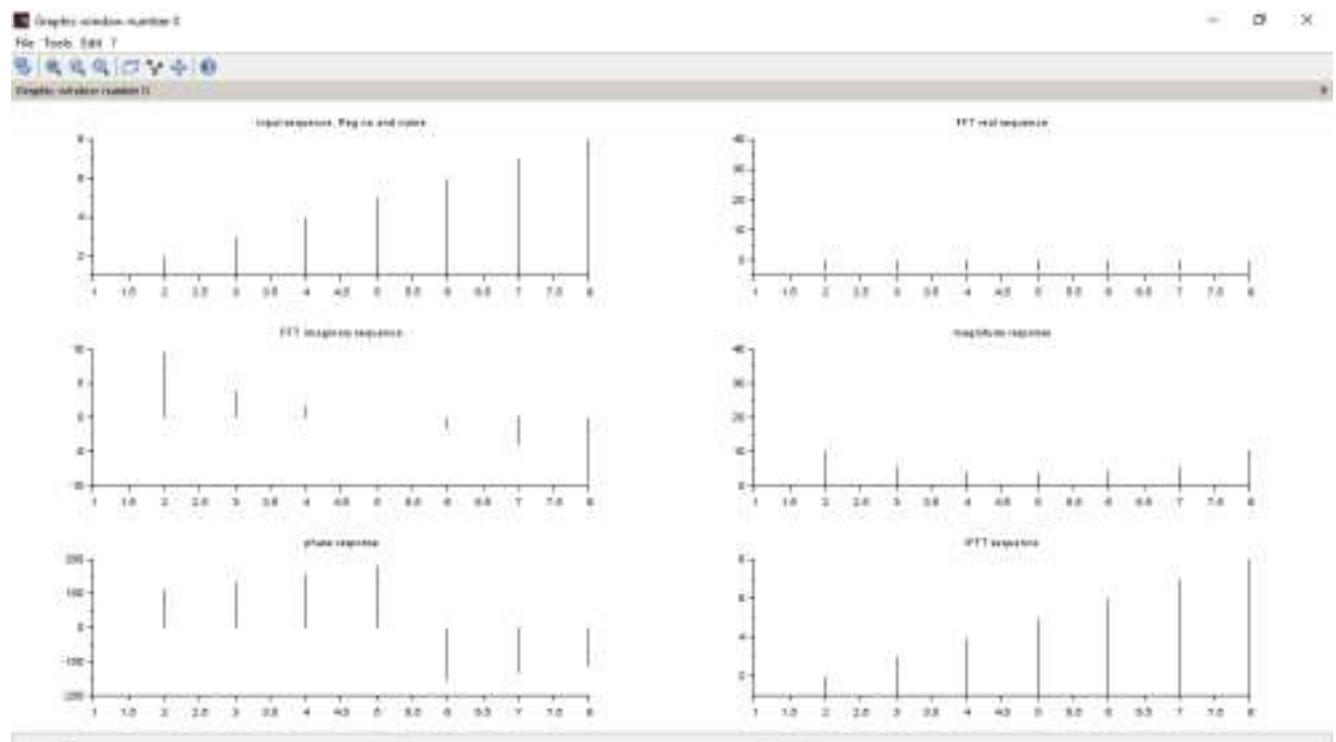
```
1 // program for calculation of FFT of a signal
2 clc ;
3 clf ;
4 clear all;
5 N = input('Enter the value of N');
6 x = input('enter input sequence');
7 y = fft(x);
8 A = real(y);
9 B = imag(y);
10 mag = abs(y);
11 x1 = atan(imag(y),real(y));
12 phase = x1 * (180/ %pi ) ;
13 disp('the resultant FFT sequence is ' );
14 disp(y);
15 disp('the magnitude response is ' );
16 disp(mag);
17 disp('the phase response is ' );
18 disp(phase);
19 z = ifft(-y);
20 disp('the resultant IFFT sequence is ' );
21 disp(z);
22 subplot(3,2,1);
23 plot2d3(x);
24 title('input sequence, Reg no and name');
25 subplot(3,2,2);
26 plot2d3(A);
27 title('FFT real sequence ' );
28 subplot(3,2,3);
29 plot2d3(B);
30 title('FFT imaginary sequence ' );
31 subplot(3,2,4);
32 plot2d3(mag);
33 title('magnitude response ' );
34 subplot(3,2,5);
```

Line 12, Column 25.

SCILAB CODE SIMULATION:-



SCILAB SIMULATION GRAPH:-



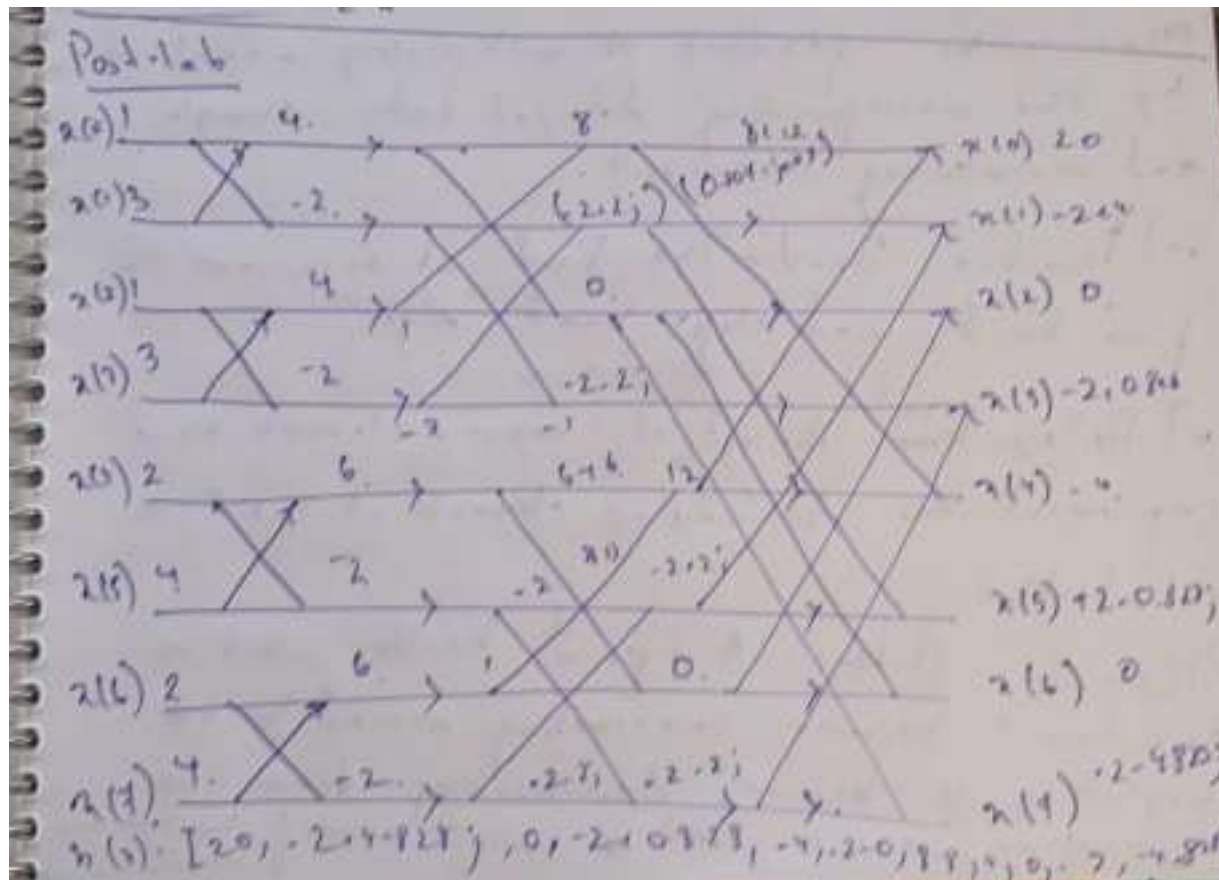
Pre-lab questions:

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.



POST LAB :-Q1(O/P)

```

DSP EXP 06.sce
1 // program for calculation of FFT of a signal
2 clc ;
3 clf ;
4 clear all;
5 N = input('Enter the value of N');
6 x = input('enter input sequence');
7 y = fft(x);
8 A = real(y);
9 B = imag(y);
10 mag = abs(y);
11 x1 = atan(imag(y),real(y));
12 phase = x1 * (180 / pi);
13 disp('the resultant FFT sequence is');
14 disp(y);
15 disp('the magnitude response is');
16 disp(mag);
17 disp('the phase response is');
18 disp(phase);
19 z = ifft(y);
20 disp('the resultant IFFT sequence is');
21 disp(z);
22 subplot(3,2,1);
23 plot2d3(x);
24 title('input sequence, Reg.no and name');
25 subplot(3,2,2);
26 plot2d3(A);
27 title('FFT real sequence');
28 subplot(3,2,3);
29 plot2d3(B);
30 title('FFT imaginary sequence');
31 subplot(3,2,4);
32 plot2d3(mag);
33 title('magnitude response');
34 subplot(3,2,5);

```

Enter the value of N 8

enter input sequence [1 2 1 2 3 4 3 4]

'the resultant FFT sequence is
,

coluem 1 to S
 20. + 0.i -2.
 + 4.8284271i
 0. + 0.i -2. +
 0.8284271i -4.
 + 0.i coluem 6
 to S
 -2. - 0.8284271i 0. +

```
0.i -2. - 4.8284271i
```

```
'the phase response is'
```

```
•the resultant IFM sequence is•
```

Result:

The computation of DFT and IDFT using FFT in Scilab code was simulated successful

Laboratory Report Cover Sheet

| |
|---|
| SRM Institute of Science and Technology
College of Engineering and Technology
Department of Electronics and Communication Engineering |
| 18ECC204J DIGITAL SIGNAL PROCESSING
Fifth Semester, 2021-22 (Odd semester) |

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4th /FN

Venue : ONLINE(G -Meet)

Title of Experiment : DESIGN OF DIGITAL FIR LOW PASS, HIGH PASS, BAND PASS, BAND STOP FILTER USING RECTANGULAR WINDOW

Date of Conduction : 31/08/2021

Date of Submission : 20/09/2021

| Particulars | Max. Marks | Marks Obtained |
|------------------------|------------|----------------|
| Pre lab and Post lab | 10 | |
| Lab Performance | 10 | |
| Simulation and results | 10 | |
| Total | 30 | |

REPORT VERIFICATION

Staff Name : Mrs.D.VIJAYALAKSHMI

Mrs.A.ANILETBALA

Signature :

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  
// F i l t e r 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]=');  
  
// P l o t t i n g 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

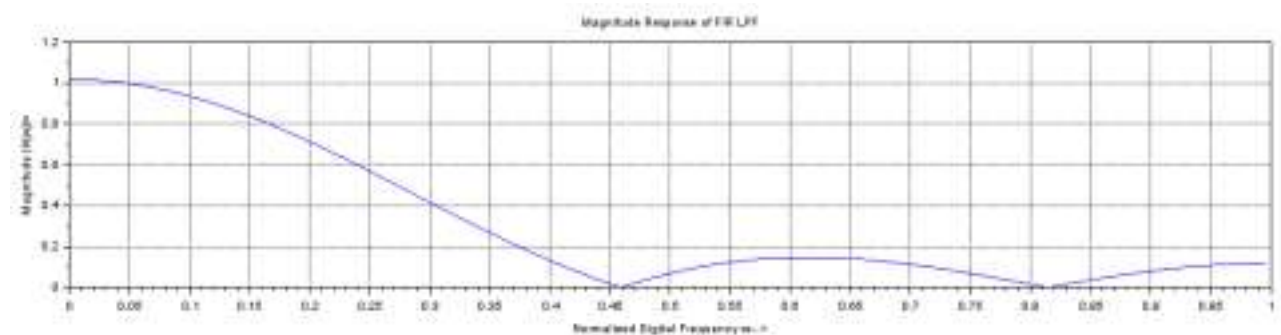
"Digital cut off frequency in radians,cycles/samples"

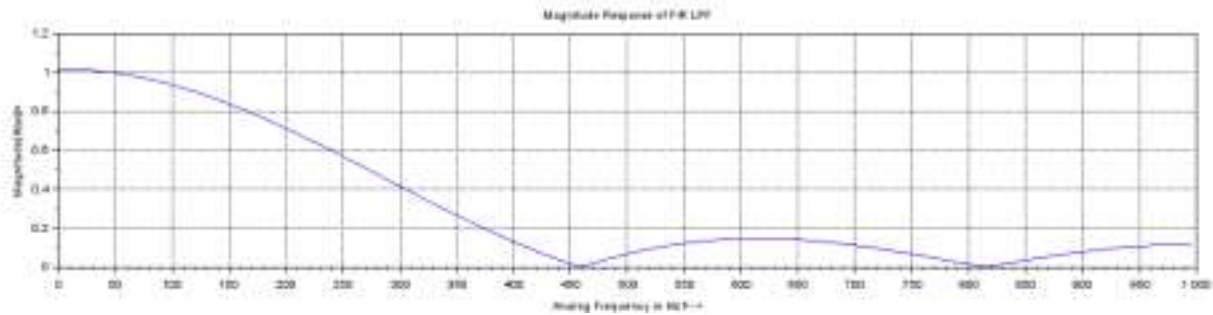
0.25

"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

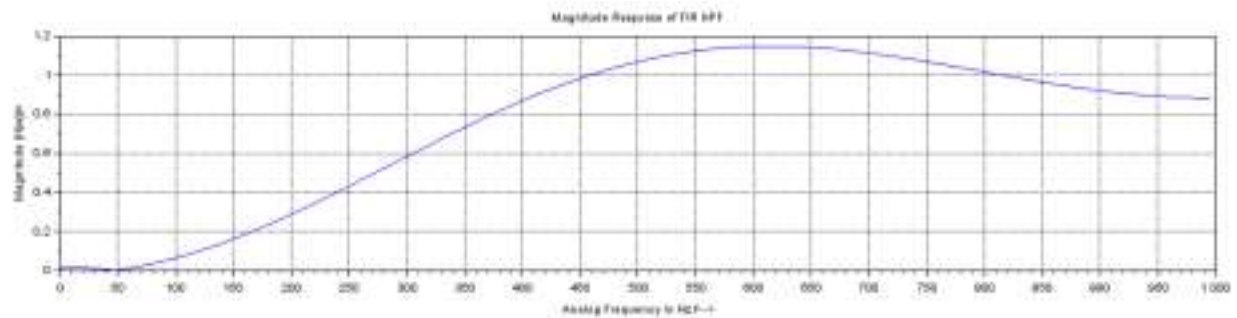
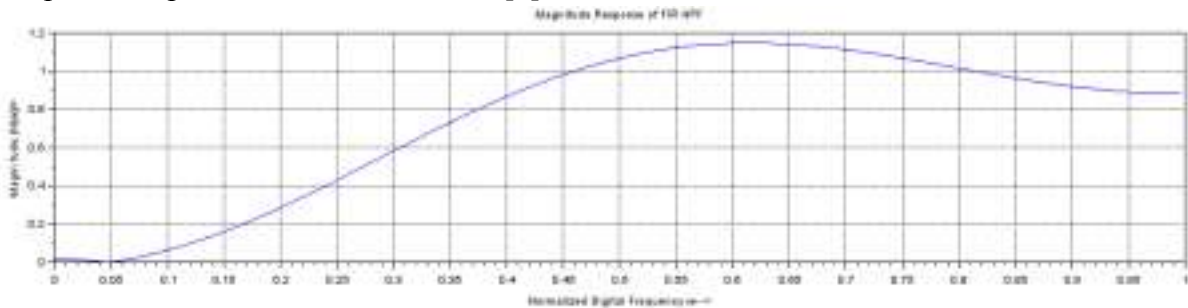
"Digital cut off frequency in radians cycles/samples"

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]=wfirm('bp',M+1,[wc1/2,wc2/2],'re',[0,0]);
disp(wft,'Impulse Response of BPF FIR Filter : h[ n]= ');
// P l o t t i n g t h e M a g n i t u d e R e s p o n s e o f H P F F I R
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

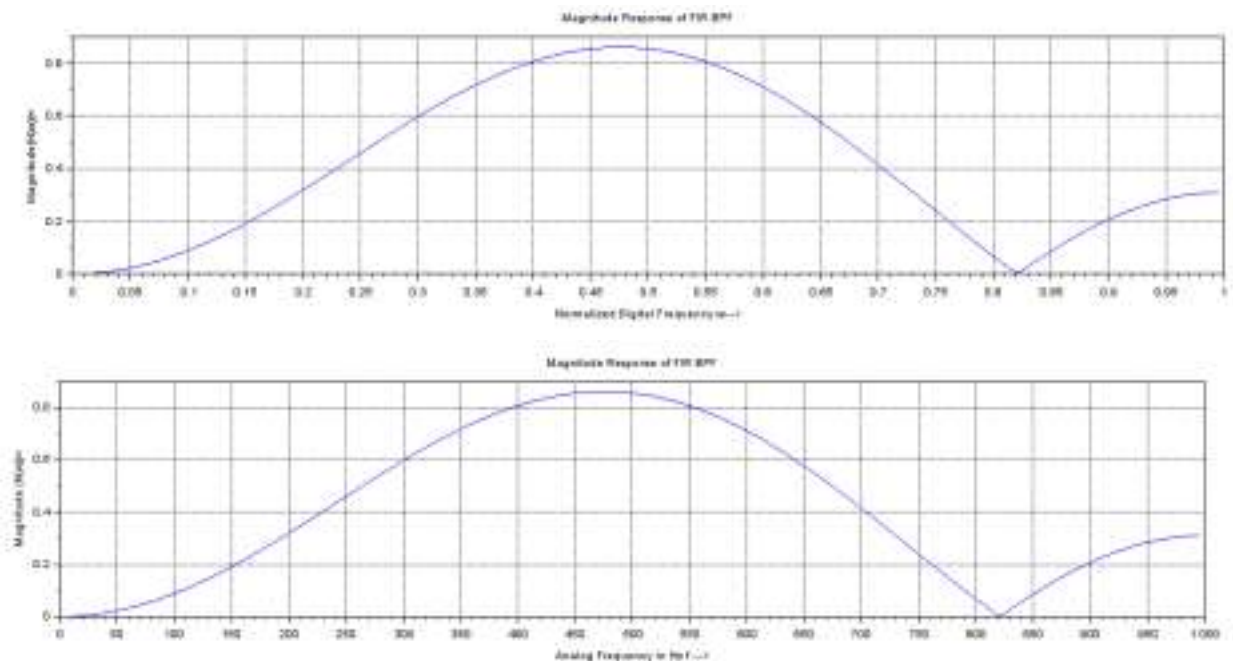
"Normalized digital lower cut off frequency in cycles/ samples"

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]= ' ); //
Pl o t t i n g t h e M a g n i t u d e R e s p o n s e o f H P F F I R F i l t e r
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

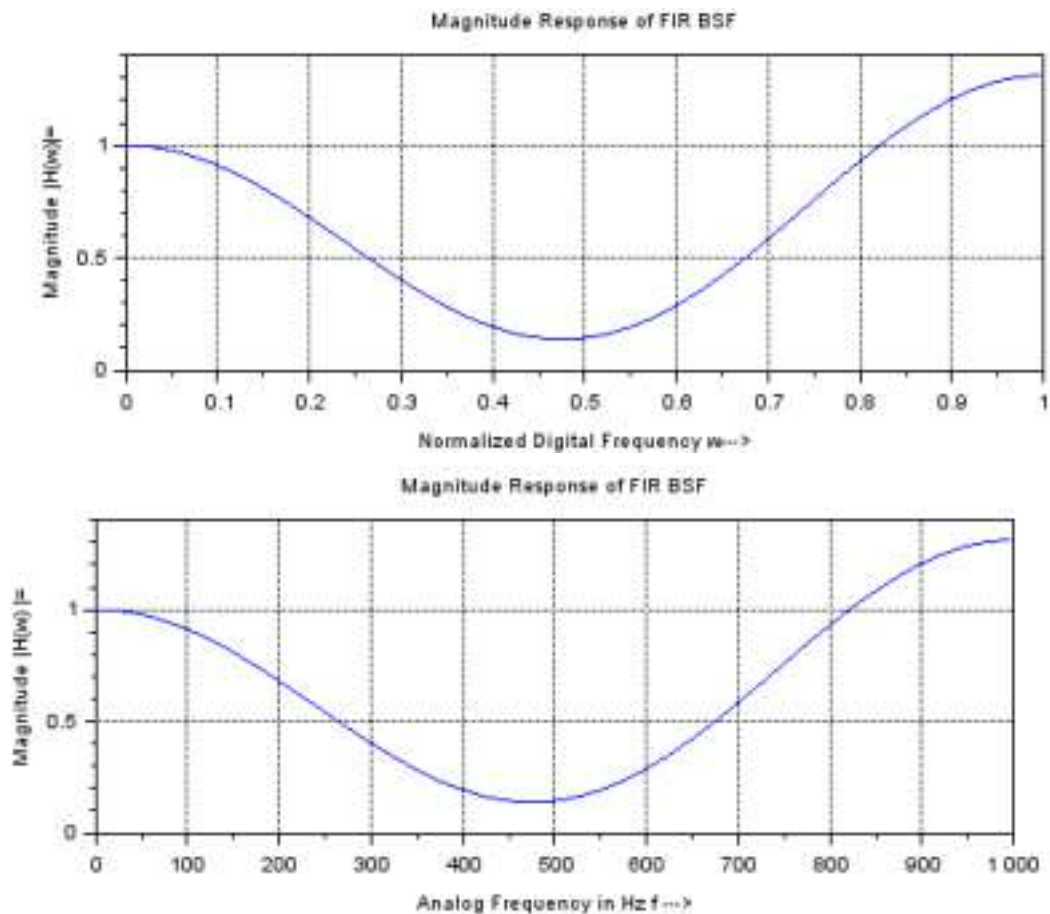
"Normalized digital lower cut off frequency in cycles / samples "

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 : $h[n]$ = "



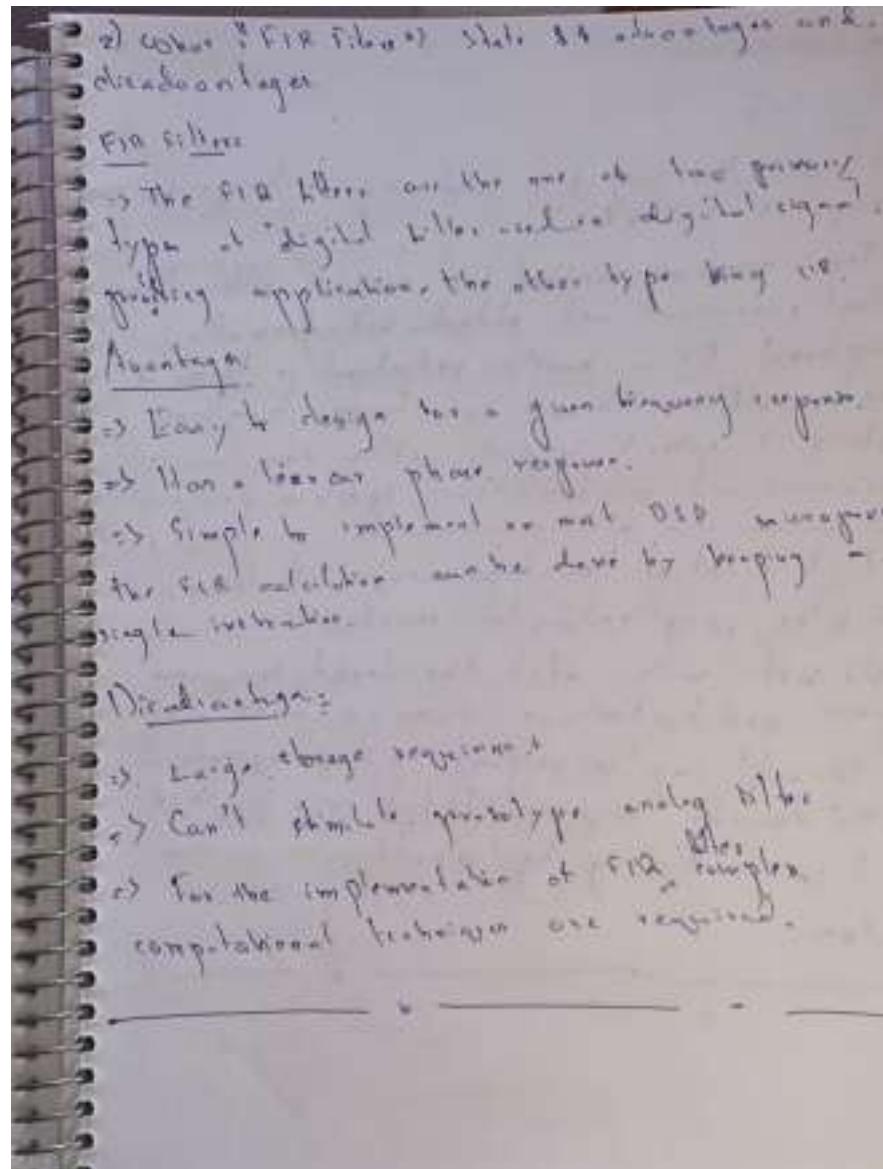
Prelab:

1. What terms are used in describing FIR filters?
2. What are "FIR filters?" State its advantages and disadvantages.

Post lab:

1. State the need for windowing technique for the design of FIR filters.
2. List in detail the steps required to design FIR filter using rectangular window.

Pre lab Answers :-



Post lab Answers :-

Post-lab

1) State the need for windowing technique for the design of FIR filter.

2) The window method for digital filter design is fast, convenient and efficient, but generally suboptimal. It is easily understood in terms of the convolution theorem for Fourier transforms, making it suitable to study after the Fourier theorems and windows for spectrum analysis.

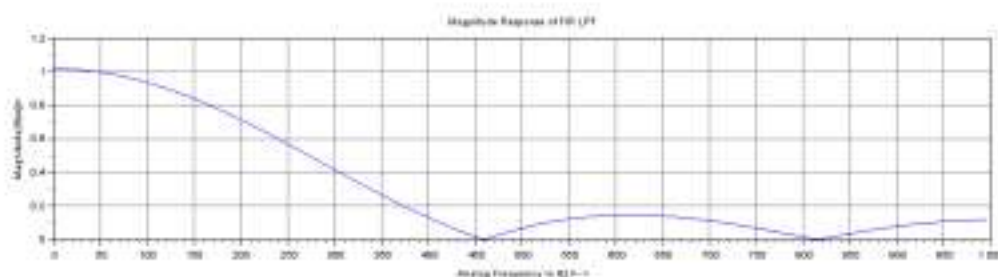
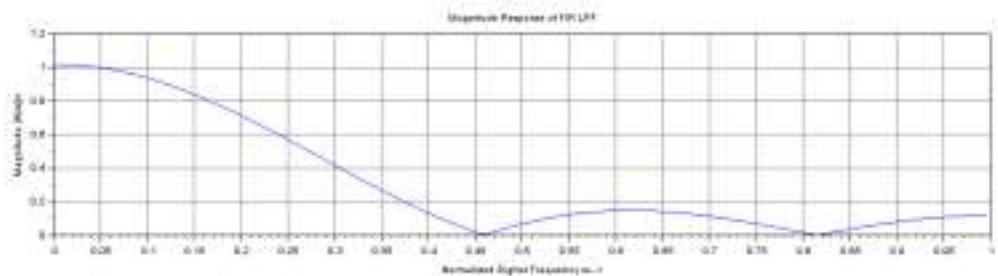
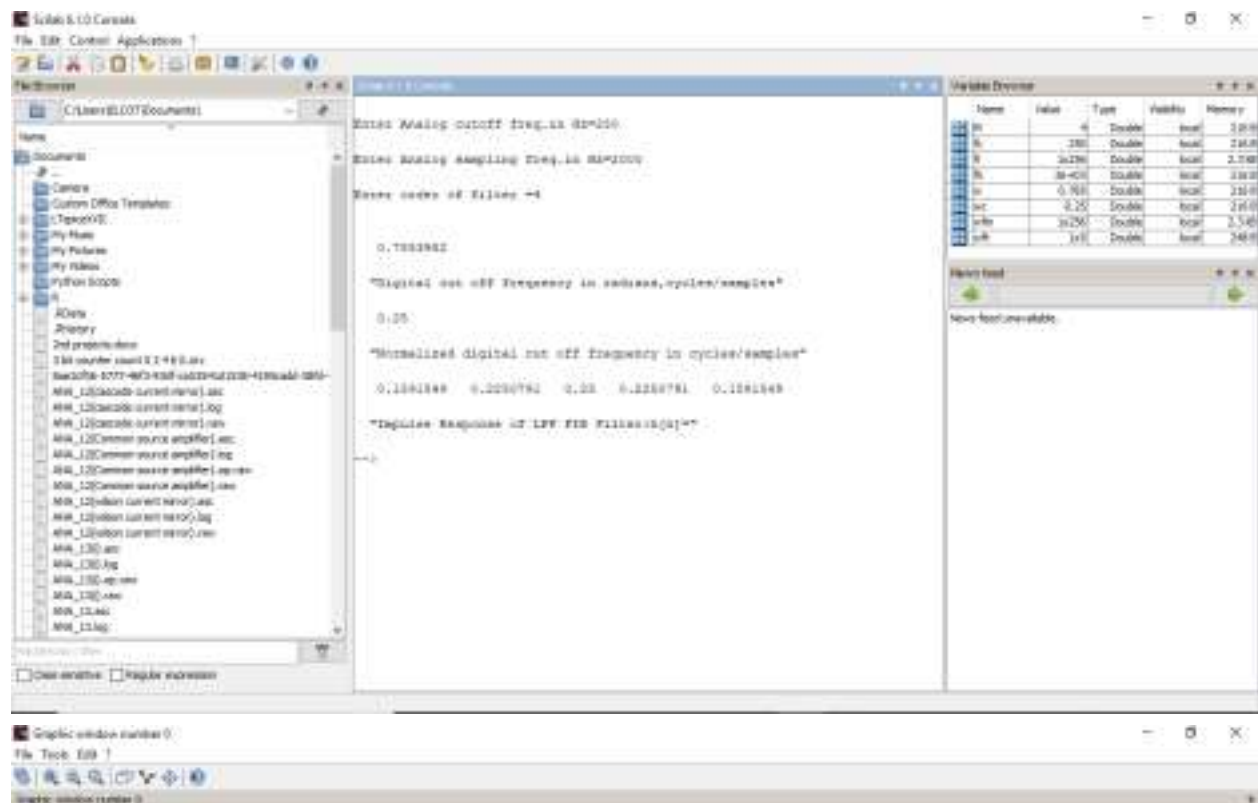
3) List in detail the steps required to design

FIR filter using rectangular window

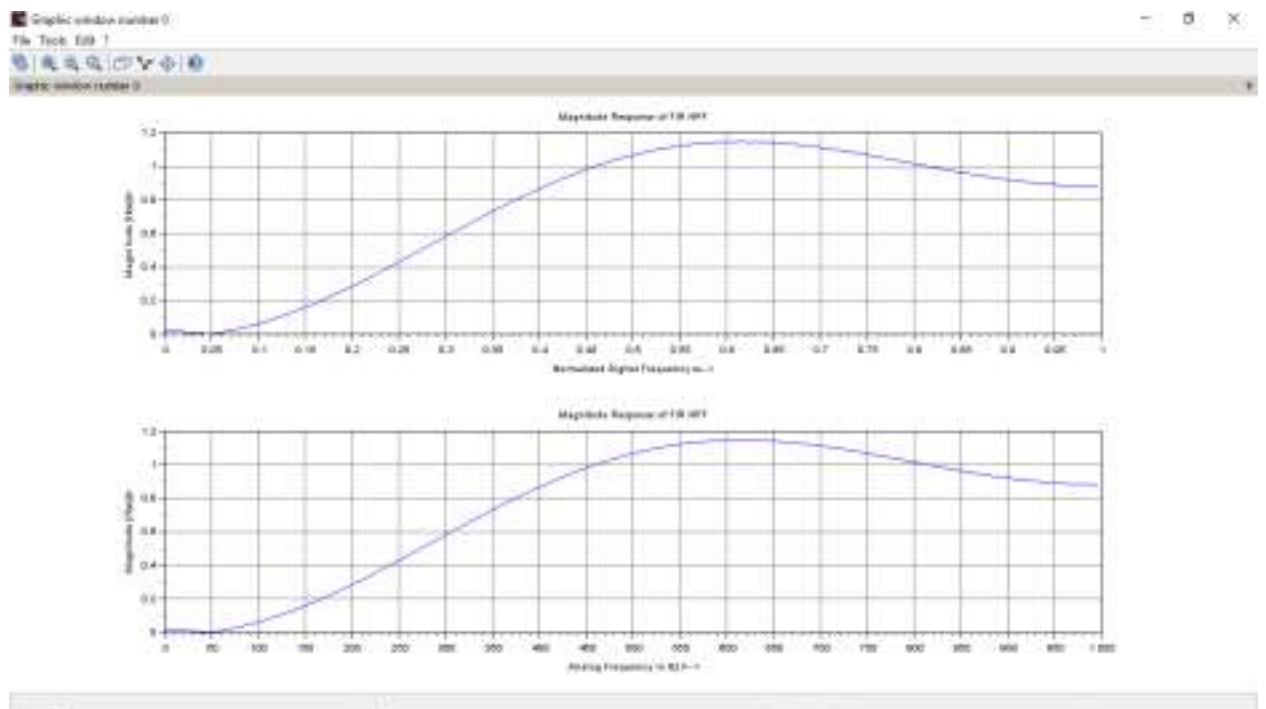
4) We start with $d(n)$, the desired frequency response and by this we come up with $d(n)$. Then the unit impulse response is done by integral of the desired response. $h(n)$ is a length of $(N+1)$ is multiplying by a rectangular window is done.

