Scilab Manual for
Digital Signal Processing
by Mr Vijay P Sompur
Electronics Engineering
Visvesvraya Technological University[1]

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

November 21, 2024

[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 4

1 Verification of Sampling theorem. 6

2 Impulse response of a given system 11

3 Linear Circular convolution of two given sequences 14

4 Autocorrelation of a given sequence and verification of its properties. 18

5 Cross correlation of given sequences and verification of its properties. 21

6 Solving a given difference equation. 24

7 Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum. 26

8 Linear convolution of two sequences using DFT and IDFT. 30

9 Circular convolution of two given sequences using DFT and IDFT 33

10 Design and implementation of FIR filter to meet given specifications. 35

11 Design and implementation of IIR filter to meet given specifications. 39

12 Circular convolution of two given sequences 49

List of Experiments

Solution 1.1	Verification of Sampling Theorem	6
Solution 2.2	Program to find impulse response and Frequency Response of a system	11
Solution 3.1	Program to Compute the Convolution of Two Sequences	14
Solution 4.1	Program to Compute the Autocorrelation of a Sequence And verfication of Autocorrelation property	18
Solution 5.1	Program to Compute the Crosscorrelation of a Sequence And verfication of crosscorrelation property	21
Solution 6.1	Solving Difference Equation Direct Form II Realization	24
Solution 7.1	Program to find the spectral information of discrete time signal Calculation of DFT and IDFT	26
Solution 8.1	Linear Convolution using Circular Convolution DFT IDFT method	30
Solution 9.1	Circular Convolution using DFT IDFT method .	33
Solution 10.1	To Design an Low Pass FIR Filter	35
Solution 11.1	To obtain Digital IIR Butterworth low pass filter Frequency response	39
Solution 11.2	To obtain Digital IIR Chebyshev low pass filter Frequency response	43
Solution 12.1	Program to perform circular convolution of two sequences	49

List of Figures

1.1 Verification of Sampling Theorem 9
1.2 Verification of Sampling Theorem 10

2.1 Program to find impulse response and Frequency Response of a system 12

3.1 Program to Compute the Convolution of Two Sequences . . 15

7.1 Program to find the spectral information of discrete time signal Calculation of DFT and IDFT 27

8.1 Linear Convolution using Circular Convolution DFT IDFT method 31

10.1 To Design an Low Pass FIR Filter 36

11.1 To obtain Digital IIR Butterworth low pass filter Frequency response 40
11.2 To obtain Digital IIR Chebyshev low pass filter Frequency response 44

Experiment: 1

Verification of Sampling theorem.

Scilab code Solution 1.1 Verification of Sampling Theorem

```
//Caption: Verification of Sampling Theorem
//[1].Right Sampling [2]. Under Sampling [3]. Over
   Sampling
clc;
close;
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);
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot(tm,x);
title('ORIGINAL SIGNAL');
xlabel('Time');
```

```
ylabel('Amplitude');
xgrid(1)
//Sampling Rate(Nyquist Rate)=2*fm
fnyq=2*fm;
// UNDER SAMPLING
fs=(3/4)*fnyq;
n=0:1/fs:k/fm;
xn=A*cos(2*%pi*fm*n);
figure(2);
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,xn);
plot(n,xn,'r');
title('Under Sampling');
xlabel('Time');
ylabel('Amplitude');
legend('Sampled Signal', 'Reconstructed Signal');
xgrid(1)
//NYQUIST SAMPLING
fs=fnyq;
n=0:1/fs:k/fm;
xn=A*cos(2*%pi*fm*n);
figure(3);
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,xn);
plot(n,xn,'r');
title('Nyquist Sampling');
xlabel('Time');
ylabel('Amplitude');
legend('Sampled Signal', 'Reconstructed Signal');
xgrid(1)
//OVER SAMPLING
fs=fnyq*10;
n=0:1/fs:k/fm;
xn=A*cos(2*%pi*fm*n);
```

```
figure(4);
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,xn);
plot(n,xn,'r');
title('Over Sampling');
xlabel('Time');
ylabel('Amplitude');
legend('Sampled Signal', 'Reconstructed Signal');
xgrid(1)
//Result
//Enter the input signal frequency:100
//
//Enter the number of Cycles of input signal:2
//
//Enter the amplitude of input signal:2
```

