Scilab Manual for
Digital Signal Processing
by Prof R.Senthilkumar, Assistant Professor
Electronics Engineering
Institute of Road and Transport Technology[1]

Solutions provided by
Mr. R.Senthilkumar- Assistant Professor
Electronics Engineering
Institute of Road and Transport Technology

February 12, 2026

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website http://scilab.in

Contents

List of Scilab Solutions 3

1 Generation of Discrete Signals 4

2 Linear and Circular Convolution of two sequences 7

3 Circular convolution using FFT 11

4 Linear Convolution using Circular Convolution 13

5 Calculation of FFT and IFFT of a sequence 15

6 Time and Frequency Response of LTI systems 17

7 Sampling, Verification of Sampling and Effect of aliasing 20

8 Design of FIR Filters Window Design 22

9 Design of FIR Filters Frequency Sampling 29

10 Design of IIR Filters- Butterworth 32

11 Design of IIR Filters Chebyshev 37

12 Decimation by polyphase decomposition 41

13 Periodogram based Spectral Estimation 44

List of Experiments

Solution 1.1	Unit Sample Sequence	4
Solution 1.2	Unit Step Sequence	4
Solution 1.3	Discrete Ramp Sequence	5
Solution 1.4	Exponentially Decreasing Signal	6
Solution 1.5	Exponentially Increasing Signal	6
Solution 2.1	Program for Linear Convolution	7
Solution 2.2	Program to find the Cicrcular Convolution	9
Solution 3.1	Performing Circular COnvolution Using DFT IDFT method	11
Solution 4.1	Performing Linear Convolution using Circular Convolution	13
Solution 5.5	Performing FFT and IFFT of a discrete sequence	15
Solution 6.1	Time and Frequency Response	17
Solution 7.1	Sampling and Reconstruction of a Signal	20
Solution 8.1	Program to Design FIR Low Pass Filter	22
Solution 8.2	rogram to Design FIR High Pass Filter	23
Solution 8.3	Program to Design FIR Band Pass Filter	25
Solution 8.4	Program to Design FIR Band Reject Filter	27
Solution 9.1	Design of FIR LPF Filter using Frequecny Sampling Technique	29
Solution 10.1	Digital IIR First Order Butterworth LPF Filter	32
Solution 10.2	HPF Using Digital Filter Transformation	33
Solution 10.3	BPF using Digital Transformation	34
Solution 10.4	BSF using Digital Transformation	35
Solution 11.1	To Design the Digtial Chebyshev IIR Filter	37
Solution 12.1	Design of Ployphase Decimator	41
Solution 13.1	Periodogram Estimate of Given Discrete Sequence	44
AP 1	sinc function	46

Experiment: 1

Generation of Discrete Signals

Scilab code Solution 1.1 Unit Sample Sequence

```
//Caption:Unit Sample Sequence
clear;
clc;
close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),1,zeros(1,L)];
b = gca();
b.y_location = "middle";
plot2d3('gnn',n,x)
a=gce();
a.children(1).thickness =4;
xtitle('Graphical Representation of Unit Sample
   Sequence','n','x[n]');
```

Scilab code Solution 1.2 Unit Step Sequence

```
//Caption: Unit Step Sequence
```

```
clear;
clc;
close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),ones(1,L+1)];
a=gca();
a.y_location = "middle";
plot2d3('gnn',n,x)
title('Graphical Representation of Unit Step Signal')
xlabel('n');
ylabel('x[n]');
```

Scilab code Solution 1.3 Discrete Ramp Sequence

```
//Caption: Discrete Ramp Sequence
clear;
clc;
close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),0:L];
b = gca();
b.y_location = 'middle';
plot2d3('gnn',n,x)
a=gce();
a.children(1).thickness =2;
xtitle('Graphical Representation of Discrete Unit Ramp Sequence','n','x[n]');
```

Scilab code Solution 1.4 Exponentially Decreasing Signal

```
//Caption: Exponentially Decreasing Signal
clear;
clc;
close;
a =0.5;
n = 0:10;
x = (a)^n;
a=gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x)
a.thickness = 2;
xtitle('Graphical Representation of Exponentially
   Decreasing Signal','n','x[n]');
```

Scilab code Solution 1.5 Exponentially Increasing Signal

```
//Caption:Exponentially Increasing Signal
clear;
clc;
close;
a =1.5;
n =1:10;
x = (a)^n;
a=gca();
a.thickness = 2;
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Exponentially
   Increasing Signal','n','x[n]');
```

