Scilab Textbook Companion for Digital Signal Processing: Principle, Algorithms And Applications by J. G. Proakis And D. G. Manolakis[1]

Created by
Prof. R. Senthilkumar
B. tech and M. Tech
Electronics Engineering
Institute of Road and Transport Technology
College Teacher
Na
Cross-Checked by
Mrs. Phani Swathi Chitta

July 17, 2017

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

Book Description

Title: Digital Signal Processing: Principle, Algorithms And Applications

Author: J. G. Proakis And D. G. Manolakis

Publisher: Prentice Hall Of India, New Delhi

Edition: 3

Year: 1997

ISBN: 81-203-1129-9

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes

Chapter 1

Introduction

Scilab code Exa 1.2.1 Discrete Time Signal

```
1 //Graphical//
2 //Implementation of Equation 1.2.1 in Chapter 1
3 //Digital Signal Processing by Proakis, Third
     Edition, PHI
4 //Page 9
5
6 clear; clc; close;
7 n = 0:10;
8 x = (0.8)^n;
9 //plot2d4(n,x)
10 a=gca();
11 a.thickness = 2;
12 plot2d3('gnn',n,x)
13 xtitle('Graphical Representation of Discrete Time
     Signal','n','x[n]');
```

Chapter 2

Discrete Time Signals and Systems

Scilab code Exa 2.1.2 Signal advanced version

```
1 //Graphical//
2 //Implementation of Eample 2.1.2 in Chapter 2
3 //Digital Signal Processing by Proakis, Third
     Edition, PHI
4 //Page 52
5
6
7 clear;
8 clc;
9 close;
10
11 x = [0 0 0 -1 0 1 2 3 4 5 5 5 5 5];
12
13 x1 =  [0 0 0 x];          //x(n-3)
14
15 x2 =[x 0 0];              //x(n+2)
16
17
18 a=gca();
```

```
19 a.thickness = 2;
20 a.y_location = "middle";
21 a.x_location = "middle"
22
23 subplot(3,1,1)
24 a=gca();
25 a.thickness = 2;
26 a.y_location = "middle";
27 a.x_location = "middle"
28 plot2d3('gnn',1:length(x),x)
29 xtitle('Graphical Representation of signal x','--->
       n','--->x[n]');
30
31 subplot(3,1,2)
32 a=gca();
33 a.thickness = 2;
34 a.y_location = "middle";
35 a.x_location = "middle"
36 plot2d3('gnn',x1)
37 xtitle('Graphical Representation of signal delayed
      version of x','---> n','--->x[n-3]');
38
39 subplot(3,1,3)
40 a=gca();
41 a.thickness = 2;
42 a.y_location = "middle";
43 a.x_location = "middle"
44 plot2d3('gnn',x2)
45 xtitle('Graphical Representation of signal advanced
      version of x','---> n','--->x[n+2]');
```

Scilab code Exa 2.1.6 Unit Sample Sequence

```
1 //Graphical//
2 //Implementation of Equation 2.1.6 in Chapter 2
```

```
3 //Digital Signal Processing by Proakis, Third
     Edition, PHI
4 //Page 45
5
6 clear; clc; close;
7 L = 4;  //Upperlimit
8 n = -L:L;
9 x = [zeros(1,L),1,zeros(1,L)];
10 a=gca();
11 a.thickness = 2;
12 a.y_location = "middle";
13 plot2d3('gnn',n,x)
14 xtitle('Graphical Representation of Unit Sample
     Sequence','n','x[n]');
```

Scilab code Exa 2.1.7 Unit Step Signal

```
1 //Graphical//
2 //Implementation of Equation 2.1.7 in Chapter 2
3 //Digital Signal Processing by Proakis, Third
     Edition, PHI
4 //Page 45
5
6 clear; clc; close;
7 L = 4;  //Upperlimit
8 n = -L:L;
9 x = [zeros(1,L),ones(1,L+1)];
10 a=gca();
11 a.thickness = 2;
12 a.y_location = "middle";
13 plot2d3('gnn',n,x)
14 xtitle('Graphical Representation of Unit Step Signal
     ','n','x[n]');
```

Scilab code Exa 2.1.8 Unit Ramp Signal

```
1 //Graphical//
2 //Implementation of Equation 2.1.8 in Chapter 2
3 //Digital Signal Processing by Proakis, Third
     Edition, PHI
4 //Page 45
5
6 clear; clc; close;
7 L = 4;  //Upperlimit
8 n = -L:L;
9 x = [zeros(1,L),0:L];
10 a=gca();
11 a.thickness = 2;
12 a.y_location = "middle";
13 plot2d3('gnn',n,x)
14 xtitle('Graphical Representation of Unit Ramp Signal
     ','n','x[n]');
```

Scilab code Exa 2.01.09 Exponential Increasing-Decreasing Signal

```
1 //Graphical//
2 //Implementation of Equation 2.01.09b in Chapter 2
3 //Digital Signal Processing by Proakis, Third
     Edition, PHI
4 //Page 46
5 // a < 0
6 clear;
7 clc;
8 close;
9 a =-1.5;
10 n = 0:10;
```

```
11 x = (a)^n;
12 a=gca();
13 a.thickness = 2;
14 a.x_location = "origin";
15 a.y_location = "origin";
16 plot2d3('gnn',n,x)
17 xtitle('Graphical Representation of Exponential
      Increasing-Decreasing Signal','n','x[n]');
```

Scilab code Exa 2.1.09 Exponential Decreasing Signal

```
1 //Graphical//
2 //Implementation of Equation 2.1.09c in Chapter 2
3 //Digital Signal Processing by Proakis, Third
      Edition, PHI
4 //Page 46
5 // a < 1
6 clear;
7 clc;
8 close;
9 a =0.5;
10 n = 0:10;
11 x = (a)^n;
12 a=gca();
13 a.thickness = 2;
14 a.x_location = "middle";
15 plot2d3('gnn',n,x)
16 xtitle('Graphical Representation of Exponential
      Decreasing Signal','n','x[n]');
```

Scilab code Exa 2.1.9 Exponential Signal

```
1 //Graphical//
```

```
2 //Implementation of Equation 2.1.9 in Chapter 2
3 //Digital Signal Processing by Proakis, Third
     Edition, PHI
4 //Page 46
5 clear;
6 clc;
7 close;
8 a =1.5;
9 n =1:10;
10 x = (a)^n;
11 a=gca();
12 a.thickness = 2;
13 plot2d3('gnn',n,x)
14 xtitle('Graphical Representation of Exponential
     Signal','n','x[n]');
```

Scilab code Exa 2.1.24 Even Signal

```
1 //Graphical//
2 //Implementation of Equation 2.1.24 in Chapter 2
3 //Digital Signal Processing by Proakis, Third
     Edition, PHI
4 //Page 51
5
6 clear; clc; close;
7 n = -7:7;
8 x1 = [0 0 0 1 2 3 4];
9 x = [x1,5,x1(length(x1):-1:1)];
10 a=gca();
11 a.thickness = 2;
12 a.y_location = "middle";
13 plot2d3('gnn',n,x)
14 xtitle('Graphical Representation of Even Signal','n'
     ,'x[n]');
```

Scilab code Exa 2.1.25 ODD Signal

```
1 //Graphical//
2 //Implementation of Equation 2.1.25 in Chapter 2
3 //Digital Signal Processing by Proakis, Third
     Edition, PHI
4 //Page 51
5 clear;
6 clc;
7 close;
8 n = -5:5;
9 x1 = [0 1 2 3 4 5];
10 x = [-x1($:-1:2),x1];
11 a=gca();
12 a.thickness = 2;
13 a.y_location = "middle";
14 a.x_location = "middle"
15 plot2d3('gnn',n,x)
16 xtitle('Graphical Representation of ODD Signal','
                n','                x[n]');
```

Chapter 3

The z Transformation and its Applications to the Analysis of LTI Systems

Scilab code Exa 3.1.0 Ztransfer

```
1 //Graphical//
2 function [Ztransfer]=ztransfer_new(sequence,n)
3 z = poly(0, 'z', 'r')
4 Ztransfer=sequence*(1/z)^n'
5 endfunction
```

Scilab code Exa 3.1.1 Z Transform of Finite Duration SIgnals

```
1 //Graphical//
2 //Example 3.1.1
3 //Z Transform of Finite Duration SIgnals
4 clear;
5 clc;
6 close;
```

```
function [Ztransfer]= ztransfer_new (sequence,n)
  z = poly (0, 'z' , 'r' );  Ztransfer = sequence
     *(1/z )^n'
endfunction
x1 = [1,2,5,7,0,1];
n1 = 0:length(x1)-1;
X1 = ztransfer_new(x1,n1)
x2 = [1,2,5,7,0,1];
n2 = -2:3;
X2 = ztransfer_new(x2,n2)
x3 =[0,0,1,2,5,7,0,1];
n3 = 0:length(x3)-1;
X3 = ztransfer_new(x3,n3)
x4 = [2,4,5,7,0,1];
n4 = -2:3;
X4 = ztransfer_new(x4,n4)
x5 = [1,0,0];  //S(n) Unit Impulse sequence
n5 = 0:length(x5)-1;
X5 = ztransfer_new(x5,n5)
x6 = [0,0,0,1];  //S(n-3) unit impulse sequence
   shifted
n6 = 0:length(x6)-1;
X6 = ztransfer_new(x6,n6)
x7 = [1,0,0,0];  //S(n+3) Unit impulse sequence
   shifted
n7 =  -3:0;
X7 = ztransfer_new(x7,n7)
```

Scilab code Exa 3.1.2 Z transform of x[n] = (0.5)^n. u[n]

```
//Graphical//
//Example 3.1.2
//Z transform of x[n] = (0.5)^n. u[n]
clear;
clc;
```

```
close;
syms n z;
x=(0.5)^n
X=symsum(x*(z^(-n)),n,0,%inf)
disp(X,"ans=")
```

Scilab code Exa 3.1.4 Z transform of x[n] = -alpha^n. u[-n-1]

```
//Graphical//
//Example 3.1.4
//Z transform of x[n] = -alpha^n. u[-n-1]
//alpha = 0.5
clear;
close;
clc;
syms n z;
x=-(0.5)^(-n)
X=symsum(x*(z^(n)),n,1,%inf)
disp(X,"ans=")
```

Scilab code Exa 3.1.5 Z transform of x[n] = a^n.u[n]+b^n.u[-n-1]

```
//Graphical//
//Example 3.1.5
//Z transform of x[n] = a^n.u[n]+b^n.u[-n-1]
//a = 0.5 and b = 0.6
clear;
close;
clc;
syms n z;
x1=(0.5)^(n)
X1=symsum(x1*(z^(-n)),n,0,%inf)
x2=(0.6)^(-n)
```

```
12 X2=symsum(x2*(z^(n)),n,1,%inf)
13 X = (X1+X2)
14 disp(X,"ans=")
```

Scilab code Exa 3.2.01 Z transform of x[n] = 3.2^n.u[n]-4.3^n.u[n]

```
1 //Graphical//
2 //Example 3.2.01
3 //Z transform of x[n] = 3.2^n.u[n]-4.3^n.u[n]
4 clear;
5 close;
6 clc;
7 syms n z;
8 x1=(2)^(n)
9 X1=symsum(3*x1*(z^(-n)),n,0,%inf)
10 x2=(3)^(n)
11 X2=symsum(4*x2*(z^(-n)),n,0,%inf)
12 X = (X1-X2)
13 disp(X,"ans=")
```

