

Scilab Textbook Companion for  
Digital Signal Processing: A Computer Based  
Approach  
by S. K. Mitra<sup>1</sup>

Created by  
Sanjeev Irny  
B.Tech  
Instrumentation Engineering  
Shri Gurugobind Singhji Institute of Engg. & Tech.  
College Teacher  
Dr. B. M. Patre  
Cross-Checked by

May 24, 2016

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

# Book Description

**Title:** Digital Signal Processing: A Computer Based Approach

**Author:** S. K. Mitra

**Publisher:** Tata McGraw - Hill Education

**Edition:** 3

**Year:** 2008

**ISBN:** 978-0-07-066756-3

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

**Exa** Example (Solved example)

**Eqn** Equation (Particular equation of the above book)

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

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

# Contents

List of Scilab Codes	4
2 Discreet Time Signals and Systems	5
3 Discreet TIme Fourier Transform	19
4 Digital Processing of Continous TIme Systems	26
5 Finite Length Discreet Transform	30
6 $z$ Transform	42
7 LTI Discreet Time systems in the Transform Domain	58
8 Digital Filter Structures	61
9 IIR digital filter design	66
10 FIR digital filter design	71
11 DSP Algorithm implementation	77
12 Analysis of Finite Wordlength Effects	80
13 Multirate DIgital Signal Processing Findamentals	81
14 Applications of Digital Signal Processing	84

# List of Scilab Codes

Exa 2.1	Ensemble Averaging . . . . .	5
Exa 2.2	Basic operations . . . . .	6
Exa 2.3	Unequal length sequence . . . . .	6
Exa 2.5	Generating symmetric parts . . . . .	7
Exa 2.6	Energy Signal . . . . .	8
Exa 2.7	Power Signal . . . . .	9
Exa 2.9	Square wave generation . . . . .	9
Exa 2.16	Linearity of accumulator . . . . .	10
Exa 2.20	Passive system . . . . .	10
Exa 2.22	Impulse response of Accumulator . . . . .	11
Exa 2.26	Convolution . . . . .	12
Exa 2.27	Convolution . . . . .	12
Exa 2.28	Convolution . . . . .	13
Exa 2.29	Convolution . . . . .	14
Exa 2.30	Convolution . . . . .	15
Exa 2.31	Stability of causal system . . . . .	16
Exa 2.32	Stability of Anti causal system . . . . .	16
Exa 2.33	Stability of a system . . . . .	17
Exa 2.46	Cross correlation computation . . . . .	18
Exa 3.5	DTFT computation . . . . .	19
Exa 3.6	DTFT computation . . . . .	20
Exa 3.7	Plotting real and imaginary part . . . . .	21
Exa 3.10	DTFT of finite length exponential sequence . . . . .	22
Exa 3.12	Plotting DTFT of exponential sequence . . . . .	22
Exa 3.13	DTFT computation . . . . .	23
Exa 3.14	Energy of signal . . . . .	24
Exa 3.15	Energy of exponential sequence . . . . .	25
Exa 4.5	Passband and Stopband ripple computation . . . . .	26

Exa 4.6	Order of Analog filter . . . . .	26
Exa 4.7	Order of Analog Chebyshev Filter . . . . .	27
Exa 4.8	Order of Analog Lowpass Elliptic Filter . . . . .	28
Exa 4.16	Design of Analog Butterworth HP Filter . . . . .	28
Exa 5.1	DFT computation . . . . .	30
Exa 5.2	DFT of sinusoidal sequence . . . . .	30
Exa 5.3	DFT computation . . . . .	31
Exa 5.4	IDFT Computation . . . . .	32
Exa 5.5	DFT computation . . . . .	34
Exa 5.7	Circular convolution computation . . . . .	34
Exa 5.8	Circular convolution computation . . . . .	35
Exa 5.10	Generating symmetric parts . . . . .	36
Exa 5.11	Circular convolution computation . . . . .	37
Exa 5.12	Linear Convolution using DFT . . . . .	38
Exa 5.14	DFT computation using single DFT . . . . .	39
Exa 5.15	DFT computation using single DFT of shorter length . . . . .	39
Exa 6.1	$z$ Transform of causal exponential sequence . . . . .	42
Exa 6.2	$z$ transform of anticausal sequence . . . . .	42
Exa 6.3	$z$ Transform . . . . .	43
Exa 6.4	$z$ Transform . . . . .	43
Exa 6.5	$Z$ transform of causal sequence . . . . .	43
Exa 6.9	$z$ Transform . . . . .	44
Exa 6.10	Rational form of $z$ Transform from its zero and pole locations . . . . .	44
Exa 6.11	Inverse $z$ Transform . . . . .	45
Exa 6.12	Inverse $z$ Transform . . . . .	45
Exa 6.13	Proper fraction of Rational $z$ Transform . . . . .	46
Exa 6.14	Inverse $z$ Transform by partial fraction expansion . . . . .	47
Exa 6.15	residue computation using coefficient matching approach . . . . .	47
Exa 6.16	Inverse $z$ Transform by power series expansion . . . . .	48
Exa 6.17	Coefficients of rational form . . . . .	49
Exa 6.18	Inverse $z$ Transform using long division . . . . .	49
Exa 6.19	Inverse $z$ Transform using long division . . . . .	50
Exa 6.20	Inverse $z$ Transform . . . . .	50
Exa 6.22	$z$ Transform . . . . .	50
Exa 6.23	$z$ Transform . . . . .	51
Exa 6.24	sum of sequences of non overlapping ROC . . . . .	51
Exa 6.25	$z$ Transform . . . . .	52

Exa 6.26	z Transform . . . . .	52
Exa 6.27	Inverse z Transform . . . . .	53
Exa 6.28	Enlargement of ROC by pole zero cancellation . . . . .	54
Exa 6.30	Convolution . . . . .	54
Exa 6.31	Convolution . . . . .	55
Exa 6.33	Transfer Function of Moving Average Filter . . . . .	56
Exa 6.34	Transfer function determination . . . . .	57
Exa 7.1	Bounded real function . . . . .	58
Exa 7.2	Transfer function determination . . . . .	59
Exa 7.6	FIR Transfer function . . . . .	60
Exa 8.1	Analysis of Cascaded lattice digital filter structure . . . . .	61
Exa 8.6	Factorization of FIR Transfer Function . . . . .	61
Exa 8.7	Factorization of IIR Transfer Function . . . . .	62
Exa 8.10	Cascaded lattice realization of IIR digital Transfer Function . . . . .	62
Exa 8.12	Gray Markel method of realization . . . . .	63
Exa 8.18	Cascaded lattice realization . . . . .	64
Exa 9.1	Computating ripple values . . . . .	66
Exa 9.2	conversion of bandedged frequencies to Normalized digital frequencies . . . . .	66
Exa 9.3	Design of HP Digital Filter . . . . .	67
Exa 9.6	Changing passband edge frequencies to LP IIR digital frequencies . . . . .	68
Exa 9.7	Design of HP IIR Digital Filter from LP Digital Filter . . . . .	69
Exa 9.12	Minimum order of Type 2 Chebyshev HP IIR digital filter . . . . .	70
Exa 10.1	Kaiser formula . . . . .	71
Exa 10.2	Bellenger formula . . . . .	72
Exa 10.3	Hermann formula . . . . .	72
Exa 10.4	Order Estimation . . . . .	73
Exa 10.6	Filter length estimation for window based design . . . . .	74
Exa 10.7	Order Estimation . . . . .	75
Exa 10.8	Kaiser window . . . . .	75
Exa 11.3	Reconstruction of Transfer function from Impulse response coefficients . . . . .	77
Exa 11.11	Cascaded lattice Filter structure . . . . .	78
Exa 12.3	Signal to Quantisation Noise Ratio . . . . .	80
Exa 13.1	Up sampling operation . . . . .	81

Exa 13.2	Down sampling operation . . . . .	82
Exa 13.6	Decimator Computation complexity . . . . .	82
Exa 14.1	Effect of DFT length . . . . .	84
Exa 14.2	Effect of DFT length . . . . .	85



## Chapter 2

# Discreet Time Signals and Systems

Scilab code Exa 2.1 Ensemble Averaging

```
1 //EXAMPLE 2.1
2 //Ensemble averaging
3 clear ;
4 clc ;
5 n = 1:50;
6 clf();
7 figure(0)
8 a=gca();
9 a.x_location="origin";
10 a.y_location="origin";
11     for i =1:length(n)
12         s(i)=2*n(i)*((0.9)^n(i));
13         d(i)=(-0.1)^n(i);           //arbitrary noise
14         signal.
15     end
16 M=length(n);
17
18     for i =1:M
```

```

19     d(i)=(-0.1)^i;
20     S=sum(d);
21     end
22 Eav=(s+S/M)';           //Ensemble average.
23 disp(Eav,'The output of Ensemble averaging is')
24 plot2d3(n,s)
25 plot(n,s,'r. ')
26 xtitle('Ensemble averaging','n','Eav-s');
27 a.children.children.thickness=2;
28 a.children.children.foreground=2;

```

---

### Scilab code Exa 2.2 Basic operations

```

1 //EXAMPLE 2.2 , BASIC OPERATIONS.
2 clear;
3 clc;
4 c=[3.2 41 36 -9.5 0];
5 disp(c,'c = ');
6 d=[1.7 -0.5 0 0.8 1];
7 disp(d,'d = ');
8 w1=c.*d;           //Multiplication
9 disp(w1,'The product of two input vectors is =');
10 w2=c+d;           //addition
11 disp(w2,'The addition of two input vectors is =');
12 w3=3.5*c;
13 disp(w3,'The scaling of first input vector is =');

```

---

### Scilab code Exa 2.3 Unequal length sequence

```

1 //EXAMPLE 2.3 , Basic ops on unequal length sequence
2 clear;
3 clc;
4 c=[3.2 41 36 -9.5 0];

```

```

5 disp(c, 'c = ');
6 g=[-21 1.5 3];
7 disp(g, 'g = ');
8 a=length(g);
9 b=length(c);
10 i=0;
11     while(i<b-a)
12         g(b-i)=0;
13         i=i+1;
14     end
15 w4=g.*c;
16 disp(w4, 'The product of two sequences is =');
17 w5=c+g;
18 disp(w5, 'The addition of two sequences is =');

```

---

#### Scilab code Exa 2.5 Generating symmetric parts

```

1 //EXAMPLE 2.5, Conjugate–Antisymmetric & Conjugate–
  symmetric parts of Sequence
2 clc;
3 clear;
4 g=[0, 1+%i*4, -2+%i*3, 4-%i*2, -5-%i*6, -%i*2, 3];
5 disp(g, 'g = ')
6 g1=conj(g); //Conjugate of g;
7 disp(g1, conj(g));
8 a=length(g);
9 for i=1:a
10     g2(1,i)=g1(a-i+1);
11 end
12
13 gcs=(g+g2)/2 //Conjugate–Symmetric part
14 disp(gcs, 'The Conjugate symmetric part is =');
15 gcas=(g-g2)/2; //Conjugate–Antisymmetric part
16 disp(gcas, 'The Conjugate antisymmetric part is =');

```

---

## Scilab code Exa 2.6 Energy Signal

```
1 //EXAMPLE 2.6 , Energy Signal
2
3 clear ;
4 clc ;
5 n = -5:5;
6 for i =1:length ( n )
7     if(n(i)>=1)
8         h(i)=1/n(i);
9     else
10        h(i)=0;
11    end
12 end
13
14 Sum=0;
15 N=1:10000;
16     for i=1:length(N)
17         h(i)=(1/N(i))^2;
18     end
19
20 Energy = sum(h);
21
22 if (Energy<%inf ) then
23     disp ( 'Energy Signal' ) ;
24     disp(Energy, 'Energy of signal = ');
25 else
26     if (Energy/length(N)<%inf ) then
27         disp ( 'Power Signal' ) ;
28
29     else
30         disp ( 'Niether Energy nor Power Signal' ) ;
31     end
32 end
```

