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

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

**Prb** Problem (Unsolved problem)

**Exa** Example (Solved example)

**Equ** 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 Appendix 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
   10.1 Scilab Code ........................................ 52

11 Linear Predictions and Optimum Linear Filter
   11.1 Scilab Code ........................................ 59

12 Power Spectrum Estimation
   12.1 Scilab Code ........................................ 60
# List of Scilab Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eqn 1.2.1</td>
<td>Discrete time signal</td>
<td>2</td>
</tr>
<tr>
<td>Eqn 2.1.6</td>
<td>Unit sample sequence</td>
<td>3</td>
</tr>
<tr>
<td>Eqn 2.1.7</td>
<td>Unit step sequence</td>
<td>3</td>
</tr>
<tr>
<td>Eqn 2.1.8</td>
<td>Unit ramp sequence</td>
<td>4</td>
</tr>
<tr>
<td>Eqn 2.1.9a</td>
<td>Exponential sequence</td>
<td>4</td>
</tr>
<tr>
<td>Eqn 2.1.9b</td>
<td>Exponential increasing sequence</td>
<td>5</td>
</tr>
<tr>
<td>Eqn 2.1.9c</td>
<td>Exponential decreasing sequence</td>
<td>5</td>
</tr>
<tr>
<td>Eqn 2.1.24</td>
<td>Even signal</td>
<td>6</td>
</tr>
<tr>
<td>Eqn 2.1.25</td>
<td>Odd signal</td>
<td>6</td>
</tr>
<tr>
<td>Exa 3.1.1</td>
<td>Direct Ztransform</td>
<td>8</td>
</tr>
<tr>
<td>Exa 3.1.2</td>
<td>Z transform</td>
<td>9</td>
</tr>
<tr>
<td>Exa 3.1.4</td>
<td>Z transform</td>
<td>9</td>
</tr>
<tr>
<td>Exa 3.1.5</td>
<td>Z transform</td>
<td>10</td>
</tr>
<tr>
<td>Exa 3.2.1</td>
<td>Z transform</td>
<td>10</td>
</tr>
<tr>
<td>Exa 3.2.2</td>
<td>Sinusoidal Signals</td>
<td>10</td>
</tr>
<tr>
<td>Exa 3.2.3</td>
<td>Time Shifting Property</td>
<td>11</td>
</tr>
<tr>
<td>Exa 3.2.4</td>
<td>Z transform</td>
<td>12</td>
</tr>
<tr>
<td>Exa 3.2.6</td>
<td>Z transform</td>
<td>12</td>
</tr>
<tr>
<td>Exa 3.2.7</td>
<td>Z transform</td>
<td>12</td>
</tr>
<tr>
<td>Exa 3.2.9</td>
<td>Convolution</td>
<td>13</td>
</tr>
<tr>
<td>Exa 3.2.10</td>
<td>Autocorrelation</td>
<td>13</td>
</tr>
<tr>
<td>ARC 3A</td>
<td>Ztransfer</td>
<td>14</td>
</tr>
<tr>
<td>Exa 4.1.2</td>
<td>Fourier Transform</td>
<td>15</td>
</tr>
<tr>
<td>Exa 4.2.7</td>
<td>Continuous Time Fourier Transform</td>
<td>16</td>
</tr>
<tr>
<td>Exa 4.3.4</td>
<td>Convolution Property</td>
<td>17</td>
</tr>
<tr>
<td>Exa 4.4.2</td>
<td>Frequency Response</td>
<td>19</td>
</tr>
<tr>
<td>Exa 4.4.4</td>
<td>Frequency Response</td>
<td>20</td>
</tr>
<tr>
<td>Exa 5.1.2</td>
<td>Discrete Fourier Transform</td>
<td>22</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>Exa 12.5.1 Additive Noise Parameters</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>AE 4.2.7 Discrete Time Fourier Transform</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>ARC 4A sinc</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>AE 4.4.2 Frequency response</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>AE 8.2.28A Design of Filter</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>AE 8.2.28B Design of Filter</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>AE 8.2.28C Design of Filter</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>AE 8.3.5 First Order Butterworth Filter</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>AE 8.3.6 IIR Butterworth Filter</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>AE 8.4.1 First Order Butterworth Filter</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>AE 8.4.2A IIR Filter Design Butterworth Filter</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>AE 8.4.2B IIR Filter Design Butterworth Filter</td>
<td>74</td>
<td></td>
</tr>
</tbody>
</table>
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

```matlab
// Implementation of Equation 1.2.1 in Chapter 1
// 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

1  //Implementation of Equation 2.1.6 in Chapter 2
3  //Page 45

4  clear; clc; close;

5  L = 4;  //Upper limit
6  n = -L:L;
7  x = [zeros(1,L),1,zeros(1,L)];
8  a=gca();
9  a.thickness = 2;
10  a.y_location = ”middle”;
11  plot2d3(’gnn’,n,x)
12  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
3 // Page 45
4
5 clear; clc; close;
6 L = 4; // Upper limit
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
3 // Page 45
4
5 clear; clc; close;
6 L = 4; // Upper limit
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
Scilab code Eqn 2.1.9b Exponential increasing sequence

Scilab code Eqn 2.1.9c Exponential decreasing sequence
1 // Implementation of Equation 2.1.9c in Chapter 2
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
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
// Implementation of Equation 2.1.25 in Chapter 2

// 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