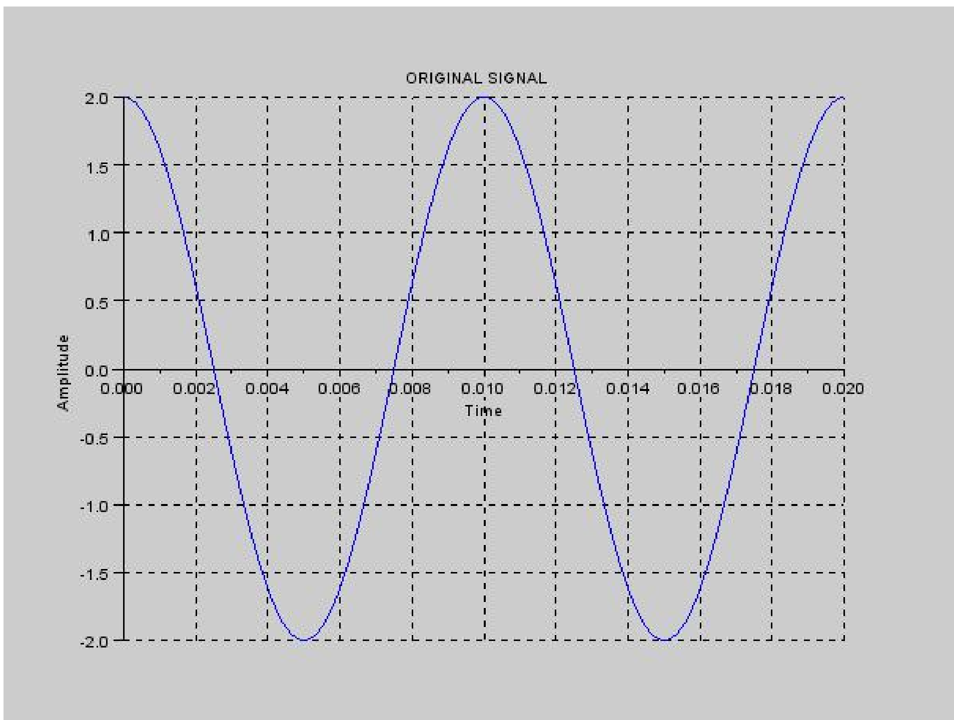

Figure 1.1: Verification of Sampling Theorem

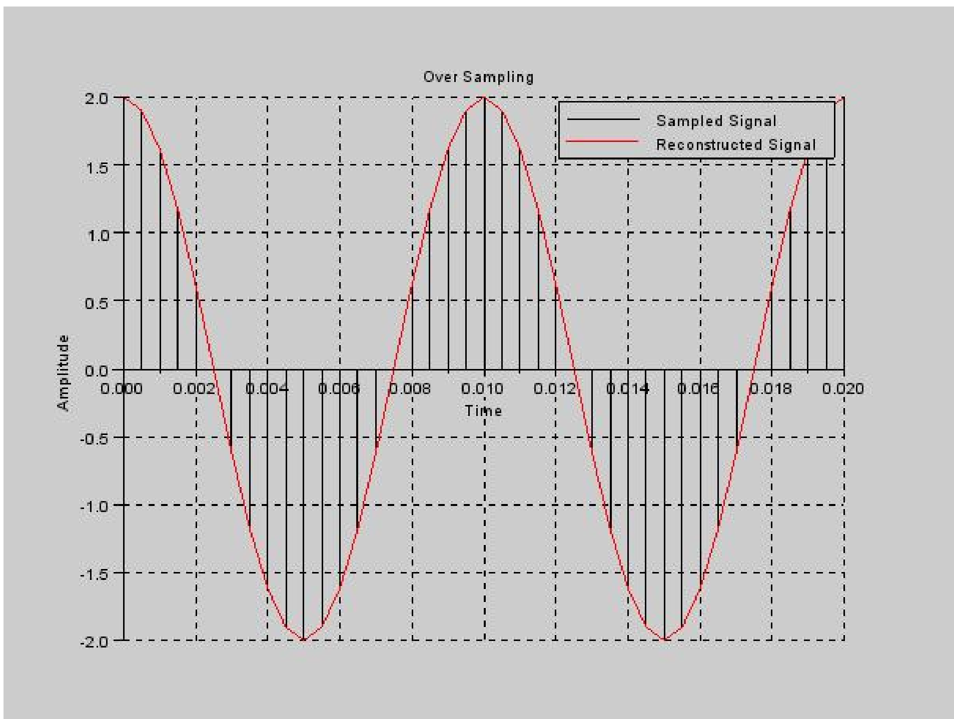

Figure 1.2: Verification of Sampling Theorem

Experiment: 2

Impulse response of a given system

Scilab code Solution 2.2 Program to find impulse response and Frequency Response of a system

```
//Caption: Program to find impulse response and
//Frequency Response of a system
//y[n] = a*y[n-1]+x[n]
//Assume y[n] = h[n], x[n]=delta[n]=unit impulse
    response
//a = 0.9
//h[n] = 0.9*h[n-1]+delta[n]
clc;
clear;
close;
a = 0.9;//constant a = 0.9 less than 1
h0 = 1;
h1 = a; //first two values of impulse response
h = [h0,h1,zeros(1,100)];
for i = 1:100
  h(i+2) = ((a)^(i+1))*h(i+1);//impulse response
```

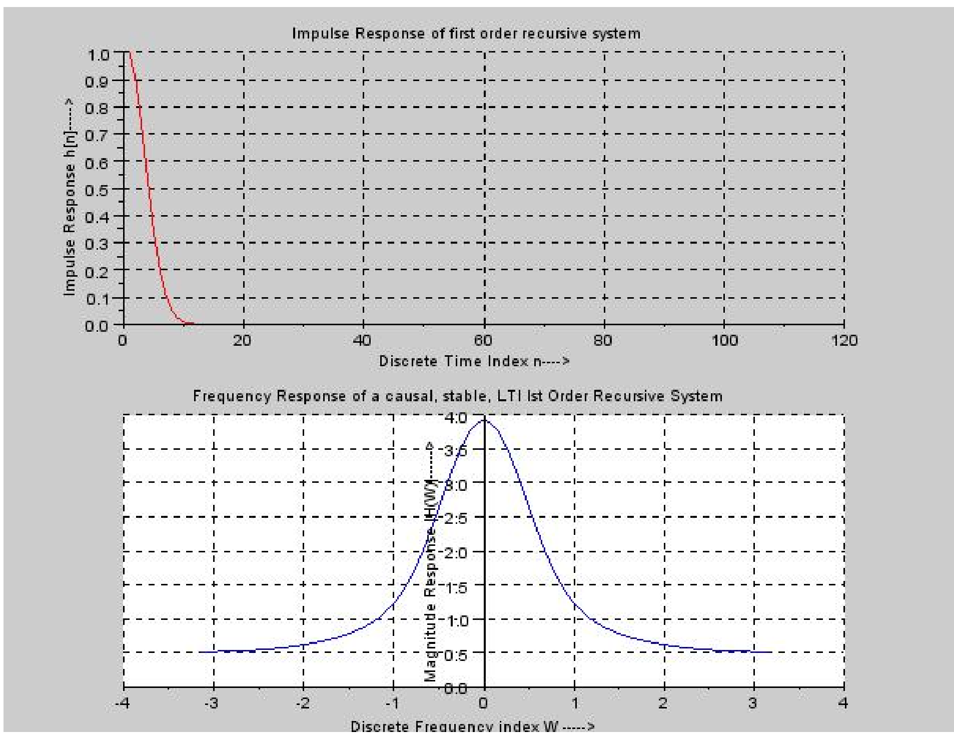

Figure 2.1: Program to find impulse response and Frequency Response of a system

```
end
[HW,W] = frmag(h,512); //frequency response
figure(1)
subplot(2,1,1)
a = gca();
a.x_location = 'origin';
a.y_location = 'origin';
plot([1:length(h)],h,'r');
xlabel('Discrete Time Index n---->');
ylabel('Impulse Response h[n]------>');
title('Impulse Response of first order recursive system')
xgrid(1)
subplot(2,1,2)
a = gca();
a.x_location = 'origin';
a.y_location = 'origin';
plot([mtlb_fliplr(-2*%pi*W),2*%pi*W(2:$)],[mtlb_fliplr(abs(HW)),abs(HW(2:$))])
xlabel('Discrete Frequency index W------>')
ylabel('Magnitude Response |H(W)|------>')
title('Frequency Response of a causal, stable, LTI Ist Order Recursive System');
xgrid(1)
```

