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

April 6, 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 Experiments

List of Figures

Experiment: 1

Verification of Sampling theorem.

Scilab code Solution 1.1 Verification of Sampling Theorem

```
1  //Caption: Verification of Sampling Theorem
2  //[1].Right Sampling [2]. Under Sampling [3]. Over
       Sampling
3  clc;
4  close;
5  clear;
6  fm=input('Enter the input signal frequency:');
7  k=input('Enter the number of Cycles of input signal:
       ');
8  A=input('Enter the amplitude of input signal:');
9  tm=0:1/(fm*fm):k/fm;
10 x=A*cos(2*%pi*fm*tm);
11 figure(1);
12 a = gca();
13 a.x_location = "origin";
14 a.y_location = "origin";
15 plot(tm,x);
16 title('ORIGINAL SIGNAL');
17 xlabel('Time');
```

```
18 ylabel('Amplitude');
19 xgrid(1)
20 //Sampling Rate(Nyquist Rate)=2*fm
21 fnyq=2*fm;
22 // UNDER SAMPLING
23 fs=(3/4)*fnyq;
24 n=0:1/fs:k/fm;
25 xn=A*cos(2*%pi*fm*n);
26 figure(2);
27 a = gca();
28 a.x_location = "origin";
29 a.y_location = "origin";
30 plot2d3('gnn',n,xn);
31 plot(n,xn,'r');
32 title('Under Sampling');
33 xlabel('Time');
34 ylabel('Amplitude');
35 legend('Sampled Signal', 'Reconstructed Signal');
36 xgrid(1)
37 //NYQUIST SAMPLING
38 fs=fnyq;
39 n=0:1/fs:k/fm;
40 xn=A*cos(2*%pi*fm*n);
41 figure(3);
42 a = gca();
43 a.x_location = "origin";
44 a.y_location = "origin";
45 plot2d3('gnn',n,xn);
46 plot(n,xn,'r');
47 title('Nyquist Sampling');
48 xlabel('Time');
49 ylabel('Amplitude');
50 legend('Sampled Signal', 'Reconstructed Signal');
51 xgrid(1)
52 //OVER SAMPLING
53 fs=fnyq*10;
54 n=0:1/fs:k/fm;
55 xn=A*cos(2*%pi*fm*n);
```

```
56 figure(4);
57 a = gca();
58 a.x_location = "origin";
59 a.y_location = "origin";
60 plot2d3('gnn',n,xn);
61 plot(n,xn,'r');
62 title('Over Sampling');
63 xlabel('Time');
64 ylabel('Amplitude');
65 legend('Sampled Signal', 'Reconstructed Signal');
66 xgrid(1)
67 //Result
68 //Enter the input signal frequency:100
69 //
70 //Enter the number of Cycles of input signal:2
71 //
72 //Enter the amplitude of input signal:2
```

