

Scilab Manual for  
Digital Signal Processing  
by Prof R.Senthilkumar, Assistant Professor  
Electronics Engineering  
Institute of Road and Transport Technology<sup>1</sup>

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

April 3, 2026

<sup>1</sup>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 Circular Convolution . . . . .	9
Solution 3.1	Performing Circular CONvolution Using DFT IDFT method . . . . .	11
Solution 4.1	Performing Linear Convolution using Circular Con- volution . . . . .	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 Sam- pling 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 Digital 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

```
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
```

```

2 clear;
3 clc;
4 close;
5 L = 4; //Upperlimit
6 n = -L:L;
7 x = [zeros(1,L),ones(1,L+1)];
8 a=gca();
9 a.y_location = "middle";
10 plot2d3('gmn',n,x)
11 title('Graphical Representation of Unit Step Signal'
      )
12 xlabel('                                n'
      );
13 ylabel('                                x
      [n] ');

```

---

### Scilab code Solution 1.3 Discrete Ramp Sequence

```

1 //Caption: Discrete Ramp Sequence
2 clear;
3 clc;
4 close;
5 L = 4; //Upperlimit
6 n = -L:L;
7 x = [zeros(1,L),0:L];
8 b = gca();
9 b.y_location = 'middle';
10 plot2d3('gmn',n,x)
11 a=gce();
12 a.children(1).thickness =2;
13 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
```

```

    =')
20 //Method 2 Using Inbuilt Function
21 f = convol(x,h)
22 disp(f,'Convolution Sum Result using Inbuilt Funtion
    =')
23 //Method 3 Using frequency Domain multiplication
24 N = n+m-1;
25 x = [x zeros(1,N-m)];
26 h = [h zeros(1,N-n)];
27 f1 = fft(x)
28 f2 = fft(h)
29 f3 = f1.*f2; // freq domain multiplication
30 f4 = ifft(f3)
31 disp(f4,'Convolution Sum Result DFT – IDFT method =')
    )
32 //f4 = real(f4)
33 subplot(3,1,1);
34 plot2d3('gnn',x)
35 xtitle('Graphical Representation of Input signal x')
    ;
36 subplot(3,1,2);
37 plot2d3('gnn',h)
38 xtitle('Graphical Representation of Impulse signal h
    ');
39 subplot(3,1,3);
40 plot2d3('gnn',y)
41 xtitle('Graphical Representation of Output signal y')
    );
42 //Result
43 //enter x seq [1 1 1 1]
44 //enter h seq [1 2 3]
45 // Convolution Sum using Direct Formula Method =
46 //      1.      3.      6.      6.      5.      3.
47 // Convolution Sum Result using Inbuilt Funtion =
48 //      1.      3.      6.      6.      5.      3.
49 // Convolution Sum Result DFT – IDFT method =
50 //      1.      3.      6.      6.      5.      3.

```

---

## Scilab code Solution 2.2 Program to find the Circular Convolution

```
1 //Caption: Program to find the Circular 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);
```

```

18 X = fft(x1,-1);
19 //Multiplication of 2 DFTs
20 Y = X.*H
21 //Linear Convolution Result
22 y =abs(fft(Y,1))
23 disp(X, 'DFT of i/p X(k)=')
24 disp(H, 'DFT of impulse sequence H(k)=')
25 disp(Y, 'DFT of Linear Filter o/p Y(k)=')
26 disp(y, 'Linear Convolution result y[n]=')
27 //Result
28 // Length of Output Response y(n)
29 //
30 //      6.
31 //
32 // DFT of i/p X(k)=
33 //
34 //      6.  - 3.4641016i    0    0    0    3.4641016i
35 //
36 // DFT of impulse sequence H(k)=
37 //
38 //      6.      0.5 - 4.330127i  - 1.5 + 0.8660254i
39 //      2.  - 1.5 - 0.8660254i    0.5 + 4.330127i
40 //
41 // DFT of Linear Filter o/p Y(k)=
42 //
43 //      36.  - 15. - 1.7320508i    0    0    0  - 15.
44 //      + 1.7320508i
45 //
46 //      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  $H(j\omega) = 1/(j\omega + a)$ 
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 = [-mtlbfliplr(W), W(2:1001)]; // Omega from -
      Wmax to Wmax
29 HW_Mag = [mtlbfliplr(HW_Mag),HW_Mag(2:1001)];
30 [HW_Phase,db] = phasemag(HW);
31 HW_Phase = [-mtlbfliplr(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('
      (jW)')

```

<H

```
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        ;
13    end
14 end
15 //Rectangular Window
16 for n = 1:M
17     W(n) = 1;
```

```

17 end
18 //Windowing Filter 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 Of 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 Of 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 Filtler 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 of 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 Of 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:M
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 Filter 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 of 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 of 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 Filter 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 Of 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 Of 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 Frequency Sampling Technique

```
1 //Caption: Design of FIR LPF Filter using Frequency
   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
```