Experiment: 3

Linear Circular convolution of two given sequences

Scilab code Solution 3.1 Program to Compute the Convolution of Two Sequences

```
//Caption: Program to Compute the Convolution of Two
    Sequences
clc;
clear;
close;
x = input('Enter the input Sequence:=');
m = length(x);
lx = input('Enter the lower index of input sequence
   :=')
hx = lx+m-1;
n = lx:1:hx;
h = input('Enter impulse response sequence:=')
l = length(h);
lh = input('Enter the lower index of impulse
   response:=')
hh = lh+l-1;
```

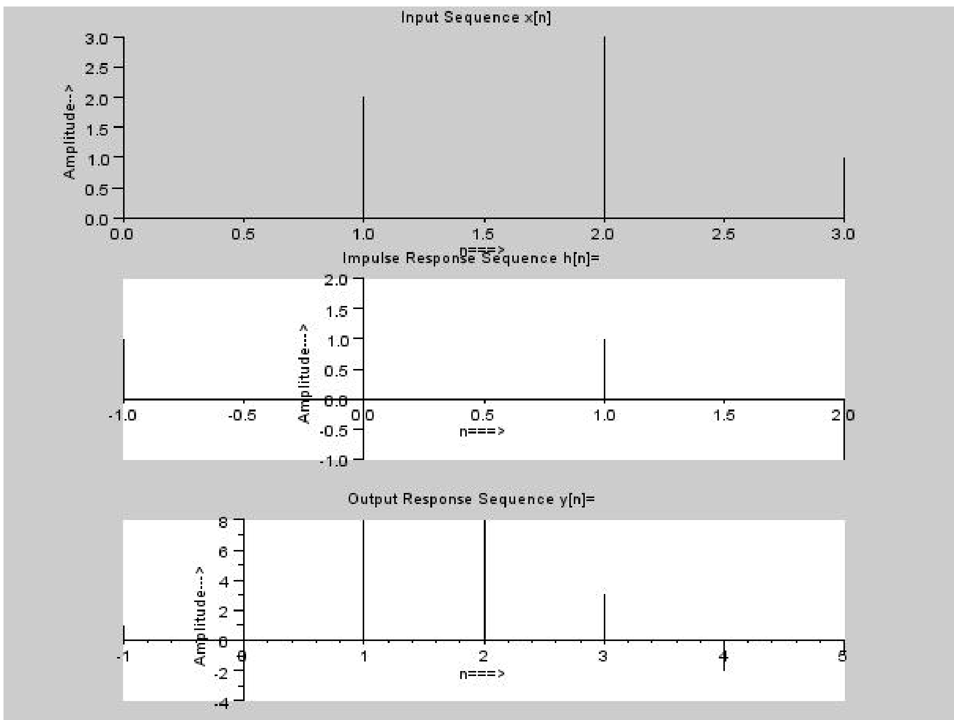

Figure 3.1: Program to Compute the Convolution of Two Sequences

```
g = lh:1:hh;
nx = lx+lh;
nh = nx+m+l-2;
y = convol(x,h)
r = nx:nh;
figure(1)
subplot(3,1,1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude-->')
title('Input Sequence x[n]')
subplot(3,1,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',g,h)
xlabel('n===>')
ylabel('Amplitude--->')
title('Impulse Response Sequence h[n]=')
subplot(3,1,3)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',r,y)
xlabel('n===>')
ylabel('Amplitude--->')
title('Output Response Sequence y[n]=')
//Example
//Enter the input Sequence:=[1,2,3,1]
//
//Enter the lower index of input sequence:=0
//
//Enter impulse response sequence:=[1,2,1,-1]
//
//Enter the lower index of impulse response:=-1
```

```
//
//
//-->y
// y  =
//
//    1.    4.    8.    8.    3.  - 2.  - 1.
//
```

Experiment: 4

Autocorrelation of a given sequence and verification of its properties.

Scilab code Solution 4.1 Program to Compute the Autocorrelation of a Sequence And verfication of Autocorrelation property

```
//Caption: Program to Compute the Autocorrelation of
    a Sequence
//And verfication of Autocorrelation property
clc;
clear;
close;
x = input('Enter the input Sequence:=');
m = length(x);
lx = input('Enter the lower index of input sequence
   :=')
hx = lx+m-1;
n = lx:1:hx;
x_fold = x($:-1:1);
nx = lx+lx;
nh = nx+m+m-2;
r = nx:nh;
```

```
Rxx = convol(x,x_fold);
disp(Rxx,'Auto Correlation Rxx[n]:=')
//Property 1: Autocorrelation of a sequence has even symmetry
//Rxx[n] = Rxx[-n]
Rxx_flip = Rxx([$:-1:1]);
if Rxx_flip==Rxx then
    disp('Property 1:Auto Correlation has Even Symmetry');
    disp(Rxx_flip,'Auto Correlation time reversed Rxx[-n]:=');
end
//Property 2: Center value Rxx[0]= total power of the sequence
Tot_Px = sum(x.^2);
Mid = ceil(length(Rxx)/2);
if Tot_Px == Rxx(Mid) then
    disp('Property 2:Rxx[0]=center value=max. value= Total power of i/p sequence');
end
subplot(2,1,1)
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude--->')
title('Input Sequence x[n]')
subplot(2,1,2)
plot2d3('gnn',r,Rxx)
xlabel('n===>')
ylabel('Amplitude--->')
title('Auto correlation Sequence Rxx[n]')
//Example
//Enter the input Sequence:=[2,-1,3,4,1]
//
//Enter the lower index of input sequence:=-2
//
// Auto Correlation Rxx[n]:=
//
//     2.     7.     5.     11.     31.     11.     5.
```

```
    7.      2.
//
// Property 1:Auto Correlation has Even Symmetry
//
// Auto Correlation time reversed Rxx[-n]:=
//
//      2.      7.      5.      11.      31.      11.      5.
    7.      2.
//
// Property 2:Rxx[0]=center value=max. value=Total
    power of i/p sequence
```

Experiment: 5

Cross correlation of given sequences and verification of its properties.