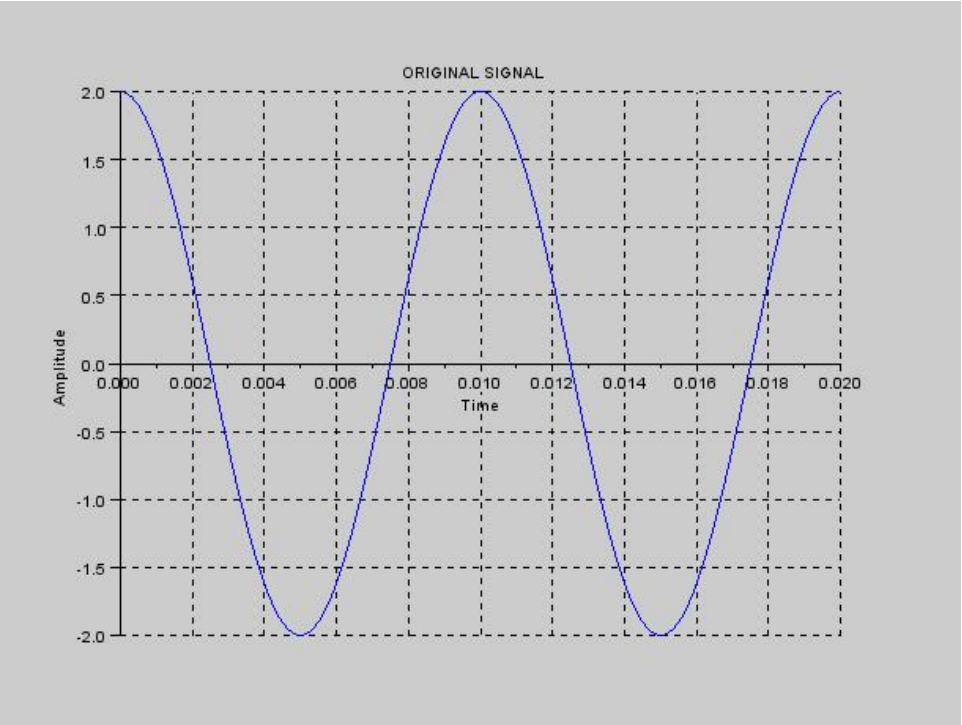

Figure 1.1: Verification of Sampling Theorem

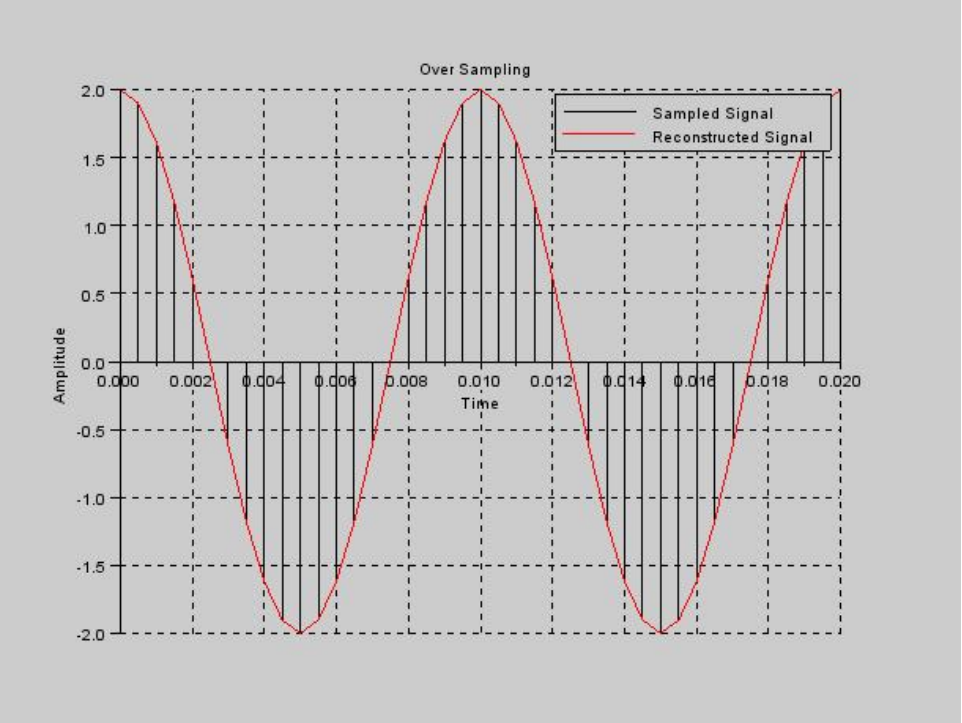

Figure 1.2: Verification of Sampling Theorem

Experiment: 2

Impulse response of a given system

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

```
1  //Caption: Program to find impulse response and
2  //Frequency Response of a system
3  //y[n] = a*y[n-1]+x[n]
4  //Assume y[n] = h[n], x[n]=delta[n]=unit impulse
       response
5  //a = 0.9
6  //h[n] = 0.9*h[n-1]+delta[n]
7  clc;
8  clear;
9  close;
10 a = 0.9;//constant a = 0.9 less than 1
11 h0 = 1;
12 h1 = a; //first two values of impulse response
13 h = [h0,h1,zeros(1,100)];
14 for i = 1:100
15   h(i+2) = ((a)^(i+1))*h(i+1);//impulse response
```

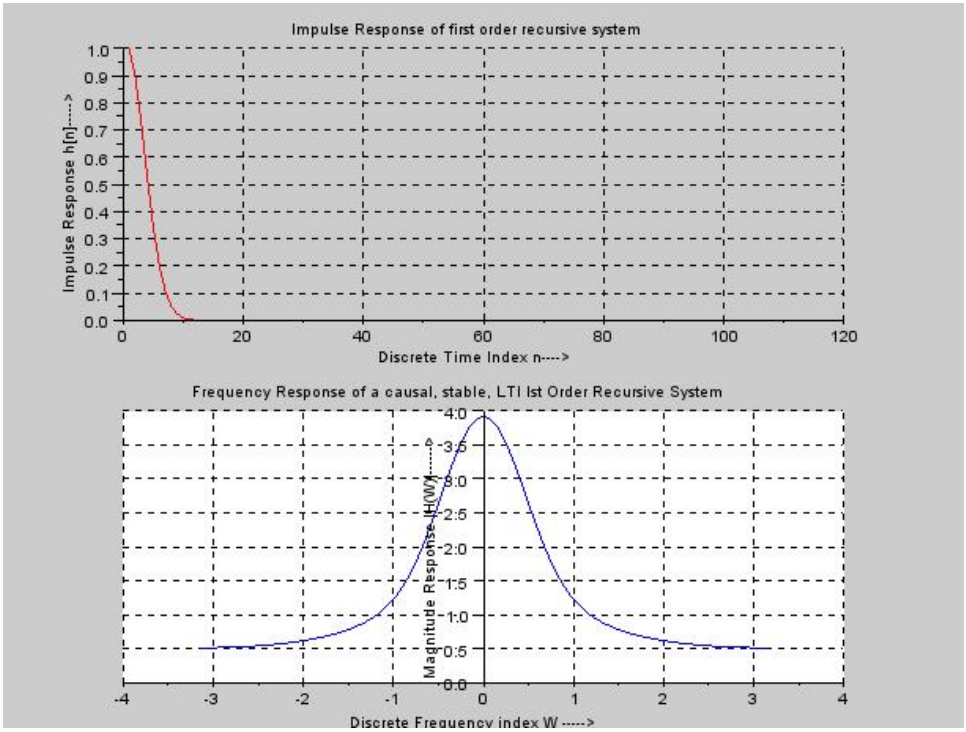

Figure 2.1: Program to find impulse response and Frequency Response of a system

```
16 end
17 [HW,W] = frmag(h,512); //frequency response
18 figure(1)
19 subplot(2,1,1)
20 a = gca();
21 a.x_location = 'origin';
22 a.y_location = 'origin';
23 plot([1:length(h)],h,'r');
24 xlabel('Discrete Time Index n------>');
25 ylabel('Impulse Response h[n]------>');
26 title('Impulse Response of first order recursive
      system')
27 xgrid(1)
28 subplot(2,1,2)
29 a = gca();
30 a.x_location = 'origin';
31 a.y_location = 'origin';
32 plot([mtlb_fliplr(-2*%pi*W),2*%pi*W(2:$)],[
      mtlb_fliplr(abs(HW)),abs(HW(2:$))])
33 xlabel('Discrete Frequency index W------>')
34 ylabel('Magnitude Response |H(W)|------>')
35 title('Frequency Response of a causal, stable, LTI
      Ist Order Recursive System');
36 xgrid(1)
```

