

Scilab Code for
Digital Signal Processing
Principles, Algorithms and Applications
by J. G. Proakis & D. G. Manolakis¹

Created by
Prof. R. Senthilkumar
Institute of Road and Transport Technology
rsenthil_signalprocess@in.com

Cross-Checked by
Mrs. Phani Swathi Chitta
Research Scholar, IITB
under the guidance of
Prof. Saravanan Vijayakumaran, IIT Bombay
sarva@ee.iitb.ac.in

23 August 2010

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

Book Details

Authors: J. G. Proakis and D. G. Manolakis

Title: Digital Signal Processing

Publisher: Prentice Hall India

Edition: 3rd

Year: 1997

Place: New Delhi

ISBN: 81-203-1129-9

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

Prb Problem (Unsolved problem)

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

ARC Additionally Required Code (Scilab Code that is not part of the above book but required to solve a particular Example)

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

CF Code for Figure(Scilab code that is used for plotting the respective figure of the above book)

For example, Prb 4.56 means Problem 4.56 of the above book. 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	5
1 Introduction	2
1.1 Scilab Code	2
2 Discrete Time Signals and Systems	3
2.1 Scilab Code	3
3 The z Transformation and its Applications to the Analysis of LTI Systems	8
3.1 Scilab Code	8
4 Frequency Analysis of Signal and Systems	15
4.1 Scilab Code	15
5 Discrete Fourier Transform: its Properties and Applications	22
5.1 Scilab Code	22
6 Efficient Computation of DFT: Fast Fourier Transform, Algorithms	27
6.1 Scilab Code	27
7 Implementation of Discrete Time System	30
7.1 Scilab Code	30
8 Design of Digital Filters	33
8.1 Scilab Code	33

10 Multirate Digital Signal Processing	52
10.1 Scilab Code	52
11 Linear Predictions and Optimum Linear Filter	59
11.1 Scilab Code	59
12 Power Spectrum Estimation	60
12.1 Scilab Code	60

List of Scilab Codes

Eqn 1.2.1	Discrete time signal	2
Eqn 2.1.6	Unit sample sequence	3
Eqn 2.1.7	Unit step sequence	3
Eqn 2.1.8	Unit ramp sequence	4
Eqn 2.1.9a	Exponential sequence	4
Eqn 2.1.9b	Exponential increasing sequence	5
Eqn 2.1.9c	Exponential decreasing sequence	5
Eqn 2.1.24	Even signal	6
Eqn 2.1.25	Odd signal	6
Exa 3.1.1	Direct Ztransform	8
Exa 3.1.2	Z transform	9
Exa 3.1.4	Z transform	9
Exa 3.1.5	Z transform	10
Exa 3.2.1	Z transform	10
Exa 3.2.2	Sinusoidal Signals	10
Exa 3.2.3	Time Shifting Property	11
Exa 3.2.4	Z transform	12
Exa 3.2.6	Z transform	12
Exa 3.2.7	Z transform	12
Exa 3.2.9	Convolution	13
Exa 3.2.10	Autocorrelation	13
ARC 3A	Ztransfer	14
Exa 4.1.2	Fourier Transform	15
Exa 4.2.7	Continuous Time Fourier Transform	16
Exa 4.3.4	Convolution Property	17
Exa 4.4.2	Frequency Response	19
Exa 4.4.4	Frequency Response	20
Exa 5.1.2	Discrete Fourier Transform	22

Exa 5.1.3	Discrete Fourier Transform	23
Exa 5.2.1	Example 5.2.1 and 5.2.2	23
Exa 5.3.1	Linear Filtering DFT	24
Exa 5.4.1	Zero Padding	25
Exa 6.4.1	SNR DFT	27
Exa 6.4.2	SNR FFT Algorithm	27
Prb 6.8	FFT Exercise1	28
Prb 6.11	FFT Exercise2	28
Exa 7.6.3	Coeffecient Quantization Noise	30
Eqn 7.7.1	Dead Band	30
Exa 7.7.1	Round off noise variance	31
Eqn 7.7.40	SQNR	32
Exa 8.2.1	Design of Filter	33
Exa 8.2.2	Design of Filter	34
Exa 8.2.3	Remez Algorithm Based	35
Exa 8.2.4	Remez Algorithm Based	36
Exa 8.2.5	FIR Differentiator	37
Exa 8.2.6	Hilbert Transform	38
Eqn 8.2.28	Design of Filter	40
Exa 8.3.2	Analog to Digital	41
Exa 8.3.4	Analog to Digital	41
Exa 8.3.5	Analog to Digital	42
Exa 8.3.6	IIR Filter Design Butterworth Filter	43
Exa 8.3.7	IIR Filter Design	45
Exa 8.4.1	Digital IIR Butterworth Filter	46
Exa 8.4.2	IIR Filter Design Butterworth Filter	47
CF 8.5	Window Functions	49
CF 8.6	Window Functions	49
CF 8.7	Window Functions	50
Exa 10.5.1	FIR Decimation	52
Exa 10.5.2	FIR Interpolation	53
Exa 10.6.1	Sampling Rate Conversion Decimation	54
Exa 10.8.1	Signal Distortion Ratio	56
Exa 10.8.2	Signal Distortion Ratio	56
Exa 10.9.1	Sampling Rate Conversion Decimation Interpolation	57
Exa 11.6.1	Wiener Filter	59
Exa 12.1.1	Spectrum of Signal	60
Exa 12.1.2	Spectrum using DFT	61

Exa 12.5.1 Additive Noise Parameters	63
AE 4.2.7 Discrete Time Fourier Transform	65
ARC 4A sinc	66
AE 4.4.2 Frequency response	66
AE 8.2.28A Design of Filter	67
AE 8.2.28B Design of Filter	68
AE 8.2.28C Design of Filter	70
AE 8.3.5 First Order Butterworth Filter	71
AE 8.3.6 IIR Butterworth Filter	72
AE 8.4.1 First Order Butterworth Filter	73
AE 8.4.2A IIR Filter Design Butterworth Filter	73
AE 8.4.2B IIR Filter Design Butterworth Filter	74

Chapter 1

Introduction

Install Symbolic Toolbox. Refer the spoken tutorial on the link (www.spoken-tutorial.org) for the installation of Symbolic Toolbox.

1.1 Scilab Code

Scilab code Eqn 1.2.1 Discrete time signal as implemented in the book on Page 9

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

Chapter 2

Discrete Time Signals and Systems

2.1 Scilab Code

Scilab code Eqn 2.1.6 Unit sample sequence, also known as unit impulse sequence and delta sequence

```
1 //Implementation of Equation 2.1.6 in Chapter 2
2 //Digital Signal Processing by Proakis, Third
   Edition, PHI
3 //Page 45
4
5 clear; clc; close;
6 L = 4; //Upperlimit
7 n = -L:L;
8 x = [zeros(1,L),1,zeros(1,L)];
9 a=gca();
10 a.thickness = 2;
11 a.y_location = "middle";
12 plot2d3('gnn',n,x)
13 xtitle('Graphical Representation of Unit Sample
   Sequence','n','x[n]');
```

Scilab code Eqn 2.1.7 Unit step sequence

```

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

```

Scilab code Eqn 2.1.8 Unit ramp sequence

```

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

```

Scilab code Eqn 2.1.9a Exponential sequence

```

1 //Implementation of Equation 2.1.9 in Chapter 2

```

```

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

```

Scilab code Eqn 2.1.9b Exponential increasing sequence

```

1 //Implementation of Equation 2.1.9b in Chapter 2
2 //Digital Signal Processing by Proakis , Third
   Edition , PHI
3 //Page 46
4 // a < 0
5 clear;
6 clc;
7 close;
8 a =-1.5;
9 n = 0:10;
10 x = (a)^n;
11 a=gca();
12 a.thickness = 2;
13 a.x_location = "origin";
14 a.y_location = "origin";
15 plot2d3('gmn',n,x)
16 xtitle('Graphical Representation of Exponential
   Increasing-Decreasing Signal', 'n', 'x[n]');

```

Scilab code Eqn 2.1.9c Exponential decreasing sequence

```

1 //Implementation of Equation 2.1.9c in Chapter 2
2 //Digital Signal Processing by Proakis, Third
   Edition, PHI
3 //Page 46
4 // a < 1
5 clear;
6 clc;
7 close;
8 a =0.5;
9 n = 0:10;
10 x = (a)^n;
11 a=gca();
12 a.thickness = 2;
13 a.x_location = "middle";
14 plot2d3('gnn',n,x)
15 xtitle('Graphical Representation of Exponential
   Decreasing Signal', 'n', 'x[n]');

```

Scilab code Eqn 2.1.24 Even signal

```

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

