Course outcomes for Digital Signal Processing Lab:

Course outcomes: Upon successful completion of the course, students will be able to:

CO1: Determine the response of a system for an arbitrary input **(L3)**

CO2: Perform spectral analysis of a signal and determine the frequency response of a system**(L4)**

CO3: Design a digital FIR filter for a given specification and use it in implementing special systems like Hilbert transformer, differentiator**(L4)**

CO4: Design a digital IIR filter for a given specification and use it to suppress unwanted signals**(L4)**

CO5: Implement basic signal processing algorithms for applications in communication, signal processing**(L4)**

(*Knowledge levels :* *L1:Remembering, L2: Understanding, L3: Applying, L4:Analyzing, L5:Evaluating, L6:Creating***)**

Course Articulation matrix

PO ⇨ CO ⇩	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**
CO1	**3**	**1**			**3**			**1**	**2**	**1**		**2**	**2**	**2**
CO2	**3**	**2**		**2**	**3**			**1**	**2**	**1**		**2**	**2**	**2**
CO3	**3**	**2**	**3**	**2**	**3**			**1**	**2**	**1**		**2**	**3**	**2**
CO4	**3**	**2**	**3**	**2**	**3**			**1**	**2**	**1**		**2**	**3**	**2**
CO5	**3**	**2**	**3**	**2**	**3**			**1**	**2**	**1**		**2**	**3**	**2**
CO	**3**	**2**	**3**	**2**	**3**			**1**	**2**	**1**		**2**	**3**	**2**

Attainment Levels: 1-Slight(Low), 2:Moderate(Medium), 3. Substantial(high)
(***PO:GA :*** *a:Engineering Knowledge, b:Problem analysis, c: design/development of solutions, d:conduct investigations of complex problems, e: modern tool usage, f:engineer and society, g:environment and sustainability, h:ethics, i:individual and team work, j:communication, k:project management and finance, l:life long learning)*

CO1: Impulse response, Difference equation, Linear convolution

CO2: DTFT, DFT , Sampling theorem

CO3: FIR filter design, Hilbert transformer, Differentiator

CO4: IIR filter design, application

CO5: DSP processor Implementation, Open ended experiments

Contents

PART - A: Experiments using SciLab

1. Generation of signals
2. DTFT of discrete-time signals
3. Discrete Fourier transform
4. Linear filtering in time and frequency domains
5. Spectral analysis using DFT
6. Design of IIR Butterworth low pass filters
7. Design of IIR Chebyshev low pass filters
8. Design of low pass FIR filters using windowing
9. Application of FIR filters: Design of Differentiator and Hilbert transformer.

PART - B: Signal processing experiments using TMS3206713 Simulator

1. Linear convolution
2. Circular convolution
3. DFT computation
4. Design of IIR Butterworth low pass filters and verification using synthesized signals.

PART – C : Open Ended Experiments

PART – A

USING SCILAB

Introduction to SCILAB

Scilab is a free and open-source, cross-platform numerical computational package and a high-level, numerically oriented programming language. It can be used for signal processing, statistical analysis, image enhancement, fluid dynamics simulations, numerical optimization and modeling, simulation of explicit and implicit dynamical systems and symbolic manipulations.

1. Generation of signals

Important functions : plot(), plot2d(), plot2d3(),ones(), zeros(), figure(), xtitle(), subplot(), xlabel(), ylabel(),........

Scilab 5.5.2 , OS: Ubuntu 14.04

i) Unit Sample Sequence

```
clear ;clc ;close ;
L = 4;                    // U p p e r l i m i t
n = -L:L;                 // index of the sequence
x = [zeros(1,L),1,zeros(1,L)];
figure (1);
plot2d3(n,x);             //For discrete time signal
xtitle(' Unit Sample Sequence','n','x1[n]') ;
```

ii) Unit step function

```
clear ;clc ;close ;
n=0:5
x=[ones(1,6)];
figure(1); plot2d3(n,x);
xtitle('Discrete Unit Step Sequence','n','x[n]') ;
```

iii) Unit ramp function

```
clear ;clc ;close ;
L = 4;
n = -L : L;
x = [zeros(1,L ),0:L ];
plot2d3(n,x);
xtitle(' Discrete Unit Ramp Sequence','n','x[n]') ;
```

iv) Discrete time Exponential signal

```
clear ;clc ;close ;
a =0.5;   //For decreasing a<1 and For increasing exponential a>1
n = 0:10;
x = (a).^n ;
plot2d3(n,x);xtitle('Exponentially Decreasing Signal ','n','x[n]');
```

v) Complex valued signals

```
clc;clear;
n= [-10:1:10];
a= -0.1+0.3*%i;
x=exp(a*n);
subplot(221), plot2d3(n,real(x));xtitle('Real part','n');
subplot(223), plot2d3(n,imag(x));xtitle('Imaginary','n');
subplot(222), plot2d3(n,abs(x));xtitle('Magnitude part','n');
theta=(180/%pi)*atan(imag(x),real(x));
subplot(224), plot2d3(n,theta);xtitle('Phase part','n');
```

vi) Sinusoidal signal

```
clc;clear;
fm=input('Enter the input signal frequency:');
k=input('Enter the number of Cycles of input signal:');
A=input('Enter the amplitude of input signal:');
tm=0:1/(fm*fm):k/fm;
x=A*cos(2*%pi*fm*tm);
figure(1);plot2d3(tm,x);
title('Graphical Representation of Sinusoidal Signal');
```

```
xlabel('Time');ylabel('Amplitude');
xgrid(1)
```

vii) Square wave

```
clc;clear;
t=(0:0.1:4*%pi)';
plot2d3(4*%pi*squarewave(t));
xtitle('square wave','time','amplitude');
```

viii) Triangular wave

```
clear;clc;
A=input('enter the amplitude:');
K=input('enter number of cycles:');
x1 = [0:A  A-1:-1:1];
x=x1;
for i=1:K-1
x=[x x1];
end
n=0:length(x)-1;     // Index of the sequence
plot2d3(n,x);xtitle('Triangular wave','time','amplitude');
```

ix) Sawtooth wave

```
clc;clear;
A=input('enter the amplitude:');
K=input('enter number of cycles:');
x1 = [0:A];
x=x1;
for i=1:K-1
   x=[x x1];
end
n=0:length(x)-1;
plot2d3(n,x);xtitle('Sawtooth wave','time','amplitude');
```

x) Writing your own function

Syntax: function [output variables] = function_name(input variables);

```
clear all;clc
function [Ex] = energy(x)
Ex= sum(x.*conj(x));
end function
x=[1 2 3 4 5];
n=0:length(x);
E=energy(x)
```

Problems:

1. Decompose a real sequence x[n] into even and odd components.

2. DTFT of discrete time signals

a) To compute the discrete-time Fourier transform of a sequence h[n] and plot its magnitude and phase spectrum.