Experiment: 3

Linear Circular convolution of two given sequences

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

```
1  //Caption: Program to Compute the Convolution of Two
       Sequences
2  clc;
3  clear;
4  close;
5  x = input('Enter the input Sequence:=');
6  m = length(x);
7  lx = input('Enter the lower index of input sequence
       :=')
8  hx = lx+m-1;
9  n = lx:1:hx;
10 h = input('Enter impulse response sequence:=')
11 l = length(h);
12 lh = input('Enter the lower index of impulse
       response:=')
13 hh = lh+l-1;
```

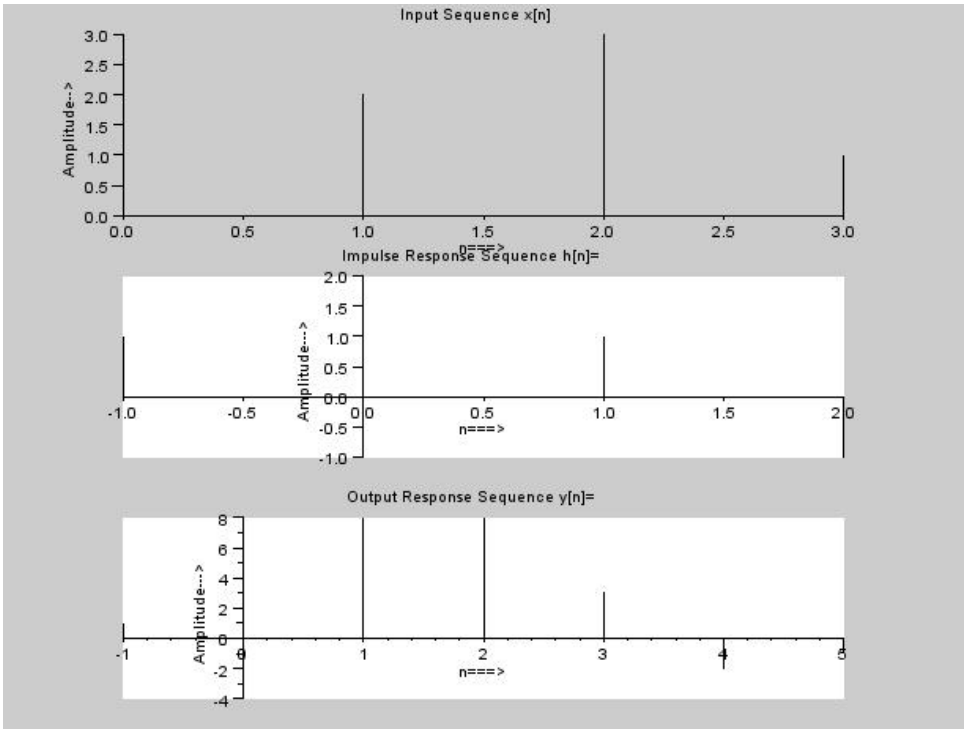

Figure 3.1: Program to Compute the Convolution of Two Sequences

```
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
```

```
52 //
53 //
54 //-->y
55 //  y   =
56 //
57 //      1.      4.      8.      8.      3.   -  2.   -  1.
58 //
```

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

```
1  //Caption: Program to Compute the Autocorrelation of
       a Sequence
2  //And verfication of Autocorrelation property
3  clc;
4  clear;
5  close;
6  x = input('Enter the input Sequence:=');
7  m = length(x);
8  lx = input('Enter the lower index of input sequence
       :=')
9  hx = lx+m-1;
10 n = lx:1:hx;
11 x_fold = x($:-1:1);
12 nx = lx+lx;
13 nh = nx+m+m-2;
14 r = nx:nh;
```

```
15 Rxx = convol(x,x_fold);
16 disp(Rxx,'Auto Correlation Rxx[n]:=')
17 //Property 1: Autocorrelation of a sequence has even
       symmetry
18 //Rxx[n] = Rxx[-n]
19 Rxx_flip = Rxx([$:-1:1]);
20 if Rxx_flip==Rxx then
21     disp('Property 1:Auto Correlation has Even
          Symmetry');
22     disp(Rxx_flip,'Auto Correlation time reversed
          Rxx[-n]:=');
23 end
24 //Property 2: Center value Rxx[0]= total power of
     the sequence
25 Tot_Px = sum(x.^2);
26 Mid = ceil(length(Rxx)/2);
27 if Tot_Px == Rxx(Mid) then
28     disp('Property 2:Rxx[0]=center value=max. value=
          Total power of i/p sequence');
29 end
30 subplot(2,1,1)
31 plot2d3('gnn',n,x)
32 xlabel('n===>')
33 ylabel('Amplitude-->')
34 title('Input Sequence x[n]')
35 subplot(2,1,2)
36 plot2d3('gnn',r,Rxx)
37 xlabel('n===>')
38 ylabel('Amplitude-->')
39 title('Auto correlation Sequence Rxx[n]')
40 //Example
41 //Enter the input Sequence:=[2,-1,3,4,1]
42 //
43 //Enter the lower index of input sequence:=-2
44 //
45 // Auto Correlation Rxx[n]:=
46 //
47 //    2.    7.    5.    11.    31.    11.    5.
```

```
    7.    2.
48 //
49 // Property 1:Auto Correlation has Even Symmetry
50 //
51 // Auto Correlation time reversed Rxx[-n]:=
52 //
53 //    2.    7.    5.    11.    31.    11.    5.
    7.    2.
54 //
55 // 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