Scilab code Solution 5.1 Program to Compute the Crosscorrelation of a Sequence And verfication of crosscorrelation property

```
//Caption: Program to Compute the Crosscorrelation
    of a Sequence
//And verfication of crosscorrelation property
clc;
clear;
close;
x = input('Enter the First input Sequence:=');
y = input('Enter the second input Sequence:=')
mx = length(x);
my = length(y);
lx = input('Enter the lower index of first input
    sequence:=')
ly = input('Enter the lower index of second input
    sequence:=')
hx = lx+mx-1;
n = lx:1:hx;
```

```
x_fold = x($:-1:1);
y_fold = y($:-1:1);
nx = lx+ly;
ny = nx+mx+my-2;
r = nx:ny;
Rxy = convol(x,y_fold);
Ryx = convol(x_fold,y);
disp(Rxy,'Cross Correlation Rxy[n]:=')
count =1;
//Property 1: crosscorrelation of a sequence has Antisymmetry
//Rxy[n] = Ryx[-n]
Ryx_flip = Ryx([$:-1:1]);
for i = 1:length(Rxy)
    if (ceil(Ryx_flip(i))==ceil(Rxy(i))) then
       count = count+1;
    end
end
if (count==length(Rxy)) then
    disp('Property 1:Cross Correlation has AntiSymmetry: Rxy[n]=Ryx[-n]');
end
//Property 2:% Verification of Energy Property of Rxy
Ex = sum(x.^2);
Ey = sum(y.^2);
E = sqrt(Ex*Ey);
Mid = ceil(length(Rxy)/2);
if (E >= Rxy(Mid)) then
    disp('Property 2:Energy Property of Cross Correlation verified')
end
subplot(2,1,1)
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude-->')
title('Input Sequence x[n]')
subplot(2,1,2)
```

```
plot2d3('gnn',r,Rxy)
xlabel('n===>')
ylabel('Amplitude-->')
title('Cross correlation Sequence Rxy[n]')
//Example
//Enter the First input Sequence:=[1,2,1,1]
//Enter the second input Sequence:=[1,1,2,1]
//Enter the lower index of first input sequence:=0
//Enter the lower index of second input sequence:=0
//Cross Correlation Rxy[n]:=
//     1.    4.    6.    6.    5.    2.    1.
//Property 1:Cross Correlation has AntiSymmetry: Rxy
    [n]=Ryx[-n]
//
// Property 2:Energy Property of Cross Correlation
    verified
```

Experiment: 6

Solving a given difference equation.

Scilab code Solution 6.1 Solving Difference Equation Direct Form II Realization

```
//Caption: Solving Difference Equation
//Direct Form-II Realization
//Finding out the Output Response of the first order
//system(Filter)
clc;
clear;
close;
x =
   [1,1/2,1/4,1/8,1/16,1/32,1/64,1/128,1/256,1/512];
b = [3,-4/3]; //numerator polynomials
a = [1,-1/3]; //denominator polynomials
p = length(a)-1;
q = length(b)-1;
pq = max(p,q);
a = a(2:p+1);
w = zeros(1,pq);
for i = 1:length(x)
  wnew = x(i)-sum(w(1:p).*a);
```

```
  w = [wnew,w];
  y(i) = sum(w(1:q+1).*b);
end
disp(y,'Output Response y[n]=');
//Result
//Output Response y[n]=
//      3.
//      1.1666667
//      0.4722222
//      0.1990741
//      0.0871914
//      0.0394805
//      0.0183685
//      0.0087270
//      0.0042111
//      0.0020547
```

Experiment: 7

Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum.

Scilab code Solution 7.1 Program to find the spectral information of discrete time signal Calculation of DFT and IDFT

```
//Caption: Program to find the spectral information
   of discrete time  signal
//Calculation of DFT and IDFT
//Plotting Magnitude and Phase Spectrum
clc;
close;
clear;
xn = input('Enter the real input discrete sequence x
   [n]=');
N = length(xn);
XK = zeros(1,N);
IXK = zeros(1,N);
//Code block to find the DFT of the Sequence
for K = 0:N-1
```