Design steps:

a) 1. Use $X(e^{j\omega}) = \sum x[n]\, e^{-j\omega n}$ or the matrix notation X =Wx where $W = [e^{(-j2\pi k\, n/M)}]$, k=0,1,...M -1 and n=0,1,..N-1 and $x= [x[0]\ x[1]...]^T$ to compute DTFT

2. Plot the magnitude and phase response
3. Ensure that the frequency axis is from [0, π] and verify the periodicity and conjugate symmetry property.

Design example: Plot the frequency response of the sequence h[n] = (0.9 **)^n, $0 \le n \le 9$.**

Using matrix notation $X^T= x^T\,[\, e^{(-j2\pi\, nTk/M)]}$

```
clear;clc;close;
h=input('enter the input sequence');       //h=[1 1 1 1 1]
n=0:length(h)-1;
w=-%pi:%pi/4:%pi;                          //To evaluate X(ejw) at equi-spaced frequencies
wn= n'*w;
minus_jwn= -%i*n'*w
H=h*exp(minus_jwn);
magH=abs(H);
angH=atan(imag(H),real(H));
figure();
subplot(311),plot2d3(n,h);
xtitle('Impulse Response','time index n');
subplot(312),plot2d(w,magH);
xtitle('Magnitude Response', 'frequency in rad', 'Magnitude');
subplot(313),plot2d(w,angH);
xtitle('Phase Response', 'frequency in rad', 'Phase');

//Expected Result
//H= [1, 0.4142136i,1,2.4142136i,5,- 2.4142136i,1,- 0.4142136i,1]
```

Problems:

1. $x[n] = \begin{cases} 2 - (\frac{1}{2})^n & ; |n| \leq 4 \\ 0 & ; otherwise \end{cases}$

2. x[n] = $(0.5)^n$ {u(n+3)-u(n-2)}

3. x[n]= $(0.25)^n$ {u(n+4)-u(n-4)}

2. b) To determine the frequency response of a discrete-time system from its difference equation

Design steps: Given $a_0\, y[n] = -a_2\, y[n-2] - a_1\, y[n-1] + b_0\, x[n] + b_1\, x[n-1] + b_2\, x[n-2]$

1. System function $H(z) = b_0 + b_1\, z^{-1} + b_2\, z^{-2} / 1 + a_1\, z^{-1} + a_2\, z^{-2}$
2. Put $z= e^{(jw)}$ to get the frequency response

Design example: Plot the magnitude and phase response of the system represented by
$6y[n]+5y[n-1]+y[n-2]=18x[n] + 8x[n-1]$

```
clear;clc;close;
b=[18, 8];
a=[6 5 1];
m= 0: length(b)-1; p=0:length(a)-1;
w=-2*%pi:%pi/100:2*%pi;
num = b* exp(-%i*m'*w);
den = a*exp(-%i*p'*w);
H= num./den;
magH = abs(H); angH= atan(imag(H),real(H));
figure;
subplot(211), plot( w, magH);
xtitle('Magnitude response','Frequency in rad','Magnitude');
subplot(212),plot(w, angH);
xtitle('Phase Response','Frequency in rad','Phase');
```

Problems:

1. $y[n]= 0.5\, y[n-1] – 0.06\, y[n-2] + 0.2\, x[n] -0.5\, x[n-1]$
2. $y[n] = x[n] – x[n-1] -0.5\, y[n-1]$
3. $y[n] = x[n] -x[n-1] + x[n-2] +0.95\, y[n-1] -0.9025\, y[n-2]$
4. $y[n] = x[n-1] + (1/9)\, y[n-2]$ with $y[-1] = 1$, $y[-2]=0$ & $x[n]=u[n]$

3. Discrete Fourier Transform (DFT)

To compute the DFT and IDFT of a given sequence. Plot the magnitude and phase spectrum of a discrete-time sequence

Design steps

1. Using signal flow graph or DFT formula compute the DFT of the sequence x(n).
2. Compute the magnitude response |X(k)|.
3. Compute the phase response of X(k).
4. Compute the N-point IDFT of X (k) and verify that it is same as x (n).

Design example: Compute 8 point DFT of the sequence x(n)={1,1,1,1,0,0,0,0} and hence plot its magnitude and phase response.

Solution:

1 Using signal flow graph (either DFT or DIF FFT) compute 8 point DFT of x (n).

X (k) = [4, 1-2.412j, 0, 1-0.4142j, 0, 1+0.4142j, 0, 1+2.4142j]

2. Represent X (k) in polar form

|X (k)|= [4, 2.6131, 0, 1.0824, 0, 1.0824, 0, 2.6131]

$\angle$ X(k) in(radians) = [0,-1.1781, 0, -0.3927, 0, 0.3927, 0, 1.1781]

3. Plot magnitude and phase spectrum as a function of w=2Π k / N

4. compute 8-point IDFT of X (k) and verify that IDFT [X (k)] =x (n)

```
clc;clear;close;
x=input ('enter the input sequence values x (n)= ');   //[1 0 1 0 1 0 1 0]
N=input('enter the number of frequency samples = ');
```

//To find FFT

```
for k=1:N
   X(k)=0;
  for n=1:length(x)
     X(k)=X(k)+x(n).*exp((-%i).*2.*%pi.*(n-1).*(k-1)./N);
  end
end

X_mag=abs(X);                                  //Magnitude spectrum
X_ph=atan(imag(X),real(X));                    //Phase spectrum
```

//To plot magnitude and phase response of X(k).

```
k=0:N-1;
subplot(211);plot2d3(k,X_mag);xtitle('magnitude response', 'k','|X(k)|')
subplot(212);plot2d3(k,X_ph);xtitle('Phase response', 'k','phase of X(k)')
```

//To find IFFT

```
for n=1:length(x)
    x_cap(n)=0;
   for k=1:N
        Y(k)=X(k).*exp((%i).*2.*%pi.*(n-1).*(k-1)./N)
       x_cap(n)=x_cap(n)+(1/N).*Y(k);
    end
end
disp(real(x_cap),'Reconstructed signal is:')

//Expected result
//FFT: X=[4 0 0 0 4 0 0 0]
//IFFT x=[ 1 0 1 0 1 0 1 0]
```

Problems: (1) $x(n) = (-1)^n$, $0<n<3$
 (2) $x(n) = (2)^{-n} u(n) – (2)^{-n} u(n-4)$

4. Linear filtering in time and frequency domain

a) Linear convolution in time domain

b) Circular convolution in frequency domain

Solution: Calculate the output sequence y[n] = x[n] *h[n] either by using formula or by graphical method or matrix method.

