Scilab Code for Digital Signal Processing Principles, Algorithms and Applications by J. G. Proakis & D. G. Manolakis[1]

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

[1]Funded 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

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

```
//Implementation of Equation 1.2.1 in Chapter 1
//Digital Signal Processing by Proakis, Third Edition, PHI
//Page 9

clear; clc; close;
n = 0:10;
x = (0.8)^n;
//plot2d4(n,x)
a=gca();
a.thickness = 2;
plot2d3('gnn',n,x)
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

```
//Implementation of Equation 2.1.6 in Chapter 2
//Digital Signal Processing by Proakis, Third
    Edition, PHI
//Page 45

clear; clc; close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),1,zeros(1,L)];
a=gca();
a.thickness = 2;
a.y_location = "middle";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Unit Sample
    Sequence','n','x[n]');
```

Scilab code Eqn 2.1.7 Unit step sequence

```
//Implementation of Equation 2.1.7 in Chapter 2
//Digital Signal Processing by Proakis, Third
    Edition, PHI
//Page 45

clear; clc; close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),ones(1,L+1)];
a=gca();
a.thickness = 2;
a.y_location = "middle";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Unit Step Signal
    ','n','x[n]');
```

Scilab code Eqn 2.1.8 Unit ramp sequence

```
//Implementation of Equation 2.1.8 in Chapter 2
//Digital Signal Processing by Proakis, Third
    Edition, PHI
//Page 45

clear; clc; close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),0:L];
a=gca();
a.thickness = 2;
a.y_location = "middle";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Unit Ramp Signal
    ','n','x[n]');
```

Scilab code Eqn 2.1.9a Exponential sequence

```
//Implementation of Equation 2.1.9 in Chapter 2
```

```
//Digital Signal Processing by Proakis, Third
   Edition, PHI
//Page 46
clear;
clc;
close;
a =1.5;
n =1:10;
x = (a)^n;
a=gca();
a.thickness = 2;
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Exponential
   Signal','n','x[n]');
```

Scilab code Eqn 2.1.9b Exponential increasing sequence

```
//Implementation of Equation 2.1.9b in Chapter 2
//Digital Signal Processing by Proakis, Third
   Edition, PHI
//Page 46
// a < 0
clear;
clc;
close;
a =-1.5;
n = 0:10;
x = (a)^n;
a=gca();
a.thickness = 2;
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Exponential
   Increasing-Decreasing Signal','n','x[n]');
```

Scilab code Eqn 2.1.9c Exponential decreasing sequence

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

Scilab code Eqn 2.1.24 Even signal

```
//Implementation of Equation 2.1.24 in Chapter 2
//Digital Signal Processing by Proakis, Third
   Edition, PHI
//Page 51

clear; clc; close;
n = -7:7;
x1 = [0 0 0 1 2 3 4];
x = [x1,5,x1(length(x1):-1:1)];
a=gca();
a.thickness = 2;
a.y_location = "middle";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Even Signal','n'
   ,'x[n]');
```

Scilab code Eqn 2.1.25 Odd signal

```
//Implementation of Equation 2.1.25 in Chapter 2
//Digital Signal Processing by Proakis, Third
   Edition, PHI
//Page 51
clear;
clc;
close;
n = -5:5;
x1 = [0 1 2 3 4 5];
x = [-x1($:-1:2),x1];
a=gca();
a.thickness = 2;
a.y_location = "middle";
a.x_location = "middle"
plot2d3('gnn',n,x)
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

```
//Example 3.1.1
//Z Transform of Finite Duration SIgnals
clear all;
clc;
close;
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)
```

*Refer to the following for Scilab code of ztransfer new
ARC 3A

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

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

```
//Example 3.1.4
//Z transform of x[n] = -alpha^n. u[-n-1]
//alpha = 0.5
clear all;
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 $x(n) = a^n.u(n) + b^n.u(-n-1)$

```
//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 all;
close;
clc;
syms n z;
x1=(0.5)^(n)
X1=symsum(x1*(z^(-n)),n,0,%inf)
x2=(0.6)^(-n)
X2=symsum(x2*(z^(n)),n,1,%inf)
X = (X1+X2)
disp(X,"ans=")
```

Scilab code Exa 3.2.1 Z transform of $x(n) = 3.2^n.u(n) - 4.3^n.u(n)$

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

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

```
//Example 3.2.2
//Z transform of x[n] = cos(Wo.n).u[n]
//Z transform of y[n] = sin(Wo.n).u[n]
clear all;
close;
clc;
syms n z;
Wo =2;
x1=exp(sqrt(-1)*Wo*n);
X1=symsum(x1*(z^(-n)),n,0,%inf);
x2=exp(-sqrt(-1)*Wo*n);
X2=symsum(x2*(z^(-n)),n,0,%inf)
X =(X1+X2)
disp(X,"ans=")
Y =(1/(2*sqrt(-1)))*(X1-X2)
disp(Y,"ans=")
```

Scilab code Exa 3.2.3 Time shifting property of Z transform

```
//Example 3.2.3
//Time Shifting Property of Z-transform
clear all;
clc;
close;
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 = 0-2:length(x1)-1-2;
X2 = ztransfer_new(x1,n2)
//x3 =[0,0,1,2,5,7,0,1];
n3 = 0+2:length(x1)-1+2;
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 $x(n) = u(n)$

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

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

```
//Example 3.2.6
//Z transform of x[n] = u[-n]
clear all;
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 $x(n) = n.a^n.u(n)$

```
//Example 3.2.7
//Z transform of x[n] = n.a^n.u[n]
clear all;
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

```
//Example 3.2.9
//Convolution Property Proof
clear all;
clc;
close;
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
```

*Refer to the following for Scilab code of ztransfer new
ARC 3A

Scilab code Exa 3.2.10 Correlation Property Proof

```
//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))
```

