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

April 23, 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

Experiment: 1

Generation of Discrete Signals

Scilab code Solution 1.1 Unit Sample Sequence

```
1  //Caption:Unit Sample Sequence
2  clear;
3  clc;
4  close;
5  L = 4;  //Upperlimit
6  n = -L:L;
7  x = [zeros(1,L),1,zeros(1,L)];
8  b = gca();
9  b.y_location = "middle";
10 plot2d3('gnn',n,x)
11 a=gce();
12 a.children(1).thickness =4;
13 xtitle('Graphical Representation of Unit Sample
      Sequence','n','x[n]');
```

Scilab code Solution 1.2 Unit Step Sequence

```
1  //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

```
1  //Caption: Exponentially Decreasing Signal
2  clear;
3  clc;
4  close;
5  a =0.5;
6  n = 0:10;
7  x = (a)^n;
8  a=gca();
9  a.x_location = "origin";
10 a.y_location = "origin";
11 plot2d3('gnn',n,x)
12 a.thickness = 2;
13 xtitle('Graphical Representation of Exponentially
      Decreasing Signal','n','x[n]');
```

Scilab code Solution 1.5 Exponentially Increasing Signal

```
1  //Caption:Exponentially Increasing Signal
2  clear;
3  clc;
4  close;
5  a =1.5;
6  n =1:10;
7  x = (a)^n;
8  a=gca();
9  a.thickness = 2;
10 plot2d3('gnn',n,x)
11 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

```
1  //Caption:Program for Linear Convolution
2  clc;
3  clear all;
4  close ;
5  x = input('enter x seq');
6  h = input('enter h seq');
7  m = length(x);
8  n = length(h);
9  //Method 1 Using Direct Convolution Sum Formula
10 for i = 1:n+m-1
11     conv_sum = 0;
12     for j = 1:i
13         if (((i-j+1) <= n)&(j <= m))
14             conv_sum = conv_sum + x(j)*h(i-j+1);
15         end;
16         y(i) = conv_sum;
17     end;
18 end;
19 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

```
1  //Caption: Program to find the Cicrcular Convolution
       of given
2  //discrete sequences using Matrix method
3
4  clear;
5  clc;
6  x1 = [2,1,2,1]; //First sequence
7  x2 = [1,2,3,4]; //Second sequence
8  m = length(x1); //length of first sequence
9  n = length(x2); //length of second sequence
10 //To make length of x1 and x2 are Equal
11 if (m >n)
12    for i = n+1:m
13      x2(i) = 0;
14    end
15 elseif (n>m)
16    for i = m+1:n
17      x1(i) = 0;
18    end
19 end
20 N = length(x1);
21 x3 = zeros(1,N); //x3 = Circular convolution result
22 a(1) = x2(1);
23 for j = 2:N
24   a(j) = x2(N-j+2);
25 end
26 for i =1:N
27   x3(1) = x3(1)+x1(i)*a(i);
28 end
29 X(1,:)=a;
30 //Calculation of circular convolution
31 for k = 2:N
```

```
32      for j =2:N
33         x2(j) = a(j-1);
34      end
35         x2(1) = a(N);
36         X(k,:)= x2;
37      for i = 1:N
38          a(i) = x2(i);
39          x3(k) = x3(k)+x1(i)*a(i);
40      end
41  end
42  disp(X,'Circular Convolution Matrix x2[n]=')
43  disp(x3,'Circular Convolution Result x3[n] = ')
44  //Result
45  //Circular Convolution Matrix x2[n]=
46  //
47  //     1.     4.     3.     2.
48  //     2.     1.     4.     3.
49  //     3.     2.     1.     4.
50  //     4.     3.     2.     1.
51  //
52  // Circular Convolution Result x3[n] =
53  //
54  //     14.    16.    14.    16.
```

Experiment: 3

Circular convolution using FFT

Scilab code Solution 3.1 Performing Circular COnvolution Using DFT IDFT method