Experiment: 2

Linear and Circular Convolution of two sequences

Scilab code Solution 2.1 Program for Linear Convolution

```
//Caption:Program for Linear Convolution
clc;
clear all;
close ;
x = input('enter x seq');
h = input('enter h seq');
m = length(x);
n = length(h);
//Method 1 Using Direct Convolution Sum Formula
for i = 1:n+m-1
    conv_sum = 0;
    for j = 1:i
        if (((i-j+1) <= n)&(j <= m))
            conv_sum = conv_sum + x(j)*h(i-j+1);
        end;
        y(i) = conv_sum;
    end;
end;
disp(y','Convolution Sum using Direct Formula Method
```

```
=')
//Method 2 Using Inbuilt Function
f = convol(x,h)
disp(f,'Convolution Sum Result using Inbuilt Funtion =')
//Method 3 Using frequency Domain multiplication
N = n+m-1;
x = [x zeros(1,N-m)];
h = [h zeros(1,N-n)];
f1 = fft(x)
f2 = fft(h)
f3 = f1.*f2;    // freq domain multiplication
f4 = ifft(f3)
disp(f4,'Convolution Sum Result DFT - IDFT method =')
//f4 =  real(f4)
subplot(3,1,1);
plot2d3('gnn',x)
xtitle('Graphical Representation of Input signal x');
subplot(3,1,2);
plot2d3('gnn',h)
xtitle('Graphical Representation of Impulse signal h');
subplot(3,1,3);
plot2d3('gnn',y)
xtitle('Graphical Representation of Output signal y');
//Result
//enter x seq [1 1 1 1]
//enter h seq [1 2 3]
// Convolution Sum using Direct Formula Method =
//     1.     3.     6.     6.     5.     3.
// Convolution Sum Result using Inbuilt Funtion =
//     1.     3.     6.     6.     5.     3.
// Convolution Sum Result DFT - IDFT method =
//     1.     3.     6.     6.     5.     3.
```

Scilab code Solution 2.2 Program to find the Cicrcular Convolution

```
//Caption: Program to find the Cicrcular Convolution of given
//discrete sequences using Matrix method

clear;
clc;
x1 = [2,1,2,1]; //First sequence
x2 = [1,2,3,4]; //Second sequence
m = length(x1); //length of first sequence
n = length(x2); //length of second sequence
//To make length of x1 and x2 are Equal
if (m >n)
   for i = n+1:m
     x2(i) = 0;
   end
elseif (n>m)
   for i = m+1:n
     x1(i) = 0;
   end
end
N = length(x1);
x3 = zeros(1,N); //x3 = Circular convolution result
a(1) = x2(1);
for j = 2:N
  a(j) = x2(N-j+2);
end
for i =1:N
  x3(1) = x3(1)+x1(i)*a(i);
end
X(1,:)=a;
//Calculation of circular convolution
for k = 2:N
```

```
    for j =2:N
       x2(j) = a(j-1);
    end
       x2(1) = a(N);
       X(k,:)= x2;
    for i = 1:N
        a(i) = x2(i);
        x3(k) = x3(k)+x1(i)*a(i);
    end
end
disp(X,'Circular Convolution Matrix x2[n]=')
disp(x3,'Circular Convolution Result x3[n] = ')
//Result
//Circular Convolution Matrix x2[n]=
//
//     1.     4.     3.     2.
//     2.     1.     4.     3.
//     3.     2.     1.     4.
//     4.     3.     2.     1.
//
// Circular Convolution Result x3[n] =
//
//     14.     16.     14.     16.
```

Experiment: 3

Circular convolution using FFT

Scilab code Solution 3.1 Performing Circular COnvolution Using DFT IDFT method

```
//Caption:Performing Circular COnvolution Using DFT-IDFT method
clear all;
clc;
close;
L = 4; //Length of the Sequence
N = 4;  // N -point DFT
x1 = [2,1,2,1];
x2 = [1,2,3,4];
//Computing DFT
X1 = fft(x1,-1);
X2 = fft(x2,-1);
disp(X1,'DFT of x1[n] is X1(k)=')
disp(X2,'DFT of x1[n] is X2(k)=')
//Multiplication of 2 DFTs
X3 = X1.*X2;
disp(X3,'DFT of x3[n] is X3(k)=')
//Circular Convolution Result
x3 =abs(fft(X3,1))
disp(x3,'Circular Convolution Result x3[n]=')
```

```
//Result
// DFT of x1[n] is X1(k)=
//
//     6.     0     2.     0
//
// DFT of x1[n] is X2(k)=
//
//     10.  - 2. + 2.i  - 2.  - 2. - 2.i
//
// DFT of x3[n] is X3(k)=
//
//     60.     0  - 4.     0
//
// Circular Convolution Result x3[n]=
//
//     14.     16.     14.     16.
```