```plaintext
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;
```
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 \( x(n) = 0.5^n u(n) \)

```plaintext
// 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 \( x(n) = \alpha^n \)

```plaintext
// Example 3.1.4
// Z transform of x[n] = -\alpha^n . u[-n-1]
// \alpha = 0.5
clear all;
close;
clc;
```

9
Scilab code Exa 3.1.5 Z transform of $x(n) = a^n u(n) + b^n u(-n - 1)$

```scilab
// 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;
cclc;
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 $x(n) = 3.2^n u(n) - 4.3^n u(n)$

```scilab
// Example 3.2.1
// Z transform of \( x[n] = 3.2^n u[n] - 4.3^n u[n] \)
clear all;
close;
cclc;
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(W0.n).u[n]
// Z transform of y[n] = sin(W0.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

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 $x(n) = u(n)$

```scilab
// 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 $x(n) = u(-n)$

```scilab
// 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 $x(n) = n.a^n.u(n)$

```scilab
// 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

```matlab
// 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

```matlab
// 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** Ztransfer of a sequence

```scilab
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

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

*For further extension of the example 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 = [-mtlb_fliplr(w), w(2:K+1)]; // Omega from -w to w
17 X1 = [mtlb_fliplr(X1), X1(2:K+1)];
18 X2 = [mtlb_fliplr(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*mtlb_fliplr(w), 2*w];
26 X = [mtlb_fliplr(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
Scilab code Exa 4.4.2 Frequency Response of Three point Moving Average System \( y(n) = \frac{1}{3}[x(n+1) + x(n) + x(n-1)] \)

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 example 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

```scilab
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;
```
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)
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

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

27
// 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)
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

```scilab
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 \( y(n) = ay(n-1) + x(n); a = (1/2) anda = (3/4) \)

```scilab
1  // Equation 7.7.1
2  // Program to find Dead band of First order Recursive System
```
// 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); // pole 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
\[ \sigma = \left( \frac{2}{1 - a_1^2} \right) - \left( \frac{2}{1 - a_1a_2} \right) + \left( \frac{1}{1 - a_2^2} \right) \sigma_e^2 \]

\[ \text{noise_variance} = \sigma + \sigma_2 \]

---

**Scilab code Eqn 7.7.40 Signal to Quantization Noise Ratio**

1. // Equation 6.4.17
2. // page 492
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 = \( \frac{2}{3} \times 2^{-2b} \)
11. Signal_Power = \( \frac{1}{3N} \)
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