```
1 //Caption:Performing Circular COnvolution Using DFT-
    IDFT method
2 clear all;
3 clc;
4 close;
5 L = 4; //Length of the Sequence
6 N = 4;  // N-point DFT
7 x1 = [2,1,2,1];
8 x2 = [1,2,3,4];
9 //Computing DFT
10 X1 = fft(x1,-1);
11 X2 = fft(x2,-1);
12 disp(X1,'DFT of x1[n] is X1(k)=')
13 disp(X2,'DFT of x1[n] is X2(k)=')
14 //Multiplication of 2 DFTs
15 X3 = X1.*X2;
16 disp(X3,'DFT of x3[n] is X3(k)=')
17 //Circular Convolution Result
18 x3 =abs(fft(X3,1))
19 disp(x3,'Circular Convolution Result x3[n]=')
```

```
20 //Result
21 // DFT of x1[n] is X1(k)=
22 //
23 //     6.     0     2.     0
24 //
25 // DFT of x1[n] is X2(k)=
26 //
27 //     10.  - 2. + 2.i  - 2.  - 2. - 2.i
28 //
29 // DFT of x3[n] is X3(k)=
30 //
31 //     60.     0  - 4.     0
32 //
33 // Circular Convolution Result x3[n]=
34 //
35 //     14.     16.     14.     16.
```

Experiment: 4

Linear Convolution using Circular Convolution

Scilab code Solution 4.1 Performing Linear Convolution using Circular Convolution

```
1  //Caption: Performing Linear Convolution using
       Circular Convolution
2
3  clear;
4  clc;
5  close;
6  h = [1,2,3];     //Impulse Response of LTI System
7  x = [1,2,2,1];   //Input Response of LTI System
8  N1 = length(x);
9  N2 = length(h);
10 N = N1+N2-1
11 disp(N,'Length of Output Response y(n)')
12 //Padding zeros to Make Length of 'h' and 'x'
13 //Equal to length of output response 'y'
14 h1 = [h,zeros(1,N-N2)];
15 x1 = [x,zeros(1,N-N1)];
16 //Computing FFT
17 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

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

```
19 b.children(1).thickness =3;
20 xtitle('Graphical Representation of Discrete
      Sequence','n','x[n]');
21 subplot(2,1,2)
22 a=gce();
23 a.data_bounds=[0,0;5,10];
24 plot2d3('gnn',0:length(X)-1,abs(X))
25 b = gce();
26 b.children(1).thickness =3;
27 xtitle('Graphical Representation of Discrete
      Spectrum','k','X(k)');
28 //Result
29 //FFT of x[n] is X(k)=
30 //
31 //     10.  - 2. + 2.i  - 2.  - 2. - 2.i
32 //
33 //IFFT of X(k) is x[n]=
34 //
35 //     1.    2.    3.    4.
```

Experiment: 6

Time and Frequency Response of LTI systems

Scilab code Solution 6.1 Time and Frequency Response

```
1 //Caption: Program to generate and plot the impulse
    response and frequency
2 //response of a Linear constant coefficient first
    order Differential Equation
3 //[1].Impulse response h(t)= exp(-a*t)u(t), A>0
4 //[2].Frequency response H(jw) = 1/(jw+a)
5 clear;
6 clc;
7 close;
8 //[1]. To generate and plot the impulse response
9 a =1;     //Constant coefficient a =1
10 Dt = 0.005;
11 t = 0:Dt:10;
12 ht = exp(-a*t);
13 figure(1)
14 a = gca();
15 a.y_location = "origin";
16 plot(t,ht);
17 xlabel('time t ------->');
```

```
18 ylabel('h(t)')
19 title('Impulse Repsonse of Ist Order Linear Constant
       Coeff. Differential Equ.')
20 //
21 //[2]. Finding Frequency response using Continuous
     Time Fourier Transform
22 Wmax = 2*%pi*1;          //Analog Frequency = 1Hz
23 K = 4;
24 k = 0:(K/1000):K;
25 W = k*Wmax/K;
26 HW = ht* exp(-sqrt(-1)*t'*W) * Dt;
27 HW_Mag = abs(HW);
28 W = [-mtlb_fliplr(W), W(2:1001)]; // Omega from -
     Wmax to Wmax
29 HW_Mag = [mtlb_fliplr(HW_Mag),HW_Mag(2:1001)];
30 [HW_Phase,db] = phasemag(HW);
31 HW_Phase = [-mtlb_fliplr(HW_Phase),HW_Phase(2:1001)
     ];
32 figure(2)
33 //Plotting Magnitude Response
34 subplot(2,1,1);
35 a = gca();
36 a.y_location = "origin";
37 plot(W,HW_Mag);
38 xlabel('Frequency in Radians/Seconds--> W');
39 ylabel('abs(H(jW))')
40 title('Magnitude Response')
41 //Plotting Phase Reponse
42 subplot(2,1,2);
43 a = gca();
44 a.y_location = "origin";
45 a.x_location = "origin";
46 plot(W,HW_Phase*%pi/180);
47 xlabel('                          Frequency in
     Radians/Seconds--> W');
48 ylabel('
                                                  <H
     (jW)')
```

```
49 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

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