---

### Scilab code Exa 2.7 Power Signal

```
1 //EXAMPLE 2.7, Example of Power signal
2 clear ;
3 clc ;
4
5 Sum=0;
6 N=1:10000;
7     for i=1:length(N)
8         h1= 3*((-1)^i);
9         h=h1^2;
10        end
11
12 Energy = sum(h);
13     if (Energy/(2*(length(N))+1)<%inf ) then
14         disp ('Power Signal') ;
15         disp(Energy/2, 'Power Signal = ');
16     else
17         disp ('Not a Power Signal') ;
18     end
```

---

### Scilab code Exa 2.9 Square wave generation

```
1 //EXAMPLE 2.9, Generation of a Square wave sequence:
2 clc;
3 clear;
4 clf();
5 a=gca();
6 figure(0);
7 a.x_location="origin";
8 x=[0:1:80];
9 y1=sin(x*.05*%pi);
```

```

10 y2=sin(x*.15*%pi);
11 y3=sin(x*.25*%pi);
12 y4=y1+y2/3+y3/5;
13 plot2d3(x,y4,2)
14 plot(x,y4,'r. ')
15 xtitle('Approximate Square wave','x','y4');
16 a.children.children.thickness=3;

```

---

### Scilab code Exa 2.16 Linearity of accumulator

```

1 //EXAMPLE 2.16 ,
2 clear;
3 clc;
4 //Given input sequence = [3 4 5]
5 x=[0 3 4 5 0];
6 disp([3 4 5], 'Input sequence = ')
7 //determining median filter
8 //first sequence
9 for k=2:4
10     if x(k)>x(k-1) & x(k+1)>x(k-1) & x(k+1)>x(k)
11         y(k-1)=x(k);
12     else
13         x(k-1)>x(k+1) & x(k)>x(k+1) & x(k)>x(k-1)
14         y(k-1)=x(k-1);
15     end
16 end
17 disp(y', 'The Median Filter of the given input is =')
    ;

```

---

### Scilab code Exa 2.20 Passive system

```

1 //EXAMPLE 2.20 ,Passive or lossless system.
2 clear;

```

```

3  clc;
4  a=input("any value of a less than or equal to one")
5  n=-10:1:10;
6  x=n;
7  y=a*n;
8  S=0;
9  for i=1:length(n)
10     S=S+y^2;
11 end
12
13 if a<1 then
14     disp('the system is passive')
15 else
16
17     a==1
18     disp('the system is lossless')
19
20 end

```

---

### Scilab code Exa 2.22 Impulse response of Accumulator

```

1  //EXAMPLE 2.22,impulse response of accumulator
2
3  clear;
4  clc;
5  d=[1];
6  t=-1:.01:1;
7  h=0;
8  clf();
9  figure(0);
10 a=gca();
11 a.x_location="origin";
12
13 for i=1:length(t)
14     if t(i)<0

```

```

15     h=0;
16     else
17         h=d;
18         plot2d3(i-101,h)
19         plot(i-101,h,'r')
20         xtitle('Impulse Response of accumulator','t'
                , 'Y');
21         a.children.children.thickness=1;
22         a.children.children.foreground=2;
23     end
24 end
25 disp(h, 'The impulse response of Accumulator is =')

```

---

#### Scilab code Exa 2.26 Convolution

```

1 //EXAMPLE 2.26, convolution of x & h
2 x=[-2 0 1 -1 3];
3 disp(x, 'x = ');
4 h=[1 2 0 -1];
5 disp(h, 'h = ');
6 n=0:7;
7 y=convol(x,h);
8 disp(y, 'The convolution of the two inputs is :')

```

---

#### Scilab code Exa 2.27 Convolution

```

1 //EXAMPLE 2.27, convolution of an exponential
  sequence
2 clear;
3 clc;
4 n=0:.5:5
5 c=0.5;
6 b=0.4;

```



```

7 clf();
8 figure(0);
9 a=gca();
10 a.x_location="origin";
11 x = c^n;
12 subplot(2,2,1);
13 plot2d3(n,x,2);
14 plot(n,x, 'r');
15 xtitle('','n','x');
16 h = b^n ;
17 subplot(2,2,2);
18 plot2d3(n,h,2)
19 plot(n,h, 'r')
20 xtitle('','n','h');
21 N=0:.5:10;
22 y = convol (x , h );
23 subplot(2,2,3);
24 plot2d3(N,y,2)
25 plot(N,y, 'r')
26 xtitle('convol(x,h) ','n','y');
27 disp(y,'Convolution of the two exponential sequences
      is =')

```

---

### Scilab code Exa 2.28 Convolution

```

1 //EXAMPLE 2.28,graphical representation of
  convolution of x & h.
2 clear;
3 clc;
4 x=[-2 0 1 -1 3];
5 disp(x, 'x');
6 h=[1 2 0 -1];
7 disp(h, 'h');
8 n=0:7;
9 y=convol(x,h);

```

```

10 disp(y, 'convolution = ');
11 clf();
12 figure(0);
13 a=gca();
14 a.x_location="origin";
15 a.y_location="origin";
16 plot2d3(n,y)
17 plot(n,y, 'r. ')
18 xtitle('convolution', 'n', 'Y');
19 a.children.children.thickness=2;
20 a.children.children.foreground=2;

```

---

### Scilab code Exa 2.29 Convolution

```

1 //Example 2.29, Convolution using Tabular method.
2 clear;
3 clc;
4 x=[-2 0 1 -1 3];
5 h=[1 2 0 -1];
6 q=length(x);
7 w=length(h);
8 z=q+w-1;
9 y0=0;
10 for i=1:z;
11     y(i)=0;
12     for k=1:i;
13         if k>q
14             x(k)=0;
15         else
16             if (i-k+1)>w
17                 h(i-k+1)=0;
18             else
19                 y(i)= y(i) + x(k)*h(i-k+1);
20             end
21         end

```

```

22     end
23 end
24 disp(y', 'The Convolution of the two sequences is =')

```

---

### Scilab code Exa 2.30 Convolution

```

1 //EXAMPLE 2.30
2 //Convolution of two sided sequences
3 clear;
4 clc;
5 g=[3 -2 4]; //originating at n=-1
6 h=[4 2 -1]; //originating at n=0
7 q=length(g);
8 w=length(h);
9 z=q+w-1;
10 y0=0;
11 for i=1:z;
12     y(i)=0;
13     for k=1:i;
14         if k>q
15             g(k)=0;
16         else
17             if (i-k+1)>w
18                 h(i-k+1)=0;
19             else
20                 y(i)= y(i) + g(k)*h(i-k+1);
21             end
22         end
23     end
24 end
25 n=-1:z-2;
26 disp(y, 'The Convolution of the two sequences is =')
27 clf();
28 a=gca();
29 figure(0);

```

```
30 a.x_location="origin";
31 plot2d3(n,y,2);
32 plot(n,y,'r. ');
33 xtitle('convolution','n','y');
```

---

### Scilab code Exa 2.31 Stability of causal system

```
1 //EXAMPLE 2.31, Stability for causal system.
2 //h[i]=impulse response of LTI system.
3 clear;
4 clc;
5 n= -5:0.001:5;
6 a=0.6;
7
8 for i=1:length(n)
9     if (n(i)<0)
10         h(i)=0;
11     else
12         h(i)=abs(a^n(i));
13
14     end
15 end
16 S=sum(h);
17 if(S<%inf)
18     disp('BIBO stable system');
19 else
20     disp('BIBO unstable system');
21
22 end
```

---

### Scilab code Exa 2.32 Stability of Anti causal system

```
1 //EXAMPLE2.32 Stability for anti-Causal system.
```

```

2 //h[i]=impulse response of LTI system.
3 clear;
4 clc;
5 n= -5:1/1000:5;
6 a=5;
7 for i=1:length(n)
8     if (n(i)>-1)
9         h(i)=0;
10    else
11        h(i)=a^n(i);
12        S=sum(h);
13    end
14 end
15
16 if(S<%inf)
17     disp('BIBO stable system');
18 else
19     disp('BIBO unstable system');
20
21 end

```

---

### Scilab code Exa 2.33 Stability of a system

```

1 //EXAMPLE 2.33 ,stability of finite impulse response
.
2 //h[i]=impulse response of LTI system.
3 clear;
4 clc;
5 n= -5:1/100:5;
6 a= input('value of a');
7 N1=input('lower limit');
8 N2=input('upper limit');
9 for i=1:length(a)
10    if (n(i)<N1 & n(i)>N2)
11        h(i)=0;

```

```

12     else
13         h(i)=a^n(i);
14         S=sum(h);
15     end
16 end
17
18 if(S<%inf)
19     disp('BIBO stable system');           //as long as
20         N1,N2!=%inf
21 else
22     disp('BIBO unstable system');
23 end

```

---

#### Scilab code Exa 2.46 Cross coreation computation

```

1 //EXAMPLE 2.46, Cross corelation Computation.
2 // Given two finite length sequence.x[n],y[n]:
3 clear;
4 clc;
5 x=[1 3 -2 1 2 -1 4 4 2];
6 disp(x, 'x');
7 y=[2 -1 4 1 -2 3];
8 disp(y, 'y');
9 //Cross corelation rxy[n]:
10
11 rxy=convol(x,mtlbfliplr(y));
12 disp(rxy, 'The Cross-Corelation Operation of the
    Inputs is =')

```

---

# Chapter 3

## Discreet Time Fourier Transform

Scilab code Exa 3.5 DTFT computation

```
1 //EXAMPLE 3.5
2 //DTFT of unit sample sequence
3 clc;
4 clear;
5 //a=0.5;
6 n=0:9;
7 x = [1, zeros(1,9)];
8 disp(x, 'x[n] = ')
9
10 K = 4;
11 k = 0:4/1000:4;
12 W = k*2*%pi/K;
13 X = (x)*exp(%i*n'*W);
14 disp(X, 'DTFT, x[n] --> ')
15 X_mag = abs(X);
16 X_phase = phasemag(X); //no phase exists
17
18 figure(0);
19 plot2d3(mtlbfliplr(W), X_mag);
```

```

20 xtitle('Magnitude plot', 'W ---->', 'X_mag ---->');
21 figure(1);
22 plot2d3(mtlbfliplr(W), X_phase);
23 xtitle(' zero phase plot', 'W ---->', 'X_phase ---->');

```

---

### Scilab code Exa 3.6 DTFT computation

```

1 //EXAMPLE 3.6
2 //Determine DTFT of sequence
3 //PROGRAM REQUIRES MAXIMA SCILAB TOOLBOX
4
5 clc;
6 clear;
7 //Symbolic calculation
8 Syms n w a ;
9 x1=(a^n)*exp(-%i*n*w);
10 X1=nusum(x1,n,0,%inf);
11 disp(X1, 'DFT,X = ');
12
13 //Given:
14 a=0.5;
15 n=0:9;
16 //x[n]=a^n*u[n]
17 for i = 0:9
18     x(i+1) = a^i;
19 end
20 //The DTFT of the sequence
21 K = 4;
22 k = 0:4/1000:4;
23 W = k*6*%pi/K;
24 X = (x')*exp(%i*n'*W);
25 X_mag = abs(X);
26 [X_phase,db] = phasemag(X);
27
28 clf();

```



```

29 a=gca();
30 figure(0);
31 //Note %pi ~ 3.14
32 plot2d3(mtlbfliplr(W),X_mag);
33 xtitle('Magnitude response','W ---->','Amplitude ---->
      ');
34 figure(1);
35 plot2d3(mtlbfliplr(W),X_phase);
36 xtitle('Phase response','W ---->','Phase in degrees
      ---->');