Scilab code Exa 3.2.1 Z transform of x[n] = 3.2^n.u[n]-4.3^n.u[n]

```
1 //Graphical//
2 //Example 3.2.1
3 //Z transform of x[n] = 3.2^n.u[n]-4.3^n.u[n]
4 clear;
5 close;
6 clc;
7 syms n z;
8 x1=(2)^(n)
9 X1=symsum(3*x1*(z^(-n)),n,0,%inf)
10 x2=(3)^(n)
11 X2=symsum(4*x2*(z^(-n)),n,0,%inf)
```

```
12 X = (X1-X2)
13 disp(X,"ans=")
```

Scilab code Exa 3.2.2 Z transform of x[n] = cos(Wo.n).u[n]

```
1 //Graphical//
2 //Example 3.2.2
3 //Z transform of x[n] = cos(Wo.n).u[n]
4 //Z transform of y[n] = sin(Wo.n).u[n]
5 clear;
6 close;
7 clc;
8 syms n z;
9 Wo =2;
10 x1=exp(sqrt(-1)*Wo*n);
11 X1=symsum(x1*(z^(-n)),n,0,%inf);
12 x2=exp(-sqrt(-1)*Wo*n);
13 X2=symsum(x2*(z^(-n)),n,0,%inf)
14 X =(X1+X2)
15 disp(X,"ans=")
16 Y =(1/(2*sqrt(-1)))*(X1-X2)
17 disp(Y,"ans=")
```

Scilab code Exa 3.2.3 Time Shifting Property of Z-transform

```
1 //Graphical//
2 //Example 3.2.3
3 //Time Shifting Property of Z-transform
4 clear;
5 clc;
6 close;
7 function [Ztransfer]= ztransfer_new (sequence,n)
```

```
8    z = poly (0, 'z' , 'r' );  Ztransfer = sequence
       *(1/z )^n'
9  endfunction
10 x1 = [1,2,5,7,0,1];
11 n1 = 0:length(x1)-1;
12 X1 = ztransfer_new(x1,n1)
13 //x2 = [1,2,5,7,0,1];
14 n2 = 0-2:length(x1)-1-2;
15 X2 = ztransfer_new(x1,n2)
16 //x3 =[0,0,1,2,5,7,0,1];
17 n3 = 0+2:length(x1)-1+2;
18 X3 = ztransfer_new(x1,n3)
```

Scilab code Exa 3.2.4 Z transform of x[n] = u[n]

```
1  //Graphical//
2  //Example 3.2.4
3  //Z transform of x[n] = u[n]
4  clear;
5  clc;
6  close;
7  syms n z;
8  x=(1)^n
9  X=symsum(x*(z^(-n)),n,0,%inf)
10 disp(X,"ans=")
```

Scilab code Exa 3.2.6 Z transform of x[n] = u[-n]

```
1  //Graphical//
2  //Example 3.2.6
3  //Z transform of x[n] = u[-n]
4  clear;
5  clc;
```

```
close;
syms n z;
x=(1)^n
X=symsum(x*(z^(n)),n,0,%inf)
disp(X,"ans=")
```

Scilab code Exa 3.2.7 Z transform of x[n] = n.a^n.u[n]

```
//Graphical//
//Example 3.2.7
//Z transform of x[n] = n.a^n.u[n]
clear;
clc;
close;
syms n z;
x=(1)^n;
X=symsum(x*(z^(-n)),n,0,%inf)
disp(X,"ans=")
Y = diff(X,z)
```

Scilab code Exa 3.2.9 Convolution Property Proof

```
//Graphical//
//Example 3.2.9
//Convolution Property Proof
clear;
clc;
close;
function [Ztransfer]= ztransfer_new (sequence,n)
  z = poly (0, 'z' , 'r' );  Ztransfer = sequence
     *(1/z )^n'
endfunction
x1 = [1,-2,1];
```

```
n1 = 0:length(x1)-1;
X1 = ztransfer_new(x1,n1)
x2 = [1,1,1,1,1,1];
n2 = 0:length(x2)-1;
X2 = ztransfer_new(x2,n2)
X = X1.*X2
```

Scilab code Exa 3.2.10 Correlation Property Proof

```
//Graphical//
//Example 3.2.10
//Correlation Property Proof
syms n z;
x1 = (0.5)^n
X1 = symsum(x1*(z^(-n)),n,0,%inf)
X2 = symsum(x1*(z^(n)),n,0,%inf)
disp(X1,"X1 =")
disp(X2,"X2 =")
X = X1*X2
disp(X,"X=")
//Result
//Which is equivalent to Rxx(Z) = 1/(1-0.5(z+z^-1)
   +(0.5^2))
//i.e for a = 0.5 Rxx(Z) = 1/(1-a(z+z^-1)+(a^2))
```

Chapter 4

Frequency Analysis of Signal and Systems

Scilab code Exa 4.1.2 Continuous Time Fourier Transform

```
1 //Graphical//
2 //Example 4.1.2 Continuous Time Fourier Transform
3 //and Energy Density Function of a Square Waveform
4 // x(t)= A, from -T/2 to T/2
5 clear;
6 clc;
7 close;
8 // Analog Signal
9 A =1;     //Amplitude
10 Dt = 0.005;
11 T = 4;  //Time in seconds
12 t = -T/2:Dt:T/2;
13 for i = 1:length(t)
14   xa(i) = A;
15 end
16 //
17 // Continuous-time Fourier Transform
18 Wmax = 2*%pi*2;          //Analog Frequency = 2Hz
19 K = 4; k = 0:(K/800):K;
```

```
20 W = k*Wmax/K;
21 disp(size(xa))
22 Xa=xa'*exp(-sqrt(-1)*t'*W)*Dt;
23 Xa = real(Xa);
24 W = [-mtlb_fliplr(W), W(2:501)]; // Omega from -Wmax
       to Wmax
25 Xa = [mtlb_fliplr(Xa), Xa(2:501)];
26 ESD = Xa^2;  //Energy Density Spectrum
27 subplot(3,1,1);
28 plot(t,xa);
29 xlabel('t in msec.');
30 ylabel('xa(t)')
31 title('Analog Signal')
32 subplot(3,1,2);
33 plot(W/(2*%pi),Xa);
34 xlabel('Frequency in Hz');
35 ylabel('Xa(jW)')
36 title('Continuous-time Fourier Transform')
37 subplot(3,1,3);
38 plot(W/(2*%pi),ESD);
39 xlabel('Frequency in Hz');
40 ylabel('SXX')
41 title('Energy Density Spectrum')
```

Scilab code Exa 4.02.7 Plotting Discrete Time Fourier Transform

```
1 //Graphical//
2 //Example 4.02.7 Sampling a Nonbandlimited Signal
3 //Plotting Discrete Time Fourier Transform of
4 //Discrete Time Signal x(nT)= exp(-A*T*abs(n))
5 clear;
6 clc;
7 close;
8 // Analog Signal
9 A =1;     //Amplitude
```

```
10 Dt = 0.005;
11 t = -2:Dt:2;
12 //Continuous Time Signal
13 xa = exp(-A*abs(t));
14 //Discrete Time Signal
15 Fs =input('Enter the Sampling Frequency in Hertz');
     //Fs = 1Hz(or)20Hz
16 Ts = 1/Fs;
17 n = -5:1:5;
18 nTs = n*Ts;
19 x = exp(-A*abs(nTs));
20 // Analog Signal reconstruction
21 Dt = 0.005;
22 t = -2:Dt:2;
23 Xa = x *sinc_new(Fs*(ones(length(nTs),1)*t-nTs'*ones
     (1,length(t))));
24 // check
25 error = max(abs(Xa - xa))
26 subplot(2,1,1);
27 a =gca();
28 a.x_location = "origin";
29 a.y_location = "origin";
30 plot(t,xa);
31 xlabel('t in msec.');
32 ylabel('xa(t)')
33 title('Original Analog Signal')
34 subplot(2,1,2);
35 a =gca();
36 a.x_location = "origin";
37 a.y_location = "origin";
38 xlabel('t in msec.');
39 ylabel('xa(t)')
40 title('Reconstructed Signal from x(n) using sinc
     function');
41 plot(t,Xa);
```

---

check Appendix AP 1 for dependency:

`sinc_new4_02_7.sci`

Scilab code Exa 4.2.07 Sampling a Nonbandlimited Signal

```
1  //Graphical//
2  //Example 4.2.07 Sampling a Nonbandlimited Signal
3  //Plotting Discrete Time Fourier Transform of
4  //Discrete Time Signal x(nT)= exp(-A*T*abs(n))
5  clear;
6  clc;
7  close;
8  // Analog Signal
9  A =1;     //Amplitude
10 Dt = 0.005;
11 t = -2:Dt:2;
12 //Continuous Time Signal
13 xa = exp(-A*abs(t));
14 //Discrete Time Signal
15 Fs =input('Enter the Sampling Frequency in Hertz');
     //Fs = 2Hz(or)20Hz
16 Ts = 1/Fs;
17 n = -5:1:5;
18 x = exp(-A*abs(n*Ts));
19 // Discrete-time Signal
20 //Discrete-time Fourier transform
21 K = 500;
22 k = 0:1:K;
23 w = %pi*k/K;
24 X = x * exp(-sqrt(-1)*n'*w);
25 X = real(X);
26 w = [-mtlb_fliplr(w), w(2:K+1)]; // Omega from -w to
     w
27 X = [mtlb_fliplr(X), X(2:K+1)];
28 subplot(3,1,1);
29 plot(t,xa );
30 xlabel('t in msec.');
```

```
31 ylabel('xa(t)')
32 title('Analog Signal')
33 subplot(3,1,2);
34 plot2d3('gnn',n,x )
35 xlabel('Discrete Time n.');
36 ylabel('x(nT)')
37 title('Discrete Signal');
38 subplot(3,1,3);
39 plot(w/(2*%pi),X );
40 xlabel('Frequency in pi units');
41 ylabel('X(w)')
42 title('Discrete-time Fourier Transform')
```

Scilab code Exa 4.2.7 Plotting Continuous Time Fourier Transform

```
1  //Graphical//
2  //Example 4.2.7 Sampling a Nonbandlimited Signal
3  //Plotting Continuous Time Fourier Transform of
4  //Continuous Time Signal x(t)= exp(-A*abs(t))
5  clear;
6  clc;
7  close;
8  // Analog Signal
9  A =1;     //Amplitude
10 Dt = 0.005;
11 t = -2:Dt:2;
12 xa = exp(-A*abs(t));
13 //
14 // Continuous-time Fourier Transform
15 Wmax = 2*%pi*2;          //Analog Frequency = 2Hz
16 K = 4;
17 k = 0:(K/500):K;
18 W = k*Wmax/K;
19 Xa = xa * exp(-sqrt(-1)*t'*W) * Dt;
20 Xa = real(Xa);
```

```
21 W = [-mtlb_fliplr(W), W(2:501)]; // Omega from -Wmax
       to Wmax
22 Xa = [mtlb_fliplr(Xa), Xa(2:501)];
23 subplot(2,1,1);
24 a =gca();
25 a.x_location = "origin";
26 a.y_location = "origin";
27 plot(t,xa);
28 xlabel('t in msec.');
29 ylabel('xa(t)')
30 title('Analog Signal')
31 subplot(2,1,2);
32 a =gca();
33 a.x_location = "origin";
34 a.y_location = "origin";
35 plot(W/(2*%pi),Xa);
36 xlabel('Frequency in Hz');
37 ylabel('Xa(jW)*1000')
38 title('Continuous-time Fourier Transform')
```