*Refer to the following for Scilab code of ztransfer new
ARC 3A

Scilab code ARC 3A Ztarnsfer of a sequence

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

```
//Example 4.1.2 Continuous Time Fourier Transform
//and Energy Density Function of a Square Waveform
// x(t)= A, from -T/2 to T/2
clear all;
clc;
close;
// Analog Signal
A =1;      //Amplitude
Dt = 0.005;
T = 4;   //Time in seconds
t = -T/2:Dt:T/2;
for i = 1:length(t)
  xa(i) = A;
end
//
// Continuous-time Fourier Transform
Wmax = 2*%pi*2;         //Analog Frequency = 2Hz
```

```
K = 4; k = 0:(K/800):K;
W = k*Wmax/K;
disp(size(xa))
Xa=xa'*exp(-sqrt(-1)*t'*W)*Dt;
Xa = real(Xa);
W = [-mtlb_fliplr(W), W(2:501)]; // Omega from -Wmax
    to Wmax
Xa = [mtlb_fliplr(Xa), Xa(2:501)];
ESD = Xa^2;  //Energy Density Spectrum
subplot(3,1,1);
plot(t,xa);
xlabel('t in msec.');
ylabel('xa(t)')
title('Analog Signal')
subplot(3,1,2);
plot(W/(2*%pi),Xa);
xlabel('Frequency in Hz');
ylabel('Xa(jW)')
title('Continuous-time Fourier Transform')
subplot(3,1,3);
plot(W/(2*%pi),ESD);
xlabel('Frequency in Hz');
ylabel('SXX')
title('Energy Density Spectrum')
```

Scilab code Exa 4.2.7 Sampling a Nonbandlimited Signal

```
//Example 4.2.7 Sampling a Nonbandlimited Signal
//Plotting Continuous Time Fourier Transform of
//Continuous Time Signal x(t)= exp(-A*abs(t))
clear all;
clc;
close;
// Analog Signal
A =1;     //Amplitude
Dt = 0.005;
t = -2:Dt:2;
xa = exp(-A*abs(t));
```

```
//
// Continuous-time Fourier Transform
Wmax = 2*%pi*2;          //Analog Frequency = 2Hz
K = 4;
k = 0:(K/500):K;
W = k*Wmax/K;
Xa = xa * exp(-sqrt(-1)*t'*W) * Dt;
Xa = real(Xa);
W = [-mtlb_fliplr(W), W(2:501)]; // Omega from -Wmax
    to Wmax
Xa = [mtlb_fliplr(Xa), Xa(2:501)];
subplot(2,1,1);
a =gca();
a.x_location = "origin";
a.y_location = "origin";
plot(t,xa);
xlabel('t in msec.');
ylabel('xa(t)')
title('Analog Signal')
subplot(2,1,2);
a =gca();
a.x_location = "origin";
a.y_location = "origin";
plot(W/(2*%pi),Xa);
xlabel('Frequency in Hz');
ylabel('Xa(jW)*1000')
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 $x1(n) = x2(n) = [1, 1, 1]$

```
//Example 4.3.4
//Convolution Property Example
```

```
//x1(n)=x2(n)= [1,1,1]
clear all;
clc;
close;
n =-1:1;
x1 = [1,1,1];
x2 = x1;
//Discrete-time Fourier transform
K = 500;
k = 0:1:K;
w = %pi*k/K;
X1 = x1 * exp(-sqrt(-1)*n'*w);
X2 = x2 * exp(-sqrt(-1)*n'*w);
w = [-mtlb_fliplr(w), w(2:K+1)]; // Omega from -w to
    w
X1 = [mtlb_fliplr(X1), X1(2:K+1)];
X2 = [mtlb_fliplr(X2), X2(2:K+1)];
Freq_X1 = real(X1);
Freq_X2 = real(X2);
X = X1.*X2;
K1 = length(X)
k1 = 0:1:K1;
w1 = %pi*k1/K1;
w1 = [-2*mtlb_fliplr(w), 2*w];
X = [mtlb_fliplr(X), X(1:K1)];
Freq_X = real(X);
//Inv_X = X.*exp(sqrt(-1)*n'*w)
x = convol(x1,x2)
//Plotting Magitude Responses
figure(1)
a =gca();
a.x_location = 'middle'
a.y_location = 'middle'
a.x_label
a.y_label
plot2d(w/%pi,Freq_X1)
x_label =a.x_label
y_label = a.y_label
```

```
x_label.text ='                    Frequency in Radians'
y_label.text ='                                      X1(w)'
//xlabel('Frequency in Radians')
//ylabel('X1(w)')
title('Frequency Response')
figure(2)
a =gca();
a.x_location = 'middle'
a.y_location = 'middle'
a.x_label
a.y_label
plot2d(w/%pi,Freq_X2)
x_label =a.x_label
y_label = a.y_label
x_label.text ='                                   Frequency
    in Radians'
y_label.text ='
                                                 X2(w)'
title('Frequency Response')
figure(3)
a =gca();
a.y_location = 'middle'
a.x_label
a.y_label
plot2d(w1/(2*%pi),Freq_X)
x_label =a.x_label
y_label = a.y_label
x_label.text ='                                   Frequency
    in Radians'
y_label.text ='
                                                  X(w)'
title('Frequency Response')
```

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