```

---

### Scilab code Exa 3.7 Plotting real and imaginary part

```

1 //EXAMPLE 3.12
2 //x[n]=((-1)^n)*(a^n)*u[n]..... given a=0.5;
3
4 clc;
5 clear;
6
7 a=0.5;
8 n=0:9;
9 for i = 0:9
10     x(i+1) = (a*exp(-%i*%pi))^i;
11 end
12
13 //The DTFT of the sequence
14 K = 4;
15 k = 0:4/1000:4;
16 W = k*6*%pi/K;
17 X = (x')*exp(%i*n'*W);
18 X_mag = abs(X);
19 X_phase = phasemag(X);
20
21 //PLOTING GRAPHS FOR THE INTERVAL OF 0 TO 6*%pi
22 clf();

```

```

23 a=gca();
24 figure(0);
25 plot2d3(mtlbfliplr(W),X_mag);
26 xtitle('Magnitude response','W','Amplitude');
27 figure(1);
28 plot2d3(mtlbfliplr(W),X_phase);
29 xtitle('Phase response','W','X_phase,degrees');

```

---

### Scilab code Exa 3.10 DTFT of finite length exponential sequence

```

1 //EXAMPLE 3.10
2 // DTFT of a sequence
3 clc;
4 clear;
5 syms a n M w;
6 x=a^n;
7 X=nusum(x*(exp(-%i*w*n)),n,0,M-1)
8 disp(limit(X),'The DTFT of the given sequence, X = '
9 )

```

---

### Scilab code Exa 3.12 Plotting DTFT of exponential sequence

```

1 //EXAMPLE 3.12
2 //x[n]=((-1)^n)*(a^n)*u[n]..... given a=0.5;
3
4 clc;
5 clear;
6
7 a=0.5;
8 n=0:9;
9 for i = 0:9
10     x(i+1) = (a*exp(-%i*%pi))^i;
11 end

```

```

12
13 //The DTFT of the sequence
14 K = 4;
15 k = 0:4/1000:4;
16 W = k*6*%pi/K;
17 X = (x')*exp(%i*n'*W);
18 X_mag = abs(X);
19 X_phase = phasemag(X);
20
21 //PLOTING GRAPHS FOR THE INTERVAL OF 0 TO 6*%pi
22 clf();
23 a=gca();
24 figure(0);
25 plot2d3(mtlb_fliplr(W),X_mag);
26 xtitle('Magnitude response', 'W', 'Amplitude');
27 figure(1);
28 plot2d3(mtlb_fliplr(W),X_phase);
29 xtitle('Phase response', 'W', 'X_phase, degrees');

```

---

### Scilab code Exa 3.13 DTFT computation

```

1 //EXAMPLE 3.13
2
3 clc;
4 clear;
5 a=0.5;
6 n=0:9;
7 for i = 0:9
8     x(i+1) = a^i;
9 end
10 //The DTFT of the sequence
11 K = 4;
12 k = 0:4/1000:4;
13 W = k*6*%pi/K;
14 X1 = (x')*exp(%i*n'*W);

```

```

15 X = %i*diff(X1);
16 X = [X,0] + X1;
17
18 X_mag = abs(X);
19 [X_phase,db] = phasemag(X);
20
21 clf();
22 a=gca();
23 figure(0);
24 plot2d3(mtlbfliplr(W),X_mag);
25 xtitle('Magnitude response', 'W', 'X_mag');
26 figure(1);
27 plot2d3(mtlbfliplr(W),X_phase);
28 xtitle('Phase response', 'W', 'X_phase');

```

---

### Scilab code Exa 3.14 Energy of signal

```

1 //EXAMPLE 3.14
2 //ENERGY OF LP DISCREET TIME SIGNAL
3 //PROGRAM REQUIRES MAXIMA SCILAB TOOLBOX
4 clc;
5 clear;
6 syms n wc w;
7 wc = input("the value of wc ( less than %pi)= ");
8 n = -5:0.05:5;
9
10 for i =0:length (n)
11     hlp(i+1) = (wc/%pi)*sinc((wc*i)/%pi);
12     E(i+1)=(abs(hlp(i+1)))^2;
13 end
14
15 Energy = sum(E);
16 if (Energy<%inf ) then
17     disp ('The filter is Energy Signal') ;
18 else

```

```

19  if (Energy/length(N)<%inf ) then
20  disp ('Power Signal') ;
21  else
22  disp ('Niether Energy nor Power Signal') ;
23  end
24  end
25  disp(Energy, 'the energy is = ');

```

---

### Scilab code Exa 3.15 Energy of exponential sequence

```

1  //EXAMPLE 3.15
2  //ENERGY OF A SIGNAL  $x[n]=a^n*u[n]$ 
3  clc;
4  clear;
5  a=0.5;
6  n=0:0.1:9.9;
7  // $x[n]=a^n*u[n]$ 
8  for i = 0:length(n)
9      x(i+1) = a^i;
10     E=(abs(x))^2;
11 end
12 Energy=sum(E);
13 disp(Energy, 'Energy of the signal = ');

```

---

## Chapter 4

# Digital Processing of Continuous Time Systems

Scilab code Exa 4.5 Passband and Stopband ripple computation

```
1 //EXAMPLE 4.5
2 //determine ripple values in db;
3 clc
4 clear;
5 ap = 0.01//Peak passband ripple in dB
6 as = 70//min. stopband atteuation in dB
7 dp = 1-10^-(ap/20);
8 ds = 10^-(as/20);
9 disp( dp, 'dp = ');
10 disp( ds, 'ds = ');
```

---

Scilab code Exa 4.6 Order of Analog filter

```
1 //EXAMPLE 4.6
2 //Order of LP filter
3 clc;
```

```

4 clear;
5 ap = 1 //Peak passband ripple in dB
6 as = 40 //min. stopband attenuation in dB
7 wp = 1000 //Hz
8 ws = 5000 //Hz
9 k = wp/ws;
10 disp(1/k, '1/k = ');
11 k1 = 1/(sqrt(((10^(0.1* as))-1)/(10^(0.1*ap))-1)));
12 disp(1/k1, '1/k1 = ');
13 N=ceil(log10(sqrt(((10^(0.1* as))-1)/(10^(0.1*ap))-1)))
    /log10(1/k));
14 disp(N, 'order of the filter is :');

```

---

#### Scilab code Exa 4.7 Order of Analog Chebyshev Filter

```

1 //EXAMPLE 4.7
2 //Determine the order of Analog Chebyshev LP filter.
3 clc;
4 clear;
5 ap = 1 //dB
6 as = 40 //dB
7 wp = 1000 //Hz
8 ws = 5000 //Hz
9 k = wp/ws;
10 disp(1/k, '1/k = ');
11 k1 = 1/(sqrt(((10^(0.1* as))-1)/((10^(0.1*ap))-1)));
12 disp(1/k1, '1/k1 = ');
13 N = acosh(1/k1)/acosh(1/k);
14 disp(N, 'N = ');
15 disp('Since order of the filter is always an integer
    , ');
16 disp(ceil(N), 'Order of the filter is , N = ');

```

---

#### Scilab code Exa 4.8 Order of Analog Lowpass Elliptic Filter

```
1 //EXAMPLE 4.8
2 //Determine the order of Analog Elliptic LP filter.
3 clc;
4 clear;
5 ap = 1 //dB
6 as = 40 //dB
7 Fp = 1000 //Hz
8 Fs = 5000 //Hz
9 wp = Fp*2*%pi;
10 ws = Fs*2*%pi;
11
12
13 k1 = 1/(sqrt((10^(0.1* as)-1)/(10^(0.1*ap)-1)));
14 disp(1/k1, '1/k1 = ');
15 k = wp/ws;
16 k2 = sqrt(1 - (k*k));
17 disp(k2, "k2 = ");
18 po = (1 - sqrt(k2))/(2*(1 + sqrt(k2)));
19 disp(po, 'po = ');
20 p = po +2*po^5 + 15*po^9 + 150*po^13;
21 disp(p, 'p = ');
22 N = (2*log10(4/k1))/log10(1/p);
23 disp(N, 'N = ');
24 disp('Since order of the filter is always an integer
    , ');
25 disp(ceil(N), 'Order of the filter is , N = ');
```

---

#### Scilab code Exa 4.16 Design of Analog Butterworth HP Filter

```
1 //EXAMPLE 4.16
2 //Design analog butterworth High pass filter
3 clc;
4 clear;
```



```

5  wp=4000;
6  ws=1000;
7  ap=0.1;
8  as=40;
9
10 Ap=1; // assumption
11 As=(2*pi*wp)*Ap/(2*pi*ws);
12
13 N=ceil(log10(sqrt((10^(0.1* as)-1)/(10^(0.1*ap)-1)))
        /log10(As/Ap));
14 disp(N,'order of the filter is :');
15
16 Ac = As/((10^(0.1*as)-1)^(1/(N*2)));
17 disp(Ac,'cutoff frequency = ')
18
19 // [hs , pole , zero , gain]=analpf(N, 'butt ', Ac);
20
21 s=%s;
22 hs=1/((s + 1)*(s^2 + 0.61803*s + 1)*(s^2 + 1.61803*s
        + 1));
23 Hs=horner(hs , s/Ac);
24 H1 = numer(Hs)/0.0976514;
25 H2 = denom(Hs)/0.0976514;
26 disp(H1/H2,'the low pass transfer function is ,HLP(s)
        = ');
27 Hs=horner(hs , Ac/s);
28 H1 = numer(Hs);
29 H2 = denom(Hs);
30 disp(H1/H2,'the High pass transfer function is ,HHP(s)
        ) = ');

```

---

# Chapter 5

## Finite Length Discreet Transform

Scilab code Exa 5.1 DFT computation

```
1 //EXAMPLE 5.1
2 //DETERMINE DFT OF GIVEN SEQUENCE
3 clc;
4 clear;
5 N = input("length of sequence = ");
6 x = [1,zeros(1,N-1)];
7 disp(x,'The sequence is ,x = ');
8 X = dft(x,0);
9 disp(X,'DFT of the sequence is X = ');
10 m = input("value of some intemediate (mth) point =
           ");
11 y = [zeros(1,m-1),1,zeros(1,N-m)];
12 Y = dft(y,0);
13 disp(Y,'DFT of the sequence is Y = ');
```

---

Scilab code Exa 5.2 DFT of sinusoidal sequence

```

1 //EXAMPLE 5.2
2 //DFT of sinusoidal sequence
3 clc;
4 clear;
5
6 N = input(" input value of N ");
7 r = input(" input r value ");
8 n = 0:N-1;
9 x = cos(2*%pi*r*n/N)
10 X = dft(x,-1)
11 //X exists only at n={r,N-r} where X = N/2
12 clf();
13 a=gca();
14 a.x_location = "origin";
15 a.y_location = 'origin';
16 plot2d3(n,X,2);
17 a.thickness=1;
18 plot(n,X,'r. ');
19
20 xtitle('DFT', 'K -->', 'X[K] -->');
21 X = disp(X, 'DFT of x--> ');

```

---

### Scilab code Exa 5.3 DFT computation

```

1 //EXAMPLE 5.3
2 //DETERMINE DFT OF GIVEN SEQUENCE
3
4 clc;
5 clear;
6 N = input("length of sequence ,N = ");
7 M = input("M point DFT = ");
8 if M > N
9     x = [ones(1,N), zeros(1,M-N)];
10     disp(x, 'the sequence is :');
11     for n=0:M-1

```

```

12         for k=0:M-1
13             W(n+1,k+1) = exp(-(i*2*pi*k/M)*n);
14         end
15     end
16     X = W*x';
17     disp(X, 'DFT is , X = ');
18 else
19     disp('invalid computation');
20 end
21
22
23 n=0:M-1;
24 clf();
25 figure(0)
26 a = gca();
27 plot2d3(n,x,2) // plotting the sequence
28 plot(n,x,'r. ');
29 a.x_location = 'origin';
30 a.y_location = 'origin';
31 poly1 = a . children (1) . children (1) ;
32 poly1.thickness = 2.5;
33 xtitle('original sequence', 'n', 'x[n]');
34
35 figure(1)
36 a = gca();
37 plot2d3(n,abs(X),2) // plotting absolute value of
    DFT of sequence
38 plot(n,abs(X), 'r. ');
39 a.x_location = 'origin';
40 a.y_location = 'origin';
41 poly1 = a . children (1) . children (1) ;
42 poly1.thickness = 2.5;
43 xtitle('magnitude plot', 'M', 'Absolute value');