SIMULATION RESULT :-

Low pass filter



High pass filter :-



Band pass filter :-





Band stop filter :-



RESULT :-

Digital FIR Low pass, High pass, Band pass, Band stop filter using rectangular window verified by using Scilab software.

Laboratory Report Cover Sheet

| |
|---|
| SRM Institute of Science and Technology
College of Engineering and Technology
Department of Electronics and Communication Engineering |
| 18ECC204J DIGITAL SIGNAL PROCESSING
Fifth Semester, 2021-22 (Odd semester) |

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4th /FN

Venue : ONLINE(G -Meet)

Experiment : Design of digital FIR Low Pass and High Pass filter using .Hanning and Hamming window Theory

Date of Conduction : 07/09/2021

Date of Submission : 20/09/2021

| Particulars | Max. Marks | Marks Obtained |
|------------------------|------------|----------------|
| Pre lab and Post lab | 10 | |
| Lab Performance | 10 | |
| Simulation and results | 10 | |
| Total | 30 | |

REPORT VERIFICATION

Staff Name : Mrs.D.VIJAYALAKSHMI

Mrs.A.ANILETBALA

Signature :

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

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

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.

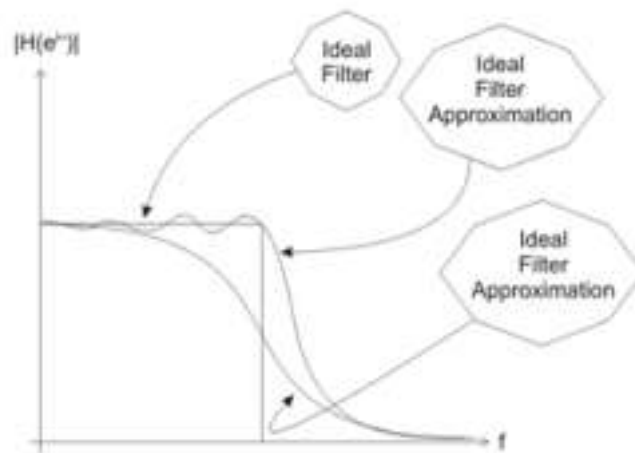


Fig-1

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

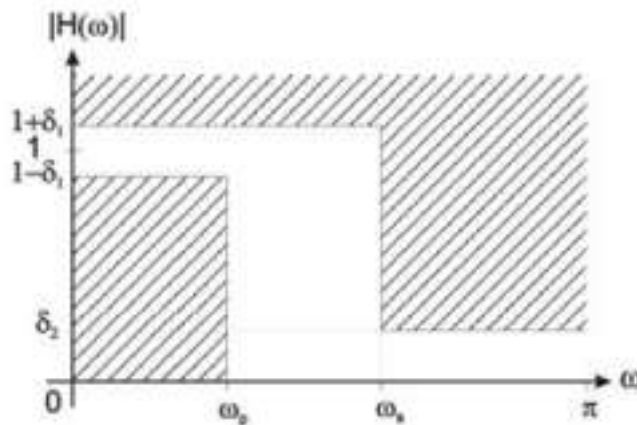


Figure 2.a: Low-pass digital filter specification

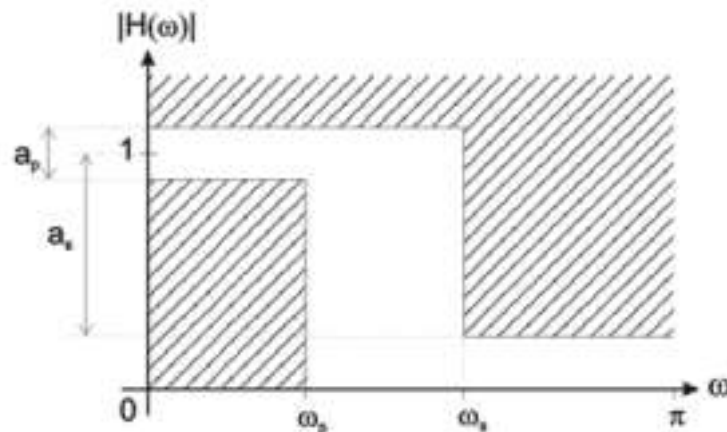


Figure 2.b: Low-pass digital filter specification

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].

$$a_p = 20 \log_{10} \left(\frac{1 + \delta_1}{1 - \delta_1} \right)$$

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

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

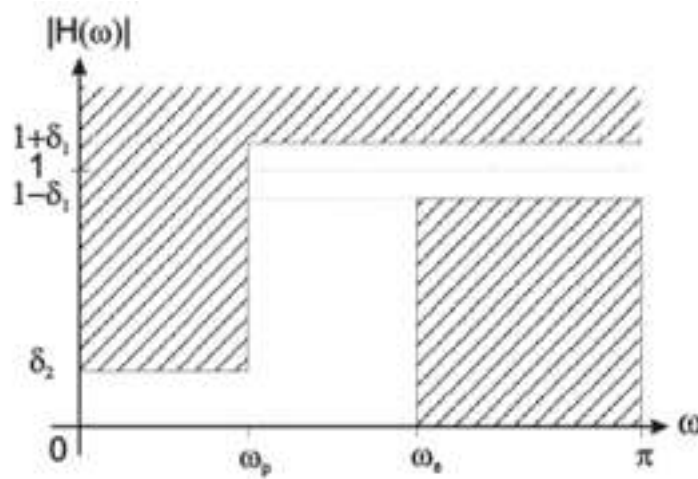


Figure 3.b: High-pass digital filter specification

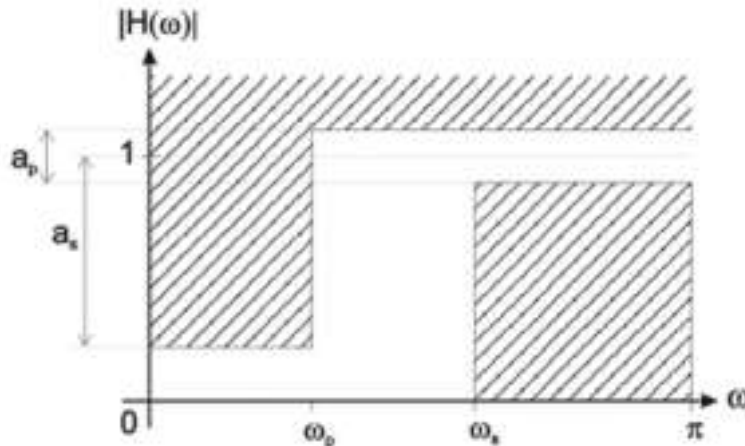


Figure 3.b: High-pass digital filter specification

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

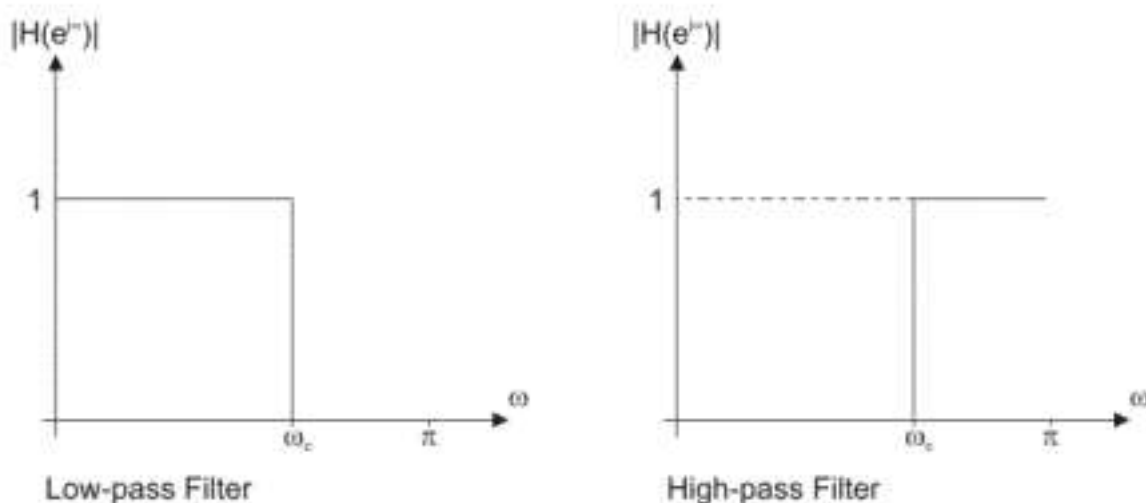


Figure 4.a and 4.b: Transfer functions of two standard ideal filters

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 $N+1$ sample. If it is needed to find frequency response of a non-standard ideal filter, the expression for inverse Fourier transform must be used:

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

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:

$$H(e^{j\omega}) = \sum_{n=0}^{N-1} h[n] e^{-j\omega n}$$

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

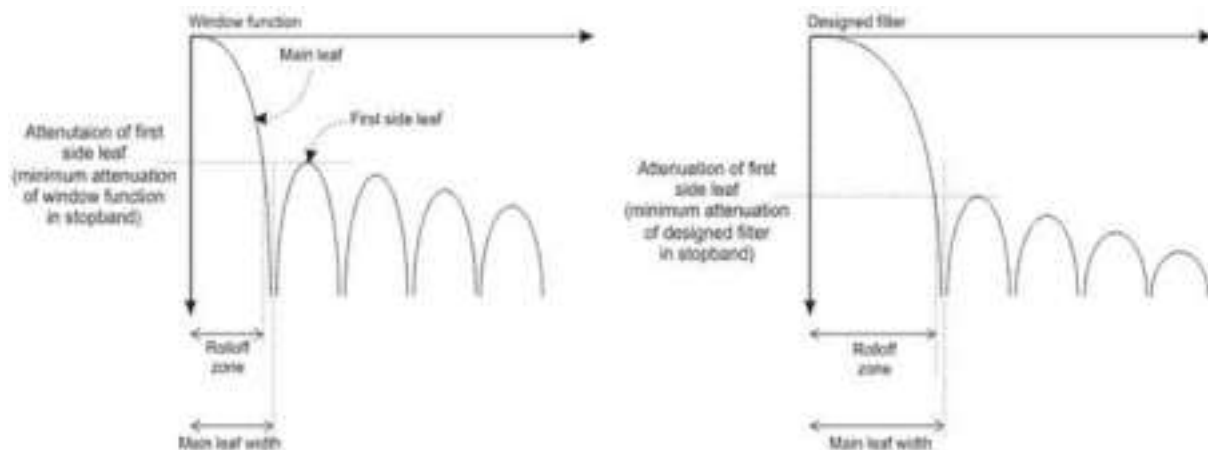


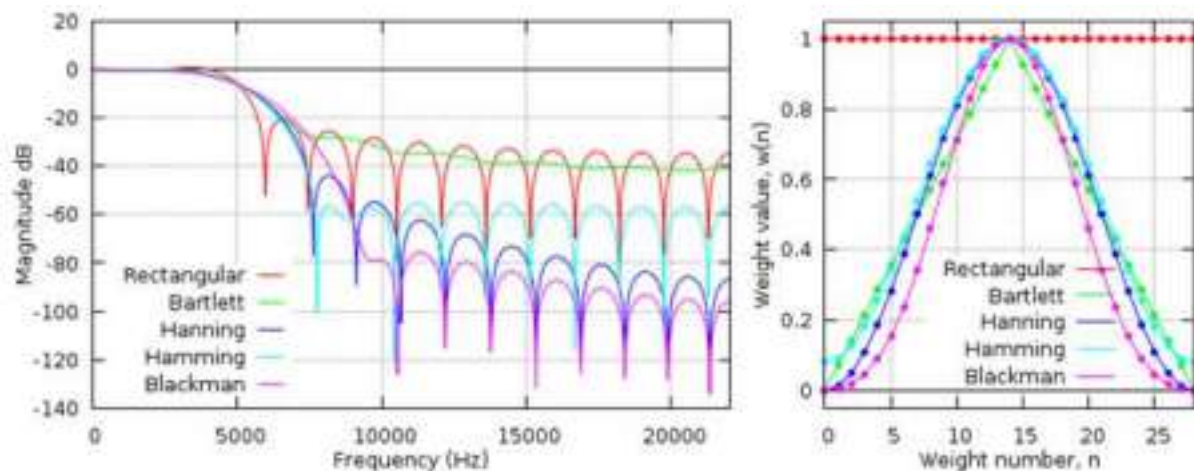
Figure 5: Main lobe, main lobe width, side lobes, transition region

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 |
| Bartlett (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
<i>$I_0(x)$ is the zeroth order Bessel function.</i> | $\frac{I_0[\alpha \sqrt{(\frac{N-1}{2})^2 - (n - \frac{N-1}{2})^2}]}{I_0[\alpha (\frac{N-1}{2})]}$ |

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 bellow 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 $h_u(n)$ followed by $h(n)$ (Multiplying $h_hann(n) * h_u(n)$).

```
--> N=11;
--> U=6;
--> h_hann = window ( 'hann',N )
h_hann =
    0.0826497    0.0554915    0.3454915    0.6545085    0.5045085    1.0000000    0.3045085    0.6545085    0.3454915    0.0554915    0.0826497
--> for n = -5:U :1:5+U
> if n==0
> 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 $\omega=0^\circ$ like wise 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*”.

```

% hz_db = 20*log10( hz / min_hz )
hz_db =

```

```

column 1 to 15

```

```

-21.52167 -21.518256 -21.678142 -21.603728 -21.911665 -22.19483 -21.95431 -22.591383 -22.707389 -22.503883 -22.292454 -22.044658 -21.782214 -21.524651 -21.248561

```

```

column 16 to 30

```

```

-21.942857 -21.666788 -21.363896 -21.055507 -21.741485 -21.424229 -21.104294 -20.782582 -20.459935 -20.137841 -21.814627 -21.493221 -21.170361 -20.855565 -20.548055

```

```

column 31 to 45

```

```

-24.22736 -23.917725 -23.611411 -23.308443 -23.009412 -22.714476 -22.42337 -22.136492 -21.853461 -21.574214 -21.301128 -21.032404 -20.768104 -20.508204 -20.249774

```

```

column 46 to 60

```

```

-9.9967445 -9.7491237 -9.5058524 -9.2673288 -9.0324938 -8.8012625 -8.5742942 -8.3516548 -8.1336814 -7.9204489 -7.7142031 -7.5158895 -7.3257777 -7.1331982 -6.9387044

```

```

column 61 to 75

```

```

-6.7270008 -6.5411797 -6.3580412 -6.1785838 -6.002785 -5.8304509 -5.6617781 -5.5000298 -5.3417383 -5.1842837 -5.0301315 -4.8782894 -4.7284437 -4.5802218 -4.4339167

```

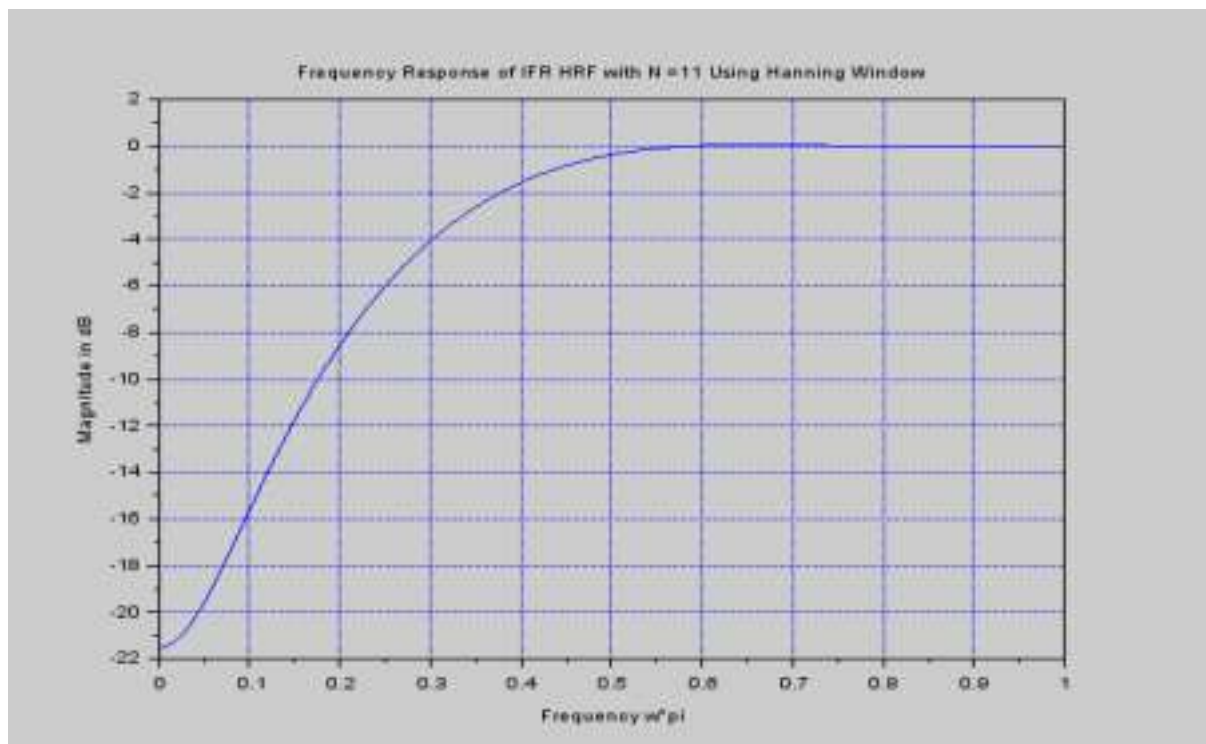


Fig.1: Magnitude response of HPF using Hanning Window

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

$$H_d e^{j\omega} = e^{-j3\omega} \text{ for } -\pi/4 \leq \omega \leq \pi$$

= 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 ) ;

```

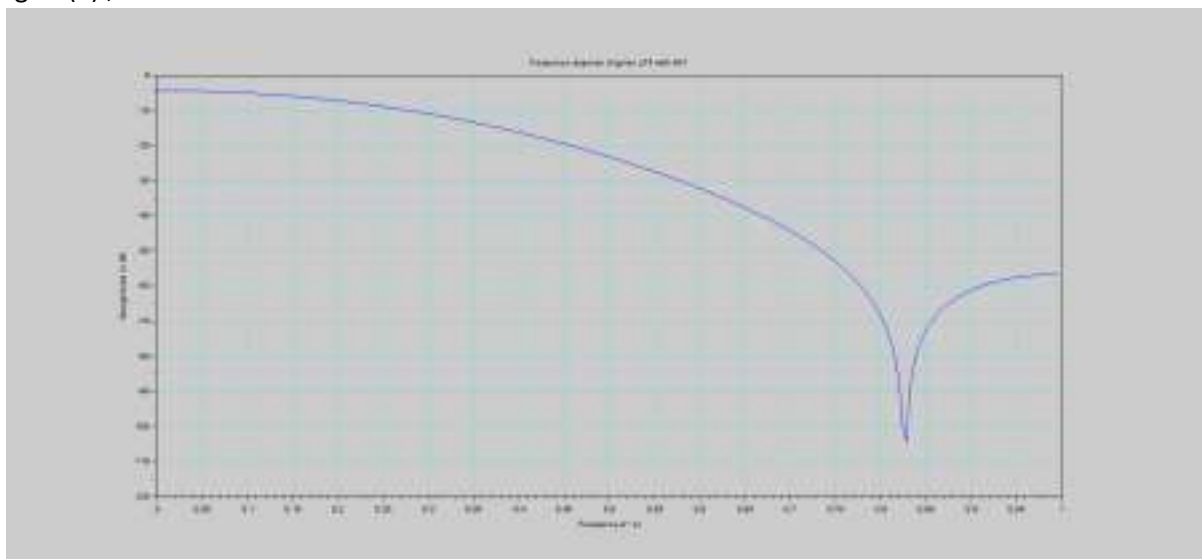
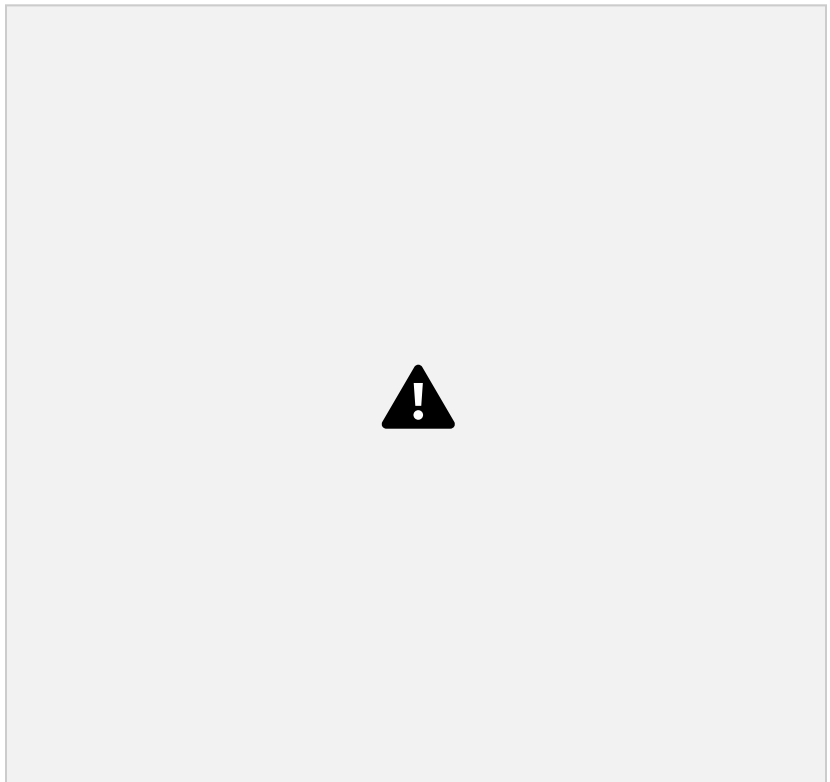
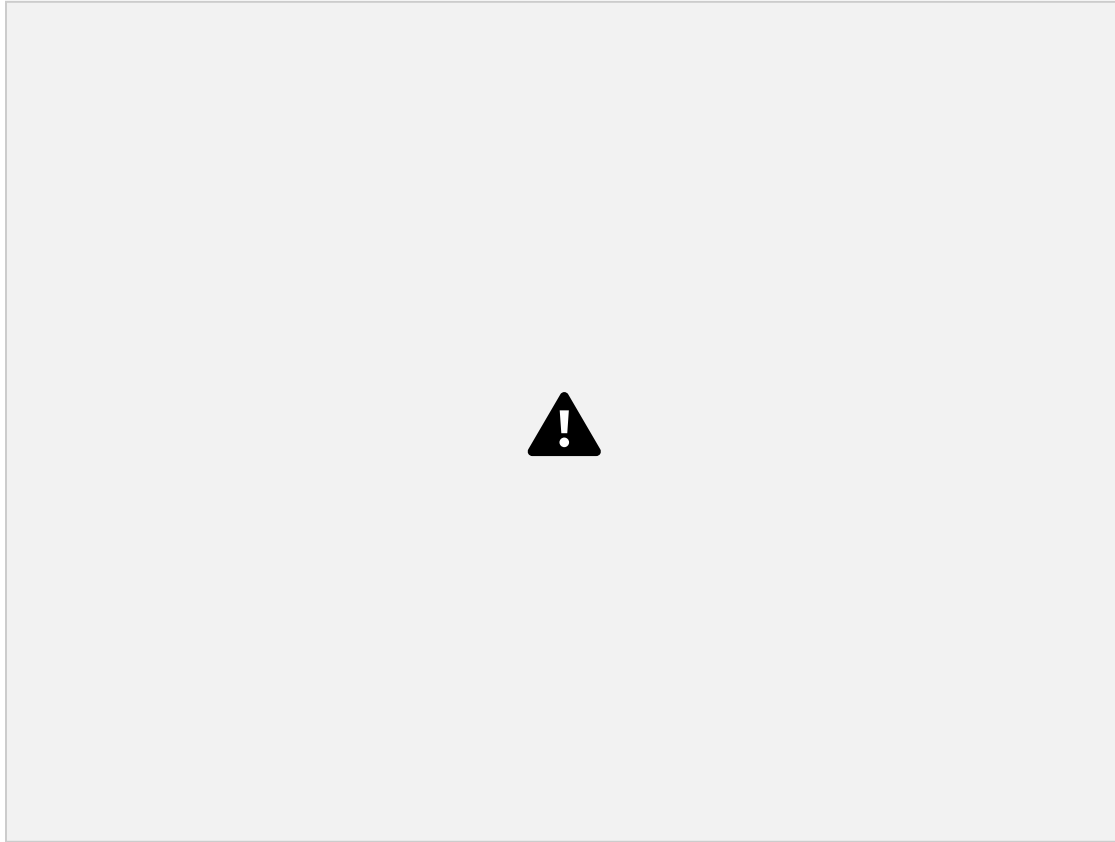


Fig.2: Magnitude response of LPF filter using Hamming Window

SCI LAB SCIMULATION:-

High Pass Filter using Hanning Window



High Pass Filter using Hamming Window



Low Pass Filter using Hanning Window



Low Pass 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

PRE LAB:-

Hanning Window:-





Hamming Window





POST LAB:-



Hamming Window

Result:

Design of digital FIR Low Pass and High Pass filter using Hanning and Hamming Window was simulated successfully.

| |
|---|
| SRM Institute of Science and Technology
Faculty of Engineering and Technology
Department of Electronics and Communication Engineering |
| 18ECC204J DIGITAL SIGNAL PROCESSING |
| Fifth Semester, 2020-21 (Odd semester) |

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4TH /FN

Venue : Online(G Meet)

Title of Experiment : 9.Design of digital FIR filter using frequency sampling method

Date of Conduction : 15 SEP 2021

Date of Submission : 29 SEP 2021

| Particulars | Max. Marks | Marks Obtained |
|------------------------|------------|----------------|
| Pre lab and Post lab | 10 | |
| Lab Performance | 10 | |
| Simulation and results | 10 | |
| Total | 30 | |

REPORT VERIFICATION

**Staff Name : Mrs.D.VIJAYALAKSHMI
Mrs.A.ANILETBALA**

Signature :

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

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

$$Hae^{j\omega} = e^{-j(N-1)\omega/2} \text{ for } 0 \leq |\omega| \leq \pi/2$$
$$= 0 \text{ for } \pi/2 \leq |\omega| \leq \pi. \text{ For } N=7.$$

```
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)=")
```

```
--> close ;
--> N =7;
--> U =1; // Z e r o A d j u s t
--> for n =0+ U :1: N -1+ U
> h ( n ) =(1+2* cos (2* %pi *( n -U -3) /7) ) / N
> end
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)="
```

SCILAB SIMULATION:-

```
exp 9.sce (C:\Users\ELCOT\Downloads\exp 9.sce) - SciNotes
File Edit Format Options Window Execute ?
exp 9.sce (C:\Users\ELCOT\Downloads\exp 9.sce) - SciNotes
EXP 09.sce exp 9.sce
1 clear ;
2 clc ;
3 close ;
4 N =7;
5 U =1; // Zero Adjust
6 for n =0+ U :1: N -1+ U
7 h ( n ) =(1+2* cos (2* %pi *( n -U -3) /7) ) / N
8 end
9 disp (h , "Filter Coefficients , h(n)=")
10
```

Scilab 6.1.0 Console

File Edit Control Applications ?