```
//Example 4.4.2
```

```
//Frequency Response of Three point Moving Average
   System
//y(n)= (1/3)[x(n+1)+x(n)+x(n-1)]
//h(n) = [1/3,1/3,1/3]
clear all;
clc;
close;
//Calculation of Impulse Response
n =-1:1;
h = [1/3,1/3,1/3];
//Discrete-time Fourier transform
K = 500;
k = 0:1:K;
w = %pi*k/K;
H = h * exp(-sqrt(-1)*n'*w);
//phasemag used to calculate phase and magnitude in
   dB
[Phase_H,m] = phasemag(H);
H = abs(H);
subplot(2,1,1)
plot2d(w/%pi,H)
xlabel('Frequency in Radians')
ylabel('abs(H)')
title('Magnitude Response')
subplot(2,1,2)
plot2d(w/%pi,Phase_H)
xlabel('Frequency in Radians')
ylabel('<(H)')
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

```
//Example 4.4.4
//Frequency Response of First Order Difference
   Equation
//a = 0.9 and b = 1-a
//Impulse Response h(n) = b.(a^n).u(n)
clear all;
clc;
close;
a = input('Enter the constant value of Ist order
   Difference Equation');
b= 1-a;
//Calculation of Impulse Response
n =0:50;
h =b*(a.^n) ;
//Discrete-time Fourier transform
K = 500;
k = 0:1:K;
w = %pi*k/K;
H = h * exp(-sqrt(-1)*n'*w);
//phasemag used to calculate phase and magnitude in
   dB
[Phase_H,m] = phasemag(H);
H = real(H);
subplot(2,1,1)
plot2d(w/%pi,H)
xlabel('Frequency in Radians')
ylabel('abs(H)')
title('Magnitude Response')
subplot(2,1,2)
plot2d(w/%pi,Phase_H)
xlabel('Frequency in Radians')
ylabel('<(H)')
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

```
//Example 5.1.2
//Determination of N-point DFT
//Plotting Magnitude and Phase spectrum
clear all;
clc;
close;
L = 10;  // Length of the sequence
N = 10;  // N -point DFT
for n =0:L-1
  x(n+1) = 1;
end
//Computing DFT and IDFT
X = dft(x,-1)
x_inv =abs(dft(X,1))
//Computing Magnitude and Phase Spectrum
//Using DTFT
n = 0:L-1;
K = 500;
```

```
k = 0:1:K;
w = 2*%pi*k/K;
X_W = x * exp(-sqrt(-1)*n'*w);
Mag_X = abs(X_W);
//phasemag used to calculate phase and magnitude in
   dB
Phase_X = atan(imag(X_W),real(X_W))
subplot(2,1,1)
plot2d(w,Mag_X)
xlabel('Frequency in Radians')
ylabel('abs(X)')
title('Magnitude Response')
subplot(2,1,2)
plot2d(w,Phase_X)
xlabel('Frequency in Radians')
ylabel('<(X)')
title('Phase Response')
```

Scilab code Exa 5.1.3 Finding DFT and IDFT

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

Scilab code Exa 5.2.1 Performing Circular COnvolution Using DFT

```
//Example 5.2.1 and Example 5.2.2
//Performing Circular COnvolution
//Using DFT
```

```
clear all;
clc;
close;
L = 4; //Length of the Sequence
N = 4;  // N -point DFT
x1 = [2,1,2,1];
x2 = [1,2,3,4];
//Computing DFT
X1 = dft(x1,-1)
X2 = dft(x2,-1)
//Multiplication of 2 DFTs
X3 = X1.*X2
//Circular Convolution Result
x3 =abs(dft(X3,1))
```

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

```
//Example 5.3.1
//Performing Linear Filtering (i.e) Linear
    Convolution
//Using DFT
clear all;
clc;
close;
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 = dft(h1,-1)
X = dft(x1,-1)
```

```
//Multiplication of 2 DFTs
Y = X.*H
//Linear Convolution Result
y =abs(dft(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

```
//Example 5.4.1
//Effect of Zero Padding
clear all;
clc;
close;
L = 100;  // Length of the sequence
N = 200;  // N -point DFT
n = 0:L-1;
x = (0.95).^n;
//Padding zeros to find N = 200 point DFT
x_padd = [x, zeros(1,N-L)];
//Computing DFT
X = dft(x,-1);
X_padd =  dft(x_padd,-1);
subplot(2,1,1)
plot2d(X)
```

```
xlabel('K')
ylabel('X(k)')
title('For L =100 and N =100')
subplot(2,1,2)
plot2d(X_padd)
xlabel('K')
ylabel('X(k) zero padded')
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

```
//Example 6.4.1
//Program to Calculate No.of bits required for given
//Signal to Quantization Noise Ratio
//in computing DFT
clear all;
clc;
close;
N = 1024;
SQNR = 30;  //SQNR = 30 dB
v = log2(N); //number of stages
b = (log2(10^(SQNR/10))+2*v)/2;
b = ceil(b)
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

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

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

```
//Exercise 6.8
//Program to Calculate DFT using DIF-FFT algorithm
//x[n]= 1, 0<=n<=7
clear all;
clc;
close;
x = [1,1,1,1,1,1,1,1];
X = fft(x,-1)
//Inverse FFT
x_inv = real(fft(X,1))
```

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

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

```
//Inverse FFT
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

```
//Example 7.6.3
//Program to Calculate  Quantization Noise in FIR Filter
//For M = 32 and No.of bits = 12
clear all;
clc;
close;
b = input('Enter the number of bits');
M = input('Enter the FIR filter length');
disp('Coefficient Quantization Error in FIR Filter')
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 $y(n) = ay(n-1) + x(n); a = (1/2)anda = (3/4)$

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

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

```
//Example 7.7.1
//Determination of Variance of round-off noise
//at the output of cascade realization
//H1(Z) = 1/(1-(1/2)z-1)
//H2(Z) = 1/(1-(1/4)z-1)
//H(Z) = (2/(1-(1/2)z-1))-(1/(1-(1/4)z-1))
clear all;
clc;
close;
a1 = (1/2); //pole of first system in cascade connection
a2 = (1/4); //ploe of second system in cascade connection
sigma_e = 1; //quantization noise variance
//Noise variance of H1(Z)
sigma_2 = (1/(1-a2^2))*sigma_e^2//noise variance of second system
//Noise variance of H2(Z)
sigma_1 = 1/(1-a1^2)*sigma_e^2//noise variance of first system
```

```
//Nosie variance of H(Z)
sigma = (((2^2)/(1-a1^2))-((2^2)/(1-a1*a2))+(1/(1-a2
   ^2)))*sigma_e^2
noise_variance = sigma+sigma_2 //Total noise
   variance
//Result
//sigma_2  =   1.0666667
//sigma_1  =   1.3333333
//sigma    =    1.8285714
//noise_variance  =   2.8952381
```