```
1 //Caption: Program to Compute the Crosscorrelation
     of a Sequence
2 //And verfication of crosscorrelation property
3 clc;
4 clear;
5 close;
6 x = input('Enter the First input Sequence:=');
7 y = input('Enter the second input Sequence:=')
8 mx = length(x);
9 my = length(y);
10 lx = input('Enter the lower index of first input
     sequence:=')
11 ly = input('Enter the lower index of second input
     sequence:=')
12 hx = lx+mx-1;
13 n = lx:1:hx;
```

```
14 x_fold = x($:-1:1);
15 y_fold = y($:-1:1);
16 nx = lx+ly;
17 ny = nx+mx+my-2;
18 r = nx:ny;
19 Rxy = convol(x,y_fold);
20 Ryx = convol(x_fold,y);
21 disp(Rxy,'Cross Correlation Rxy[n]:=')
22 count =1;
23 //Property 1: crosscorrelation of a sequence has
      Antisymmetry
24 //Rxy[n] = Ryx[-n]
25 Ryx_flip = Ryx([$:-1:1]);
26 for i = 1:length(Rxy)
27     if (ceil(Ryx_flip(i))==ceil(Rxy(i))) then
28         count = count+1;
29     end
30 end
31 if (count==length(Rxy)) then
32     disp('Property 1:Cross Correlation has
          AntiSymmetry: Rxy[n]=Ryx[-n]');
33 end
34 //Property 2:% Verification of Energy Property of
      Rxy
35 Ex = sum(x.^2);
36 Ey = sum(y.^2);
37 E = sqrt(Ex*Ey);
38 Mid = ceil(length(Rxy)/2);
39 if (E >= Rxy(Mid)) then
40     disp('Property 2:Energy Property of Cross
          Correlation verified')
41 end
42 subplot(2,1,1)
43 plot2d3('gnn',n,x)
44 xlabel('n===>')
45 ylabel('Amplitude-->')
46 title('Input Sequence x[n]')
47 subplot(2,1,2)
```

```
48 plot2d3('gnn',r,Rxy)
49 xlabel('n===>')
50 ylabel('Amplitude-->')
51 title('Cross correlation Sequence Rxy[n]')
52 //Example
53 //Enter the First input Sequence:=[1,2,1,1]
54 //Enter the second input Sequence:=[1,1,2,1]
55 //Enter the lower index of first input sequence:=0
56 //Enter the lower index of second input sequence:=0
57 //Cross Correlation Rxy[n]:=
58 //      1.     4.     6.     6.     5.     2.     1.
59 //Property 1:Cross Correlation has AntiSymmetry: Rxy
      [n]=Ryx[-n]
60 //
61 // 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

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

```
18    w = [wnew,w];
19    y(i) = sum(w(1:q+1).*b);
20  end
21  disp(y,'Output Response y[n]=');
22  //Result
23  //Output Response y[n]=
24  //     3.
25  //     1.1666667
26  //     0.4722222
27  //     0.1990741
28  //     0.0871914
29  //     0.0394805
30  //     0.0183685
31  //     0.0087270
32  //     0.0042111
33  //     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

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

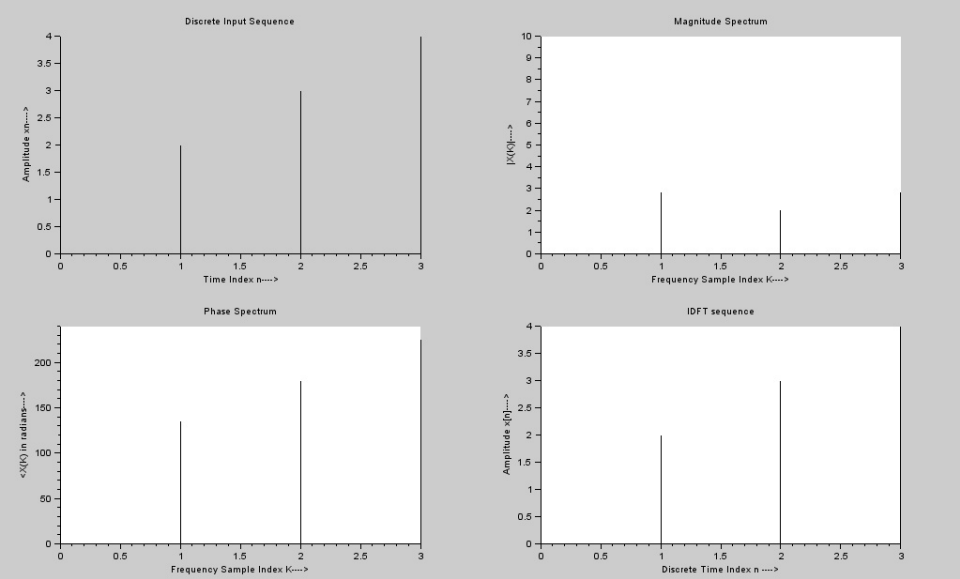

Figure 7.1: Program to find the spectral information of discrete time signal Calculation of DFT and IDFT

```
13      for n = 0:N-1
14          XK(K+1) = XK(K+1)+xn(n+1)*exp(-%i*2*%pi*K*n/
                N);
15      end
16  end
17  [phase,db] = phasemag(XK)
18  disp(XK,'Discrete Fourier Transform X(k)=')
19  disp(abs(XK),'Magnitude Spectral Samples=')
20  disp(phase,'Phase Spectral Samples=')
21  n = 0:N-1;
22  K = 0:N-1;
23  figure(1)
24  subplot(2,2,1)
25  a = gca();
26  a.x_location = "origin";
27  a.y_location = "origin";
28  plot2d3('gnn',n,xn)
29  xlabel('Time Index n------>')
```