Experiment: 4

Linear Convolution using Circular Convolution

Scilab code Solution 4.1 Performing Linear Convolution using Circular Convolution

```
//Caption: Performing Linear Convolution using
   Circular Convolution

clear;
clc;
close;
h = [1,2,3];     //Impulse Response of LTI System
x = [1,2,2,1];   //Input Response of LTI System
N1 = length(x);
N2 = length(h);
N = N1+N2-1
disp(N,'Length of Output Response y(n)')
//Padding zeros to Make Length of 'h' and 'x'
//Equal to length of output response 'y'
h1 = [h,zeros(1,N-N2)];
x1 = [x,zeros(1,N-N1)];
//Computing FFT
H = fft(h1,-1);
```

```
X = fft(x1,-1);
//Multiplication of 2 DFTs
Y = X.*H
//Linear Convolution Result
y =abs(fft(Y,1))
disp(X,'DFT of i/p X(k)=')
disp(H,'DFT of impulse sequence H(k)=')
disp(Y,'DFT of Linear Filter o/p Y(k)=')
disp(y,'Linear Convolution result y[n]=')
//Result
// Length of Output Response y(n)
//
//     6.
//
// DFT of i/p X(k)=
//
//     6.  - 3.4641016i    0    0    0    3.4641016i
//
// DFT of impulse sequence H(k)=
//
//     6.    0.5 - 4.330127i  - 1.5 + 0.8660254i    2.  - 1.5 - 0.8660254i    0.5 + 4.330127i
//
// DFT of Linear Filter o/p Y(k)=
//
//     36.  - 15. - 1.7320508i    0    0    0  - 15. + 1.7320508i
//
// Linear Convolution result y[n]=
//
//     1.    4.    9.    11.    8.    3.
```

Experiment: 5

Calculation of FFT and IFFT of a sequence

Scilab code Solution 5.5 Performing FFT and IFFT of a discrete sequence

```
//Caption:Performing FFT and IFFT of a discrete
   sequence
clear;
clc;
close;
L = 4; //Length of the Sequence
N = 4;  // N -point DFT
x = [1,2,3,4];
//Computing DFT
X = fft(x,-1);
disp(X,'FFT of x[n] is X(k)=')
x =abs(fft(X,1))
disp(x,'IFFT of X(k) is x[n]=')
//Plotting the spectrum of Discrete Sequence
subplot(2,1,1)
a=gca();
a.data_bounds=[0,0;5,10];
plot2d3('gnn',0:length(x)-1,x)
b = gce();
```

```
b.children(1).thickness =3;
xtitle('Graphical Representation of Discrete
    Sequence','n','x[n]');
subplot(2,1,2)
a=gce();
a.data_bounds=[0,0;5,10];
plot2d3('gnn',0:length(X)-1,abs(X))
b = gce();
b.children(1).thickness =3;
xtitle('Graphical Representation of Discrete
    Spectrum','k','X(k)');
//Result
//FFT of x[n] is X(k)=
//
//     10.  - 2. + 2.i  - 2.  - 2. - 2.i
//
//IFFT of X(k) is x[n]=
//
//     1.    2.    3.    4.
```

Experiment: 6

Time and Frequency Response of LTI systems

Scilab code Solution 6.1 Time and Frequency Response

```
//Caption: Program to generate and plot the impulse response and frequency
//response of a Linear constant coefficient first order Differential Equation
//[1].Impulse response h(t)= exp(-a*t)u(t), A>0
//[2].Frequency response H(jw) = 1/(jw+a)
clear;
clc;
close;
//[1]. To generate and plot the impulse response
a =1;     //Constant coefficient a =1
Dt = 0.005;
t = 0:Dt:10;
ht = exp(-a*t);
figure(1)
a = gca();
a.y_location = "origin";
plot(t,ht);
xlabel('time t ------->');
```

```
ylabel('h(t)')
title('Impulse Repsonse of Ist Order Linear Constant Coeff. Differential Equ.')
//
//[2]. Finding Frequency response using Continuous Time Fourier Transform
Wmax = 2*%pi*1;          //Analog Frequency = 1Hz
K = 4;
k = 0:(K/1000):K;
W = k*Wmax/K;
HW = ht* exp(-sqrt(-1)*t'*W) * Dt;
HW_Mag = abs(HW);
W = [-mtlb_fliplr(W), W(2:1001)]; // Omega from -Wmax to Wmax
HW_Mag = [mtlb_fliplr(HW_Mag),HW_Mag(2:1001)];
[HW_Phase,db] = phasemag(HW);
HW_Phase = [-mtlb_fliplr(HW_Phase),HW_Phase(2:1001)];
figure(2)
//Plotting Magnitude Response
subplot(2,1,1);
a = gca();
a.y_location = "origin";
plot(W,HW_Mag);
xlabel('Frequency in Radians/Seconds---> W');
ylabel('abs(H(jW))')
title('Magnitude Response')
//Plotting Phase Reponse
subplot(2,1,2);
a = gca();
a.y_location = "origin";
a.x_location = "origin";
plot(W,HW_Phase*%pi/180);
xlabel'                                Frequency in Radians/Seconds---> W');
ylabel('                                                                    <H(jW)')
```

```
title('Phase Response in Radians')
```