Scilab code Eqn 7.7.40 Signal to Quantization Noise Ratio

```
//Equation 6.4.17
//page492
//Program to Calculate  Signal to Quantization Noise
    Ratio
//in FFT algorithm
clear all;
clc;
close;
N = input('Enter the N point FFT value');
b = log2(N)
Quantization_Noise = (2/3)*(2^(-2*b))
Signal_Power = (1/(3*N))
SQNR = Signal_Power/Quantization_Noise
//RESULT
//Enter the N point FFT value 1024
// b  =    10.
// Quantization_Noise  =   0.0000006
// Signal_Power  =   0.0003255
// SQNR  =    512.
//-->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 Frequecny Sampling Technique

```
//Example 8.2.1
//Design of FIR Filter using Frequecny Sampling
   Technique
//Low Pass Filter Design
clear all;
clc;
close;
M =15;
Hr = [1,1,1,1,0.4,0,0,0];
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 8.2.2 Design of FIR Filter using Frequecny Sampling Technique

```
//Example 8.2.2
//Design of FIR Filter using Frequecny Sampling
   Technique
//Low Pass Filter Design
clear all;
clc;
close;
M =32;
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 Of 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 Of FIR LPF using Frequency
    Sampling Technique with M = 15 with Cutoff
   Frequency = 0.466')
xgrid(2)
```

Scilab code Exa 8.2.3 Low Pass FIlter

```
//Example 8.2.3
//Low Pass FIlter of length M = 61
```

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

Scilab code Exa 8.2.4 Band Pass FIlter

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

```
//magnitude for pass band = 1 & stop band = 0 (i.e) [0 1 0]
//Weighting function =[10 1 10]
clear all;
clc;
close;
hn = 0;
hm = 0;
hn=eqfir(32,[0 .1;.2 .35;.425 .5],[0 1 0],[10 1 10]);
[hm,fr]=frmag(hn,256);
disp('The Filter Coefficients are:')
hn
figure
plot(fr,hm)
a =gca();
xlabel('Normalized Digital Frequency fr');
ylabel('Magnitude');
title('Frequency Response of FIR BPF using REMEZ algorithm M=32')
xgrid(2)
figure
plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
a = gca();
xlabel('Normalized Digital Frequency fr');
ylabel('Magnitude in dB');
title('Frequency Response of FIR BPF using REMEZ algorithm M=32')
xgrid(2)
```

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

```
//Example 8.2.5
//Linear Phase FIR Differentiator of length M = 60
//Pass Band Edge frequency fp = 0.1
clear all;
clc;
close;
```

```
M =60;
tuo = (M/2)-1;
Wc = 0.1;
h =  zeros(1,M);
for n = 1:M
  if n ~= M/2
    h(n) =cos((n-1-tuo)*Wc)/(n-1-tuo);
  end
end
[hm,fr]=frmag(h,1024);
disp('The Filter Coefficients are:')
h
figure
plot(fr,hm/max(hm))
a =gca();
xlabel('Normalized Digital Frequency fr');
ylabel('Magnitude');
title('Frequency Response of FIR Differentiator for
   M=60')
xgrid(2)
```

Scilab code Exa 8.2.6 Hilbert Transform of Length M = 31

```
//Example 8.2.6
// Plotting Hibert Transformer of Length M = 31
//Default Window Rectangular Window
//Chebyshev approx default parameter = [0 0]
clear all;
clc;
close;
M =31;//Hibert Transformer Length = 31
tuo = (M-1)/2;
Wc = %pi;
h =  zeros(1,M);
for n = 1:M
   if n ~= ((M-1)/2)+1
     h(n) =(2/%pi)*(sin((n-1-tuo)*Wc/2)^2)/(n-1-tuo)
         ;
```

```
  end
end
disp('The Hilbert Coefficients are:')
h
Rec_Window = ones(1,M);//Rectangular Window generation
h_Rec = h.*Rec_Window;//Windowing With Rectangular window
//Hamming Window geneartion
for n=1:M
  hamm_Window(n) = 0.54-0.46*cos(2*%pi*(n-1)/(M-1));
end
h_hamm = h.*hamm_Window';//Windowing With hamming window;
//Hilbert Transformer using Rectangular window
[hm_Rec,fr]=frmag(h_Rec,1024);
hm_Rec_dB = 20*log10(hm_Rec);
figure
plot(fr,hm_Rec_dB)
a =gca();
xlabel('Normalized Digital Frequency fr');
ylabel('Magnitude');
title('Frequency Response of FIR Hibert Transformer using Rectangular window for M=31')
xgrid(2)
//Hilbert Transformer using Hamming window
[hm_hamm,fr]=frmag(h_hamm,1024);
disp('The Hilbert Coefficients are:')
hm_hamm_dB = 20*log10(hm_hamm);
figure
plot(fr,hm_hamm_dB)
a =gca();
xlabel('Normalized Digital Frequency fr');
ylabel('Magnitude');
title('Frequency Response of FIR Hibert Transformer using hamming window for M=31')
xgrid(2)
```

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

```
//Figure 8.9 and 8.10
//PROGRAM TO DESIGN AND OBTAIN THE FREQUENCY
    RESPONSE OF FIR FILTER
//LOW PASS FILTER
clear all;
clc;
close;
M = 61                  //Filter length = 61
Wc = %pi/5;             //Digital Cutoff frequency
Tuo = (M-1)/2          //Center Value
for n = 1:M
    if (n == Tuo+1)
      hd(n) = Wc/%pi;
    else
      hd(n) =  sin(Wc*((n-1)-Tuo))/(((n-1)-Tuo)*%pi)
          ;
  end
end
//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(fr,hzm)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude');
title('Frequency Response 0f FIR LPF using
    Rectangular window M=61')
subplot(2,1,2)
```

```
plot(fr,hzm_dB)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response 0f 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

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

Scilab code Exa 8.3.4 Bilinear Transformation