Scilab code Exa 4.3.4 Convolution Property Example

```
1 //Graphical//
2 //Example 4.3.4
3 //Convolution Property Example
4 //x1(n)=x2(n)= [1,1,1]
5 clear;
6 clc;
7 close;
8 n =-1:1;
9 x1 = [1,1,1];
10 x2 = x1;
11 //Discrete-time Fourier transform
12 K = 500;
13 k = 0:1:K;
```

```
14 w = %pi*k/K;
15 X1 = x1 * exp(-sqrt(-1)*n'*w);
16 X2 = x2 * exp(-sqrt(-1)*n'*w);
17 w = [-mtlb_fliplr(w), w(2:K+1)]; // Omega from -w to
       w
18 X1 = [mtlb_fliplr(X1), X1(2:K+1)];
19 X2 = [mtlb_fliplr(X2), X2(2:K+1)];
20 Freq_X1 = real(X1);
21 Freq_X2 = real(X2);
22 X = X1.*X2;
23 K1 = length(X)
24 k1 = 0:1:K1;
25 w1 = %pi*k1/K1;
26 w1 = [-2*mtlb_fliplr(w), 2*w];
27 X = [mtlb_fliplr(X), X(1:K1)];
28 Freq_X = real(X);
29 //Inv_X = X.*exp(sqrt(-1)*n'*w)
30 x = convol(x1,x2)
31 //Plotting Magitude Responses
32 figure(1)
33 a =gca();
34 a.x_location = 'middle'
35 a.y_location = 'middle'
36 a.x_label
37 a.y_label
38 plot2d(w/%pi,Freq_X1)
39 x_label =a.x_label
40 y_label = a.y_label
41 x_label.text ='                Frequency in Radians'
42 y_label.text ='                                    X1(w)'
43 //xlabel('Frequency in Radians')
44 //ylabel('X1(w)')
45 title('Frequency Response')
46 figure(2)
47 a =gca();
48 a.x_location = 'middle'
49 a.y_location = 'middle'
50 a.x_label
```

```
51 a.y_label
52 plot2d(w/%pi,Freq_X2)
53 x_label =a.x_label
54 y_label = a.y_label
55 x_label.text ='                                    Frequency
      in Radians'
56 y_label.text ='
                                                      X2(w)'
57 title('Frequency Response')
58 figure(3)
59 a =gca();
60 a.y_location = 'middle'
61 a.x_label
62 a.y_label
63 plot2d(w1/(2*%pi),Freq_X)
64 x_label =a.x_label
65 y_label = a.y_label
66 x_label.text ='                                    Frequency
      in Radians'
67 y_label.text ='
                                                      X(w)'
68 title('Frequency Response')
```

Scilab code Exa 4.4.2 Frequency Response of Three point Moving Average System

```
1 //Graphical//
2 //Example 4.4.2
3 //Frequency Response of Three point Moving Average
     System
4 //y(n)= (1/3)[x(n+1)+x(n)+x(n-1)]
5 //h(n) = [1/3,1/3,1/3]
6 clear;
7 clc;
8 close;
9 //Calculation of Impulse Response
```

```
10 n =-1:1;
11 h = [1/3,1/3,1/3];
12 //Discrete-time Fourier transform
13 K = 500;
14 k = 0:1:K;
15 w = %pi*k/K;
16 H = h * exp(-sqrt(-1)*n'*w);
17 //phasemag used to calculate phase and magnitude in
     dB
18 [Phase_H,m] = phasemag(H);
19 H = abs(H);
20 subplot(2,1,1)
21 plot2d(w/%pi,H)
22 xlabel('Frequency in Radians')
23 ylabel('abs(H)')
24 title('Magnitude Response')
25 subplot(2,1,2)
26 plot2d(w/%pi,Phase_H)
27 xlabel('Frequency in Radians')
28 ylabel('<(H)')
29 title('Phase Response')
```

Scilab code Exa 4.4.4 Frequency Response of First Order Difference Equation

```
1 //Graphical//
2 //Example 4.4.4
3 //Frequency Response of First Order Difference
     Equation
4 //a = 0.9 and b = 1-a
5 //Impulse Response h(n) = b.(a^n).u(n)
6 clear;
7 clc;
8 close;
9 a = input('Enter the constant value of Ist order
     Difference Equation');
```

```
10 b= 1-a;
11 //Calculation of Impulse Response
12 n =0:50;
13 h =b*(a.^n) ;
14 //Discrete-time Fourier transform
15 K = 500;
16 k = 0:1:K;
17 w = %pi*k/K;
18 H = h * exp(-sqrt(-1)*n'*w);
19 //phasemag used to calculate phase and magnitude in
     dB
20 [Phase_H,m] = phasemag(H);
21 H = real(H);
22 subplot(2,1,1)
23 plot2d(w/%pi,H)
24 xlabel('Frequency in Radians')
25 ylabel('abs(H)')
26 title('Magnitude Response')
27 subplot(2,1,2)
28 plot2d(w/%pi,Phase_H)
29 xlabel('Frequency in Radians')
30 ylabel('<(H)')
31 title('Phase Response')
```

Chapter 5

Discrete Fourier Transform its Properties and Applications

Scilab code Exa 5.1.2 Determination of N-point DFT

```
1  //Graphical//
2  //Example 5.1.2
3  //Determination of N-point DFT
4  //Plotting Magnitude and Phase spectrum
5  clear;
6  clc;
7  close;
8  L = 10;  // Length of the sequence
9  N = 10;  // N -point DFT
10 for n =0:L-1
11   x(n+1) = 1;
12 end
13 //Computing DFT and IDFT
14 X = fft(x,-1)
15 x_inv =abs(fft(X,1))
16 //Computing Magnitude and Phase Spectrum
17 //Using DTFT
18 n = 0:L-1;
19 K = 500;
```

```
20 k = 0:1:K;
21 w = 2*%pi*k/K;
22 X_W = x' * exp(-sqrt(-1)*n'*w);
23 Mag_X = abs(X_W);
24 //phasemag used to calculate phase and magnitude in
     dB
25 Phase_X = atan(imag(X_W),real(X_W))
26 subplot(2,1,1)
27 plot2d(w,Mag_X)
28 xlabel('Frequency in Radians')
29 ylabel('abs(X)')
30 title('Magnitude Response')
31 subplot(2,1,2)
32 plot2d(w,Phase_X)
33 xlabel('Frequency in Radians')
34 ylabel('&lt;(X)')
35 title('Phase Response')
```

Scilab code Exa 5.1.3 Finding DFT and IDFT

```
1 //Graphical//
2 //Example 5.1.3
3 //Finding DFT and IDFT
4 clear;
5 clc;
6 close;
7 L = 4;  // Length of the sequence
8 N = 4;  // N-point DFT
9 x = [0,1,2,3];
10 //Computing DFT
11 X = fft(x,-1)
12 //Computing IDFT
13 x_inv = real(fft(X,1))
```

Scilab code Exa 5.2.1 Performing Circular COnvolution

```
1 //Graphical//
2 //Example 5.2.1 and Example 5.2.2
3 //Performing Circular COnvolution
4 //Using DFT
5 clear;
6 clc;
7 close;
8 L = 4; //Length of the Sequence
9 N = 4;  // N-point DFT
10 x1 = [2,1,2,1];
11 x2 = [1,2,3,4];
12 //Computing DFT
13 X1 = fft(x1,-1)
14 X2 = fft(x2,-1)
15 //Multiplication of 2 DFTs
16 X3 = X1.*X2
17 //Circular Convolution Result
18 x3 =abs(fft(X3,1))
```

Scilab code Exa 5.3.1 Performing Linear Filtering (i.e) Linear Convolution

```
1 //Graphical//
2 //Example 5.3.1
3 //Performing Linear Filtering (i.e) Linear
     Convolution
4 //Using DFT
5 clear;
6 clc;
7 close;
8 h = [1,2,3];     //Impulse Response of LTI System
```

```
x = [1,2,2,1];  //Input Response of LTI System
N1 = length(x)
N2 = length(h)
disp('Length of Output Response y(n)')
N = N1+N2-1
//Padding zeros to Make Length of 'h' and 'x'
//Equal to length of output response 'y'
h1 = [h,zeros(1,8-N2)]
x1 = [x,zeros(1,8-N1)]
//Computing DFT
H = fft(h1,-1)
X = fft(x1,-1)
//Multiplication of 2 DFTs
Y = X.*H
//Linear Convolution Result
y =abs(fft(Y,1))
for i =1:8
  if(abs(H(i))<0.0001)
    H(i) =0;
  end
  if(abs(X(i))<0.0001)
    X(i) =0;
  end
  if(abs(y(i))<0.0001)
    y(i) =0;
  end
end
disp(X,'X=')
disp(H,'H=')
disp(y,'Output response using Convolution function')
y = convol(x,h)
```

Scilab code Exa 5.4.1 Effect of Zero Padding

```
//Graphical//
```

```
2  //Example 5.4.1
3  //Effect of Zero Padding
4  clear;
5  clc;
6  close;
7  L = 100;  // Length of the sequence
8  N = 200;  // N-point DFT
9  n = 0:L-1;
10 x = (0.95).^n;
11 //Padding zeros to find N = 200 point DFT
12 x_padd = [x, zeros(1,N-L)];
13 //Computing DFT
14 X = fft(x,-1);
15 X_padd =  fft(x_padd,-1);
16 subplot(2,1,1)
17 plot2d(X)
18 xlabel('K')
19 ylabel('X(k)')
20 title('For L =100 and N =100')
21 subplot(2,1,2)
22 plot2d(X_padd)
23 xlabel('K')
24 ylabel('X(k) zero padded')
25 title('For L =100 and N =200')
```

Chapter 6

Efficient Computation of DFT Fast Fourier Transform and Algorithms

Scilab code Exa 6.4.1 Signal to Quantization Noise Ratio

```
1 //Graphical//
2 //Example 6.4.1
3 //Program to Calculate No.of bits required for given
4 //Signal to Quantization Noise Ratio
5 //in computing DFT
6 clear;
7 clc;
8 close;
9 N = 1024;
10 SQNR = 30;  //SQNR = 30 dB
11 v = log2(N); //number of stages
12 b = (log2(10^(SQNR/10))+2*v)/2;
13 b = ceil(b)
14 disp(b,'The number of bits required rounded to:')
```

Scilab code Exa 6.4.2 Signal to Quantization Noise Ratio

```
1 //Graphical//
2 //Example 6.4.2
3 //Program to Calculate No.of bits required for given
4 //Signal to Quantization Noise Ratio
5 //in FFT algorithm
6 clear;
7 clc;
8 close;
9 N = 1024;
10 SQNR = 30;  //SQNR = 30 dB
11 v = log2(N); //number of stages
12 b = (log2(10^(SQNR/10))+v+1)/2;
13 b = ceil(b)
14 disp(b,'The number of bits required rounded to:')
```

Scilab code Exa 6.8 DFT using DIF-FFT algorithm

```
1 //Graphical//
2 //Exercise 6.8
3 //Program to Calculate DFT using DIF-FFT algorithm
4 //x[n]= 1, 0<=n<=7
5 clear;
6 clc;
7 close;
8 x = [1,1,1,1,1,1,1,1];
9 X = fft(x,-1)
10 //Inverse FFT
11 x_inv = real(fft(X,1))
```

Scilab code Exa 6.11 DFT using DIF-FFT algorithm

```
1  //Graphical//
2  //Exercise 6.11
3  //Program to Calculate DFT using DIF-FFT algorithm
4  //x[n]= [1/2,1/2,1/2,1/2,0,0,0,0]
5  clear;
6  clc;
7  close;
8  x = [1/2,1/2,1/2,1/2,0,0,0,0];
9  X = fft(x,-1)
10 //Inverse FFT
11 x_inv = real(fft(X,1))
```