```

---

Scilab code Exa 5.4 IDFT Computation

```

1 //EXAMPLE 5.4
2 //DETERMINE IDFT OF GIVEN SEQUENCE
3 clc;
4 clear;
5 K = input(" value of K ");
6 disp('input M > K');
7 M = input(" value of M ");
8 k1 = 0:K-1;
9 V1 = k1./K; //DFT
10 k=0:M-1;
11
12 N = length(V1);
13 V = [V1,zeros(1,M-N)];
14 v = dft(V,1); //IDFT
15
16 clf();
17 subplot(1,2,1)
18
19 a = gca();
20 plot2d3(k,real(v),2);
21 plot(k,real(v),'r. ');
22 a.x_location = 'origin';
23 a.y_location = 'origin';
24 poly1 = a . children (1) . children (1) ;
25 poly1.thickness = 2;
26 xtitle('real part','N','v');
27
28 subplot(1,2,2)
29 a = gca();
30 plot2d3(k,imag(v),2)
31 plot(k,imag(v),'r. ');
32 a.x_location = 'origin';
33 a.y_location = 'origin';
34 poly1 = a . children (1) . children (1) ;
35 poly1.thickness = 2;
36 xtitle('imaginary part','N','v');
37 v = disp(v);

```

---

### Scilab code Exa 5.5 DFT computation

```
1 //EXAMPLE 5.5
2 //DFT computation
3 clc;
4 clear;
5
6 N = 16 ;
7 r = 3 ;
8 n = 0:N-1;
9 x = cos(2*%pi*r*n/N)
10 X = fft(x,-1)//DFT of the sequence
11 clf();
12 a = gca();
13 plot2d3(n,X,2);
14 plot(n,X,'r. ');
15 a.x_location = 'origin';
16 a.y_location = 'origin';
17 poly1 = a . children (1) . children (1) ;
18 poly1.thickness = 3;
19 xtitle('DFT', 'k', 'X');
20 X = disp(real(X), ' X = ');
```

---

### Scilab code Exa 5.7 Circular convolution computation

```
1 //EXAMPLE 5.7
2 //Circular convolution
3 clear;
4 clc;
5 g = [1 2 0 1];
6 disp(g, 'g[n] = ');
7 h = [2 2 1 1];
```

```

8 disp(h, 'h[n] = ');
9 G = fft(g,-1);
10 H = fft(h,-1);
11 Y = G.*H;
12 yc = fft(Y,1);
13 n1 = 0:length(yc)-1;
14 y1 = convol(g,h);
15 n2 = 0:length(y1)-1;
16
17 clf();
18 subplot(2,1,1)
19 a = gca();
20 plot2d3(n1,yc,2);
21 plot(n1,yc,'r. ');
22 a.x_location = 'origin';
23 a.y_location = 'origin';
24 poly1 = a . children (1) . children (1) ;
25 poly1.thickness = 3;
26 xtitle('circular convolution','n','yc');
27
28 subplot(2,1,2)
29 a = gca();
30 plot2d3(n2,y1,2);
31 plot(n2,y1,'r. ');
32 a.x_location = 'origin';
33 a.y_location = 'origin';
34 poly1 = a . children (1) . children (1) ;
35 poly1.thickness = 3;
36 xtitle('linear convolution','n','y1');
37
38 disp(real(yc)," circular convolution ,yc = ");
39 disp(y1," linear convolution ,y1 = ");

```

---

**Scilab code Exa 5.8** Circular convolution computation

```

1 //EXAMPLE 5.8
2 //Circular convolution
3 clc;
4 clear;
5 g = [1 2 0 1];
6 disp(g, 'g[n] = ');
7 h = [2 2 1 1];
8 disp(h, 'h[n] = ');
9 G = fft(g, -1);
10 H = fft(h, -1);
11 Y = G.*H;
12 yc = fft(Y, 1); //IDFT of Y
13 disp(yc, "circular convolution, yc = ")
14 n=0:3;
15 clf();
16 figure(0);
17 a = gca();
18 plot2d3(n, yc, 2);
19 plot(n, yc, 'r. ');
20 a.x_location = 'origin';
21 a.y_location = 'origin';
22 poly1 = a . children (1) . children (1) ;
23 poly1.thickness = 3;
24 xtitle('Circular convolution', 'n', 'yc');

```

---

#### Scilab code Exa 5.10 Generating symmetric parts

```

1 //EXAMPLE 5_10
2 //conjugate symmetric & anti-symmetric parts of
   complex sequence
3 clear;
4 clc;
5 un=[1+%i*4, -2+%i*3, 4-%i*2, -5-%i*6];
6 disp(un, 'u[n] = ');
7 u1=conj(un);

```



```

8 disp(u1, 'u*[n] = ');
9 //modulo-4 circularly time reversed version:
10 disp(pmodulo(0,4), 'u[<-0>4] = ');
11 disp(pmodulo(-1,4), 'u[<-1>4] = ');
12 disp(pmodulo(-2,4), 'u[<-2>4] = ');
13 disp(pmodulo(-3,4), 'u[<-3>4] = ');
14 un1=[u1(pmodulo(-0,4)+1), u1(pmodulo(-1,4)+1), u1(
    pmmodulo(-2,4)+1), u1(pmodulo(-3,4)+1)];
15 disp(un1, 'u*[<-n>4] = ');
16 disp(0.5*(un+un1), 'ucs[n] = ');
17 disp(0.5*(un-un1), 'uca[n] = ');

```

---

#### Scilab code Exa 5.11 Circular convolution computation

```

1 //EXAMPLE 5.11
2 //Circular convolution using DFT
3 clc;
4 clear;
5 g = [1 2 0 1];
6 disp(g, 'g[n] = ');
7 h = [2 2 1 1];
8 disp(h, 'h[n] = ');
9 M=4;
10 for n=0:M-1
11     for k=0:M-1
12         W(n+1,k+1) = exp(-(%i*2*%pi*k/M)*n);
13     end
14 end
15 G = W*g';
16 H = W*h';
17 disp(G, 'DFT is , G = ');
18 disp(H, 'DFT is , H = ');
19
20 Y=G.*H;
21 y=(1/4)*conj(W)*(Y);

```

```
22 disp(real(y), 'Circular convolution = ');
```

---

### Scilab code Exa 5.12 Linear Convolution using DFT

```
1 //EXAMPLE 5.12
2 //Linear convolution using Circular convolution
3 clc;
4 clear;
5 g = [1 2 0 1];
6 disp(g, 'g[n] = ');
7 h = [2 2 1 1];
8 disp(h, 'h[n] = ');
9
10 //linea convolution length = 4+4-1 = 7
11 //appending the two signals with zeros
12 g = [g, zeros(1,3)]
13 h = [h, zeros(1,3)]
14 G = fft(g,-1);
15 H = fft(h,-1);
16 Y = G.*H; //element wise multiplication
17 y = fft(Y,1); //IDFT
18
19 //Plotting linear convolution
20 n=0:6;
21 figure(0);
22 clf();
23 a = gca();
24 a.x_location = 'origin';
25 a.y_location = 'origin';
26 plot2d3(n,y,2);
27 plot(n,y,'r. ');
28 poly1 = a . children (1) . children (1) ;
29 poly1.thickness = 2;
30 xtitle('Linear convolution ', 'n', 'y');
31 disp(y, " linear convolution ,y = ");
```

---

**Scilab code Exa 5.14** DFT computation using single DFT

```
1 //EXAMPLE 5.14
2 //DFT of two real sequences using one DFT
3 clear;
4 clc;
5 g = [1 2 0 1];
6 disp(g, 'g[n] = ');
7 h = [2 2 1 1];
8 disp(h, 'h[n] = ');
9 x = g + %i.*h;
10 disp(x, 'x[n] = ');
11 X = fft(x, -1);
12 disp(X, 'The DFT, X[k] = ');
13 X1 = conj(X);
14 disp(X1, 'X*[k] = ');
15
16 for i=0:3;
17     a(i+1)=pmodulo(-i,4);
18     X2(i+1)=X1(a(i+1)+1);
19 end
20
21 X3 = conj(X2');
22 disp(X3, 'X*[\<4-k>4] = ');
23 disp(0.5*(X + X3), 'G[k] = ');
24 disp((X - X3)/(2*%i), 'H[k] = ');
```

---

**Scilab code Exa 5.15** DFT computation using single DFT of shorter length

```
1 //EXAMPLE 5.15
2 //DFT computation using DFT of shorter length
   sequences
```

```

3
4 clc;
5 clear;
6 v = [1 2 2 2 0 1 1 1];
7 disp(v, 'Length-8 real sequence v[n] = ')
8 for i=1:4
9     g(i)=v(2*i-1);
10    h(i)=v(2*i);
11 end
12 G = fft(g, -1);
13 H = fft(h, -1);
14 M=length(v);
15 //for n=0:M-1
16     for k=0:M-1
17         W(1,k+1) = exp(-(%i*2*%pi*k/M)*1);
18     end
19 //end
20 G=[G(1) G(2) G(3) G(4) G(1) G(2) G(3) G(4)] ;
21 H=[H(1) H(2) H(3) H(4) H(1) H(2) H(3) H(4)] ;
22 V=G + W.*H;
23 disp(V, 'DFt, V[k] = ');
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40 //for k = 0:3

```

```
41 //      V1(k+1) = G(k+1) + (exp(-2*%pi*%i*k/8))*H(k+1)
      ;
42 //end
43
44 //for k = 4:7
45 //      V2(k) = G(k-3) + (exp(-2*%pi*%i*k/8))*H(k-3);
46 //end
47 //disp ([V1, zeros(1,3)]+V2)
```

---

# Chapter 6

## z Transform

Scilab code Exa 6.1 z Transform of causal exponential sequence

```
1 //EXAMPLE 6.1
2 //Z-Transform of causal sequence
3 clc;
4 clear;
5 syms n a z;
6 x = a^n;
7 X = nusum(x*(1/z)^n,n,0,%inf);
8 limit(X);
9 disp(' X = ',X);
10 disp(' ROC = |z|>|a|')
```

---

Scilab code Exa 6.2 z transform of anticausal sequence

```
1
2 clc;
3 clear;
4 syms n a z;
5 x = a^n;
```

```

6 X = nusum(-x*(1/z)^n,n,-%inf,-1);
7 limit(X);
8 disp(' X = ',X);
9 disp(' ROC = |z|<|a| ')

```

---

### Scilab code Exa 6.3 z Transform

```

1 //EXAMPLE 6.3
2 //Z-Transform
3 clc;
4 clear;
5 syms n a z M N;
6 x = a^n;
7 X = nusum(x*(1/z)^n,n,M,N-1);
8 limit(X);
9 disp(X,' X = ');

```

---

### Scilab code Exa 6.4 z Transform

```

1 //EXAMPLE 6.4
2 //Z-Transform
3 clc;
4 clear;
5 syms n z;
6 x = (-0.6)^n;
7 X = nusum(x*((1/z)^n),n,0,%inf);
8 limit(X);
9 disp(X,' X = ');

```

---

### Scilab code Exa 6.5 Z transform of causal sequence

```

1 //EXAMPLE 6.9
2 //Z-Transform of causal sequence
3 clc;
4 clear;
5 //z=%z;
6 syms n a z M N;
7 x = a^n;
8 X = nusum(x*(1/z)^n,n,-M,N);
9 limit(X);
10 disp(X, ' X = ');