```

Scilab code Eqn 2.1.25 Odd signal

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

Chapter 3

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

3.1 Scilab Code

Scilab code Exa 3.1.1 Z transform of Finite duration signals

```
1 //Example 3.1.1
2 //Z Transform of Finite Duration Signals
3 clear all;
4 clc;
5 close;
6 x1 = [1,2,5,7,0,1];
7 n1 = 0:length(x1)-1;
8 X1 = ztransfer_new(x1,n1)
9 x2 = [1,2,5,7,0,1];
10 n2 = -2:3;
11 X2 = ztransfer_new(x2,n2)
12 x3 = [0,0,1,2,5,7,0,1];
13 n3 = 0:length(x3)-1;
14 X3 = ztransfer_new(x3,n3)
15 x4 = [2,4,5,7,0,1];
16 n4 = -2:3;
```

```

17 X4 = ztransfer_new(x4,n4)
18 x5 = [1,0,0]; //S(n) Unit Impulse sequence
19 n5 = 0:length(x5)-1;
20 X5 = ztransfer_new(x5,n5)
21 x6 = [0,0,0,1]; //S(n-3) unit impulse sequence
    shifted
22 n6 = 0:length(x6)-1;
23 X6 = ztransfer_new(x6,n6)
24 x7 = [1,0,0,0]; //S(n+3) Unit impulse sequence
    shifted
25 n7 = -3:0;
26 X7 = ztransfer_new(x7,n7)

```

*Refer to the following for Scilab code of ztransfer new
[ARC 3A](#)

Scilab code Exa 3.1.2 Z transform of $x(n) = 0.5^n \cdot u(n)$

```

1 //Example 3.1.2
2 //Z transform of x[n] = (0.5)^n. u[n]
3 clear all;
4 clc;
5 close;
6 syms n z;
7 x=(0.5)^n
8 X=symsum(x*(z^(-n)),n,0,%inf)
9 disp(X,"ans=")

```

Scilab code Exa 3.1.4 Z transform of $x(n) = \alpha^n$

```

1 //Example 3.1.4
2 //Z transform of x[n] = -alpha^n. u[-n-1]
3 //alpha = 0.5
4 clear all;
5 close;
6 clc;

```

```

7 syms n z;
8 x=-(0.5)^(-n)
9 X=symsum(x*(z^(n)),n,1,%inf)
10 disp(X,"ans=")

```

Scilab code Exa 3.1.5 Z transform of $x(n) = a^n \cdot u(n) + b^n \cdot u(-n - 1)$

```

1 //Example 3.1.5
2 //Z transform of x[n] = a^n.u[n]+b^n.u[-n-1]
3 //a = 0.5 and b = 0.6
4 clear all;
5 close;
6 clc;
7 syms n z;
8 x1=(0.5)^(n)
9 X1=symsum(x1*(z^(-n)),n,0,%inf)
10 x2=(0.6)^(-n)
11 X2=symsum(x2*(z^(n)),n,1,%inf)
12 X = (X1+X2)
13 disp(X,"ans=")

```

Scilab code Exa 3.2.1 Z transform of $x(n) = 3 \cdot 2^n \cdot u(n) - 4 \cdot 3^n \cdot u(n)$

```

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

```

Scilab code Exa 3.2.2 Z transform of $x(n) = \cos(Wo \cdot n) \cdot u(n)$, $y(n) = \sin(Wo \cdot n) \cdot u(n)$

```

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

```

Scilab code Exa 3.2.3 Time shifting property of Z transform

```

1 //Example 3.2.3
2 //Time Shifting Property of Z-transform
3 clear all;
4 clc;
5 close;
6 x1 = [1,2,5,7,0,1];
7 n1 = 0:length(x1)-1;
8 X1 = ztransfer_new(x1,n1)
9 //x2 = [1,2,5,7,0,1];
10 n2 = 0-2:length(x1)-1-2;
11 X2 = ztransfer_new(x1,n2)
12 //x3 = [0,0,1,2,5,7,0,1];
13 n3 = 0+2:length(x1)-1+2;
14 X3 = ztransfer_new(x1,n3)

```

*Refer to the following for Scilab code of ztransfer new
[ARC 3A](#)

Scilab code Exa 3.2.4 Z transform of $x(n) = u(n)$

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

Scilab code Exa 3.2.6 Z transform of $x(n) = u(-n)$

```
1 //Example 3.2.6
2 //Z transform of x[n] = u[-n]
3 clear all;
4 clc;
5 close;
6 syms n z;
7 x=(1)^n
8 X=symsum(x*(z^(n)),n,0,%inf)
9 disp(X,"ans=")
```

Scilab code Exa 3.2.7 Z transform of $x(n) = n.a^n.u(n)$

```
1 //Example 3.2.7
2 //Z transform of x[n] = n.a^n.u[n]
3 clear all;
4 clc;
5 close;
6 syms n z;
7 x=(1)^n;
8 X=symsum(x*(z^(-n)),n,0,%inf)
9 disp(X,"ans=")
10 Y = diff(X,z)
```

Scilab code Exa 3.2.9 Convolution Property Proof

```
1 //Example 3.2.9
2 //Convolution Property Proof
3 clear all;
4 clc;
5 close;
6 x1 = [1, -2, 1];
7 n1 = 0:length(x1)-1;
8 X1 = ztransfer_new(x1, n1)
9 x2 = [1, 1, 1, 1, 1, 1];
10 n2 = 0:length(x2)-1;
11 X2 = ztransfer_new(x2, n2)
12 X = X1.*X2
```

*Refer to the following for Scilab code of ztransfer new
[ARC 3A](#)

Scilab code Exa 3.2.10 Correlation Property Proof

```
1 //Example 3.2.10
2 //Correlation Property Proof
3 syms n z;
4 x1 = (0.5)^n
5 X1 = symsum(x1*(z^(-n)), n, 0, %inf)
6 X2 = symsum(x1*(z^(n)), n, 0, %inf)
7 disp(X1, "X1 =")
8 disp(X2, "X2 =")
9 X = X1*X2
10 disp(X, "X=")
11 //Result
12 //Which is equivalent to  $R_{xx}(Z) = 1/(1-0.5(z+z^{-1})+(0.5^2))$ 
13 //i.e for  $a = 0.5$   $R_{xx}(Z) = 1/(1-a(z+z^{-1})+(a^2))$ 
```

*Refer to the following for Scilab code of ztransfer new
[ARC 3A](#)

Scilab code ARC 3A Ztarnsfer of a sequence

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

Chapter 4

Frequency Analysis of Signal and Systems

4.1 Scilab Code

Scilab code Exa 4.1.2 Continuous time Fourier transform and Energy Density Function of Square waveform

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

```

18 K = 4; k = 0:(K/800):K;
19 W = k*Wmax/K;
20 disp(size(xa))
21 Xa=xa'*exp(-sqrt(-1)*t'*W)*Dt;
22 Xa = real(Xa);
23 W = [-mtlbfliplr(W), W(2:501)]; // Omega from -Wmax
    to Wmax
24 Xa = [mtlbfliplr(Xa), Xa(2:501)];
25 ESD = Xa^2; //Energy Density Spectrum
26 subplot(3,1,1);
27 plot(t,xa);
28 xlabel('t in msec. ');
29 ylabel('xa(t)')
30 title('Analog Signal')
31 subplot(3,1,2);
32 plot(W/(2*pi),Xa);
33 xlabel('Frequency in Hz');
34 ylabel('Xa(jW)')
35 title('Continuous-time Fourier Transform')
36 subplot(3,1,3);
37 plot(W/(2*pi),ESD);
38 xlabel('Frequency in Hz');
39 ylabel('SXX')
40 title('Energy Density Spectrum')

```

Scilab code Exa 4.2.7 Sampling a Nonbandlimited Signal

```

1 //Example 4.2.7 Sampling a Nonbandlimited Signal
2 //Plotting Continuous Time Fourier Transform of
3 //Continuous Time Signal x(t)= exp(-A*abs(t))
4 clear all;
5 clc;
6 close;
7 // Analog Signal
8 A =1; //Amplitude
9 Dt = 0.005;
10 t = -2:Dt:2;
11 xa = exp(-A*abs(t));

```

```

12 //
13 // Continuous-time Fourier Transform
14 Wmax = 2*pi*2;           //Analog Frequency = 2Hz
15 K = 4;
16 k = 0:(K/500):K;
17 W = k*Wmax/K;
18 Xa = xa * exp(-sqrt(-1)*t'*W) * Dt;
19 Xa = real(Xa);
20 W = [-mtlbfliplr(W), W(2:501)]; // Omega from -Wmax
    to Wmax
21 Xa = [mtlbfliplr(Xa), Xa(2:501)];
22 subplot(2,1,1);
23 a =gca();
24 a.x_location = "origin";
25 a.y_location = "origin";
26 plot(t,xa);
27 xlabel('t in msec. ');
28 ylabel('xa(t)')
29 title('Analog Signal')
30 subplot(2,1,2);
31 a =gca();
32 a.x_location = "origin";
33 a.y_location = "origin";
34 plot(W/(2*pi),Xa);
35 xlabel('Frequency in Hz');
36 ylabel('Xa(jW)*1000')
37 title('Continuous-time Fourier Transform')