Chapter 7

Implementation of Discrete Time System

Scilab code Exa 6.4.17 Signal to Quantization Noise Ratio

```
1 //Graphical//
2 //Equation 6.4.17
3 //page492
4 //Program to Calculate  Signal to Quantization Noise
      Ratio
5 //in FFT algorithm
6 clear;
7 clc;
8 close;
9 N = input('Enter the N point FFT value');
10 b = log2(N)
11 Quantization_Noise = (2/3)*(2^(-2*b))
12 Signal_Power = (1/(3*N))
13 SQNR = Signal_Power/Quantization_Noise
14 //RESULT
15 //Enter the N point FFT value 1024
16 // b  =    10.
17 // Quantization_Noise  =   0.0000006
18 // Signal_Power  =   0.0003255
```

```
19 // SQNR   =     512.
20 //-->10*log10(SQNR) =   27.0927
```

Scilab code Exa 7.6.3 Quantization Noise in FIR Filter

```
1 //Graphical//
2 //Example 7.6.3
3 //Program to Calculate  Quantization Noise in FIR
     Filter
4 //For M = 32 and No.of bits = 12
5 clear;
6 clc;
7 close;
8 b = input('Enter the number of bits');
9 M = input('Enter the FIR filter length');
10 disp('Coefficient Quantization Error in FIR Filter')
11 Sigma_e_square = (2^(-2*(b+1)))*M/12
```

Scilab code Exa 7.7.01 Determination of Variance of round-off noise

```
1 //Graphical//
2 //Example 7.7.01
3 //Determination of Variance of round-off noise
4 //at the output of cascade realization
5 //H1(Z) = 1/(1-(1/2)z-1)
6 //H2(Z) = 1/(1-(1/4)z-1)
7 //H(Z) = (2/(1-(1/2)z-1))-(1/(1-(1/4)z-1))
8 clear;
9 clc;
10 close;
11 a1 = (1/2); //pole of first system in cascade
     connection
```

```
12 a2 = (1/4); //ploe of second system in cascade
      connection
13 sigma_e = 1; //quantization noise variance
14 //Noise variance of H1(Z)
15 sigma_2 = (1/(1-a2^2))*sigma_e^2//noise variance of
      second system
16 //Noise variance of H2(Z)
17 sigma_1 = 1/(1-a1^2)*sigma_e^2//noise variance of
      first system
18 //Nosie variance of H(Z)
19 sigma = (((2^2)/(1-a1^2))-((2^2)/(1-a1*a2))+(1/(1-a2
      ^2)))*sigma_e^2
20 noise_variance = sigma+sigma_2 //Total noise
      variance
21 //Result
22 //sigma_2  =    1.0666667
23 //sigma_1  =    1.3333333
24 //sigma    =     1.8285714
25 //noise_variance  =    2.8952381
```

Scilab code Exa 7.7.1 Dead band of First order Recursive System

```
1 //Graphical//
2 //Equation 7.7.1
3 //Program to find Dead band of First order Recursive
     System
4 //y(n) = a y(n-1)+x(n); a = (1/2) and a = (3/4)
5 clear;
6 clc;
7 close;
8 a = input('Enter the constant value of first
      Recursive system');
9 b = 4; //No. of bits used to represent
10 Dead_Band = (2^-b)*[(1/2)*(1/(1-a)),-(1/2)*(1/(1-a))
      ]
```

```
11 //Result
12 //For a = (1/2)
13 //Dead Band = [0.0625  - 0.0625]
14 //For a = (3/4)
15 //Dead Band = [0.125  - 0.125]
```

Chapter 8

Design of Digital Filters

Scilab code Exa 8.2.1 Graphical

```
1  //Graphical//
2  //Example 8.2.1
3  //Design of FIR Filter using Frequecny Sampling
       Technique
4  //Low Pass Filter Design
5  clear;
6  clc;
7  close;
8  M =15;
9  Hr = [1,1,1,1,0.4,0,0,0];
10 for k =1:length(Hr)
11     G(k)=((-1)^(k-1))*Hr(k);
12 end
13 h = zeros(1,M);
14 U = (M-1)/2
15 for n = 1:M
16   h1 = 0;
17   for k = 2:U+1
18     h1 =G(k)*cos((2*%pi/M)*(k-1)*((n-1)+(1/2)))+h1;
19   end
20  h(n) = (1/M)* (G(1)+2*h1);
```

```
21 end
22 h
23 [hzm,fr]=frmag(h,256);
24 hzm_dB = 20*log10(hzm)./max(hzm);
25 figure
26 plot(2*fr,hzm)
27 a=gca();
28 xlabel('Normalized Digital Frequency W');
29 ylabel('Magnitude');
30 title('Frequency Response 0f FIR LPF using Frequency
       Sampling Technique with M = 15 with Cutoff
      Frequency = 0.466')
31 xgrid(2)
32 figure
33 plot(2*fr,hzm_dB)
34 a=gca();
35 xlabel('Normalized Digital Frequency W');
36 ylabel('Magnitude in dB');
37 title('Frequency Response 0f FIR LPF using Frequency
       Sampling Technique with M = 15 with Cutoff
      Frequency = 0.466')
38 xgrid(2)
```

Scilab code Exa 8.2.2 Graphical

```
1 //Graphical//
2 //Example 8.2.2
3 //Design of FIR Filter using Frequecny Sampling
     Technique
4 //Low Pass Filter Design
5 clear;
6 clc;
7 close;
8 M =32;
9 T1 = 0.3789795; //for alpha = 0 (Type I)
```

```
Hr = [1,1,1,1,1,1,T1,0,0,0,0,0,0,0,0,0];
for k =1:length(Hr)
    G(k)=((-1)^(k-1))*Hr(k);
end
h = zeros(1,M);
U = (M-1)/2
for n = 1:M
  h1 = 0;
  for k = 2:U+1
    h1 =G(k)*cos((2*%pi/M)*(k-1)*((n-1)+(1/2)))+h1;
  end
 h(n) = (1/M)* (G(1)+2*h1);
end
h
[hzm,fr]=frmag(h,256);
hzm_dB = 20*log10(hzm)./max(hzm);
figure
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)
figure
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)
```

Scilab code Exa 8.2.3 Graphical

```
1 //Graphical//
2 //Example 8.2.3
3 //Low Pass FIlter of length M = 61
4 //Pass band Edge frequency fp = 0.1 and a Stop edge
    frequency fs = 0.15
5 // Choose the number of cosine functions and create
    a dense grid
6 // in [0,0.1) and [0.15,0.5)
7 //magnitude for pass band = 1 & stop band = 0 (i.e)
    [1 0]
8 //Weighting function =[1 1]
9 clear;
10 clc;
11 close;
12 hn=eqfir(61,[0 .1;.15 .5],[1 0],[1 1]);
13 [hm,fr]=frmag(hn,256);
14 disp('The Filter Coefficients are:')
15 hn
16 figure
17 plot(fr,hm)
18 xlabel('Normalized Digital Frequency fr');
19 ylabel('Magnitude');
20 title('Frequency Response of FIR LPF using REMEZ
    algorithm M=61')
21 figure
22 plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
23 xlabel('Normalized Digital Frequency fr');
24 ylabel('Magnitude in dB');
25 title('Frequency Response of FIR LPF using REMEZ
    algorithm M=61')
```

Scilab code Exa 8.2.4 Graphical

```
1  //Graphical//
2  //Example 8.2.4
3  //Band Pass FIlter of length M = 32
4  //Lower Cutoff frequency fp = 0.2 and Upper Cutoff
       frequency fs = 0.35
5  // Choose the number of cosine functions and create
       a dense grid
6  // in [0,0.1) and [0.2,0.35] and [0.425,0.5]
7  //magnitude for pass band = 1 & stop band = 0 (i.e)
       [0 1 0]
8  //Weighting function =[10 1 10]
9  clear;
10 clc;
11 close;
12 hn = 0;
13 hm = 0;
14 hn=eqfir(32,[0 .1;.2 .35;.425 .5],[0 1 0],[10 1 10])
       ;
15 [hm,fr]=frmag(hn,256);
16 disp('The Filter Coefficients are:')
17 hn
18 figure
19 plot(fr,hm)
20 a =gca();
21 xlabel('Normalized Digital Frequency fr');
22 ylabel('Magnitude');
23 title('Frequency Response of FIR BPF using REMEZ
       algorithm M=32')
24 xgrid(2)
25 figure
26 plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
27 a = gca();
28 xlabel('Normalized Digital Frequency fr');
29 ylabel('Magnitude in dB');
30 title('Frequency Response of FIR BPF using REMEZ
       algorithm M=32')
31 xgrid(2)
```

Scilab code Exa 8.2.5 Graphical

```
1  //Graphical//
2  //Example 8.2.5
3  //Linear Phase FIR Differentiator of length M = 60
4  //Pass Band Edge frequency fp = 0.1
5  clear;
6  clc;
7  close;
8  M =60;
9  tuo = (M/2)-1;
10 Wc = 0.1;
11 h =  zeros(1,M);
12 for n = 1:M
13   if n ~= M/2
14     h(n) =cos((n-1-tuo)*Wc)/(n-1-tuo);
15   end
16 end
17 [hm,fr]=frmag(h,1024);
18 disp('The Filter Coefficients are:')
19 h
20 figure
21 plot(fr,hm/max(hm))
22 a =gca();
23 xlabel('Normalized Digital Frequency fr');
24 ylabel('Magnitude');
25 title('Frequency Response of FIR Differentiator for
     M=60')
26 xgrid(2)
```

Scilab code Exa 8.2.6 Graphical

```
1  //Graphical//
2  //Example 8.2.6
3  // Plotting Hibert Transformer of Length M = 31
4  //Default Window Rectangular Window
5  //Chebyshev approx default parameter = [0 0]
6  clear;
7  clc;
8  close;
9  M =31;//Hibert Transformer Length = 31
10 tuo = (M-1)/2;
11 Wc = %pi;
12 h =  zeros(1,M);
13 for n = 1:M
14    if n ~= ((M-1)/2)+1
15      h(n) =(2/%pi)*(sin((n-1-tuo)*Wc/2)^2)/(n-1-tuo)
           ;
16    end
17 end
18 disp('The Hilbert Coefficients are:')
19 h
20 Rec_Window = ones(1,M);//Rectangular Window
      generation
21 h_Rec = h.*Rec_Window;//Windowing With Rectangular
      window
22 //Hamming Window geneartion
23 for n=1:M
24   hamm_Window(n) = 0.54-0.46*cos(2*%pi*(n-1)/(M-1));
25 end
26 h_hamm = h.*hamm_Window';//Windowing With hamming
      window;
27 //Hilbert Transformer using Rectangular window
28 [hm_Rec,fr]=frmag(h_Rec,1024);
29 hm_Rec_dB = 20*log10(hm_Rec);
30 figure
31 plot(fr,hm_Rec_dB)
32 a =gca();
33 xlabel('Normalized Digital Frequency fr');
34 ylabel('Magnitude');
```

```
35 title('Frequency Response of FIR Hibert Transformer
      using Rectangular window for M=31')
36 xgrid(2)
37 //Hilbert Transformer using Hamming window
38 [hm_hamm,fr]=frmag(h_hamm,1024);
39 disp('The Hilbert Coefficients are:')
40 hm_hamm_dB = 20*log10(hm_hamm);
41 figure
42 plot(fr,hm_hamm_dB)
43 a =gca();
44 xlabel('Normalized Digital Frequency fr');
45 ylabel('Magnitude');
46 title('Frequency Response of FIR Hibert Transformer
      using hamming window for M=31')
47 xgrid(2)
```