```
30 ylabel('Amplitude xn------>')
31 title('Discrete Input Sequence')
32 subplot(2,2,2)
33 a = gca();
34 a.x_location = "origin";
35 a.y_location = "origin";
36 plot2d3('gnn',K,abs(XK))
37 xlabel('Frequency Sample Index K------>')
38 ylabel('|X(K)|------>')
39 title('Magnitude Spectrum')
40 subplot(2,2,3)
41 a = gca();
42 a.x_location = "origin";
43 a.y_location = "origin";
44 plot2d3('gnn',K,phase)
45 xlabel('Frequency Sample Index K------>')
46 ylabel('<X(K) in radians------>')
47 title('Phase Spectrum')
48 //Code block to find the IDFT of the sequence
49 for n = 0:N-1
50     for K = 0:N-1
51         IXK(n+1) = IXK(n+1)+XK(K+1)*exp(%i*2*%pi*K*n
              /N);
52     end
53 end
54 IXK = IXK/N;
55 ixn = real(IXK);
56 subplot(2,2,4)
57 a = gca();
58 a.x_location = "origin";
59 a.y_location = "origin";
60 plot2d3('gnn',[0:N-1],ixn)
61 xlabel('Discrete Time Index n ------>')
62 ylabel('Amplitude x[n]------>')
63 title('IDFT sequence')
64 //Example
65 //
66 //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

```
1 //Caption: Linear Convolution using Circular
     Convolution
2 //DFT-IDFT method
3 clc;
4 clear;
5 close;
6 x = input('Enter the input discrete sequence:=')
7 h = input('Enter the impulse discrete sequence:=')
8 N1 = length(x);
9 N2 = length(h);
10 N = N1+N2-1;//Linear Convolution result length
11 h = [h,zeros(1,N-N2)];
12 x = [x,zeros(1,N-N1)];
13 //Computing DFT-IDFT
```

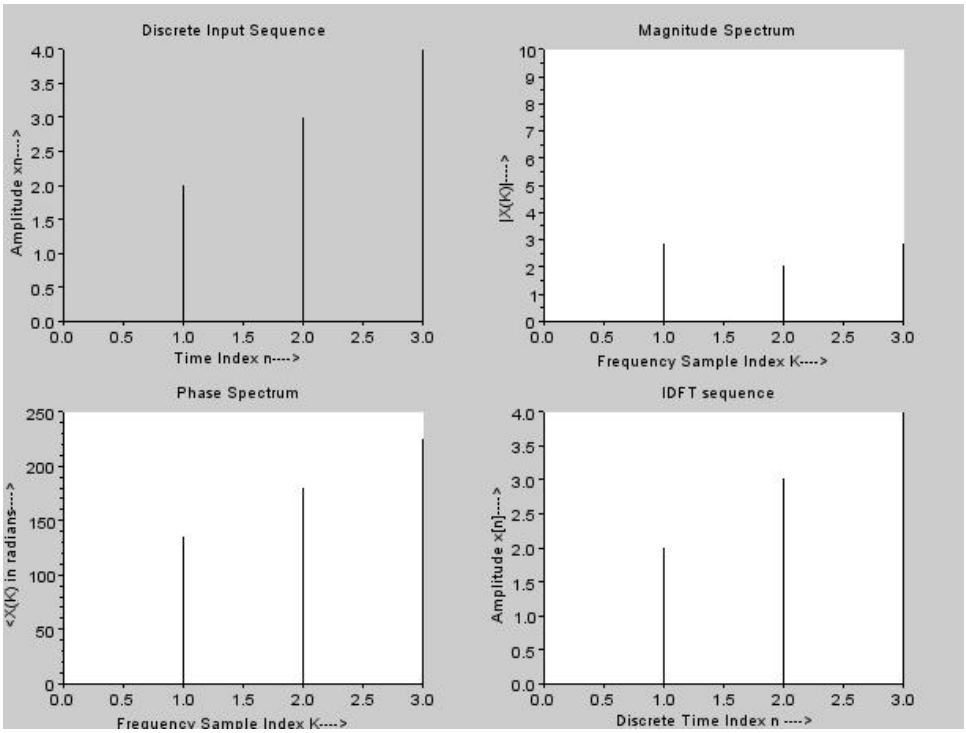

Figure 8.1: Linear Convolution using Circular Convolution DFT IDFT method

```
14 XK = dft(x,-1);//N point DFT of i/p sequence
15 HK = dft(h,-1);//N point DFT of impulse sequence
16 //Multiplication of 2 DFT's
17 YK = XK.*HK;
18 //Linear Convolution result
19 yn = dft(YK,1);//IDFT of Y(K)(o/p sequence)
20 disp(real(yn),'Linear Convolution result y[n]:=')
21 //Example
22 //Enter the input discrete sequence:= [1,2,3]
23 //Enter the impulse discrete sequence:=[1,2,2,1]
24 //Linear Convolution result y[n]:=
25 //
26 //     1.
27 //     4.
28 //     9.
29 //     11.
30 //     8.
31 //     3.
```