```
//Example 8.3.4
//Bilinear Transformation
//To convert analog filter into digital filter
clear all;
clc;
close;
s = poly(0,'s');
H = (s+0.1)/((s+0.1)^2+16);
Omega_Analog = 4;
Omega_Digital = %pi/2;
//Finding Sampling Period
```

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

Scilab code Exa 8.3.5 Single pole filter

```
//Example 8.3.5 Sigle pole analog filter
//Bilinear Transformation
//To convert analog filter into digital filter
clear all;
clc;
close;
s = poly(0,'s');
Omegac = 0.2*%pi;
H = Omegac/(s+Omegac);
T =1;//Sampling period T = 1 Second
z = poly(0,'z');
Hz = horner(H,(2/T)*((z-1)/(z+1)))
disp(Hz,'Hz =')
HW  =frmag(Hz(2),Hz(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 Single pole LPF Filter Cutoff frequency = 0.2*pi','Digital Frequency --->','Magnitude');
//Result
//Hz =
//
//    0.6283185 + 0.6283185z
//    ----------------------
//  - 1.3716815 + 2.6283185z
//
//-->Hz(3)=Hz(3)/2.6283185
```

```
//  Hz   =
//
//      0.6283185 + 0.6283185z
//     ------------------------
//       - 0.5218856 + z
//
//-->Hz(2)=Hz(2)/2.6283185
//  Hz   =
//
//      0.2390572 + 0.2390572z
//     ------------------------
//       - 0.5218856 + z
//
//       which is equivalent to
//Hz =
//
//    0.2390572(1 + z^-1)
//     ------------------------
//      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

```
//Example 8.3.6
// To Design an Analog Butterworth Filter
//For the given cutoff frequency Wc = 500 Hz
clear all;
clc;
close;
omegap = 2*%pi*500;
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
```

*For further extension of the exapmle refer to AE 8.3.6

Scilab code Exa 8.3.7 Chebyshev Filter

```
//Example 8.3.7
//To Design an Analog Chebyshev Filter
//For the given cutoff frequency = 500 Hz
clear all;
clc;
close;
omegap = 1000*%pi; //Analog Passband Edge frequency in radians/sec
omegas = 2000*%pi; //Analog Stop band edge frequency in radians/sec
delta1_in_dB = -1;
delta2_in_dB = -40;
delta1 = 10^(delta1_in_dB/20);
delta2 = 10^(delta2_in_dB/20);
delta = sqrt(((1/delta2)^2)-1)
epsilon  = sqrt(((1/delta1)^2)-1)
//Calculation of Filter order
num = ((sqrt(1-delta2^2))+(sqrt(1-((delta2^2)*(1+epsilon^2)))))/(epsilon*delta2)
den = (omegas/omegap)+sqrt((omegas/omegap)^2-1)
N = log10(num)/log10(den)
//N = (acosh(delta/epsilon))/(acosh(omegas/omegap))
N = floor(N)
//Cutoff frequency
omegac = omegap
//Calculation of poles and zeros
[pols,Gn] = zpch1(N,epsilon,omegap)
disp(N,'Filter order N =');
disp(pols,'Poles of a type I lowpass Chebyshev filter are Sk =')
//Analog Filter Transfer Function
h = poly(Gn,'s','coeff')/real(poly(pols,'s'))
//Magnitude Response of Chebyshev filter
[h2]=cheb1mag(N,omegac,epsilon,1:1000)
```

```
//Magnitude in dB
mag=20*log10(h2);
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 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

```
//Caption:Conveting single pole LPF Butterworth
    filter into BPF
//Exa8.4.1
//page698
clc;
Op = sym('Op'); //pass band edge frequency of low
    pass filter
s = sym('s');
Ol = sym('Ol');  //lower cutoff frequency of band
    pass filter
Ou = sym('Ou'); //upper cutoff frequency of band
    pass filter
s1 = Op*(s^2+Ol*Ou)/(s*(Ou-Ol)); //Analog
    transformation for LPF to BPF
H_Lpf = Op/(s+Op); //single pole analog LPF
    Butterworth filter
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 8.4.2 Digital Filter Transformation

```
//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 all;
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
//
//H(z) =
//
//                -2
//     0.245-  0.245z
//     ----------------
//           -2
//      0.509+z
//
```

```
//
```

*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

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

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

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

```
M1 = 31;
M2 = 61;
for n = 1:M1
  h_hann_31(n) = 0.5-0.5*cos(2*%pi*(n-1)/(M1-1));
  h_hamm_31(n) = 0.54-0.46*cos(2*%pi*(n-1)/(M1-1));
end
for n = 1:M2
  h_hann_61(n) = 0.5-0.5*cos(2*%pi*(n-1)/(M2-1));
  h_hamm_61(n) = 0.54-0.46*cos(2*%pi*(n-1)/(M2-1));
end
subplot(2,1,1)
[h_hann_31_M,fr]=frmag(h_hann_31,512);
[h_hann_61_M,fr]=frmag(h_hann_61,512);
h_hann_31_M = 20*log10(h_hann_31_M./max(h_hann_31_M)
   );
h_hann_61_M = = 20*log10(h_hann_61_M./max(
   h_hann_61_M));
plot2d(fr,h_hann_31_M,2);
plot2d(fr,h_hann_61_M,5);
legend(['Length M = 31';'Length M = 61']);
title('Frequency Response Of Hanning window')
subplot(2,1,2)
[h_hamm_31_M,fr]=frmag(h_hamm_31,512);
[h_hamm_61_M,fr]=frmag(h_hamm_61,512);
h_hamm_31_M = 20*log10(h_hamm_31_M./max(h_hamm_31_M)
   );
h_hamm_61_M = = 20*log10(h_hamm_61_M./max(
   h_hamm_61_M));
plot2d(fr,h_hamm_31_M,2);
plot2d(fr,h_hamm_61_M,5);
legend(['Length M = 31';'Length M = 61']);
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