```

18  h(n) = (1/M)* (G(1)+2*h1);
19  end
20  disp(h, 'Filter Coefficients are h(n)=')
21  [hzm, fr]=frmag(h,256);
22  hzm_dB = 20*log10(hzm)./max(hzm);
23  subplot(2,1,1)
24  plot(2*fr, hzm)
25  a=gca();
26  xlabel('Normalized Digital Frequency W');
27  ylabel('Magnitude');
28  title('Frequency Response Of FIR LPF using Frequency
        Sampling Technique with M = 15 with Cutoff
        Frequency = 0.466 ')
29  xgrid(2)
30  subplot(2,1,2)
31  plot(2*fr, hzm_dB)
32  a=gca();
33  xlabel('Normalized Digital Frequency W');
34  ylabel('Magnitude in dB');
35  title('Frequency Response Of FIR LPF using Frequency
        Sampling Technique with M = 15 with Cutoff
        Frequency = 0.466 ')
36  xgrid(2)
37  //Result
38  //Filter Coefficients are h(n)=
39  //column 1 to 7
40  //
41  // -0.0141289   -0.0019453   0.04   0.0122345
42  //          -0.0913880   -0.0180899   0.3133176
43  //
44  //column 8 to 14
45  //0.52         0.3133176   - 0.0180899   - 0.0913880
46  //          0.0122345         0.04   - 0.0019453
47  //
48  //column 15
49  //
50  // - 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 Digital Chebyshev IIR Filter

```
1 //Program To Design the Digital 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)
```

```

18 Hs = poly(gn, 's', 'coeff')/real(poly(pols, 's'))
19 z = poly(0, 'z');
20 Hz = horner(Hs, ((2/T)*((z-1)/(z+1))))
21 HW = frmag(Hz(2), Hz(3), 512); //Frequency response
    for 512 points
22 W = 0:%pi/511:%pi;
23 a=gca();
24 a.thickness = 1;
25 plot(W/%pi, abs(HW), 'r')
26 a.foreground = 1;
27 a.font_style = 9;
28 xgrid(1)
29 xtitle('Magnitude Response of Chebyshev LPF Filter',
    'Normalized Digital Frequency—>', 'Magnitude in
    dB');
30 //RESULT
31 //Enter the Digital Pass Band Edge Frequency 0.2*%pi
32 //Enter the Digital Stop Band Edge Frequency 0.6*%pi
33 //Sampling Interval 1
34 // T =
35 //
36 // 1.
37 // OmegaP =
38 //
39 // 0.6498394
40 // OmegaS =
41 //
42 // 2.7527638
43 //Enter the Pass Band Ripple 0.8
44 //Enter the Stop Band Ripple 0.2
45 // Delta =
46 //
47 // 4.8989795
48 // Epsilon =
49 //
50 // 0.75
51 // N =
52 //

```

```

53 //      1.2079548
54 // N =
55 //
56 //      2.
57 // OmegaC =
58 //
59 //      0.7503699
60 // gn =
61 //
62 //      0.2815275
63 // polys =
64 //
65 //      - 0.2652958 + 0.5305916 i - 0.2652958 -
66 //      0.5305916 i
67 //
68 //      0.2815275
69 //      -----
70 //
71 //      0.3519094 + 0.5305916 s + s2
72 // Hz =
73 //
74 //
75 //      0.2815275 + 0.5630550 z + 0.2815275 z2
76 //      -----
77 //
78 //      3.2907261 - 7.2961813 z + 5.4130926 z2
79 // -->0.5*0.5629
80 // ans =
81 //
82 //      0.28145
83 //
84 // -->Hz(2)= Hz(2)/5.4130926
85 // Hz =
86 //
87 //
88 //      0.0520086 + 0.1040172 z + 0.0520086 z2
89 //      -----

```

```

90 //
91 //      3.2907261 - 7.2961813z + 5.4130926z2
92 //
93 //→Hz(3) = Hz(3)/5.4130926
94 // Hz =
95 //
96 //
97 //      0.0520086 + 0.1040172z + 0.0520086z2
98 //      -----
99 //
100 //      0.6079198 - 1.3478767z + z2
101 //

```

---

## Experiment: 12

# Decimation by polyphase decomposition

Scilab code Solution 12.1 Design of Polyphase 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]);
```

```

17 disp(hn, 'The LPF Filter Coefficients are:')
18 //Obtaining Polyphase Filter Coefficients from hn
19 p = zeros(D,M/D);
20 for k = 1:D
21     for n = 1:(length(hn)/D)
22         p(k,n) = hn(D*(n-1)+k);
23     end
24 end
25 disp(p, 'The Polyphase Decimator for D =2 are:')
26 //Result
27 //The LPF Filter Coefficients are:
28 //column 1 to 7
29 //0.0060203 - 0.0128037 - 0.0028534 0.0136687
   // - 0.0046761 - 0.0197002 0.0159915
30
31 //column 8 to 14
32 //0.0213811 - 0.0349808 - 0.0156251 0.0640230
   // - 0.0073600 - 0.1187325 0.0980522
33 //column 15 to 21
34 //0.4922476 0.4922476 0.0980522 - 0.1187325
   // - 0.0073600 0.0640230 - 0.0156251
35 //column 22 to 28
36 // - 0.0349808 0.0213811 0.0159915 - 0.0197002
   // - 0.0046761 0.0136687 - 0.0028534
37
38 //column 29 to 30
39 // - 0.0128037 0.0060203
40
41 //The Polyphase Decimator for D =2 are:
42 //column 1 to 7
43 //0.0060203 - 0.0028534 - 0.0046761 0.0159915
   // - 0.0349808 0.0640230 - 0.1187325
44 // - 0.0128037 0.0136687 - 0.0197002 0.0213811
   // - 0.0156251 - 0.0073600 0.0980522
45
46 //column 8 to 14
47 //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('gmn', [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 + 0.1213203 i
28 //      2. + i
29 // - 2.7071068 + 4.1213203 i
30 //      3. - 3.674D-16i
31 // - 2.7071068 - 4.1213203 i
32 //      2. - i
33 // - 1.2928932 - 0.1213203 i
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 11 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

---