```

*For further extension of the exapmle refer to
[AE 4.2.7](#)

Scilab code Exa 4.3.4 Convolution Property Example $x1(n) = x2(n) = [1, 1, 1]$

```

1 //Example 4.3.4
2 //Convolution Property Example

```

```

3 //x1(n)=x2(n)= [1,1,1]
4 clear all;
5 clc;
6 close;
7 n =-1:1;
8 x1 = [1,1,1];
9 x2 = x1;
10 //Discrete-time Fourier transform
11 K = 500;
12 k = 0:1:K;
13 w = %pi*k/K;
14 X1 = x1 * exp(-sqrt(-1)*n'*w);
15 X2 = x2 * exp(-sqrt(-1)*n'*w);
16 w = [-mtlbfliplr(w), w(2:K+1)]; // Omega from -w to
    w
17 X1 = [mtlbfliplr(X1), X1(2:K+1)];
18 X2 = [mtlbfliplr(X2), X2(2:K+1)];
19 Freq_X1 = real(X1);
20 Freq_X2 = real(X2);
21 X = X1.*X2;
22 K1 = length(X)
23 k1 = 0:1:K1;
24 w1 = %pi*k1/K1;
25 w1 = [-2*mtlbfliplr(w), 2*w];
26 X = [mtlbfliplr(X), X(1:K1)];
27 Freq_X = real(X);
28 //Inv_X = X.*exp(sqrt(-1)*n'*w)
29 x = convol(x1,x2)
30 //Plotting Magitude Responses
31 figure(1)
32 a =gca();
33 a.x_location = 'middle'
34 a.y_location = 'middle'
35 a.x_label
36 a.y_label
37 plot2d(w/%pi,Freq_X1)
38 x_label =a.x_label
39 y_label = a.y_label

```

```

40 x_label.text = '          Frequency in Radians '
41 y_label.text = '                                X1(w) '
42 //xlabel('Frequency in Radians ')
43 //ylabel('X1(w) ')
44 title('Frequency Response')
45 figure(2)
46 a =gca();
47 a.x_location = 'middle'
48 a.y_location = 'middle'
49 a.x_label
50 a.y_label
51 plot2d(w/%pi,Freq_X2)
52 x_label =a.x_label
53 y_label = a.y_label
54 x_label.text = '          Frequency
    in Radians '
55 y_label.text = '                                X2(w) '

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

67 title('Frequency Response')

```

Scilab code Exa 4.4.2 Frequency Response of Three point Moving Average System $y(n) = (1/3)[x(n+1) + x(n) + x(n-1)]$

1 //Example 4.4.2

```

2 //Frequency Response of Three point Moving Average
  System
3 //y(n)= (1/3) [x(n+1)+x(n)+x(n-1)]
4 //h(n) = [1/3,1/3,1/3]
5 clear all;
6 clc;
7 close;
8 //Calculation of Impulse Response
9 n =-1:1;
10 h = [1/3,1/3,1/3];
11 //Discrete-time Fourier transform
12 K = 500;
13 k = 0:1:K;
14 w = %pi*k/K;
15 H = h * exp(-sqrt(-1)*n'*w);
16 //phasemag used to calculate phase and magnitude in
  dB
17 [Phase_H,m] = phasemag(H);
18 H = abs(H);
19 subplot(2,1,1)
20 plot2d(w/%pi,H)
21 xlabel('Frequency in Radians')
22 ylabel('abs(H)')
23 title('Magnitude Response')
24 subplot(2,1,2)
25 plot2d(w/%pi,Phase_H)
26 xlabel('Frequency in Radians')
27 ylabel('<(H)')
28 title('Phase Response')

```

*For further extension of the exapmle refer to
[AE 4.4.2](#)

Scilab code Exa 4.4.4 Frequency Response of First order Difference Equation

```

1 //Example 4.4.4
2 //Frequency Response of First Order Difference
   Equation
3 //a = 0.9 and b = 1-a
4 //Impulse Response  $h(n) = b \cdot (a^n) \cdot u(n)$ 
5 clear all;
6 clc;
7 close;
8 a = input('Enter the constant value of 1st order
   Difference Equation');
9 b= 1-a;
10 //Calculation of Impulse Response
11 n =0:50;
12 h =b*(a.^n) ;
13 //Discrete-time Fourier transform
14 K = 500;
15 k = 0:1:K;
16 w = %pi*k/K;
17 H = h * exp(-sqrt(-1)*n'*w);
18 //phasemag used to calculate phase and magnitude in
   dB
19 [Phase_H,m] = phasemag(H);
20 H = real(H);
21 subplot(2,1,1)
22 plot2d(w/%pi,H)
23 xlabel('Frequency in Radians')
24 ylabel('abs(H)')
25 title('Magnitude Response')
26 subplot(2,1,2)
27 plot2d(w/%pi,Phase_H)
28 xlabel('Frequency in Radians')
29 ylabel('<(H)')
30 title('Phase Response')

```

Chapter 5

Discrete Fourier Transform: its Properties and Applications

5.1 Scilab Code

Scilab code Exa 5.1.2 Determination of N-point DFT

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

```

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

```

Scilab code Exa 5.1.3 Finding DFT and IDFT

```

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

```

Scilab code Exa 5.2.1 Performing Circular CONvolution Using DFT

```

1 //Example 5.2.1 and Example 5.2.2
2 //Performing Circular CONvolution
3 //Using DFT

```

```

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

```

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

```

1 //Example 5.3.1
2 //Performing Linear Filtering (i.e) Linear
   Convolution
3 //Using DFT
4 clear all;
5 clc;
6 close;
7 h = [1,2,3]; //Impulse Response of LTI System
8 x = [1,2,2,1]; //Input Response of LTI System
9 N1 = length(x)
10 N2 = length(h)
11 disp('Length of Output Response y(n)')
12 N = N1+N2-1
13 //Padding zeros to Make Length of 'h' and 'x'
14 //Equal to length of output response 'y'
15 h1 = [h,zeros(1,8-N2)]
16 x1 = [x,zeros(1,8-N1)]
17 //Computing DFT
18 H = dft(h1,-1)
19 X = dft(x1,-1)

```

```

20 // Multiplication of 2 DFTs
21 Y = X.*H
22 // Linear Convolution Result
23 y =abs(dft(Y,1))
24 for i =1:8
25     if(abs(H(i))<0.0001)
26         H(i) =0;
27     end
28     if(abs(X(i))<0.0001)
29         X(i) =0;
30     end
31     if(abs(y(i))<0.0001)
32         y(i) =0;
33     end
34 end
35 disp(X, 'X=')
36 disp(H, 'H=')
37 disp(y, 'Output response using Convolution function')
38 y = convol(x,h)

```

Scilab code Exa 5.4.1 Effect of Zero padding

```

1 //Example 5.4.1
2 //Effect of Zero Padding
3 clear all;
4 clc;
5 close;
6 L = 100; // Length of the sequence
7 N = 200; // N -point DFT
8 n = 0:L-1;
9 x = (0.95).^n;
10 //Padding zeros to find N = 200 point DFT
11 x_padd = [x, zeros(1,N-L)];
12 //Computing DFT
13 X = dft(x, -1);
14 X_padd = dft(x_padd, -1);
15 subplot(2,1,1)
16 plot2d(X)

```

```
17 xlabel('K')
18 ylabel('X(k)')
19 title('For L =100 and N =100')
20 subplot(2,1,2)
21 plot2d(X_padd)
22 xlabel('K')
23 ylabel('X(k) zero padded')
24 title('For L =100 and N =200')
```

Chapter 6

Efficient Computation of DFT: Fast Fourier Transform, Algorithms

6.1 Scilab Code

Scilab code Exa 6.4.1 Calculation of No.of bits required for given Signal to Quantization Noise Ratio in DFT

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

Scilab code Exa 6.4.2 Calculation of No.of bits required for given Signal to Quantization Noise Ratio in FFT algorithm

```

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

```

Scilab code Prb 6.8 Program to Calculate DFT using DIF-FFT algorithm

```

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

```

Scilab code Prb 6.11 Program to Calculate DFT using DIF-FFT algorithm

```

1 //Exercise 6.11
2 //Program to Calculate DFT using DIF-FFT algorithm
3 //x[n]= [1/2,1/2,1/2,1/2,0,0,0,0]
4 clear all;
5 clc;
6 close;
7 x = [1/2,1/2,1/2,1/2,0,0,0,0];
8 X = fft(x,-1)

```

```
9 //Inverse FFT
10 x_inv = real(fft(X,1))
```

Chapter 7

Implementation of Discrete Time System

7.1 Scilab Code

Scilab code Exa 7.6.3 Program to Calculate Quantization Noise in FIR Filter For $M = 32$ and No.of bits = 12