File Browser C:\Users\ELCOT\Documents\

Documents

- Camera
- Custom Office Templates
- [TspiceKV1]
- My Music
- My Pictures
- My Videos
- Python Scripts
- R
- .RData
- .Rhistory
- 18ECP1011-MASSIVE OPENI.patx
- 2nd projects.docx
- 3 bit counter count 0 2 4 6 0.drc
- 8ae2c106-5777-4613-93df-ca53541d1538-4195cadd-38fj-
- ANA_12(cascade current mirror).asc
- ANA_12(cascade current mirror).log
- ANA_12(cascade current mirror).raw
- ANA_12(Common source amplifier).acc
- ANA_12(Common source amplifier).log
- ANA_12(Common source amplifier).op.raw
- ANA_12(Common source amplifier).raw
- ANA_12(nelson current mirror).asc
- ANA_12(nelson current mirror).log
- ANA_12(nelson current mirror).raw
- ANA_13().asc
- ANA_13().log
- ANA_13().op.raw
- ANA_13().raw
- ANA_13.acc

File/Directory Filter

☐ Case sensitive ☐ Regular expression

Scilab 6.1.0 Console

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

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

Variable Browser

| Name | Value | Type | Visibility | Memory |
|------|-------|--------|------------|--------|
| N | 7 | Double | local | 216 B |
| U | 1 | Double | local | 216 B |
| h | 7x1 | Double | local | 264 B |
| n | 7 | Double | local | 216 B |

News feed

News feed unavailable.

Pre-lab Question

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

Exp. 09

RA911004010565

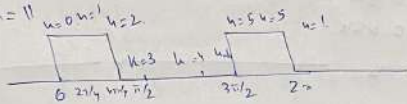
Pushpa Das.

Pre-lab

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

$$1) \frac{1}{2} (e^{j\omega}) = 1 ; 0 \leq |\omega| \leq \pi/2$$

$$= 0 ; \pi/2 < |\omega| < \pi/2$$

For $N=11$ 

$$H(u) = 1/2 (e^{j\omega}) |_{\omega = \frac{2\pi k}{N}} ; k = 0, 1, 2, \dots, N-1$$

$$\omega = \frac{2\pi k}{N} ; k = 0, 1, \dots, 10$$

$$k=0, \omega=0 ; k=1, \omega=\frac{2\pi}{11} ; k=2, \omega=\frac{4\pi}{11}$$

$$k=3, \omega=\frac{6\pi}{11} ; k=4, \omega=\frac{8\pi}{11} ; k=5, \omega=\frac{10\pi}{11}$$

$$k=6, \omega=\frac{12\pi}{11} ; k=7, \omega=\frac{14\pi}{11} ; k=8, \omega=\frac{16\pi}{11}$$

$$k=9, \omega=\frac{18\pi}{11} ; k=10, \omega=\frac{20\pi}{11}$$

$$H(u) = 1/2 (e^{j\omega}) = \frac{1}{2} \frac{e^{j\omega}}{11}$$

$$H(u) = 0 ; u = -3, 4, 5, 6, 7, 8$$

$$= e^{-j\frac{2\pi k}{N}} ; k = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$$

$$h(n) = \frac{1}{N} \left[\sum_{k=0}^{N-1} H(k) e^{-j\frac{2\pi k}{N} n} \right]$$

$$= \frac{1}{11} \left[(1 + e^{j\frac{2\pi}{11}(n-5)} + e^{j\frac{4\pi}{11}(n-5)} + e^{j\frac{6\pi}{11}(n-5)} + e^{j\frac{8\pi}{11}(n-5)} + e^{j\frac{10\pi}{11}(n-5)} + e^{j\frac{12\pi}{11}(n-5)} + e^{j\frac{14\pi}{11}(n-5)} + e^{j\frac{16\pi}{11}(n-5)} + e^{j\frac{18\pi}{11}(n-5)} + e^{j\frac{20\pi}{11}(n-5)}) \right]$$

$$h(n) = \frac{1}{11} \left[1 + 2 \cos \left(\frac{2\pi}{11} (n-5) \right) + 2 \cos \left(\frac{4\pi}{11} (n-5) \right) \right]$$

$h(0) = 0.0694 = h(10)$
 $h(1) = -0.054 = h(9)$
 $h(2) = -0.109 = h(8)$
 $h(3) = -0.47 = h(7)$
 $h(4) = -0.819 = h(6)$
 $h(5) = -0.454$

Pre lab (1 ans):-

```

exp 9.sce (C:\Users\ELCOT\Downloads\exp 9.sce) - SciNotes
File Edit Format Options Window Execute ?

exp 9.sce (C:\Users\ELCOT\Downloads\exp 9.sce) - SciNotes
EXP 09.sce exp 9.sce
1 clear ;
2 c1o ;
3 close ;
4 N=11;
5 U=1; // Deco Adjust
6 for n =0+U-1: N-1+U
7   h ( n ) =(1+2* cos (2* pi *( n -U-5) /7) ) / N
8 end
9 disp (h , " Filter Coefficients , h(n)=")
10

```

SciLab 6.1.0 Console

File Edit Control Applications ?

File Browser

C:\Users\ELCOT\Documents\

Documents

- Camera
- Custom Office Templates
- LTspiceXVI
- My Music
- My Pictures
- My Videos
- Python Scripts
- R
- RData
- Rhistory
- 18ECP1011-MASSIVE OPEN.pptx
- 2nd projects.docx
- 3 bit counter count 0 2 4 6 0.drc
- 8ec2f0f6-5777-4613-931f-ca53541d1538-4195cadd-38fd-
- ANA_12(cascode current mirror).asc
- ANA_12(cascode current mirror).log
- ANA_12(cascode current mirror).raw
- ANA_12(Common source amplifier).asc
- ANA_12(Common source amplifier).log
- ANA_12(Common source amplifier).op raw
- ANA_12(Common source amplifier).raw
- ANA_12(nelson current mirror).asc
- ANA_12(nelson current mirror).log
- ANA_12(nelson current mirror).raw
- ANA_13(0).asc
- ANA_13(0).log
- ANA_13(0).op.raw
- ANA_13(0).raw
- ANA_13.asc

File/directory filter

☐ Case sensitive ☐ Regular expression

SciLab 6.1.0 Console

```

-0.0729034
0.0504507
0.2042709
0.2727273
0.2042709
0.0504507
-0.0729034
-0.0729034
0.0504507
0.2042709
0.2727273

" Filter Coefficients , h(n)="
--> |

```

Variable Browser

| Name | Value | Type | Visibility | Memory |
|------|-------|--------|------------|--------|
| N | 11 | Double | local | 216 B |
| U | 1 | Double | local | 216 B |
| h | 11x1 | Double | local | 296 B |
| n | 11 | Double | local | 216 B |

News feed

News feed unavailable.

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.

POST LAB:-

Post lab

1) What is the general formulae to find the values of $H(k)$, $\theta(k)$ and $h(n)$ using frequency sampling method?

$$H(k) = \left. \text{Id} \left(e^{j\omega} \right) \right|_{\omega = \frac{2\pi k}{N}}; k = 0, 1, \dots, (N-1)$$

N is odd -

$$h(n) = \frac{1}{N} \left[h(0) + 2 \sum_{k=1}^{N/2-1} \text{Re} \left[h(k) e^{j\frac{2\pi kn}{N}} \right] \right]$$

N is even -

$$h(n) = \frac{1}{N} \left[h(0) + 2 \sum_{k=1}^{N/2-1} \text{Re} \left[h(k) e^{j\frac{2\pi kn}{N}} \right] \right]$$
$$\theta(k) = - \left(\frac{N-1}{N} \right) \pi$$

2) List the steps in defining the k value of frequency sampling method.

For proper k value -

- Find two cutoff frequency ω_{c1} , ω_{c2} , $\omega_{c3} = \frac{2\pi k}{T_s}$
- Draw the logic band diagram to get an ideal ω_{c1} , ω_{c2} -
- And $H_s(e^{j\omega})$, $e^{-j\omega d}$, $k = \frac{N-1}{2}$ (odd) or $d = \frac{N}{2}$ (even)
- Find $H(k) = \text{Id} \left(e^{j\frac{2\pi kn}{N}} \right)$
- Take values at ω_{c1} given $k = 0, 1, 2, \dots, (N-1)$
- Verify it in substituting in equation and verify whether it is in the two proper cutoff frequency. If not ignore.

Result:-

The design of digital FIR filter using Frequency Sampling Method using SCILAB was done successfully.

Laboratory Report Cover Sheet

SRM Institute of Science and Technology
Faculty of Engineering and Technology
Department of Electronics and Communication Engineering

**18ECC204J DIGITAL SIGNAL
PROCESSING** Fifth Semester, 2020-21 (Odd semester)

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4th/ FN

Venue : Online (G Meet)

**Title of Experiment : 10(a) Design of Analog Butterworth filter
10(b) Design of Analog Chebyshev filter**

Date of Conduction : 21 SEP 2021

Date of Submission : 9 OCTOBER 2021

| Particulars | Max. Marks | Marks
Obtained |
|------------------------|------------|-------------------|
| Pre lab and Post lab | 1
0 | |
| Lab Performance | 1
0 | |
| Simulation and results | 1
0 | |
| Total | 30 | |

REPORT VERIFICATION

Staff Name : Mrs.D.VIJAYALAKSHMI

Mrs.A.ANILETBALA

Signature :

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, (A_{max}).

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
```

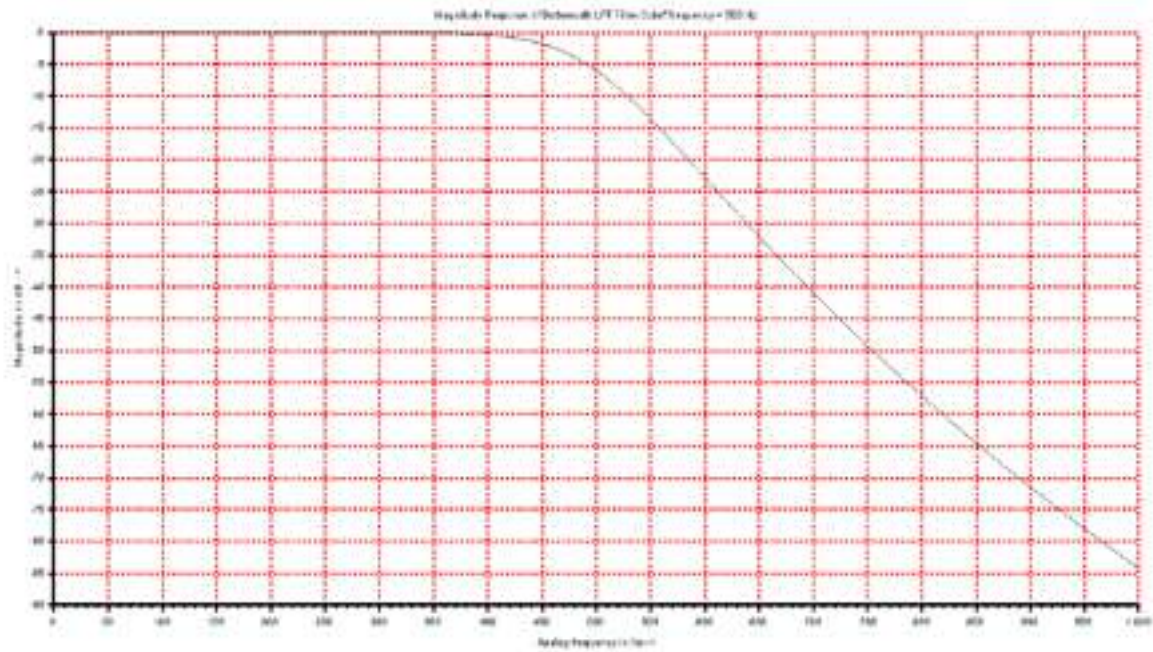
```
omegas = 1000; //stop band edge frequency
```

```

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:

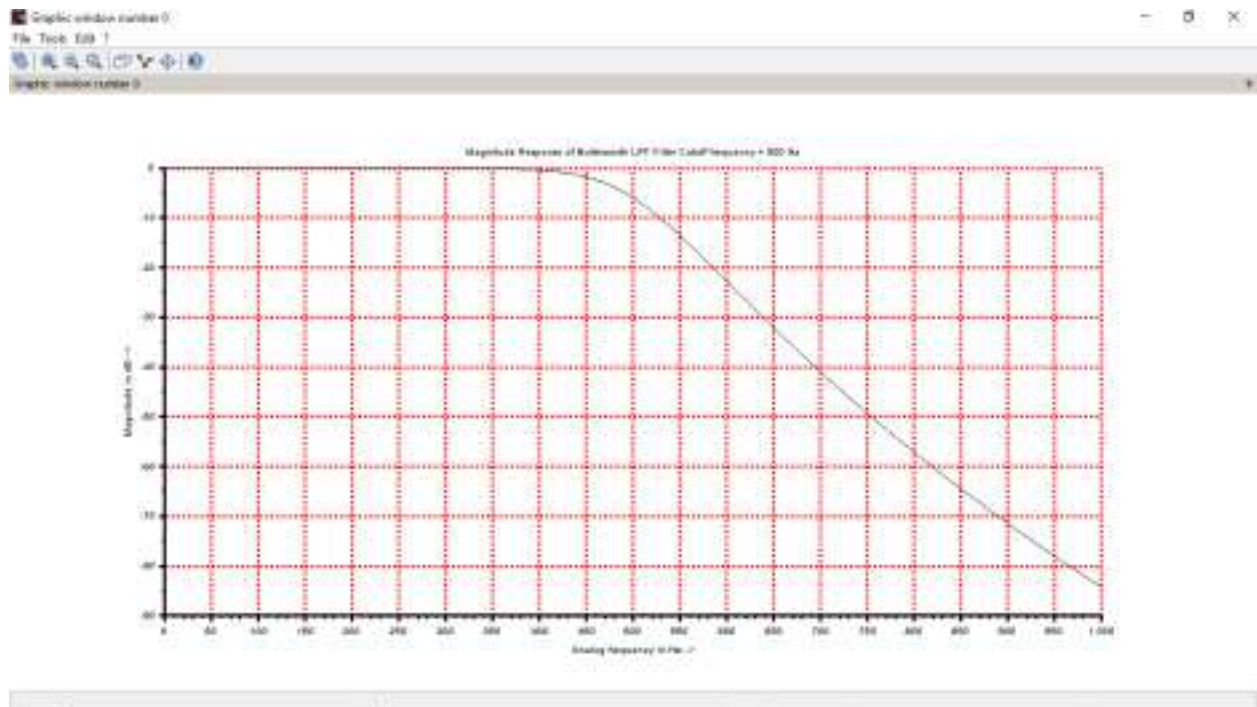


SCI LAB CODE & SIMULATION:-

```

1 // To Design a Second-Order Filter
2 // The filter is a second-order filter
3 // The filter is a second-order filter
4 // The filter is a second-order filter
5 // The filter is a second-order filter
6 // The filter is a second-order filter
7 // The filter is a second-order filter
8 // The filter is a second-order filter
9 // The filter is a second-order filter
10 // The filter is a second-order filter
11 // The filter is a second-order filter
12 // The filter is a second-order filter
13 // The filter is a second-order filter
14 // The filter is a second-order filter
15 // The filter is a second-order filter
16 // The filter is a second-order filter
17 // The filter is a second-order filter
18 // The filter is a second-order filter
19 // The filter is a second-order filter
20 // The filter is a second-order filter
21 // The filter is a second-order filter
22 // The filter is a second-order filter
23 // The filter is a second-order filter
24 // The filter is a second-order filter
25 // The filter is a second-order filter
26 // The filter is a second-order filter
27 // The filter is a second-order filter
28 // The filter is a second-order filter
29 // The filter is a second-order filter
30 // The filter is a second-order filter
31 // The filter is a second-order filter
32 // The filter is a second-order filter
33 // The filter is a second-order filter
34 // The filter is a second-order filter
35 // The filter is a second-order filter
36 // The filter is a second-order filter
37 // The filter is a second-order filter
38 // The filter is a second-order filter
39 // The filter is a second-order filter
40 // The filter is a second-order filter
41 // The filter is a second-order filter
42 // The filter is a second-order filter
43 // The filter is a second-order filter
44 // The filter is a second-order filter
45 // The filter is a second-order filter
46 // The filter is a second-order filter
47 // The filter is a second-order filter
48 // The filter is a second-order filter
49 // The filter is a second-order filter
50 // The filter is a second-order filter
51 // The filter is a second-order filter
52 // The filter is a second-order filter
53 // The filter is a second-order filter
54 // The filter is a second-order filter
55 // The filter is a second-order filter
56 // The filter is a second-order filter
57 // The filter is a second-order filter
58 // The filter is a second-order filter
59 // The filter is a second-order filter
60 // The filter is a second-order filter
61 // The filter is a second-order filter
62 // The filter is a second-order filter
63 // The filter is a second-order filter
64 // The filter is a second-order filter
65 // The filter is a second-order filter
66 // The filter is a second-order filter
67 // The filter is a second-order filter
68 // The filter is a second-order filter
69 // The filter is a second-order filter
70 // The filter is a second-order filter
71 // The filter is a second-order filter
72 // The filter is a second-order filter
73 // The filter is a second-order filter
74 // The filter is a second-order filter
75 // The filter is a second-order filter
76 // The filter is a second-order filter
77 // The filter is a second-order filter
78 // The filter is a second-order filter
79 // The filter is a second-order filter
80 // The filter is a second-order filter
81 // The filter is a second-order filter
82 // The filter is a second-order filter
83 // The filter is a second-order filter
84 // The filter is a second-order filter
85 // The filter is a second-order filter
86 // The filter is a second-order filter
87 // The filter is a second-order filter
88 // The filter is a second-order filter
89 // The filter is a second-order filter
90 // The filter is a second-order filter
91 // The filter is a second-order filter
92 // The filter is a second-order filter
93 // The filter is a second-order filter
94 // The filter is a second-order filter
95 // The filter is a second-order filter
96 // The filter is a second-order filter
97 // The filter is a second-order filter
98 // The filter is a second-order filter
99 // The filter is a second-order filter
100 // The filter is a second-order filter

```



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-sinc 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

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

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:

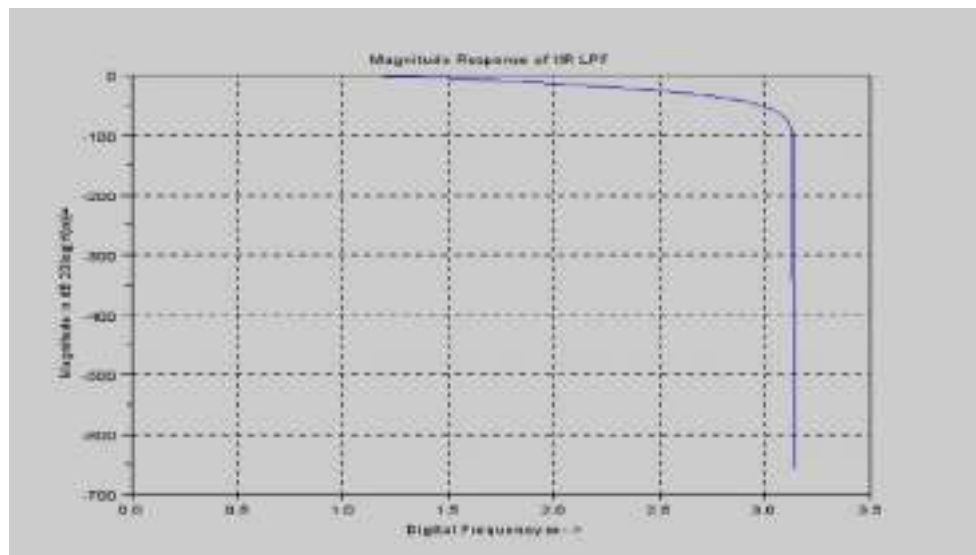
```
// Capt ion : To o b t a i n D i g i t a l I R Chebyshev low p a s s
filter
// Fr equency r e s p o n s e
clc ;
clear ;
close ;
fp= input ( ' Enter the pass band edge (Hz ) = ' );
fs= input ( ' Enter the stop band edge (Hz ) = ' );
kp= input ( ' Enter the pass band attenuation (dB) = ' );
ks= input ( ' 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 ( r a d / s a m p l e s )
wp =2* %pi *fp *1/ Fs;
ws =2* %pi *fs *1/ Fs;
```

```

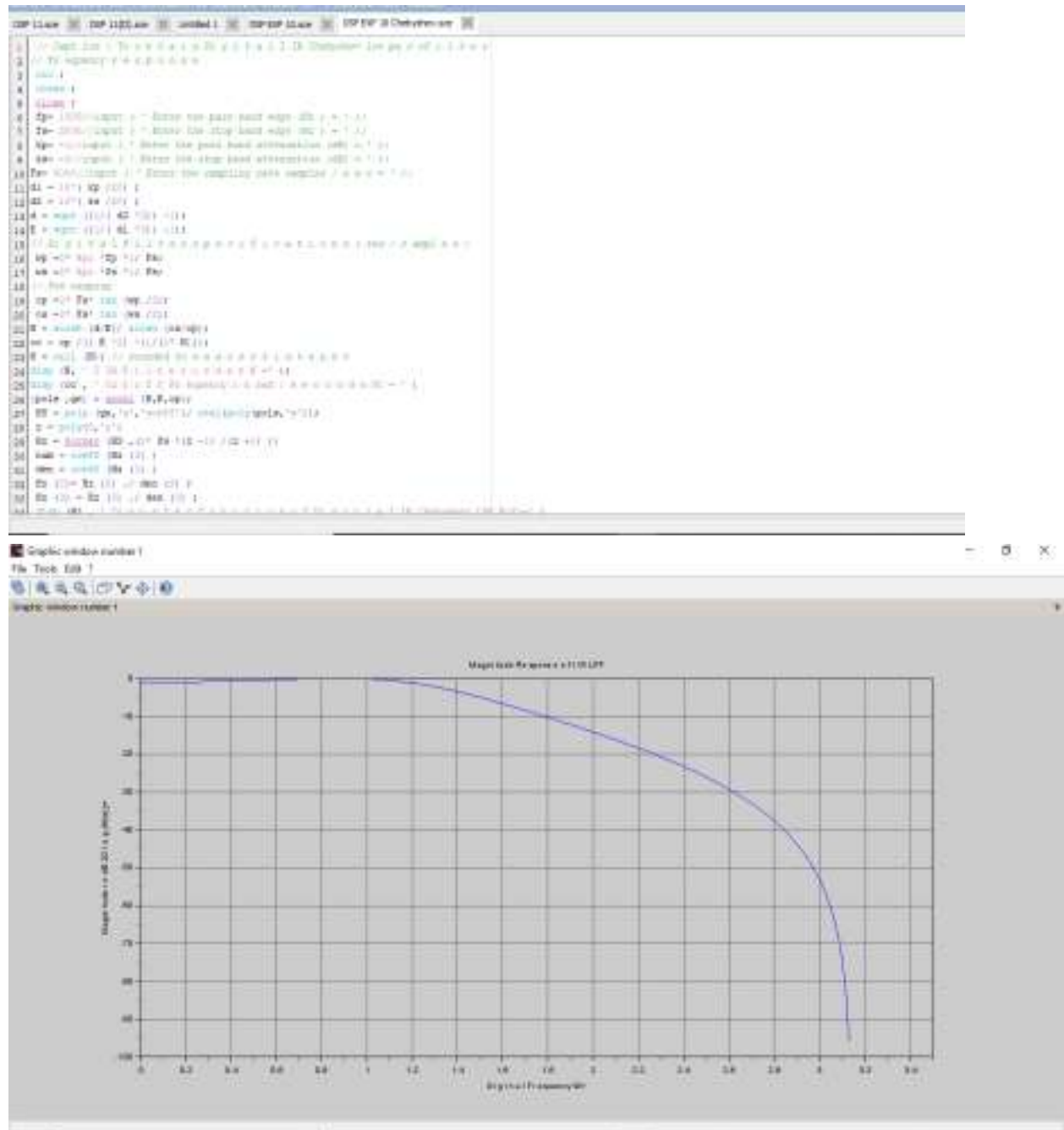
// Pre warping
op = 2* Fs* tan (wp /2);
os = 2* Fs* tan (ws /2);
N = acosh (d/E)/ acosh (os/op);
oc = op /(( E ^2) ^(1/(2* N)));
N = ceil (N); // rounded to nearest integer
disp (N, ' IIR Filter order N = ');
disp (oc, ' Cut off Frequency in rad / seconds OC = ')
[pols ,gn] = zpchl (N,E,op);
HS = poly (gn, 's', 'coeff')/ 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, ' Transfer function of Di git al IIR Chebyshev LPF H(Z)= '
) [Hw ,w] = frmag (Hz ,256) ;
figure (1)
plot (2*w*%pi ,20* log10 ( abs (Hw)));
xlabel ( ' Di git al Frequency W>' )
ylabel ( 'Magni tude i n dB 20 l o g jH(w) j= ' )
title ( 'Magni tude Re spons e o f IIR LPF ' )
xgrid (1)

```

Simulation Output



SCILAB CODE & SIMULATION:-



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 $|H(e^{j\omega})|$ in the pass band and stopband

PRE LAB ANS:-

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 $N=8,9,10$. **POST**

LAB ANS:-

RESULT:-

The design and analyzing the function of an analog Butterworth Filter and an analog Chebyshev Filter was done successfully using SCILAB.

Laboratory Report Cover Sheet

SRM
Institute of
Science and
Technology
Faculty of Engineering and Technology
Department of Electronics and Communication Engineering

18ECC204J DIGITAL SIGNAL PROCESSING Fifth Semester, 2020-21 (Odd semester)

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4th /FN

Venue :Online (Google Meet)

Title of Experiment :Design of Digital Butterworth filter

Date of Conduction : 28 SEP 21

Date of Submission : 9 OCTOBER 21

| Particulars | Max. Marks | Marks
Obtained |
|------------------------|-------------------|---------------------------|
| Prelab and Post lab | 10 | |
| Lab Performance | 10 | |
| Simulation and results | 10 | |
| Total | 30 | |

REPORT VEEVERIFICATION

Staff Name : Mrs. Vijayalakshmi & Mrs. A Anilatbala

Signature :

Experiment 11(a) Design of Digital Butterworth filter using Bilinear Transformation

Aim: To Design an digital Butterworth filter using Bilinear Transformation method

Software Requirement:SCILab

Theory: Butterworth filters are optimal in the sense of having a *maximally flat amplitude response*, as measured using a Taylor series expansion .The trivial filter $H(Z)=1$ has a perfectly flat amplitude response, but that's an all pass, not a low pass filter. Therefore, to constrain the optimization to the space of low pass filters, we need *constraints* on the design, such as $H(1)=1$ and $H(-1)=0$. That is, we may require the dc gain to be 1, and the gain at half the sampling rate to be 0.

It turns out Butterworth filters are much easier to design as *analog filters* which are then converted to digital filters. This means carrying out the design over the s plane instead of the z plane, where the z plane is the complex plane over which analog filter transfer functions are defined. The analog s transfer function $H_a(s)$ is very much like the digital transfer function $H(z)$, except that it is $j\omega$

interpreted relative to the analog frequency axis $s = j\omega_a$ (the ``axis") instead of the digital frequency axis $z = e^{j\omega_d}$ (the ``unit circle"). In particular, analog filter poles are stable if and only if they are all in the *left-half* of the plane, *i.e.*, their real parts are *negative*.

Digital Butterworth Filter using Bilinear Transformation

The bilinear s -transform is a mathematical transformation from the z -domain to the s -domain which preserves the frequency characteristics and is defined by:

$$s = \frac{2}{T} \frac{1 - z^{-1}}{1 + z^{-1}} \quad \text{where } T = \text{sampling period}$$

The bilinear transformation gives a non-linear relationship between analog ω_a

frequency ω_c and digital frequency ω_d

$$\omega_a = \frac{2}{T} \tan \frac{\omega_d T}{2}$$

Algorithm:

Step 1: From the given specifications, find prewarping analog frequencies, using the formula

$$\omega_a = \frac{2}{T} \tan \frac{\omega_d T}{2}$$

Step2: Using the analog frequencies, find $H_a(s)$ of the analog filter.

Step3: Select the sampling rate of the digital filter, Call it 'T' seconds per sample. Step 4: Substitute

$$s = \frac{2}{T} \frac{1 - z^{-1}}{1 + z^{-1}} \quad \text{where } T = \text{sampling period}$$

into the transfer function found in step 2.

