Scilab Manual for Digital Signal Processing by Prof Akhtar Nadaf Electronics and Telecommunication Engineering Nagesh Karajagi Orchid College Of Engineering & Technology, Solapur[1]

Solutions provided by
Mr Akhtar Nadaf
Electronics and Telecommunication Engineering
N K Orchid College Of Engineering & Technology

April 21, 2026

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

Contents

List of Experiments

List of Figures

Experiment: 1

Waveform generation using discrete time signals

Scilab code Solution 1.1 Waveform generation using DT signals

```
1  // Expt 1. Waveform generation using discrete time
       signals using Scilab
2  // O.S. Windows 10
3  ////Scilab 6.0.0
4  //Generation of unit step Discrete signal
5  clear;
6  clc;
7  t=0:4;
8  y=ones(1,5);
9  subplot(3,2,1);
10 plot2d3 (t,y);
11 xlabel('n');
12 ylabel('u(n)');
13 title('Unit Step Discrete Signal');
14
15 //Generation of Unit Ramp Discrete signal
16 n1=0:8;
```

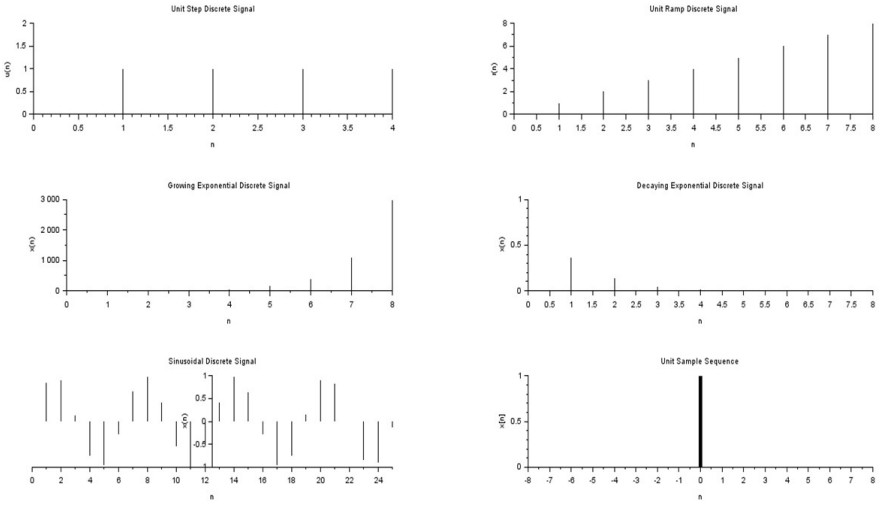

Figure 1.1: Waveform generation using DT signals

```
17 y1=n1;
18 subplot(3,2,2);
19 plot2d3 (n1,y1);
20 xlabel('n');
21 ylabel('r(n)');
22 title('Unit Ramp Discrete Signal');
23
24 //Generation of Growing Exponential Discrete signal
25 n1=0:8;
26 y1=n1;
27 y2=exp(n1);
28 subplot(3,2,3);
29 plot2d3 (n1,y2);
30 xlabel('n');
31 ylabel('x(n)');
32 title('Growing Exponential Discrete Signal');
33
34 //Generation of Decaying Exponential Discrete signal
35 n1=0:8;
36 y1=n1;
```

```
y2=exp(-n1);
subplot(3,2,4);
plot2d3 (n1,y2);
xlabel('n');
ylabel('x(n)');
title('Decaying Exponential Discrete Signal');

//Generation of sinusoidal discrete signal
n1=0:25;
y1=n1;
y2=sin(n1);
subplot(3,2,5);
plot2d3 (n1,y2);
xlabel('n');
ylabel('x(n)');
title('Sinusoidal Discrete Signal');

//Generation of unit impulse sequence
l=7;
n=-l:l;
x=[zeros(1,l),1,zeros(1,l)];
b=gca();
b.y_location="middle";
subplot(3,2,6);
plot2d3('gnn',n,x);
a= gce ();
a. children (1) . thickness =5;
xtitle ('Unit Sample Sequence','n','x[n]');
```