```
// Example 8.2.1
// Design of FIR Filter using Frequency 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))*cos((n-1)+(1/2))+h1;
    end
```

```
\[
\begin{align*}
\text{h}(n) &= (1/M) \ast (G(1) + 2 \ast h1); \\
\text{end} \\
\text{h} \\
\text{[hzm, fr]} &= \text{frmag}(h, 256); \\
\text{hzm}\_\text{dB} &= 20 \ast \log_{10}(\text{hzm})/\max(\text{hzm}); \\
\text{figure} \\
\text{plot}(2 \ast \text{fr}, \text{hzm}) \\
\text{a}=\text{gca}(); \\
\text{xlabel}('\text{Normalized Digital Frequency W}'); \\
\text{ylabel}('\text{Magnitude}'); \\
\text{title}('\text{Frequency Response of FIR LPF using Frequency Sampling Technique with M = 15 with Cutoff Frequency = 0.466}') \\
\text{xgrid}(2) \\
\text{figure} \\
\text{plot}(2 \ast \text{fr}, \text{hzm}\_\text{dB}) \\
\text{a}=\text{gca}(); \\
\text{xlabel}('\text{Normalized Digital Frequency W}'); \\
\text{ylabel}('\text{Magnitude in dB}'); \\
\text{title}('\text{Frequency Response of FIR LPF using Frequency Sampling Technique with M = 15 with Cutoff Frequency = 0.466}') \\
\text{xgrid}(2)
\end{align*}
\]

**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, T1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
10. for k = 1:length(Hr)
\( G(k) = ((-1)^{(k-1)}) \cdot Hr(k); \)

\[ \text{end} \]

\( h = \text{zeros}(1,M); \)

\( U = (M-1)/2 \)

\[ \text{for } n = 1:M \]
\[ \quad h1 = 0; \]
\[ \quad \text{for } k = 2:U+1 \]
\[ \quad \\ h1 = G(k) \cdot \cos \left( \frac{2\pi}{M} \cdot (k-1) \cdot ((n-1)+(1/2)) \right) + h1; \]
\[ \text{end} \]
\[ \quad h(n) = \frac{1}{M} \cdot (G(1) + 2 \cdot h1); \]
\[ \text{end} \]

\( h \)

\[ [hzm,fr]=\text{frmag}(h,256); \]

\( hzm\_dB = 20 \cdot \log_{10}(hzm)/\max(hzm); \)

\[ \text{figure} \]
\[ \text{plot}(2*fr,hzm) \]
\[ \text{xlabel('Normalized Digital Frequency W');} \]
\[ \text{ylabel('Magnitude');} \]
\[ \text{title('Frequency Response of FIR LPF using Frequency Sampling Technique with M = 15 with Cutoff Frequency = 0.466');} \]

\[ \text{xgrid(2)} \]

\[ \text{figure} \]
\[ \text{plot}(2*fr,hzm\_dB) \]
\[ \text{xlabel('Normalized Digital Frequency W');} \]
\[ \text{ylabel('Magnitude in dB');} \]
\[ \text{title('Frequency Response of FIR LPF using Frequency Sampling Technique with M = 15 with Cutoff Frequency = 0.466');} \]

\[ \text{xgrid(2)} \]

---

**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)
7  // Weighting function = [1 1]
8  clear all;
9  clc;
10  close;
11  hn=eqfir(61,[0.1;0.15;0.5],[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]
// Example 8.2.5
// Linear Phase FIR Differentiator of length M = 60
// Pass Band Edge frequency fp = 0.1

clear all;
clc;
close;

hn = eqfir(32, [0.1; 0.2; 0.35; 0.425; 0.5], [0 1 0 1 0 1 0], [10 1 10]);

hm, fr = frmag(hn, 256);
disp('The Filter Coefficients are:
');