```
13 //Continuous Time Signal
14 xa = exp(-A*abs(t));
15 //Discrete Time Signal
16 Fs =input('Enter the Sampling Frequency in Hertz');
     //Fs = 1Hz,2Hz,4Hz,20Hz,100Hz
17 Ts = 1/Fs;
18 nTs = -2:Ts:2;
19 x = exp(-A*abs(nTs));
20 // Analog Signal reconstruction
21 Dt = 0.005;
22 t = -2:Dt:2;
23 Xa = x *sincnew(Fs*(ones(length(nTs),1)*t-nTs'*ones
     (1,length(t))));
24 //Plotting the original signal and reconstructed
     signal
25 subplot(2,1,1);
26 a =gca();
27 a.x_location = "origin";
28 a.y_location = "origin";
29 plot(t,xa);
30 xlabel('t in sec.');
31 ylabel('xa(t)')
32 title('Original Analog Signal')
33 subplot(2,1,2);
34 a =gca();
35 a.x_location = "origin";
36 a.y_location = "origin";
37 xlabel('t in sec.');
38 ylabel('xa(t)')
39 title('Reconstructed Signal using sinc function, Fs
     = 100Hz');
40 plot(t,Xa);
```

Experiment: 8

Design of FIR Filters Window Design

Scilab code Solution 8.1 Program to Design FIR Low Pass Filter

```
1  //Caption: Program to Design FIR Low Pass Filter
2  clc;
3  close;
4  M = input('Enter the Odd Filter Length =');
                  //Filter length
5  Wc = input('Enter the Digital Cutoff frequency =');
     //Digital Cutoff frequency
6  Tuo = (M-1)/2      //Center Value
7  for n = 1:M
8      if (n == Tuo+1)
9        hd(n) = Wc/%pi;
10     else
11       hd(n) =  sin(Wc*((n-1)-Tuo))/(((n-1)-Tuo)*%pi)
             ;
12     end
13 end
14 //Rectangular Window
15 for n = 1:M
16   W(n) = 1;
```

```
17 end
18 //Windowing Fitler Coefficients
19 h = hd.*W;
20 disp(h,'Filter Coefficients are')
21
22 [hzm,fr]=frmag(h,256);
23 hzm_dB = 20*log10(hzm)./max(hzm);
24 subplot(2,1,1)
25 plot(2*fr,hzm)
26 xlabel('Normalized Digital Frequency W');
27 ylabel('Magnitude');
28 title('Frequency Response 0f FIR LPF using
      Rectangular window')
29 xgrid(1)
30 subplot(2,1,2)
31 plot(2*fr,hzm_dB)
32 xlabel('Normalized Digital Frequency W');
33 ylabel('Magnitude in dB');
34 title('Frequency Response 0f FIR LPF using
      Rectangular window')
35 xgrid(1)
36 //Result
37 //Enter the Odd Filter Length = 7
38 //Enter the Digital Cutoff frequency = %pi/2
39 //
40 // Filter Coefficients are
41 //
42 //   - 0.1061033
43 //     1.949D-17   = 0.0
44 //     0.3183099
45 //     0.5
46 //     0.3183099
47 //     1.949D-17 = 0.0
48 //   - 0.1061033
```

Scilab code Solution 8.2 rogram to Design FIR High Pass Filter