Experiment: 2

Z-transform and pole zero plot of a system

Scilab code Solution 2.1 Z transform of DT sequence

```
1  //Expt2: To draw the pole-zero plot
2  //O.S: Windows 10;
3  //Scilab: 6.0.0
4  clear;
5  clc ;
6  //Z- transform of [1 0 3 -1 2]
7  clear;
8  clc ;
9  close ;
10 function[za]=ztransfer(sequence,n)
11 z=poly(0,'z','r')
12 za=sequence*(1/z)^n'
13 endfunction
14 x1=[1 0 3 -1 2];
15 n=0:length(x1)-1;
16 zz=ztransfer(x1,n);
17 //Display the result in command window
18 disp (zz,"Z-transform of sequence is:");
19 // Expected Output:
```

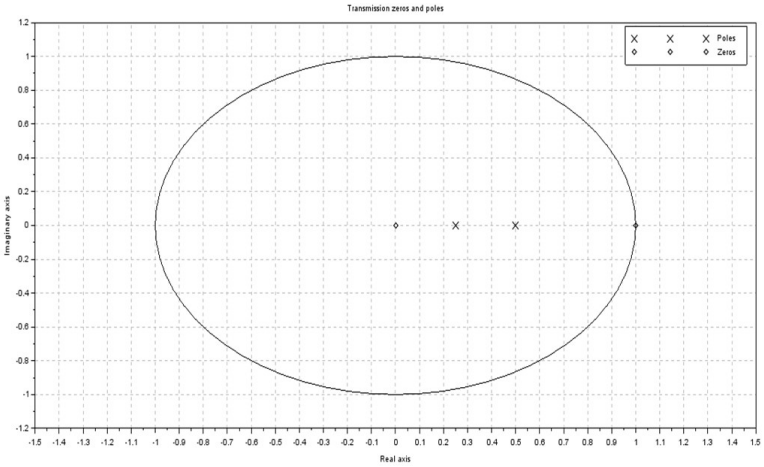

Figure 2.1: Pole Zero Plot of a system

```
20 //Z-transform of sequence is:
21 //                2    4
22 //   2 - z + 3z + z
23 //   ----------------
24 //           4
25 //          z
26 disp('ROC is the entire plane except z = 0');
27 //ROC is the entire plane except z = 0
```

Scilab code Solution 2.2 Pole Zero Plot of a system

```
1 //Expt2: To draw the pole-zero plot
2 //O.S: Windows 10;
3 //Scilab: 6.0.0
4 clear;
5 clc ;
```

```
6  close ;
7  z=%z
8  H1Z=((z)*(z-1))/((z-0.25)*(z-0.5));
9  xset('window',1);
10 plzr(H1Z);
```

Experiment: 3

Linear convolution

Scilab code Solution 3.1 Linear Convolution

```
1  //Experiment no 3
2  //Linear Convolution
3  // SciLab version : 6.0.0
4  // O.S. : Windows 10
5  clc;
6  close ;
7  t=0:6;
8  x=[1,2,1,2,1,3,2];
9  subplot(2,2,1);
10 plot2d3 (t,x);
11 xlabel('n');
12 ylabel('x(n)');
13 title('Input sequence x(n)');
14
15 t=0:5;
16 h=[1,-1,2,-2,1,1];
17 subplot(2,2,2);
18 plot2d3 (t,h);
19 xlabel('n');
```