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
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 Hilbert Transformer of Length M = 31
// Default Window Rectangular Window
// Chebyshev approx default parameter = [0 0]
clear all;
clc;
close;
M = 31; // Hilbert 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
The Hilbert Coefficients are:

```matlab
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 generation
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 Hilbert 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 Hilbert 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

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    end
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)
plot(fr,hzm_dB)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response of FIR LPF using Rectangular window M=61')

*For further extension of the example refer to
AE 8.2.28A  AE 8.2.28B  AE 8.2.28C

---

**Scilab code Exa 8.3.2** Backward Difference

```scilab
// 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

```scilab
// 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
```
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
For further extension of the example refer to AE 8.3.5

Scilab code Exa 8.3.6 Analog Filter Transformation

```plaintext
// 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 = $10^{\text{delta2\_in\_dB}/20}$

// Calculation of Filter Order
N = $\log_{10}((1/(\text{delta2}^2))^{-1})/(2*\log_{10}(\text{omegas}/\text{omegap}))$
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)
xtitile('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 example 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/delta1)^2)-1)
14 epsilon = sqrt(((1/delta2)^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] = zpch1(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)
//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

1 //Caption: Converting 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

// 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))*(z^2));
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) =
//
// \[ 2 3 \]
// \[ 4 \]
// \[ 0.245 - 1.577D-17z - 0.245z + 1.577D-17z + 1.360D-17z \]
//
// which is equivalent to
// H(z) =
//
// \[ 2 3 \]
// \[ 4 \]
// \[ -0.509 + 1.299D-16z - z + 6.438D-17z + 5.551D \]
\[ -17z \]
//
// H(z) =
//
// \[ 2 \]
// \[ 0.245 - 0 - 0.245z + 0 + 0 \]
//
// H(z) =
//
// \[ 2 \]
// \[ 0.245 - 0.245z \]
//
// H(z) =
//
// \[ 2 \]
// \[ -0.509 - z \]
//
// H(z) =
//
// \[ -2 \]
// \[ 0.245 - 0.245z \]
//
// H(z) =
//
// \[ -2 \]
// \[ 0.509+z \]
Scilab code CF 8.5 Program to generate different window functions

```scilab
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 frequency response of (1) Hanning window (2)Hamming window for M = 31 and M = 61

```scilab
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
```
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 of 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

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
5  //edge frequency fs = 0.31
6  // Choose the number of cosine functions and create
7  // a dense grid
8  // in [0,0.25] and [0.31,0.5]
9  //magnitude for pass band = 1 & stop band = 0 (i.e)
10  [1 0]
11  //Weighting function = [2 1]
12  clear all;
13  clc;
14  close;
15  M = 30;  //Filter Length
16  D = 2;  //Decimation Factor = 2
17  Wc = %pi/2;  //Cutoff Frequency
18  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)
1 //magnitude for pass band = 1 & stop band = 0 (i.e) [1 0]
2 //Weighting function = [3 1]
3 clear all;
4 clc;
5 close;
6 M = 30; // Filter Length
7 I = 5; // Interpolation Factor = 5
8 Wc = %pi /5; // Cutoff Frequency
9 Wp = Wc /(2*%pi); // Passband Edge Frequency
10 Ws = 0.16; // Stopband Edge Frequency
11 hn = eqfir(M,[0 Wp;Ws .5],[1 0],[3 1]);
12 [hm,fr] = frmag(hn,256);
13 disp(’The LPF Filter Coefficients are:’)
14 hn
15 // Obtaining Polyphase Filter Coefficients from hn
16 p = zeros(I,M/I);
17 for k = 1:I
18    for n = 1:(length(hn)/I)
19       p(k,n) = hn(I*(n-1)+k);
20    end
21 end
22 disp(’The Polyphase Interpolator for I =5 are:’) 
23 p
24 figure
25 plot(fr,hm)
26 xlabel(’Normalized Digital Frequency fr’);
27 ylabel(’Magnitude’);
28 title(’Frequency Response of FIR LPF using REMEZ algorithm M=61’)
29 figure
30 plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
31 xlabel(’Normalized Digital Frequency fr’);
32 ylabel(’Magnitude in dB’);
33 title(’Frequency Response of INTERPOLATOR(I=5) using REMEZ algorithm M=30’)
Scilab code Exa 10.6.1  Multistage Implementation of Sampling Rate Conversion

```matlab
// 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)
```