```
//Figure 8.6 and Figure 8.7
//Program to frequency response of
```

```
//(1) Hanning window (2)Hamming window for M = 31
clear all;
close;
clc
M = 31;
for n = 1:M
  h_hann_31(n) = 0.5-0.5*cos(2*%pi*(n-1)/(M-1));
  h_hamm_31(n) = 0.54-0.46*cos(2*%pi*(n-1)/(M-1));
end
subplot(2,1,1)
[h_hann_31_M,fr]=frmag(h_hann_31,512);
h_hann_31_M = 20*log10(h_hann_31_M./max(h_hann_31_M)
   );
plot2d(fr,h_hann_31_M);
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response 0f Hanning window M = 31')
subplot(2,1,2)
[h_hamm_31_M,fr]=frmag(h_hamm_31,512);
h_hamm_31_M = 20*log10(h_hamm_31_M./max(h_hamm_31_M)
   );
plot2d(fr,h_hamm_31_M);
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
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

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

```
Ws = 0.31;   //Stopband Edge Frequency
hn=eqfir(M,[0 Wp;Ws .5],[1 0],[2 1]);
[hm,fr]=frmag(hn,256);
disp('The LPF Filter Coefficients are:')
hn
//Obtaining Polyphase Filter Coefficients from hn
p = zeros(D,M/D);
for k = 1:D
  for n = 1:(length(hn)/D)
    p(k,n) = hn(D*(n-1)+k);
  end
end
disp('The Polyphase Decimator for D =2 are:')
p
figure
plot(fr,hm)
xlabel('Normalized Digital Frequency fr');
ylabel('Magnitude');
title('Frequency Response of FIR LPF using REMEZ algorithm M=61')
figure
plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
xlabel('Normalized Digital Frequency fr');
ylabel('Magnitude in dB');
title('Frequency Response of DECIMATOR (D=2) using REMEZ algorithm M=30')
```

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

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

```
//magnitude for pass band = 1 & stop band = 0 (i.e)
   [1 0]
//Weighting function =[3 1]
clear all;
clc;
close;
M = 30;  //Filter Length
I = 5;  //Interpolation Factor = 5
Wc = %pi/5;  //Cutoff Frequency
Wp = Wc/(2*%pi);  //Passband Edge Frequency
Ws = 0.16;  //Stopband Edge Frequency
hn=eqfir(M,[0 Wp;Ws .5],[1 0],[3 1]);
[hm,fr]=frmag(hn,256);
disp('The LPF Filter Coefficients are:')
hn
//Obtaining Polyphase Filter Coefficients from hn
p = zeros(I,M/I);
for k = 1:I
  for n = 1:(length(hn)/I)
    p(k,n) = hn(I*(n-1)+k);
  end
end
disp('The Polyphase Interpolator for I =5 are:')
p
figure
plot(fr,hm)
xlabel('Normalized Digital Frequency fr');
ylabel('Magnitude');
title('Frequency Response of FIR LPF using REMEZ
   algorithm M=61')
figure
plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
xlabel('Normalized Digital Frequency fr');
ylabel('Magnitude in dB');
title('Frequency Response of INTERPOLATOR(I=5) using
    REMEZ algorithm M=30')
```

Scilab code Exa 10.6.1 Multistage Implementation of Sampling Rate Conversion

```
//Example 10.6.1
//Multistage Implementation of Sampling Rate Conversion
//Decimation factor D = 50
//D = D1xD2, D1 = 25, D2 =2
clear all;
clc;
close;
Fs = 8000;  //Sampling Frequency = 8000Hz
Fpc = 75;  //Passband Frequency
Fsc = 80;  //Stopband Frequency
Delta_F = (Fsc-Fpc)/Fs;  //Transition Band
Pass_Band = [0,Fpc];
Transition_Band = [Fpc,Fsc];
Delta1  = (10^-2);  //Passband Ripple
Delta2 = (10^-4);  //Stopband Ripple
D = Fs/(2*Fsc);    //Decimation Factor
//Decimator Implemented in Two Stages
D1 = D/2;  //Decimator 1
D2 = 2;  //Decimator 2
//Decimator Single Stage Implementation
M = ((-10*log10(Delta1*Delta2)-13)/(14.6*Delta_F))+1;
M = ceil(M)
//Decimator Multistage Implementation
//First Stage Implementation
F1 = Fs/D1;  //New passband for stage1
Fsc1 = F1-Fsc;  //New Stopband for stage1
Delta_F1 = (Fsc1-Fpc)/Fs  //New Transition for stage1
Delta11 = Delta1/2;  //New Passband Ripple
Delta21  = Delta2;  //Stopband Ripple same
M1 = ((-10*log10(Delta11*Delta21)-13)/(14.6*Delta_F1))+1
M1 = floor(M1)
```

```
//Second Stage Implementation
F2 = F1/D2;  //New passband for stage2
Fsc2 = F2-Fsc;  //New Stopband for stage2
Delta_F2 = (Fsc2-Fpc)/F1  //New Transition for stage2
Delta12 = Delta1/2;  //New Passband Ripple
Delta22  = Delta2;  //Stopband Ripple same
M2 = ((-10*log10(Delta12*Delta22)-13)/(14.6*Delta_F2))+1
M2 = floor(M2)
disp('The Filter length Required in Single stage Implementation of Decimator is:')
M
disp('The Filter length Required in Multistage Implementation of Decimator is:')
M1+M2
//Calculation of Reduction Factor
R = M/(M1+M2);
disp('The Reduction in Filter Length is:')
R
```

Scilab code Exa 10.8.1 Signal to Distortion Ratio

```
//Example 10.8.1
//Signal to Distortion Ratio
//Calculation of no. of subfilters
clear all;
clc;
close;
SDR_dB = 50; //Signal to distortion ratio = 50 dB
Wx = 0.8*%pi; //Digital maximum frequency of input data
SDR = 10^(SDR_dB/10)
disp('The Number of subfilters required')
I = Wx*sqrt(SDR/12);
I = ceil(I)
```