Experiment: 7

Sampling, Verification of Sampling and Effect of aliasing

check Appendix AP 1 for dependency:

```
sincnew.sce
```

Scilab code Solution 7.1 Sampling and Reconstruction of a Signal

```
//Caption: Sampling and Reconstruction of a Signal x(t) = exp(-A*|t|)
//Discrete Time Sampled Signal x(nT)= exp(-A*|nT|)
//Following Sampling Frequencies are used:
//[1].Fs = 1 Hz [2].Fs = 2 Hz [3].Fs = 4Hz [4].Fs =20 Hz [5].Fs =100Hz
//Aliasing Effect: As the Sampling frequency increases aliasing effect decreases
clear;
clc;
close;
// Analog Signal
A =1;     //Amplitude
Dt = 0.005;
t = -2:Dt:2;
```

```
//Continuous Time Signal
xa = exp(-A*abs(t));
//Discrete Time Signal
Fs =input('Enter the Sampling Frequency in Hertz');
    //Fs = 1Hz,2Hz,4Hz,20Hz,100Hz
Ts = 1/Fs;
nTs = -2:Ts:2;
x = exp(-A*abs(nTs));
// Analog Signal reconstruction
Dt = 0.005;
t = -2:Dt:2;
Xa = x *sincnew(Fs*(ones(length(nTs),1)*t-nTs'*ones
    (1,length(t))));
//Plotting the original signal and reconstructed
    signal
subplot(2,1,1);
a =gca();
a.x_location = "origin";
a.y_location = "origin";
plot(t,xa);
xlabel('t in sec.');
ylabel('xa(t)')
title('Original Analog Signal')
subplot(2,1,2);
a =gca();
a.x_location = "origin";
a.y_location = "origin";
xlabel('t in sec.');
ylabel('xa(t)')
title('Reconstructed Signal using sinc function, Fs
    = 100Hz');
plot(t,Xa);
```

Experiment: 8

Design of FIR Filters Window Design

Scilab code Solution 8.1 Program to Design FIR Low Pass Filter

```
//Caption: Program to Design FIR Low Pass Filter
clc;
close;
M = input('Enter the Odd Filter Length =');
                //Filter length
Wc = input('Enter the Digital Cutoff frequency =');
   //Digital Cutoff frequency
Tuo = (M-1)/2      //Center Value
for n = 1:M
    if (n == Tuo+1)
      hd(n) = Wc/%pi;
    else
      hd(n) =  sin(Wc*((n-1)-Tuo))/(((n-1)-Tuo)*%pi)
         ;
    end
end
//Rectangular Window
for n = 1:M
  W(n) = 1;
```

```
end
//Windowing Fitler Coefficients
h = hd.*W;
disp(h,'Filter Coefficients are')

[hzm,fr]=frmag(h,256);
hzm_dB = 20*log10(hzm)./max(hzm);
subplot(2,1,1)
plot(2*fr,hzm)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude');
title('Frequency Response 0f FIR LPF using
    Rectangular window')
xgrid(1)
subplot(2,1,2)
plot(2*fr,hzm_dB)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response 0f FIR LPF using
    Rectangular window')
xgrid(1)
//Result
//Enter the Odd Filter Length = 7
//Enter the Digital Cutoff frequency = %pi/2
//
// Filter Coefficients are
//
//  - 0.1061033
//    1.949D-17  = 0.0
//    0.3183099
//    0.5
//    0.3183099
//    1.949D-17 = 0.0
//  - 0.1061033
```

Scilab code Solution 8.2 rogram to Design FIR High Pass Filter

```
//Caption: Program to Design FIR High Pass Filter
clear;
clc;
close;
M = input('Enter the Odd Filter Length =');
                //Filter length
Wc = input('Enter the Digital Cutoff frequency =');
   //Digital Cutoff frequency
Tuo = (M-1)/2      //Center Value
for n = 1:M
    if (n == Tuo+1)
      hd(n) = 1-Wc/%pi;
    else
      hd(n) = (sin(%pi*((n-1)-Tuo)) -sin(Wc*((n-1)-
         Tuo)))/(((n-1)-Tuo)*%pi);
    end
end
//Rectangular Window
for n = 1:M
  W(n) = 1;
end
//Windowing Fitler Coefficients
h = hd.*W;
disp(h,'Filter Coefficients are')
[hzm,fr]=frmag(h,256);
hzm_dB = 20*log10(hzm)./max(hzm);
subplot(2,1,1)
plot(2*fr,hzm)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude');
title('Frequency Response 0f FIR HPF using
   Rectangular window')
xgrid(1)
subplot(2,1,2)
plot(2*fr,hzm_dB)
xlabel('Normalized Digital Frequency W');
```

```
ylabel('Magnitude in dB');
title('Frequency Response 0f FIR HPF using
   Rectangular window')
xgrid(1)
//Result
//Enter the Odd Filter Length = 5
//Enter the Digital Cutoff frequency = %pi/4
//Filter Coefficients are
//
//  - 0.1591549
//  - 0.2250791
//    0.75
//  - 0.2250791
//  - 0.1591549
```