55
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)
disp('The Filter length Required in Single stage Implementation of Decimator is: ')
40 M
41 disp('The Filter length Required in Multistage Implementation of Decimator is: ')
42 M1+M2
43 // Calculation of Reduction Factor
44 R = M/(M1+M2);
disp('The Reduction in Filter Length is: ')
45 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)
disp('The Number of subfilters required ')
10 I = Wx*sqrt(SDR/12);
11 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
\begin{verbatim}
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
\end{verbatim}
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

```scilab
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

```scilab
// 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);
// Time domain filter coefficients
wft;
// Frequency domain filter values
wfm;
// Frequency sample points
fr;
WFM_dB = 20*log10(wfm); // Frequency response in dB
for n = 1:N
```

60
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);

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;
close;
clore;
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;
17 $f_{k1} = k_1 ./ L(1)$;
18 $f_{k2} = k_2 ./ L(2)$;
19 $f_{k3} = k_3 ./ L(3)$;
20 $f_{k4} = k_4 ./ L(4)$;
21 for $i = 1$: length($f_{k1}$)
22    $P_{xx1_{f_{k1}}}(i) = 0$;
23    $P_{xx2_{f_{k1}}}(i) = 0$;
24    for $m = 1$: $N$
25        $P_{xx1_{f_{k1}}}(i) = P_{xx1_{f_{k1}}}(i) + x_1(m) * \exp(-\text{s}qrt(-1) * 2 * \%pi * (m-1) * f_{k1}(i))$;
26        $P_{xx2_{f_{k1}}}(i) = P_{xx1_{f_{k1}}}(i) + x_1(m) * \exp(-\text{s}qrt(-1) * 2 * \%pi * (m-1) * f_{k1}(i))$;
27    end
28 $P_{xx1_{f_{k1}}}(i) = (P_{xx1_{f_{k1}}}(i)^2)/N$;
29 $P_{xx2_{f_{k1}}}(i) = (P_{xx2_{f_{k1}}}(i)^2)/N$;
30 end
31 for $i = 1$: length($f_{k2}$)
32    $P_{xx1_{f_{k2}}}(i) = 0$;
33    $P_{xx2_{f_{k2}}}(i) = 0$;
34    for $m = 1$: $N$
35        $P_{xx1_{f_{k2}}}(i) = P_{xx1_{f_{k2}}}(i) + x_1(m) * \exp(-\text{s}qrt(-1) * 2 * \%pi * (m-1) * f_{k2}(i))$;
36        $P_{xx2_{f_{k2}}}(i) = P_{xx1_{f_{k2}}}(i) + x_1(m) * \exp(-\text{s}qrt(-1) * 2 * \%pi * (m-1) * f_{k2}(i))$;
37    end
38 $P_{xx1_{f_{k2}}}(i) = (P_{xx1_{f_{k2}}}(i)^2)/N$;
39 $P_{xx2_{f_{k2}}}(i) = (P_{xx2_{f_{k2}}}(i)^2)/N$;
40 end
41 for $i = 1$: length($f_{k3}$)
42    $P_{xx1_{f_{k3}}}(i) = 0$;
43    $P_{xx2_{f_{k3}}}(i) = 0$;
44    for $m = 1$: $N$
45        $P_{xx1_{f_{k3}}}(i) = P_{xx1_{f_{k3}}}(i) + x_1(m) * \exp(-\text{s}qrt(-1) * 2 * \%pi * (m-1) * f_{k3}(i))$;
46        $P_{xx2_{f_{k3}}}(i) = P_{xx1_{f_{k3}}}(i) + x_1(m) * \exp(-\text{s}qrt(-1) * 2 * \%pi * (m-1) * f_{k3}(i))$;
47    end
48 $P_{xx1_{f_{k3}}}(i) = (P_{xx1_{f_{k3}}}(i)^2)/N$;
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) = Pxx2_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 variance of Additive noise
% Example 12.5.1
% Determination of power, frequency and variance of
% Additive noise
clear all;
clc;
close;
ryy = [0,1,3,1,0]; % Autocorrelation of signal
center_value = ceil(length(ryy)/2); % center value of autocorrelation
% Method 1
% 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 = variance of noise
% Method 2
% To find out the variance of the additive Noise
P = [1,-sqrt(2),1]; % Polynomial in decreasing order
Z = roots(P); % roots of the polynomial
P1 = ryy(center_value+1)/real(Z(1)); % power of the sinusoid
A = sqrt(2*P1); % amplitude of the sinusoid
sigma_w1 = ryy(center_value)-P1; % variance of noise method 2
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 Nonbandlimited signal