```
1  //Caption: Program to Design FIR High Pass Filter
2  clear;
3  clc;
4  close;
5  M = input('Enter the Odd Filter Length =');
                  //Filter length
6  Wc = input('Enter the Digital Cutoff frequency =');
     //Digital Cutoff frequency
7  Tuo = (M-1)/2      //Center Value
8  for n = 1:M
9      if (n == Tuo+1)
10       hd(n) = 1-Wc/%pi;
11     else
12       hd(n) = (sin(%pi*((n-1)-Tuo)) -sin(Wc*((n-1)-
            Tuo)))/(((n-1)-Tuo)*%pi);
13     end
14 end
15 //Rectangular Window
16 for n = 1:M
17   W(n) = 1;
18 end
19 //Windowing Fitler Coefficients
20 h = hd.*W;
21 disp(h,'Filter Coefficients are')
22 [hzm,fr]=frmag(h,256);
23 hzm_dB = 20*log10(hzm)./max(hzm);
24 subplot(2,1,1)
25 plot(2*fr,hzm)
26 xlabel('Normalized Digital Frequency W');
27 ylabel('Magnitude');
28 title('Frequency Response 0f FIR HPF using
     Rectangular window')
29 xgrid(1)
30 subplot(2,1,2)
31 plot(2*fr,hzm_dB)
32 xlabel('Normalized Digital Frequency W');
```

```
33 ylabel('Magnitude in dB');
34 title('Frequency Response 0f FIR HPF using
      Rectangular window')
35 xgrid(1)
36 //Result
37 //Enter the Odd Filter Length = 5
38 //Enter the Digital Cutoff frequency = %pi/4
39 //Filter Coefficients are
40 //
41 //   - 0.1591549
42 //   - 0.2250791
43 //     0.75
44 //   - 0.2250791
45 //   - 0.1591549
```

Scilab code Solution 8.3 Program to Design FIR Band Pass Filter

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

```
17          n
18       hd(n) = (sin(Wc2*((n-1)-Tuo)) -sin(Wc1*((n-1)-
            Tuo)))/(((n-1)-Tuo)*%pi);
19    end
20    if(abs(hd(n))<(0.00001))
21      hd(n)=0;
22    end
23  end
24  hd;
25  //Rectangular Window
26  for n = 1:M
27    W(n) = 1;
28  end
29  //Windowing Fitler Coefficients
30  h = hd.*W;
31  disp(h,'Filter Coefficients are')
32  [hzm,fr]=frmag(h,256);
33  hzm_dB = 20*log10(hzm)./max(hzm);
34  subplot(2,1,1)
35  plot(2*fr,hzm)
36  xlabel('Normalized Digital Frequency W');
37  ylabel('Magnitude');
38  title('Frequency Response 0f FIR BPF using
       Rectangular window')
39  xgrid(1)
40  subplot(2,1,2)
41  plot(2*fr,hzm_dB)
42  xlabel('Normalized Digital Frequency W');
43  ylabel('Magnitude in dB');
44  title('Frequency Response 0f FIR BPF using
       Rectangular window')
45  xgrid(1)
46  //Result
47  //Enter the Odd Filter Length = 11
48  //Enter the Digital Cutoff frequency = [%pi/4,3*%pi
       /4]
49  //Filter Coefficients are
50  // 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

```
1  //Caption: Program to Design FIR Band Reject Filter
2  clear ;
3  clc;
4  close;
5  M = input('Enter the Odd Filter Length =');
                  //Filter length
6  //Digital Cutoff frequency [Lower Cutoff, Upper
      Cutoff]
7  Wc = input('Enter the Digital Cutoff frequency =');
8  Wc2 = Wc(2)
9  Wc1 = Wc(1)
10 Tuo = (M-1)/2      //Center Value
11 hd = zeros(1,M);
12 W = zeros(1,M);
13 for n = 1:M
14  if (n == Tuo+1)
15  hd(n) = 1-((Wc2-Wc1)/%pi);
16  else
17  hd(n)=(sin(%pi*((n-1)-Tuo))-sin(Wc2*((n-1)-Tuo))+
       sin(Wc1*((n-1)-Tuo)))/(((n-1)-Tuo)*%pi);
18  end
19   if(abs(hd(n))<(0.00001))
20     hd(n)=0;
21   end
22 end
23
24 //Rectangular Window
25 for n = 1:M
26   W(n) = 1;
27 end
28 //Windowing Fitler Coefficients
```

```
29 h = hd.*W;
30 disp(h,'Filter Coefficients are')
31 [hzm,fr]=frmag(h,256);
32 hzm_dB = 20*log10(hzm)./max(hzm);
33 subplot(2,1,1)
34 plot(2*fr,hzm)
35 xlabel('Normalized Digital Frequency W');
36 ylabel('Magnitude');
37 title('Frequency Response 0f FIR BSF using
      Rectangular window')
38 xgrid(1)
39 subplot(2,1,2)
40 plot(2*fr,hzm_dB)
41 xlabel('Normalized Digital Frequency W');
42 ylabel('Magnitude in dB');
43 title('Frequency Response 0f FIR BSF using
      Rectangular window')
44 xgrid(1)
45 //Result
46 //Enter the Odd Filter Length = 11
47 //Enter the Digital Cutoff frequency =[%pi/3,2*%pi
      /3]
48 //Filter Coefficients are
49 //column 1 to 9
50 //    0.  - 0.1378322    0.    0.2756644    0.
      0.6666667    0.    0.2756644    0.
51 //column 10 to 11
52 //  - 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