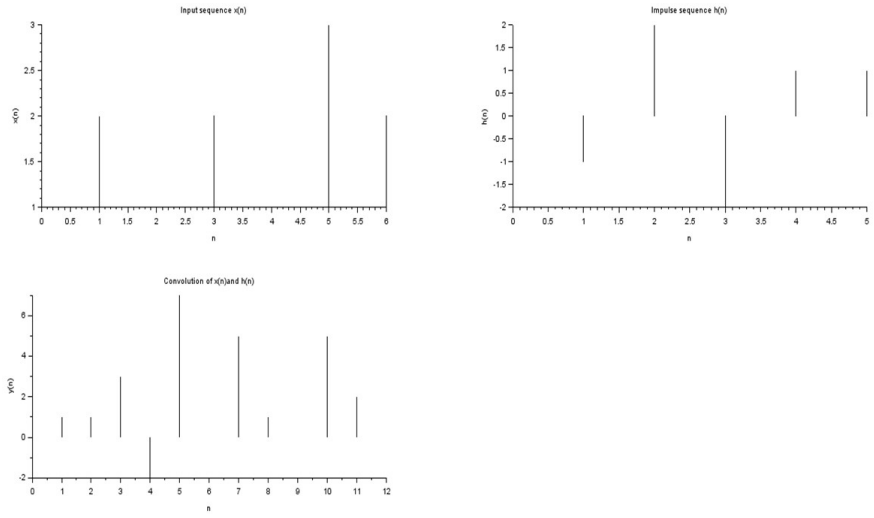

Figure 3.1: Linear Convolution

```
20 ylabel('h(n)');
21 title('Impulse sequence h(n)');
22
23 m = length(x);
24 n = length(h);
25 //Direct Convolution Sum Formula
26 for i = 1:n+m-1
27     conv_sum = 0;
28     for j = 1:i
29         if (((i-j+1) <= n)&(j <= m))
30             conv_sum = conv_sum + x(j)*h(i-j+1);
31         end;
32         y(i) = conv_sum;
33     end;
34 end;
35 disp(y,'y=')
36 subplot(2,2,3);
37 l=length(y);
38 t=0:(l-1);
39 plot2d3 (t,y);
```

```
40 xlabel('n');
41 ylabel('y(n)');
42 title('Convolution of x(n)and h(n)');
```

Experiment: 4

Auto co-relation and cross co-relation

Scilab code Solution 4.1 Auto correlation

```
1  //Experiment no 4
2  //Auto Correlation
3  // SciLab version : 6.0.0
4  // O.S. : Windows 10
5  clear;
6  clc;
7  close;
8  x = input('Enter the given discrete time sequence');
       // Enter a sequence x(n)={1,2,3,4}
9  l = length(x);
10 t=0:l-1;
11 subplot(1,2,1);
12 plot2d3 (t,x);
13 xlabel('n');
14 ylabel('x(n)');
15 title('Input sequence x(n)');
16 h = zeros(1,l);
```

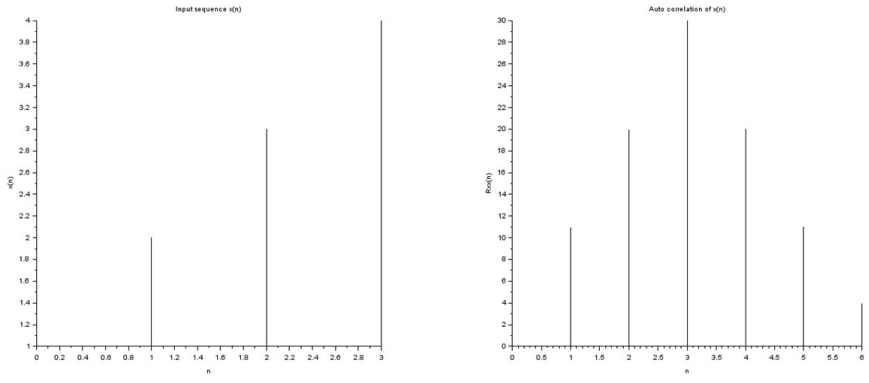

Figure 4.1: Auto correlation

```
for i = 1:l
  h(l-i+1) = x(i);
end
N = 2*l-1;
Rxx = zeros(1,N);
for i = l+1:N
   h(i) = 0;
end
for i = l+1:N
    x(i) = 0;
end
for n = 1:N
  for k = 1:N
    if(n >= k)
      Rxx(n) = Rxx(n)+x(n-k+1)*h(k);
    end
  end
end
disp(Rxx,'Auto Correlation Result is');//Expected
   output Rxx(n) ={11,20,30,20,11}
```