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

Scilab code Eqn 7.7.1 Program to find Dead band of First order Recursive System $y(n) = ay(n - 1) + x(n)$; $a = (1/2)$ and $a = (3/4)$

```
1 //Equation 7.7.1
2 //Program to find Dead band of First order Recursive
  System
```

```

3 //y(n) = a y(n-1)+x(n); a = (1/2) and a = (3/4)
4 clear all;
5 clc;
6 close;
7 a = input('Enter the constant value of first
Recursive system ');
8 b = 4; //No. of bits used to represent
9 Dead_Band = (2^-b)*[(1/2)*(1/(1-a)), -(1/2)*(1/(1-a))
]
10 //Result
11 //For a = (1/2)
12 //Dead Band = [0.0625 - 0.0625]
13 //For a = (3/4)
14 //Dead Band = [0.125 - 0.125]

```

Scilab code Exa 7.7.1 Determination of Variance of round-off noise at the output of cascade realization

```

1 //Example 7.7.1
2 //Determination of Variance of round-off noise
3 //at the output of cascade realization
4 //H1(Z) = 1/(1-(1/2)z-1)
5 //H2(Z) = 1/(1-(1/4)z-1)
6 //H(Z) = (2/(1-(1/2)z-1)) - (1/(1-(1/4)z-1))
7 clear all;
8 clc;
9 close;
10 a1 = (1/2); //pole of first system in cascade
connection
11 a2 = (1/4); //ploer of second system in cascade
connection
12 sigma_e = 1; //quantization noise variance
13 //Noise variance of H1(Z)
14 sigma_2 = (1/(1-a2^2))*sigma_e^2//noise variance of
second system
15 //Noise variance of H2(Z)
16 sigma_1 = 1/(1-a1^2)*sigma_e^2//noise variance of
first system

```

```

17 //Noise variance of H(Z)
18 sigma = (((2^2)/(1-a1^2))-((2^2)/(1-a1*a2))+(1/(1-a2
    ^2)))*sigma_e^2
19 noise_variance = sigma+sigma_2 //Total noise
    variance
20 //Result
21 //sigma_2 = 1.0666667
22 //sigma_1 = 1.3333333
23 //sigma = 1.8285714
24 //noise_variance = 2.8952381

```

Scilab code Eqn 7.7.40 Signal to Quantization Noise Ratio

```

1 //Equation 6.4.17
2 //page492
3 //Program to Calculate Signal to Quantization Noise
    Ratio
4 //in FFT algorithm
5 clear all;
6 clc;
7 close;
8 N = input('Enter the N point FFT value');
9 b = log2(N)
10 Quantization_Noise = (2/3)*(2^(-2*b))
11 Signal_Power = (1/(3*N))
12 SQNR = Signal_Power/Quantization_Noise
13 //RESULT
14 //Enter the N point FFT value 1024
15 // b = 10.
16 // Quantization_Noise = 0.0000006
17 // Signal_Power = 0.0003255
18 // SQNR = 512.
19 //-->10*log10(SQNR) = 27.0927

```

Chapter 8

Design of Digital Filters

8.1 Scilab Code

Scilab code Exa 8.2.1 Design of FIR Filter using Frequency Sampling Technique

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

```

19  h(n) = (1/M)* (G(1)+2*h1);
20  end
21  h
22  [hzm,fr]=frmag(h,256);
23  hzm_dB = 20*log10(hzm)./max(hzm);
24  figure
25  plot(2*fr,hzm)
26  a=gca();
27  xlabel('Normalized Digital Frequency W');
28  ylabel('Magnitude');
29  title('Frequency Response Of FIR LPF using Frequency
        Sampling Technique with M = 15 with Cutoff
        Frequency = 0.466 ')
30  xgrid(2)
31  figure
32  plot(2*fr,hzm_dB)
33  a=gca();
34  xlabel('Normalized Digital Frequency W');
35  ylabel('Magnitude in dB');
36  title('Frequency Response Of FIR LPF using Frequency
        Sampling Technique with M = 15 with Cutoff
        Frequency = 0.466 ')
37  xgrid(2)

```

Scilab code Exa 8.2.2 Design of FIR Filter using Frequency Sampling Technique

```

1  //Example 8.2.2
2  //Design of FIR Filter using Frequency Sampling
   Technique
3  //Low Pass Filter Design
4  clear all;
5  clc;
6  close;
7  M =32;
8  T1 = 0.3789795; //for alpha = 0 (Type I)
9  Hr = [1,1,1,1,1,1,1,T1,0,0,0,0,0,0,0,0];
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 Of 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 Of FIR LPF using Frequency
        Sampling Technique with M = 15 with Cutoff
        Frequency = 0.466')
38 xgrid(2)

```

Scilab code Exa 8.2.3 Low Pass Filter

```

1 //Example 8.2.3
2 //Low Pass Filter of length M = 61

```

```

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

```

Scilab code Exa 8.2.4 Band Pass Filter

```

1 //Example 8.2.4
2 //Band Pass Filter of length M = 32
3 //Lower Cutoff frequency fp = 0.2 and Upper Cutoff
  frequency fs = 0.35
4 // Choose the number of cosine functions and create
  a dense grid
5 // in [0,0.1) and [0.2,0.35] and [0.425,0.5]

```

```

6 //magnitude for pass band = 1 & stop band = 0 (i.e)
  [0 1 0]
7 //Weighting function =[10 1 10]
8 clear all;
9 clc;
10 close;
11 hn = 0;
12 hm = 0;
13 hn=eqfir(32,[0 .1;.2 .35;.425 .5],[0 1 0],[10 1 10])
  ;
14 [hm,fr]=frmag(hn,256);
15 disp('The Filter Coefficients are:')
16 hn
17 figure
18 plot(fr,hm)
19 a =gca();
20 xlabel('Normalized Digital Frequency fr');
21 ylabel('Magnitude');
22 title('Frequency Response of FIR BPF using REMEZ
  algorithm M=32')
23 xgrid(2)
24 figure
25 plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
26 a = gca();
27 xlabel('Normalized Digital Frequency fr');
28 ylabel('Magnitude in dB');
29 title('Frequency Response of FIR BPF using REMEZ
  algorithm M=32')
30 xgrid(2)

```

Scilab code Exa 8.2.5 Linear Phase FIR Differentiator of length $M = 60$

```

1 //Example 8.2.5
2 //Linear Phase FIR Differentiator of length M = 60
3 //Pass Band Edge frequency fp = 0.1
4 clear all;
5 clc;
6 close;

```

```

7 M =60;
8 tuo = (M/2)-1;
9 Wc = 0.1;
10 h = zeros(1,M);
11 for n = 1:M
12     if n ~= M/2
13         h(n) =cos((n-1-tuo)*Wc)/(n-1-tuo);
14     end
15 end
16 [hm,fr]=frmag(h,1024);
17 disp('The Filter Coefficients are:')
18 h
19 figure
20 plot(fr,hm/max(hm))
21 a =gca();
22 xlabel('Normalized Digital Frequency fr');
23 ylabel('Magnitude');
24 title('Frequency Response of FIR Differentiator for
        M=60')
25 xgrid(2)

```

Scilab code Exa 8.2.6 Hilbert Transform of Length $M = 31$

```

1 //Example 8.2.6
2 // Plotting Hibert Transformer of Length M = 31
3 //Default Window Rectangular Window
4 //Chebyshev approx default parameter = [0 0]
5 clear all;
6 clc;
7 close;
8 M =31;//Hibert Transformer Length = 31
9 tuo = (M-1)/2;
10 Wc = %pi;
11 h = zeros(1,M);
12 for n = 1:M
13     if n ~= ((M-1)/2)+1
14         h(n) =(2/%pi)*(sin((n-1-tuo)*Wc/2)^2)/(n-1-tuo)
15         ;

```

```

15     end
16 end
17 disp('The Hilbert Coefficients are:')
18 h
19 Rec_Window = ones(1,M); //Rectangular Window
    generation
20 h_Rec = h.*Rec_Window; //Windowing With Rectangular
    window
21 //Hamming Window geneartion
22 for n=1:M
23     hamm_Window(n) = 0.54-0.46*cos(2*%pi*(n-1)/(M-1));
24 end
25 h_hamm = h.*hamm_Window'; //Windowing With hamming
    window;
26 //Hilbert Transformer using Rectangular window
27 [hm_Rec ,fr]=frmag(h_Rec ,1024);
28 hm_Rec_dB = 20*log10(hm_Rec);
29 figure
30 plot(fr ,hm_Rec_dB)
31 a =gca();
32 xlabel('Normalized Digital Frequency fr');
33 ylabel('Magnitude');
34 title('Frequency Response of FIR Hibert Transformer
    using Rectangular window for M=31')
35 xgrid(2)
36 //Hilbert Transformer using Hamming window
37 [hm_hamm ,fr]=frmag(h_hamm ,1024);
38 disp('The Hilbert Coefficients are:')
39 hm_hamm_dB = 20*log10(hm_hamm);
40 figure
41 plot(fr ,hm_hamm_dB)
42 a =gca();
43 xlabel('Normalized Digital Frequency fr');
44 ylabel('Magnitude');
45 title('Frequency Response of FIR Hibert Transformer
    using hamming window for M=31')
46 xgrid(2)