Scilab code Exa 8.3.2 Graphical

```
1 //Graphical//
2 //Example 8.3.2
3 //mapping = (z-(z^-1))/T
4 //To convert analog filter into digital filter
5 clear;
6 clc;
7 close;
8 s = poly(0,'s');
9 H = 1/((s+0.1)^2+9)
10 T =1;//Sampling period T = 1 Second
11 z = poly(0,'z');
12 Hz = horner(H,(1/T)*(z-(z^-1)))
```

Scilab code Exa 8.3.4 Graphical

```
1  //Graphical//
2  //Example 8.3.4
3  //Bilinear Transformation
4  //To convert analog filter into digital filter
5  clear;
6  clc;
7  close;
8  s = poly(0,'s');
9  H = (s+0.1)/((s+0.1)^2+16);
10 Omega_Analog = 4;
11 Omega_Digital = %pi/2;
12 //Finding Sampling Period
13 T = (2/Omega_Analog)*(tan(Omega_Digital/2))
14 z = poly(0,'z');
15 Hz = horner(H,(2/T)*((z-1)/(z+1)))
```

Scilab code Exa 8.03.5 Graphical

```
1  //Graphical//
2  //Example 8.03.5
3  //First Order Butterworth Filter
4  //Low Pass Filter
5  clear;
6  clc;
7  close;
8  s = poly(0,'s');
9  Omegac = 0.2*%pi;
10 H = Omegac/(s+Omegac);
11 T =1;//Sampling period T = 1 Second
12 z = poly(0,'z');
13 Hz = horner(H,(2/T)*((z-1)/(z+1)))
14 HW  =frmag(Hz(2),Hz(3),512);
15 W = 0:%pi/511:%pi;
16 plot(W/%pi,HW)
17 a=gca();
```

```
18 a.thickness = 3;
19 a.foreground = 1;
20 a.font_style = 9;
21 xgrid(1)
22 xtitle('Magnitude Response of Single pole LPF Filter
       Cutoff frequency = 0.2*pi','Digital Frequency
   --->','Magnitude');
```

Scilab code Exa 8.3.05 Graphical

```
1  //Graphical//
2  //Example 8.3.05
3  //First Order Butterworth Filter
4  //High Pass Filter
5  //Table 8.13: Using Digital Filter Transformation
6  clear;
7  clc;
8  close;
9  s = poly(0,'s');
10 Omegac = 0.2*%pi;
11 H = Omegac/(s+Omegac);
12 T =1;//Sampling period T = 1 Second
13 z = poly(0,'z');
14 Hz_LPF = horner(H,(2/T)*((z-1)/(z+1)));
15 alpha = -(cos((Omegac+Omegac)/2))/(cos((Omegac-
      Omegac)/2));
16 HZ_HPF=horner(Hz_LPF,-(z+alpha)/(1+alpha*z))
17 HW  =frmag(HZ_HPF(2),HZ_HPF(3),512);
18 W = 0:%pi/511:%pi;
19 plot(W/%pi,HW)
20 a=gca();
21 a.thickness = 3;
22 a.foreground = 1;
23 a.font_style = 9;
24 xgrid(1)
```

```
25 xtitle('Magnitude Response of Single pole HPF Filter
       Cutoff frequency = 0.2*pi','Digital Frequency
     ---&gt;','Magnitude');
```

Scilab code Exa 8.3.5 Graphical

```
1  //Graphical//
2  //Example 8.3.5 Sigle pole analog filter
3  //Bilinear Transformation
4  //To convert analog filter into digital filter
5  clear;
6  clc;
7  close;
8  s = poly(0,'s');
9  Omegac = 0.2*%pi;
10 H = Omegac/(s+Omegac);
11 T =1;//Sampling period T = 1 Second
12 z = poly(0,'z');
13 Hz = horner(H,(2/T)*((z-1)/(z+1)))
14 disp(Hz,'Hz =')
15 HW  =frmag(Hz(2),Hz(3),512);
16 W = 0:%pi/511:%pi;
17 plot(W/%pi,HW)
18 a=gca();
19 a.thickness = 3;
20 a.foreground = 1;
21 a.font_style = 9;
22 xgrid(1)
23 xtitle('Magnitude Response of Single pole LPF Filter
       Cutoff frequency = 0.2*pi','Digital Frequency
     --->','Magnitude');
24 //Result
25 //Hz =
26 //
27 //     0.6283185 + 0.6283185z
```

```
28 //     ------------------------
29 //   - 1.3716815 + 2.6283185z
30 //
31 //-->Hz(3)=Hz(3)/2.6283185
32 // Hz  =
33 //
34 //      0.6283185 + 0.6283185z
35 //     ------------------------
36 //       - 0.5218856 + z
37 //
38 //-->Hz(2)=Hz(2)/2.6283185
39 // Hz  =
40 //
41 //      0.2390572 + 0.2390572z
42 //     ------------------------
43 //       - 0.5218856 + z
44 //
45 //       which is equivalent to
46 //Hz =
47 //
48 //    0.2390572(1 + z^-1)
49 //     ----------------------
50 //      1 - 0.5218856*z^-1
```

Scilab code Exa 08.3.6 Graphical

```
1 //Graphical//
2 //Example 08.3.6
3 // To Design an Analog Butterworth Filter
4 //For the given cutoff frequency Wc = 500 Hz
5 clear;
6 clc;
7 close;
8 omegap =  500;
9 omegas =  1000;
```

```
delta1_in_dB = -3;
delta2_in_dB = -40;
delta1 = 10^(delta1_in_dB/20)
delta2 = 10^(delta2_in_dB/20)
//Calculation of Filter Order
N = log10((1/(delta2^2))-1)/(2*log10(omegas/omegap))
N = ceil(N)
omegac = omegap;
//Poles and Gain Calculation
[pols,gain]=zpbutt(N,omegac);
//Magnitude Response of Analog IIR Butterworth
   Filter
h=buttmag(N,omegac,1:1000);
//Magnitude in dB
mag=20*log10(h);
plot2d((1:1000),mag,[0,-180,1000,20]);
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(5)
xtitle('Magnitude Response of Butterworth LPF Filter
    Cutoff frequency = 500 Hz','Analog frequency in
   Hz---->','Magnitude in dB -->');
```

Scilab code Exa 8.03.06 Graphical

```
//Graphical//
//Example 8.03.06
// To Design an Analog Butterworth Filter
//For the given cutoff frequency Wc = 500 Hz
clear;
clc;
close;
omegap =  500;
```

```
omegas =  1000;
delta1_in_dB = -3;
delta2_in_dB = -40;
delta1 = 10^(delta1_in_dB/20)
delta2 = 10^(delta2_in_dB/20)
//Calculation of Filter Order
N = log10((1/(delta2^2))-1)/(2*log10(omegas/omegap))
N = ceil(N)
omegac = omegap;
//Poles and Gain Calculation
[pols,gain]=zpbutt(N,omegac);
//Magnitude Response of Analog IIR Butterworth
   Filter
h=buttmag(N,omegac,1:1000);
//Magnitude in dB
mag=20*log10(h);
plot2d((1:1000),mag,[0,-180,1000,20]);
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(5)
xtitle('Magnitude Response of Butterworth LPF Filter
    Cutoff frequency = 500 Hz','Analog frequency in
   Hz--->','Magnitude in dB -->');
```

Scilab code Exa 8.03.6 Graphical

```
//Graphical//
//Example 8.03.6
// To Design an Analog Low Pass IIR Butterworth
   Filter
//For the given cutoff frequency Wc = 500 Hz
clear;
clc;
```

```
close;
omegap =  500;
omegas =  1000;
delta1_in_dB = -3;
delta2_in_dB = -40;
delta1 = 10^(delta1_in_dB/20)
delta2 = 10^(delta2_in_dB/20)
//Calculation of Filter Order
N = log10((1/(delta2^2))-1)/(2*log10(omegas/omegap))
N = ceil(N)
omegac = omegap;
//Poles and Gain Calculation
[pols,gain]=zpbutt(N,omegac);
//Magnitude Response of Analog IIR Butterworth
   Filter
h=buttmag(N,omegac,1:1000);
//Magnitude in dB
mag=20*log10(h);
plot2d((1:1000),mag,[0,-180,1000,20]);
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(5)
xtitle('Magnitude Response of Butterworth LPF Filter
    Cutoff frequency = 500 Hz','Analog frequency in
   Hz--->','Magnitude in dB -->');
```

Scilab code Exa 8.3.06 Graphical

```
//Graphical//
//Example 8.3.06
// To Convert LPF to Analog [1].High Pass [2].Band
   Pass [3].Band Stop  IIR Butterworth Filter
//Using Analog Filter Transformations:Refer Table
```

```
    :8.12
5  //For the given cutoff frequency Wc = 500 Hz
6  clear;
7  clc;
8  close;
9  omegap =  500;
10 omegas =  1000;
11 delta1_in_dB = -3;
12 delta2_in_dB = -40;
13 delta1 = 10^(delta1_in_dB/20)
14 delta2 = 10^(delta2_in_dB/20)
15 //Calculation of Filter Order
16 N = log10((1/(delta2^2))-1)/(2*log10(omegas/omegap))
17 N = ceil(N)
18 omegac = omegap;
19 //Poles and Gain Calculation
20 [pols,gain]=zpbutt(N,omegac);
21 N =1;
22 //
23 omega_LPF = omegap;  //Analog LPF Cutoff frequency
24 omega_HPF = omega_LPF;  //Analog HPF Cutoff
     frequency
25 omega2 = 600;    //Upper Cutoff frequency
26 omega1 = 300;     //Lower Cutoff Frequency
27 omega0 = (omega2*omega1);
28 BW =  omega2 -  omega1;  //Bandwidth
29 disp('Analog LPF Transfer function')
30 [hs,pols,zers,gain] = analpf(N,'butt',[0,0],
     omega_LPF)
31 hs_LPF = hs;
32 hs_LPF(2) = hs_LPF(2)/500;
33 hs_LPF(3)= hs_LPF(3)/500;
34 s =poly(0,'s');
35 disp('Analog HPF Transfer function')
36 h_HPF = horner(hs_LPF,omega_LPF*omega_HPF/s)
37 disp('Analog BPF Transfer function')
38 num = (s^2)+omega0
39 den = BW*s
```

```
h_BPF = horner(hs_LPF,omega_LPF*(num/den))
disp('Analog BSF Transfer function')
num = s*BW
den = (s^2)+omega0
h_BSF = horner(hs_LPF,omega_LPF*(num/den))
//Plotting Low Pass Filter Frequency Response
figure
fr=0:.1:1000;
hf=freq(hs_LPF(2),hs_LPF(3),%i*fr);
hm=abs(hf);
plot(fr,hm)
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of LPF Filter Cutoff
   frequency = 500Hz','Analog Frequency---->','
   Magnitude');
//Plotting High Pass Filter Frequency Response
figure
fr=0:.1:1000;
hf_HPF=freq(h_HPF(2),h_HPF(3),%i*fr);
hm_HPF=abs(hf_HPF);
plot(fr,hm_HPF)
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of HPF Filter Cutoff
   frequency = 500Hz','Analog Frequency---->','
   Magnitude');
//Plotting Band Pass Filter Frequency Response
figure
fr=0:.1:1000;
hf_BPF=freq(h_BPF(2),h_BPF(3),%i*fr);
hm_BPF=abs(hf_BPF);
```

```
74 plot(fr,hm_BPF)
75 a=gca();
76 a.thickness = 3;
77 a.foreground = 1;
78 a.font_style = 9;
79 xgrid(1)
80 xtitle('Magnitude Response of BPF Filter Upper
      Cutoff frequency = 600Hz & Lower Cutoff frequency
       = 300Hz','Analog  Frequency---->','Magnitude');
81 //Plotting Band Stop Filter Frequency Response
82 figure
83 fr=0:.1:1000;
84 hf_BSF=freq(h_BSF(2),h_BSF(3),%i*fr);
85 hm=abs(hf_BSF);
86 plot(fr,hf_BSF)
87 a=gca();
88 a.thickness = 3;
89 a.foreground = 1;
90 a.font_style = 9;
91 xgrid(1)
92 xtitle('Magnitude Response of BSF Filter Upper
      Cutoff frequency = 600Hz & Lower Cutoff frequency
       = 300Hz','Analog  Frequency---->','Magnitude');
```