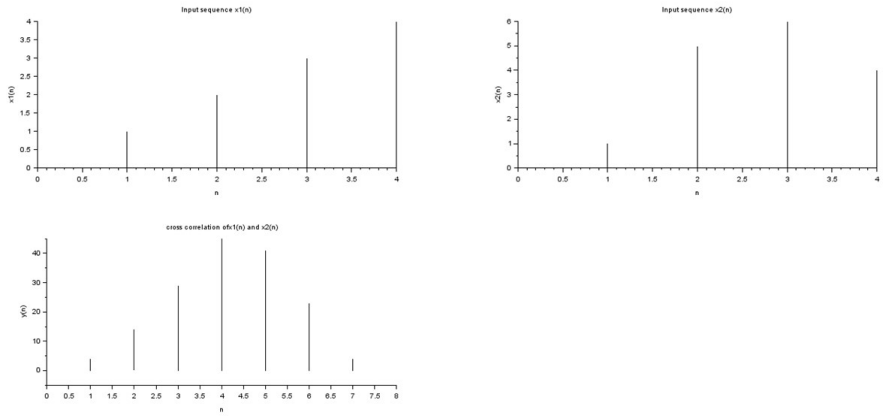

Figure 4.2: Cross corelation

```
36 L=length(Rxx);
37 t=0:L-1;
38 subplot(1,2,2);
39 plot2d3 (t,Rxx);
40 xlabel('n');
41 ylabel('Rxx(n)');
42 title('Auto correlation of x(n)');
```

Scilab code Solution 4.2 Cross corelation

```
1 //Experiment no 4b
2 //cross correlation
3 // SciLab version : 6.0.0
4 // O.S. : Windows 10
5 clc;
6 close ;
```

```
t1=0:4;
x1=[0,1,2,3,4];
subplot(2,2,1);
plot2d3 (t1,x1);
xlabel('n');
ylabel('x1(n)');
title('Input sequence x1(n)');

t2=0:4;
x2=[0,1,5,6,4];
subplot(2,2,2);
plot2d3 (t2,x2);
xlabel('n');
ylabel('x2(n)');
title('Input sequence x2(n)');

y=xcorr(x1,x2);
l=length(y);
t3=0:l-1;
subplot(2,2,3);
plot2d3 (t3,y);
xlabel('n');
ylabel('y(n)');
title('cross correlation ofx1(n) and x2(n)');
```

Experiment: 5

Implementation of DFT and IDFT

Scilab code Solution 5.1 Implementation of DFT

```
1 // Expt 5. Implementation of  8 point DFT
2 // O.S. Windows 10
3 ////Scilab 6.0.0
4
5 clear;
6 clc;
7 x1=input('Enter a sequence');// input a sequence x1
    ={1,1,1,1,0,0,0,0}
8 //DFT Computation
9 X1 = fft (x1 , -1);
10
11 disp(X1,"X1[k]=");//Expected outpput sequences X1[k]
     in command window {4,1-2.4142136i,0,1-0.4142136i
    ,0,1+0.4142136i,0,1+2.4142136i}
12 mag = abs (X1);
13 subplot(1,2,1);
14 plot2d3 (mag);
```