```

Scilab code Eqn 8.2.28 DESIGN AND OBTAIN THE FREQUENCY RESPONSE OF FIR FILTER LowPass

```
1 //Figure 8.9 and 8.10
2 //PROGRAM TO DESIGN AND OBTAIN THE FREQUENCY
  RESPONSE OF FIR FILTER
3 //LOW PASS FILTER
4 clear all;
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) = Wc/%pi;
13     else
14         hd(n) = sin(Wc*((n-1)-Tuo))/(((n-1)-Tuo)*%pi)
15         ;
16     end
17 //Rectangular Window
18 for n = 1:M
19     W(n) = 1;
20 end
21 //Windowing Filter 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(fr,hzm)
29 xlabel('Normalized Digital Frequency W');
30 ylabel('Magnitude');
31 title('Frequency Response of FIR LPF using
  Rectangular window M=61');
32 subplot(2,1,2)
```

```

33 plot(fr,hzm_dB)
34 xlabel('Normalized Digital Frequency W');
35 ylabel('Magnitude in dB');
36 title('Frequency Response Of FIR LPF using
        Rectangular window M=61')

```

*For further extension of the exapmle refer to
[AE 8.2.28A](#) [AE 8.2.28B](#) [AE 8.2.28C](#)

Scilab code Exa 8.3.2 Backward Difference

```

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

```

Scilab code Exa 8.3.4 Bilinear Transformation

```

1 //Example 8.3.4
2 //Bilinear Transformation
3 //To convert analog filter into digital filter
4 clear all;
5 clc;
6 close;
7 s = poly(0,'s');
8 H = (s+0.1)/((s+0.1)^2+16);
9 Omega_Analog = 4;
10 Omega_Digital = %pi/2;
11 //Finding Sampling Period

```

```

12 T = (2/Omega_Analog)*(tan(Omega_Digital/2))
13 z = poly(0, 'z');
14 Hz = horner(H, (2/T)*((z-1)/(z+1)))

```

Scilab code Exa 8.3.5 Single pole filter

```

1 //Example 8.3.5 Single pole analog filter
2 //Bilinear Transformation
3 //To convert analog filter into digital filter
4 clear all;
5 clc;
6 close;
7 s = poly(0, 's');
8 Omegac = 0.2*%pi;
9 H = Omegac/(s+Omegac);
10 T =1; //Sampling period T = 1 Second
11 z = poly(0, 'z');
12 Hz = horner(H, (2/T)*((z-1)/(z+1)))
13 disp(Hz, 'Hz =')
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');
23 //Result
24 //Hz =
25 //
26 //      0.6283185 + 0.6283185z
27 //      —————
28 // - 1.3716815 + 2.6283185z
29 //
30 //——>Hz(3)=Hz(3)/2.6283185

```

```

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

```

*For further extension of the exapmle refer to
[AE 8.3.5](#)

Scilab code Exa 8.3.6 Analog Filter Transformation

```

1 //Example 8.3.6
2 // To Design an Analog Butterworth Filter
3 //For the given cutoff frequency Wc = 500 Hz
4 clear all;
5 clc;
6 close;
7 omegap = 2*%pi*500;
8 omegas = 2*%pi*1000;
9 delta1_in_dB = -3;
10 delta2_in_dB = -40;
11 delta1 = 10^(delta1_in_dB/20)

```

```

12 delta2 = 10^(delta2_in_dB/20)
13 //Calculation of Filter Order
14 N = log10((1/(delta2^2))-1)/(2*log10(omegas/omegap))
15 N = ceil(N)
16 omegac = omegap;
17 //Poles and Gain Calculation
18 [pols ,gain]=zpbutt(N,omegac);
19 disp(N,'Filter order N =')
20 disp(pols,'Pole positions are pols =')
21 //Magnitude Response of Analog IIR Butterworth
    Filter
22 h=buttmag(N,omegac,1:1000);
23 //Magnitude in dB
24 mag=20*log10(h);
25 plot2d((1:1000),mag,[0,-180,1000,20]);
26 a=gca();
27 a.thickness = 3;
28 a.foreground = 1;
29 a.font_style = 9;
30 xgrid(5)
31 xtitle('Magnitude Response of Butterworth LPF Filter
    Cutoff frequency = 500 Hz','Analog frequency in
    Hz-->','Magnitude in dB -->');
32 //Result
33 //Filter order N =      7.
34 //s =
35 // column 1 to 3
36 // -699.07013+3062.8264i   -1958.751+2456.196i
    -2830.4772+1363.086i
37 // column 4 to 6
38 // -3141.5927+3.847D-13i   -2830.4772-1363.086i
    -1958.751-2456.196i
39 //column 7
40 //- 699.07013-3062.8264i

```

*For further extension of the exapmle refer to
[AE 8.3.6](#)

Scilab code Exa 8.3.7 Chebyshev Filter

```
1 //Example 8.3.7
2 //To Design an Analog Chebyshev Filter
3 //For the given cutoff frequency = 500 Hz
4 clear all;
5 clc;
6 close;
7 omegap = 1000*%pi; //Analog Passband Edge frequency
   in radians/sec
8 omegas = 2000*%pi; //Analog Stop band edge frequency
   in radians/sec
9 delta1_in_dB = -1;
10 delta2_in_dB = -40;
11 delta1 = 10^(delta1_in_dB/20);
12 delta2 = 10^(delta2_in_dB/20);
13 delta = sqrt(((1/delta2)^2)-1)
14 epsilon = sqrt(((1/delta1)^2)-1)
15 //Calculation of Filter order
16 num = ((sqrt(1-delta2^2))+sqrt(1-((delta2^2)*(1+
   epsilon^2))))/(epsilon*delta2)
17 den = (omegas/omegap)+sqrt((omegas/omegap)^2-1)
18 N = log10(num)/log10(den)
19 //N = (acosh(delta/epsilon))/(acosh(omegas/omegap))
20 N = floor(N)
21 //Cutoff frequency
22 omegac = omegap
23 //Calculation of poles and zeros
24 [pols,Gn] = zpchl(N,epsilon,omegap)
25 disp(N,'Filter order N =');
26 disp(pols,'Poles of a type I lowpass Chebyshev
   filter are Sk =')
27 //Analog Filter Transfer Function
28 h = poly(Gn,'s','coeff')/real(poly(pols,'s'))
29 //Magnitude Response of Chebyshev filter
30 [h2]=cheb1mag(N,omegac,epsilon,1:1000)
```

```

31 //Magnitude in dB
32 mag=20*log10(h2);
33 plot2d((1:1000),mag,[0,-180,1000,20]);
34 a=gca();
35 a.thickness = 3;
36 a.foreground = 1;
37 a.font_style = 9;
38 xgrid(5)
39 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 Design an Digital IIR Butterworth Filter from Analog IIR Butterworth Filter

```

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

```

Scilab code Exa 8.4.2 Digital Filter Transformation

```
1 //Example 8.4.2
2 //To Design an Digital IIR Butterworth Filter from
   Analog IIR Butterworth Filter
3 //and to plot its magnitude response
4 //TRANSFORMATION OF LPF TO BPF USING DIGITAL
   TRANSFORMATION
5 clear all;
6 clc;
7 close;
8 omegaP = 0.2*%pi;
9 omegaL = (2/5)*%pi;
10 omegaU = (3/5)*%pi;
11 z=poly(0, 'z ');
12 H_LPF = (0.245)*(1+(z^-1))/(1-0.509*(z^-1))
13 alpha = (cos((omegaU+omegaL)/2)/cos((omegaU-omegaL)
   /2));
14 k = (cos((omegaU - omegaL)/2)/sin((omegaU - omegaL)
   /2))*tan(omegaP/2);
15 NUM = -((z^2) - ((2*alpha*k/(k+1))*z) + ((k-1)/(k+1)));
16 DEN = (1 - ((2*alpha*k/(k+1))*z) + (((k-1)/(k+1))*(z^2))
   );
17 HZ_BPF=horner(H_LPF, NUM/DEN)
18 disp(HZ_BPF, 'Digital BPF IIR Filter H(Z)= ')
19 HW =frmag(HZ_BPF(2), HZ_BPF(3), 512);
20 W = 0:%pi/511:%pi;
21 plot(W/%pi, HW)
22 a=gca();
23 a.thickness = 3;
24 a.foreground = 1;
25 a.font_style = 9;
26 xgrid(1)
27 xtitle('Magnitude Response of BPF Filter ', 'Digital
   Frequency—>', 'Magnitude');
28 //Result
```