Scilab code Exa 8.3.6 Graphical

```
1 //Graphical//
2 //Example 8.3.6
3 // To Design an Analog Butterworth Filter
4 //For the given cutoff frequency Wc = 500 Hz
5 clear;
6 clc;
7 close;
8 omegap = 2*%pi*500;
9 omegas = 2*%pi*1000;
```

```
delta1_in_dB = -3;
delta2_in_dB = -40;
delta1 = 10^(delta1_in_dB/20)
delta2 = 10^(delta2_in_dB/20)
//Calculation of Filter Order
N = log10((1/(delta2^2))-1)/(2*log10(omegas/omegap))
N = ceil(N)
omegac = omegap;
//Poles and Gain Calculation
[pols,gain]=zpbutt(N,omegac);
disp(N,'Filter order N =')
disp(pols,'Pole positions are pols =')
//Magnitude Response of Analog IIR Butterworth
   Filter
h=buttmag(N,omegac,1:1000);
//Magnitude in dB
mag=20*log10(h);
plot2d((1:1000),mag,[0,-180,1000,20]);
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(5)
xtitle('Magnitude Response of Butterworth LPF Filter
    Cutoff frequency = 500 Hz','Analog frequency in
   Hz--->','Magnitude in dB -->');
//Result
//Filter order N =      7.
//s  =
// column 1 to 3
//  -699.07013+3062.8264i   -1958.751+2456.196i
   -2830.4772+1363.086i
// column 4 to 6
//-3141.5927+3.847D-13i   -2830.4772-1363.086i
   -1958.751-2456.196i
//column 7
//- 699.07013-3062.8264i
```

Scilab code Exa 8.3.7 Graphical

```
1  //Graphical//
2  //Example 8.3.7
3  //To Design an Analog Chebyshev Filter
4  //For the given cutoff frequency = 500 Hz
5  clear;
6  clc;
7  close;
8  omegap = 1000*%pi; //Analog Passband Edge frequency
      in radians/sec
9  omegas = 2000*%pi; //Analog Stop band edge frequency
       in radians/sec
10 delta1_in_dB = -1;
11 delta2_in_dB = -40;
12 delta1 = 10^(delta1_in_dB/20);
13 delta2 = 10^(delta2_in_dB/20);
14 delta = sqrt(((1/delta2)^2)-1)
15 epsilon  = sqrt(((1/delta1)^2)-1)
16 //Calculation of Filter order
17 num = ((sqrt(1-delta2^2))+(sqrt(1-((delta2^2)*(1+
      epsilon^2)))))/(epsilon*delta2)
18 den = (omegas/omegap)+sqrt((omegas/omegap)^2-1)
19 N = log10(num)/log10(den)
20 //N = (acosh(delta/epsilon))/(acosh(omegas/omegap))
21 N = floor(N)
22 //Cutoff frequency
23 omegac = omegap
24 //Calculation of poles and zeros
25 [pols,Gn] = zpch1(N,epsilon,omegap)
26 disp(N,'Filter order N =');
27 disp(pols,'Poles of a type I lowpass Chebyshev
      filter are Sk =')
28 //Analog Filter Transfer Function
```

```
29 h = poly(Gn,'s','coeff')/real(poly(pols,'s'))
30 //Magnitude Response of Chebyshev filter
31 [h2]=cheb1mag(N,omegac,epsilon,1:1000)
32 //Magnitude in dB
33 mag=20*log10(h2);
34 plot2d((1:1000),mag,[0,-180,1000,20]);
35 a=gca();
36 a.thickness = 3;
37 a.foreground = 1;
38 a.font_style = 9;
39 xgrid(5)
40 xtitle('Magnitude Response of Chebyshev Type 1 LPF
   Filter Cutoff frequency = 500 Hz','Analog
   frequency in Hz--->','Magnitude in dB -->');
```

Scilab code Exa 8.4.1 Graphical

```
1 //Graphical//
2 //Caption:Conveting single pole LPF Butterworth
   filter into BPF
3 //Exa8.4.1
4 //page698
5 clc;
6 Op = sym('Op'); //pass band edge frequency of low
   pass filter
7 s = sym('s');
8 Ol = sym('Ol');  //lower cutoff frequency of band
   pass filter
9 Ou = sym('Ou'); //upper cutoff frequency of band
   pass filter
10 s1 = Op*(s^2+Ol*Ou)/(s*(Ou-Ol)); //Analog
   transformation for LPF to BPF
11 H_Lpf = Op/(s+Op); //single pole analog LPF
   Butterworth filter
12 H_Bpf = limit(H_Lpf,s,s1); //analog BPF Butterworth
```

```
    filter
disp(H_Lpf,'H_Lpf =')
disp(H_Bpf,'H_Bpf =')
//Result
//H_Lpf = Op/(s+Op)
//H_Bpf = (Ou-Ol)*s/(s^2+(Ou-Ol)*s+Ol*Ou)
```

Scilab code Exa 8.4.02 Graphical

```
//Graphical//
//Example 8.4.02
//To Design an Digital IIR Butterworth Filter from
   Analog IIR Butterworth Filter
//and to plot its magnitude response
//TRANSFORMATION OF LPF TO BSF USING DIGITAL
   TRANSFORMATION
clear;
clc;
close;
omegaP = 0.2*%pi;
omegaL =  (2/5)*%pi;
omegaU =  (3/5)*%pi;
z=poly(0,'z');
H_LPF = (0.245)*(1+(z^-1))/(1-0.509*(z^-1))
alpha = (cos((omegaU+omegaL)/2)/cos((omegaU-omegaL)
   /2));
k = tan((omegaU - omegaL)/2)*tan(omegaP/2);
NUM =((z^2)-((2*alpha/(1+k))*z)+((1-k)/(1+k)));
DEN = (1-((2*alpha/(1+k))*z)+(((1-k)/(1+k))*(z^2)));
HZ_BPF=horner(H_LPF,NUM/DEN)
HW  =frmag(HZ_BPF(2),HZ_BPF(3),512);
W = 0:%pi/511:%pi;
plot(W/%pi,HW)
a=gca();
a.thickness = 3;
```

```
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of BSF Filter','Digital
    Frequency---->','Magnitude');
```

Scilab code Exa 8.4.2 Graphical

```
//Graphical//
//Example 8.4.2
//To Design an Digital IIR Butterworth Filter from
    Analog IIR Butterworth Filter
//and to plot its magnitude response
//TRANSFORMATION OF LPF TO BPF USING DIGITAL
    TRANSFORMATION
clear;
clc;
close;
omegaP = 0.2*%pi;
omegaL =  (2/5)*%pi;
omegaU =  (3/5)*%pi;
z=poly(0,'z');
H_LPF = (0.245)*(1+(z^-1))/(1-0.509*(z^-1))
alpha = (cos((omegaU+omegaL)/2)/cos((omegaU-omegaL)
    /2));
k = (cos((omegaU - omegaL)/2)/sin((omegaU - omegaL)
    /2))*tan(omegaP/2);
NUM =-((z^2)-((2*alpha*k/(k+1))*z)+((k-1)/(k+1)));
DEN = (1-((2*alpha*k/(k+1))*z)+(((k-1)/(k+1))*(z^2))
    );
HZ_BPF=horner(H_LPF,NUM/DEN)
disp(HZ_BPF,'Digital BPF IIR Filter H(Z)= ')
HW  =frmag(HZ_BPF(2),HZ_BPF(3),512);
W = 0:%pi/511:%pi;
plot(W/%pi,HW)
```

```
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of BPF Filter','Digital
   Frequency---->','Magnitude');
//Result
//Digital BPF IIR Filter H(Z)=
//                                    2             3
                4
//     0.245 - 1.577D-17z - 0.245z + 1.577D-17z +
   1.360D-17z
//
   ------------------------------------------------------------

//                                    2             3
                4
//     - 0.509 + 1.299D-16z - z + 6.438D-17z + 5.551D
   -17z
//
// which is equivalent to
// H(z) =
//
//                          2
//     0.245 - 0 - 0.245z + 0 + 0
//     ---------------------------
//                       2
//     - 0.509 + 0 - z + 0+ 0
//
//H(z) =
//
//                     2
//     0.245- 0.245z
//     --------------
//                2
//     - 0.509-z
//
```

```
54 //H(z) =
55 //
56 //                    -2
57 //     0.245 - 0.245 z
58 //     ---------------
59 //                -2
60 //       0.509 + z
61 //
62 //
```

Scilab code Exa 8.5 Graphical

```
1  //Graphical//
2  //Figure 8.5
3  //Program to generate different window functions
4  clear;
5  close;
6  clc
7  M =61 ;
8  for n = 1:M
9    h_Rect(n) = 1;
10   h_hann(n) = 0.5-0.5*cos(2*%pi*(n-1)/(M-1));
11   h_hamm(n) = 0.54-0.46*cos(2*%pi*(n-1)/(M-1));
12   h_balckmann(n) = 0.42-0.5*cos(2*%pi*n/(M-1))+0.08*
       cos(4*%pi*n/(M-1));
13 end
14 plot2d(1:M,[h_Rect,h_hann,h_hamm,h_balckmann
     ],[2,5,7,9]);
15 legend(['Rectangular Window';'Hanning';'Hamming';'
     Balckmann']);
16 title('Window Functions for Length M = 61')
```

Scilab code Exa 8.6 Graphical

```
1  //Graphical//
2  //Figure 8.6 and Figure 8.7
3  //Program to frequency response of
4  //(1) Hanning window (2)Hamming window for M = 31
       and M = 61
5  clear;
6  close;
7  clc
8  M1 = 31;
9  M2 = 61;
10 for n = 1:M1
11   h_hann_31(n) = 0.5-0.5*cos(2*%pi*(n-1)/(M1-1));
12   h_hamm_31(n) = 0.54-0.46*cos(2*%pi*(n-1)/(M1-1));
13 end
14 for n = 1:M2
15   h_hann_61(n) = 0.5-0.5*cos(2*%pi*(n-1)/(M2-1));
16   h_hamm_61(n) = 0.54-0.46*cos(2*%pi*(n-1)/(M2-1));
17 end
18 subplot(2,1,1)
19 [h_hann_31_M,fr]=frmag(h_hann_31,512);
20 [h_hann_61_M,fr]=frmag(h_hann_61,512);
21 h_hann_31_M = 20*log10(h_hann_31_M./max(h_hann_31_M)
       );
22 h_hann_61_M = = 20*log10(h_hann_61_M./max(
       h_hann_61_M));
23 plot2d(fr,h_hann_31_M,2);
24 plot2d(fr,h_hann_61_M,5);
25 legend(['Length M = 31';'Length M = 61']);
26 title('Frequency Response 0f Hanning window')
27 subplot(2,1,2)
28 [h_hamm_31_M,fr]=frmag(h_hamm_31,512);
29 [h_hamm_61_M,fr]=frmag(h_hamm_61,512);
30 h_hamm_31_M = 20*log10(h_hamm_31_M./max(h_hamm_31_M)
       );
31 h_hamm_61_M = = 20*log10(h_hamm_61_M./max(
       h_hamm_61_M));
32 plot2d(fr,h_hamm_31_M,2);
33 plot2d(fr,h_hamm_61_M,5);
```

```
34 legend(['Length M = 31';'Length M = 61']);
35 title('Frequency Response of Hamming window')
```