```

---

**Scilab code Exa 6.9** z Transform

```

1 //EXAMPLE 6.9
2 //Determination of ROC
3 clc;
4 clear;
5 z=%z;
6 a=2*z^4+16*z^3+44*z^2+56*z+32;
7 b=3*z^4+3*z^3-15*z^2+18*z-12;
8 [h1,g1]=factors(a);
9 [h2,g2]=factors(b);
10 disp(h1, 'h1 = ');
11 disp(h2, 'h2 = ');
12 c=a/b;
13 disp(c, 'function is = ');
14 plzr(c);

```

---

**Scilab code Exa 6.10** Rational form of z Transform from its zero and pole locations

```

1 //EXAMPLE 6.10
2 //Z-transform from pole-zero locations

```



```

3  clc;
4  clear;
5  z=%z;
6  //using the pole & zero locations provided
7  num=(z-0.21)*(z-3.14)*(z-(-0.3+%i*0.5))*(z-(-0.3-%i
      *0.5));
8  den=(z+0.45)*(z-0.67)*(z-(0.81+%i*0.72))*(z-(0.81-%i
      *0.72));
9  k=2.2;
10 Gz=(num/den);
11 disp(k*Gz, 'Gz = ');

```

---

#### Scilab code Exa 6.11 Inverse z Transform

```

1  //EXAMPLE 6.11
2  //Inverse Z-transform
3  clc;
4  clear;
5  syms n z1;
6  z = %z;
7  num = z;          //given |z|>1;
8  den = (z-1)^2;
9
10
11 //Power series expansion
12 x=ldiv(num,den,20);
13 disp(x, 'x = ');
14 disp('x = n*u[n] ');

```

---

#### Scilab code Exa 6.12 Inverse z Transform

```

1  //EXAMPLE 6.12
2  //Inverse Z-transform

```

```

3  clear;
4  clc;
5  z=%z;
6  num = 0.5*z;
7  den = z^2 -z + 0.25;
8  func = num/den;
9  v = factors(den);
10 disp(v, 'factors are = ');
11 h1=ldiv(num,den,10);
12 disp(h1, 'h = ');
13 //using the property of z-trasnform
14 disp('observing v(1) & v(2) we conclude          h=n
      *(0.5)^n ')

```

---

### Scilab code Exa 6.13 Proper fraction of Rational z Transform

```

1  //EXAMPLE 6.13
2  //Determining proper fraction
3  clc;
4  clear;
5  z=%z;
6  num = 2*z^3 + 0.8*z^2 + 0.5*z +0.3;
7  den = (z^3 + 0.8*z^2 + 0.2*z);
8  func = num/den;
9  disp(func, 'the polynomial function is H = ')
10
11  if degree(num)>=degree(den)
12      disp('An improper fraction');
13  else disp('A proper fraction');
14  end
15
16 disp('decomposing the fraction we get .....');
17
18 H1=func-(-3.5*z + 1.5)/z;
19 disp(H1, 'H1 = ');

```

20 `disp('H1 is a Proper fraction')`

---

**Scilab code Exa 6.14** Inverse z Transform by partial fraction expansion

```
1 //EXAMPLE 6.14
2 //Inverse Z-transform
3 clear;
4 clc;
5 z = %z;
6 num=z*(z+2);
7 den=(z-0.2)*(z+0.6);
8 H=num/den;
9 elts=factors(den);
10 disp(elts);
11 //solving Partial Fractions, we get:
12 Hz = 2.75/(1-(0.2)/z) - 1.75/(1+(0.6)/z);
13 disp(Hz);
14 //disp(h = 2.75*(0.2^n) - 1.75*(0.6^n)*u(n));
15 h1= ldiv(2.75*z,(z-(0.2)),10)
16 disp(h1/2.75,'h1 = ');
17 h1= ldiv(1.75*z,(z+(0.6)),10)
18 disp(h1/1.75,'h2 = ');
19 disp('the inverse z-transform is :')
20 disp('h = 2.75*(0.2^n)*u(n) - 1.75*(-0.6^n)*u(n)')
```

---

**Scilab code Exa 6.15** residue computation using coefficient matching approach

```
1 //EXAMPLE 6.15
2 //solving for coefficients;
3 clear;
4 clc;
5 z = %z;
```

```

6 num=z*(z+2);
7 den=(z-0.2)*(z+0.6);
8 H=num/den;
9 disp('the factors are :');
10 elts=factors(den);
11 disp(elts);
12 //coeff are:
13 disp('The coefficients are p1,p2:');
14 p1 = horner((z+2)/(z+0.6),0.2);
15 disp(p1,'p1 = ');
16 p2 = horner((z+2)/(z-0.2),-0.6);
17 disp(p2,'p2 = ');

```

---

**Scilab code Exa 6.16** Inverse z Transform by power series expansion

```

1 //EXAMPLE 6.16
2 //Partial fraction expansion
3
4 clc;
5 clear;
6 z=%z;
7 num = z^3;
8 den = 18*z^3 + 3*z^2 - 4*z - 1;
9 elts=factors(den);
10 disp(elts,'the factors are :') ;
11 func = num/den;
12 //the partial fraction gives:
13 p1 = horner((1/(1+0.3333333/z)^2),0.5);
14 disp(p1,'p1 = ');
15 p2 = horner(1/((1-0.5/z)), -0.3333333);
16 disp(p2,'p2 = ');
17 p3 = horner(0.6/((1-0.5/z)), -0.3333333);
18 disp(p3,'p3 = ');
19 disp('partial fraction gives : ');
20 disp(p1*z/elts(1),'h1 = ');

```

```
21 disp(p3*z/elts(3), 'h2 = ');
22 disp(p2*z^2/(elts(2)*elts(2)), 'h3 = ');
```

---

**Scilab code Exa 6.17** Coefficients of rational form

```
1 //EXAMPLE 6.16
2 //Coefficients of Rational form
3
4 clc;
5 clear;
6 z=%z;
7 num = 18*z^3;
8 den = 18*z^3 + 3*z^2 - 4*z - 1;
9 disp(coeff(num)/18, 'the Numerator polynomial
   coefficients are:');
10 disp(coeff(den)/18, 'the denominator polynomial
   coefficients are:');
```

---

**Scilab code Exa 6.18** Inverse z Transform using long division

```
1 //EXAMPLE 6.18
2 //Inverse Z-transform using power series expansion
3 clc;
4 clear;
5 z=%z;
6 Xnum=z;
7 Xden=(z-1)^2;
8 xn=ldiv(Xnum, Xden, 15);
9 disp(xn, 'The function is = ');
10 disp(' Thus, xn = n*u(n)');
```

---

**Scilab code Exa 6.19** Inverse z Transform using long division

```
1 //EXAMPLE 6.19
2 //Inverse Z-transform using Long division method
3 clc;
4 clear;
5 z=%z;
6 Hnum=z^2 + 2*z;
7 Hden=z^2 + 0.4*z -0.12;
8 hn=ldiv(Hnum,Hden,20);
9 disp(hn,'The function is , hn = ');
```

---

**Scilab code Exa 6.20** Inverse z Transform

```
1 //EXAMPLE 6.20
2 //Inverse Z-transform using power series expansion
3 clc;
4 clear;
5 z=%z;
6 Hnum=z^2 + 2*z;
7 Hden=z^2 + 0.4*z -0.12;
8 hn=ldiv(Hnum,Hden,20);
9 disp(hn,'The impulse response is , hn = ');
```

---

**Scilab code Exa 6.22** z Transform

```
1 //Example 6.22
2 //MAXIMA SCILAB TOOLBOX REQUIRED FOR THIS PROGRAM
3 //Z transform of  $r^n \cdot \cos(\omega_0 n)$ 
4 clc ;
5 clear;
6 syms r wo n z ;
7 x1 =(r^n)*exp(%i*wo*n) ;
```

```

8 X1 = nusum(x1*(z^-n),n,0,%inf);
9 x2 =(r^n)*exp(-%i*wo*n) ;
10 X2 = nusum(x2*(z^-n),n,0,%inf) ;
11 X =(X1+X2)/2 ;
12 disp(X, 'X(z)=') ;
13 disp('ROC : |z|>r ');

```

---

### Scilab code Exa 6.23 z Transform

```

1 //Example 6.23
2 //MAXIMA SCILAB TOOLBOX REQUIRED FOR THIS PROGRAM
3 //Z transform of w(n) =((-0.5)^(n-2) +(0.2)^(n-1))*
   u(n)
4
5 syms n z ;
6 w1 = 4*(-0.5)^n
7 W1 = nusum(w1,n,0,%inf);
8 disp(W1, 'ROC = |z|> 0.5 ,W1 = ');
9 w2 = 5*(0.2)^n
10 W2 = nusum(w2,n,0,%inf);
11 disp(W2, ' ROC = |z|> 0.2 , W2 = ');
12 disp(W1+W2, 'The Z-Transform is = ');
13 disp('ROC = |z|>0.5 ');

```

---

### Scilab code Exa 6.24 sum of sequences of non overlapping ROC

```

1 //Example 6.24
2 //MAXIMA SCILAB TOOLBOX REQUIRED FOR THIS PROGRAM
3 //Z transform of v(n) =(a)^(n)*u(n) - (b)^(n)*u(-n
   -1)
4 clc;
5 clear;
6 disp('assuming |b| > a ');

```

```

7 syms a n;
8 x1 = a^n;
9 X1 = nusum(x1,n,0,%inf);
10 x1 = b^n;
11 X1 = nusum(x2,n,-%inf,-1);
12 Vz = X1 + X2;
13 disp(Vz,'The Z-transform is = ');
14 disp('ROC = |a|<|z|<|b|');

```

---

#### Scilab code Exa 6.25 z Transform

```

1 //Example 6.25
2 //Z transform of Vz , d0*v[n] + d1*v[n-1] = p0*d[n]
   + p1*d[n-1];
3 //MAXIMA SCILAB TOOLBOX REQUIRED FOR THIS EXAMPLE
4 clc;
5 clear;
6 syms p0 p1 d0 d1;
7 z= %z;
8 disp('given that v[n] --> V(z).Using Time shifting
   property ,we get : ');
9 disp(' d0*Vz + d1*Vz*(1/z) = p0 + p1*(1/z) ');
10 disp('Rearranging the terms ... ');
11 Vz = (p0 + p1/z)/(d0 + d1/z);
12 disp(Vz,'Z-transform is Vz =');

```

---

#### Scilab code Exa 6.26 z Transform

```

1 // Example 6.26
2 //MAXIMA SCILAB TOOLBOX
3 //Z transform of (n+1)*a^n*u(n)
4
5 clear ;

```



```

6  clc ;
7  syms a n z ;
8  x1 =(a)^n ;
9  X1 = symsum(x1*(z^(-n)),n,0,%inf);
10 X2 = -z*(diff (X,z,1)) ;
11 X = X1 + X2;
12 disp (X , 'Z transform of (n+1)*a^n*u(n) is X = ');

```

---

### Scilab code Exa 6.27 Inverse z Transform

```

1  // Example 6.27
2  //inverse Z-transform of z^3/(z-0.5)*(z+1/3)^2;
3
4  clear ;
5  clc ;
6  z=%z;
7  Gnum = z^3;
8  Gden = (z-0.5)*(z+1/3)^2;
9  G = Gnum/Gden;
10 g1=ldiv(Gnum,Gden,10);
11 elts=factors(Gden);
12
13 //the partial fraction gives:
14 p1 = horner((1/(1+0.3333333/z)^2),0.5);
15 disp(p1,'p1 = ');
16 p2 = horner(1/((1-0.5/z)), -0.3333333);
17 disp(p2,'p2 = ');
18 p3 = horner(0.6/((1-0.5/z)), -0.3333333);
19 disp(p3,'p3 = ');
20 disp('partial fraction gives : ');
21 disp(p1*z/elts(1),'h1 = ');
22 disp(p3*z/elts(3),'h2 = ');
23 disp(p2*z^2/(elts(2)*elts(2)),'h3 = ');
24 disp('gn = 0.36*(0.5)^n + 0.24*(-1/3)^n + 0.4*(n+1)
      *(-1/3)^n');