```
1  //Cpation: Design of FIR LPF Filter using Frequecny
       Sampling Technique
2
3  clear;
4  clc;
5  close;
6  M =15;
7  Hr = [1,1,1,1,0.4,0,0,0];
8  for k =1:length(Hr)
9      G(k)=((-1)^(k-1))*Hr(k);
10 end
11 h = zeros(1,M);
12 U = (M-1)/2
13 for n = 1:M
14   h1 = 0;
15   for k = 2:U+1
16     h1 =G(k)*cos((2*%pi/M)*(k-1)*((n-1)+(1/2)))+h1;
17   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

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

```
15 a.thickness = 1;
16 plot(W/%pi,HW,'r')
17 a.foreground = 1;
18 a.font_style = 9;
19 xgrid(1)
20 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

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

```
19 a.foreground = 1;
20 a.font_style = 9;
21 xgrid(1)
22 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

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

```
18 HW  =frmag(HZ_BPF(2),HZ_BPF(3),512);//frequency
     response
19 W = 0:%pi/511:%pi;
20 a=gca();
21 a.thickness = 1;
22 plot(W/%pi,HW,'r')
23 a.foreground = 1;
24 a.font_style = 9;
25 xgrid(1)
26 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

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

```
    Numerator
14 DEN = (1-((2*alpha/(1+k))*z)+(((1-k)/(1+k))*(z^2)));
    //Denominator
15 HZ_BSF=horner(H_LPF,NUM/DEN); //LPF to BSF
    conversion using digital transformation
16 HW  =frmag(HZ_BSF(2),HZ_BSF(3),512); //frequency
    response for 512 points
17 W = 0:%pi/511:%pi;
18 a=gca();
19 a.thickness = 1;
20 plot(W/%pi,HW,'r')
21 a.foreground = 1;
22 a.font_style = 9;
23 xgrid(1)
24 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

```
1  //Program To Design the Digtial Chebyshev IIR Filter
2  clear;
3  clc;
4  close;
5  Wp = input('Enter the Digital Pass Band Edge
       Frequency');
6  Ws = input('Enter the Digital Stop Band Edge
       Frequency');
7  T = input('Sampling Interval')
8  OmegaP = (2/T)*tan(Wp/2)
9  OmegaS = (2/T)*tan(Ws/2)
10 Delta1 = input('Enter the Pass Band Ripple');
11 Delta2 = input('Enter the Stop Band Ripple');
12 Delta = sqrt(((1/Delta2)^2)-1)
13 Epsilon = sqrt(((1/Delta1)^2)-1)
14 N = (acosh(Delta/Epsilon))/(acosh(OmegaS/OmegaP))
15 N = ceil(N)
16 OmegaC = OmegaP/((((1/Delta1)^2)-1)^(1/(2*N)))
17 [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  =
//
```

```
53 //      1.2079548
54 // N   =
55 //
56 //      2.
57 // OmegaC   =
58 //
59 //      0.7503699
60 // gn   =
61 //
62 //      0.2815275
63 // pols   =
64 //
65 //   - 0.2652958 + 0.5305916 i   - 0.2652958 -
       0.5305916 i
66 // Hs   =
67 //
68 //                0.2815275
69 //      ------------------------------
70 //                                    2
71 //      0.3519094 + 0.5305916 s + s
72 // Hz   =
73 //
74 //                                              2