Example: x[n] = {-2,-1,3,2,8}
h[n] = {5,-4,3,2}

a) To perform linear convolution in time domain

```
clc ;clear all;close;
x=input ('enter the input sequence values x(n)= ');
h=input('enter the impulse sequence values h(n) = ');

L1 = length(x);
L2 = length(h);
L3=L1+L2-1

for n = 1: L3
 conv_sum = 0;
 for k= 1: n
 if (((n – k+1) <= L2 ) &( k < =L1 ) )
 conv_sum = conv_sum + x ( k ) * h (n-k +1) ;
 end ;
y(n) = conv_sum ;
end
end
disp(y,' Convolution Sum using Direct Formula Method= ' )

 //Method 2 Using In built Function
f = convol(x,h)
disp(f, ' Convolution Sum Result using Inbuilt Function = ')

//Method 3 Using frequency Domain multiplication
```

```
N = L1 +L2 -1;                    //Linear convolution output length
x = [ x zeros(1 ,N - L1 ) ];
h = [ h zeros(1 ,N - L2 ) ];
f1 = fft(x)
f2 = fft(h)
f3 = f1.* f2 ;                    // Multiplication in frequency domain
f4 = ifft(f3)
disp (f4 , 'Convolution Sum Result DFT and IDFT method = ')

//To plot input, impulse and output signals.
subplot (5,1,1) ;plot2d3(x);xtitle('Input signal x ','n','x[n]');
subplot(5,1,2) ;plot2d3(h);xtitle('Impulse signal h','n','h[n]');
subplot(5,1,3) ;plot2d3(y);xtitle('Output signal ','n','y[n]');
subplot(5,1,4) ;plot2d3(f);xtitle('Output signal ','n','f[n]');
subplot(5,1,5) ;plot2d3(f4);xtitle('Output signal ','n','f4[n]');
```

4. b) Circular convolution in time and frequency domain

// Time domain

```
clc ;clear all;close;
x1=input ('enter the input sequence values x(n)= ');
x2=input('enter the impulse sequence values h(n) = ');
N=input('enter the value of N point circular convolution N = ');
L1=length(x1);
L2=length(x2);
x1 = [x1,zeros(1,(N-L1)];
x2 = [x2,zeros(1,(N-L2)];

for n = 0: N -1
 x3(n+1) = 0;
 for k= 0: N -1
i=modulo(n-k,N)
if (i<0)
i=i+N;
end
x3(n+1) = x3(n+1)+ x1 ( k+1 ) * x2 (i +1) ;
 end
end
disp (x3 , 'Circular Convolution x3[n] = ')
```

//To plot input, impulse and output signals.

```
subplot (3,1,1) ;plot2d3(x1);xtitle('Input signal x1 ','n','x1[n]');
subplot(3,1,2) ;plot2d3(x2);xtitle('Impulse signal x2','n','x2[n]');
subplot(3,1,3) ;plot2d3(x3);xtitle('Output signal x3 ','n','x3[n]');
```

// Frequency domain

```
X1=fft(x1);
X2=fft(x2);
Y=X1.*X2;
y=ifft(Y);
disp (y , 'Circular Convolution y = ')
```

Problems:

1. $x(n) = \delta(n) - 2\,\delta(n-1) + 3\,\delta(n-2)$, $h(n) = u(n) - u(n-3)$
2. $x(n) = \cos(n\pi/4)$, $0<n<3$, $h(n) = \sin(n\pi/4)$, $0<n<3$
3. $x[n] = \{1,2,\underset{\uparrow}{2},1\}$ and $h[n] = \{2, 3, \underset{\uparrow}{1}, 1\}$
4. $x[n]= \{1, 2, 3, 4\}$ and $h[n] = \{4, 2, 3, 3\}$

5. To perform spectral analysis using DFT and study the effect of sampling, leakage, zero padding, and interpolation

```
% This program demonstrates Spectral Leakage whenever either
 % No_cycles(m) or No_Samples is not INTEGER
%Check the result for the following cases
%case(1) fm=10;fs=100;m=2;  // m is number of cycles
%case(2) fm=10;fs=125;m=1;
%case(3) fm=10;fs=125;m=2;
%case(4) fm=200;fs=10000;m=2.5;
%case(5) fm=75;fs=250;m=3;
clc;
fm=input('Enter the message frequency (in Hz):');
fs=input('Enter the sampling frequency (in Hz):');
m=input('Enter the no. of cycles:');
t=0.0001:1/fs:m/fm;
x=sin(2*pi*fm*t);
figure(4);
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot('gnn',t,x);


No_samples=m*fs/fm; // should be non integer to obtain spectral leakage
x_fft=fft(x);
x_mag=abs(x_fft);
N=length(x);
f=(0:N-1).*fs./N;   // frequency axis in Hz
plot2d3(f,x_mag);
//verify the following cases (with and without aliasing for smpling theorem)
```

1. fm=10;fs=100;m=2; (fs>2fm—No aliasing)
2. fm=10;fs=20, m=2; (fs=2fm—No aliasing)
3. fm=10;fs=15, m=2; (fs<2fm—aliasing)

//verify the following cases (examples without spectral leakage)

1. fm=10;fs=100;m=2;

2. fm=10;fs=125;m=2;
3. fm=75;fs=250;m=3;

//verify the following cases (examples with spectral leakage)

1. fm=10;fs=125;m=1;
2. fm=200;fs=100000;m=2.5;

% Effect of Zero padding

```
x= input ('enter the input sequence values x(n)= ');
k= input ('enter the number of zeros to be padded= ');
N=length(x);
x_pad=[x zeros(1,k)];
N1=length(x_pad);
f=0:N-1;
f1=0:N1-1;
X=abs(fft(x));
X_pad= abs(fft(x_pad));
figure(1);
subplot(211), plot2d3 (f,X),xtitle('DFT of original seq','freq','Amp');
subplot(211), plot2d3 (f1,X_pad),xtitle('DFT of zero padded  seq','freq','Amp');
```