Experiment: 9

Circular convolution of two given sequences using DFT and IDFT

Scilab code Solution 9.1 Circular Convolution using DFT IDFT method

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

```
15 x3 = real(x3);
16 disp(x3,'Circular Convolution result:x3[n]=');
17 //Example
18 //Enter the first discrete sequence:x1[n]= [2,1,2,1]
19 //Enter the second discrete sequence:x2[n]=
     [1,2,3,4]
20 //
21 // Circular Convolution result:x3[n]=
22 //
23 //    14.
24 //    16.
25 //    14.
26 //    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

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

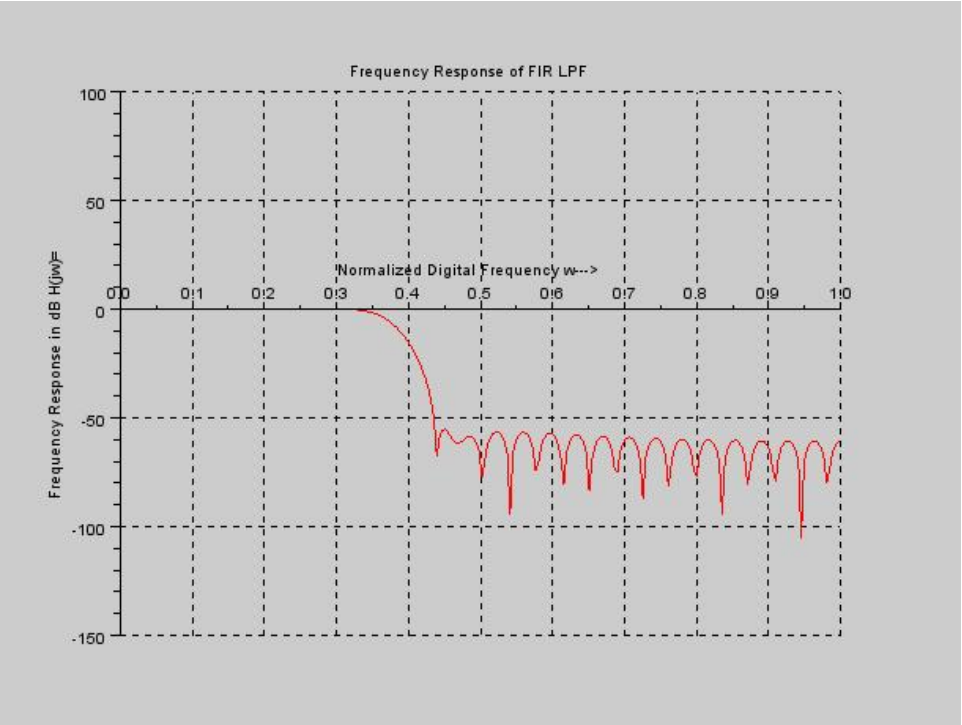

Figure 10.1: To Design an Low Pass FIR Filter

```
16 figure(1)
17 a = gca();
18 a.x_location = "origin";
19 a.y_location = "origin";
20 a.data_bounds = [0,-150;1,50];
21 plot(2*fr,20*log10(wfm),'r')
22 xlabel('Normalized Digital Frequency w-->')
23 ylabel('Frequency Response in dB H(jw)=')
24 title('Frequency Response of FIR LPF')
25 xgrid(1)
26 //Result
27 //Enter the pass band edge (rad)= 0.3*%pi
28 //Enter the stop band edge (rad)=  0.45*%pi
29 //Enter the stop band attenuation (dB)=  50
30 //N  =    54.
31 //-->wc
32 // wc  =    0.375
33 //-->disp(wft,'Impulse Response of FIR LPF=')
34 // Impulse Response of FIR LPF=
35 //    column 1 to 7
36 //    0.0003609  - 0.0007195  - 0.0010869    1.575D
     -18    0.0016485    0.0015927  - 0.0010883
37 //    column  8 to 14
38 //  - 0.0035703  - 0.0017009    0.0038764
     0.0061896  - 5.965D-18  - 0.0090208  - 0.0082516
39 //    column 15 to 21
40 //     0.0053105    0.0164428    0.0074408  -
     0.0162551  - 0.0251602    1.191D-17    0.0359480
41 //    column 22 to 28
42 //    0.0334760  - 0.0225187  - 0.0756838  -
     0.0394776    0.1111441    0.2931653    0.375
43 //    column 29 to 35
44 //    0.2931653    0.1111441  - 0.0394776  -
     0.0756838  - 0.0225187    0.0334760    0.0359480
45 //    column 36 to 42
46 //    1.191D-17  - 0.0251602  - 0.0162551
     0.0074408    0.0164428    0.0053105  - 0.0082516
47 //    column 43 to 49
```

```
48 //    - 0.0090208   - 5.965D-18     0.0061896
      0.0038764   - 0.0017009   - 0.0035703   - 0.0010883
       column 50 to 55
49 //    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

```
1 //Caption: To obtain Digital IIR Butterworth low
     pass filter
2 //Frequency response
3 clc;
4 clear;
5 close;
6 fp= input('Enter the pass band edge (Hz) = ');
7 fs= input('Enter the stop band edge (Hz) = ');
8 kp= input('Enter the pass band attenuation (dB) = ')
     ;
9 ks= input('Enter the stop band attenuation (dB) = ')
     ;
10 Fs= input('Enter the sampling rate  samples/sec = ')
     ;
```