```

```
25 disp(g1, 'the first 10 samples of g[n] = ');
```

---

**Scilab code Exa 6.28** Enlargement of ROC by pole zero cancellation

```
1 // Example 6.28
2 //Enlargement of ROC by pole-zero cancellation
3 clc;
4 clear;
5 z=%z;
6 Gz = (2 + 1.2*(1/z))/(1 - 0.2*(1/z))
7 disp(Gz, 'Gz = ');
8 disp('ROC = |z|>0.2');
9 Hz = 3/(1 + 0.6*(1/z));
10 disp(Hz, 'Hz = ');
11 disp('ROC = |z|>0.6');
12 Xz = Gz*Hz;
13 disp(Xz, 'Xz = ');
14 disp('ROC = |z|>0.2');
```

---

**Scilab code Exa 6.30** Convolution

```
1 //EXAMPLE 6.30
2 //PROGRAM REQUIRES MAXIMA SCILAB TOOLBOX
3 //USE Z-TRANSFORM TO EVALUATE CONVOLUTION OF TWO
  SEQUENCES:
4 clc;
5 clear;
6 syms n z;
7 x = [-2 0 1 -1 3];
8 h = [1 2 0 -1 0];
9
10 for n=0:4
11     X(n+1) = x(n+1)*z^(-n);
```

```

12     H(n+1) = h(n+1)*z^(-n);
13 end
14 disp(X', 'X = ');
15 disp(H', 'H = ');
16
17 for i=1:5
18     U(i)=0;
19     for j=1:5
20         U(i)=U(i)+X(i)*H(j);
21     end
22 end
23 Y=0;
24 for i=1:5;
25     Y = Y + U(i);
26 end
27 disp(Y, 'Y = ');
28
29 disp('y = [-2 -4 1 3 1 5 1 -3]')

```

---

### Scilab code Exa 6.31 Convolution

```

1 //EXAMPLE 6.31
2 //PROGRAM REQUIRES MAXIMA SCILAB TOOLBOX
3 //USE Z-TRANSFORM TO EVALUATE CONVOLUTION OF TWO
  SEQUENCES:
4 clc;
5 clear;
6 syms n z;
7 x = [3 -2 4];
8 h = [4 2 -1];
9
10 for n=-1:1
11     X(n+2) = x(n+2)*(z^-n);
12 end
13 disp(X', 'X = ');

```

```

14
15 for n=0:2
16     H(n+1) = h(n+1)*(z^-n);
17 end
18 disp(H', 'H = ');
19
20
21 for i=1:3
22     U(i)=0;
23     for j=1:3
24         U(i)=U(i)+X(i)*H(j);
25     end
26 end
27 Y=0;
28 for i=1:3;
29     Y = Y + U(i);
30 end
31 disp(Y, 'Y = ');
32
33 disp('y = [12 -2 9 10 -4]');

```

---

### Scilab code Exa 6.33 Transfer Function of Moving Average Filter

```

1 //EXAMPLE 6.33
2 //Transfer function of moving average filter
3 clear;
4 clc;
5 syms n z M;
6 x=z^(-n);
7 H1=nusum(x,n,0,M-1);
8 H=H1/M;
9 disp(H, 'Transfer function , Hz = ');

```

---

**Scilab code Exa 6.34** Transfer function determination

```
1 //EXAMPLE 6.34
2 //y[n]=x[n-1] - 1.2*x[n-2] + x[n-3] + 1.3*y[n-1]
   -1.04*y[n-2] + 0.222*y[n-3]
3 //Transfer function determination
4
5 clc;
6 clear;
7 z=%z;
8 disp('Given the difference equation taking
   ztransform on both sides :')
9 Yz = z^2 -1.2*z +1;
10 Xz = z^3 -1.3*z^2 + 1.04*z -0.222;
11 Hz = Yz/Xz;
12 disp(Hz, 'The transfer function is = ')
13 elts = factors(Xz);
14 disp(elts, 'factors of Xz are = ')
15 plzr(Hz);
```

---

# Chapter 7

## LTI Discreet Time systems in the Transform Domain

Scilab code Exa 7.1 Bounded real function

```
1 //EXAMPLE 7.1
2 //PROGRAM REQUIRES MAXIMA SCILAB TOOLBOX
3
4 clc;
5 clear;
6 syms K a z w;
7
8 hzden = (1-a*(z^-1)); //0<|a|<1;
9 Hz = K/hzden;
10 disp(' |H(e^(jw))|^2 = K^2/((1+a)^2 - 2*cos(w)) ');
11 // considering a>0
12 disp('(at w = %pi),K^2/(1+a)^2 < |H|^2 < K^2/(1-a)
    ^2,(at w = 0)');
13 //considering a<0
14 disp('(at w = 0),K^2/(1+a)^2 < |H|^2 < K^2/(1-a)
    ^2,(at w = %pi)');
15 disp('if K = +/- (1-a), observe ... ');
16 disp(' |H(e^(jw))| <= 1 Hence a Bounded real
    function. ');
```

```

17 //w=0:%pi;
18 // [a, b]=freq (hznum , hzden , w) ;
19 disp (abs (Hz))

```

---

### Scilab code Exa 7.2 Transfer function determination

```

1 //EXAMPLE 7.2
2 // |H(e^(jw))|^2 = 4*((1.09 + 0.6*cosw)*(1.16 - 0.8*
   cosw))/((1.04 - 0.2*cosw)*(1.25 + cosw))
3 //REPLACING cosw = (z + z^(-1))/2
4 clc;
5 clear;
6 z=%z;
7 H1=4*((1.09 + (0.3)*(z+1/z))*(1.16 - (0.4)*(z+1/z)))
   ;
8 H2=((1.04 - (0.2)*(z+1/z))*(1.25 + (0.5)*(z+1/z)));
9 H=H1/H2;
10 disp(H, 'The transfer function is , H = ');
11 elts1=factors( numer(H));
12 disp(elts1, 'The factors of numerator are :');
13 elts2=factors( denom(H));
14 disp(elts2, 'The factors of denominator are :');
15 disp('The Four possible stable transfer function with
   same square magnitude function are :');
16 h1=2*((1+(0.3)/z)*(1- (0.4)/z))/((1-(0.2)/z)
   *(1+(0.5)/z));
17 disp(h1, 'stable transfer function ,h1 = ');
18 h2=2*((1+(0.3)/z)*((0.4)- (1)/z))/((1-(0.2)/z)
   *(1+(0.5)/z));
19 disp(h2, 'stable transfer function ,h2s = ');
20 h3=2*((((0.3)+1/z)*((1)- (0.4)/z))/((1-(0.2)/z)
   *(1+(0.5)/z));
21 disp(h3, 'stable transfer function ,h3 = ');
22 h4=2*((((0.3)+1/z)*((0.4)- (1)/z))/((1-(0.2)/z)
   *(1+(0.5)/z));

```

```
23 disp(h4, 'stable transfer function ,h4 = ');
```

---

### Scilab code Exa 7.6 FIR Transfer function

```
1 //EXAMPLE 7.6
2 //FIR Trasnfer functions with different Phase.
3 clc;
4 clear;
5 z = %z;
6 W = 0:(1/400):1;
7 z = exp(%i*2*%pi*W);
8 for i=1:401
9     H1z(i)= -1+ 2/z(i) - 3/(z(i)^2) + 6/(z(i)^3)
10         -3/(z(i)^4) +2/(z(i)^5) -1/z(i)^6);
11 end
12 H1z_phase = phasemag(H1z);
13 clf();
14 figure(0);
15 plot2d(W/(2*%pi),H1z_phase ,1);
16 xtitle('phase response ', 'W/(2*%pi)', 'H2z_phase in
17     degrees ');
18 for i=1:401
19     H2z(i)= +1 - 2/z(i) + 3/(z(i)^2) - 6/(z(i)^3) +
20         3/(z(i)^4) - 2/(z(i)^5) + 1/z(i)^6);
21 end
22 H2z_phase = phasemag(H2z);
23 plot2d(W/(2*%pi),H2z_phase ,2);
24 xtitle('phase response ', 'W/(2*%pi)', 'H2z_phase in
25     degrees ');
```

---



# Chapter 8

## Digital Filter Structures

Scilab code Exa 8.1 Analysis of Cascaded lattice digital filter structure

```
1 //EXAMPLE 8.1
2 //MAXIMA SCILAB TOOLBOX REQUIRED FOR THIS EXAMPLE
3 //Digital filter structure
4 clear;
5 clc;
6 syms W1 W2 W3 X Y a d B y E z;
7 //Equations obtained are as follows:
8 W1 = X - a*W3/z;
9 W2 = W1 - d*W2/z;
10 W3 = W2/z + E*W2;
11 Y = B*W1 +y*W3/z;
12 //Solving the above equations:
13 Hz=(B + (B*d+y*E)/z + y/(z^2))/(1 + (d+a*E)/z + a/(z
    ^2))
14 disp(Hz, 'Hz = ');
```

---

Scilab code Exa 8.6 Factorization of FIR Transfer Function

```

1 //Example 8.6
2 //Factorization of FIR Transfer Function
3 clear;
4 clc;
5 z=%z;
6 Hz=50.4+28.02/z+13.89/z^2+7.42/z^3+6.09/z^4+3/z^5+1/
   z^6;
7 disp(factors( numer(Hz)), 'The Factors of the FIR
   Transfer Function are = ');

```

---

**Scilab code Exa 8.7** Factorization of IIR Transfer Function

```

1 //Example 8.7
2 //Factorization of IIR Transfer Function
3 clear;
4 clc;
5 z=%z;
6 //Numerator of the transfer function
7 Numz=6+17.1/z+33.05/z^2+24.72/z^3+19.908/z^4-5.292/z
   ^5+18.144/z^6;
8 //Denominator of the transfer function
9 Denz=1+2.2/z+2.56/z^2+1.372/z^3+0.118/z^4-0.332/z
   ^5-0.168/z^6;
10 Fn=factors( numer(Numz));
11 disp(Fn, 'Factors of the numerator of the Transfer
   Function = ');
12 Fd=factors( numer(Denz));
13 disp(Fd, 'Factors of the denominator of the Transfer
   Function = ');

```

---

**Scilab code Exa 8.10** Cascaded lattice realization of IIR digital Transfer Function

```

1 //Example 8.10
2 //Cascaded lattice realization of IIR Transfer
   Function
3 clear;
4 clc;
5 z=%z;
6 P3z= -0.2 + 0.18/z + 0.4/(z^2) + 1/(z^3);
7 D3z= 1 + 0.4/z + 0.18/(z^2) - 0.2/(z^3);
8 A3z=P3z/D3z;
9 p1=coeff( numer(P3z));
10 p=mtlbfliplr(p1);
11 disp(mtlbfliplr(p), 'The coefficients of numerator
   are = ');
12 d1=coeff( numer(D3z-1));
13 d=mtlbfliplr(d1)
14 disp((d), 'The coefficients of numerator are = ');
15 d1_1dash=(d(1)-d(3)*d(2))/(1-d(3)*d(3));
16 disp(d1_1dash, 'd1_1dash = ');
17 d2_1dash=(d(2)-d(3)*d(1))/(1-d(3)*d(3));
18 disp(d2_1dash, 'd2_1dash ');
19 d1_2dash=(d1_1dash)/(1+d2_1dash);
20 disp(d1_2dash, 'd1_2dash = ');
21 A1z=(d1_2dash + 1/z)/(1 + d1_2dash/z);
22 disp(A1z, 'A1z = ');
23 A2z=(d2_1dash + d1_1dash*1/z + 1/z^2)/(1 + d1_1dash/
   z - d2_1dash/z^2);
24 disp(A2z, 'A2z = ');