% Effect of inserting zeros in between samples (Interpolation)

```
 x= input ('enter the input sequence values x(n)= ');
k= input ('enter the number of zeros to be inserted= ');
x_mod=[];
N=length(x);

for  i= 1: N
x_mod=[ x_mod, x[i], zeros(1,k)];
end
N_mod=length(x_mod);
X=abs(fft(x));
X_pad= abs(fft(x_pad));
figure(1);
subplot(211), plot2d3 (f,X),xtitle('DFT of original seq','freq','Amp');
subplot(211), plot2d3 (f1,X_pad),xtitle('DFT of zero inserted  seq','freq','Amp');
```

```
% Effect of repetition in time domain
x= input ('enter the input sequence values x(n)= ');
k= input ('enter the number of times to be repeated= ');

x_mod=x;
N=length(x);
for  i= 1: k
x_mod=[ x_mod, x];
end
N_mod=length(x_mod);
X=abs(fft(x));
X_pad= abs(fft(x_pad));
figure(1);
subplot(211), plot2d3 (f,X),xtitle('DFT of original seq','freq','Amp');
subplot(211), plot2d3 (f1,X_pad),xtitle('DFT of repeated  seq','freq','Amp');
```

Problems for effecting zero padding and interpolation:

4. x[n] = {1,-3,5,-7}
5. x[n] = {2,6,8,4}

6. Realise an IIR Butterworth filter with pass band edge at 1500Hz and stop band edge at 2000 Hz for a sampling frequency of 8 kHz. Variations of gain within pass band is 1 dB and stop band attenuation of _____ dB. Use bilinear transformation.

Design specifications:

a) rp=passband attenuation(dB)= -1dB

b) rs=stopband attenuation(dB)= -3 dB

c) f1=Passband edge(Hz)=1500 Hz

d) f2=Stopband edge(Hz)=2000 Hz

e) Fs=1/Ts=Sampling rate(samples/sec)=8000 Hz

Design steps:

I. Obtain the equivalent digital filter specifications:

i) w1=2*pi*f1*Ts rad.

ii) w2=2*pi*f2*Ts rad.

II. Prewarp the frequencies w1 & w2:

i) o1=2/Ts*tan(w1/2)

ii) o2=2/Ts*tan(w2/2)

III. Design an analog butterworth filter specifications rp,rs,o1,o2:

i) order of the filter=n=$\log_{10}[(10^{-k1/10}-1)/(10^{-k2/10}-1)]/2*\log_{10}(o1/o2)$

ii) cut-off frequency = oc=$o2/(10^{-k2/10}-1)^{1/2n}$

iii) from the butterworth polynomial table write down the transfer function of the normalized analog butterworth filter Hn(s).

iv) unnormalise the filter using substitution H(s)=Hn(s) with s=s/oc.

IV. Using blt obtain equivalent digital filter system function:

H(z) = H(s), where $s=2/Ts*(1-z^{-1})/(1+z^{-1})$

V. Verify the frequency response as to whether the filter satisfies the given specifications(w1,w2,rp,rs)

VI. For analog filter frequency response, x-axis is in Hz.

That is pass band edge after prewarping in Hz=o1/2*pi.

& stop band edge after prewarping in Hz=o2/2*pi.

For digital filter frequency response, x-axis is in rad. & it is normalized.

That is pass band edge=w1/pi.

And stop band edge=w2/pi.

Design example: Specifications:

a) pass band edge=f1=1500Hz
b) stop band edge=f2=2000 Hz
c) sampling rate =Fs=8000 Hz = 1/Ts
d) passband attenuation = -1db
e) stop attenuation = -3 db

Solution:

I. Equivalent digital specifications:

w_1=(2*pi*1500)/8000=1.178 rad,

w_2=(2*pi*2000)/8000=1.5708 rad

II. Prewarping:

o1 = 2/Ts * tan(w1/2)=10689.73 rad/sec

o2 = 2/Ts * tan(w2/2)=16000 rad/sec

III. Design of analog filter

a) order $n=\log_{10}[(10^{-k1/10}-1)/(10^{-k2/10}-1)]/2*\log_{10}(o1/o2)\cong 2$
b) cut-off frequency $oc=o2/(10^{-k2/10}-1)^{1/2n}$ = 16000 rad/sec
c) $Hn(s) = 1/(s^2 +1.414 s + 1)$
d) $H(s) = H(s) \mid s = s/ oc = 256 * 10^6 /(s^2 +22654.29s+256 * 10^6)$

IV. Digital filter using Bilinear transformation

$H(z) = H(s) \mid s = 2/Ts * (1- z^{-1}) / (1+z^{-1})$

$H(z) = 0.2932z^2+0.586z+0.2932$
----
z2+0.0014z+0.1716

%PROGRAM:

```
f1=input('Enter the pass band edge(Hz)= ');
f2=input('Enter the stop band edge(Hz)= ');
rp=input('Enter the pass band ripple(dB)= ');
rs=input('Enter the stop band attenuation(dB)= ');
fs=input('Enter the sampling rate(Hz)= ');
```

Digital filter specifications(rad)

```
w1=2*%pi*f1*1/fs;
w2=2*%pi*f2*1/fs;
```

Pre warping

```
o1=2*fs*tan(w1/2);
o2=2*fs*tan(w2/2);
```

Design of analog filter

```
num=log10(((10^(-k1/10))-1)/((10^(-k2/10))-1));
den=2*log10(o1/02);
N=ceil(num/den);
oc=o2/((10^(-k2/10)-1)^(1/(2*n)));
hs=analpf(n,'butt',[],oc);
wc=2*atan(oc/(2*fs));
cutoff=wc/(2*%pi);
hz=iir(n,'lp','butt',cutoff,[]);
[hzm fr]=frmag(hz,256);
fr_rad=fr* 2*%pi;
gain_db=20*log10(hzm);
figure;plot2d(freq_rad,gain_db);
```

Problems:

1. Realize an IIR Butterworth filter with pass band edge at 1200 Hz and stop band edge at 2000 Hz for a sampling frequency of 8 kHz. Variations of gain within pass band is 0.5 dB and stop band attenuation of 40 dB. Use bilinear transformation.

2. Realize an IIR Butterworth filter with pass band edge at 1000 Hz and stop band edge at 3000 Hz for a sampling frequency of 8 kHz. Variations of gain within pass band is 2 dB and stop band attenuation of 20 dB. Use bilinear transformation.

7. Design an IIR Chebyshev filter with pass band edge at 1500 Hz and stop band edge at 2000 Hz for a sampling rate of 8000 Hz. Variation of gain within pass ripple is 1 dB and stop band attenuation of ----- dB. Use bilinear transformation.