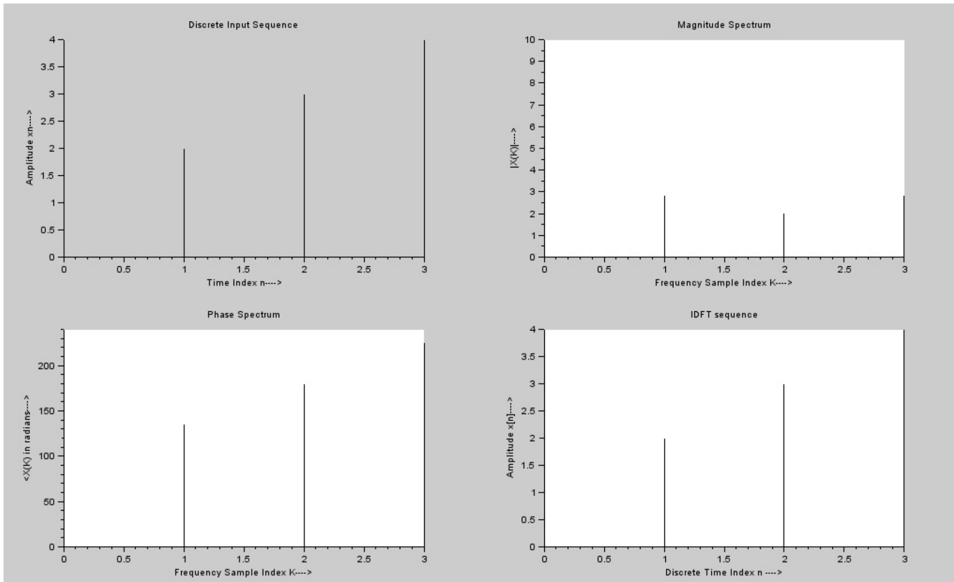

Figure 7.1: Program to find the spectral information of discrete time signal Calculation of DFT and IDFT

```
    for n = 0:N-1
        XK(K+1) = XK(K+1)+xn(n+1)*exp(-%i*2*%pi*K*n/
          N);
    end
end
[phase,db] = phasemag(XK)
disp(XK,'Discrete Fourier Transform X(k)=')
disp(abs(XK),'Magnitude Spectral Samples=')
disp(phase,'Phase Spectral Samples=')
n = 0:N-1;
K = 0:N-1;
figure(1)
subplot(2,2,1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,xn)
xlabel('Time Index n---->')
```

```
ylabel('Amplitude xn------>')
title('Discrete Input Sequence')
subplot(2,2,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',K,abs(XK))
xlabel('Frequency Sample Index K------>')
ylabel('|X(K)|------>')
title('Magnitude Spectrum')
subplot(2,2,3)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',K,phase)
xlabel('Frequency Sample Index K------>')
ylabel('<X(K) in radians------>')
title('Phase Spectrum')
//Code block to find the IDFT of the sequence
for n = 0:N-1
    for K = 0:N-1
        IXK(n+1) = IXK(n+1)+XK(K+1)*exp(%i*2*%pi*K*n
          /N);
    end
end
IXK = IXK/N;
ixn = real(IXK);
subplot(2,2,4)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',[0:N-1],ixn)
xlabel('Discrete Time Index n ------>')
ylabel('Amplitude x[n]------>')
title('IDFT sequence')
//Example
//
//Enter the real input discrete sequence x[n
```

```
   ]=[1,2,3,4]
//
// Discrete Fourier Transform X(k)=
//
//     10.  - 2. + 2.i   - 2. - 9.797D-16i   - 2. - 2.i
//
// Magnitude Spectral Samples=
//
//     10.     2.8284271     2.     2.8284271
//
// Phase Spectral Samples=
//
//     0.     135.     180.     225.
//
```

Experiment: 8

Linear convolution of two sequences using DFT and IDFT.

Scilab code Solution 8.1 Linear Convolution using Circular Convolution DFT IDFT method

```
//Caption: Linear Convolution using Circular
    Convolution
//DFT–IDFT method
clc;
clear;
close;
x = input('Enter the input discrete sequence:=')
h = input('Enter the impulse discrete sequence:=')
N1 = length(x);
N2 = length(h);
N = N1+N2-1;//Linear Convolution result length
h = [h,zeros(1,N-N2)];
x = [x,zeros(1,N-N1)];
//Computing DFT–IDFT
```

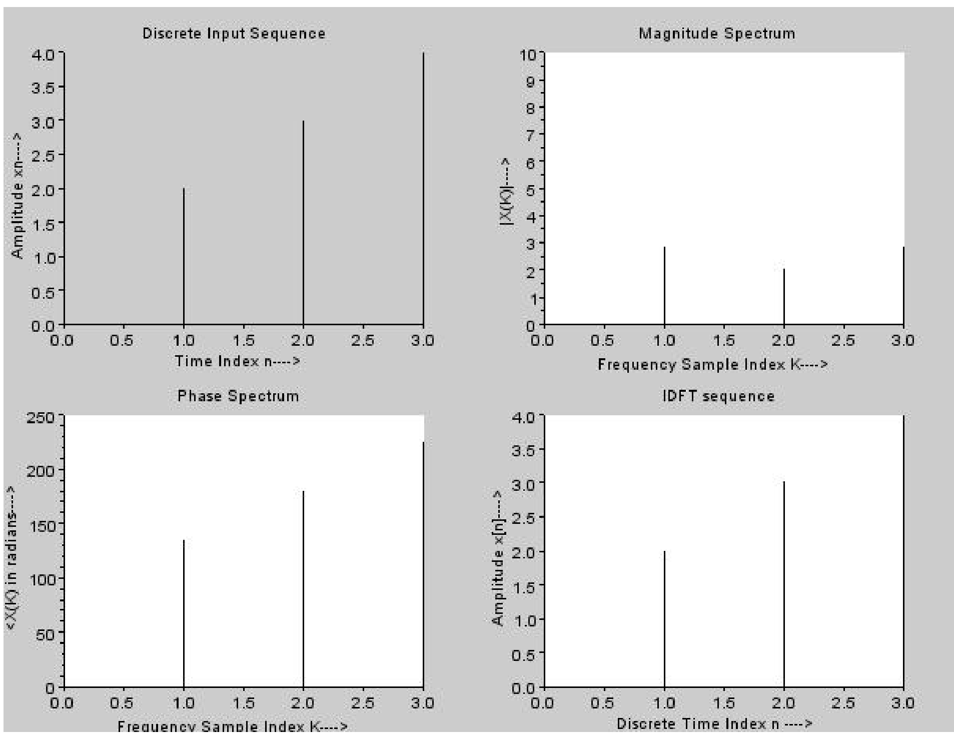

Figure 8.1: Linear Convolution using Circular Convolution DFT IDFT method

```
XK = dft(x,-1);//N point DFT of i/p sequence
HK = dft(h,-1);//N point DFT of impulse sequence
//Multiplication of 2 DFT's
YK = XK.*HK;
//Linear Convolution result
yn = dft(YK,1);//IDFT of Y(K)(o/p sequence)
disp(real(yn),'Linear Convolution result y[n]:=')
//Example
//Enter the input discrete sequence:= [1,2,3]
//Enter the impulse discrete sequence:=[1,2,2,1]
//Linear Convolution result y[n]:=
//
//     1.
//     4.
//     9.
//     11.
//     8.
//     3.
```