Scilab code Exa 10.8.2 Signal to Distortion Ratio using Linear Interpolation

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

Scilab code Exa 10.9.1 Multistage Implementation of Sampling Rate Conversion

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

```
Delta2 = (10^-4);   //Stopband Ripple
D = Fs/(2*Fsc);     //Decimation Factor
//Decimator Implemented in Two Stages
D1 = D/2;   //Decimator 1
D2 = 2;   //Decimator 2
//Decimator Single Stage Implementation
M = ((-10*log10(Delta1*Delta2/2)-13)/(14.6*Delta_F))+1;
M = ceil(M)
//Decimator Multistage Implementation
//First Stage Implementation
Delta_F1 = 0.020625 //Obtained from Example 10.6.1
M1 = ((-10*log10(Delta1*Delta2/4)-13)/(14.6*Delta_F1))+1
M1 = floor(M1)
//Second Stage Implementation
Delta_F2 = 0.015625  //Obtained from Example 10.6.1
M2 = ((-10*log10(Delta1*Delta2/4)-13)/(14.6*Delta_F2))+1
M2 = floor(M2)
disp('The Filter length Required in Single stage Implementation of Decimator is:')
M
disp('The Filter length Required in Multistage Implementation of Decimator is:')
M1+M2
//Calculation of Reduction Factor
R = M/(M1+M2);
disp('The Reduction in Filter Length is:')
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

```
//Example 11.6.1
//Design of wiener filter of Length M =2
clear all;
close;
clc;
M =2;  //Wiener Filter Length
Rdx = [0.6 2 0.6] //Cross correlation matrix between
    the desired input sequence and actual input
   sequence
C = Rdx(M:$) //Right sided sequence
To_M = toeplitz(C)
Rxx = [0.6 1 0.6] //Auto correlation matrix
Rss = Rxx(M:$)
//Filter coefficients
h = [0.451 0.165]
//Calculation of Minimum Mean Square Error
sigma_d = 1; //Average power of desired sequence
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

```
//Example 12.1.1
//Determination of spectrum of a signal
//With maximum normalized frequency f = 0.1
//using Rectangular window and Blackmann window
clear all;
close;
clc;
N = 61;
cfreq = [0.1 0];
[wft,wfm,fr]=wfir('lp',N,cfreq,'re',0);
wft;                    // Time domain filter
   coefficients
wfm;                    // Frequency domain filter
   values
fr;                     // Frequency sample points
WFM_dB = 20*log10(wfm);//Frequency response in dB
for n = 1:N
```

```
 h_balckmann(n)=0.42-0.5*cos(2*%pi*n/(N-1))+0.08*cos(4*%pi*n/(N-1));
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 Using N-point DFT

```
//Example 12.1.2
//Evaluating power spectrum of a discrete sequence
//Using N-point DFT
clear all;
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;
```

```
  Pxx2_fk3(i) = (Pxx1_fk3(i)^2)/N;
end
for i =1:length(fk4)
  Pxx1_fk4(i) = 0;
  Pxx2_fk4(i) = 0;
  for m = 1:N
    Pxx1_fk4(i) =Pxx1_fk4(i)+x1(m)*exp(-sqrt(-1)*2*
       %pi*(m-1)*fk4(i));
    Pxx2_fk4(i) =Pxx1_fk4(i)+x1(m)*exp(-sqrt(-1)*2*
       %pi*(m-1)*fk4(i));
  end
  Pxx1_fk4(i) = (Pxx1_fk4(i)^2)/N;
  Pxx2_fk4(i) = (Pxx1_fk4(i)^2)/N;
end
figure
title('for frequency separation = 0.06')
subplot(2,2,1)
plot2d3('gnn',k1,abs(Pxx1_fk1))
subplot(2,2,2)
plot2d3('gnn',k2,abs(Pxx1_fk2))
subplot(2,2,3)
plot2d3('gnn',k3,abs(Pxx1_fk3))
subplot(2,2,4)
plot2d3('gnn',k4,abs(Pxx1_fk4))
figure
title('for frequency separation = 0.01')
subplot(2,2,1)
plot2d3('gnn',k1,abs(Pxx2_fk1))
subplot(2,2,2)
plot2d3('gnn',k2,abs(Pxx2_fk2))
subplot(2,2,3)
plot2d3('gnn',k3,abs(Pxx2_fk3))
subplot(2,2,4)
plot2d3('gnn',k4,abs(Pxx2_fk4))
```

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

```
//Example 12.5.1
//Determination of power, frequency and varaince of
//Additive noise
clear all;
clc;
close;
ryy = [0,1,3,1,0];//Autocorrelation of signal
cen_ter_value = ceil(length(ryy)/2);//center value of autocorrelation
//Method1
//TO find out the variance of the additive Noise
C = ryy(ceil(length(ryy)/2):$);
corr_matrix = toeplitz(C);//correlation matrix
evals =spec(corr_matrix);//Eigen Values computation
sigma_w = min(evals);//Minimum of eigen value = varinace of noise
//Method2
//TO find out the variance of the additive Noise
P = [1,-sqrt(2),1];//Ploynomial in decreasing order
Z = roots(P);//roots of the polynomial
P1 = ryy(cen_ter_value+1)/real(Z(1));//power of the sinusoid
A = sqrt(2*P1);//amplitude of the sinusoid
sigma_w1 = ryy(cen_ter_value)-P1;//variance of noise method2
disp(P1,'Power of the additive noise')
f1 = acos(real(Z(1)))/(2*%pi)
disp(f1,'frequency of the additive noise')
disp(sigma_w1,'Variance of the additive noise')
```

Appendix to Examples

Scilab code AE 4.2.7 Sampling a Nonbandlimted signal

```
//Example 4.2.7 Sampling a Nonbandlimited Signal
//Plotting Discrete Time Fourier Transform of
//Discrete Time Signal x(nT)= exp(−A∗T∗abs(n))
clear all;
clc;
close;
// Analog Signal
A =1;     //Amplitude
Dt = 0.005;
t = -2:Dt:2;
//Continuous Time Signal
xa = exp(-A*abs(t));
//Discrete Time Signal
Fs =input('Enter the Sampling Frequency in Hertz');
    //Fs = 1Hz(or)20Hz
Ts = 1/Fs;
n = -5:1:5;
nTs = n*Ts;
x = exp(-A*abs(nTs));
// Analog Signal reconstruction
Dt = 0.005;
t = -2:Dt:2;
Xa = x *sinc_new(Fs*(ones(length(nTs),1)*t-nTs'*ones
    (1,length(t))));
// check
error = max(abs(Xa - xa))
```