Design specifications:

a) rp=passband ripple(dB) = + 1 dB
b) rs=stopband attenuation(dB) = - 3 dB
c) f1=passband edge(Hz) = 1500 Hz
d) f2=stopband edge(Hz) = 2000 Hz
e) fs=1/Ts=sampling rate(samples/sec) = 8000 Hz

Design steps:

I. Obtain the equivalent digital filter specifications:

i) w1=2*pi*f1*Ts rad.
ii) w2=2*PI*f2*Ts rad.

II. Prewarp the frequencies w1 & w2:

i) o1=2/Ts*tan(w1/2)
ii) o2=2/Ts*tan(w2/2)

III. Design an analog chebyshev filter for specifications rp,rs,o1,o2:

Determine A and ε from the specifications:

$-10 \log10(1+\varepsilon^2) = rp$

$-10 \log10(A^2) = rs$

Determine $g = ((A^2-1)^{1/2}/ \varepsilon)$.

Obtain the stop band edge for the normalized filter or = o2/01

Order of the filter = $n = \log_{10}[g+(g^2-1)^{1/2}] / \log_{10}[\text{ or } +(or^2 -1)^{1/2}]$

Cut-off frequency = oc= o1

From the chebyshev polynomial table write down the transfer function of the normalized analog Chebyshev Filter Hn(s)=k/ Chebyshev polynomial

Where $k=b_o$ for n odd, $b_o/(1+\varepsilon^2)^{1/2}$ for n even.

Un-normalized the filter using substitution H(s) = Hn(s) with s = s/oc.

Using BLT, obtain equivalent digital filter system function:

$H(z) = H(s)(s=2/Ts*(1-z^{-1})/(1+z^{-1}))$

Verify the frequency response as to whether the filter satisfies the given specifications(w1,w2,rp,rs)

For analog filter frequency response, x-axis is in Hz.

That is pass band edge after prewarping in Hz=o1/2*pi.

& stop band edge after prewarping in Hz=o2/2*pi.

For digital filter frequency response, x-axis is in rad. & it is normalized.

That is pass band edge=w1/pi.

And stop band edge=w2/pi.

Design Example:

a) Pass band edge =f1=1500 Hz
b) Stop band edge=f2 = 2000 Hz
c) Sampling rate=Fs = 8000 Hz
d) Passband attenuation = + 1 dB
e) Stopband attenuation = - 3 dB

Solution:

Equivalent digital specifications:

w1=(2*pi*1500)/8000= 1.178 rad,

w2=(2*pi*2000)/8000= 1.5708 rad;

Prewarping:

o1=2/Ts*tan(w1/2)=10689.73 rad/sec

o2=2/Ts*tan(w2/2)=16000 rad/sec

or=o2/o1=1.496

Design of analog filter

$-10\log(1+\varepsilon^2)= -1.$ $\varepsilon = 0.5088$

$-10\log(A^2) = -3$ $A=1.413$

$g=((A^2-1))^{1/2}/ \varepsilon) = 1.962$

order $n=\log10[g+ (g^2-1)^{1/2}] / \log10[or+(or^2-1)^{1/2}]=2$

cut-off frequency wc=o1=10689.73 rad/sec

$Hn(s)= 0.975/(s^2 +1.095 s +1.1)$

$H(s)= Hn(s) |_{s=s/o1}$

$$H(s)=\frac{112.3*10^6}{s^2+11735.72s+126.01*10^6}$$

Digital Filter using BLT

$$H(z)= H(s)|_{s = 2/T_s*(1-z^{-1})/ (1+z^{-1})}$$

$$H(z) = \frac{0.197z^2+0.3942z+0.197}{z^2-0.45627z+0.3409}$$

PROGRAM:

```
clear all;clc;close;

f1=input('Enter the pass band edge(Hz)= ');
f2=input('Enter the stop band edge(Hz)= ');
rp=input('Enter the pass band ripple(dB)= ');
rs=input('Enter the stop band attenuation(dB)= ');
fs=input('Enter the sampling rate(Hz)= ');

rp_ratio=10^(rp/20);
```

Digital filter specifications(rad)

```
w1=2*%pi*f1*1/fs;
w2=2*%pi*f2*1/fs;
```

Pre warping

```
o1=2*fs*tan(w1/2);
o2=2*fs*tan(w2/2);
```

Design of analog filter

```
or=o2/o1;         //Stop-band edge of normalized lowpass filter
A2 =10.^(-rs/10);
A=sqrt(A2);
epsilon2 = (10.^(-rp/10)-1);
epsilon=sqrt(epsilon2);
g=((A2-1).^0.5./epsilon);

n = (acosh(g))/(acosh(or));
n = ceil(n);
```

```
oc=o1;

wc=2*atan(oc/(2*fs));

hs=analpf(n,'cheb1',[1-rp_ratio],oc);
```

Design of digital filter

```
hz=iir(n,'lp','cheb1',wc/(2*%pi),[1-rp_ratio 1]);
[hzm,fr]=frmag(hz,256);
magz=20*log10(hzm)';
figure();
plot2d(fr*(2*%pi),magz);xtitle('Digital IIR filter: lowpass','frequency(rad)', 'gain(dB)',' ');
```

% Note: Use zoom/axis commands to verify the design specifications.

%Note: If there is a precision problem to verify the filter co-efficients then int32 ( ) can be used.

Problems:

1. Design an IIR Chebyshev filter with pass band edge at 3000 Hz and stop band edge at 4000 Hz for a sampling rate of 9000 Hz. Variation of gain within pass ripple is 1 dB and stop band attenuation of -40 dB. Use bilinear transformation.

2. Design an normalized IIR Chebyshev Type –I filter with pass band edge at 0.6 rad/sec and stop band edge at 1 rad/sec. Variation of gain within pass ripple is 1 dB and stop band attenuation of -35 dB.

8 (a) Design of FIR filters using windowing

Design a digital FIR low pass filter with following specifications.

a) Pass band cut-off frequency :wp=__________rad
b) Pass band ripple :rp=__________dB
c) Stop band cut-off frequency :ws=__________rad
d) Stop band attenuation :rs=__________dB

Choose an appropriate window function and determine impulse response and provide a plot of frequency response of the designed filter.

Design steps:

1. Choose an appropriate window function from the stop band attenuation using the following table.

Sl. No	Transistion width	Window function	Stop band attenuation
1	4π / N	Rectangular window	21 dB
2	8π / N	Barlett (triangular) window	27 dB
3	8π / N	Hanning window	44 dB
4	8π / N	Hamming window	53 dB
5	12π/ N	Blackman window	75 dB

II. Determine the order of the FIR filter (number of coefficients) using the relation
$N \geq 2\pi k / (ws-wp)$
where k=2 for rectangular window and k=4 for Barlett, Hanning and Hamming windows.