Scilab code Solution 8.3 Program to Design FIR Band Pass Filter

```
//Caption: Program to Design FIR Band Pass Filter
clear;
clc;
close;
M = input('Enter the Odd Filter Length =');
                //Filter length
//Digital Cutoff frequency [Lower Cutoff, Upper
   Cutoff]
Wc = input('Enter the Digital Cutoff frequency =');
Wc2 = Wc(2)
Wc1 = Wc(1)
Tuo = (M-1)/2      //Center Value
hd = zeros(1,M);
W = zeros(1,M);
for n = 1:11
  if (n == Tuo+1)
     hd(n) = (Wc2-Wc1)/%pi;
  else
```

```
        n
    hd(n) = (sin(Wc2*((n-1)-Tuo)) -sin(Wc1*((n-1)-Tuo)))/(((n-1)-Tuo)*%pi);
  end
  if(abs(hd(n))<(0.00001))
    hd(n)=0;
  end
end
hd;
//Rectangular Window
for n = 1:M
  W(n) = 1;
end
//Windowing Fitler Coefficients
h = hd.*W;
disp(h,'Filter Coefficients are')
[hzm,fr]=frmag(h,256);
hzm_dB = 20*log10(hzm)./max(hzm);
subplot(2,1,1)
plot(2*fr,hzm)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude');
title('Frequency Response 0f FIR BPF using Rectangular window')
xgrid(1)
subplot(2,1,2)
plot(2*fr,hzm_dB)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response 0f FIR BPF using Rectangular window')
xgrid(1)
//Result
//Enter the Odd Filter Length = 11
//Enter the Digital Cutoff frequency = [%pi/4,3*%pi/4]
//Filter Coefficients are
// 0.    0.    0.  - 0.3183099    0.    0.5    0.  -
```

```
0.3183099    0.    0.    0.
```

Scilab code Solution 8.4 Program to Design FIR Band Reject Filter

```
//Caption: Program to Design FIR Band Reject Filter
clear ;
clc;
close;
M = input('Enter the Odd Filter Length =');
                    //Filter length
//Digital Cutoff frequency [Lower Cutoff, Upper
   Cutoff]
Wc = input('Enter the Digital Cutoff frequency =');
Wc2 = Wc(2)
Wc1 = Wc(1)
Tuo = (M-1)/2      //Center Value
hd = zeros(1,M);
W = zeros(1,M);
for n = 1:M
 if (n == Tuo+1)
 hd(n) = 1-((Wc2-Wc1)/%pi);
 else
 hd(n)=(sin(%pi*((n-1)-Tuo))-sin(Wc2*((n-1)-Tuo))+
    sin(Wc1*((n-1)-Tuo)))/(((n-1)-Tuo)*%pi);
 end
  if(abs(hd(n))<(0.00001))
    hd(n)=0;
  end
end

//Rectangular Window
for n = 1:M
  W(n) = 1;
end
//Windowing Fitler Coefficients
```

```
h = hd.*W;
disp(h,'Filter Coefficients are')
[hzm,fr]=frmag(h,256);
hzm_dB = 20*log10(hzm)./max(hzm);
subplot(2,1,1)
plot(2*fr,hzm)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude');
title('Frequency Response 0f FIR BSF using Rectangular window')
xgrid(1)
subplot(2,1,2)
plot(2*fr,hzm_dB)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response 0f FIR BSF using Rectangular window')
xgrid(1)
//Result
//Enter the Odd Filter Length = 11
//Enter the Digital Cutoff frequency =[%pi/3,2*%pi/3]
//Filter Coefficients are
//column 1 to 9
//    0.  - 0.1378322    0.    0.2756644    0.    0.6666667    0.    0.2756644    0.
//column 10 to 11
//  - 0.1378322    0.
```