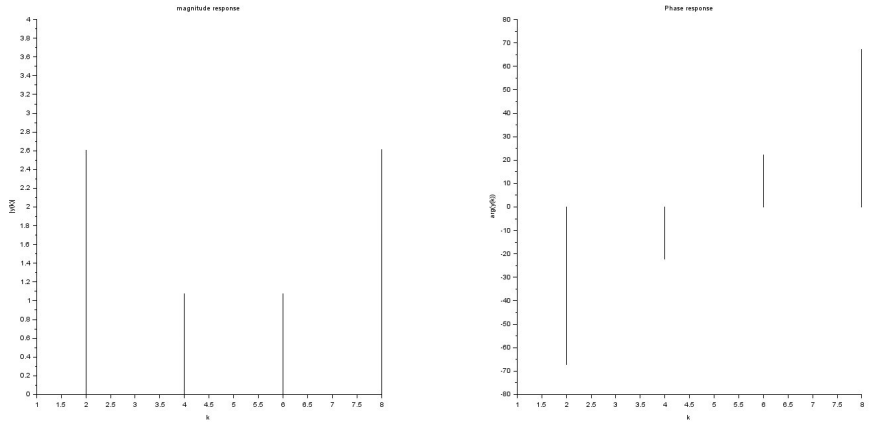

Figure 5.1: Implementation of DFT

```
15 xlabel('k');
16 ylabel('|y(k)|');
17 title('magnitude response');
18
19 x1= atan ( imag (X1),real (X1));
20 phase =x1 *(180/ %pi );
21 subplot(1,2,2);
22 plot2d3 (phase);
23 xlabel('k');
24 ylabel('arg(y(k))');
25 title('Phase response');
```

Scilab code Solution 5.2 Implementation of IDFT

```
1 // Expt 5. IDFT of sequence X[k]=[5,0,1-j,0,1,0,1+j
    ,0]
2 // O.S. Windows 10
3 ////Scilab 6.0.0
4 clear;
```

```
clc ;
clear;
clc ;
j=sqrt(-1);
X = [5,0,1-j,0,1,0,1+j,0];
//IDFT Computation
x = fft (X , 1);
//Display sequences x[n]in command window
disp(x,"x[n]=");
// outputx[n]=[1,0.75,0.5,0.25,1,0.75,0.5,0.25]
```

Experiment: 6

circular convolution using FFT

Scilab code Solution 6.1 Circular Convolution using FFT

```
1  // Expt 6. Circular Convolution using FFT
2  // O.S. Windows 10
3  ////Scilab 6.0.0
4  //x1[n]=[1,-1,-2,3,-1]
5  //x2[n]=[1,2,3]
6  clear;
7  clc ;
8  close ;
9  x1=[1,-1,-2,3,-1];
10 x2=[1,2,3];
11 //Loop for zero padding the smaller sequence out of
      the two
12 n1=length(x1);
13 n2=length(x2);
14 n3=n2-n1;
15 if (n3>=0) then
16  x1=[x1,zeros(1,n3)];
17 else
18  x2=[x2,zeros(1,-n3)];
```