```

29 // Digital BPF IIR Filter H(Z)=
30 //
31 //
32 //
33 //
34 //
35 //
36 // which is equivalent to
37 // H(z) =
38 //
39 //
40 //
41 //
42 //
43 //
44 //
45 //H(z) =
46 //
47 //
48 //
49 //
50 //
51 //
52 //
53 //H(z) =
54 //
55 //
56 //
57 //
58 //
59 //
60 //

```

$$\frac{0.245 - 1.577D^{-17}z - 0.245z + 1.577D^{-17}z + 1.360D^{-17}z}{-0.509 + 1.299D^{-16}z - z + 6.438D^{-17}z + 5.551D^{-17}z}$$

$$\frac{0.245 - 0.245z^2}{-0.509 - z^2}$$

$$\frac{0.245 - 0.245z^{-2}}{0.509 + z^{-2}}$$

61 //

*For further extension of the exapmle refer to
[AE 8.4.2A](#) [AE 8.4.2B](#)

Scilab code CF 8.5 Program to generate different window functions

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

Scilab code CF 8.6 Program to find find frequency response of (1) Han-
ning window (2)Hamming window for M = 31 and M = 61

```
1 //Figure 8.6 and Figure 8.7
2 //Program to frequency response of
3 //(1) Hanning window (2)Hamming window for M = 31
    and M = 61
4 clear all;
5 close;
6 clc
```

```

7 M1 = 31;
8 M2 = 61;
9 for n = 1:M1
10     h_hann_31(n) = 0.5-0.5*cos(2*pi*(n-1)/(M1-1));
11     h_hamm_31(n) = 0.54-0.46*cos(2*pi*(n-1)/(M1-1));
12 end
13 for n = 1:M2
14     h_hann_61(n) = 0.5-0.5*cos(2*pi*(n-1)/(M2-1));
15     h_hamm_61(n) = 0.54-0.46*cos(2*pi*(n-1)/(M2-1));
16 end
17 subplot(2,1,1)
18 [h_hann_31_M,fr]=frmag(h_hann_31,512);
19 [h_hann_61_M,fr]=frmag(h_hann_61,512);
20 h_hann_31_M = 20*log10(h_hann_31_M./max(h_hann_31_M)
    );
21 h_hann_61_M = = 20*log10(h_hann_61_M./max(
    h_hann_61_M));
22 plot2d(fr,h_hann_31_M,2);
23 plot2d(fr,h_hann_61_M,5);
24 legend(['Length M = 31';'Length M = 61']);
25 title('Frequency Response Of Hanning window')
26 subplot(2,1,2)
27 [h_hamm_31_M,fr]=frmag(h_hamm_31,512);
28 [h_hamm_61_M,fr]=frmag(h_hamm_61,512);
29 h_hamm_31_M = 20*log10(h_hamm_31_M./max(h_hamm_31_M)
    );
30 h_hamm_61_M = = 20*log10(h_hamm_61_M./max(
    h_hamm_61_M));
31 plot2d(fr,h_hamm_31_M,2);
32 plot2d(fr,h_hamm_61_M,5);
33 legend(['Length M = 31';'Length M = 61']);
34 title('Frequency Response of Hamming window')

```

Scilab code CF 8.7 Program to find frequency response of (1) Hanning window (2)Hamming window for M = 31

```

1 //Figure 8.6 and Figure 8.7
2 //Program to frequency response of

```

```

3 //(1) Hanning window (2)Hamming window for M = 31
4 clear all;
5 close;
6 clc
7 M = 31;
8 for n = 1:M
9     h_hann_31(n) = 0.5-0.5*cos(2*%pi*(n-1)/(M-1));
10    h_hamm_31(n) = 0.54-0.46*cos(2*%pi*(n-1)/(M-1));
11 end
12 subplot(2,1,1)
13 [h_hann_31_M,fr]=frmag(h_hann_31,512);
14 h_hann_31_M = 20*log10(h_hann_31_M./max(h_hann_31_M)
    );
15 plot2d(fr,h_hann_31_M);
16 xlabel('Normalized Digital Frequency W');
17 ylabel('Magnitude in dB');
18 title('Frequency Response of Hanning window M = 31')
19 subplot(2,1,2)
20 [h_hamm_31_M,fr]=frmag(h_hamm_31,512);
21 h_hamm_31_M = 20*log10(h_hamm_31_M./max(h_hamm_31_M)
    );
22 plot2d(fr,h_hamm_31_M);
23 xlabel('Normalized Digital Frequency W');
24 ylabel('Magnitude in dB');
25 title('Frequency Response of Hamming window M =31')

```

Chapter 10

Multirate Digital Signal Processing

10.1 Scilab Code

Scilab code Exa 10.5.1 Decimation by 2, Filter Length = 30

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

```

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

```

Scilab code Exa 10.5.2 Interpolation by 5, Filter Length = 30

```

1 //Example 10.5.2
2 //Interpolation by 5, Filter Length = 30
3 //Cutoff Frequency Wc = %pi/5
4 //Pass band Edge frequency fp = 0.1 and a Stop band
   edge frequency fs = 0.16
5 // Choose the number of cosine functions and create
   a dense grid
6 // in [0,0.1) and [0.16,0.5)

```

```

7 //magnitude for pass band = 1 & stop band = 0 (i.e)
  [1 0]
8 //Weighting function =[3 1]
9 clear all;
10 clc;
11 close;
12 M = 30; //Filter Length
13 I = 5; //Interpolation Factor = 5
14 Wc = %pi/5; //Cutoff Frequency
15 Wp = Wc/(2*%pi); //Passband Edge Frequency
16 Ws = 0.16; //Stopband Edge Frequency
17 hn=eqfir(M,[0 Wp;Ws .5],[1 0],[3 1]);
18 [hm,fr]=frmag(hn,256);
19 disp('The LPF Filter Coefficients are:')
20 hn
21 //Obtaining Polyphase Filter Coefficients from hn
22 p = zeros(I,M/I);
23 for k = 1:I
24     for n = 1:(length(hn)/I)
25         p(k,n) = hn(I*(n-1)+k);
26     end
27 end
28 disp('The Polyphase Interpolator for I =5 are:')
29 p
30 figure
31 plot(fr,hm)
32 xlabel('Normalized Digital Frequency fr');
33 ylabel('Magnitude');
34 title('Frequency Response of FIR LPF using REMEZ
  algorithm M=61')
35 figure
36 plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
37 xlabel('Normalized Digital Frequency fr');
38 ylabel('Magnitude in dB');
39 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 //Example 10.6.1
2 //Multistage Implementation of Sampling Rate
  Conversion
3 //Decimation factor D = 50
4 //D = D1xD2, D1 = 25, D2 =2
5 clear all;
6 clc;
7 close;
8 Fs = 8000; //Sampling Frequency = 8000Hz
9 Fpc = 75; //Passband Frequency
10 Fsc = 80; //Stopband Frequency
11 Delta_F = (Fsc-Fpc)/Fs; //Transition Band
12 Pass_Band = [0,Fpc];
13 Transition_Band = [Fpc,Fsc];
14 Delta1 = (10^-2); //Passband Ripple
15 Delta2 = (10^-4); //Stopband Ripple
16 D = Fs/(2*Fsc); //Decimation Factor
17 //Decimator Implemented in Two Stages
18 D1 = D/2; //Decimator 1
19 D2 = 2; //Decimator 2
20 //Decimator Single Stage Implementation
21 M = ((-10*log10(Delta1*Delta2)-13)/(14.6*Delta_F))
  +1;
22 M = ceil(M)
23 //Decimator Multistage Implementation
24 //First Stage Implementation
25 F1 = Fs/D1; //New passband for stage1
26 Fsc1 = F1-Fsc; //New Stopband for stage1
27 Delta_F1 = (Fsc1-Fpc)/Fs //New Transition for
  stage1
28 Delta11 = Delta1/2; //New Passband Ripple
29 Delta21 = Delta2; //Stopband Ripple same
30 M1 = ((-10*log10(Delta11*Delta21)-13)/(14.6*Delta_F1
  ))+1
31 M1 = floor(M1)

```

```

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

```

Scilab code Exa 10.8.1 Signal to Distortion Ratio

```

1 //Example 10.8.1
2 //Signal to Distortion Ratio
3 //Calculation of no. of subfilters
4 clear all;
5 clc;
6 close;
7 SDR_dB = 50; //Signal to distortion ratio = 50 dB
8 Wx = 0.8*pi; //Digital maximum frequency of input
    data
9 SDR = 10^(SDR_dB/10)
10 disp('The Number of subfilters required')
11 I = Wx*sqrt(SDR/12);
12 I = ceil(I)

```

Scilab code Exa 10.8.2 Signal to Distortion Ratio using Linear Interpolation

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

Scilab code Exa 10.9.1 Multistage Implementation of Sampling Rate Conversion