Program:

Using Bilinear transformation, design a high pass filter ,monotonic in pass band with cutoff frequency of 1000 and down 10dB at 350 Hz.The sampling frequency is 5000Hz.

```
//To design digital Butterworth filter using bilinear transformation
clear all;
clc;
close;
ap=input('enter the value of ap in dB');
as=input('enter the value of as in dB Hz');
fp=input('enter the value of fp in Hz');
fs=input('enter the value of fs in Hz');
f=input('enter the value of f');
```

```
T=1/f;
wp=2*pi*fp;
ws=2*pi*fs;
op=2/T*tan(wp*T/2);
os=2/T*tan(ws*T/2);
```

```

N=log(sqrt((10^(0.1*as)-1)/(10^(0.1*ap)-1)))/log(op/os);
disp(ceil(N));
s=%s;
HS=1/(s+1);
oc=op;
HS1=horner(HS,oc/s);
disp(HS1,'Normalized transfer function,H(s)=');
z=%z;
HZ=horner(HS,(2/T)*(z-1)/(z+1));
disp(HZ,'H(z)=');

```

Simulation Output:

enter the value of ap in dB 3

enter the value of as in dB Hz 10

enter the value of fp in Hz 1000

enter the value of fs in Hz 350

enter the value of f 5000

1.

s

7265.4253 +s

"Normalized transfer function,H(s)="

1 +z

-9999 +10001z

"H(z)="

OUTPUT SIMULATION:-



Compute the z-transform of the digital filter by using the formula



Algorithm:

Step 1: From the given specifications, find prewarping analog frequencies, using the

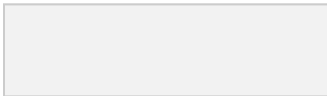
formula



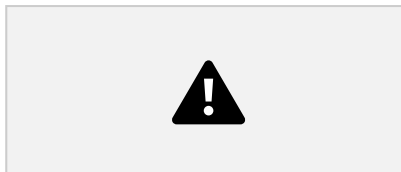
Step2: Using the analog frequencies, find $H_a(s)$ of the analog filter.

Step3: Select the sampling rate of the digital filter, Call it 'T' seconds per sample.

Step4: Express the analog transfer function as the sum of single pole filters:



Step5: Compute the z-transform of the digital filter by using the



formula

Program:

```
clear all;
```

```
clc ;
```

```
close ;
```

```

s=%s;
T=1;
HS=(2)/(s^2+3*s+2);
elts = pfss(HS);
disp(elts, 'Factorized HS=');

//The poles associated are 2 and 1
p1 = -2;
p2 = -1;

z=%z;
HZ=(2/(1 - %e^(p2*T)*z^(-1))) - (2/(1 - %e^(p1*T)*z^(-1)));

disp(HZ, 'HZ=');

```

Output Simulation:-



Prelab Questions:

1. What are the properties that are maintained same in the transfer of analog filter into a digital filter?
2. What is meant by impulse invariant method of designing IIR filter?
3. What are the properties of the bilinear transformation

PRE LAB ANSWER:-

Postlab Questions: (Scilab code and Output)

1. An analog filter has a transfer function $H(s) = \frac{10}{s^2 + 7s + 10}$. Design a digital filter equivalent to this using impulse invariant method for $T=0.2$ sec.

```
clear all;
clc;
close;
s=%s;
T=0.2;
HS=(10)/(s^2+7*s+10);
elts=pfss(HS);
disp(elts,'Factorized
HS=');
//The poles associated are P1 and P2 p1=-2; p2=-1;
z=%z;
HZ=(2/(1-%e^(p2*T)*z^(-1)))-(2/(1-%e^(p1*T)*z^(-1)));
disp(HZ,'HZ=');
```



2. For the analog transfer function, $H(s) = \frac{2}{s^2}$, determine $H(z)$ using bilinear transformation if (a) $T=1$ s.

♦♦+3♦♦+2

second (b) T=0.1 second.

```
clear all;
clc; close;
ap=input('enter the value of ap in dB');
as=input('enter the value of as in dB Hz');
fp=input('enter the value of fp in Hz');
fs=input('enter the value of fs in Hz'); f=input('enter the value of f');
T=1/f;
wp=2*%pi*fp;
ws=2*%pi*fs;
op=2/T*tan(wp*T/2);
os=2/T*tan(ws*T/2);
N=log(sqrt((10^(0.1*as)-1)/(10^(0.1*ap)-1)))/log(op/os);
disp(ceil(N));
s=%s;
HS=1/(s+1);
oc=op;
HS1=horner(HS,oc/s);
disp(HS1,'Normalized transfer function,H(s)=');
z=%z;
HZ=horner(HS,(2/T)*(z-1)/(z+1));
disp(HZ,'H(z)=');
```

Postlab Questions:

this using impulse invariant method for T=0.2 sec.

2. An analog filter has a transfer function $H(s) = 10^0$. Design a digital filter equivalent to

♦♦+7♦♦+10

3. For the analog transfer function, $H(s) = 2$, determine $H(z)$ using bilinear transformation if (a) z

♦♦+3♦♦+2

T=1 second (b) T=0.1 second.

POST LAB ANSWERS:-

Result:

The design of Butterworth filter using bilinear transformation and impulsive invariant method was performed using Scilab.

Laboratory Report Cover Sheet

| |
|---|
| SRM Institute of Science and Technology
College of Engineering and Technology
Department of Electronics and Communication Engineering |
| 18ECC204J DIGITAL SIGNAL PROCESSING
Fifth Semester, 2021-22 (Odd semester) |

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4th /FN

Venue : ONLINE(G -Meet)

**Experiment : To design chebyshev filter using bilinear and
. . impulse invariance method**

Date of Conduction : 12 OCT 2021

Date of Submission : 23 OCT 2021

| Particulars | Max. Marks | Marks Obtained |
|------------------------|------------|----------------|
| Pre lab and Post lab | 10 | |
| Lab Performance | 10 | |
| Simulation and results | 10 | |
| Total | 30 | |

REPORT VERIFICATION

**Staff Name : Mrs.D.VIJAYALAKSHMI
Mrs.A.ANILETBALA**

Signature :

Experiment 12: To design chebyshev filter using bilinear and impulse invariance method

Aim: To design chebyshev filter using bilinear and impulse invariance method using MATLAB

Theory:

The Chebyshev polynomials play an important role in the theory of approximation. The N^{th} -order Chebyshev polynomial can be computed by using

$$\begin{aligned} T_N(\Omega) &= \cos(N \cos^{-1}(\Omega)) \quad , \quad |\Omega| \leq 1 \\ &= \cosh(N \cosh^{-1}(\Omega)) \quad , \quad |\Omega| > 1. \end{aligned} \quad (1.1)$$

The first few Chebyshev polynomials are listed in Table 1, and some are plotted on Figure 1. Using $T_0(\Omega) = 1$ and $T_1(\Omega) = \Omega$, the Chebyshev polynomials may be generated recursively by using the relationship

$$T_{N+1}(\Omega) = 2\Omega T_N(\Omega) - T_{N-1}(\Omega), \quad (1.2)$$

$N \geq 1$. They satisfy the relationships:

1. For $|\Omega| \leq 1$, the polynomial magnitudes are bounded by 1, and they oscillate between ± 1 .
2. For $|\Omega| > 1$, the polynomial magnitudes increase monotonically with Ω .
3. $T_N(1) = 1$ for all n .
4. $T_N(0) = \pm 1$ for n even and $T_N(0) = 0$ for N odd.
5. The zero crossing of $T_N(\Omega)$ occur in the interval $-1 \leq \Omega \leq 1$.

| N | $T_N(\Omega)$ |
|---|-----------------------------|
| 0 | 1 |
| 1 | Ω |
| 2 | $2\Omega^2 - 1$ |
| 3 | $4\Omega^3 - 3\Omega$ |
| 4 | $8\Omega^4 - 8\Omega^2 + 1$ |

Table 1: Some low-order Chebyshev polynomials

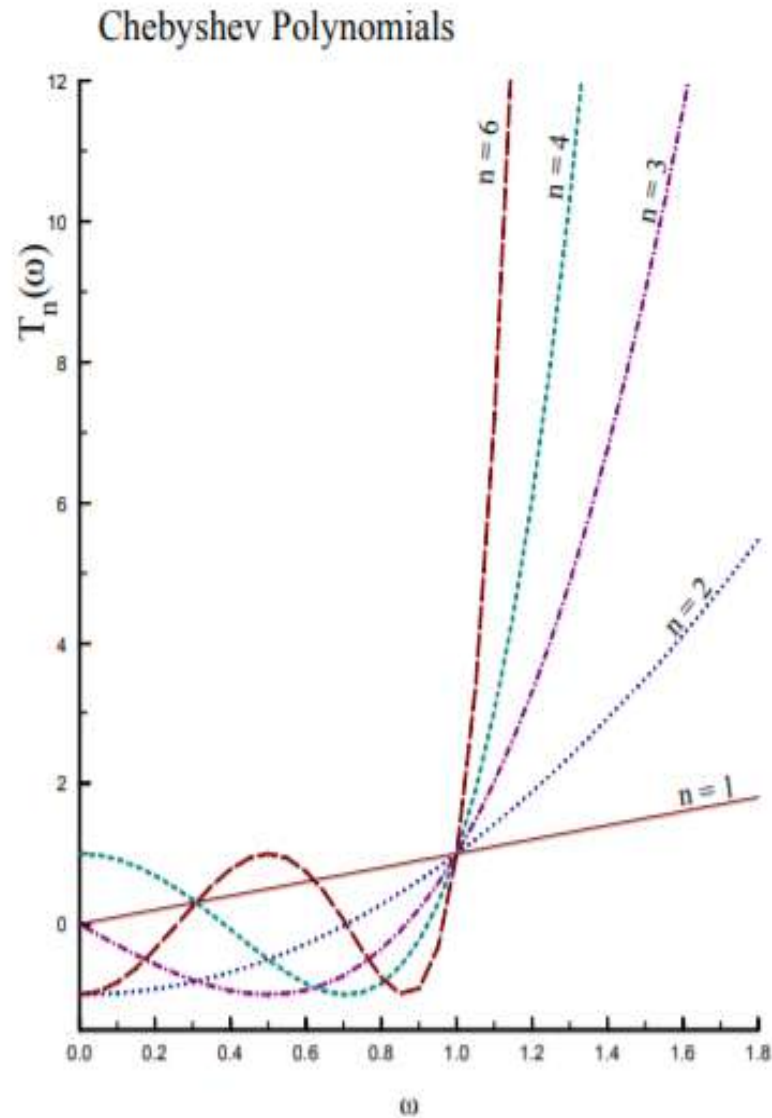


Figure 1: Some low-order Chebyshev polynomials.

Chebyshev Low-pass Filters

There are two types of Chebyshev low-pass filters, and both are based on Chebyshev polynomials. A *Type I* Chebyshev low-pass filter has an all-pole transfer function. It has an equi-ripple pass band and a monotonically decreasing stop band. A *Type II* Chebyshev low-pass filter has both poles and zeros; its pass-band is monotonically decreasing, and it has an equiripple stop band. By allowing some ripple in the pass band or stop band magnitude response, a Chebyshev filter can achieve a “steeper” pass- to stop-band transition region (*i.e.*, filter “roll-

off" is faster) than can be achieved by the same order Butterworth filter.

Type I Chebyshev Low-Pass Filter

A Type I filter has the magnitude response

$$|H_a(j\Omega)|^2 = \frac{1}{1 + \epsilon^2 T_N^2(\Omega/\Omega_p)}, \quad (1.3)$$

where N is the filter order, ϵ is a user-supplied parameter that controls the amount of pass-band ripple, and Ω_p is the upper pass band edge. Figure 2 depicts the magnitude response of several Chebyshev Type 1 filters, all with the same normalized pass-band edge $\Omega_p = 1$. As order N increases, the number of pass-band ripples increases, and the "roll – off" rate increases. For N odd (alternatively, even), there are $(N+1)/2$ (alternatively, $N/2$) pass-band peaks. As ripple parameter ϵ increases, the ripple amplitude and the "roll – off" rate increases.

On the interval $0 < \Omega < \Omega_p$, $T_N^2(\Omega/\Omega_p)$ oscillates between 0 and 1, and this causes $|H_a(\Omega/\Omega_p)|^2$ to oscillate between 1 and $1/(1 + \epsilon^2)$, as can be seen on Figure 2. In applications, parameter ϵ is chosen so that the peak-to-peak value of pass-band ripple

$$\text{Peak – to – Peak Passband Ripple} = 1 - 1/\sqrt{1 + \epsilon^2}. \quad (1.4)$$

is an acceptable value. If the magnitude response is plotted on a dB scale, the pass-band ripple becomes

$$\text{Pass – Band Ripple in dB} = 20\text{Log}\left[\frac{1}{1/\sqrt{1 + \epsilon^2}}\right] = 10\text{Log}(1 + \epsilon^2). \quad (1.5)$$

In general, pass-band ripple is undesirable, but a value of 1dB or less is acceptable in most

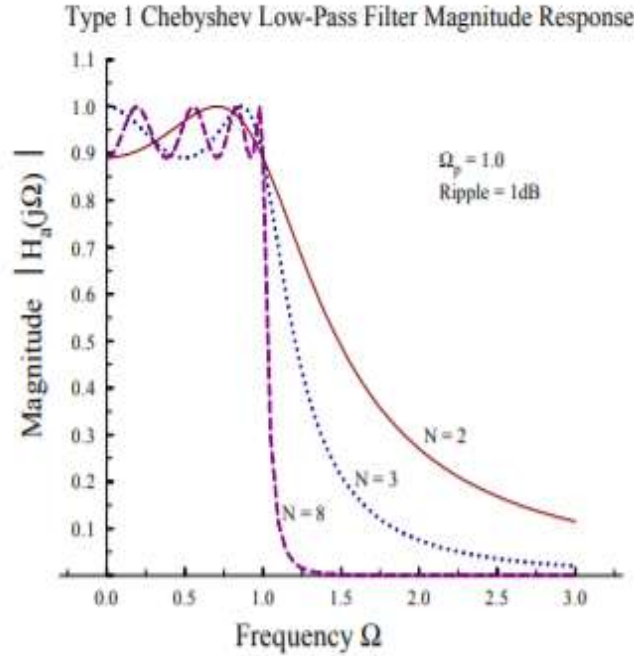


Figure 2: Type 1 Chebyshev magnitude response with one dB of pass-band ripple

applications.

The stop-band edge, Ω_s , can be specified in terms of a *stop-band attenuation parameter*. For $\Omega > \Omega_p$, the magnitude response decreases monotonically, and stop-band edge Ω_s can be specified as the frequency for which

$$\frac{1}{\sqrt{1 + \epsilon^2 T_N^2(\Omega / \Omega_p)}} < \frac{1}{A}, \quad \Omega > \Omega_s > \Omega_p, \quad (1.6)$$

where A is a user-specified *Stop-Band attenuation* parameter. In Decibels, for $\Omega > \Omega_s$, the magnitude response is down $20\text{Log}(A)$ dB or more from the pass band peak value.

Type 1 Chebyshev Low-Pass Filter Design Procedure

To start, we must have Ω_p , Ω_s , pass-band ripple value and the stop-band attenuation value. These are used to compute ϵ , N , and the pole locations for $H_a(s)$, as outlined below.

1) Using (1.5), compute

$$\varepsilon = \sqrt{10^{(\text{Pass-Band Ripple in dB})/10} - 1} \quad (1.7)$$

2) Compute the necessary filter order N . At $\Omega = \Omega_s$, we have

$$\frac{1}{\sqrt{1 + \varepsilon^2 T_N^2(\Omega_s / \Omega_p)}} = \frac{1}{A} \Rightarrow \sqrt{1 + \varepsilon^2 T_N^2(\Omega_s / \Omega_p)} = A, \quad (1.8)$$

which can be solved for

$$T_N(\Omega_s / \Omega_p) = \cosh(N \cosh^{-1}(\Omega_s / \Omega_p)) = \sqrt{\frac{A^2 - 1}{\varepsilon^2}}. \quad (1.9)$$

Finally, N is the smallest positive integer for which

$$N \geq \frac{\cosh^{-1} \sqrt{(A^2 - 1) / \varepsilon^2}}{\cosh^{-1}(\Omega_s / \Omega_p)}. \quad (1.10)$$

3) Compute the $2N$ poles of $H_a(s)H_a(-s)$. The first N poles are in the left-half s -plane, and they are assigned to $H_a(s)$. Using reasoning similar to that used in the development of the Butterworth filter, we can write

$$H_a(s)H_a(-s) = \frac{1}{1 + \varepsilon^2 T_N^2(s / j\Omega_p)}. \quad (1.11)$$

To simplify what follows, we will use $\Omega_p = 1$ and compute the pole locations for

$$H_a(s)H_a(-s) = \frac{1}{1 + \epsilon^2 T_N^2(s/j)} \quad (1.12)$$

Once computed, the pole values can be scaled (multiplied) by any desired value of Ω_p .

From inspection of (1.12), it is clear that the poles must satisfy

$$T_N(s/j) = \pm \sqrt{-1/\epsilon^2} = \pm j/\epsilon \quad (1.13)$$

(two cases are required here: the first where + is used and the second where – is used). Using (1.1), we formulate

$$\cos\left[N \cos^{-1}(s/j)\right] = \pm j/\epsilon \quad (1.14)$$

We must solve this equation for $2N$ distinct roots s_k , $1 \leq k \leq 2N$. Define

$$\cos^{-1}(s_k/j) = \alpha_k - j\beta_k \quad (1.15)$$

so that (1.14) yields

$$\cos\left[N(\alpha_k - j\beta_k)\right] = \cos(N\alpha_k) \cosh(N\beta_k) + j \sin(N\alpha_k) \sinh(N\beta_k) = \pm \frac{j}{\epsilon} \quad (1.16)$$

by using the identities $\cos(jx) = \cosh(x)$ and $\sin(jx) = j\{\sinh(x)\}$. In (1.16), equate real and imaginary components to obtain

$$\begin{aligned}\cos(N\alpha_k)\cosh(N\beta_k) &= 0 \\ \sin(N\alpha_k)\sinh(N\beta_k) &= \pm \frac{1}{\epsilon}.\end{aligned}\tag{1.17}$$

Since $\cosh(N\beta_k) \neq 0$, the first of (1.17) implies that $N\alpha_k$ must be an odd multiple of $\pi/2$ so that

$$N\alpha_k = (2k-1)\frac{\pi}{2} \quad \Rightarrow \quad \alpha_k = (2k-1)\frac{\pi}{2N}, \quad k = 1, 2, 3, \dots, 2N.\tag{1.18}$$

The α_k take on values that range from $\pi/2N$ to $2\pi - \pi/2N$ in steps of size $\pi/2N$. Since $\sin(N\alpha_k) = (-1)^{k-1}$, and the sign in the second (1.17) can be + or -, we can use

$$\beta_k = \frac{1}{N} \sinh^{-1}\left(\frac{1}{\epsilon}\right)\tag{1.19}$$

(all β_k are identical; the $2N$ distinct α_k will give us our $2N$ roots). Now, substitute (1.18) and (1.19) into (1.15) to obtain

$$s_k = j\cos(\alpha_k - j\beta_k) = \sigma_k + j\omega_k,\tag{1.20}$$

where

$$\begin{aligned}\sigma_k &= -\sin\left[(2k-1)\frac{\pi}{2N}\right]\sinh\left[\frac{1}{N}\sinh^{-1}\left(\frac{1}{\epsilon}\right)\right] \\ \omega_k &= \cos\left[(2k-1)\frac{\pi}{2N}\right]\cosh\left[\frac{1}{N}\sinh^{-1}\left(\frac{1}{\epsilon}\right)\right],\end{aligned}\tag{1.21}$$

$1 \leq k \leq 2N$, for the poles of $H_a(s)H_a(-s)$. The first half of the poles, s_1, s_2, \dots, s_n , are in the left-

half of the s-plane, while the remainder are in the right-half s-plane.

For $k = 1, 2, \dots, 2N$, the poles of $H_a(s)H_a(-s)$ are located on an s-plane ellipse, as illustrated by Figure 3 for the case $N = 4$. The ellipse has major and minor axes of length

$$d_2 = \cosh \left[\frac{1}{N} \sinh^{-1} \left(\frac{1}{\epsilon} \right) \right] \quad (1.22)$$

$$d_1 = \sinh \left[\frac{1}{N} \sinh^{-1} \left(\frac{1}{\epsilon} \right) \right],$$

respectively. To see that the poles fall on an ellipse, note that

$$\frac{\sigma_k^2}{d_1^2} + \frac{\omega_k^2}{d_2^2} = 1 \quad (1.23)$$

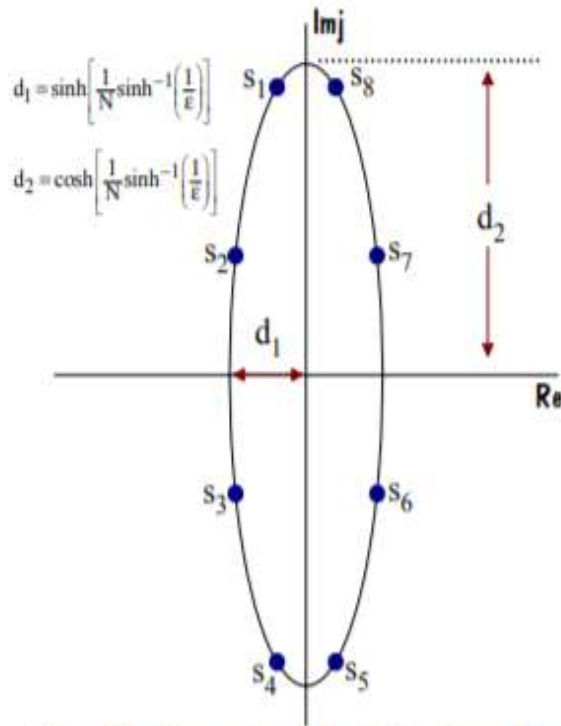


Figure 3: S-plane ellipse detailing poles of $H_a(s)H_a(-s)$ for a fourth-order ($N = 4$), Chebyshev filter with 1dB of passband ripple.

for $1 \leq k \leq 2N$.

4) Use only the left-half plane poles s_1, s_2, \dots, s_N , and write down the $\Omega_p = 1$ transfer function as

$$H_a(s) = K \frac{(-1)^N s_1 s_2 \cdots s_N}{(s - s_1)(s - s_2) \cdots (s - s_N)}, \quad (1.24)$$

where K is the filter DC gain. To obtain a peak pass-band gain of unity, we must use

$$\begin{aligned} K &= \frac{1}{\sqrt{1 + \epsilon^2}}, \quad N \text{ even} \\ &= 1, \quad N \text{ odd.} \end{aligned} \quad (1.25)$$

However, K can be set to any desired value, within technological constraints.

5) For a non-unity value of Ω_p , the transfer function becomes

$$H_a(s) = K \frac{(-1)^N s_1 s_2 \cdots s_N}{\left(\frac{s}{\Omega_p} - s_1 \right) \left(\frac{s}{\Omega_p} - s_2 \right) \cdots \left(\frac{s}{\Omega_p} - s_N \right)}. \quad (1.26)$$

Most hand calculators do not support hyperbolic trigonometry functions. It is desirable to develop non-hyperbolic-function forms for the Chebyshev filter formulas. To accomplish this, we use $\sinh^{-1}(x) = \ln\left(x + \sqrt{x^2 + 1}\right)$ to implement the simplification

$$\frac{1}{N} \sinh^{-1}\left(\frac{1}{\epsilon}\right) = \frac{1}{N} \ln\left(\frac{1 + \sqrt{1 + \epsilon^2}}{\epsilon}\right) = \ln(\Gamma), \quad (1.27)$$

where Γ is defined as

$$\Gamma \equiv \left[\frac{1 + \sqrt{1 + \epsilon^2}}{\epsilon} \right]^{1/N} . \quad (1.28)$$

In terms of parameter Γ , we can use (1.27) to write

$$\begin{aligned} \cosh \left(\frac{1}{N} \sinh^{-1} \left(\frac{1}{\epsilon} \right) \right) &= \cosh (\ln(\Gamma)) = \frac{e^{\ln(\Gamma)} + e^{-\ln(\Gamma)}}{2} = \frac{\Gamma + 1/\Gamma}{2} = \frac{\Gamma^2 + 1}{2\Gamma} \\ \sinh \left(\frac{1}{N} \sinh^{-1} \left(\frac{1}{\epsilon} \right) \right) &= \sinh (\ln(\Gamma)) = \frac{e^{\ln(\Gamma)} - e^{-\ln(\Gamma)}}{2} = \frac{\Gamma - 1/\Gamma}{2} = \frac{\Gamma^2 - 1}{2\Gamma} . \end{aligned} \quad (1.29)$$

Now, use (1.29) in the pole formulas (1.20) and (1.21) to obtain

$$s_k = -\sin \left[(2k-1) \frac{\pi}{2N} \right] \frac{\Gamma^2 - 1}{2\Gamma} + j \cos \left[(2k-1) \frac{\pi}{2N} \right] \frac{\Gamma^2 + 1}{2\Gamma} , \quad (1.30)$$

$1 \leq k \leq 2N$, for the $2N$ poles of $H_a(s)H_a(-s)$. Also, the poles are on an s -plane ellipse with major and minor axes

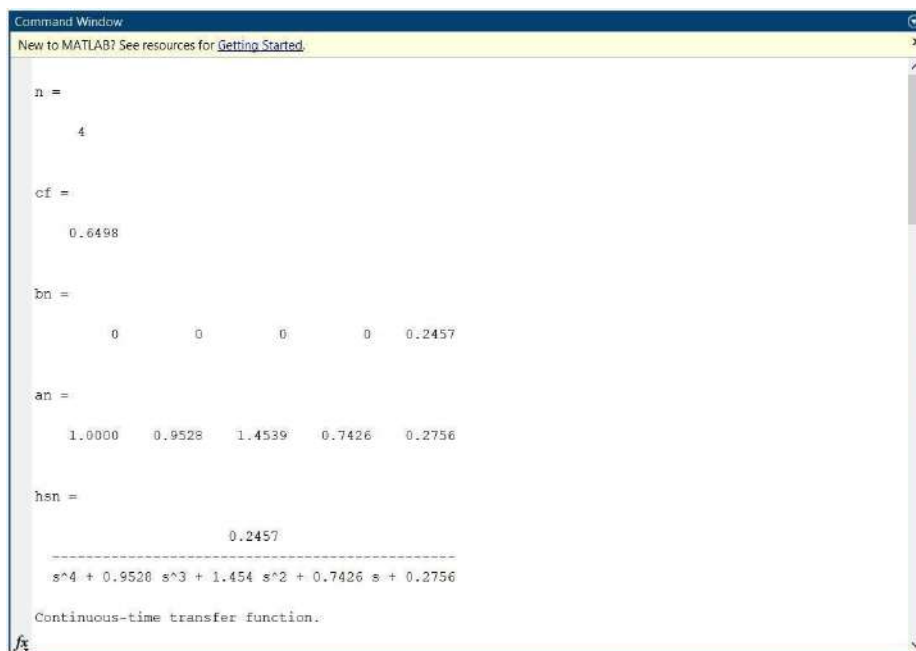
$$\begin{aligned} d_2 &= \cosh \left[\frac{1}{N} \sinh^{-1} \left(\frac{1}{\epsilon} \right) \right] = \frac{\Gamma^2 + 1}{2\Gamma} \\ d_1 &= \sinh \left[\frac{1}{N} \sinh^{-1} \left(\frac{1}{\epsilon} \right) \right] = \frac{\Gamma^2 - 1}{2\Gamma} , \end{aligned} \quad (1.31)$$

respectively.