```scilab
// 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))
```

65
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  sinxbyx

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

Scilab code AE 4.4.2 Frequency Response

1 clear all;
2 close;
3 clc;
$W = -\pi : (1/500) : \pi ;$

$z = \exp (\sqrt{-1} * W) ;$

$H = z ./ (z - 0.8) ;$

$\text{Mag}_H = \text{abs}(H) ;$

$[\text{Phase}_H, m] = \text{phasemag}(H) ;$

// phasemag used to calculate phase and magnitude in dB

$\text{subplot}(2, 1, 1) ;$

$\text{plot2d}(W, \text{Mag}_H ) ;$

xlabel('Frequency in Radians ')

ylabel('abs(H) ')

title('Magnitude Response ')

$\text{subplot}(2, 1, 2) ;$

$\text{plot2d}(W, \text{Phase}_H ) ;$

xlabel('Frequency in Radians ')

ylabel('<(H) ')

$\text{title}(\text{Phase Response'})$

---

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;

$M = 11 ; // Filter length = 11$

$Wc = [\%\pi /4 , 3*\%\pi /4] ; // Digital Cutoff frequency$

$Wc_2 = Wc(2) ;$

$Wc_1 = Wc(1) ;$

$Tuo = (M-1)/2 ; // Center Value$

$hd = \text{zeros}(1, M) ;$

$W = \text{zeros}(1, M) ;$

for $n = 1:11$

if ($n == Tuo+1)$

$hd(n) = (Wc_2-Wc_1)/\%\pi ;$

end

end

67
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 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(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

68
clear all;
clc;
close;
M = 11  // Filter length = 11
Wc = [π/4, 3π/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)/π);
    else
        hd(n) = (sin(π*((n-1)-Tuo))-sin(Wc2*((n-1)-Tuo))+sin(Wc1*((n-1)-Tuo)))/(((n-1)-Tuo)*π);
    end
    if(abs(hd(n))<(0.00001))
        hd(n)=0;
    end
end
hd
// 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(2*fr, hzm)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude');
Scilab code AE 8.2.28C DESIGN AND OBTAIN THE FREQUENCY RESPONSE OF FIR FILTER

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/π;
13     else
14         hd(n) = sin(Wc*((n-1)-Tuo))/(((n-1)-Tuo)*π)
15     end
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;
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)
**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)
xtile('Magnitude Response of Butterworth LPF Filter
cutoff frequency = 500 Hz','Analog frequency in Hz')
```
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
3 Analog IIR Butterworth Filter
4 //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 : Converting single pole LPF Butterworth filter into BPF
// Exa 8.4.1
// page 698
clc;

Op = sym('Op'); // pass band edge frequency of low pass filter
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)