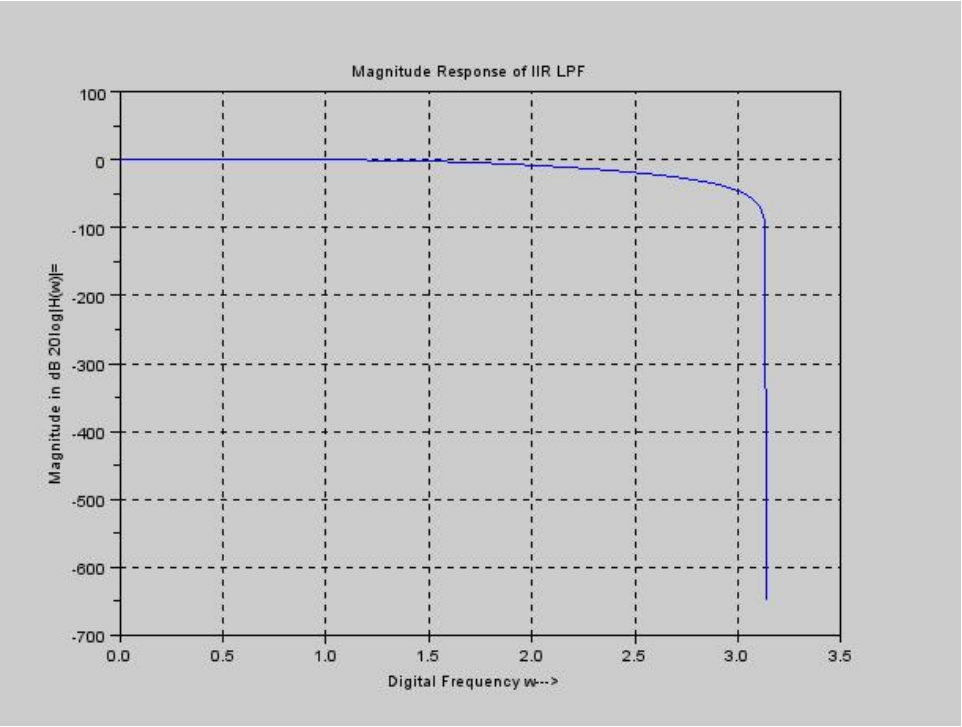

Figure 11.1: To obtain Digital IIR Butterworth low pass filter Frequency response

```
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.
//
```

```
80 // Cutoff Frequency in rad/seconds OC =
81 //
82 //    16022.769
83 //
84 // Gain of Analog IIR Butterworth LPF Gain =
85 //
86 //    2.567D+08
87 //
88 // Poles of Analog IIR Butterworth LPF Poles =
89 //
90 //  - 11329.809 + 11329.809i  - 11329.809 -
   11329.809i
91 //
92 // Transfer function of Ananlog IIR Butterworth LPF
   H(S)=
93 //
94 //              2.567D+08
95 //    ------------------------------
96 //                                  2
97 //    2.567D+08 + 22659.618s + s
98 //
99 // Transfer function of Digitla IIR Butterworth LPF
   H(Z)=
100 //
101 //                                          2
102 //    0.2933099 + 0.5866197z + 0.2933099z
103 //    ------------------------------------
104 //                                    2
105 //         0.1715734 + 0.0016661z + z
106 //
```