PROGRAM

```
clear all;
clc;
ap= 1;
as= 15;
wp= 0.2*pi;
ws= 0.3*pi;
t=1;
op=(2/t)*tan(wp/2);
os=(2/t)*tan(ws/2);
[n,cf]=cheb1ord(op,os,ap,as,'s')
[bn,an]= cheby1(n,ap,1,'s')
hsn=tf(bn,an)
[b,a]= cheby1(n,ap,cf,'s')
hs=tf(b,a)
[num,den]=bilinear(b,a,1/t)
hz=tf(num,den,t)
w=0:pi/16:pi
hw=freqz(num,den,w)
hw_mag=abs(hw)
plot(w/pi,hw_mag)
```

Output :



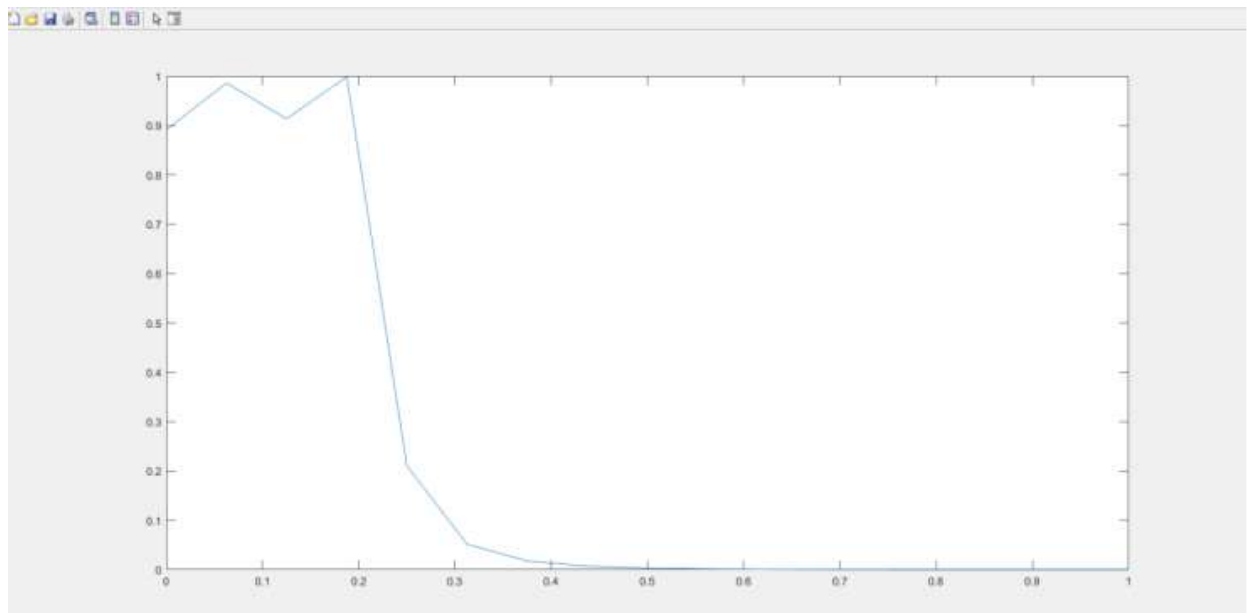
New to MATLAB? See resources for [Getting Started](#).

$$\frac{0.04381}{s^2 + 0.614s + 0.2038}$$

0.0018 0.0073 0.0110 0.0073 0.0018

en =

1.0000 -3.0543 3.8290 -2.2925 0.5507



PROGRAM

```
clc;
clear all;
close all;
T=1;
fs=1/T;
wp = input('enter the passband frequency ');
ws = input('enter the stopband frequency ');
rp = input('enter the pass band attenuation ');
rs = input('enter the stop band attenuation ');
%calculation of op and os in iiv method
op=wp/T;
os=ws/T;
display(op);
display(os);
[n,wc] = cheb1ord(op,os,rp,rs,'s');
display(n);
display(wc);
[z,p,k] = cheb1ap(n,rp);
display(z);
display(p);
display(k);
[b,a] = cheby1(n,rp,1,'low','s');
display(b);
display(a);
[bt,at] = lp2lp(b,a,wc);
display(bt);
display(at);
s= tf (bt,at);
display(s);
w=logspace(-10,10);
freqs(bt,at,w);
[bz,az] =impinvar(bt,at,fs);
display(bz);
display(az);
z=tf (bz,az,T,'variable','z^-1');
display(z);%required transfer function
w=0:pi/100:pi;
freqz(bz,az,w);
zplane(bz,az);%zero pole plot
fvtool(bz,az);%filter visualization technique
```

OUTPUT:

```
Command Window
enter the passband frequency 0.2*pi
enter the stopband frequency 0.3*pi
enter the pass band attenuation 1
enter the stop band attenuation 15

op =

    0.6283

os =

    0.9425

n =

     4

wc =

    0.6283

z =

    []

p =

fx -0.1395 + 0.9834i
```

```

Command Window

1.0000    0.5987    0.5740    0.1842    0.0430

|
s =

          0.03829
-----
s^4 + 0.5987 s^3 + 0.574 s^2 + 0.1842 s + 0.04296

Continuous-time transfer function.

bz =

    0.0000    0.0054    0.0181    0.0040

az =

    1.0000   -3.0591    3.8323   -2.2919    0.5495

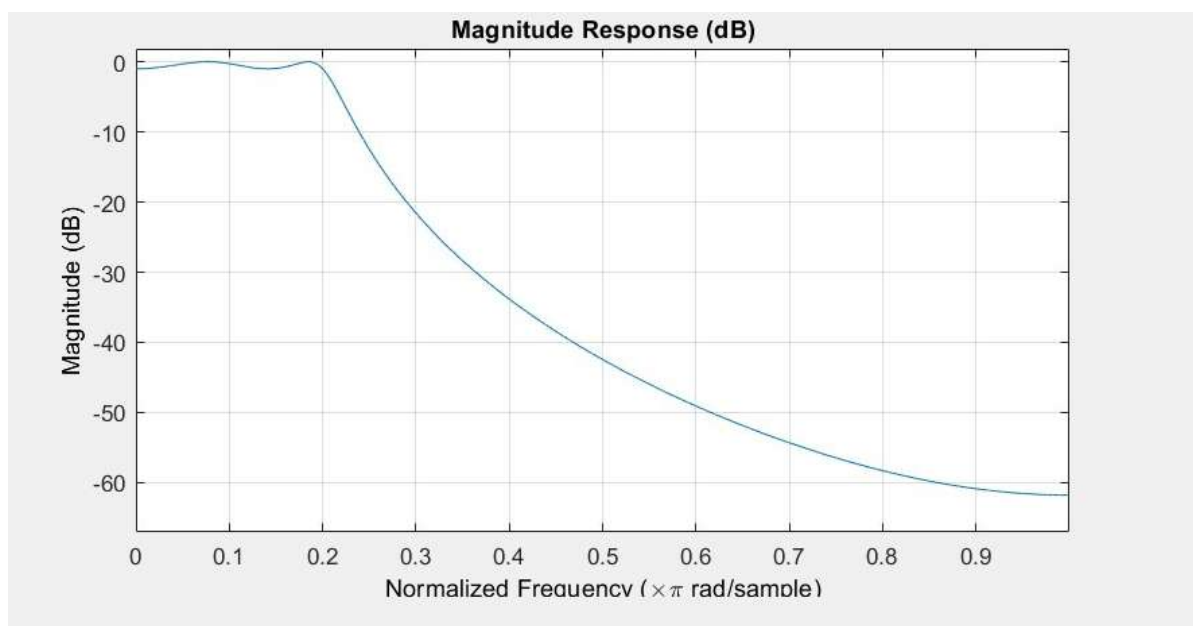
z =

1.388e-17 + 0.005373 z^-1 + 0.0181 z^-2 + 0.003985 z^-3
-----
1 - 3.059 z^-1 + 3.832 z^-2 - 2.292 z^-3 + 0.5495 z^-4

Sample time: 1 seconds
Discrete-time transfer function.

fx >>

```



SCI CODE AND SIMULATIONS:-



```

1
2 //CODE FOR CHEBYSHEV FILTER USING BILINEAR:
3
4 clear ;
5 clc ;
6 close ;
7 Wp = 0.2*pi;
8 Ws = 0.6*pi;
9 T = input ( "Sampling Interval in s: " );
10 OmegaP = (2/ T ) * tan ( Wp /2)
11 OmegaS = (2/ T ) * tan ( Ws /2)
12 Delta1 = input ( "Enter the Pass-Band Ripple: " );
13 Delta2 = input ( "Enter the Stop-Band Ripple: " );
14 Delta = sqrt ( (1/ Delta2 )^2 -1)
15 Epsilon = sqrt ( (1/ Delta ) ^2 -1)
16 N = ( acosh ( Delta / Epsilon ) ) / ( acosh ( OmegaS / OmegaP ) )
17 N = ceil ( N )
18 OmegaC = OmegaP / ( ( (1/ Delta ) ^2 -1) ^ (1/ (2* N ) ) )
19 [ poles , gn ] = zp2bilinear ( N , Epsilon , OmegaP )
20 Hs = poly ( gn , "s" , "coeff" ) / real ( poly ( poles , "s" ) )
21 z = poly ( 0 , "z" ) ;
22 Hz = hzorder ( Hs , (2/ T ) * ((z -1) / (z +1) ) ) ;
23 HW = fmag ( Hz (2) , Hz (3) , 512 ) ; // Frequency response for 512 points
24 W = 0: $pi / 511: $pi ;
25 a = qca ( ) ;
26 a . thickness = 1;
27 plot ( W/$pi , -abs(HW) , "r" )
28 a . foreground = 1;
29 a . font_style = 9;
30 xgrid (1)
31 xtitle ( " Magnitude Response of Chebyshev LPF Filter " , " Normalized Digital Frequency --> " , " Magnitude in dB " ) ;
32 disp (Hz , "H(z)=");
33
34

```


Sampling Interval

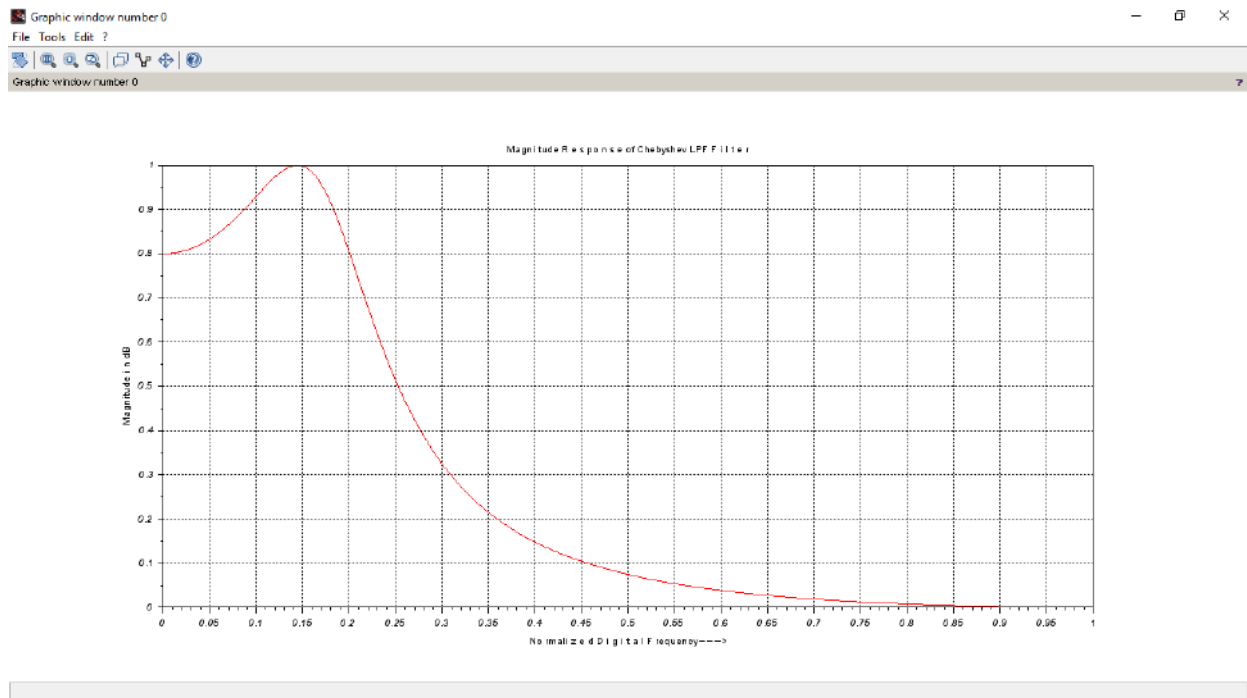
Enter the Pass Band Ripple 0.8

Enter the Stop Band Ripple 0.2

$$\frac{0.2815275 + 0.563055z + 0.2815275z^2}{3.2907261 - 7.2961813z + 5.4130926z^2}$$

"H(z) ="

-->



PRELAB:

- 1) What is chebyshev filter?
- 2) Give 2 difference between butterworth and chebyshev filter.

PRE LAB ANS:-

Exp-12

RA1911004010565
Poochal Dar.

Relab

1) What is chebyshev filter?

⇒ Chebyshev filter are analog or digital filters having a steeper roll-off than butterworth filters, and have pass band ripple (type 2) or stop band ripple (type 1).

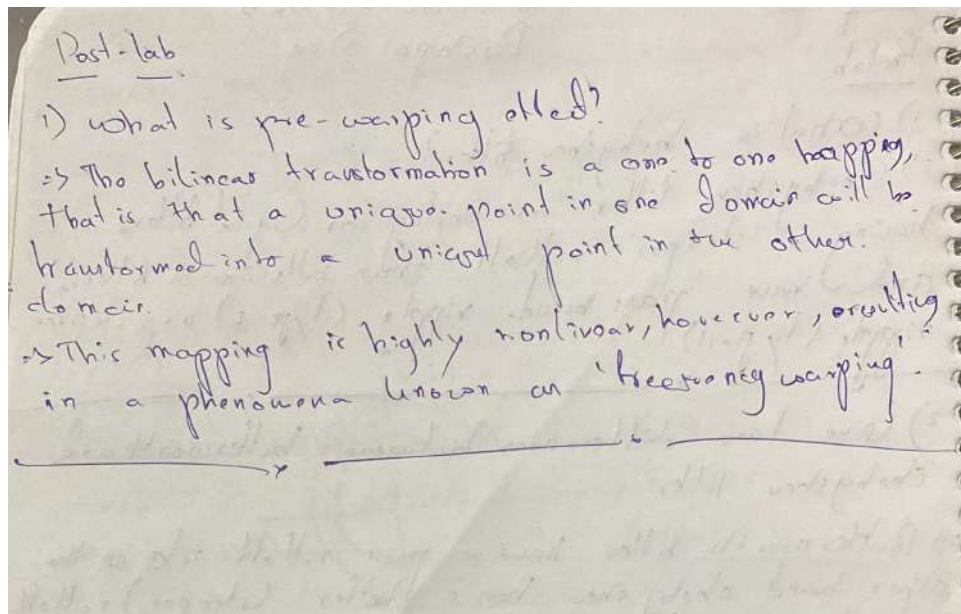
2) Give two differences between butterworth and chebyshev filter.

⇒ Butterworth filter have a poor roll-off rate on the other hand chebyshev has a better (steeper) roll-off rate because the ripple increases. Compared to a butterworth filter, a chebyshev-I filter achieves a sharper transition between the passband and the stop band with a lower order filter.

POSTLAB:

1) What is pre-warping effect?

POST LAB ANS:-



RESULT:

Thus, we design chebyshev filter using bilinear and impulse invariance method by using scilab and verified.

Laboratory Report Cover Sheet

| |
|---|
| SRM Institute of Science and Technology
College of Engineering and Technology
Department of Electronics and Communication Engineering |
| 18ECC204J DIGITAL SIGNAL PROCESSING
Fifth Semester, 2021-22 (Odd semester) |

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4th /FN

Venue : ONLINE(G -Meet)

Title of Experiment : 13A) INTERPOLATION IN TIME DOMAIN
13B) INTERPOLATION IN THE FREQUENCY DOMAIN

Date of Conduction : 12 OCT 2021

Date of Submission : 23 OCT 2021

| Particulars | Max. Marks | Marks Obtained |
|------------------------|------------|----------------|
| Pre lab and Post lab | 10 | |
| Lab Performance | 10 | |
| Simulation and results | 10 | |
| Total | 30 | |

REPORT VERIFICATION

Staff Name : Mrs.D.VIJAYALAKSHMI

Mrs.A.ANILETBALA

Signature :

EXPERIMENT 13

13a. INTERPOLATION IN TIME DOMAIN

Aim: To write code for interpolation of signal in SCILAB

The idea of interpolation is a very familiar concept to most of us and has its origin in numerical analysis. Typically, interpolation is performed on a table of numbers representing a mathematical function. Such a table may be printed in a handbook or stored in a computer memory device. The interpolation, often simply linear (or straight line) approximation, creates an error called the *interpolation error*. The main difference between interpolation in digital signal processing and interpolation in numerical

analysis is that we will assume that the given data is bandlimited to some band of frequencies and develop schemes that are optimal on this basis, whereas a numerical analyst typically assumes that the data consists of samples of polynomials (or very nearly so) and develops schemes to minimize the resulting error. To motivate this concept of interpolation in signal processing, it is helpful to think of an underlying (or original) analog signal $x_a(t)$ that was sampled to produce the given discrete signal $x(n)$. If the $x_a(t)$ was sampled at the minimum required rate, then, according to the sampling theorem, it can be recovered completely from the samples $x(n)$. If we now sample this recovered analog signal, at say twice the old rate, we have succeeded in doubling the sampling rate or interpolating by a factor of 2 with zero interpolation error.

The process of sampling rate conversion in the digital domain can be viewed as a linear filtering operation, as illustrated in Figure. The input signal $x(n)$ is characterized by the sampling rate $F_x = 1/T_x$, and the output signal $y(m)$ is characterized by the sampling rate $F_y = 1/T_y$, where T_x and T_y are the corresponding sampling intervals. In our treatment, the ratio F_y/F_x is constrained to be rational

$$\frac{F_y}{F_x} = \frac{I}{D}$$

Let $v(m)$ denote the intermediate sequence with a rate $F_y = IF_x$, which is obtained from $x(n)$ by adding $I - 1$ zeros between successive values of $x(n)$. Thus,

$$v(m) = \begin{cases} x(m/I), & m = 0, \pm I, \pm 2I, \dots \\ 0, & \text{otherwise} \end{cases}$$

and its sampling rate is identical to the rate of $v(m)$. The block diagram of the upsampler is shown in Figure. Again, any system containing the upsampler is a time-varying system

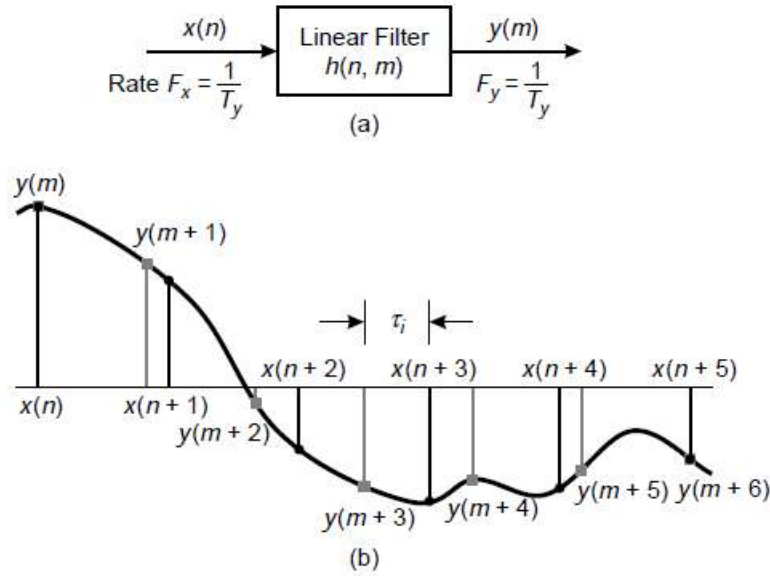


Figure1 : Sampling rate conversion views as a linear filtering process

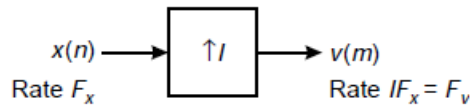


Figure 2: Interpolation by factor I

```
// PROGRAM FOR INTERPOLATION

// Program to do Interpolation of Signal

// clear work space variables

clf(); clc; clear; close;

// Set Initialization Variables

interp_fac = 2;

samp_freq = 100;

freq_cycles = 3;

time_index = 0:99;

// compute interpolation frequency with Sampling

inter_samp_freq = samp_freq*(interp_fac);

// Map time axis for before and after interpolation

time_axis = time_index/samp_freq;

new_time_index = 0:1/interp_fac:99;
```



```
new_time_axis = new_time_index / samp_freq;
```

```
// Define the input signal
```

```
inp_sig = sin(2*%pi*freq_cycles*time_axis);
```

```
// Do the interpolation using interp1 function on SCILAB
```

```
out_sig = interp1(time_axis,inp_sig,new_time_axis);
```

```
// Display the output
```

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

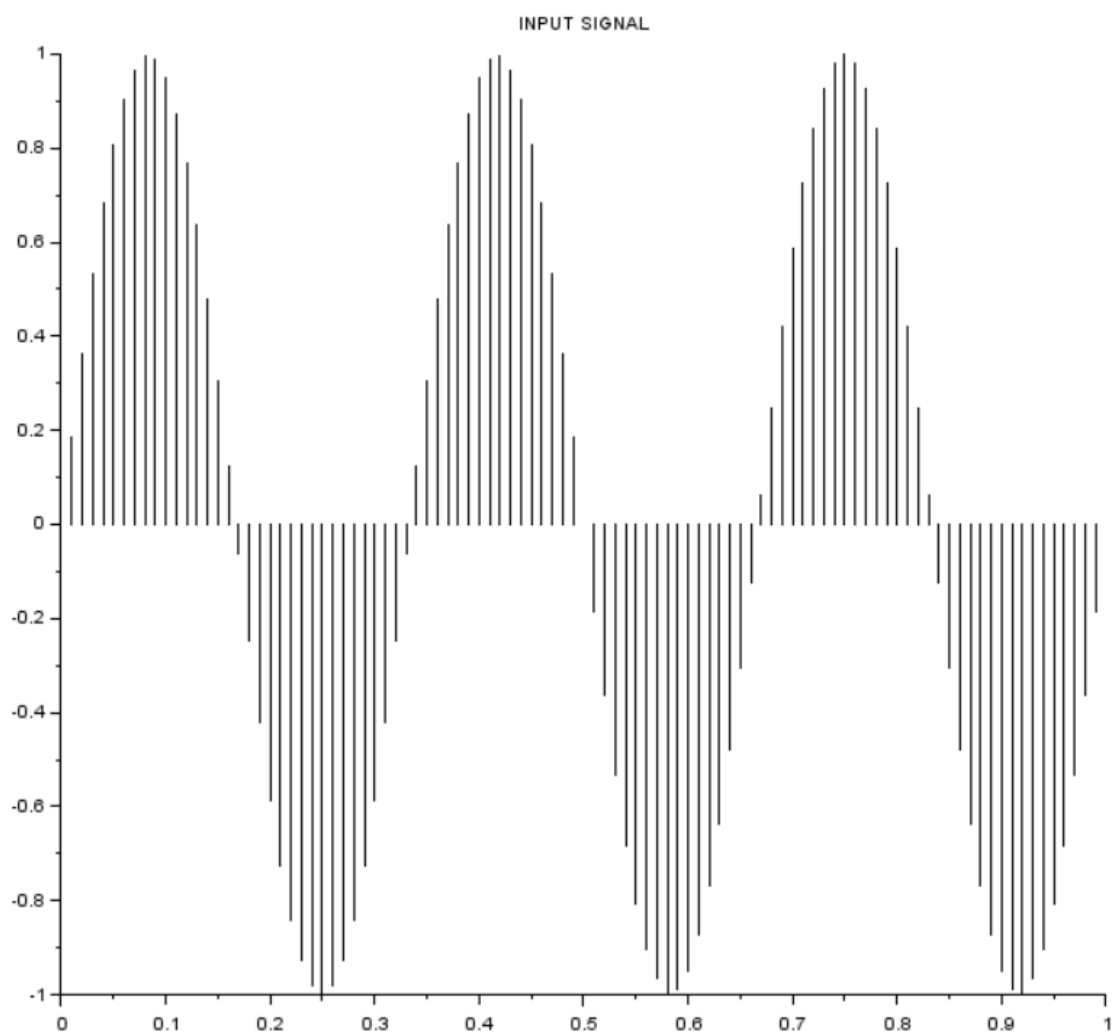
```
plot2d3(time_axis,inp_sig);
```

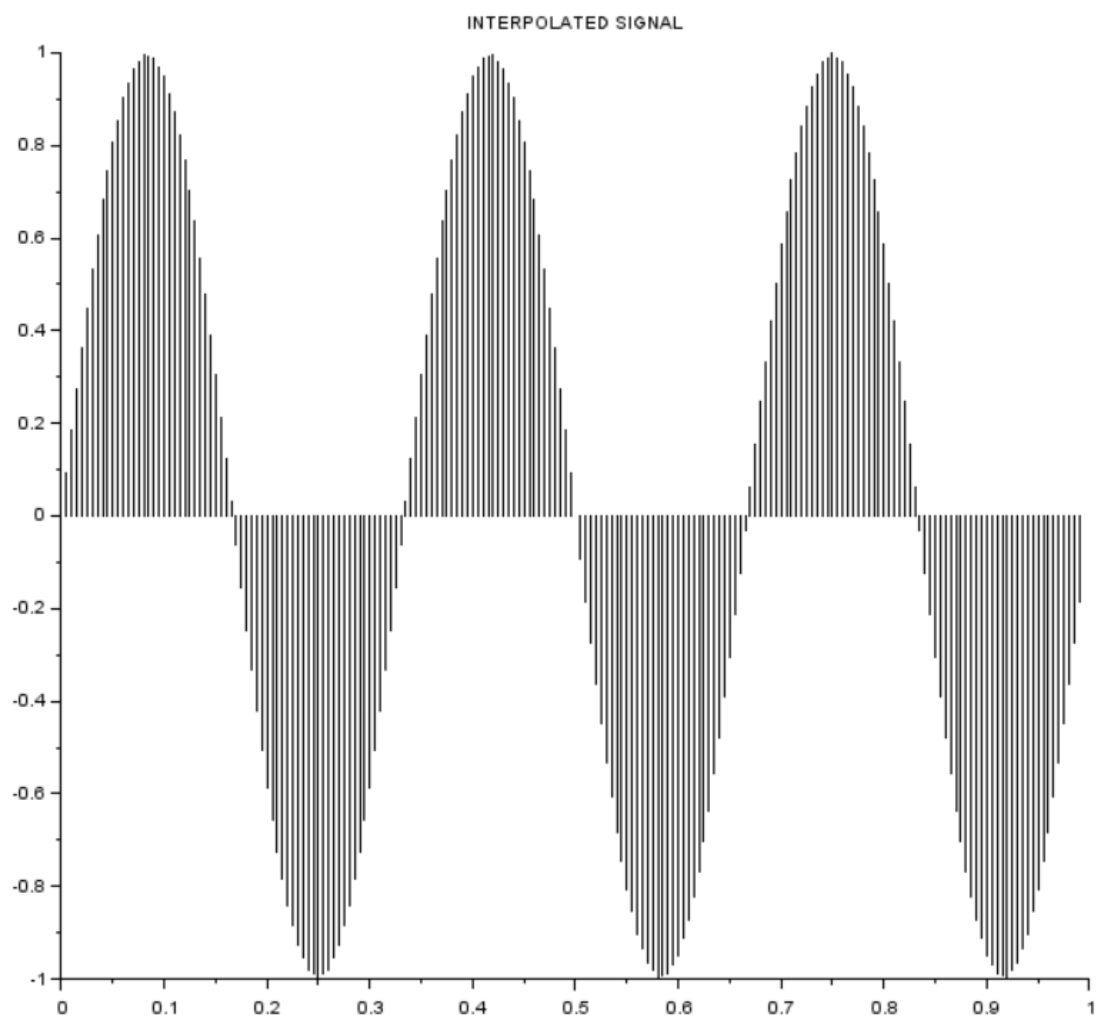
```
title('INPUT SIGNAL');
```

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

```
plot2d3(new_time_axis,out_sig);
```

```
title('INTERPOLATED SIGNAL')
```

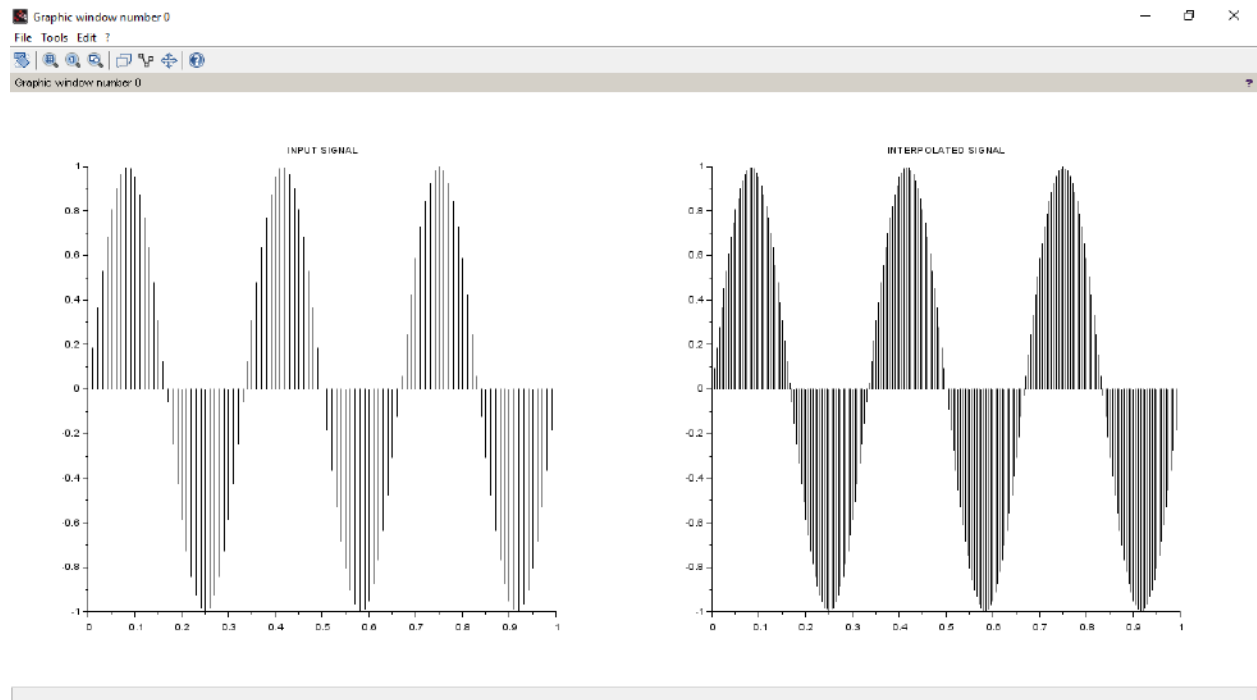




SCI CODE AND SIMULATION:-

```
DSP EXP 13.sce (C:\Users\ELCOT\3D Objects\DSP LAB\DSP EXP 13.sce) - SciNotes
File Edit Format Options Window Execute ?

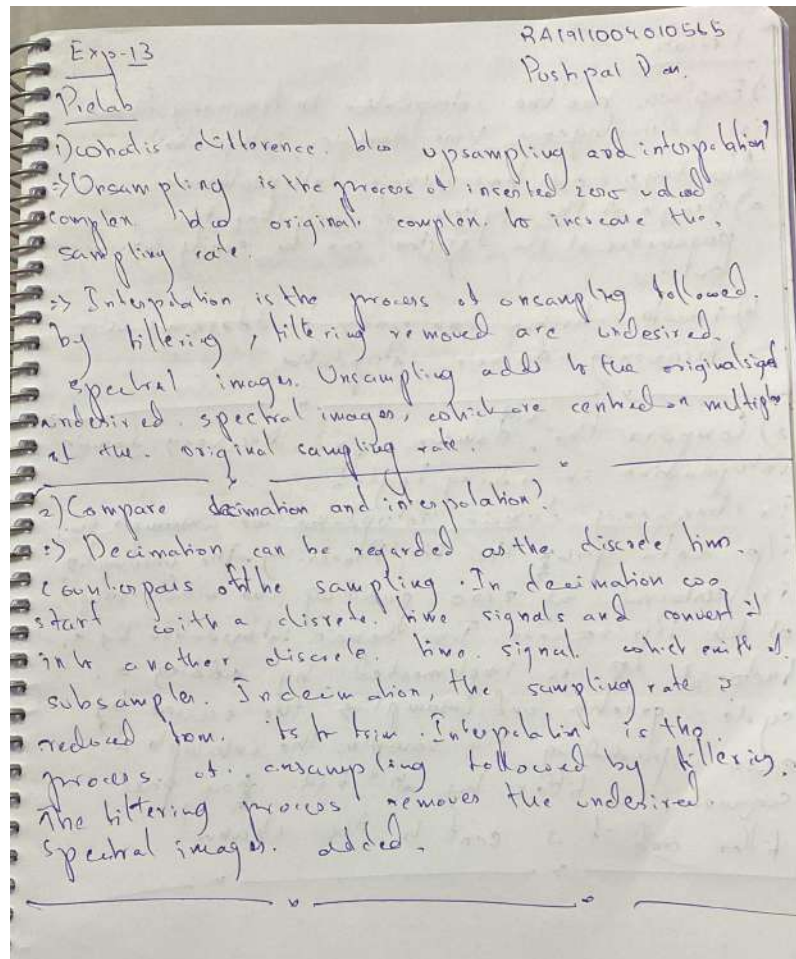
DSP EXP 13.sce (C:\Users\ELCOT\3D Objects\DSP LAB\DSP EXP 13.sce) - SciNotes
DSP EXP 12.sce DSP 12.sce DSP 12(1).sce Untitled 1 DSP EXP 13.sce
6
7 clf(); clf; clear; close;
8
9 //Set Initialization Variables
10
11 interp_fac = 2;
12
13 samp_freq = 100;
14
15 freq_cycles = 3;
16
17 time_index = 0:99;
18
19 // compute interpolation frequency with Sampling
20
21 inter_samp_freq = samp_freq*(interp_fac);
22
23 // Map time-axis for before and after interpolation
24
25 time_axis = time_index/samp_freq;
26
27 new_time_index = 0:1/interp_fac:99;
28
29 new_time_axis = new_time_index / samp_freq;
30
31 // Define the input signal
32
33 inp_sig = sin(2*pi*freq_cycles*time_axis);
34
35 // Do the interpolation using interp function on SCI LAB
36
37 out_sig = interp(time_axis,inp_sig,new_time_axis);
38
39 //Display the output
```



Pre lab

1. What is difference between Upsampling & Interpolation?
2. Compare Decimation & Interpolation?

PRE LAB ANS:-



Post lab

1. Modify the program to work with 4 Cycle Cos Signal and present the code & output?
2. Modify the above program for interpolation factor $I = 3$ and present the code & output?