```

---

#### Scilab code Exa 8.12 Gray Markel method of realization

```

1 //Example 8.12
2 //Gray Markel method of Realisation
3 clear;
4 clc;
5 z=%z;

```

```

6 P3z= 0 + 0.44/z + 0.362/(z^2) +0.02/(z^3);
7 D3z= 0.4/z + 0.18/(z^2) - .2/(z^3);
8 Hz=P3z/D3z;
9 p1=coeff( numer(P3z));
10 p=mtlbfliplr(p1)
11 disp(mtlbfliplr(p), 'The coefficients of numerator
    are = ');
12 d1=coeff( numer(D3z));
13 d=mtlbfliplr(d1)
14 disp(mtlbfliplr(d), 'The coefficients of numerator
    are = ');
15 d1_1dash=(d(1)-d(3)*d(2))/(1-d(3)*d(3));
16 disp(d1_1dash, "d1_1dash = ");
17 d2_1dash=(d(2)-d(3)*d(1))/(1-d(3)*d(3));
18 disp(d2_1dash, "d2_1dash  ");
19 d1_2dash=(d1_1dash)/(1+d2_1dash);
20 disp(d1_2dash, "d1_2dash = ");
21 a1=p(3);
22 disp(p(3), 'a1 = ');
23 a2=p(2)-a1*d(1);
24 disp(p(2)-a1*d(1), 'a2 = ');
25 a3=p(1)-a1*d(2)-a2*d1_1dash;
26 disp(p(1)-a1*d(2)-a2*d1_1dash, 'a3 = ');
27 disp(0-a1*d(3)-a3*d1_2dash-a2*d2_1dash, 'a4 = ');

```

---

### Scilab code Exa 8.18 Cascaded lattice realization

```

1 //Example 8.18
2 //Cascaded lattice realization of Power-symmetric
   FIR Transfer Function
3 clear;
4 clc;
5 z=%z;
6
7 H5z=(1 + 0.3/z + 0.2/z^2 - 0.376/z^3 - 0.06/z^4 +

```

```
    0.2/z^5);
8  disp(H5z, 'FIR filter = ');
9  G5=horner(H5z, -1/z);
10 G5z=G5/z^5;
11 disp(G5z, 'FIR filter = ');
12 k5=0.2;
13 H3z=(1/(1+k5^2))*(H5z - k5*G5z);
14 disp(H3z, 'Synthesis eqn, H3z = ');
15 G3z=(1/(1+k5^2))*(k5*H5z + G5z);
16 disp(G3z, 'Synthesis eqn, G3z = ');
17 k=coeff( numer(G3z));
18 disp(k(4), 'k3 = ');
19 disp(k(2), 'k1 = ');
```

---

# Chapter 9

## IIR digital filter design

Scilab code Exa 9.1 Computing ripple values

```
1 //EXAMPLE 9.1
2 //pass band & stop band ripple
3 clc;
4 clear;
5 ap=0.1; //peak passband ripple in dB
6 as=35; //min. stopband attenuation in dB
7
8 //calculation of peak ripple values
9 dp=1-10-(ap/20);
10 disp(dp, 'dp = ');
11 ds=10-(as/20);
12 disp(ds, 'ds = ');
```

---

Scilab code Exa 9.2 conversion of bandedged frequencies to Normalized digital frequencies

```
1 //EXAMPLE 9.2
2 //analog passband & stopband frequencies (in KHz) :
```

```

3  clc;
4  clear;
5  ap=7;
6  as=3;
7  //Sampling frequency (in KHz):
8  FT=25;
9  //digital frequencies:
10 wp=2*%pi*ap/FT;
11 disp(wp, 'wp = ');
12 ws=2*%pi*as/FT;
13 disp(ws, 'ws = ');

```

---

### Scilab code Exa 9.3 Design of HP Digital Filter

```

1  //Example 9.3
2  //Design of HP IIR filter
3  clc;
4  clear;
5  Fp=700//Hz
6  Fs=500//Hz
7  ap=1//dB
8  as=32//dB
9  FT=2000//Hz
10 //normalized angular edge frequencies in rad/sec
11 wp=2*%pi*Fp/FT;
12 ws=2*%pi*Fs/FT;
13 //prewarp the digital edge frequencies
14 Ap1=tan(wp/2);
15 As1=tan(ws/2);
16
17 Ap=1; //assuming
18 As=(2*%pi*Ap1)*Ap/(2*%pi*As1);
19 disp(As, 'As = ')
20 //Order 'N' of the filter
21 k = Ap/As;

```

```

22 disp(1/k, '1/k = ');
23 k1 = 1/(sqrt(((10^(0.1* as))-1)/((10^(0.1*ap))-1)));
24 disp(1/k1, '1/k1 = ');
25 N = ceil(acosh(1/k1)/acosh(1/k));
26 disp(N, 'N = ');
27 disp(N, 'Order of the filter is, N = ');
28
29 e=sqrt(10^(0.1*ap)-1);
30 u=1/e + sqrt(1+(1/(e*e)));
31 a=Ap*(u^(1/N) - u^(-1/N))/2;
32 b=Ap*(u^(1/N) + u^(-1/N))/2;
33
34 for i=1:N
35     phi(i)= %pi/2 + (2*i -1)*(%pi)/(2*N);
36     p(i)=a*cos(phi(i)) + %i*b*sin(phi(i));
37 end
38 s=%s;
39 z=%z;
40 H1=1;
41 //Numerator of H(s)
42 for i=1:N
43     H1=H1*(s + p(i))
44 end
45 //Denominator of H(s)
46 H2=horner(H1,0);
47 //Transfer function
48 H=H2/H1;
49 disp(H, 'H = ')
50 //Bilinear Transformaation, s=((z-1)/(z+1));
51 Hz=horner(H,(z-1)/(z+1));
52 disp(Hz, 'The digital HP filter is Hz = ');

```

---

**Scilab code Exa 9.6** Changing passband edge frequencies to LP IIR digital frequencies



```

1 //EXAMPLE 9.6
2 //LP TO LP Transformation
3
4 clc;
5 clear;
6 z=%z;
7 w=0:0.001*%pi:%pi;
8 Glz=(0.0662272*(1+1/z)^3)/((1-0.2593284/z)
      *(1-0.6762858/z+0.3917468/(z^2)))
9 wc=0.25*%pi;//Oringinal passband edge
10 Wc=0.35*%pi;//Required passband edge
11 l=sin((wc-Wc)/2)/sin((wc+Wc)/2)
12 disp(l, 'lambda = ');
13 Gdz=horner(Glz,((1-1/z)/(1/z-1)));
14 disp(Gdz, 'The transfer function is Gdz = ');

```

---

**Scilab code Exa 9.7** Design of HP IIR Digital Filter from LP Digital Filter

```

1 //EXAMPLE 9.7
2 //LP TO HP Transformation
3
4 clc;
5 clear;
6 z=%z;
7 Glz=(0.0662272*(1+1/z)^3)/((1-0.2593284/z)
      *(1-0.6762858/z+0.3917468/(z^2)));
8 wc=0.25*%pi;//Oringinal passband edge
9 Wc=0.55*%pi;//Required passband edge
10 l=-cos((wc+Wc)/2)/cos((wc-Wc)/2);
11 disp(l, 'lambda = ');
12
13 w=0:0.001:1;
14 Ghz=horner(Glz,-((z + 1)/(1 + 1*z))); //LP TO HP
      Transformation formula
15 den=factors(denom(Ghz));

```

```

16 disp(Ghz, 'The transfer function is Gdz = ');
17 disp(den, 'the factors of the denominator are = ');

```

---

**Scilab code Exa 9.12** Minimum order of Type 2 Chebyshev HP IIR digital filter

```

1 //EXAMPLE 9.12
2 //Minimum order of type-2 Chebyshev highpass digital
   filter
3 clc;
4 clear;
5 ap = 1 //dB
6 as = 40 //dB
7 Fp = 1000 //Hz
8 Fs = 600 //Hz
9 Wp = Fp*2*%pi;
10 Ws = Fs*2*%pi;
11
12 F = 4000 //Hz
13 T=1/F;
14
15 Ap=(2/T)*(tan(Wp*T/2))
16 As=(2/T)*(tan(Ws*T/2))
17
18
19 k = Ap/As;
20 disp(1/k, '1/k = ');
21 k1 = 1/(sqrt(((10^(0.1* as))-1)/((10^(0.1*ap))-1)));
22 disp(1/k1, '1/k1 = ');
23 N = acosh(1/k1)/acosh(k); //order of the filter
24 disp(N, 'N = ');
25 disp('Since order of the filter is always an integer
   , ');
26 disp(ceil(N), 'Order of the filter is , N = ');

```

---

# Chapter 10

## FIR digital filter design

Scilab code Exa 10.1 Kaiser formula

```
1 //Example 10.01
2 //Order estimation using Kaiser's formula
3 clear;
4 clc;
5 Fp=1800; //Passband edge freq. in Hz
6 Fs=2000; //stopband edge freq. in Hz
7 ap=0.1; //peak passband ripple in dB
8 as=35; //min. stopband attenuation in dB
9 FT=12000; //Sampling freq. in Hz
10
11 //calculation of peak ripple values
12 dp=1-10-(ap/20);
13 disp(dp, 'dp = ');
14 ds=10-(as/20);
15 disp(ds, 'ds = ');
16
17 //Order of the FIR filter
18 N=(-(20*log10(sqrt(ds*dp))) - 13)/((14.6)*(Fs-Fp)/FT
    );
19 disp(ceil(N), 'Order of the filter is N = ')
```

---

### Scilab code Exa 10.2 Bellenger formula

```
1 //Example 10.01
2 // Order estimation using Bellanger 's formula
3 clear;
4 clc;
5 Fp=1800; //Passband edge freq. in Hz
6 Fs=2000; //stopband edge freq. in Hz
7 ap=0.1; //peak passband ripple in dB
8 as=35; //min. stopband attenuation in dB
9 FT=12000; //Sampling freq. in Hz
10
11 //calculation of peak ripple values
12 dp=1-10-(ap/20);
13 disp(dp, 'dp = ');
14 ds=10-(as/20);
15 disp(ds, 'ds = ');
16
17 //Order of the FIR filter
18 N=((-2*log10(10*ds*dp)) /((3)*(Fs-Fp)/FT)) -1 ;
19 disp(ceil(N), 'Order of the filter is N = ')
```

---

### Scilab code Exa 10.3 Hermann formula

```
1 //Example 10.03
2 //Order estimation using Hermann 's formula
3 clear;
4 clc;
5 Fp=1800; //Passband edge freq. in Hz
6 Fs=2000; //stopband edge freq. in Hz
7 ap=0.1; //peak passband ripple in dB
8 as=35; //min. stopband attenuation in dB
```

```

 9 FT=12000; //Sampling freq. in Hz
10
11 //calculation of peak ripple values
12 dp=1-10-(ap/20);
13 disp(dp, 'dp = ');
14 ds=10-(as/20);
15 disp(ds, 'ds = ');
16
17 a1=0.005309;
18 a2=0.07114;
19 a3=-0.4761;
20 a4=0.00266;
21 a5=0.5941;
22 a6=0.4278;
23 D_infi=((a1*(log10(dp))^2 + a2*log10(dp) + a3)*log10
      (ds))-(a4*(log10(dp))^2 + a5*(log10(dp)) + a6);
24 disp(D_infi, 'D_infi = ');
25 b1=11.01217;
26 b2=0.51244;
27 F=b1 + b2*((log10(dp))-(log10(ds)));
28 disp(F, 'F = ');
29
30 //Order of the FIR filter
31 N=(D_infi - F*((Fs-Fp)/FT)^2)/((Fs-Fp)/FT);
32 disp(ceil(N), 'Order of the filter is N = ')

```

---

#### Scilab code Exa 10.4 Order Estimation

```

1 //Example 10.04
2 //Kaiser's formula for bandpass filter
3 clear;
4 clc;
5 Fp1=300; //Passband edge freq. in Hz
6 Fs1=350; //stopband edge freq. in Hz
7 Fp2=1000; //Passband edge freq. in Hz