<u>**Note:**</u> Choose N as odd. If N is even select next odd integer.

III. Choose the cut-off frequency of the filter as wc=wp.

IV. Choose $\alpha = (N-1)/2$ so that linear phase is ensured.

Barlett (triangular) window : 1 - 2 (n – [(N-1) / 2]) / (N-1)
Hanning window : 0.5 – 0.5cos (2πn) / (N-1)
Hamming window : 0.54 – 0.46cos (2πn) / (N-1)
Blackman window : 0.54 – [0.42cos (2πn) / (N-1)] + 0.8cos (4πn) / (N-1)

V. The impulse response of the filter is given by

$$h(n) = \frac{\sin(wc(n-\alpha))}{\pi(n-\alpha)} * w(n)$$ where w(n) is window function for0<n<N-1 & $n \neq \alpha$

$$= \frac{wc}{\pi} * w(n) \qquad \text{for } n = \alpha$$

$H(e^{jw}) = e^{-jw(\alpha)} * [\sum 2h(n)\cos(w(n-\alpha))+h((N-1)/2)]$

Design example:

Design a digital FIR low pass filter with following specifications.

a) Pass band cut-off frequency	:0.3π rad
b) Pass band ripple	:0.25 dB
c) Stop band cut-off frequency	:0.45π rad
a) Stop band attenuation	: 50 dB

I step: Choose an appropriate window function from the stop band attenuation specification.

As rs = 50 dB select Hamming window.

(Note: Though Blackman window can be used it results in higher transition bandwidth)

II step: Compute the order of the FIR filter (number of coefficients) using the relation

$N \geq 2\pi k/(wp-ws)$[k=4 for Hamming window]

$N \geq 53.5$ (choose N=55)

III step: Choose the cut-off frequency of the filter as wc=wp= 0.3* pi rad.

IV step: Choose $\alpha = (N-1)/2$ so that linear phase is ensured.

V step: The impulse response of the filter is given by

$$h(n-\alpha)= \frac{\sin(wc(n-\alpha))}{\pi (n-\alpha)} * w(n) \quad \text{where } w(n) \text{ is window function} \quad \text{for} 0<n<N-1 \ \& \ n \neq \alpha$$

h(n- α)=sin(0.3*pi(n-27))/(pi*(n-27))*[0.54-0.46*cos(2*pi*n/(N-1))]

for0<n<N-1 & n ≠ α

For ex : The impulse response coefficients are

h(0)=0.00029114 = h(54)

h(1)= -5.9806*10-4 = h(53)……..

Note: when n=α

i.e.   h(27) = wc/pi*w(27) =0.3*1=0.3

VI. Frequency response can be obtained using equation

$H(e^{jw}) = e^{-jw(\alpha)} * [\sum 2h(n)\cos(w(n-\alpha))+h((N-1)/2)]$

Note: The symmetry of the impulse response must be verified and the frequency response must satisfy the specifications.

SCILAB Program

```
clc;
clear all;
close all;

wp=input('enter the pass band edge in rad');
ws=input('enter the stop band edge in rad');
rs=input('enter the stop band ripple in dB');
freq_points=1024;
k=4; %Hamming window (decided based on stop band attenuation)
trw=ws-wp;
N=(k*2*pi/trw);
N=ceil(N);
if rem(N,2)==0
   N=N+1;
end
wc=wp;
aph=(N-1)/2;
for n=0:N -1
        if n== aph
                hdn_minusalph (n+1)=wc/ %pi ;
else
              hdn_minusalph (n+1)= sin(wc .*(n-aph)) ./( %pi .*(n-aph ));
end
end
```

```
n=0:N-1
wndw=window(‘hm’,N);

hn=hdn_minusalph.*wndw';
figure(1); subplot(311);plot(n,wndw);xlabel('n');ylabel('wndw');
subplot(312);plot(n,hdn_minusalph);xlabel('n');ylabel('hdn_minusalph');
subplot(313);plot(n,hn);xlabel('n');ylabel('hn');
h=[hn ' zeros (1, freq_points - length (hn)) ];
H= fft(h);
mag_H=20*log10(abs(H));
ph= atan ( imag (H),real (H));
phase_H=unwrap(ph);
k1=0: freq_points-1;
omega=k*2*pi/( freq_points);
figure(2); subplot (411) ,plot2d (omega(1: freq_points/2) , mag_H (1: freq_points/2) );
 xtitle ( 'Magnitude response' , 'w ( rad ) ' , 'Magnitude (dB)' );
subplot (412) ,plot2d (omega(1: freq_points/2)  , phase_H (1: freq_points/2) );
xtitle ( ' Phase Response ' , 'w ( rad ) ' , ' Phase ( rad ) ' );
```

Problems:

1. Design a digital FIR low pass filter with following specifications.

a) Pass band cut-off frequency	:0.4π rad
b) Pass band ripple	:0.25 dB
c) Stop band cut-off frequency	:0.6π rad
a) Stop band attenuation	: 44 dB

2. Design a digital FIR low pass filter with following specifications.

a) Pass band cut-off frequency	:0.25π rad
b) Pass band ripple	:0.25 dB
c) Stop band cut-off frequency	:0.3π rad
a) Stop band attenuation	: 50 dB

8 b) To design a FIR filter using Kaiser window

Design : Kaiser window is an adjustable window function which provides flexible transition bandwidths. The window function is given by

$w(n) = I_0[\beta \sqrt{(1 - (1 - 2n / (M-1)^2}] / I_0[\beta]$

where I_0 is the modified zero-order Bessel function given by

$I_0(x) = 1 + \sum [(x/2)^k / k !]^2$ which is positive for all real values of x.

The parameter β controls the minimum stopband attenuation and can be chosen to yield different transition bandwidths for the same order.

- If $\beta = 5.658$, then transition width is equal to 7.8 pi /N, and the minimum stopband attenuation is equal to 60 dB.

- If $\beta = 4.538$, then the transition width is equal to 5.8 pi/N, and the minimum stopband attenuation is equal to 50 dB.

Empirical design equations:

Given wp, ws, rp and As, the parameters N and β are given by

transition width = Δw = ws-wp

Filter length $N \sim (A - 7.95 / 2.285 \Delta w) + 1$

$$\text{Parameter } \beta = \begin{cases} 0.1102(A-8.7), & A \geq 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21), & 21 < A < 50 \end{cases}$$

Design example:

Design a digital FIR lowpass filter with the following specifications:

wp= 0.2π,

ws= 0.3π,

Rp= 0.25

dB, As=50 dB

Choose Kaiser window. Determine the impulse response and plot the frequency response.