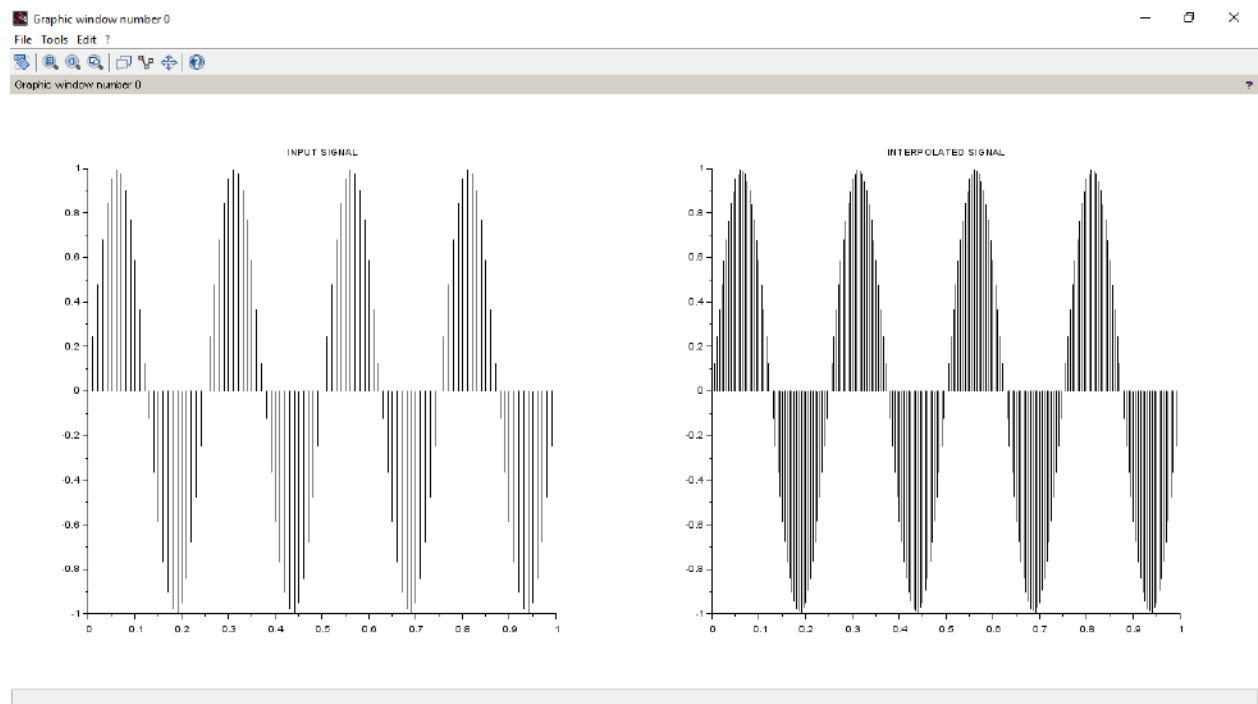
POST LAB:-

1.)

```

DSP EXP 13.sce (C:\Users\ELCOT\3D Objects\DSP LAB\DSP EXP 13.sce) - SciNotes
File Edit Format Options Window Execute ?
[Icons]
DSP EXP 13.sce DSP EXP 12.sce DSP 12(1).sce Untitled 1 DSP EXP 13.sce
7 clf(); clear; close;
8
9 //Set Initialization Variables
10
11 interp_fac = 2;
12
13 samp_freq = 100;
14
15 freq_cycles = 1;
16
17 time_index = 0:99;
18
19 // compute interpolation frequency with Sampling
20
21 inter_samp_freq = samp_freq*(interp_fac);
22
23 // Map time-axis for before and after interpolation
24
25 time_axis = time_index/samp_freq;
26
27 new_time_index = 0:1/interp_fac:1;
28
29 new_time_axis = new_time_index / samp_freq;
30
31 // Define the input signal
32
33 inp_sig = sin(2*pi*freq_cycles*time_axis);
34
35 // Do the interpolation using interp function on SCILAB
36
37 out_sig = interp(time_axis,inp_sig,new_time_axis);
38
39 //Display the output
40
Line 18, Column 18.

```



2.)

```

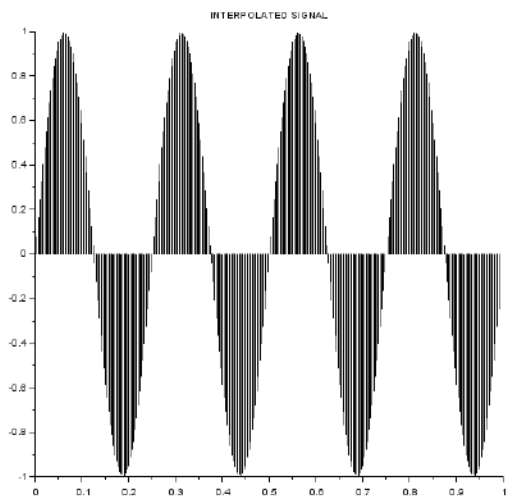
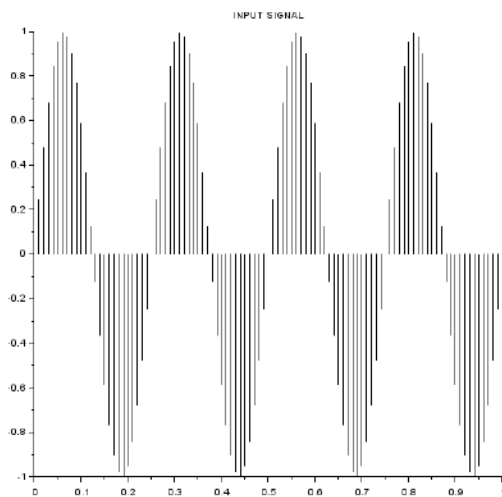
DSP EXP 13.sce (C:\Users\ELCOT\3D Objects\DSP LAB\DSP EXP 13.sce) - SciNotes
File Edit Format Options Window Execute ?
DSP EXP 13.sce (C:\Users\ELCOT\3D Objects\DSP LAB\DSP EXP 13.sce) - SciNotes
DSP EXP 12.sce DSP 12.sce DSP 12(1).sce Untitled 1 DSP EXP 13.sce
7 clc; clear; close;
8
9 //Set Initialization Variables
10
11 interp_fac = 3;
12
13 samp_freq = 100;
14
15 freq_cycles = 4;
16
17 time_index = 0:99;
18
19 // compute interpolation frequency with Sampling
20
21 inter_samp_freq = samp_freq*(interp_fac);
22
23 //Map time-axis for before and after interpolation
24
25 time_axis = time_index/samp_freq;
26
27 new_time_index = 0:1/interp_fac:5;
28
29 new_time_axis = new_time_index / samp_freq;
30
31 // Define the input signal
32
33 inp_sig = sin(2*pi*freq_cycles*time_axis);
34
35 // Do the interpolation using interp function on Scilab
36
37 out_sig = interp(time_axis,inp_sig,new_time_axis);
38
39 //Display the output
40
Line 11, Column 14.

```

Graphic window number 0

File Tools Edit ?

Graphic window number 0



RESULT:

Thus Interpolation in Time Domain is studied, simulated using Scilab and the desired output is obtained successfully.

13b. INTERPOLATION IN THE FREQUENCY DOMAIN

Aim: To write code for interpolation of signal in frequency domain using SCILAB

Using fourier transform we will be able to do interpolation using zero padding in the frequency domain which will give exact bandlimited interpolation in the frequency domain. The DFT of the band limited signals for the entire non zero portions of $x(n)$ extended by zeros yields exact interpolation of the complex spectrum.

Because fast fourier transform is so efficient, zero padding in FFT is highly practical method of interpolating spectra of finite duration signals and is used extensively in practice.

$$\omega_y = \frac{\omega_x}{I}$$

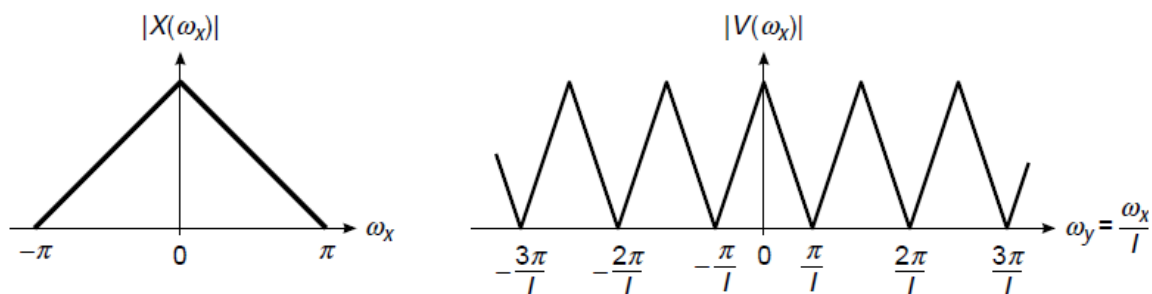


Figure 3: Frequency interpolation spectrum.

We observe that the sampling rate increase, obtained by the addition of $I - 1$ zero samples between successive values of $x(n)$, results in a signal whose spectrum is an I -fold periodic repetition of the input signal spectrum.

// Program to do interpolation in the frequency domain

```
clf();clc;clear;close;
```

// Input sinusoidal signal

```
samp_freq = 100;
```

```
sig_freq = 3;
```

```
interp_fac = 2;
```

// Time axis of the Sinusoidal Signal

```
time_index = 0:99;
```

```
time_axis = time_index/samp_freq;
```

```
inp_sig = sin(2*%pi*sig_freq*time_axis);
```

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

```
plot2d3(time_axis,inp_sig );
```

// Frequency domain interpolation using zero padding

// Transpose of the signal

```
inp_sig = inp_sig(:);
```

// Take fft of the signal

```
sig_fft = fft(inp_sig);
```

```
// Do zero padding in frequency domain for interpolation
```

```
sig_len = length(inp_sig);
```

```
// compute how many zeros needed using the interpolation  
formula (N*(l-1))
```

```
zeropad_len = round(sig_len*(interp_fac - 1));
```

```
// middle symmetry for splitting FFT in to two halves
```

```
mid_symmetry = ceil((sig_len+1)/2);
```

```
// Do zero padding here
```

```
frq_interp_sig = [sig_fft(1:mid_symmetry); zeros(zeropad_len, 1);  
sig_fft(mid_symmetry+1:$)];
```

```
// Take inverse fourier transform
```

```
time_interp_sig = interp_fac* real(ifftr(frq_interp_sig));
```

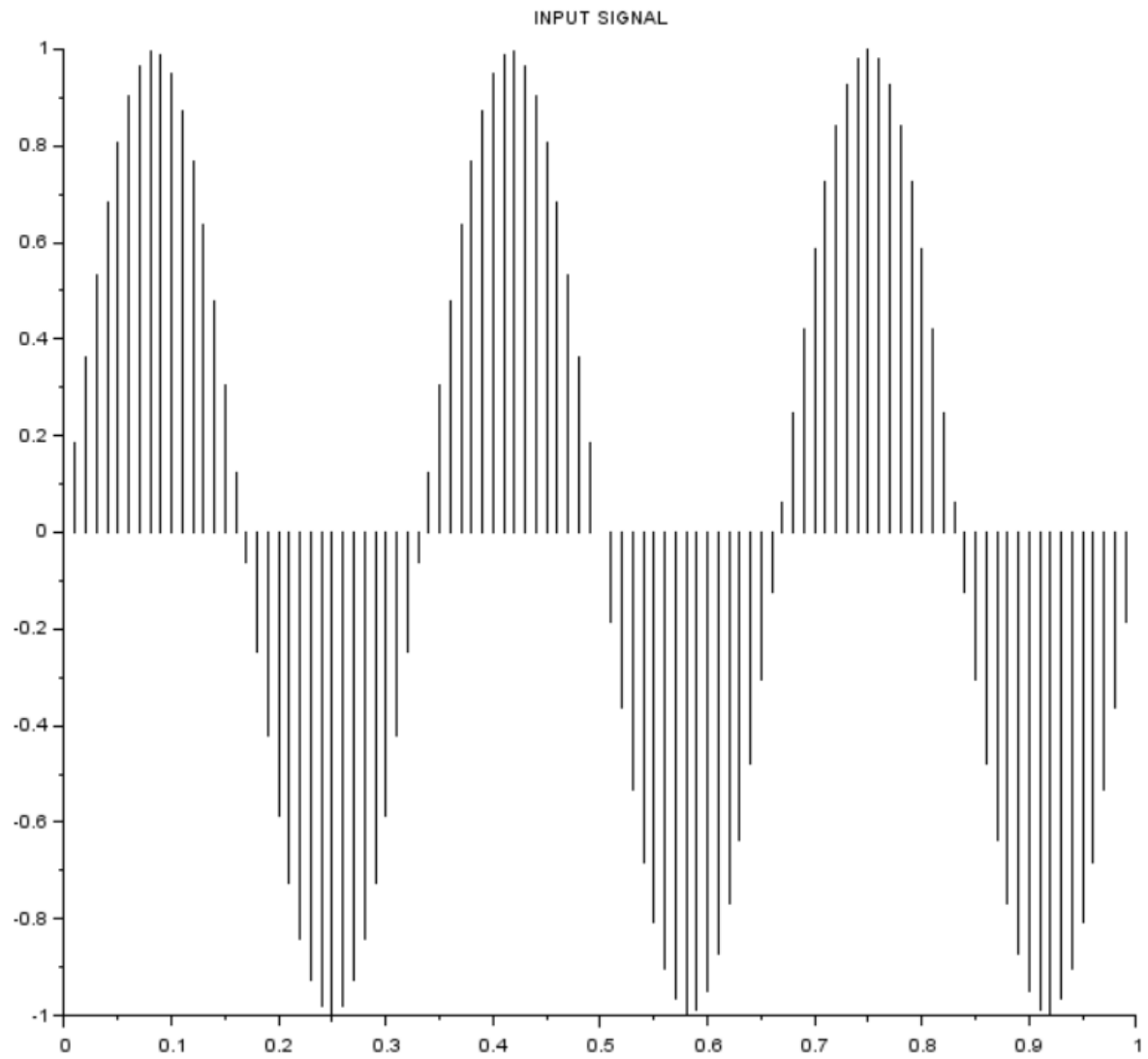
```
interp_sig_len = length(time_interp_sig);
```

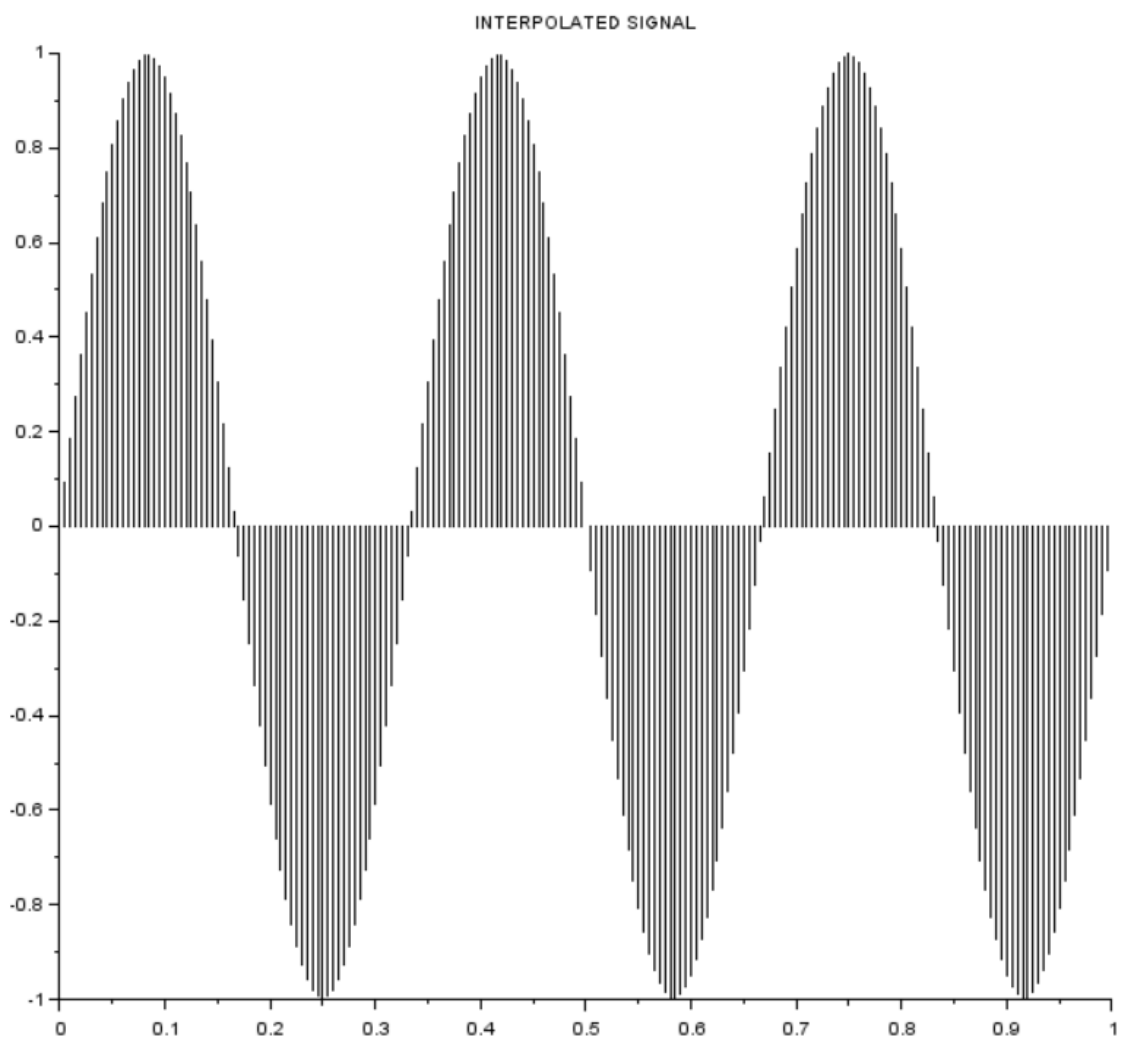
```
// compute new time axis
```

```
new_time_axis = (0:interp_sig_len-1)/(samp_freq*interp_fac);
```

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

```
plot2d3(new_time_axis, time_interp_sig)
```

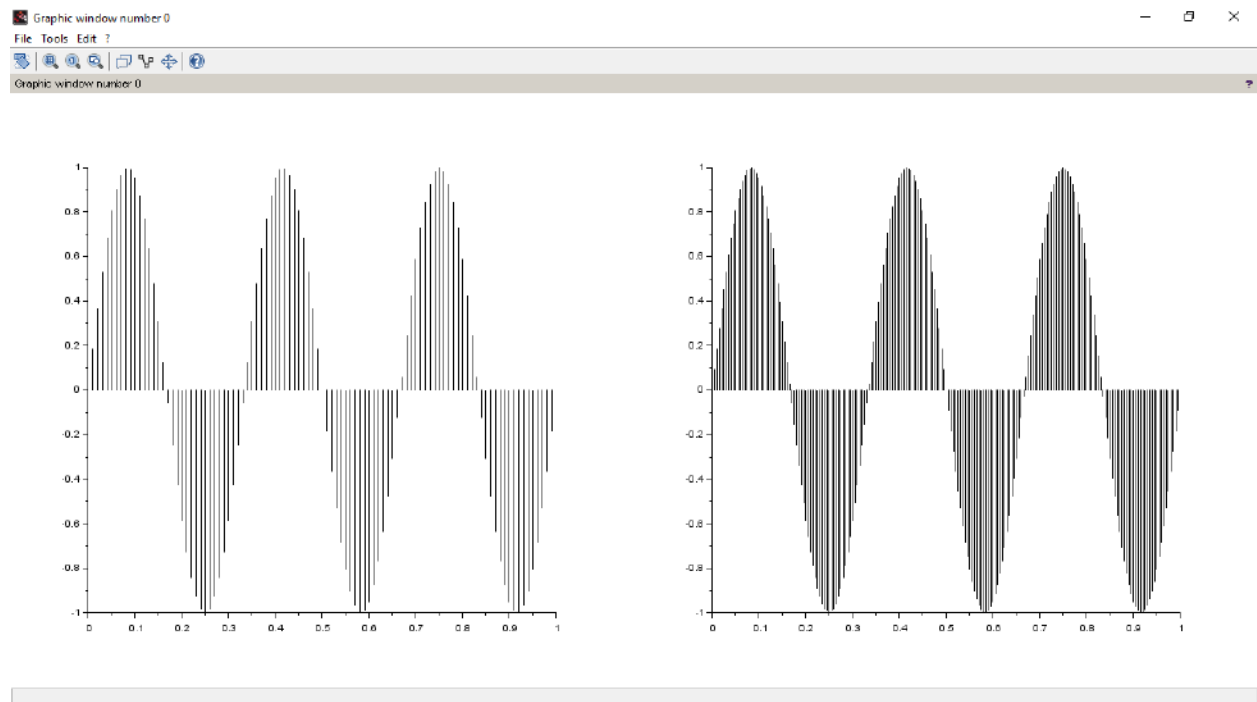




SCILAB CODE & SIMULATION:-

```
DSP EXP 13 frequency domain.sce (C:\Users\ELCOT\3D Objects\DSP LAB\DSP EXP 13 frequency domain.sce) - SciNotes
File Edit Format Options Window Execute ?
DSP EXP 13 frequency domain.sce DSP EXP 12.sce DSP 12(1).sce Untitled 1 DSP EXP 13.sce DSP EXP 13 frequency domain.sce
1 //Program to do interpolation in the frequency domain.
2
3 clf();clc;clear;close;
4
5 // Input sinusoidal signal
6
7 samp_freq = 100;
8
9 sig_freq = 3;
10
11 interp_fac = 2;
12
13 // Time axis of the Sinusoidal Signal
14
15 time_index = 0:99;
16
17 time_axis = time_index/samp_freq;
18
19 inp_sig = sin(2*pi*sig_freq*time_axis);
20
21 subplot(1,2,1)
22
23 plot3d3(time_axis,inp_sig);
24
25 // Frequency domain interpolation using zero padding
26
27 // Transpose of the signal
28
29 inp_sig = inp_sig';
30
31 // Take fft of the signal
32
33 sig_fft = fft(inp_sig);
34
35
```

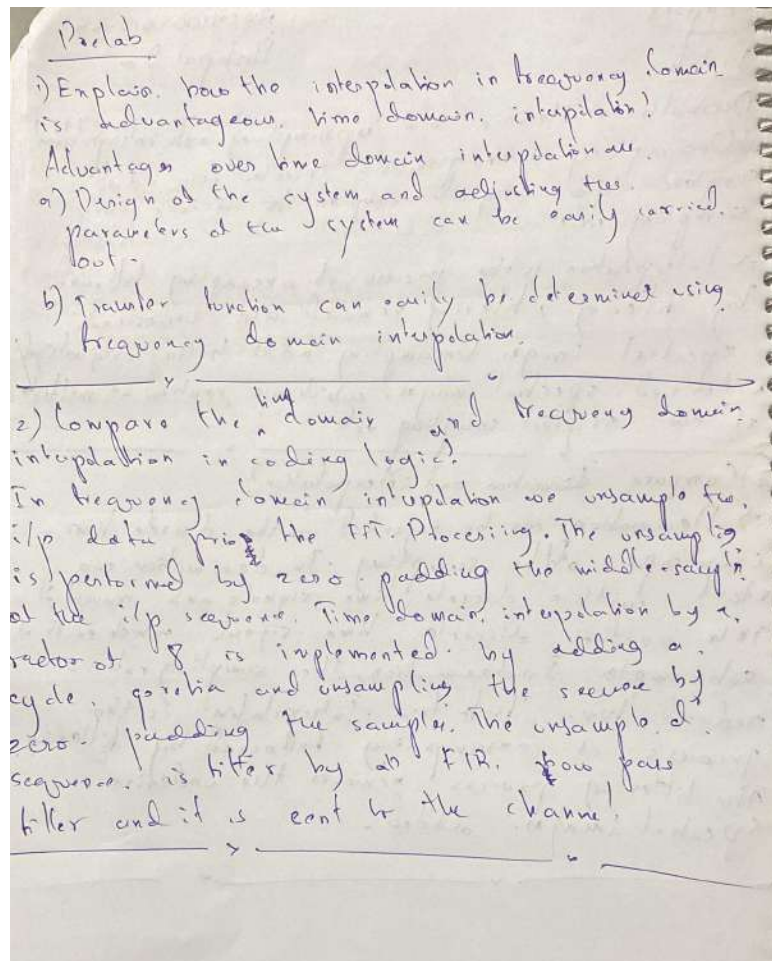
Line 3, Column 22.



PRE LAB:

1. Explain how the interpolation in frequency domain is advantageous over time domain interpolation?
2. Compare time domain & freq domain interpolation in coding logic?

PRE LAB ANS:-

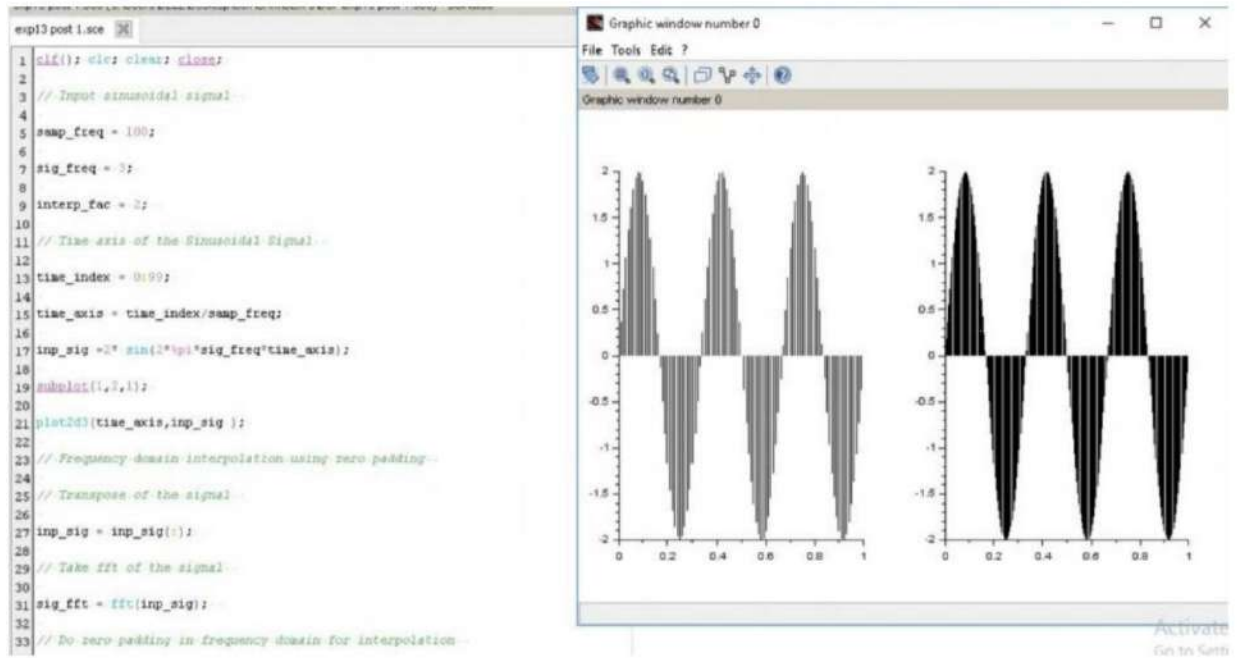


POST LAB:

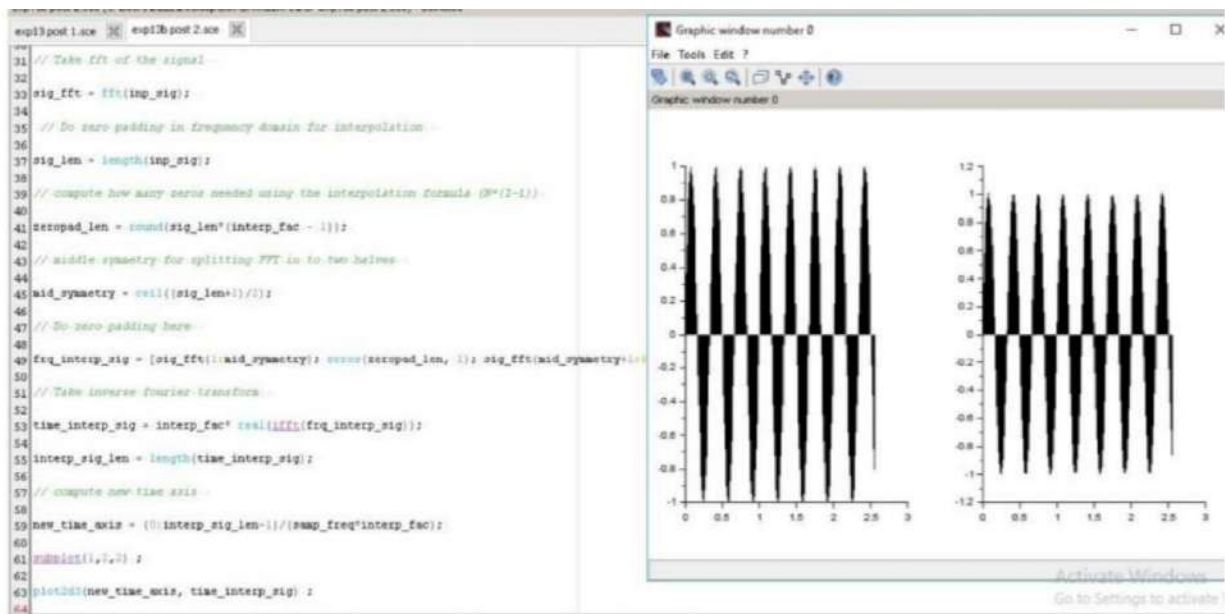
1. For the given code vary the length of the FFT to 256 and compute the output?
2. Change the amplitude of the output of the given code to $[-2 \ 2]$?