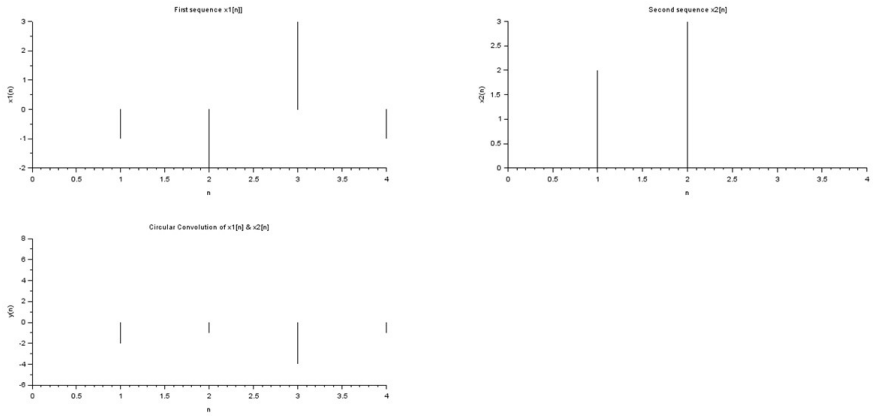

Figure 6.1: Circular Convolution using FFT

```
19 end
20 //DFT Computation
21 X1=fft(x1,-1);
22 X2=fft(x2,-1);
23 Y=X1.*X2;
24 //IDFT Computation
25 y=fft(Y,1);
26 n4=length(y);
27 //Display sequence y[n] in command window
28 disp(y,"y[n]=");
29 // Plotting of sequences
30 t=0:n1-1;
31 subplot(2,2,1);
32 plot2d3 (t,x1);
33 xlabel('n');
34 ylabel('x1(n)');
35 title('First sequence x1[n]]');
36
37 t1=0:n1-1;
38 subplot(2,2,2);
```

```
39 plot2d3 (t1,x2);
40 xlabel('n');
41 ylabel('x2(n)');
42 title('Second sequence x2[n]');
43
44 t2=0:n1-1;
45 subplot(2,2,3);
46 plot2d3 (t1,y);
47 xlabel('n');
48 ylabel('y(n)');
49 title('Circular Convolution of x1[n] & x2[n]');
```

Experiment: 7

Fast convolution using Overlap add/Overlap save method

Scilab code Solution 7.1 Fast convolution using overlap save method

```
// Expt 7 Fast convolution using overlap Save method
//O.S. Windows 10
//Scilab 6.0.0
clc;
clear all;
x =[1,2,-1,2,3,-2,-3,-1,1,1,2,-1];
h =[1,2,3,-1];
n1 = length(x);
n2 = length(h);
N = n1+n2-1;
h1 = [h zeros(1,N-n1)];
n3 = length(h1);
y = zeros(1,N);
x1 = [zeros(1,n3-n2) x zeros(1,n3)];
H = fft(h1,-1);
for i = 1:n2:N
y1 = x1(i:i+(2*(n3-n2)));
```

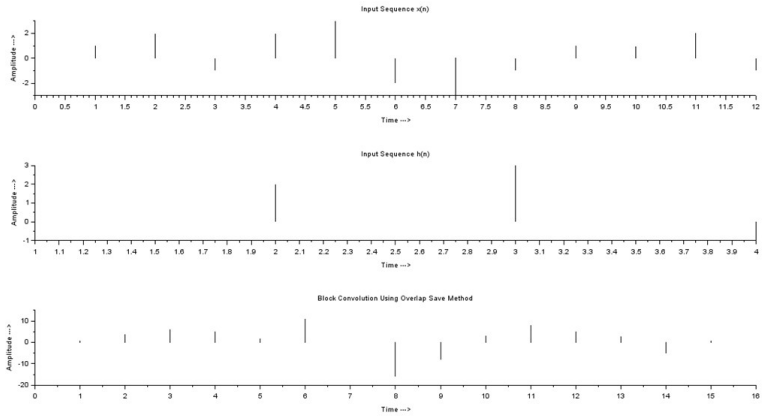

Figure 7.1: Fast convolution using overlap save method

```
y2 = fft(y1);
y3 = y2.*H;
y4 = round(fft(y3,1));
y(i:(i+n3-n2)) = y4(n2:n3);
end
subplot(3,1,1);
plot2d3(x(1:n1));
title('Input Sequence x(n)');
xlabel('Time ---->');
ylabel('Amplitude ---->');
subplot(3,1,2);
plot2d3(h(1:n2));
title('Input Sequence h(n)');
xlabel('Time ---->');
ylabel('Amplitude ---->');
subplot(3,1,3);
disp('Fast Convolution Using Overlap Save Method = ');
disp(y(1:N));
plot2d3(y(1:N));
```