Scilab code Exa 1.08 Graphical

```
1  //Graphical//
2  //PROGRAM TO DESIGN AND OBTAIN THE FREQUENCY
      RESPONSE OF FIR FILTER
3  //Band Stop FILTER (or)Band Reject Filter
4  clear;
5  clc;
6  close;
7  M = 11                  //Filter length = 11
8  Wc = [%pi/4,3*%pi/4];          //Digital Cutoff
      frequency
9  Wc2 = Wc(2)
10 Wc1 = Wc(1)
11 Tuo = (M-1)/2       //Center Value
12 hd = zeros(1,M);
13 W = zeros(1,M);
14 for n = 1:11
15  if (n == Tuo+1)
16   hd(n) = 1-((Wc2-Wc1)/%pi);
17  else     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 hd
24 //Rectangular Window
25 for n = 1:M
26   W(n) = 1;
27 end
```

```
28 //Windowing Fitler Coefficients
29 h = hd.*W;
30 disp('Filter Coefficients are')
31 h;
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 M=11')
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 BPF using
      Rectangular window M=11')
```

Scilab code Exa 1.8 Graphical

```
1 //Graphical//
2 //PROGRAM TO DESIGN AND OBTAIN THE FREQUENCY
     RESPONSE OF FIR FILTER
3 //Band PASS FILTER
4 clear;
5 clc;
6 close;
7 M = 11                  //Filter length = 11
8 Wc = [%pi/4,3*%pi/4];          //Digital Cutoff
     frequency
9 Wc2 = Wc(2)
10 Wc1 = Wc(1)
11 Tuo = (M-1)/2       //Center Value
12 hd = zeros(1,M);
```

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

Scilab code Exa 2.8 Graphical

```
1  //Graphical//
2  //PROGRAM TO DESIGN AND OBTAIN THE FREQUENCY
       RESPONSE OF FIR FILTER
3  //HIGH PASS FILTER
4  clear;
5  clc;
6  close;
7  M = 61                  //Filter length = 61
8  Wc = %pi/5;             //Digital Cutoff frequency
9  Tuo = (M-1)/2           //Center Value
10 for n = 1:M
11     if (n == Tuo+1)
12       hd(n) = 1-Wc/%pi;
13     else
14       hd(n) = (sin(%pi*((n-1)-Tuo)) -sin(Wc*((n-1)-
             Tuo)))/(((n-1)-Tuo)*%pi);
15   end
16 end
17 //Rectangular Window
18 for n = 1:M
19   W(n) = 1;
20 end
21 //Windowing Fitler Coefficients
22 h = hd.*W;
23 disp('Filter Coefficients are')
24 h;
25 [hzm,fr]=frmag(h,256);
26 hzm_dB = 20*log10(hzm)./max(hzm);
27 subplot(2,1,1)
28 plot(2*fr,hzm)
29 xlabel('Normalized Digital Frequency W');
30 ylabel('Magnitude');
```

```
31 title('Frequency Response 0f FIR HPF using
      Rectangular window M=61')
32 subplot(2,1,2)
33 plot(2*fr,hzm_dB)
34 xlabel('Normalized Digital Frequency W');
35 ylabel('Magnitude in dB');
36 title('Frequency Response 0f FIR HPF using
      Rectangular window M=61')
```

Scilab code Exa 8.10 Graphical

```
1  //Graphical//
2  //Figure 8.9 and 8.10
3  //PROGRAM TO DESIGN AND OBTAIN THE FREQUENCY
      RESPONSE OF FIR FILTER
4  //LOW PASS FILTER
5  clear;
6  clc;
7  close;
8  M = 61                  //Filter length = 61
9  Wc = %pi/5;             //Digital Cutoff frequency
10 Tuo = (M-1)/2          //Center Value
11 for n = 1:M
12     if (n == Tuo+1)
13       hd(n) = Wc/%pi;
14     else
15       hd(n) =  sin(Wc*((n-1)-Tuo))/(((n-1)-Tuo)*%pi)
            ;
16   end
17 end
18 //Rectangular Window
19 for n = 1:M
20   W(n) = 1;
21 end
22 //Windowing Fitler Coefficients
```

```
23 h = hd.*W;
24 disp('Filter Coefficients are')
25 h;
26 [hzm,fr]=frmag(h,256);
27 hzm_dB = 20*log10(hzm)./max(hzm);
28 subplot(2,1,1)
29 plot(fr,hzm)
30 xlabel('Normalized Digital Frequency W');
31 ylabel('Magnitude');
32 title('Frequency Response 0f FIR LPF using
      Rectangular window M=61')
33 subplot(2,1,2)
34 plot(fr,hzm_dB)
35 xlabel('Normalized Digital Frequency W');
36 ylabel('Magnitude in dB');
37 title('Frequency Response 0f FIR LPF using
      Rectangular window M=61')
```

Chapter 10

Multirate Digital Signal Processing

Scilab code Exa 10.5.1 Frequency Response of DECIMATOR

```
1 //Graphical//
2 //Example 10.5.1
3 //Decimation by 2, Filter Length = 30
4 //Cutoff Frequency Wc = %pi/2
5 //Pass band Edge frequency fp = 0.25 and a Stop band
      edge frequency fs = 0.31
6 // Choose the number of cosine functions and create
     a dense grid
7 // in [0,0.25] and [0.31,0.5]
8 //magnitude for pass band = 1 & stop band = 0 (i.e)
     [1 0]
9 //Weighting function =[2 1]
10 clear;
11 clc;
12 close;
13 M = 30;  //Filter Length
14 D = 2;  //Decimation Factor = 2
15 Wc = %pi/2;  //Cutoff Frequency
16 Wp = Wc/(2*%pi);  //Passband Edge Frequency
```

```
17 Ws = 0.31;  //Stopband Edge Frequency
18 hn=eqfir(M,[0 Wp;Ws .5],[1 0],[2 1]);
19 [hm,fr]=frmag(hn,256);
20 disp('The LPF Filter Coefficients are:')
21 hn
22 //Obtaining Polyphase Filter Coefficients from hn
23 p = zeros(D,M/D);
24 for k = 1:D
25   for n = 1:(length(hn)/D)
26     p(k,n) = hn(D*(n-1)+k);
27   end
28 end
29 disp('The Polyphase Decimator for D =2 are:')
30 p
31 figure
32 plot(fr,hm)
33 xlabel('Normalized Digital Frequency fr');
34 ylabel('Magnitude');
35 title('Frequency Response of FIR LPF using REMEZ
     algorithm M=61')
36 figure
37 plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
38 xlabel('Normalized Digital Frequency fr');
39 ylabel('Magnitude in dB');
40 title('Frequency Response of DECIMATOR (D=2) using
     REMEZ algorithm M=30')
```

Scilab code Exa 10.5.2 Frequency Response of INTERPOLATOR

```
1 //Graphical//
2 //Example 10.5.2
3 //Interpolation by 5, Filter Length = 30
4 //Cutoff Frequency Wc = %pi/5
5 //Pass band Edge frequency fp = 0.1 and a Stop band
     edge frequency fs = 0.16
```

```
6  // Choose the number of cosine functions and create
       a dense grid
7  // in [0,0.1) and [0.16,0.5)
8  //magnitude for pass band = 1 & stop band = 0 (i.e)
       [1 0]
9  //Weighting function =[3 1]
10 clear;
11 clc;
12 close;
13 M = 30;  //Filter Length
14 I = 5;  //Interpolation Factor = 5
15 Wc = %pi/5;  //Cutoff Frequency
16 Wp = Wc/(2*%pi);  //Passband Edge Frequency
17 Ws = 0.16;  //Stopband Edge Frequency
18 hn=eqfir(M,[0 Wp;Ws .5],[1 0],[3 1]);
19 [hm,fr]=frmag(hn,256);
20 disp('The LPF Filter Coefficients are:')
21 hn
22 //Obtaining Polyphase Filter Coefficients from hn
23 p = zeros(I,M/I);
24 for k = 1:I
25   for n = 1:(length(hn)/I)
26     p(k,n) = hn(I*(n-1)+k);
27   end
28 end
29 disp('The Polyphase Interpolator for I =5 are:')
30 p
31 figure
32 plot(fr,hm)
33 xlabel('Normalized Digital Frequency fr');
34 ylabel('Magnitude');
35 title('Frequency Response of FIR LPF using REMEZ
       algorithm M=61')
36 figure
37 plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
38 xlabel('Normalized Digital Frequency fr');
39 ylabel('Magnitude in dB');
40 title('Frequency Response of INTERPOLATOR(I=5) using
```

REMEZ algorithm M=30')

Scilab code Exa 10.6.1 Multistage Implementation of Sampling Rate Conversion

```
1  //Graphical//
2  //Example 10.6.1
3  //Multistage Implementation of Sampling Rate
       Conversion
4  //Decimation factor D = 50
5  //D = D1xD2, D1 = 25, D2 =2
6  clear;
7  clc;
8  close;
9  Fs = 8000;  //Sampling Frequency = 8000Hz
10 Fpc = 75;  //Passband Frequency
11 Fsc = 80;  //Stopband Frequency
12 Delta_F = (Fsc-Fpc)/Fs;  //Transition Band
13 Pass_Band = [0,Fpc];
14 Transition_Band = [Fpc,Fsc];
15 Delta1  = (10^-2);  //Passband Ripple
16 Delta2 = (10^-4);  //Stopband Ripple
17 D = Fs/(2*Fsc);    //Decimation Factor
18 //Decimator Implemented in Two Stages
19 D1 = D/2;  //Decimator 1
20 D2 = 2;  //Decimator 2
21 //Decimator Single Stage Implementation
22 M = ((-10*log10(Delta1*Delta2)-13)/(14.6*Delta_F))
       +1;
23 M = ceil(M)
24 //Decimator Multistage Implementation
25 //First Stage Implementation
26 F1 = Fs/D1;  //New passband for stage1
27 Fsc1 = F1-Fsc;  //New Stopband for stage1
28 Delta_F1 = (Fsc1-Fpc)/Fs  //New Transition for
       stage1
```

```
29 Delta11 = Delta1/2;  //New Passband Ripple
30 Delta21  = Delta2;  //Stopband Ripple same
31 M1 = ((-10*log10(Delta11*Delta21)-13)/(14.6*Delta_F1
   ))+1
32 M1 = floor(M1)
33 //Second Stage Implementation
34 F2 = F1/D2;  //New passband for stage2
35 Fsc2 = F2-Fsc;  //New Stopband for stage2
36 Delta_F2 = (Fsc2-Fpc)/F1  //New Transition for
   stage2
37 Delta12 = Delta1/2;  //New Passband Ripple
38 Delta22  = Delta2;  //Stopband Ripple same
39 M2 = ((-10*log10(Delta12*Delta22)-13)/(14.6*Delta_F2
   ))+1
40 M2 = floor(M2)
41 disp('The Filter length Required in Single stage
   Implementation of Decimator is:')
42 M
43 disp('The Filter length Required in Multistage
   Implementation of Decimator is:')
44 M1+M2
45 //Calculation of Reduction Factor
46 R = M/(M1+M2);
47 disp('The Reduction in Filter Length is:')
48 R
```