```
subplot(2,1,1);
a =gca();
a.x_location = "origin";
a.y_location = "origin";
plot(t,xa);
xlabel('t in msec.');
ylabel('xa(t)')
title('Original Analog Signal')
subplot(2,1,2);
a =gca();
a.x_location = "origin";
a.y_location = "origin";
xlabel('t in msec.');
ylabel('xa(t)')
title('Reconstructed Signal from x(n) using sinc
    function');
plot(t,Xa);
```

*Refer to the following for Scilab code of sinc
ARC 4A

Scilab code ARC 4A sinxbyx

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

Scilab code AE 4.4.2 Frequency Response

```
clear all;
close;
clc;
```

```
W = -%pi:(1/500):%pi;
z = exp(sqrt(-1)*W);
H = z./(z-0.8);
Mag_H = abs(H);
[Phase_H,m] = phasemag(H);
//phasemag used to calculate phase and magnitude in
   dB
subplot(2,1,1)
plot2d(W,Mag_H)
xlabel('Frequency in Radians')
ylabel('abs(H)')
title('Magnitude Response')
subplot(2,1,2)
plot2d(W,Phase_H)
xlabel('Frequency in Radians')
ylabel('<(H)')
title('Phase Response')
```

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

```
//PROGRAM TO DESIGN AND OBTAIN THE FREQUENCY
   RESPONSE OF FIR FILTER
//Band PASS FILTER
clear all;
clc;
close;
M = 11               //Filter length = 11
Wc = [%pi/4,3*%pi/4];          //Digital Cutoff
   frequency
Wc2 = Wc(2)
Wc1 = Wc(1)
Tuo = (M-1)/2      //Center Value
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 AE 8.2.28B DESIGN AND OBTAIN THE FREQUENCY RESPONSE OF FIR FILTER Band Stop

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

```
//Band Stop FILTER (or)Band Reject Filter
clear all;
clc;
close;
M = 11               //Filter length = 11
Wc = [%pi/4,3*%pi/4];          //Digital Cutoff frequency
Wc2 = Wc(2)
Wc1 = Wc(1)
Tuo = (M-1)/2      //Center Value
hd = zeros(1,M);
W = zeros(1,M);
for n = 1:11
 if (n == Tuo+1)
  hd(n) = 1-((Wc2-Wc1)/%pi);
 else    hd(n)=(sin(%pi*((n-1)-Tuo))-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 Of 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 Of FIR BPF using
    Rectangular window M=11')
```

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

```
//Figure 8.9 and 8.10
//PROGRAM TO DESIGN AND OBTAIN THE FREQUENCY
    RESPONSE OF FIR FILTER
//LOW PASS FILTER
clear all;
clc;
close;
M = 61              //Filter length = 61
Wc = %pi/5;         //Digital Cutoff frequency
Tuo = (M-1)/2      //Center Value
for n = 1:M
    if (n == Tuo+1)
      hd(n) = Wc/%pi;
    else
      hd(n) =  sin(Wc*((n-1)-Tuo))/(((n-1)-Tuo)*%pi)
         ;
  end
end
//Rectangular Window
for n = 1:M
  W(n) = 1;
end
//Windowing Filter 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(fr,hzm)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude');
title('Frequency Response 0f FIR LPF using Rectangular window M=61')
subplot(2,1,2)
plot(fr,hzm_dB)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response 0f FIR LPF using Rectangular window M=61')
```

Scilab code AE 8.3.5 High Pass Filter

```
//Example 8.3.5
//First Order Butterworth Filter
//Low Pass Filter
clear all;
clc;
close;
s = poly(0,'s');
Omegac = 0.2*%pi;
H = Omegac/(s+Omegac);
T =1;//Sampling period T = 1 Second
z = poly(0,'z');
Hz = horner(H,(2/T)*((z-1)/(z+1)))
HW =frmag(Hz(2),Hz(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 Single pole LPF Filter Cutoff frequency = 0.2*pi','Digital Frequency --->','Magnitude');
```

Scilab code AE 8.3.6 Analog Low Pass

```
//Example 8.3.6
// To Design an Analog Low Pass IIR Butterworth Filter
//For the given cutoff frequency Wc = 500 Hz
clear all;
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 AE 8.4.1 High Pass Filter

```
//Example 8.3.5
//First Order Butterworth Filter
//High Pass Filter
//Table 8.13: Using Digital Filter Transformation
clear all;
clc;
close;
s = poly(0,'s');
Omegac = 0.2*%pi;
H = Omegac/(s+Omegac);
T =1;//Sampling period T = 1 Second
z = poly(0,'z');
Hz_LPF = horner(H,(2/T)*((z-1)/(z+1)));
alpha = -(cos((Omegac+Omegac)/2))/(cos((Omegac-
    Omegac)/2));
HZ_HPF=horner(H_LPF,-(z+alpha)/(1+alpha*z))
HW  =frmag(HZ_HPF(2),HZ_HPF(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 Single pole HPF Filter
     Cutoff frequency = 0.2*pi','Digital Frequency
    ———>','Magnitude');
```

Scilab code AE 8.4.2A Analog Filter Transformation

```
//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 BSF USING DIGITAL
    TRANSFORMATION
clear all;
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 AE 8.4.2B Digital Filter Transformation

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

```
Ol = sym('Ol');  //lower cutoff frequency of band pass filter
Ou = sym('Ou'); //upper cutoff frequency of band pass filter
s1 = Op*(s^2+Ol*Ou)/(s*(Ou-Ol)); //Analog transformation for LPF to BPF
H_Lpf = Op/(s+Op); //single pole analog LPF Butterworth filter
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)
```