POST LAB ANS:-

1.)



2.)



RESULT:-

Thus, Interpolation in Frequency Domain is studied ,simulated using Scilab and the desired output is obtained successfully.

Laboratory Report Cover Sheet

SRM Institute of Science and Technology
Faculty of Engineering and Technology
Department of Electronics and Communication Engineering

18ECC204J DIGITAL SIGNAL PROCESSING
Fifth Semester, 2020-21 (Odd semester)

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 4TH / FN

Venue : Online(G Meet)

Title of Experiment : 14a. DECIMATION IN TIME DOMAIN
14b. DECIMATION IN THE FREQUENCY DOMAIN

Date of Conduction : 12 OCT 2021

Date of Submission : 23 OCT 2021

| Particulars | Max. Marks | Marks Obtained |
|------------------------|------------|----------------|
| Pre lab and Post lab | 1
0 | |
| Lab Performance | 1
0 | |
| Simulation and results | 1
0 | |
| Total | 30 | |

REPORT VERIFICATION

Staff Name : Mrs.D.VIJAYALAKSHMI

Mrs.A.ANILETBALA

Signature :

EXPERIMENT 14

14a. DECIMATION IN TIME DOMAIN

Aim: To write code for decimation of signal in SCILAB

In digital signal processing, downsampling, compression, and decimation are terms associated with the process of resampling in a multi-rate digital signal processing system. Decimation is a term that historically means the removal of every tenth one. But in signal processing, decimation by a factor of 10 actually means keeping only every tenth sample. This factor multiplies the sampling interval or, equivalently, divides the sampling rate. For example, if compact disc audio at 44,100 samples/second is decimated by a factor of 5/4, the resulting sample rate is 35,280. A system component that performs decimation is called a decimator. Decimation by an integer factor is also called compression.

Rate reduction by an integer factor M can be explained as a two-step process, with an equivalent implementation that is more efficient

1. Reduce high-frequency signal components with a digital [lowpass filter](#).
2. *Decimate* the filtered signal by M ; that is, keep only every M^{th} sample.

Step 2 alone allows high-frequency signal components to be misinterpreted by subsequent users of the data, which is a form of distortion called [aliasing](#). Step 1, when necessary, suppresses aliasing to an acceptable level. In this application, the filter is called an [anti-aliasing filter](#), and its design is discussed below.

When the anti-aliasing filter is an [IIR](#) design, it relies on feedback from output to input, prior to the second step. With [FIR filtering](#), it is an easy matter to compute only every M^{th} output. The calculation performed by a decimating FIR filter for the n^{th} output sample is a dot product.

$$y[n] = \sum_{k=0}^{K-1} x[nM - k] \cdot h[k],$$

where the $h[\bullet]$ sequence is the impulse response, and K is its length. $x[\bullet]$ represents the input sequence being downsampled. In a general purpose processor, after computing $y[n]$, the easiest way to compute $y[n+1]$ is to advance the starting index in the $x[\bullet]$ array by M , and recompute the dot product. In the case $M=2$, $h[\bullet]$ can be designed as a **half-band filter**, where almost half of the coefficients are zero and need not be included in the dot products.

Impulse response coefficients taken at intervals of M form a subsequence, and there are M such subsequences (phases) multiplexed together. The dot product is the sum of the dot products of each subsequence with the corresponding samples of the $x[\bullet]$ sequence. Furthermore, because of downsampling by M , the stream of $x[\bullet]$ samples involved in any one of the M dot products is never involved in the other dot products. Thus M low-order FIR filters are each filtering one of M multiplexed *phases* of the input stream, and the M outputs are being summed. This viewpoint offers a different implementation that might be advantageous in a multi-processor architecture. In other words, the input stream is demultiplexed and sent through a bank of M filters whose outputs are summed. When implemented that way, it is called a **polyphase filter**.

```
// Program to do Decimation of Signal
// clear work space variables
```

```
clearall
```

```
N=50;
```

```
n=0:1:N-1;
```

```
x=sin(2*%pi*n/20)+sin(2*%pi*n/15)
```

```
M=2;
```

```
x1=x(1:M:N);
```

```
n1=1:1:N/M;
```

```
subplot(2,1,1),plot2d3(n,x)
```

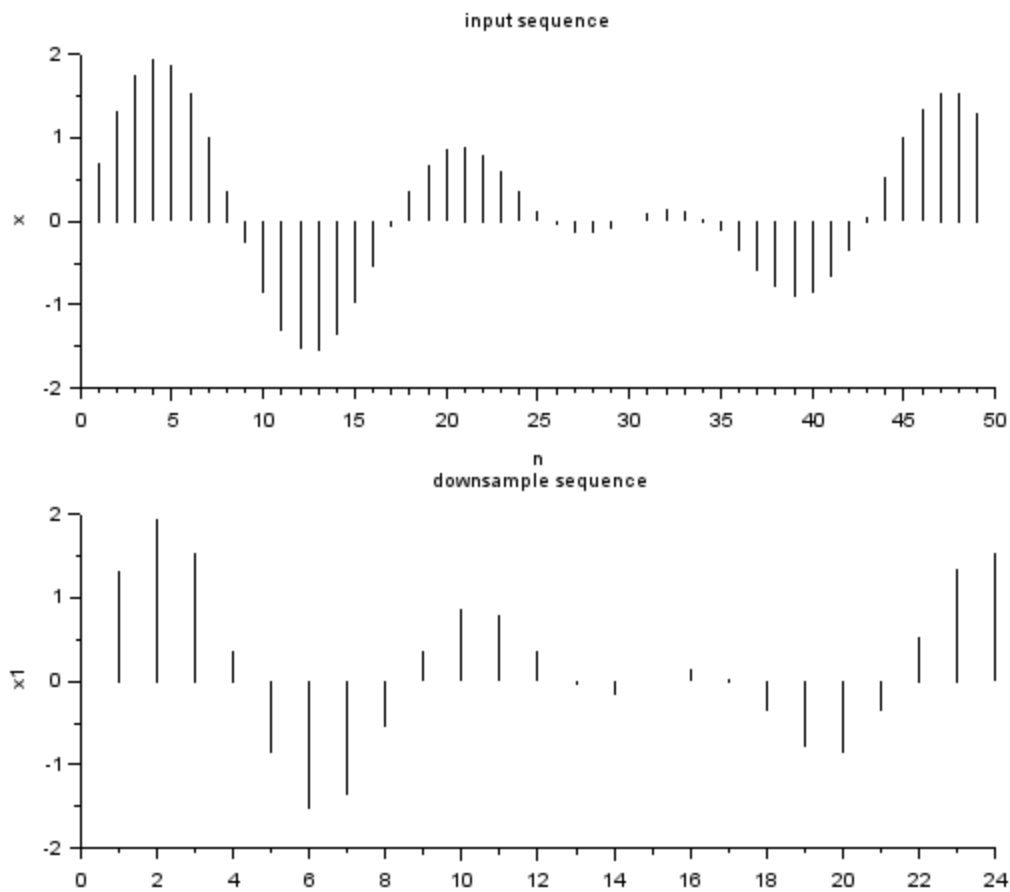
```
xlabel('n'),ylabel('x')
```

```
title('input sequence')
```

```
subplot(2,1,2),plot2d3(n1-1,x1)
```

```
xlabel('n'),ylabel('x1')
```

```
title('downsample sequence')
```



SCILAB CODE AND SIMULATION:-

```
DSP EXP 14 dcimation of the signal.sce  DSP EXP 14 dcimation in fre domain.sce
1  //.Program.to.do.Decimation.of.Signal.
2  //.clear.work.space.variables.
3
4  clear.all
5  N=50;
6  n=0:1:N-1;
7  x=sin(2*%pi*n/20)+sin(2*%pi*n/15)
8  M=2;
9  x1=x(1:M:N);
10 nl=1:1:N/M;
11 subplot(2,1,1),plot2d3(n,x)
12 xlabel('n'),ylabel('x')
13 title('input-sequence')
14 subplot(2,1,2),plot2d3(nl-1,x1)
15 xlabel('n'),ylabel('x1')
16 title('downsample-sequence')
17
```


Exp. 14

Prel-lab

RA1911004010565

Pushpal Dora

1) What is difference b/w low sampling and decimate?

⇒ Decimation is the process of reducing the sampling rate. In practice, this usually implies low-pass filtering a signal, then throwing away some of its samples. "Sampling" is a more specific term, which refers to just the process of throwing away samples, without the low pass filtering operation.

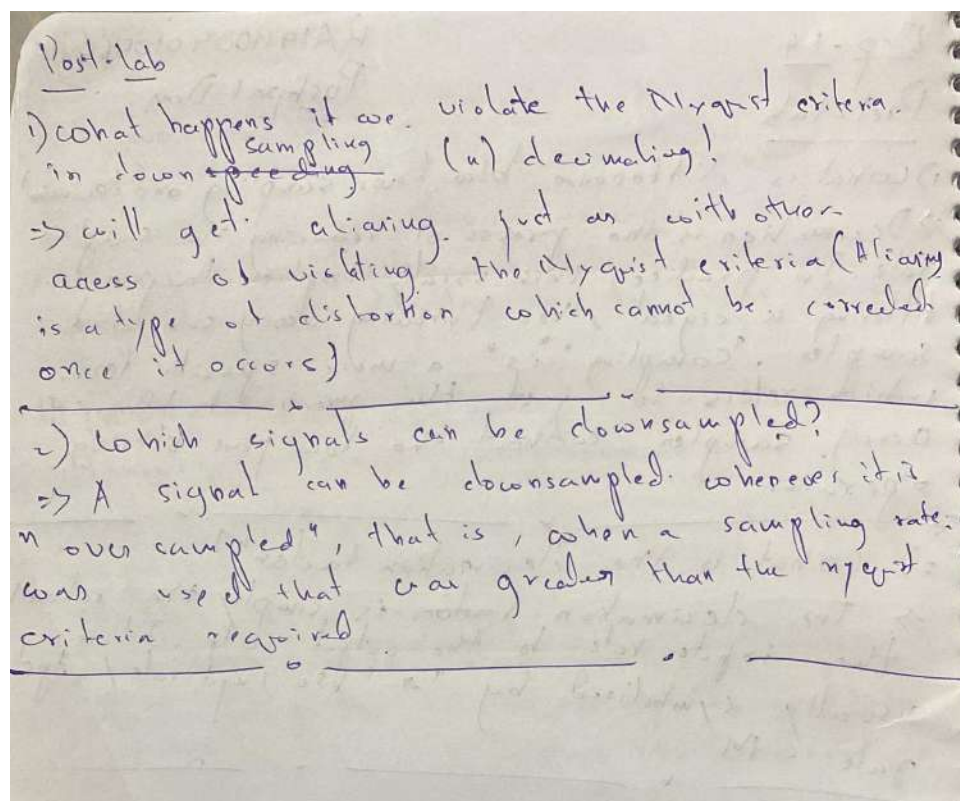
2) What is the "decimation factor"?

⇒ The decimation factor is simply the ratio of the input rate to the output rate. It is usually symbolized by M , so input rate / output rate = M .

Post lab

1. What happens if we violate the Nyquist criteria in downsampling or decimating?
2. Which signals can be downsampled?

POST LAB ANS:-



RESULT:

Thus the code for decimation of signal in SCILAB is verified.

14b. DECIMATION IN THE FREQUENCY DOMAIN

Aim: To write code for decimation of signal in frequency domain using SCILAB

Let $X(f)$ be the Fourier transform of any function, $x(t)$, whose samples at some interval, T , equal the $x[n]$ sequence. Then the discrete-time Fourier transform (DTFT) is a Fourier series representation of a periodic summation of $X(f)$.

$$\underbrace{\sum_{n=-\infty}^{\infty} \overbrace{x(nT)}^{x[n]} e^{-i2\pi f nT}}_{\text{DTFT}} = \frac{1}{T} \sum_{k=-\infty}^{\infty} X\left(f - \frac{k}{T}\right).$$

When T has units of seconds, f has units of hertz. Replacing T with MT in the formulas above gives the DTFT of the decimated sequence, $x[nM]$:

$$\sum_{n=-\infty}^{\infty} x(n \cdot MT) e^{-i2\pi f n(MT)} = \frac{1}{MT} \sum_{k=-\infty}^{\infty} X\left(f - \frac{k}{MT}\right)$$

The periodic summation has been reduced in amplitude and periodicity by a factor of M . An example of both these distributions is depicted in the two traces. Aliasing occurs when adjacent copies of $X(f)$ overlap. The purpose of the anti-aliasing filter is to ensure that the reduced periodicity does not create overlap.

// Program to do decimation in the frequency domain

```
clear;
```

```
clc;
```

```
n=0:%pi/200:2*%pi;
```

```
x=sin(%pi*n);//original signal
```

```
downsampling_x=x(1:2:length(x));//downsampled by a factor of 2
```

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

```
plot(1:length(x),x);
```

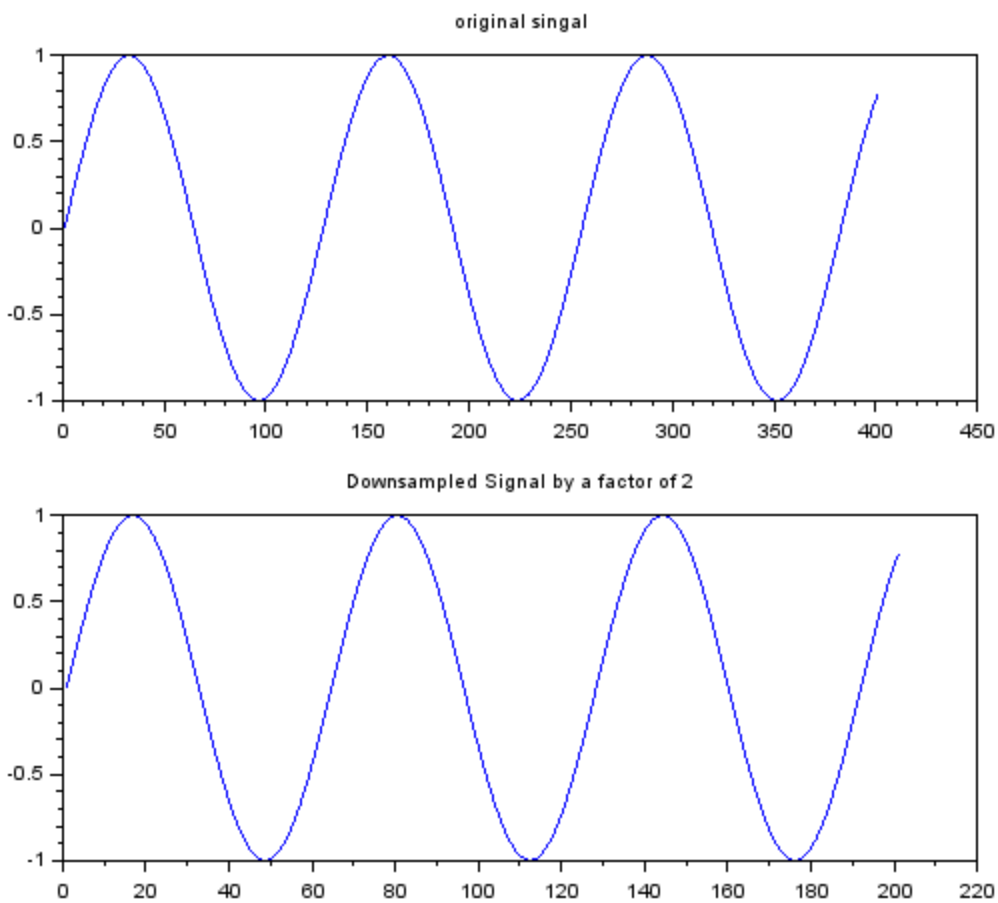
```
xtitle('original singal')
```



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

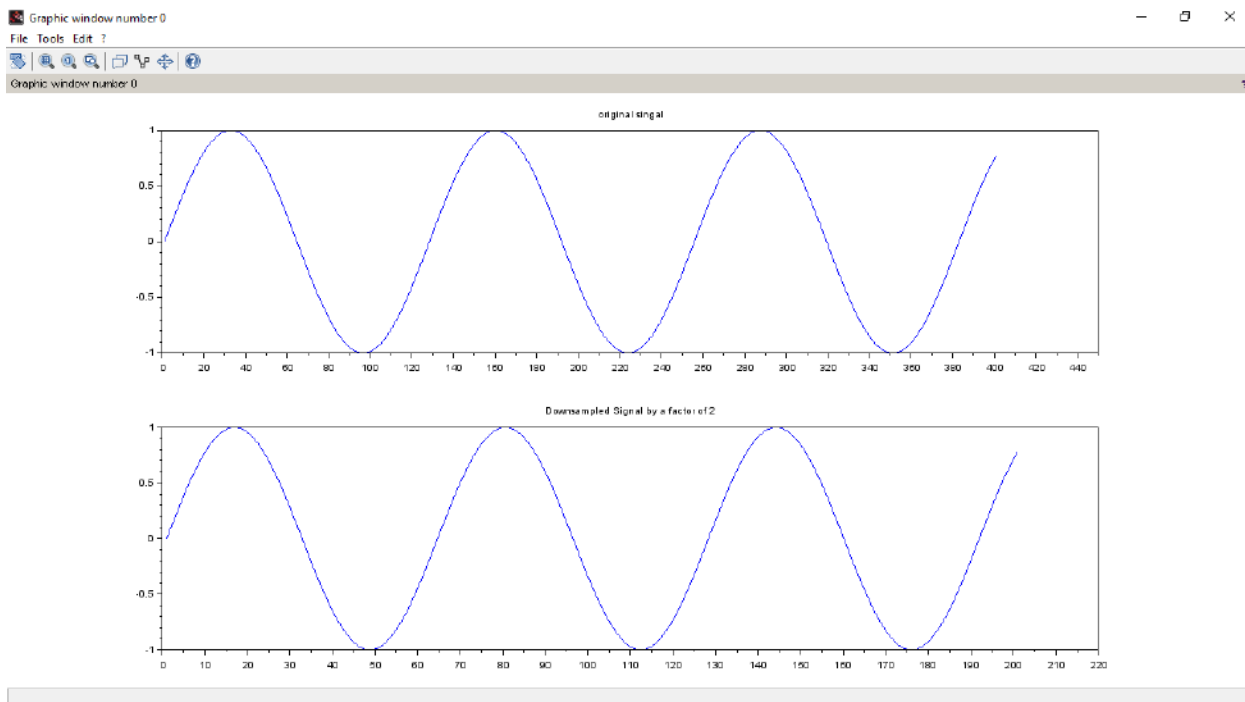
```
plot(1:length( downsampling_x),downsampling_x);
```

```
xtitle('Downsampled Signal by a factor of 2');
```



SCILAB CODE AND SIMULATION:-

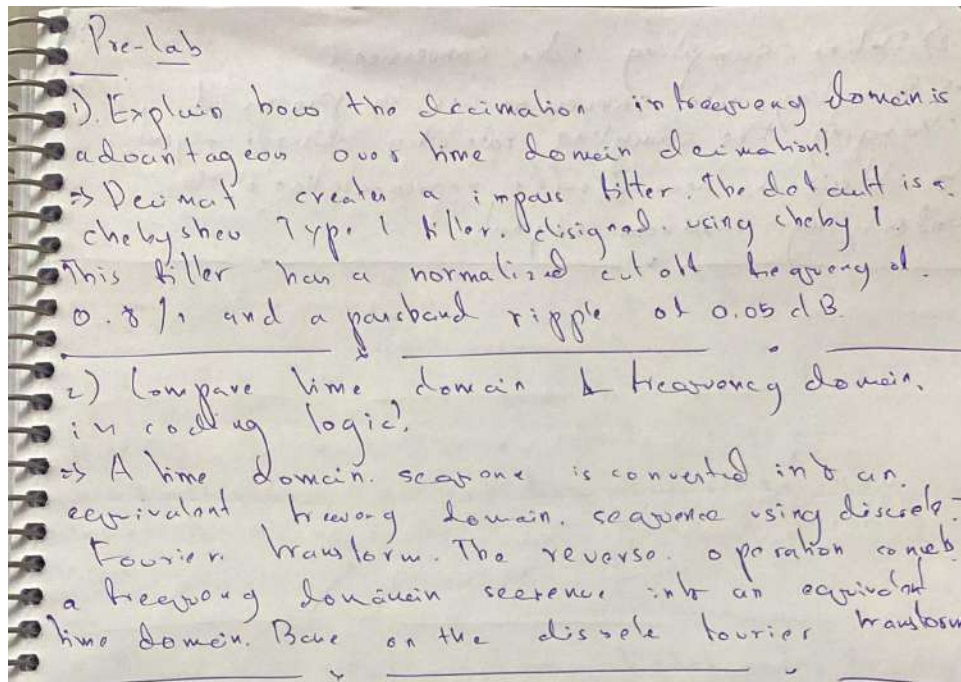
```
DSP EXP 14 dcimation of the signal.sce DSP EXP 14 dcimation in fre domain.sce
1 //Program to do decimation in the frequency domain.
2
3 clear;
4 clc;
5 n=0:%pi/200:2*%pi;
6 x=sin(%pi*n); //original signal
7 downsampling_x=x(1:2:length(x)); //downsampled by a factor of 2
8 subplot(2,1,1)
9 plot(1:length(x),x);
10 xtitle('original singal')
11 subplot(2,1,2)
12 plot(1:length(downsampling_x),downsampling_x);
13 xtitle('Downsampled Signal by a factor of 2');
14
```



PRE LAB:

1. Explain how the decimation in frequency domain is advantageous over time domain decimation?
2. Compare time domain & frequency domain decimation in coding logic?

PRE LAB ANS:-



POST LAB:

1. What is the need for anti-aliasing filter prior to downsampling?
2. Define sampling rate conversion.

POST LAB ANS:-

Postlab

1) What is the use of anti-aliasing filter prior to down sampling.

⇒ An anti-aliasing (or) anti-imaging filter is set before the A/D converter to convert analog frequencies more accurately than a large portion of the sampling rate from being digitized.

2) Define sampling rate conversion.

⇒ Sampling rate conversion is the process of changing the sampling rate of a discrete signal to obtain a new discrete representation of the underlying continuous signal.

RESULT:

Hence code for decimation of signal in frequency domain using SCILAB is verified.

Laboratory Report Cover Sheet

SRM Institute of Science and Technology
College of Engineering and Technology
Department of Electronics and Communication Engineering

18ECC204J DIGITAL SIGNAL PROCESSING

Fifth Semester, 2020-21 (Odd semester)

Name : Pushpal Das

Register No. : RA1911004010565

Day / Session : 1ST / FN

Venue : Online(G Meet)

**Title of Experiment : 15a: Design of anti-aliasing filter
15b) Design of anti-imaging filter**

Date of Conduction : 12 OCT 2021

Date of Submission : 23 OCT 2021

| Particulars | Max. Marks | Marks Obtained |
|------------------------|------------|----------------|
| Prelab and Post lab | 1
0 | |
| Lab Performance | 1
0 | |
| Simulation and results | 1
0 | |
| Total | 30 | |

REPORT VERIFICATION

Staff Name : Mrs.D.VIJAYALAKSHMI

Mrs.A.ANILETBALA

Signature :

Lab 15a: Design of anti-aliasing filter

Aim: To study the characteristics of anti-aliasing filter

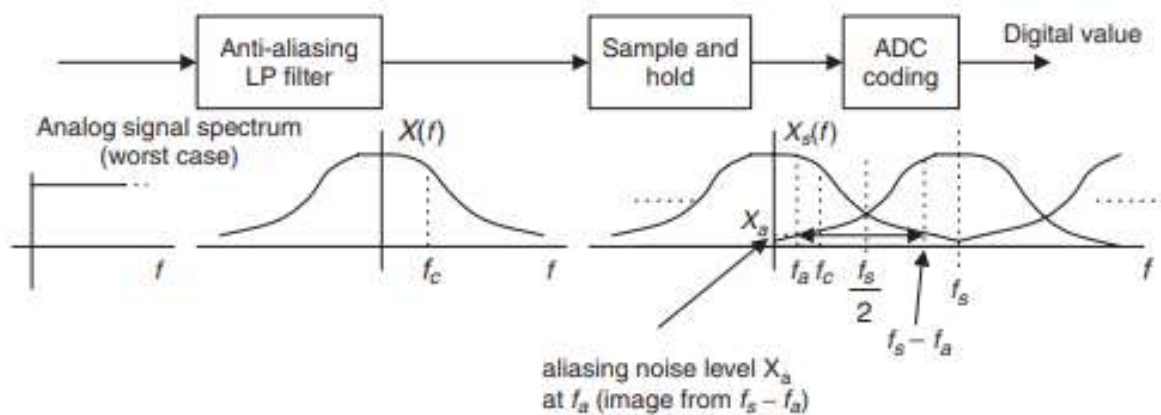
Software Requirement: SCI Lab

Theory: An **anti-aliasing filter (AAF)** is a filter used before a signal sampler to restrict the bandwidth of a signal to satisfy the Nyquist–Shannon sampling theorem over the band of interest. Since the theorem states that unambiguous reconstruction of the signal from its samples is possible when the power of frequencies above the Nyquist frequency is zero, a brick wall filter is an idealized but impractical AAF. A practical AAF trades off between bandwidth and aliasing. A practical anti-aliasing filter will typically permit some aliasing to occur or attenuate or otherwise distort some in-band frequencies close to the Nyquist limit. For this reason, many practical systems sample higher than would be theoretically required by a perfect AAF in order to ensure that all frequencies of interest can be reconstructed, a practice called oversampling.

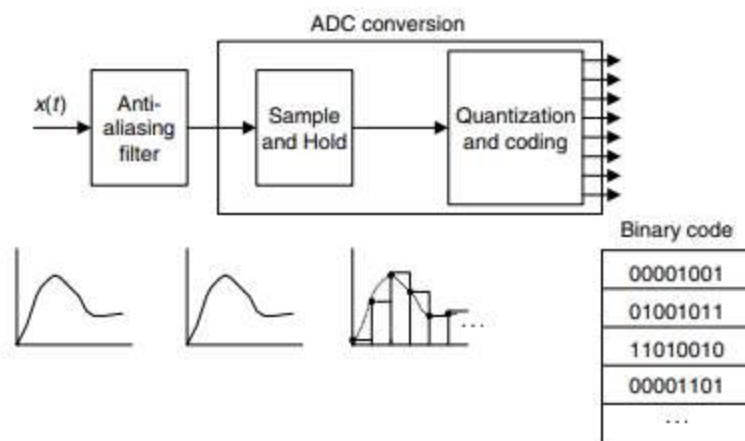
As we have seen, when sampling a signal that contains frequencies that are above half of the sampling frequency, the sampling maps these to some other frequencies within the Nyquist frequency. These so-called *image frequencies* or *alias frequencies* are often unwanted, since they do not represent the original signal. Instead, one uses an anti-aliasing filter (AAF) to filter the input signal before sampling.

The anti-aliasing filter is basically a low-pass filter with (ideal) cutoff frequency of $F_s/2$. Hence, it blocks all frequencies that would create images in the sampled signal, before sampling the signal. Accordingly, there is a loss in the information about high frequencies when applying the anti-aliasing filter. However, the same loss would again be incurred when sampling the signal with frequency $F_s/2$, since the output frequencies cannot be unambiguously mapped to the input frequencies due to the aliasing effect.

In practice, the analog signal to be digitized may contain other frequency components in addition to the folding frequency, such as high-frequency noise. To satisfy the sampling theorem condition, we apply an anti-aliasing filter to limit the input analog signal, so that all the frequency components are less than the folding frequency (half of the sampling rate). Considering the worst case, where the analog signal to be sampled has a flat frequency spectrum, the band-limited spectrum $X(f)$ and sampled spectrum $X_s(f)$ are depicted in Figure given below, where the shape of each replica in the sampled signal spectrum is the same as that of the anti-aliasing filter magnitude frequency response.



Spectrum of the sampled analog signal with a practical anti-aliasing filter.