Steps:

1. Find δp from $A_p = -20\log_{10}\left(\frac{1+\delta_p}{1-\delta_p}\right)$, δp=-0.0144
2. Find δs from $A_s = -20\log_{10}\delta_s$, δs=0.0032
3. $\delta = \min\left(\left|\delta_p\right|, \delta_s\right)$, δ=0.0032
4. $A = -20\log_{10}\delta$, A=49.897 dB
5. Choose the order of the filter using

   N = (A – 7.95 / 2.285 Δw) = 58.4 ≈59
6. Determine β based on the value of N

   β = 0.1102(A-8.7), As, A ≥ 50, β = 4.539
7. Obtain the Kaiser window function using

   w_kaiser = (kaiser(N, beta))';
8. The impulse response of the filter is given by

   h = hd .* w_kaiser;

SCILAB PROGRAM

```
clc;clear all;
wp = 0.2*%pi;
ap =0.25 ;
ws =0.3*%pi ;
as = 50;
trw = ws - wp;
dp = ((10.^(-ap/20))-1)./((10.^(-ap/20))+1);
ds = (10.^(-as/20));
d = min(abs(dp),ds);
A =-20*log10(d);
N = ceil((A-7.95)/(2.285*trw));
r=modulo(N,2);
if r==0
   N=N+1;
end
beta = 0.1102 * (A-8.7);
win=window('kr',N,beta);
wc = wp;
aph=(N-1)/2;
for n=0:N-1
   if n==aph
       hd(n+1)=wc/%pi;
  else
     hd(n+1)= sin(wc.*(n-aph))./(%pi.*(n-aph));

end
end
hn = hd.*win';
n=0:N-1;
subplot(311); plot(n,win); xlabel('n'); ylabel('wndw');
subplot(312); plot(n,hd); xlabel('n'); ylabel('hd');
subplot(313); plot(n,hn); xlabel('n'); ylabel('hn');

hn=[hn' zeros(1,1024-N)];
a=fft(hn);
mag=20*log10(abs(a));
phi=atan(imag(a),real(a));
k=0:1023;
omega=k*2*%pi/1024;
figure;
subplot(211),plot2d(omega(1:512),mag(1:512));
xtitle('Magnitude response','w (rad)','Magnitude(dB)');
subplot(212),plot2d(omega(1:512),unwrap(phi(1:512)));
xtitle('Phase Response','w (rad)','Phase (degrees)');
```

9. Applications of FIR filters:

a) Design a differentiator using a Hamming window of length N=21. Plot the time and frequency domain responses.

b) Design a length-25 digital Hilbert transformer using a Hann window.

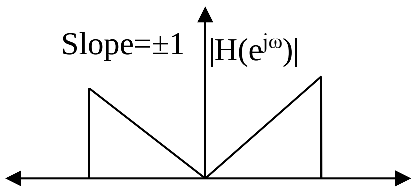
a) Ideal differentiator:

The frequency response of a linear-phase ideal differentiator is given by

$$H_d(e^{jw}) = \begin{cases} j\,\omega, & 0 < \omega < \pi \\ -jw, & -\pi < \omega < 0 \end{cases}$$

The ideal impulse response of a digital differentiator shifted by α with linear phase is given by

$$h_d(n-\alpha) = \begin{cases} \cos \pi (n-\alpha) / (n-\alpha), & n \neq \alpha \\ 0, & n = \alpha \end{cases}$$



Scilab Program:

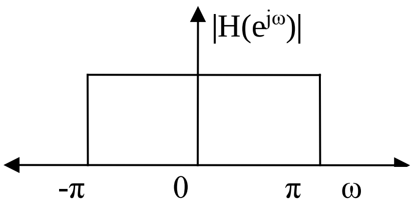
```
clc;clear;close;
N =21;
freq_points=1024;
windowfn =window('hm',N);
n = 0:N-1;
aph = (N-1)/2;
for n=0:N-1
   if n==aph
      hd(n+1)=0;
      else
    hd(n+1)= cos(%pi*(n-aph))./(n-aph);
    end
end
n=0:N-1;
 hn = hd.*windowfn';
 omega=-%pi:2*%pi/(freq_points-1):%pi;
n=0:N-1;
  figure;
plot2d3(n,hn);
hn=[hn' zeros(1,1024-N)];
a=fft(hn);
```

```
mag=abs(a);
magn=fftshift(mag);
figure()
plot2d(omega,magn);
xtitle('Magnitude response','w (rad)','Magnitude');
```

9. b) Design of Hilbert transformer

The ideal frequency response of a linear phase Hilbert transformer is given by

$$H_d(e^{jw}) = \begin{cases} -j\, e^{(-j w\alpha)}, & 0 < w < pi \\ j\, e^{(-j w\alpha)}, & -pi < w < 0 \end{cases}$$



The ideal impulse response is given by

$$h_d(n-\alpha) = \begin{cases} 2/pi \quad (\sin^2 pi(n-\alpha)/2) \,/\, (n-\alpha), & n \neq \alpha \\ 0, \; n = \alpha \end{cases}$$

SCILAB Program

```
clc;clear;close;
 N = input("enter the window length");  //55
windowfn =window('hm',N);
n = 0:N-1;
aph = (N-1)/2;
for n=0:N-1
    if n==aph
        hd(n+1)=0;
   else
 hd(n+1)=(2/%pi)*((sin((%pi/2)*(n-aph)).^2)./(n-aph));
end
end
n=0:N-1;
hn = hd.*windowfn';
omega=-%pi:2*%pi/(1024-1):%pi;
 n=0:N-1;
```

```
  figure;
  plot2d3(n,hn);
  hn=[hn' zeros(1,1024-N)];
   a=fft(hn);
  mag=abs(a);
figure(2);plot(omega,mag);xtitle('Magnitude response','w (rad)','Magnitude');
```

PART – B USING TMS320C 6713 SIMULATOR

Procedure to be followed for using TMS320C6713 simulator CCS 5.5.0

Step 1: Make sure that you are in the CCS edit window and not in the CCS debug(check in the right top corner)

Go to File $\rightarrow$ New $\rightarrow$ CCS Project, enter the project name, output type-executable, Family-6000, Variant - Generic C67xx device

Click on empty project and finish.