Experiment: 9

Circular convolution of two given sequences using DFT and IDFT

Scilab code Solution 9.1 Circular Convolution using DFT IDFT method

```
//Caption: Circular Convolution using DFT-IDFT
   method
clc;
clear;
close;
L = 4; // Length of the sequence
N = 4; //N-point DFT
x1 = input('Enter the first discrete sequence:x1[n]=
   ')
x2 = input('Enter the second discrete sequence:x2[n
   ]=')
//Computing DFT
X1K = dft(x1,-1);
X2K = dft(x2,-1);
//Multiplication of 2 DFT's
X3K = X1K.*X2K;
x3 = dft(X3K,1); //IDFT of X3(K)
```

```
x3 = real(x3);
disp(x3,'Circular Convolution result:x3[n]=');
//Example
//Enter the first discrete sequence:x1[n]= [2,1,2,1]
//Enter the second discrete sequence:x2[n]=
    [1,2,3,4]
//
// Circular Convolution result:x3[n]=
//
//     14.
//     16.
//     14.
//     16.
```

Experiment: 10

Design and implementation of FIR filter to meet given specifications.

Scilab code Solution 10.1 To Design an Low Pass FIR Filter

```
//Caption: To Design an Low Pass FIR Filter
clc;
clear;
close;
wp= input('Enter the pass band edge (rad)= ');
ws= input('Enter the stop band edge (rad)= ');
ks= input('Enter the stop band attenuation (dB)= ');
//If 43<Ks<54 choose hamming window.
//To select N,order of filter.
N= (2*%pi*4)./(ws-wp); // k=4 for Hamming window.
N= ceil(N); //To round-off N to the next integer.
wc=(wp+(ws-wp)/2)./%pi
// To obtain FIR filter Impulse Response 'wft'
//And FIR Filter Frequency response 'wfm'
[wft,wfm,fr]=wfir('lp',N+1,[wc/2,0],'hm',[0,0]);
```

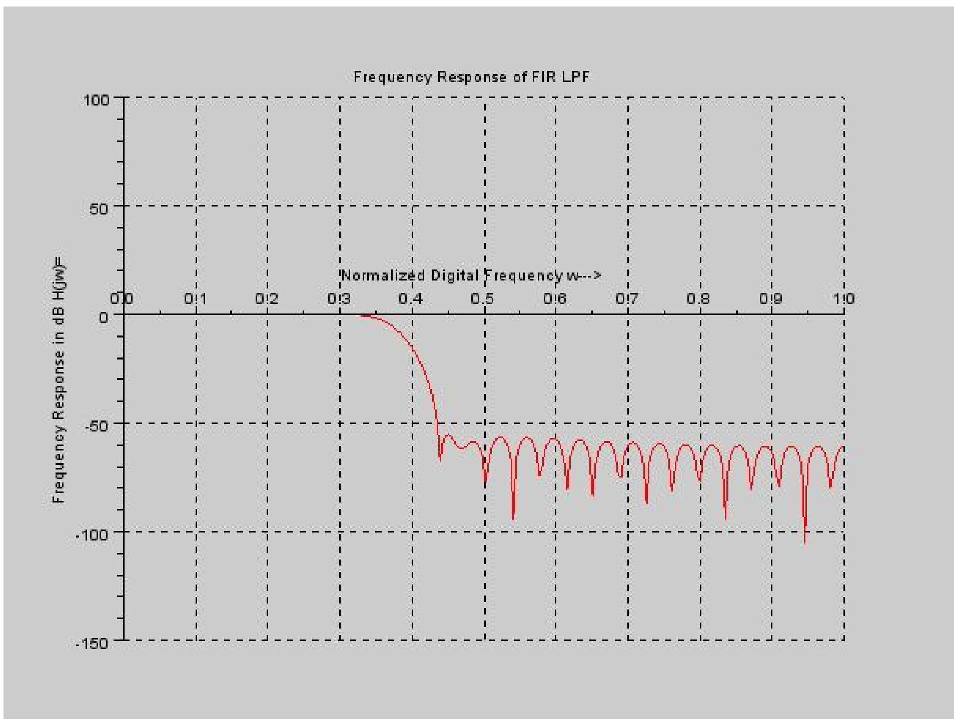

Figure 10.1: To Design an Low Pass FIR Filter

```
figure(1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-150;1,50];
plot(2*fr,20*log10(wfm),'r')
xlabel('Normalized Digital Frequency w--->')
ylabel('Frequency Response in dB H(jw)=')
title('Frequency Response of FIR LPF')
xgrid(1)
//Result
//Enter the pass band edge (rad)= 0.3*%pi
//Enter the stop band edge (rad)=  0.45*%pi
//Enter the stop band attenuation (dB)=  50
//N  =    54.
//-->wc
// wc  =    0.375
//-->disp(wft,'Impulse Response of FIR LPF=')
// Impulse Response of FIR LPF=
//    column 1 to 7
//    0.0003609  - 0.0007195  - 0.0010869    1.575D-18    0.0016485    0.0015927  - 0.0010883
//    column  8 to 14
//  - 0.0035703  - 0.0017009    0.0038764    0.0061896  - 5.965D-18  - 0.0090208  - 0.0082516
//    column 15 to 21
//     0.0053105    0.0164428    0.0074408  - 0.0162551  - 0.0251602    1.191D-17    0.0359480
//    column 22 to 28
//    0.0334760  - 0.0225187  - 0.0756838  - 0.0394776    0.1111441    0.2931653    0.375
//    column 29 to 35
//    0.2931653    0.1111441  - 0.0394776  - 0.0756838  - 0.0225187    0.0334760    0.0359480
//    column 36 to 42
//    1.191D-17  - 0.0251602  - 0.0162551    0.0074408    0.0164428    0.0053105  - 0.0082516
//    column 43 to 49
```

```
//    - 0.0090208   - 5.965D-18      0.0061896
    0.0038764   - 0.0017009   - 0.0035703   - 0.0010883
      column 50 to 55
//     0.0015927      0.0016485      1.575D-18   -
    0.0010869   - 0.0007195      0.0003609
```

Experiment: 11

Design and implementation of IIR filter to meet given specifications.