15a) Design of anti-aliasing filter

Scilab code: Anti-Aliasing Filter

```
clc ;
clf ;
clear all;
b=input('enter no of bits');
n=input('enter band width in KHZ');
As=20*log10(2^b*sqrt(6));
Vs=(10^(0.1*As)-1)^(1/(2*n));
fp=4;
fs=Vs*fp;
S=2*fs;
fa=S-fp;
Va=fa/fp;
Aa =10* log10 (1+ Va ^(2* n ) )
```

Simulation Output:

| No of bits | Frequency in
(KHz) | OUTPUT |
|------------|-----------------------|--------|
|------------|-----------------------|--------|

| | | |
|----|-----|--|
| 4 | 50 | |
| 8 | 100 | |
| 12 | 150 | |

Simulation Output:

- Output for $b = 4$ & $n = 50$ Hz

| Variable Browser | | | | | |
|------------------|------|-------|--------|------------|--------|
| | Name | Value | Type | Visibility | Memory |
| | Aa | 61.5 | Double | local | 216 B |
| | As | 31.9 | Double | local | 216 B |
| | S | 8.61 | Double | local | 216 B |
| | Va | 1.15 | Double | local | 216 B |
| | Vs | 1.08 | Double | local | 216 B |
| | a | 4.61 | Double | local | 216 B |
| | b | 4 | Double | local | 216 B |
| | fa | 4.61 | Double | local | 216 B |
| | fp | 4 | Double | local | 216 B |
| | fs | 4.3 | Double | local | 216 B |
| | n | 50 | Double | local | 216 B |

- Output for $b = 8$ & $n = 100$ Hz

| Variable Browser | | | | | |
|------------------|------|-------|--------|------------|--------|
| | Name | Value | Type | Visibility | Memory |
| | Aa | 109 | Double | local | 216 B |
| | As | 55.9 | Double | local | 216 B |
| | S | 8.53 | Double | local | 216 B |
| | Va | 1.13 | Double | local | 216 B |
| | Vs | 1.07 | Double | local | 216 B |
| | a | 4.53 | Double | local | 216 B |
| | b | 8 | Double | local | 216 B |
| | fa | 4.53 | Double | local | 216 B |
| | fp | 4 | Double | local | 216 B |
| | fs | 4.27 | Double | local | 216 B |
| | n | 100 | Double | local | 216 B |

- Output for $b = 12$ & $n = 150$ Hz

| | Name | Value | Type | Visibility | Memory |
|--|------|-------|--------|------------|--------|
| | Aa | 155 | Double | local | 216 B |
| | As | 80 | Double | local | 216 B |
| | S | 8.51 | Double | local | 216 B |
| | Va | 1.13 | Double | local | 216 B |
| | Vs | 1.06 | Double | local | 216 B |
| | a | 4.51 | Double | local | 216 B |
| | b | 12 | Double | local | 216 B |
| | fa | 4.51 | Double | local | 216 B |
| | fp | 4 | Double | local | 216 B |
| | fs | 4.25 | Double | local | 216 B |
| | n | 150 | Double | local | 216 B |

SCILAB CODE & SIMULATION:-

```

exp 15 a.sce (C:\Users\ELCOT\Downloads\exp 15 a.sce) - SciNotes
File Edit Format Options Window Execute ?
exp 15 a.sce (C:\Users\ELCOT\Downloads\exp 15 a.sce) - SciNotes
exp 15 a.sce DSP exp 15.sce
1 // Anti-Aliasing Filter
2 clc ;
3 clf ;
4 clear all;
5 b=input('enter no. of bits');
6 n=input('enter band width in KHz');
7 As=20*log10(2*b*sqrt(6));
8 Va=(10^(0.1*As)-1)/(1/(2*n));
9 fp=1;
10 fs=Va*fp;
11 S=1/fs;
12 fa=S*fp;
13 Va=fa*fp;
14 Aa =10*log10 (1+ Va ^ (2*n) ) ;
15

```

Variable Browser

File Filter ?

File Browser

C:\Users\ELCOT\Documents\

Documents

Camera

Custom Office Templates

LTSpiceXVII

My Music

My Pictures

My Videos

Python Scripts

R

RData

RHistory

18ECP 10 IL-MASSIVE OPEN.pptx

2nd projects.docx

3 bit counter count 0 2 4 6 0.circ

8ae2cf06-5777-46f3-93df-ca53541d1538-4195cadd-38fd--

ANA_12(cascode current mirror).asc

ANA_12(cascode current mirror).log

ANA_12(cascode current mirror).raw

ANA_12(Common source amplifier).asc

ANA_12(Common source amplifier).log

ANA_12(Common source amplifier).op.raw

ANA_12(Common source amplifier).raw

ANA_12(common current mirror).asc

enter no of bits4

enter band width in KHz50

-->

Variable Browser

| Name | Value | Type | Visibility | Memory |
|------|-------|--------|------------|--------|
| As | 61.5 | Double | local | 216 B |
| As | 31.9 | Double | local | 216 B |
| S | 8.61 | Double | local | 216 B |
| Va | 1.15 | Double | local | 216 B |
| Vs | 1.08 | Double | local | 216 B |
| b | 4 | Double | local | 216 B |
| fb | 4.61 | Double | local | 216 B |
| fb | 4 | Double | local | 216 B |
| fs | 4.3 | Double | local | 216 B |

News feed

News feed unavailable.

Scilab 6.1.0 Console

File Edit Control Applications ?

File Browser

C:\Users\ELCOT\Documents\

Documents

Camera

Custom Office Templates

LTSpiceXVII

My Music

My Pictures

My Videos

Python Scripts

R

RData

RHistory

18ECP 10 IL-MASSIVE OPEN.pptx

2nd projects.docx

3 bit counter count 0 2 4 6 0.circ

8ae2cf06-5777-46f3-93df-ca53541d1538-4195cadd-38fd--

enter no of bits8

enter band width in KHz100

-->

Variable Browser

| Name | Value | Type | Visibility | Memory |
|------|-----------------|---------|------------|--------|
| As | 109 | Double | local | 216 B |
| Ap | 0.316 | Double | local | 216 B |
| As | 55.9 | Double | local | 216 B |
| S | 8.53 | Double | local | 216 B |
| S1 | 1.76 | Double | local | 216 B |
| Va | 1.13 | Double | local | 216 B |
| Vs | 1.07 | Double | local | 216 B |
| e | 1x1 Graphic ... | Graphic | local | 216 B |
| b | 8 | Double | local | 216 B |

News feed

News feed unavailable.

Scilab 6.1.0 Console

File Edit Control Applications ?

File Browser

C:\Users\ELCOT\Documents\

Documents

Camera

Custom Office Templates

LTSpiceXVII

My Music

My Pictures

My Videos

Python Scripts

R

RData

RHistory

18ECP 10 IL-MASSIVE OPEN.pptx

2nd projects.docx

3 bit counter count 0 2 4 6 0.circ

8ae2cf06-5777-46f3-93df-ca53541d1538-4195cadd-38fd--

ANA_12(cascode current mirror).asc

enter no of bits12

enter band width in KHz150

-->

Variable Browser

| Name | Value | Type | Visibility | Memory |
|------|-----------------|---------|------------|--------|
| As | 155 | Double | local | 216 B |
| Ap | 0.316 | Double | local | 216 B |
| As | 89 | Double | local | 216 B |
| S | 8.51 | Double | local | 216 B |
| S1 | 1.76 | Double | local | 216 B |
| Va | 1.13 | Double | local | 216 B |
| Vs | 1.06 | Double | local | 216 B |
| e | 1x1 Graphic ... | Graphic | local | 216 B |
| b | 12 | Double | local | 216 B |

News feed

News feed unavailable.

15b) Design of anti-imaging filter

Theory : In a mixed-signal system (analog and digital), a **reconstruction filter**, sometimes called an **anti-imaging filter**, is used to construct a smooth analog signal from a digital input, as in the case of a digital to analog converter (DAC) or other sampled data output device.

While in theory a DAC outputs a series of discrete Dirac impulses, in practice, a real DAC outputs pulses with finite bandwidth and width. Both idealized Dirac pulses, zero-order held steps and other output pulses, if unfiltered, would contain spurious high-frequency replicas, "*or images*" of the original bandlimited signal. Thus, the reconstruction filter smooths the waveform to remove image frequencies (copies) above the Nyquist limit. In doing so, it reconstructs the continuous time signal (whether originally sampled, or modelled by digital logic) corresponding to the digital time sequence.

Practical filters have non-flat frequency or phase response in the pass band and incomplete suppression of the signal elsewhere. The ideal [sinc](#) waveform has an infinite response to a signal, in both the positive and negative time directions, which is impossible to perform in real time – as it would require infinite delay. Consequently, real reconstruction filters typically either allow some energy above the Nyquist rate, attenuate some in-band frequencies, or both. For this reason, oversampling may be used to ensure that frequencies of interest are accurately reproduced without excess energy being emitted out of band.

In systems that have both, the anti-aliasing filter and a reconstruction filter may be of identical design. For example, both the input and the output for audio equipment may be sampled at 44.1 kHz. In this case, both audio filters block as much as possible above 22 kHz and pass as much as possible below 20 kHz.

Alternatively, a system may have no reconstruction filter and simply tolerate some energy being wasted reproducing higher frequency images of the primary signal spectrum.

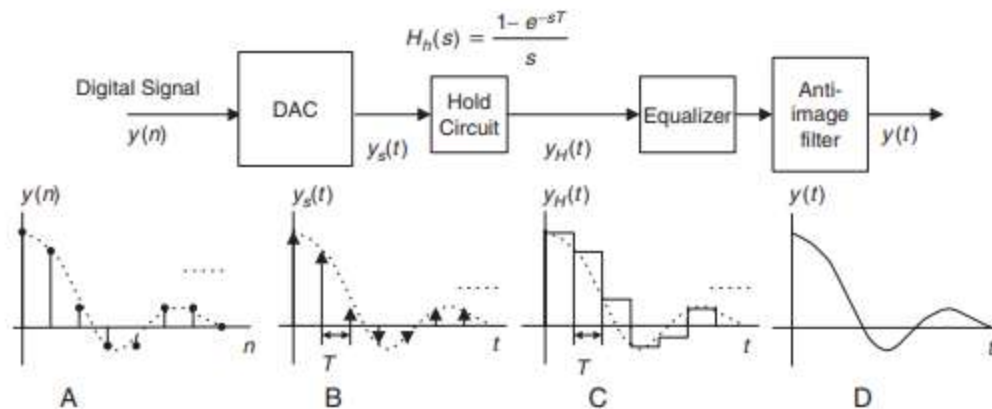


FIGURE 2.19 Signal notations at the practical reconstruction stage. (a) Processed digital signal. (b) Recovered ideal sampled signal. (c) Recovered sample-and-hold voltage. (d) Recovered analog signal.

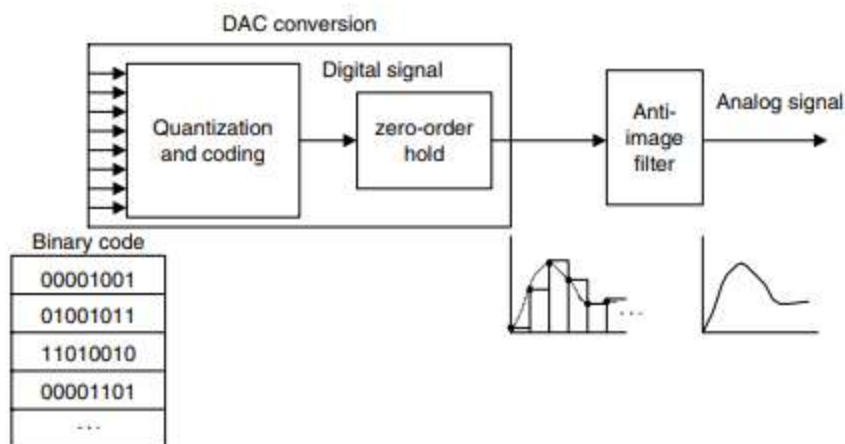


FIGURE 2.32 Typical DAC process.

Scilab code: anti-imaging filter

Program:

```
clc ;
clf ;
clear all;
// Anti Imaging Filter considerations
Ap =0.5; // pass band attenuation
```

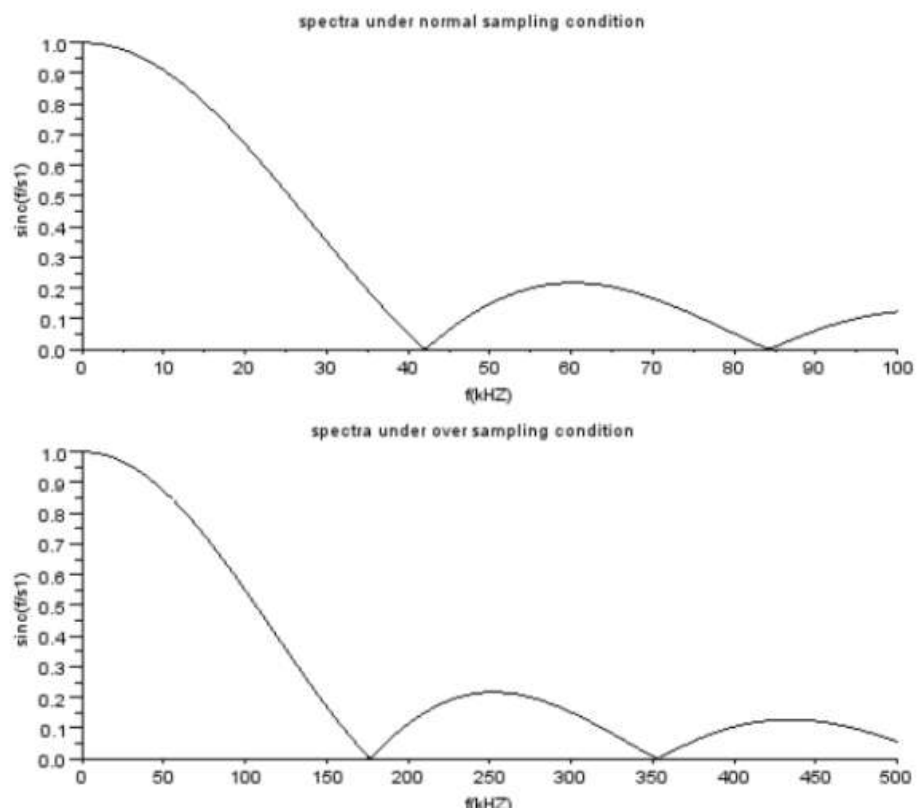
```

fp =20; // pass band edge frequency
As =60; // stop band attenuation
S =42.1;
fs =S - fp ; // stop band edge frequency
e = sqrt (10^(0.1* Ap ) -1) ;
e1 = sqrt (10^(0.1* As ) -1) ;
n =( log10 ( e1 / e ) ) /( log10 ( fs / fp ) ) ;
n = ceil ( n ) // design of nth order but worth filter
// ( b ) Assuming Zero–order hold sampling
S1 =176.4;
fs1 = S1 - fp;
Ap =0.316;
e2 = sqrt (10^(0.1* Ap ) -1);
n1 =( log10 ( e1 / e2 ) ) /( log ( fs1 / fp ) ) ; //new o r d e r of but worth filter
n1 = ceil ( n1 )
f=0:100;
x = abs( sinc ( f * %pi / S ) ) ;
f1 =0:500;
x1 =abs( sinc ( f1 * %pi / S1 ) ) ;
a = gca () ;
subplot (211) ;
plot2d (f , x ) ;
xlabel (” spectra under normal sampling condition ” ,” f(kHZ ) ” ,” s i n c ( f / s1 ) ”);
subplot (212);
plot2d ( f1,x1 );
xlabel (” spectra under over sampling condition ” ,” f(kHZ ) ” ,” s i n c ( f / s1 ) ”);

```

Simulation Output:

Anti Imaging Filter Consideration

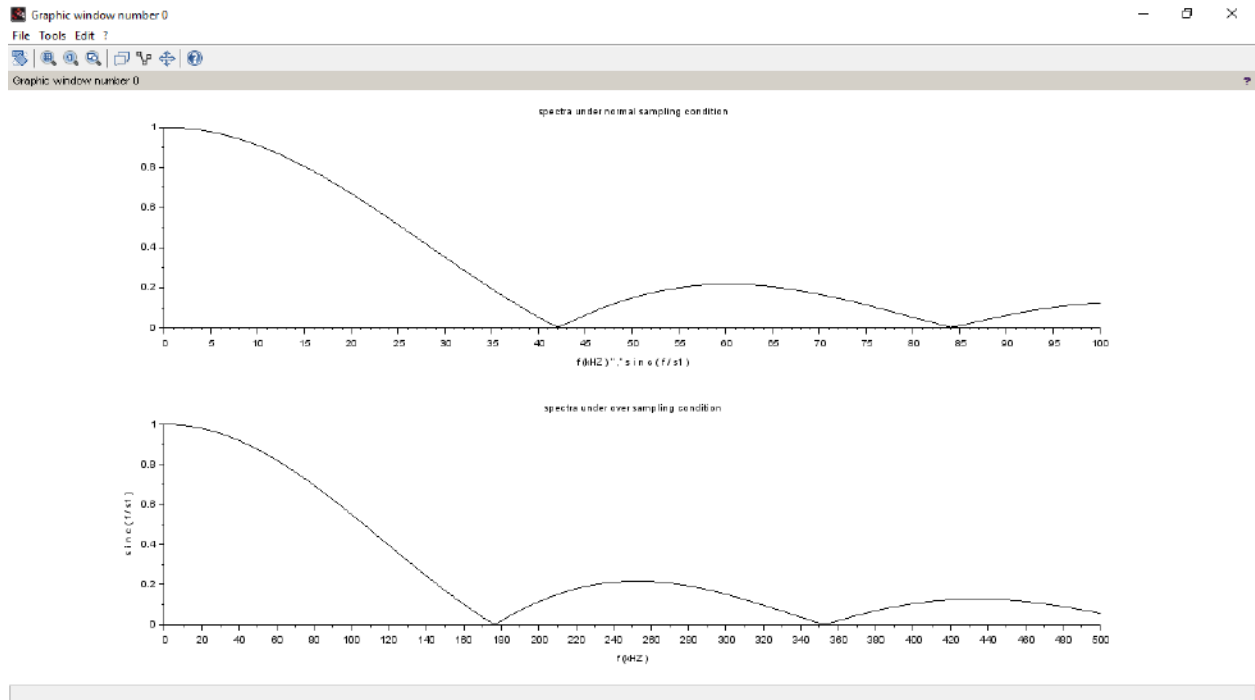


SCILAB AND SIMULATION:-


```

DSP exp 15B (1).sce (C:\Users\ELCOT\Downloads\DSP exp 15B (1).sce) - SciNotes
File Edit Format Options Window Execute ?
DSP exp 15B (1).sce (C:\Users\ELCOT\Downloads\DSP exp 15B (1).sce) - SciNotes
exp 15a.sce DSP exp 15B (1).sce
1 //anti-imaging-filter:
2 //Program:
3 clc;
4 clf;
5 clear all;
6 // Anti Imaging Filter considerations
7 Ap=0.5; // pass band attenuation
8 fp=20; // pass band edge frequency
9 As=60; // stop band attenuation
10 S=12.1;
11 fs=S-fp; // stop band edge frequency
12 e=sqrt(10^(0.1*Ap)-1);
13 e1=sqrt(10^(0.1*As)-1);
14 n=(log10(e1/e))/(log10(fs/fp));
15 n=ceil(n); //design of nth order but worth filter
16 //(-b)-Assuming zero-order hold sampling
17 S1=176.4;
18 fs1=S1-fp;
19 Ap=0.016;
20 e2=sqrt(10^(0.1*Ap)-1);
21 n1=(log10(e1/e2))/(log10(fs1/fp)); //new order of but worth filter
22 n1=ceil(n1);
23 f=0.100;
24 x=sbs(sinc(f*S1/2));
25 fl=0.500;
26 x1=sbs(sinc(fl*S1/2));
27 a=gca();
28 subplot(2,1);
29 plot2d(f,x);
30 xtitle('spectra under normal sampling condition','f (kHz)','sinc(f*S1/2)');
31 subplot(2,1);
32 plot2d(fl,x1);
33 xtitle('spectra under over sampling condition','f (kHz)','sinc(fl*S1/2)');
34
Line 3, Column 5.

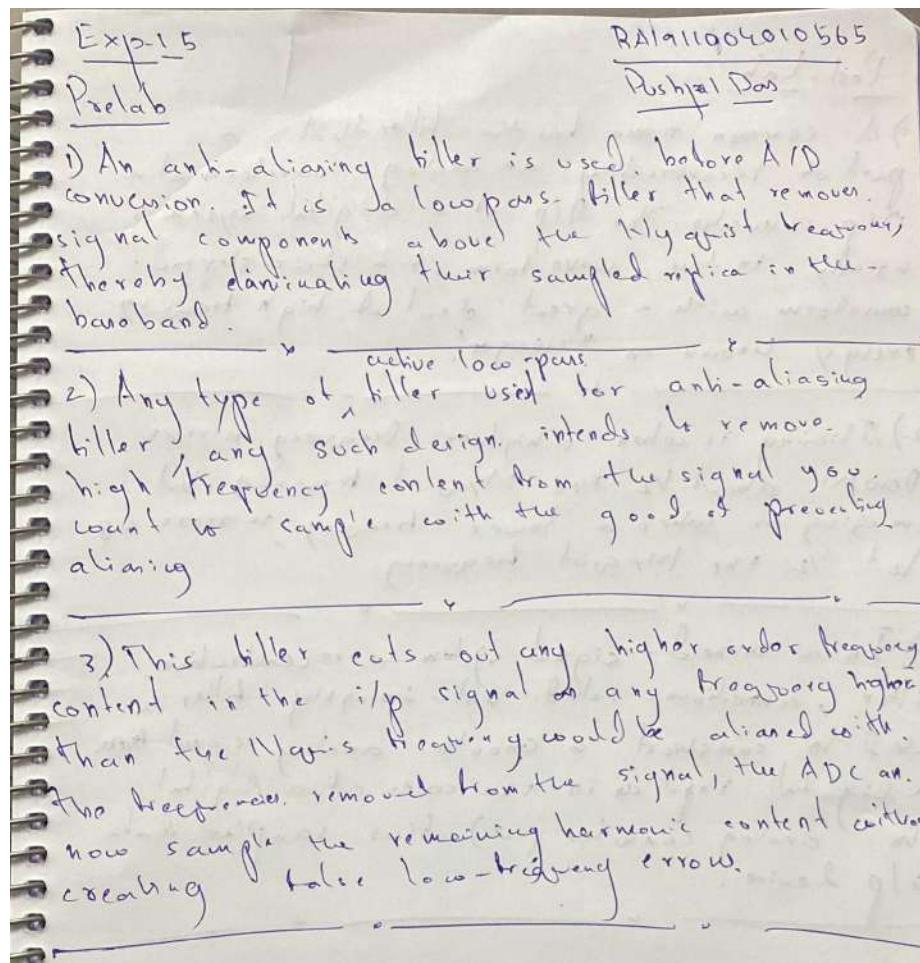
```



Pre-lab questions:

1. What is meant by anti-aliasing filter?
2. What is the filter used for anti-aliasing?
3. Mention the advantages of anti-aliasing filter.

PRE LAB ANSWERS:-



Post-Lab questions:

1. What is meant by anti-Imaging filter?
2. Mention the difference between anti-aliasing and anti-imaging filter.
3. Mention the advantages of the anti-imaging filter.

POST LAB ANSWERS:-

Post-Lab

1) A common name for the filter that is a part of reconstructing an analog waveform at a D/p converter. The D/p of a digital system is usually in the wave form of a stair-stepped waveform with a great deal of high frequency energy known as "images".

2) Aliasing is when a higher frequency mirrors down about $\frac{1}{2}$ the Nyquist frequency, but imaging is when a lower frequency mirrors up about $\frac{1}{2}$ the Nyquist frequency.

3) In a mixed signal system a reconstruction filter, sometimes called anti-imaging filter is used to construct a smooth analog signal from a digital input as in the case of a digital to analog converter (or) other sampled data o/p device.

Result:

Thus we designed an anti imaging filter by using Scilab.



SRM Institute of Science and Technology

College of Engineering and Technology

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

MINI PROJECT REPORT

ODD Semester, 2021-22

Lab code & Sub Name : 18ECC204J – DIGITAL SIGNAL PROCESSING

Year & Semester : 3rd & 5th

**Mini Project Title : Application of Sound Effect on a
audio file**

Lab Supervisor : Mrs.D.VIJAYALAKSHMI

Team Members :Pushpal Das (RA1911004010565)

| | | | |
|---|--|------------------------|--|
| Reg. No | | RA1911004010565 | |
| Mark split up | | Pushpal Das | |
| Novelty in the Mini project work
(2 marks) | | | |
| Level of understanding
(4 marks) | | | |
| Contribution to the project
(2 Marks) | | | |
| Report writing
(2 Marks) | | | |
| Total (10 Marks) | | | |

Date:

Signature of Lab In charge

Application of Sound Effect on a Audio file

OBJECTIVE :

To add Sound effect on audio file using Scilab

INTRODUCTION :

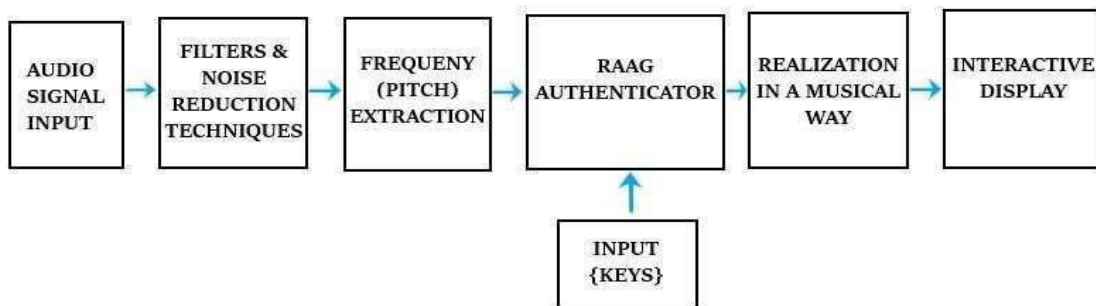
Musical tones have three identifying characteristics; volume, pitch and timbre. Volume is power, or the amplitude of the corresponding wave, and it is measured in decibels. Frequency is the measure of how "high" or "low" a tone is, which is measured in hertz (Hz). The third identifying feature, timbre, stems from the fact that musical sounds are made up of many different sine waves (as opposed to a sound that is made of just one sine wave). Each instrument has a characteristic pattern of sine waves, which is what makes it possible to distinguish between an oboe and an electric guitar playing the same note.

ABSTRACT :

The basic idea of project is that it exploits the fact that each Raag is associated with certain frequency and this frequency is processed further, to achieve the objective. It also notifies us if he goes wrong according to the laws of Indian Classical Music in his practice sessions. The musical notes played by us are nothing but the frequencies or the pitch to be extracted and processed. As the project involves intensive audio processing, DSP (Digital Signal Processing) Kit is used as hardware component, while for software assistance SCILAB is used.

BLOCK DIAGRAM :

The step by step working and analysis of basic musical tone generator. The major blocks would include Audio signal, Frequency extraction, Raag authentication and at the end the displaying content in the form of musical note.



- The input sound is taken through the 'Audio Signal Input', which is nothing but the microphone.
- This output is then given to the 'Filter and Noise Reduction Block', which consists of 'Low Pass Filter', in order to remove noise and other disturbances.
- The 'Frequency (Pitch) extraction' block is used to take out the frequencies from the clean audio signal (sound).
- The 'RAAG Authenticator' would consist of laws and regulations of Indian Classical Music. A database of music would be stored in this block, and would compare with the music and thereby indicating whether it is correct or not.
- The 'Input {Keys}' block would be interfacing of Keypad. It would have options to select the base frequency and the type of 'RAAG' which we will select after recording and then compare.
- The block 'Realization in a musical way' is used, as we have to display the Sur/Notes (Sa, Re, Ga, etc) rather than the musical frequency. This objective would be achieved by coding. For instance, if the tune played is at 440Hz, rather than displaying 440Hz, would display 'Sa' on the device.
- These Sur/Notes will be then displayed on the PC Terminal and it will also indicate whether the tune played is correct or not.

WORKING PRINCIPLE :

Sampling frequency is the reduction of continuous time signal to discrete time. Sampling can be done for function varying in space, time or any other dimension.

The program realization in a musical way is used, as we have to display the Sur/Notes rather than the musical frequency. This objective would be achieved by coding. For instance, if the tune played is at 440Hz, rather than displaying 440Hz, we would display 'Sa' on our device. These Sur/Notes will be then displayed on the PC Terminal and it will also indicate whether the tune played is correct or not.

CODE :

```
1  clc;
2  clear;
3  clear all;
4  [y,fs] = wavread("C:\Users\Admin\Downloads\dsp01.wav");
5  playsnd(y,fs);
6  halt('Original');
7  //Flanging
8  z = 0;
9  n = 1:length(y);
10 z(n) = y(n);
11 m = 11:length(y);
12 x(m) = z(m) + 0.8*z(m-10).cos(2*pi*m/fs);
13 playsnd(x,fs);
14 halt('Flange');
15
16 //Echo
17 a = 0.5;
18 [l,fs] = wavread("C:\Users\Admin\Downloads\dsp01.wav");
19 len = length(l);
20 delay = 0.4;
21 D = ceil(fs*delay);
22 m = zeros(max(size(l)));
23 for i = D+1:len
24     m(i) = l(i) + a*l(i-D);
25 end
26 playsnd(m,fs);
27 halt('Echo');
28 //Equalizer
29 //For Low frequencies
30 //clc;
31 //clear all;
32 [h,fs] = wavread("C:\Users\Admin\Downloads\dsp01.wav");
33 playsnd(h,fs);
34 j = iir(3,'bp','butt',[.01-.05],[0.03 0.03]);
35 k = flts(h(1,:),j);
36 playsnd(k,fs);
37 halt('Low');
38 //For High frequencies
39 //clc;
40 //clear all;
41 [h,fs] = wavread("C:\Users\Admin\Downloads\dsp01.wav");
42 playsnd(h,fs);
43 halt('High');
44 j = iir(3,'bp','butt',[.15 .25],[0.03 0.03]);
45 k = flts(h(1,:),j);
46 playsnd(k,fs);
47 halt('Very-low');
```


CONCLUSION :

The methodology used in this design is helpful and successful. Creating the algorithm in a SCILAB .m file was instrumental in verifying the outputs over the hardware. The Note and Pitch Detection (Frequency Extraction), Raag Identification along with error correction was successfully implemented in SCILAB.

REFERENCE :

Drive link - [DSP - Google Drive](#)