```
1 //Example 10.9.1
2 //Multistage Implementation of Sampling Rate
  Conversion
3 //Decimation factor D = 100
4 //D = D1xD2, D1 = 50, D2 =2
5 //Interpolation factor I = 100
6 //I = I1xI2, I1 = 2, I2 =50
7 clear all;
8 clc;
9 close;
10 Fs = 8000; //Sampling Frequency = 8000Hz
11 Fpc = 75; //Passband Frequency
12 Fsc = 80; //Stopband Frequency
13 Delta_F = (Fsc-Fpc)/Fs; //Transition Band
14 Pass_Band = [0,Fpc];
15 Transition_Band = [Fpc,Fsc];
16 Delta1 = (10^-2); //Passband Ripple
```

```

17 Delta2 = (10^-4); //Stopband Ripple
18 D = Fs/(2*Fsc); //Decimation Factor
19 //Decimator Implemented in Two Stages
20 D1 = D/2; //Decimator 1
21 D2 = 2; //Decimator 2
22 //Decimator Single Stage Implementation
23 M = ((-10*log10(Delta1*Delta2/2)-13)/(14.6*Delta_F))
    +1;
24 M = ceil(M)
25 //Decimator Multistage Implementation
26 //First Stage Implementation
27 Delta_F1 = 0.020625 //Obtained from Example 10.6.1
28 M1 = ((-10*log10(Delta1*Delta2/4)-13)/(14.6*Delta_F1
    ))+1
29 M1 = floor(M1)
30 //Second Stage Implementation
31 Delta_F2 = 0.015625 //Obtained from Example 10.6.1
32 M2 = ((-10*log10(Delta1*Delta2/4)-13)/(14.6*Delta_F2
    ))+1
33 M2 = floor(M2)
34 disp('The Filter length Required in Single stage
    Implementation of Decimator is:')
35 M
36 disp('The Filter length Required in Multistage
    Implementation of Decimator is:')
37 M1+M2
38 //Calculation of Reduction Factor
39 R = M/(M1+M2);
40 disp('The Reduction in Filter Length is:')
41 R

```

Chapter 11

Linear Predictions and Optimum Linear Filter

11.1 Scilab Code

Scilab code Exa 11.6.1 Design of Wiener filter of Length $M = 2$

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

Chapter 12

Power Spectrum Estimation

12.1 Scilab Code

Scilab code Exa 12.1.1 Determination of spectrum of a signal With maximum normalized frequency $f = 0.1$ using Rectangular window and Blackmann window

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

```

16  h_balckmann(n)=0.42-0.5*cos(2*%pi*n/(N-1))+0.08*cos
      (4*%pi*n/(N-1));
17  end
18  wft_blmn = wft' .* h_balckmann;
19  wfm_blmn = frmag(wft_blmn, length(fr));
20  WFM_blmn_dB = 20*log10(wfm_blmn);
21  subplot(2,1,1)
22  plot2d(fr, WFM_dB)
23  xtitle('Frequency Response of Rectangular window
      Filtered output M = 61', 'Frequency in cycles per
      samples f', 'Energy density in dB')
24  subplot(2,1,2)
25  plot2d(fr, WFM_blmn_dB)
26  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
Using N-point DFT

```

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

```

```

17 fk1 = k1./L(1);
18 fk2 = k2./L(2);
19 fk3 = k3./L(3);
20 fk4 = k4./L(4);
21 for i =1:length(fk1)
22     Pxx1_fk1(i) = 0;
23     Pxx2_fk1(i) = 0;
24     for m = 1:N
25         Pxx1_fk1(i)=Pxx1_fk1(i)+x1(m)*exp(-sqrt(-1)*2*
                %pi*(m-1)*fk1(i));
26         Pxx2_fk1(i)=Pxx1_fk1(i)+x1(m)*exp(-sqrt(-1)*2*
                %pi*(m-1)*fk1(i));
27     end
28     Pxx1_fk1(i) = (Pxx1_fk1(i)^2)/N;
29     Pxx2_fk1(i) = (Pxx2_fk1(i)^2)/N;
30 end
31 for i =1:length(fk2)
32     Pxx1_fk2(i) = 0;
33     Pxx2_fk2(i) = 0;
34     for m = 1:N
35         Pxx1_fk2(i)=Pxx1_fk2(i)+x1(m)*exp(-sqrt(-1)*2*
                %pi*(m-1)*fk2(i));
36         Pxx2_fk2(i)=Pxx1_fk2(i)+x1(m)*exp(-sqrt(-1)*2*
                %pi*(m-1)*fk2(i));
37     end
38     Pxx1_fk2(i) = (Pxx1_fk2(i)^2)/N;
39     Pxx2_fk2(i) = (Pxx1_fk2(i)^2)/N;
40 end
41 for i =1:length(fk3)
42     Pxx1_fk3(i) = 0;
43     Pxx2_fk3(i) = 0;
44     for m = 1:N
45         Pxx1_fk3(i) =Pxx1_fk3(i)+x1(m)*exp(-sqrt(-1)*2*
                %pi*(m-1)*fk3(i));
46         Pxx2_fk3(i) =Pxx1_fk3(i)+x1(m)*exp(-sqrt(-1)*2*
                %pi*(m-1)*fk3(i));
47     end
48     Pxx1_fk3(i) = (Pxx1_fk3(i)^2)/N;

```

```

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

```

Scilab code Exa 12.5.1 Determination of power, frequency and variance of Additive noise

```

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

```

Appendix to Examples

Scilab code AE 4.2.7 Sampling a Nonbandlimited signal

```
1 //Example 4.2.7 Sampling a Nonbandlimited Signal
2 //Plotting Discrete Time Fourier Transform of
3 //Discrete Time Signal  $x(nT) = \exp(-A*T*abs(n))$ 
4 clear all;
5 clc;
6 close;
7 // Analog Signal
8 A =1; //Amplitude
9 Dt = 0.005;
10 t = -2:Dt:2;
11 //Continuous Time Signal
12 xa = exp(-A*abs(t));
13 //Discrete Time Signal
14 Fs =input('Enter the Sampling Frequency in Hertz ');
    //Fs = 1Hz(or)20Hz
15 Ts = 1/Fs;
16 n = -5:1:5;
17 nTs = n*Ts;
18 x = exp(-A*abs(nTs));
19 // Analog Signal reconstruction
20 Dt = 0.005;
21 t = -2:Dt:2;
22 Xa = x *sinc_new(Fs*(ones(length(nTs),1)*t-nTs'*ones
    (1,length(t)))));
23 // check
24 error = max(abs(Xa - xa))
```

```

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 msec. ');
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 msec. ');
38 ylabel('xa(t)')
39 title('Reconstructed Signal from x(n) using sinc
      function');
40 plot(t,Xa);

```

*Refer to the following for Scilab code of sinc
[ARC 4A](#)

Scilab code ARC 4A sincbyx

```

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

```

Scilab code AE 4.4.2 Frequency Response

```

1 clear all;
2 close;
3 clc;

```

```

4 W = -%pi:(1/500):%pi;
5 z = exp(sqrt(-1)*W);
6 H = z./(z-0.8);
7 Mag_H = abs(H);
8 [Phase_H,m] = phasemag(H);
9 //phasemag used to calculate phase and magnitude in
  dB
10 subplot(2,1,1)
11 plot2d(W,Mag_H)
12 xlabel('Frequency in Radians')
13 ylabel('abs(H)')
14 title('Magnitude Response')
15 subplot(2,1,2)
16 plot2d(W,Phase_H)
17 xlabel('Frequency in Radians')
18 ylabel('<(H)')
19 title('Phase Response')

```

Scilab code AE 8.2.28A DESIGN AND OBTAIN THE FREQUENCY RESPONSE OF FIR FILTER Band Pass

```

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

```

```

16     else
17         n
18         hd(n) = (sin(Wc2*((n-1)-Tuo)) - sin(Wc1*((n-1)-
                Tuo)))/(((n-1)-Tuo)*%pi);
19     end
20     if(abs(hd(n)) < (0.00001))
21         hd(n)=0;
22     end
23 end
24 hd;
25 //Rectangular Window
26 for n = 1:M
27     W(n) = 1;
28 end
29 //Windowing Filter Coefficients
30 h = hd.*W;
31 disp('Filter Coefficients are')
32 h;
33 [hzm,fr]=frmag(h,256);
34 hzm_dB = 20*log10(hzm)./max(hzm);
35 subplot(2,1,1)
36 plot(2*fr,hzm)
37 xlabel('Normalized Digital Frequency W');
38 ylabel('Magnitude');
39 title('Frequency Response of FIR BPF using
        Rectangular window M=11')
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 M=11')

```

Scilab code AE 8.2.28B DESIGN AND OBTAIN THE FREQUENCY RESPONSE OF FIR FILTER Band Stop

```

1 //PROGRAM TO DESIGN AND OBTAIN THE FREQUENCY
  RESPONSE OF FIR FILTER