Experiment: 9

Design of FIR Filters Frequency Sampling

Scilab code Solution 9.1 Design of FIR LPF Filter using Frequecny Sampling Technique

```
//Cpation: Design of FIR LPF Filter using Frequecny
   Sampling Technique

clear;
clc;
close;
M =15;
Hr = [1,1,1,1,0.4,0,0,0];
for k =1:length(Hr)
    G(k)=((-1)^(k-1))*Hr(k);
end
h = zeros(1,M);
U = (M-1)/2
for n = 1:M
  h1 = 0;
  for k = 2:U+1
    h1 =G(k)*cos((2*%pi/M)*(k-1)*((n-1)+(1/2)))+h1;
  end
```

```
 h(n) = (1/M)* (G(1)+2*h1);
end
disp(h,'Filter Coefficients are h(n)=')
[hzm,fr]=frmag(h,256);
hzm_dB = 20*log10(hzm)./max(hzm);
subplot(2,1,1)
plot(2*fr,hzm)
a=gca();
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude');
title('Frequency Response 0f FIR LPF using Frequency Sampling Technique with M = 15 with Cutoff Frequency = 0.466')
xgrid(2)
subplot(2,1,2)
plot(2*fr,hzm_dB)
a=gca();
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response 0f FIR LPF using Frequency Sampling Technique with M = 15 with Cutoff Frequency = 0.466')
xgrid(2)
//Result
//Filter Coefficients are h(n)=
//column 1 to 7
//
// -0.0141289  -0.0019453  0.04  0.0122345  -0.0913880  -0.0180899  0.3133176
//
//column  8 to 14
//
//0.52    0.3133176  - 0.0180899  - 0.0913880  0.0122345    0.04  - 0.0019453
//
//column 15
//
//  - 0.0141289
```

Experiment: 10

Design of IIR Filters-Butterworth

Scilab code Solution 10.1 Digital IIR First Order Butterworth LPF Filter

```
//Caption: To design a digital IIR First Order Butterworth LPF Filter
//Using Bilinear Transformation
clear all;
clc;
close;
s = poly(0,'s');
Omegac = 0.2*%pi;   // Cutoff frequency
H = Omegac/(s+Omegac);  //Analog first order Butterworth filter tranfer function
T =1;//Sampling period T = 1 Second
z = poly(0,'z');
Hz = horner(H,(2/T)*((z-1)/(z+1))) //Bilinear Transformation
HW  =frmag(Hz(2),Hz(3),512);  //Frequency response for 512 points
W = 0:%pi/511:%pi;
a=gca();
```

```
a.thickness = 1;
plot(W/%pi,HW,'r')
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of Single pole LPF Filter Cutoff frequency = 0.2*pi','Normalized Digital Frequency--->','Magnitude');
```

Scilab code Solution 10.2 HPF Using Digital Filter Transformation

```
//Caption: To design First Order Butterworth Low Pass Filter and covert it into
// HPF Using Digital Filter Transformation
clear all;
clc;
close;
s = poly(0,'s');
Omegac = 0.2*%pi; //Filter cutoff frequency
H = Omegac/(s+Omegac); //First order Butterworth IIR filter
T =1;//Sampling period T = 1 Second
z = poly(0,'z');
Hz_LPF = horner(H,(2/T)*((z-1)/(z+1))); //Bilinear Transformation
alpha = -(cos((Omegac+Omegac)/2))/(cos((Omegac-Omegac)/2));
HZ_HPF=horner(Hz_LPF,-(z+alpha)/(1+alpha*z))//LPF to HPF digital transformation
HW  =frmag(HZ_HPF(2),HZ_HPF(3),512); //Frequency response for 512 points
W = 0:%pi/511:%pi;
a=gca();
a.thickness = 1;
plot(W/%pi,HW,'r')
```

```
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of Single pole HPF Filter Cutoff frequency = 0.2*pi','Normalized Digital Frequency W/pi--->','Magnitude');
```

Scilab code Solution 10.3 BPF using Digital Transformation

```
///Caption:To Design a Digital IIR Butterworth LPF Filter from Analog IIR
//Butterworth Filter and LPF to BPF using Digital Transformation
clear all;
clc;
close;
omegaP = 0.2*%pi; //Filter cutoff frequency
omegaL =  (1/5)*%pi;  //Lower Cutoff frequency for BSF
omegaU =  (3/5)*%pi;  //Upper Cutoff frequency for BSF
z=poly(0,'z');
H_LPF = (0.245)*(1+(z^-1))/(1-0.509*(z^-1)); //Bilinear transformation
alpha = (cos((omegaU+omegaL)/2)/cos((omegaU-omegaL)/2));//parameter 'alpha'
//parameter 'k'
k = (cos((omegaU - omegaL)/2)/sin((omegaU - omegaL)/2))*tan(omegaP/2);
NUM =-((z^2)-((2*alpha*k/(k+1))*z)+((k-1)/(k+1)));
DEN = (1-((2*alpha*k/(k+1))*z)+(((k-1)/(k+1))*(z^2)));
HZ_BPF=horner(H_LPF,NUM/DEN);//LPF to BPF conversion using digital transformation
disp(HZ_BPF,'Digital BPF IIR Filter H(Z)= ');
```

```
HW  =frmag(HZ_BPF(2),HZ_BPF(3),512);//frequency response
W = 0:%pi/511:%pi;
a=gca();
a.thickness = 1;
plot(W/%pi,HW,'r')
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of BPF Filter cutoff frequency [0.2,0.6]','Normalized Digital Frequency--->','Magnitude');
```