```

```

8 Fs2=1100; //stopband edge freq. in Hz
9
10 dp=0.004; //passband ripple in dB
11 ds=0.01; //stopband ripple in dB
12 FT=10000; //Sampling freq. in Hz
13
14 //Since  $(Fp1-Fs1) < (Fs2-Fp2)$ , bandwidth used is  $(Fp1-Fs1)$ 
15
16 //Order of the FIR filter
17  $N = (- (20 * \log_{10}(\sqrt{ds * dp})) - 13) / ((14.6) * (Fs1 - Fp1) / FT)$ ;
18 disp(ceil(N), 'Order of the filter is N = ')

```

---

**Scilab code Exa 10.6** Filter length estimation for window based design

```

1 //EXAMPLE 10.6
2 //Filter length for window -based design
3 clear;
4 clc;
5 wp=0.3*%pi; //rad/sec
6 ws=0.5*%pi; //rad/sec
7 as=40; //dB
8
9 wc=(wp+ws)/2; //cutoff frequency
10 Bw=ws-wp;
11 disp(Bw, 'Normalized transition bandwidth is = ')
12 //Hann window
13 M1=3.11*%pi/Bw;
14 disp(M1, 'Value of M = ')
15 //Hamming window
16 M2=3.32*%pi/Bw;
17 disp(M2, 'Value of M = ')
18 //Blackman window
19 M3=5.56*%pi/Bw;

```

```
20 disp(M3, 'Value of M = ')
```

---

#### Scilab code Exa 10.7 Order Estimation

```
1 //EXAMPLE 10.6
2 //Order estimation using Dolph-Chebyshev window
3 clear;
4 clc;
5 wp=0.3*%pi;//rad/sec
6 ws=0.5*%pi;//rad/sec
7 as=40;//dB
8
9 wc=(wp+ws)/2;//cutoff frequency
10 Bw=ws-wp;
11 disp(Bw, 'Normalized transition bandwidth is = ')
12
13 //Order of the filter
14 N = ((2.056*as) - 16.4)/(2.285*Bw);
15 disp(ceil(N), 'Order of the filter ,N = ')
```

---

#### Scilab code Exa 10.8 Kaiser window

```
1 //EXAMPLE 10.8
2 //Design of LP FIR filter using Kaiser window
3 clear;
4 clc;
5 wp=0.3*%pi;//rad/sec
6 ws=0.5*%pi;//rad/sec
7 as=40;//dB
8 wc=(wp+ws)/2;//cutoff frequency
9 Bw=ws-wp;
10 disp(Bw, 'Normalized transition bandwidth is = ')
11
```

```

12 ds=10^(-as/20);
13 B = (0.5842*(as-21)^0.4) + 0.07886*(as-21);
14
15 N = ceil((as - 8)/(2.285*Bw));
16 disp(N, 'Order of the filter ,N = ');
17 M=(N-1)*0.5;
18 disp(M, 'M = ');
19 w=window('kr',N,6); //Kaiser window
20 i=-M:1:M;
21
22     hn=(wc/%pi)*sinc(wc*i'/(%pi));
23     h=hn*w;
24
25 clf();
26 n=0:0.001:1;
27 [H,fr]=frmag(w,1001);
28
29 plot2d(2*fr,log10(H./max(H)),style=color('blue'))
30 set(gca(),'grid',[1 1]*color('gray'))
31 a = gca ();
32 xlabel ('w/%pi' );
33 ylabel ('Magnitude in dB' );
34 title ('Gain Response of Kaiser Window' );

```

---



# Chapter 11

## DSP Algorithm implementation

**Scilab code Exa 11.3** Reconstruction of Transfer function from Impulse response coefficients

```
1 //example 11.3
2 //Reconstruction of Transfer function from Impulse
   response coeff.
3 clear;
4 clc;
5 z=%z;
6 numz=2+6/z+3/(z^2);
7 denz=(1+1/z+2/z^2);
8 disp(numz/denz, 'Hz = ');
9 d=coeff( numer(denz) );
10 disp(d, 'coefficients of the denominator are = ')
11 h1=ldiv( numer(numz), numer(denz), 5);
12 disp(h1', 'The first five coefficients are of H(z) = '
   ');
13 for i=1:3
14     for j=1:3
15         if i>=j
16             h(i,j)=h1(i-j+1)
17         else
18             h(i,j)=0;
```

```

19         end
20     end
21 end
22 disp(h, 'h = ');
23 disp((h'*d'),'coefficients of the numerator are = '
    );

```

---

### Scilab code Exa 11.11 Cascaded lattice Filter structure

```

1 //Example 11.11
2 //Simulation of IIR cascaded lattice filter
   structure
3 clear;
4 clc;
5 z=%z;
6 P3z= 0 + 0.44/z + 0.362/(z^2) +0.02/(z^3);
7 D3z= 1 + 0.4/z + 0.18/(z^2) - 0.2/(z^3);
8 Hz=P3z/D3z;
9 p1=coeff( numer(P3z));
10 p=mtlbfliplr(p1)
11 disp(mtlbfliplr(p), 'The coefficients of numerator
    are = ');
12 d1=coeff( numer(D3z-1));
13 d=mtlbfliplr(d1)
14 disp(mtlbfliplr(d), 'The coefficients of numerator
    are = ');
15 d1_1dash=(d(1)-d(3)*d(2))/(1-d(3)*d(3));
16 disp(d1_1dash, "d1_1dash = ");
17 d2_1dash=(d(2)-d(3)*d(1))/(1-d(3)*d(3));
18 disp(d2_1dash, "d2_1dash  ");
19 d1_2dash=(d1_1dash)/(1+d2_1dash);
20 disp(d1_2dash, "d1_2dash = ");
21 a1=p(3);
22 disp(p(3), 'a1 = ');
23 a2=p(2)-a1*d(1);

```

```
24 disp(p(2)-a1*d(1), 'a2 = ');
25 a3=p(1)-a1*d(2)-a2*d1_1dash;
26 disp(p(1)-a1*d(2)-a2*d1_1dash, 'a3 = ');
27 disp(0-a1*d(3)-a3*d1_2dash-a2*d2_1dash, 'a4 = ');
```

---

# Chapter 12

## Analysis of Finite Wordlength Effects

Scilab code Exa 12.3 Signal to Quantisation Noise Ratio

```
1 //Example 12.3
2 //Signal-to-quantization Noise ratio
3 clear;
4 clc;
5 b=[7 9 11 13 15]; //Given values of b
6 K=[4 6 8]; //Given values of K
7 for i=1:5
8     for j=1:3
9         SNR(j,i)=6.02*b(i)+16.81-20*log10(K(j));
10    end
11 end
12 disp(SNR, 'SNR,A/D = ');
```

---

# Chapter 13

## Multirate Digital Signal Processing Fundamentals

Scilab code Exa 13.1 Up sampling operation

```
1 //Example 13.1
2 //Upsampling Operation
3 clear;
4 clc;
5 clf();
6 a=gca();
7 figure(0);
8 n=[0:0.1:4.9];
9 a.x_location="origin";
10 x=sin(%pi*n);
11 plot2d3(n,x,2);
12 xtitle('The sine wave','n','sin(x)');
13 plot(n,x,'r. ');
14 //Up sampling
15 //Up sampling value - user input
16 figure(1);
17 L=input(" The up sampling value ");
18
19 a.x_location="origin";
```

```
20 x1=sin(%pi*n/L);
21 plot2d3(n,x1,5);
22 plot(n,x1,'r. ');
23 xtitle('The sine wave', 'n', 'sin(x/L)');
```

---

### Scilab code Exa 13.2 Down sampling operation

```
1 //Example 13.2
2 //Downsampling Operation
3 clear;
4 clc;
5 clf();
6 a=gca();
7 figure(0);
8 n=[0:0.1:4.9];
9 a.x_location="origin";
10 x=sin(%pi*n);
11 plot2d3(n,x,2);
12 xtitle('The sine wave', 'n', 'sin(x)');
13 plot(n,x,'r. ');
14 //Down sampling
15 //Down sampling – user input
16 figure(1);
17 M=input(" The down sampling factor ");
18 a.x_location="origin";
19 x1=sin(%pi*n*M);
20 plot2d3(n,x1,1);
21 plot(n,x1,'r. ');
22 xtitle('The sine wave', 'n', 'sin(x*M)');
```

---

### Scilab code Exa 13.6 Decimator Computation complexity

```
1 //Example 13.6
```

```

2 //Decimator computational complexity
3 clear;
4 clc;
5 //no. of multiplications/sec =Rm
6 FT = input("Sampling Frequency");
7 N = input("The order of the FIR Hz");
8 Rm1 = N*FT;
9 disp(Rm1, 'Rm, FIR = ');
10 //M = factor of Down sampler
11 M = input("The Down Sampling factor ");
12 disp(Rm1/M, 'Rm, FIR-DEC = ');
13 K = input("The order of the IIR Hz");
14 Rm2 = (2*K + 1)*FT;
15 disp(Rm2, 'Rm, IIR = ');
16 disp( (K*FT + ((K+1)*FT/M)), 'Rm, IIR-DEC = ');

```

---

# Chapter 14

## Applications of Digital Signal Processing

Scilab code Exa 14.1 Effect of DFT length

```
1 //Example 14.1
2 //EFFECT OF DFT LENGTH ON SPECTRAL ANALYSIS
3 clear;
4 clc;
5 N=16;
6 n=0:N-1;
7 f1=0.22;
8 f2=0.34;
9
10 R = input("R point DFT(R E [16,128]) = "); //
    Input f1 = 64
11 if R >= N
12     x=0.5*(sin(2*%pi*f1*n')) + sin(sin(2*%pi*f2*n'))
    ;
13     x=[x', zeros(1,R-length(n))];
14     disp(x,'the sequence is :');
15     for n=0:R-1
16         for k=0:R-1
17             W(n+1,k+1) = exp(-(%i*2*%pi*k/R)*n);
```



```

18         end
19     end
20     X = W*x';
21     disp(X, 'DFT is , X = ')
22 else
23     disp('invalid computation ');
24 end
25 m=0:R-1;
26 clf();
27 figure(0)
28 a = gca();
29 plot2d3(m, abs(X), 2) // plotting DFT of sequence
30 plot(m, abs(X), 'r. ')
31 a.x_location = 'origin';
32 a.y_location = 'origin';
33 poly1 = a . children (1) . children (1) ;
34 poly1.thickness = 2.5;
35 xtitle('original sequence ', 'n', 'x[n] ');

```

---

#### Scilab code Exa 14.2 Effect of DFT length

```

1 //Example 14.2
2 //EFFECT OF DFT LENGTH ON SPECTRAL ANALYSIS
3 clear;
4 clc;
5 N=16;
6 n=0:N-1;
7 f1=input("Enter f1 value between 0.28 to 0.31 = ");
8 //Input f1 = 64
9 f2=0.34;
10 R = 128//
11 //DFT of the sequence x[n]
12 x=0.5*(sin(2*%pi*f1*n')) + sin((2*%pi*f2*n'));
13 x=[x', zeros(1, R-length(n))];

```

```

14     disp(x, 'the sequence is :');
15     for n=0:R-1
16         for k=0:R-1
17             W(n+1,k+1) = exp(-(i*2*pi*k/R)*n);
18         end
19     end
20     X = W*x';
21     disp(X, 'DFT is , X = ')
22 //plotting DFT of sequence
23 m=0:R-1;
24 clf();
25 figure(0)
26 a = gca();
27 plot2d3(m, abs(X), 2)
28 plot(m, abs(X), 'r. ')
29 a.x_location = 'origin';
30 a.y_location = 'origin';
31 poly1 = a . children (1) . children (1) ;
32 poly1.thickness = 2.5;
33 xtitle('original sequence', 'n', 'x[n]');

```

---