```

```

2 //Band Stop FILTER (or)Band Reject Filter
3 clear all;
4 clc;
5 close;
6 M = 11 //Filter length = 11
7 Wc = [%pi/4,3*%pi/4]; //Digital Cutoff
    frequency
8 Wc2 = Wc(2)
9 Wc1 = Wc(1)
10 Tuo = (M-1)/2 //Center Value
11 hd = zeros(1,M);
12 W = zeros(1,M);
13 for n = 1:11
14     if (n == Tuo+1)
15         hd(n) = 1-((Wc2-Wc1)/%pi);
16     else     hd(n)=(sin(%pi*((n-1)-Tuo))-sin(Wc2*((n-1)-
        Tuo))+sin(Wc1*((n-1)-Tuo)))/(((n-1)-Tuo)*%pi);
17     end
18     if(abs(hd(n))<(0.00001))
19         hd(n)=0;
20     end
21 end
22 hd
23 //Rectangular Window
24 for n = 1:M
25     W(n) = 1;
26 end
27 //Windowing Filter Coefficients
28 h = hd.*W;
29 disp('Filter Coefficients are')
30 h;
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 BPF using
        Rectangular window M=11')
38 subplot(2,1,2)
39 plot(2*fr,hzm_dB)
40 xlabel('Normalized Digital Frequency W');
41 ylabel('Magnitude in dB');
42 title('Frequency Response Of FIR BPF using
        Rectangular window M=11')

```

Scilab code AE 8.2.28C DESIGN AND OBTAIN THE FREQUENCY RESPONSE OF FIR FILTER High

```

1 //Figure 8.9 and 8.10
2 //PROGRAM TO DESIGN AND OBTAIN THE FREQUENCY
  RESPONSE OF FIR FILTER
3 //LOW PASS FILTER
4 clear all;
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) = Wc/%pi;
13     else
14         hd(n) = sin(Wc*((n-1)-Tuo))/(((n-1)-Tuo)*%pi)
15         ;
16     end
17 //Rectangular Window
18 for n = 1:M
19     W(n) = 1;
20 end
21 //Windowing Filter 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(fr,hzm)
29 xlabel('Normalized Digital Frequency W');
30 ylabel('Magnitude');
31 title('Frequency Response Of FIR LPF using
      Rectangular window M=61')
32 subplot(2,1,2)
33 plot(fr,hzm_dB)
34 xlabel('Normalized Digital Frequency W');
35 ylabel('Magnitude in dB');
36 title('Frequency Response Of FIR LPF using
      Rectangular window M=61')

```

Scilab code AE 8.3.5 High Pass Filter

```

1 //Example 8.3.5
2 //First Order Butterworth Filter
3 //Low Pass Filter
4 clear all;
5 clc;
6 close;
7 s = poly(0,'s');
8 Omegac = 0.2*%pi;
9 H = Omegac/(s+Omegac);
10 T =1;//Sampling period T = 1 Second
11 z = poly(0,'z');
12 Hz = horner(H,(2/T)*((z-1)/(z+1)))
13 HW =frmag(Hz(2),Hz(3),512);
14 W = 0:%pi/511:%pi;
15 plot(W/%pi,HW)
16 a=gca();
17 a.thickness = 3;
18 a.foreground = 1;
19 a.font_style = 9;
20 xgrid(1)

```

```

21 xtitle('Magnitude Response of Single pole LPF Filter
    Cutoff frequency =  $0.2 \cdot \pi$ ', 'Digital Frequency
    —>', 'Magnitude');

```

Scilab code AE 8.3.6 Analog Low Pass

```

1 //Example 8.3.6
2 // To Design an Analog Low Pass IIR Butterworth
  Filter
3 //For the given cutoff frequency  $\omega_c = 500$  Hz
4 clear all;
5 clc;
6 close;
7  $\omega_{gap} = 500$ ;
8  $\omega_{gas} = 1000$ ;
9  $\delta_{1\_in\_dB} = -3$ ;
10  $\delta_{2\_in\_dB} = -40$ ;
11  $\delta_1 = 10^{(\delta_{1\_in\_dB}/20)}$ 
12  $\delta_2 = 10^{(\delta_{2\_in\_dB}/20)}$ 
13 //Calculation of Filter Order
14  $N = \log_{10}((1/(\delta_2^2))-1)/(2 \cdot \log_{10}(\omega_{gas}/\omega_{gap}))$ 
15  $N = \text{ceil}(N)$ 
16  $\omega_{cac} = \omega_{gap}$ ;
17 //Poles and Gain Calculation
18  $[\text{pols}, \text{gain}] = \text{zpbutt}(N, \omega_{cac})$ ;
19 //Magnitude Response of Analog IIR Butterworth
  Filter
20  $h = \text{buttmag}(N, \omega_{cac}, 1:1000)$ ;
21 //Magnitude in dB
22  $\text{mag} = 20 \cdot \log_{10}(h)$ ;
23 plot2d((1:1000), mag, [0, -180, 1000, 20]);
24 a=gca();
25 a.thickness = 3;
26 a.foreground = 1;
27 a.font_style = 9;
28 xgrid(5)
29 xtitle('Magnitude Response of Butterworth LPF Filter
    Cutoff frequency = 500 Hz', 'Analog frequency in

```

Hz—>', 'Magnitude in dB —>');

Scilab code AE 8.4.1 High Pass Filter

```
1 //Example 8.3.5
2 //First Order Butterworth Filter
3 //High Pass Filter
4 //Table 8.13: Using Digital Filter Transformation
5 clear all;
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_LPF = horner(H, (2/T)*((z-1)/(z+1)));
14 alpha = -(cos((Omegac+Omegac)/2))/(cos((Omegac -
    Omegac)/2));
15 HZ_HPF=horner(H_LPF, -(z+alpha)/(1+alpha*z))
16 HW =frmag(HZ_HPF(2),HZ_HPF(3),512);
17 W = 0:%pi/511:%pi;
18 plot(W/%pi,HW)
19 a=gca();
20 a.thickness = 3;
21 a.foreground = 1;
22 a.font_style = 9;
23 xgrid(1)
24 xtitle('Magnitude Response of Single pole HPF Filter
    Cutoff frequency = 0.2*pi', 'Digital Frequency
    —>', 'Magnitude');
```

Scilab code AE 8.4.2A Analog Filter Transformation

```
1 //Example 8.4.2
2 //To Design an Digital IIR Butterworth Filter from
    Analog IIR Butterworth Filter
3 //and to plot its magnitude response
```

```

4 //TRANSFORMATION OF LPF TO BSF USING DIGITAL
   TRANSFORMATION
5 clear all;
6 clc;
7 close;
8 omegaP = 0.2*%pi;
9 omegaL = (2/5)*%pi;
10 omegaU = (3/5)*%pi;
11 z=poly(0, 'z ');
12 H_LPF = (0.245)*(1+(z^-1))/(1-0.509*(z^-1))
13 alpha = (cos((omegaU+omegaL)/2)/cos((omegaU-omegaL)
   /2));
14 k = tan((omegaU - omegaL)/2)*tan(omegaP/2);
15 NUM =((z^2)-((2*alpha/(1+k))*z)+((1-k)/(1+k)));
16 DEN = (1-((2*alpha/(1+k))*z)+(((1-k)/(1+k))*(z^2)));
17 HZ_BPF=horner(H_LPF,NUM/DEN)
18 HW =frmag(HZ_BPF(2),HZ_BPF(3),512);
19 W = 0:%pi/511:%pi;
20 plot(W/%pi,HW)
21 a=gca();
22 a.thickness = 3;
23 a.foreground = 1;
24 a.font_style = 9;
25 xgrid(1)
26 xtitle('Magnitude Response of BSF Filter','Digital
   Frequency—>','Magnitude');

```

Scilab code AE 8.4.2B Digital Filter Transformation

```

1 //Caption:Conveting single pole LPF Butterworth
   filter into BPF
2 //Exa8.4.1
3 //page698
4 clc;
5 Op = sym('Op'); //pass band edge frequency of low
   pass filter
6 s = sym('s');

```

```

7 0l = sym('0l'); //lower cutoff frequency of band
    pass filter
8 0u = sym('0u'); //upper cutoff frequency of band
    pass filter
9 s1 = Op*(s^2+0l*0u)/(s*(0u-0l)); //Analog
    transformation for LPF to BPF
10 H_Lpf = Op/(s+Op); //single pole analog LPF
    Butterworth filter
11 H_Bpf = limit(H_Lpf,s,s1); //analog BPF Butterworth
    filter
12 disp(H_Lpf, 'H_Lpf =')
13 disp(H_Bpf, 'H_Bpf =')
14 //Result
15 //H_Lpf = Op/(s+Op)
16 //H_Bpf = (0u-0l)*s/(s^2+(0u-0l)*s+0l*0u)

```