Scilab code Solution 10.4 BSF using Digital Transformation

```
//Caption:To Design a Digital IIR Butterworth LPF Filter from Analog IIR
//Butterworth Filter and LPF to BSF using Digital Transformation
clear all;
clc;
close;
omegaP = 0.2*%pi;   //Filter cutoff frequency
omegaL =  (1/5)*%pi;   //Lower Cutoff frequency for BSF
omegaU =  (3/5)*%pi;   //Upper Cutoff frequency for BSF
z=poly(0,'z');
H_LPF = (0.245)*(1+(z^-1))/(1-0.509*(z^-1))//Bilinear transformation
alpha = (cos((omegaU+omegaL)/2)/cos((omegaU-omegaL)/2)); //parameter 'alpha'
k = tan((omegaU - omegaL)/2)*tan(omegaP/2); //parameter 'k'
NUM =((z^2)-((2*alpha/(1+k))*z)+((1-k)/(1+k))); //
```

```
Numerator
DEN = (1-((2*alpha/(1+k))*z)+(((1-k)/(1+k))*(z^2)));
    //Denominator
HZ_BSF=horner(H_LPF,NUM/DEN); //LPF to BSF
    conversion using digital transformation
HW  =frmag(HZ_BSF(2),HZ_BSF(3),512); //frequency
    response for 512 points
W = 0:%pi/511:%pi;
a=gca();
a.thickness = 1;
plot(W/%pi,HW,'r')
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of BSF Filter cutoff freq
    [0.2,0.6] ','Normalized Digital Frequency--->','
    Magnitude');
```

Experiment: 11

Design of IIR Filters Chebyshev

Scilab code Solution 11.1 To Design the Digtial Chebyshev IIR Filter

```
//Program To Design the Digtial Chebyshev IIR Filter
clear;
clc;
close;
Wp = input('Enter the Digital Pass Band Edge
    Frequency');
Ws = input('Enter the Digital Stop Band Edge
    Frequency');
T = input('Sampling Interval')
OmegaP = (2/T)*tan(Wp/2)
OmegaS = (2/T)*tan(Ws/2)
Delta1 = input('Enter the Pass Band Ripple');
Delta2 = input('Enter the Stop Band Ripple');
Delta = sqrt(((1/Delta2)^2)-1)
Epsilon = sqrt(((1/Delta1)^2)-1)
N = (acosh(Delta/Epsilon))/(acosh(OmegaS/OmegaP))
N = ceil(N)
OmegaC = OmegaP/((((1/Delta1)^2)-1)^(1/(2*N)))
[pols,gn] = zpch1(N,Epsilon,OmegaP)
```

```
Hs = poly(gn,'s','coeff')/real(poly(pols,'s'))
z = poly(0,'z');
Hz = horner(Hs,((2/T)*((z-1)/(z+1))))
HW  =frmag(Hz(2),Hz(3),512);  //Frequency response for 512 points
W = 0:%pi/511:%pi;
a=gca();
a.thickness = 1;
plot(W/%pi,abs(HW),'r')
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of Chebyshev LPF Filter','Normalized Digital Frequency--->','Magnitude in dB');
//RESULT
//Enter the Digital Pass Band Edge Frequency 0.2*%pi
//Enter the Digital Stop Band Edge Frequency 0.6*%pi
//Sampling Interval 1
// T  =
//
//     1.
// OmegaP  =
//
//     0.6498394
// OmegaS  =
//
//     2.7527638
//Enter the Pass Band Ripple 0.8
//Enter the Stop Band Ripple 0.2
// Delta  =
//
//     4.8989795
// Epsilon  =
//
//     0.75
// N  =
//
```

```
//      1.2079548
// N   =
//
//      2.
// OmegaC   =
//
//      0.7503699
// gn   =
//
//      0.2815275
// pols   =
//
//  - 0.2652958 + 0.5305916i  - 0.2652958 -
    0.5305916i
// Hs   =
//
//                0.2815275
//     ---------------------------
//                                 2
//     0.3519094 + 0.5305916s + s
// Hz   =
//
//                                        2
//     0.2815275 + 0.5630550z + 0.2815275z
//     ---------------------------------------
//                                        2
//     3.2907261 - 7.2961813z + 5.4130926z
//-->0.5*0.5629
// ans   =
//
//     0.28145
//
//-->Hz(2)= Hz(2)/5.4130926
// Hz   =
//
//                                        2
//     0.0520086 + 0.1040172z + 0.0520086z
//     ---------------------------------------
```

```
//                                          2
//     3.2907261 - 7.2961813z + 5.4130926z
//
//-->Hz(3) = Hz(3)/5.4130926
// Hz  =
//
//                                          2
//     0.0520086 + 0.1040172z + 0.0520086z
//     ------------------------------------
//                                     2
//         0.6079198 - 1.3478767z + z
//
```