Scilab code Solution 11.1 To obtain Digital IIR Butterworth low pass filter Frequency response

```
//Caption: To obtain Digital IIR Butterworth low
    pass filter
//Frequency response
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/sec = ')
    ;
```

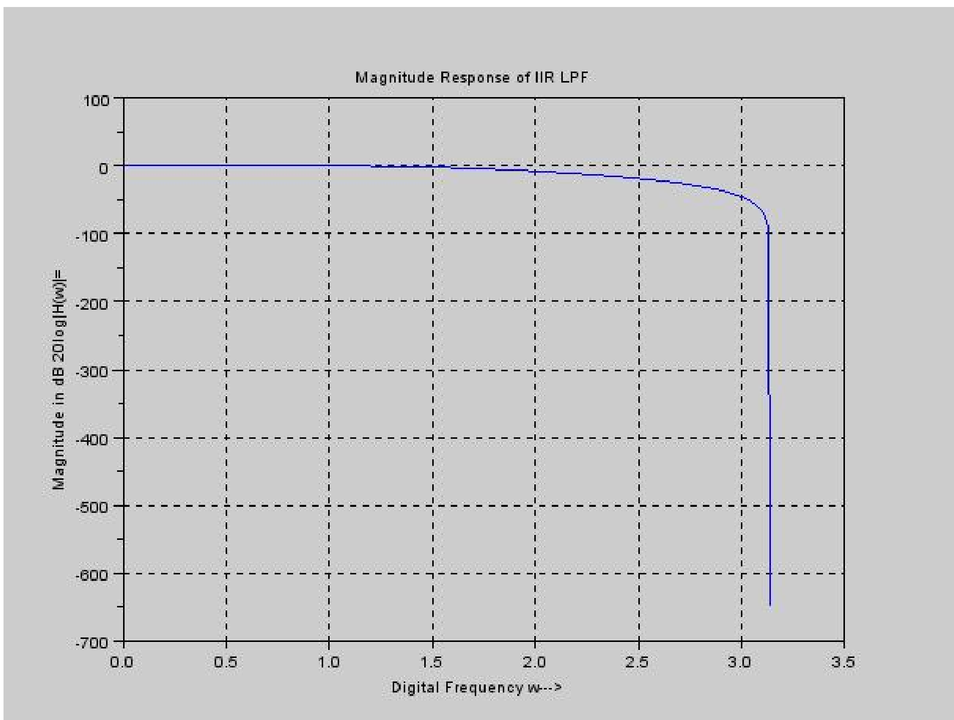

Figure 11.1: To obtain Digital IIR Butterworth low pass filter Frequency response

```
d1 = 10^(kp/20);
d2 = 10^(ks/20);
d = sqrt((1/(d2^2))-1);
E = sqrt((1/(d1^2))-1);
//Digital filter specifications(rad/samples)
wp=2*%pi*fp*1/Fs;
ws=2*%pi*fs*1/Fs;
disp(wp,'Digital Pass band edge freq in rad/samples wp=')
disp(ws,'Digital Stop band edge freq in rad/samples ws=')
//Pre warping
op=2*Fs*tan(wp/2);
os=2*Fs*tan(ws/2);
disp(op,'Analog Pass Band Edge Freq. in rad/sec op=')
disp(os,'Analog Stop band Edge Freq. in rad/sec os=')
N = log10(d/E)/log10(os/op);
oc = op/((E^2)^(1/(2*N)));
N = ceil(N);//rounded to nearest integer
disp(N,'IIR Filter order N =');
disp(oc,'Cutoff Frequency in rad/seconds OC = ')
[pols,gn] = zpbutt(N,oc);
disp(gn,'Gain of Analog IIR Butterworth LPF Gain =')
disp(pols,'Poles of Analog IIR Butterworth LPF Poles =')
HS = poly(gn,'s','coeff')/real(poly(pols,'s'));
disp(HS,'Transfer function of Ananlog IIR Butterworth LPF H(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 Digitla IIR Butterworth LPF H(Z)=')
```

```
[Hw,w] = frmag(Hz,256);
figure(1)
plot(2*w*%pi,20*log10(abs(Hw)));
xlabel('Digital Frequency w——>')
ylabel('Magnitude in dB 20log|H(w)|=')
title('Magnitude Response of IIR LPF')
xgrid(1)
//Result
//Enter the pass band edge (Hz) =  1500
//
//Enter the stop band edge (Hz) =  2000
//
//Enter the pass band attenuation (dB) =  -1
//
//Enter the stop band attenuation (dB) =  -3
//
//Enter the sampling rate  samples/sec =  8000
//
// Digital Pass band edge freq in rad/samples wp=
//
//     1.1780972
//
// Digital Stop band edge freq in rad/samples ws=
//
//     1.5707963
//
// Analog Pass Band Edge Freq. in rad/sec op=
//
//     10690.858
//
// Analog Stop band Edge Freq. in rad/sec os=
//
//     16000.
//
// IIR Filter order N =
//
//     2.
//
```

```
// Cutoff Frequency in rad/seconds OC =
//
//     16022.769
//
// Gain of Analog IIR Butterworth LPF Gain =
//
//     2.567D+08
//
// Poles of Analog IIR Butterworth LPF Poles =
//
//  - 11329.809 + 11329.809i   - 11329.809 -
    11329.809i
//
// Transfer function of Ananlog IIR Butterworth LPF
    H(S)=
//
//            2.567D+08
//     ---------------------------
//                                 2
//     2.567D+08 + 22659.618s + s
//
// Transfer function of Digitla IIR Butterworth LPF
    H(Z)=
//
//                                             2
//     0.2933099 + 0.5866197z + 0.2933099z
//     -----------------------------------------
//                                       2
//         0.1715734 + 0.0016661z + z
//
```

Scilab code Solution 11.2 To obtain Digital IIR Chebyshev low pass filter Frequency response