Scilab code Exa 10.8.1 Signal to Distortion Ratio

```
1 //Graphical//
2 //Example 10.8.1
3 //Signal to Distortion Ratio
4 //Calculation of no. of subfilters
5 clear;
6 clc;
7 close;
```

```
8  SDR_dB = 50; //Signal to distortion ratio = 50 dB
9  Wx = 0.8*%pi; //Digital maximum frequency of input
       data
10 SDR = 10^(SDR_dB/10)
11 disp('The Number of subfilters required')
12 I = Wx*sqrt(SDR/12);
13 I = ceil(I)
```

Scilab code Exa 10.8.2 Signal to Distortion Ratio using Linear Interpolation

```
1  //Graphical//
2  //Example 10.8.2
3  //Signal to Distortion Ratio using Linear
       Interpolation
4  //Calculation of no. of subfilters
5  clear;
6  clc;
7  close;
8  SDR_dB = 50; //Signal to distortion ratio = 50 dB
9  Wx = 0.8*%pi; //Digital maximum frequency of input
       data
10 SDR = 10^(SDR_dB/10)
11 disp('The Number of subfilters required')
12 I = Wx*((SDR/80)^(1/4));
13 I = ceil(I)
```

Scilab code Exa 10.9.1 Multistage Implementation of Sampling Rate Conversion

```
1  //Graphical//
2  //Example 10.9.1
3  //Multistage Implementation of Sampling Rate
       Conversion
4  //Decimation factor D = 100
```

```
5  //D = D1xD2, D1 = 50, D2 =2
6  //Interpolation factor I = 100
7  //I = I1xI2, I1 = 2, I2 =50
8  clear;
9  clc;
10 close;
11 Fs = 8000;  //Sampling Frequency = 8000Hz
12 Fpc = 75;  //Passband Frequency
13 Fsc = 80;  //Stopband Frequency
14 Delta_F = (Fsc-Fpc)/Fs;  //Transition Band
15 Pass_Band = [0,Fpc];
16 Transition_Band = [Fpc,Fsc];
17 Delta1  = (10^-2);  //Passband Ripple
18 Delta2 = (10^-4);  //Stopband Ripple
19 D = Fs/(2*Fsc);    //Decimation Factor
20 //Decimator Implemented in Two Stages
21 D1 = D/2;  //Decimator 1
22 D2 = 2;  //Decimator 2
23 //Decimator Single Stage Implementation
24 M = ((-10*log10(Delta1*Delta2/2)-13)/(14.6*Delta_F))
      +1;
25 M = ceil(M)
26 //Decimator Multistage Implementation
27 //First Stage Implementation
28 Delta_F1 = 0.020625 //Obtained from Example 10.6.1
29 M1 = ((-10*log10(Delta1*Delta2/4)-13)/(14.6*Delta_F1
      ))+1
30 M1 = floor(M1)
31 //Second Stage Implementation
32 Delta_F2 = 0.015625  //Obtained from Example 10.6.1
33 M2 = ((-10*log10(Delta1*Delta2/4)-13)/(14.6*Delta_F2
      ))+1
34 M2 = floor(M2)
35 disp('The Filter length Required in Single stage
      Implementation of Decimator is:')
36 M
37 disp('The Filter length Required in Multistage
      Implementation of Decimator is:')
```

```
38 M1+M2
39 //Calculation of Reduction Factor
40 R = M/(M1+M2);
41 disp('The Reduction in Filter Length is:')
42 R
```

Chapter 11

Linear Predictions and Optimum Linear Filter

Scilab code Exa 11.6.1 Design of wiener filter

```
1 //Graphical//
2 //Example 11.6.1
3 //Design of wiener filter of Length M =2
4 clear;
5 close;
6 clc;
7 M =2;  //Wiener Filter Length
8 Rdx = [0.6 2 0.6] //Cross correlation matrix between
      the desired input sequence and actual input
     sequence
9 C = Rdx(M:$) //Right sided sequence
10 To_M = toeplitz(C)
11 Rxx = [0.6 1 0.6] //Auto correlation matrix
12 Rss = Rxx(M:$)
13 //Filter coefficients
14 h = [0.451 0.165]
15 //Calculation of Minimum Mean Square Error
16 sigma_d = 1; //Average power of desired sequence
17 MSE = sigma_d - h*Rss'
```

Chapter 12

Power Spectrum Estimation

Scilab code Exa 12.1.1 Determination of spectrum of a signal

```
1  //Graphical//
2  //Example 12.1.1
3  //Determination of spectrum of a signal
4  //With maximum normalized frequency f = 0.1
5  //using Rectangular window and Blackmann window
6  clear;
7  close;
8  clc;
9  N = 61;
10 cfreq = [0.1 0];
11 [wft,wfm,fr]=wfir('lp',N,cfreq,'re',0);
12 wft;                  // Time domain filter
      coefficients
13 wfm;                  // Frequency domain filter
      values
14 fr;                   // Frequency sample points
15 WFM_dB = 20*log10(wfm);//Frequency response in dB
16 for n = 1:N
17  h_balckmann(n)=0.42-0.5*cos(2*%pi*n/(N-1))+0.08*cos
       (4*%pi*n/(N-1));
18 end
```

```
wft_blmn = wft'.*h_balckmann;
wfm_blmn = frmag(wft_blmn,length(fr));
WFM_blmn_dB =20*log10(wfm_blmn);
subplot(2,1,1)
plot2d(fr,WFM_dB)
xtitle('Frequency Response of Rectangular window
   Filtered output M = 61','Frequency in cycles per
   samples  f','Energy density in dB')
subplot(2,1,2)
plot2d(fr,WFM_blmn_dB)
xtitle('Frequency Response of Blackmann window
   Filtered output M = 61','Frequency in cycles per
   samples  f','Energy density in dB')
```

Scilab code Exa 12.1.2 Evaluating power spectrum of a discrete sequence

```
//Graphical//
//Example 12.1.2
//Evaluating power spectrum of a discrete sequence
//Using N-point DFT
clear;
clc;
close;
N =16;  //Number of samples in given sequence
n =0:N-1;
delta_f = [0.06,0.01];//frequency separation
x1 = sin(2*%pi*0.315*n)+cos(2*%pi*(0.315+delta_f(1))
   *n);
x2 = sin(2*%pi*0.315*n)+cos(2*%pi*(0.315+delta_f(2))
   *n);
L = [8,16,32,128];
k1 = 0:L(1)-1;
k2 = 0:L(2)-1;
k3 = 0:L(3)-1;
k4 = 0:L(4)-1;
```

```
fk1 = k1./L(1);
fk2 = k2./L(2);
fk3 = k3./L(3);
fk4 = k4./L(4);
for i =1:length(fk1)
  Pxx1_fk1(i) = 0;
  Pxx2_fk1(i) = 0;
  for m = 1:N
    Pxx1_fk1(i)=Pxx1_fk1(i)+x1(m)*exp(-sqrt(-1)*2*
       %pi*(m-1)*fk1(i));
    Pxx2_fk1(i)=Pxx1_fk1(i)+x1(m)*exp(-sqrt(-1)*2*
       %pi*(m-1)*fk1(i));
  end
  Pxx1_fk1(i) = (Pxx1_fk1(i)^2)/N;
  Pxx2_fk1(i) = (Pxx2_fk1(i)^2)/N;
end
for i =1:length(fk2)
  Pxx1_fk2(i) = 0;
  Pxx2_fk2(i) = 0;
  for m = 1:N
    Pxx1_fk2(i)=Pxx1_fk2(i)+x1(m)*exp(-sqrt(-1)*2*
       %pi*(m-1)*fk2(i));
    Pxx2_fk2(i)=Pxx1_fk2(i)+x1(m)*exp(-sqrt(-1)*2*
       %pi*(m-1)*fk2(i));
  end
  Pxx1_fk2(i) = (Pxx1_fk2(i)^2)/N;
  Pxx2_fk2(i) = (Pxx1_fk2(i)^2)/N;
end
for i =1:length(fk3)
  Pxx1_fk3(i) = 0;
  Pxx2_fk3(i) = 0;
  for m = 1:N
    Pxx1_fk3(i) =Pxx1_fk3(i)+x1(m)*exp(-sqrt(-1)*2*
       %pi*(m-1)*fk3(i));
    Pxx2_fk3(i) =Pxx1_fk3(i)+x1(m)*exp(-sqrt(-1)*2*
       %pi*(m-1)*fk3(i));
  end
  Pxx1_fk3(i) = (Pxx1_fk3(i)^2)/N;
```

```
50    Pxx2_fk3(i) = (Pxx1_fk3(i)^2)/N;
51  end
52  for i =1:length(fk4)
53    Pxx1_fk4(i) = 0;
54    Pxx2_fk4(i) = 0;
55    for m = 1:N
56      Pxx1_fk4(i) =Pxx1_fk4(i)+x1(m)*exp(-sqrt(-1)*2*
          %pi*(m-1)*fk4(i));
57      Pxx2_fk4(i) =Pxx1_fk4(i)+x1(m)*exp(-sqrt(-1)*2*
          %pi*(m-1)*fk4(i));
58    end
59    Pxx1_fk4(i) = (Pxx1_fk4(i)^2)/N;
60    Pxx2_fk4(i) = (Pxx1_fk4(i)^2)/N;
61  end
62  figure
63  title('for frequency separation = 0.06')
64  subplot(2,2,1)
65  plot2d3('gnn',k1,abs(Pxx1_fk1))
66  subplot(2,2,2)
67  plot2d3('gnn',k2,abs(Pxx1_fk2))
68  subplot(2,2,3)
69  plot2d3('gnn',k3,abs(Pxx1_fk3))
70  subplot(2,2,4)
71  plot2d3('gnn',k4,abs(Pxx1_fk4))
72  figure
73  title('for frequency separation = 0.01')
74  subplot(2,2,1)
75  plot2d3('gnn',k1,abs(Pxx2_fk1))
76  subplot(2,2,2)
77  plot2d3('gnn',k2,abs(Pxx2_fk2))
78  subplot(2,2,3)
79  plot2d3('gnn',k3,abs(Pxx2_fk3))
80  subplot(2,2,4)
81  plot2d3('gnn',k4,abs(Pxx2_fk4))
```

Scilab code Exa 12.5.1 Determination of power, frequency and varaince of Additive

```
1 //Graphical//
2 //Example 12.5.1
3 //Determination of power, frequency and varaince of
4 //Additive noise
5 clear;
6 clc;
7 close;
8 ryy = [0,1,3,1,0];//Autocorrelation of signal
9 cen_ter_value = ceil(length(ryy)/2);//center value
    of autocorrelation
10 //Method1
11 //TO find out the variance of the additive Noise
12 C = ryy(ceil(length(ryy)/2):$);
13 corr_matrix = toeplitz(C);//correlation matrix
14 evals =spec(corr_matrix);//Eigen Values computation
15 sigma_w = min(evals);//Minimum of eigen value =
    varinace of noise
16 //Method2
17 //TO find out the variance of the additive Noise
18 P = [1,-sqrt(2),1];//Ploynomial in decreasing order
19 Z = roots(P);//roots of the polynomial
20 P1 = ryy(cen_ter_value+1)/real(Z(1));//power of the
    sinusoid
21 A = sqrt(2*P1);//amplitude of the sinusoid
22 sigma_w1 = ryy(cen_ter_value)-P1;//variance of noise
     method2
23 disp(P1,'Power of the additive noise')
24 f1 = acos(real(Z(1)))/(2*%pi)
25 disp(f1,'frequency of the additive noise')
26 disp(sigma_w1,'Variance of the additive noise')
```

Appendix

Scilab code AP 1 data

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