Experiment: 12

Decimation by polyphase decomposition

Scilab code Solution 12.1 Design of Ployphase Decimator

```
//Caption:Decimation by 2, Filter Length = 30
//Cutoff Frequency Wc = %pi/2
//Pass band Edge frequency fp = 0.25 and a Stop band edge frequency fs = 0.31
// Choose the number of cosine functions and create a dense grid
// in [0,0.25] and [0.31,0.5]
//magnitude for pass band = 1 & stop band = 0 (i.e) [1 0]
//Weighting function =[2 1]
clear;
clc;
close;
M = 30;  //Filter Length
D = 2;  //Decimation Factor = 2
Wc = %pi/2;  //Cutoff Frequency
Wp = Wc/(2*%pi);  //Passband Edge Frequency
Ws = 0.31;  //Stopband Edge Frequency
hn=eqfir(M,[0 Wp;Ws .5],[1 0],[2 1]);
```

```
disp(hn,'The LPF Filter Coefficients are:')
//Obtaining Polyphase Filter Coefficients from hn
p = zeros(D,M/D);
for k = 1:D
  for n = 1:(length(hn)/D)
    p(k,n) = hn(D*(n-1)+k);
  end
end
disp(p,'The Polyphase Decimator for D =2 are:')
//Result
//The LPF Filter Coefficients are:
//column 1 to 7
//0.0060203  - 0.0128037  - 0.0028534    0.0136687  - 0.0046761  - 0.0197002    0.0159915

//column  8 to 14
//0.0213811  - 0.0349808  - 0.0156251    0.0640230  - 0.0073600  - 0.1187325    0.0980522
//column 15 to 21
//0.4922476    0.4922476    0.0980522  - 0.1187325  - 0.0073600    0.0640230  - 0.0156251
//column 22 to 28
//- 0.0349808    0.0213811    0.0159915  - 0.0197002  - 0.0046761  0.0136687  - 0.0028534

//column 29 to 30
//- 0.0128037    0.0060203

//The Polyphase Decimator for D =2 are:
//column 1 to 7
//0.0060203  - 0.0028534  - 0.0046761    0.0159915  - 0.0349808    0.0640230  - 0.1187325
//- 0.0128037    0.0136687  - 0.0197002    0.0213811  - 0.0156251  - 0.0073600  0.0980522

//column  8 to 14
//0.4922476    0.0980522  - 0.0073600  - 0.0156251  0.0213811  - 0.0197002    0.0136687
```

```
//0.4922476  - 0.1187325    0.0640230  - 0.0349808
      0.0159915  - 0.0046761  - 0.0028534
//column 15
//- 0.0128037
//   0.0060203
```

Experiment: 13

Periodogram based Spectral Estimation

Scilab code Solution 13.1 Periodogram Estimate of Given Discrete Sequence

```
//Caption: Periodogram Estimate of Given Discrete Sequence
//x(n) ={1,0,2,0,3,1,0,2}
//using DFT
clear;
clc;
close;
N =8; //8-point DFT
x = [1,0,2,0,3,1,0,2]; //given discrete sequence
X = dft(x,-1); //8-point DFT of given discrete sequence
Pxx = (1/N)*(abs(X).^2); //Peridogram Estimate
disp(X,'DFT of x(n)is X(k)=')
disp(Pxx,'Peridogram of x(n) is Pxx(k/N)=')
figure(1)
a = gca();
a.data_bounds =[0,0;8,11];
plot2d3('gnn',[1:N],Pxx)
```

```
a.foreground = 5;
a.font_color = 5;
a.font_style = 5;
title('Peridogram Estimate')
xlabel('Discrete Frequency Variable K ------>')
ylabel('Periodogram Pxx (k /N) ---->')
//Result
//DFT of x(n)is X(k)=
//
//     9.
//   - 1.2928932 + 0.1213203i
//     2. + i
//   - 2.7071068 + 4.1213203i
//     3. - 3.674D-16i
//   - 2.7071068 - 4.1213203i
//     2. - i
//   - 1.2928932 - 0.1213203i
//
// Peridogram of x(n) is Pxx(k/N)=
//
//     10.125
//     0.2107864
//     0.625
//     3.0392136
//     1.125
//     3.0392136
//     0.625
//     0.2107864
```

Appendix

Scilab code AP 1

```
function [y]=sincnew(x)
i=find(x==0);
x(i)= 1;        // don't need this is /0 warning is
    off
y = sin(%pi*x)./(%pi*x);
y(i) = 1;
endfunction
```

sinc function