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

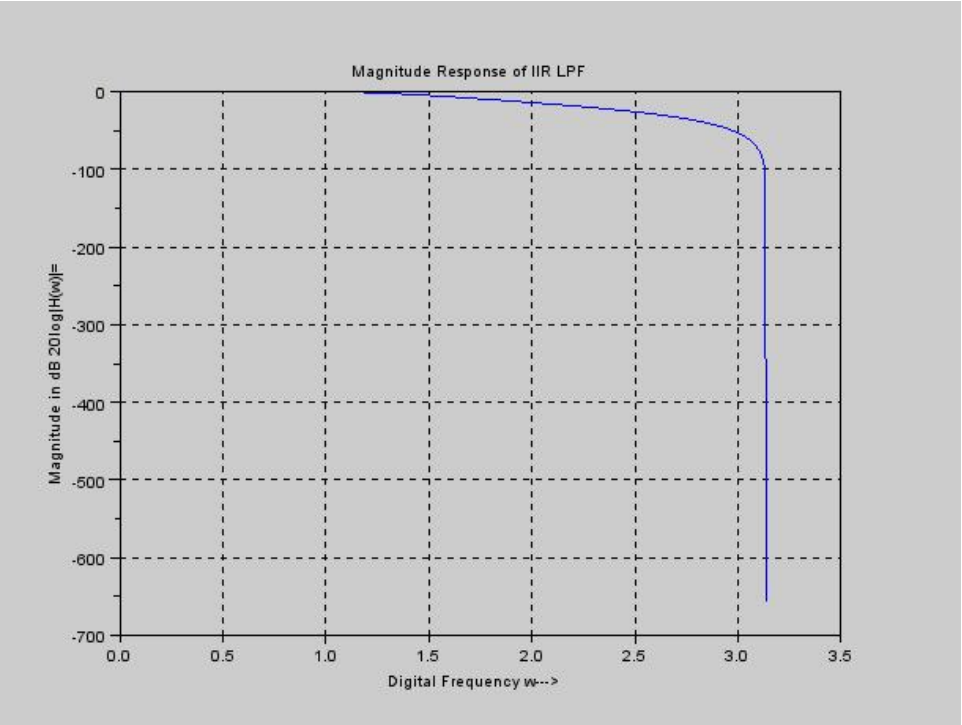

Figure 11.2: To obtain Digital IIR Chebyshev low pass filter Frequency response

```
1  //Caption: To obtain Digital IIR Chebyshev low pass
     filter
2  //Frequency response
3  clc;
4  clear;
5  close;
6  fp= input('Enter the pass band edge (Hz) = ');
7  fs= input('Enter the stop band edge (Hz) = ');
8  kp= input('Enter the pass band attenuation (dB) = ')
     ;
9  ks= input('Enter the stop band attenuation (dB) = ')
     ;
10 Fs= input('Enter the sampling rate  samples/sec = ')
     ;
11 d1 = 10^(kp/20);
12 d2 = 10^(ks/20);
13 d = sqrt((1/(d2^2))-1);
14 E = sqrt((1/(d1^2))-1);
15 //Digital filter specifications(rad/samples)
16 wp=2*%pi*fp*1/Fs;
17 ws=2*%pi*fs*1/Fs;
18 disp(wp,'Digital Pass band edge freq in rad/samples
     wp=')
19 disp(ws,'Digital Stop band edge freq in rad/samples
     ws=')
20 //Pre warping
21 op=2*Fs*tan(wp/2);
22 os=2*Fs*tan(ws/2);
23 disp(op,'Analog Pass Band Edge Freq. in rad/sec op='
     )
24 disp(os,'Analog Stop band Edge Freq. in rad/sec os='
     )
25 N = acosh(d/E)/acosh(os/op);
26 oc = op/((E^2)^(1/(2*N)));
27 N = ceil(N);//rounded to nearest integer
28 disp(N,'IIR Filter order N =');
29 disp(oc,'Cutoff Frequency in rad/seconds OC = ')
30 [pols,gn] = zpch1(N,E,op);
```

```
31 disp(gn,'Gain of Analog IIR Chebyshev Type-I LPF
     Gain =')
32 disp(pols,'Poles of Analog IIR Chebyshev Type-I LPF
     Poles =')
33 HS = poly(gn,'s','coeff')/real(poly(pols,'s'));
34 disp(HS,'Transfer function of Ananlog IIR Chebyshev
     Type-I LPF H(S)=')
35 z = poly(0,'z')
36 Hz = horner(HS,(2*Fs*(z-1)/(z+1)))
37 num = coeff(Hz(2))
38 den = coeff(Hz(3))
39 Hz(2)= Hz(2)./den(3);
40 Hz(3) = Hz(3)./den(3);
41 disp(Hz,'Transfer function of Digitla IIR Chebyshev
     LPF H(Z)=')
42 [Hw,w] = frmag(Hz,256);
43 figure(1)
44 plot(2*w*%pi,20*log10(abs(Hw)));
45 xlabel('Digital Frequency w-->')
46 ylabel('Magnitude in dB 20log|H(w)|=')
47 title('Magnitude Response of IIR LPF')
48 xgrid(1)
49 //Result
50 //Enter the pass band edge (Hz) =  1500
51 //
52 //Enter the stop band edge (Hz) =  2000
53 //
54 //Enter the pass band attenuation (dB) =  -1
55 //
56 //Enter the stop band attenuation (dB) =  -3
57 //
58 //Enter the sampling rate  samples/sec =  8000
59 //
60 // Digital Pass band edge freq in rad/samples wp=
61 //
62 //    1.1780972
63 //
64 // Digital Stop band edge freq in rad/samples ws=
```

```
65 //
66 //    1.5707963
67 //
68 // Analog Pass Band Edge Freq. in rad/sec op=
69 //
70 //    10690.858
71 //
72 // Analog Stop band Edge Freq. in rad/sec os=
73 //
74 //    16000.
75 //
76 // IIR Filter order N =
77 //
78 //    2.
79 //
80 //Cutoff Frequency in rad/seconds OC =
81 //
82 //    17642.912
83 //
84 // Gain of Analog IIR Chebyshev Type-I LPF Gain =
85 //
86 //    1.123D+08
87 //
88 // Poles of Analog IIR Chebyshev Type-I LPF Poles =
89 //
90 //  - 5867.861 + 9569.6927i  - 5867.861 - 9569.6927i
91 //
92 // Transfer function of Ananlog IIR Chebyshev Type-I
       LPF H(S)=
93 //
94 //              1.123D+08
95 //    ------------------------------
96 //                                 2
97 //    1.260D+08 + 11735.722s + s
98 //
99 // Transfer function of Digitla IIR Chebyshev LPF H(
     Z)=
100 //
```

```
101 //                                                  2
102 //      0.1971055 + 0.3942111 z + 0.1971055 z
103 //     ------------------------------------------
104 //                                        2
105 //           0.3409008 - 0.4562766 z + z
```

Experiment: 12

Circular convolution of two given sequences

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

```
1  //Caption: Program to perform circular convolution
       of two sequences
2  clc;
3  clear;
4  close;
5  x1 = input('Enter the first discrete sequence:=')
6  x2 = input('Enter the second discrete sequence:=')
7  m = length(x1);//length of first sequence
8  n = length(x2);//length of second sequence
9  //To make length of x1 and x2 are equal
10 if(m>n)
11     for i = n+1:m
12         x2(i)=0;
13     end
14 elseif(n>m)
15     for i = m+1:n
16         x1(i)=0;
17     end
```

```
18 end
19 N = length(x1);
20 x3 = zeros(1,N);//circular convolution result
      initialized to zero
21 a(1) = x2(1);
22 for j = 2:N
23     a(j) = x2(N-j+2);
24 end
25 for i = 1:N
26     x3(1) = x3(1)+x1(i)*a(i);
27 end
28 X(1,:) = a;
29 //Calculation of circular convolution
30 for k =2:N
31     for j = 2:N
32         x2(j) = a(j-1);
33     end
34 x2(1) = a(N);
35 X(k,:) = x2;
36     for i = 1:N
37         a(i) = x2(i);
38         x3(k) = x3(k)+x1(i)*a(i);
39     end
40 end
41 disp(x3,'Circular Convolution Result x3[n]=')
42 //Example
43 //Enter the first discrete sequence:= [2,1,2,1]
44 //Enter the second discrete sequence:= [1,2,3,4]
45 //Circular Convolution Result x3[n]=
46 //    14.    16.    14.    16.
47 //
```