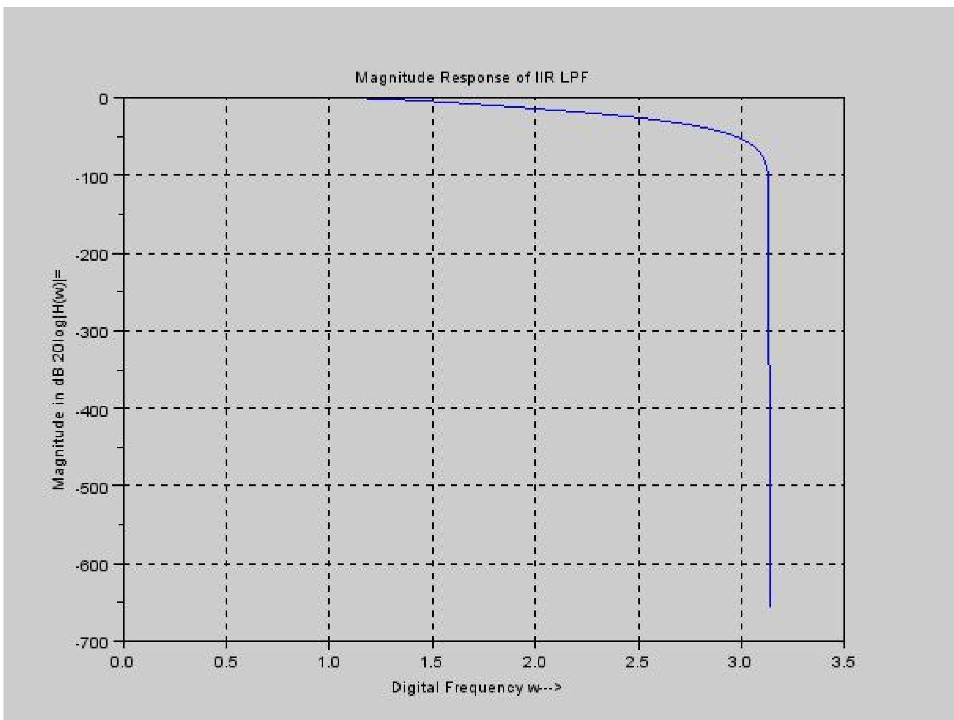

Figure 11.2: To obtain Digital IIR Chebyshev low pass filter Frequency response

```
//Caption: To obtain Digital IIR Chebyshev low pass filter
//Frequency response
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/sec = ');
d1 = 10^(kp/20);
d2 = 10^(ks/20);
d = sqrt((1/(d2^2))-1);
E = sqrt((1/(d1^2))-1);
//Digital filter specifications(rad/samples)
wp=2*%pi*fp*1/Fs;
ws=2*%pi*fs*1/Fs;
disp(wp,'Digital Pass band edge freq in rad/samples wp=')
disp(ws,'Digital Stop band edge freq in rad/samples ws=')
//Pre warping
op=2*Fs*tan(wp/2);
os=2*Fs*tan(ws/2);
disp(op,'Analog Pass Band Edge Freq. in rad/sec op=')
disp(os,'Analog Stop band Edge Freq. in rad/sec os=')
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,'Cutoff Frequency in rad/seconds OC = ')
[pols,gn] = zpch1(N,E,op);
```

```
disp(gn,'Gain of Analog IIR Chebyshev Type-I LPF Gain =')
disp(pols,'Poles of Analog IIR Chebyshev Type-I LPF Poles =')
HS = poly(gn,'s','coeff')/real(poly(pols,'s'));
disp(HS,'Transfer function of Ananlog IIR Chebyshev Type-I LPF H(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 Digitla IIR Chebyshev LPF H(Z)=')
[Hw,w] = frmag(Hz,256);
figure(1)
plot(2*w*%pi,20*log10(abs(Hw)));
xlabel('Digital Frequency w--->')
ylabel('Magnitude in dB 20log|H(w)|=')
title('Magnitude Response of IIR LPF')
xgrid(1)
//Result
//Enter the pass band edge (Hz) = 1500
//
//Enter the stop band edge (Hz) = 2000
//
//Enter the pass band attenuation (dB) = -1
//
//Enter the stop band attenuation (dB) = -3
//
//Enter the sampling rate samples/sec = 8000
//
// Digital Pass band edge freq in rad/samples wp=
//
//    1.1780972
//
// Digital Stop band edge freq in rad/samples ws=
```

```
//
//     1.5707963
//
// Analog Pass Band Edge Freq. in rad/sec op=
//
//     10690.858
//
// Analog Stop band Edge Freq. in rad/sec os=
//
//     16000.
//
// IIR Filter order N =
//
//     2.
//
//Cutoff Frequency in rad/seconds OC =
//
//     17642.912
//
// Gain of Analog IIR Chebyshev Type-I LPF Gain =
//
//     1.123D+08
//
// Poles of Analog IIR Chebyshev Type-I LPF Poles =
//
//  - 5867.861 + 9569.6927i  - 5867.861 - 9569.6927i
//
// Transfer function of Ananlog IIR Chebyshev Type-I LPF H(S)=
//
//              1.123D+08
//     ----------------------------
//                                 2
//     1.260D+08 + 11735.722s + s
//
// Transfer function of Digitla IIR Chebyshev LPF H(Z)=
//
```

```
//                                                  2
//       0.1971055 + 0.3942111z + 0.1971055z
//       ------------------------------------
//                                              2
//            0.3409008 - 0.4562766z + z
```

Experiment: 12

Circular convolution of two given sequences

Scilab code Solution 12.1 Program to perform circular convolution of two sequences

```
//Caption: Program to perform circular convolution
   of two sequences
clc;
clear;
close;
x1 = input('Enter the first discrete sequence:=')
x2 = input('Enter the second discrete 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);//circular convolution result initialized to zero
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(x3,'Circular Convolution Result x3[n]=')
//Example
//Enter the first discrete sequence:= [2,1,2,1]
//Enter the second discrete sequence:= [1,2,3,4]
//Circular Convolution Result x3[n]=
//    14.    16.    14.    16.
//
```