Step 2: Project $\rightarrow$ Right Click $\rightarrow$ Properties $\rightarrow$ General

Linker command:C: \ DSK6713\cgtools\C6713dsk.cmd

Runtime support lib:C: \ DSK6713\cgtools\rts6700.lib

Step 3: Build $\Longrightarrow$ c6000 Compiler $\Longrightarrow$ Include options

Add file system $\rightarrow$ C:\ti\ccsv5\tools\compiler\c6000.7.4.4\include

Build $\Longrightarrow$ c6000 linker $\Longrightarrow$ File search path $\Longrightarrow$

Add File system $\rightarrow$ C: \ DSK6713\cgtools\rts6700.lib $\rightarrow$

Add directory $\rightarrow$ File system $\rightarrow$ C:\DSK6713\cgstools

Step 4: Adding Target Configuration File

Project $\rightarrow$ Right click $\rightarrow$ New $\rightarrow$ Target Configuration File $\rightarrow$

C67xx_Sim.ccxml $\rightarrow$ Texas instrument Simulator $\rightarrow$

Connection $\rightarrow$ C67xx CPU cycle Accurate Simulator little Endian $\rightarrow$

Save it

Step 5: Adding source File

Project $\rightarrow$ Right Click $\rightarrow$ New $\rightarrow$ Source File $\rightarrow$ Save file in **.C** format

Step 6: Click on the debug icon (now it is in CCS Debug session).If there are error go back to CCS edit and edit the code and again debug until there are no errors.

Step 7: Run and verify the result.

Step 8: Also, go to view $\longrightarrow$ memory browser $\longrightarrow$ Enter the variable name and set the data type to 32-bit floating point and see the values in the memory browser also.

1. To find linear convolution of two causal sequences

```
    //Linear convolution of given sequences using TMS320C5416/5510
    processor(using C language)
    #include<stdio.h>
    #include<math.h>
    main(){
        float h[4]={2,2,2,2};float x[4]={1,2,3,4};
        float y[10]; int xlen=4; int hlen=4;
        int len;
        int N=xlen+hlen-1;                    //Length of linear convolution
        int k,n;
        for(n=0;n<N;n++){
                y[n]=0;
                for(k=0;k<=n;k++){
                        if(((n-k)<hlen)&(k<xlen))
                        y[n]=y[n]+x[k]*h[n-k];
                }
                printf("%f\n",y[n]);
        }
}
```

NOTE:

Convolution formula

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

2. To find circular convolution of two sequences

```
// This Program Computes N point circular convolution of x1(n) and x2(n)
// Max(L1,L2) <= N < L1+L2-1
// ARRAY LENGTH of x1 and x2 should be N always
# include<stdio.h>
# include<math.h>
# define N 8
float x1[N]={1,2,3,4,5};
float x2[N]={5,4,3,2,1};
float x3[N];
main(){
        int k,n,i;
        for(n=0;n<N;n++)
        {                                   //outer loop for y[n] array
        x3[n]=0;
        for(k=0;k<N;k++)
        {                                   //inner loop for computing each y[n] point
        i=(n-k)%N;                          //compute the index modulo N
        if(i<0) i=i+N;                      //if index is <0, say x[-1], then convert to
                                            //x[N-1]
        x3[n]=x3[n]+x2[k]*x1[i];             //compute output
        }                                   //end of inner for loop
                printf("%f\t",x3[n]);
        }
                                            //end of outer for loop
}                                           //end of main
```

NOTE:

Circular convolution formula

$$x3(n) = \sum_{k=0}^{N-1} x1((n-k))_N \, x2(k)$$

3. To find the N point DFT of a sequence x[n]

```
#include <stdio.h>
#include <math.h>\
float x[4]={1,3,2,5};
        float y[16];
        float z[8]={0,0,0,0,0,0,0,0};                    // To store  the magnitude of DFT
main(){
        float w;
        int n,k, N=8,xlen=4,i;
        for(k=0,i=0;i<N;i++,k=k+2){
                y[k]=0; y[k+1]=0;                        //initialize real & imag parts
                for(n=0;n<xlen;n++){
                        w=-2*3.1416*i*n/N;               //careful about minus sign
                        y[k]=y[k]+x[n]*cos(w);
                        y[k+1]=y[k+1]+x[n]*sin(w);
                }
                z[i]=sqrt(y[k]*y[k]+y[k+1]*y[k+1]);
                printf(" y(%d)= %f+j(%f) \n\n |y(%d)|=  %f\n\n",i,y[k],y[k+1],i,z[i]);
          }
}
```

4. To design a IIR filter using Butterworth prototype and demonstrate the filtering action of a synthesized signal.

```
//This program demonstrates removal of 4KHz Noise from the composite signal
( 1kHz+4KHz)
// STEPS:
// Design of LPBW digital IIR filter with the specification
// fp = 1.5 kHz, fs = 2 kHz. Sampling freq= 10 kHz. kp = -1 dB, k2 = -3 dB
// Take these filter co-eff(numz and denz) and obtain difference equation
// Solve this diff.eqn. with input as combination of Signal and Noise
// Observe the filter response( solution to diff. eqn ) both in Time domain and
// in frequency domain
#include <stdio.h>
#include <math.h>
# define fp     1000.0                          // message in PassBand
# define fn     4000.0                          // Noise in StopBand
# define fs     10000.0
# define pi     3.1416
# define N      50                              // 5 Cycles of input
float b0= 0.2069, b1=0.4137, b2=0.2069;         // numz obtained from SCILAB
float a1= -0.3682, a2=0.1956;                   // denz obtained from SCILAB
int i,j;
float  x1[60],x2[60],x[60],y[70],t,z;           // allot memory for i/p & o/p
main( ){
        int k=0;
for(t=.0001,k=0; t<=5*(1/fp);k++,t=t+(1/fs) ){
                x1[k] = 0.55*sin(2*pi*t*fp);
                x2[k] = 0.5*sin(2*pi*t*fn);
                x[k]= x1[k]+x2[k];              // Combination of Signal & Noise
        }
        for(j=0;j<N;j++){                       // Solution to Diff. equation
                y[j]=b0*x[j];
                if(j>0) y[j]=y[j]+b1*x[j-1]-a1*y[j-1];
                if ((j-1)>0) y[j]=y[j]+b2*x[j-2]-a2*y[j-2];
                printf("%f \t",y[j]);
        }
    }
```

PART – C
Open Ended Experiments

1. Applications related to signal representation
2. Linear filtering of long sequences
3. Applications related to efficient DFT computation
4. Design of digital systems using pole-zero concept