75 //      0.2815275 + 0.5630550 z + 0.2815275 z
76 //      ---------------------------------------
77 //                                              2
78 //      3.2907261 - 7.2961813 z + 5.4130926 z
79 //-->0.5*0.5629
80 // ans   =
81 //
82 //      0.28145
83 //
84 //-->Hz(2)= Hz(2)/5.4130926
85 // Hz   =
86 //
87 //                                              2
88 //      0.0520086 + 0.1040172 z + 0.0520086 z
89 //      ---------------------------------------
```

```
90 //                                                 2
91 //      3.2907261 - 7.2961813z + 5.4130926z
92 //
93 //-->Hz(3) = Hz(3)/5.4130926
94 // Hz  =
95 //
96 //                                                 2
97 //      0.0520086 + 0.1040172z + 0.0520086z
98 //     ------------------------------------------
99 //                                       2
100//           0.6079198 - 1.3478767z + z
101//
```

Experiment: 12

Decimation by polyphase decomposition

Scilab code Solution 12.1 Design of Ployphase Decimator

```
1 //Caption:Decimation by 2, Filter Length = 30
2 //Cutoff Frequency Wc = %pi/2
3 //Pass band Edge frequency fp = 0.25 and a Stop band
      edge frequency fs = 0.31
4 // Choose the number of cosine functions and create
     a dense grid
5 // in [0,0.25] and [0.31,0.5]
6 //magnitude for pass band = 1 & stop band = 0 (i.e)
     [1 0]
7 //Weighting function =[2 1]
8 clear;
9 clc;
10 close;
11 M = 30;  //Filter Length
12 D = 2;  //Decimation Factor = 2
13 Wc = %pi/2;  //Cutoff Frequency
14 Wp = Wc/(2*%pi);  //Passband Edge Frequency
15 Ws = 0.31;  //Stopband Edge Frequency
16 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
```

```
48 //0.4922476  - 0.1187325     0.0640230  - 0.0349808
       0.0159915  - 0.0046761  - 0.0028534
49 //column 15
50 //- 0.0128037
51 //   0.0060203
```

Experiment: 13

Periodogram based Spectral Estimation

Scilab code Solution 13.1 Periodogram Estimate of Given Discrete Sequence

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

```
17 a.foreground = 5;
18 a.font_color = 5;
19 a.font_style = 5;
20 title('Peridogram Estimate')
21 xlabel('Discrete Frequency Variable K ------>')
22 ylabel('Periodogram Pxx (k /N) ------>')
23 //Result
24 //DFT of x(n)is X(k)=
25 //
26 //    9.
27 //  - 1.2928932 + 0.1213203i
28 //    2. + i
29 //  - 2.7071068 + 4.1213203i
30 //    3. - 3.674D-16i
31 //  - 2.7071068 - 4.1213203i
32 //    2. - i
33 //  - 1.2928932 - 0.1213203i
34 //
35 // Peridogram of x(n) is Pxx(k/N)=
36 //
37 //    10.125
38 //    0.2107864
39 //    0.625
40 //    3.0392136
41 //    1.125
42 //    3.0392136
43 //    0.625
44 //    0.2107864
```

Appendix

Scilab code AP 1

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

sinc function