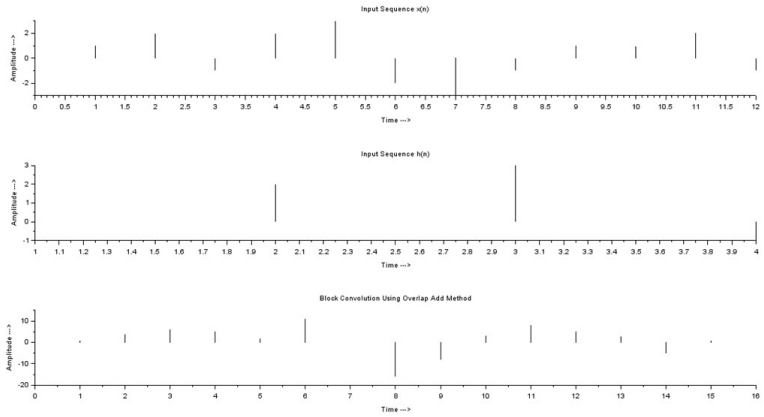

Figure 7.2: Fast convolution using overlap add method

```
37 title('Block Convolution Using Overlap Save Method')
   ;
38 xlabel('Time ---->');
39 ylabel('Amplitude ---->');
40
41 // result:Fast Convolution Using Overlap Save Method
   =
42 //    1    4    6    5    2    11    0    -16
      -8    3    8    5    3    -5    1
```

Scilab code Solution 7.2 Fast convolution using overlap add method

```
1 // Expt 7 Fast convolution using overlap add method
2 //O.S. Windows 10
3 //Scilab 6.0.0
4
```

```
clc;
clear;
x = [1,2,-1,2,3,-2,-3,-1,1,1,2,-1];
h = [1,2,3,-1];
n1 = length(x);
n2 = length(h);
N = n1+n2-1;
y = zeros(1,N);
h1 = [h zeros(1,n2-1)];
n3 = length(h1);
y = zeros(1,N+n3-n2);
H = fft(h1,-1);
for i = 1:n2:n1
if i<=(n1+n2-1)
x1 = [x(i:i+n3-n2) zeros(1,n3-n2)];
else
x1 = [x(i:n1) zeros(1,n3-n2)];
end
x2 = fft(x1,-1);
x3 = x2.*H;
x4 = round(fft(x3,1));
if (i==1)
    y(1:n3) = x4(1:n3);
else
y(i:i+n3-1) = y(i:i+n3-1)+x4(1:n3);
end
end
subplot(3,1,1);
plot2d3(x(1:n1));
title('Input Sequence x(n)');
xlabel('Time ---->');
ylabel('Amplitude ---->');
subplot(3,1,2);
plot2d3(h(1:n2));
title('Input Sequence h(n)');
xlabel('Time ---->');
ylabel('Amplitude ---->');
subplot(3,1,3);
```

```
43 disp('Fast Convolution Using Overlap Add Method = ')
   ;
44 disp(y(1:N));
45 plot2d3(y(1:N));
46 title('Fast Convolution Using Overlap Add Method');
47 xlabel('Time ---->');
48 ylabel('Amplitude ---->');
49 // Result:Fast Convolution Using Overlap Add Method
   =
50 //    1.    4.    6.    5.    2.    11.    0.   -16.
      -8.    3.    8.    5.    3.   -5.    1.
```

Experiment: 8

Realization of FIR system

Scilab code Solution 8.1 Program to determine filter coefficients obtained by sampling

```
1  // Expt 8. Program to determine filter coefficients
       obtained by sampling:
2  // O.S. Windows 10
3  //Scilab 6.0.1
4  clear;
5  clc ;
6  close ;
7  N=7;
8  U=1;          //Zero Adjust
9  for n=0+U:1:N-1+U
10 h(n)=(1+2*cos(2*%pi*(n-U-3)/7))/N
11 end
12 disp(h,"Filter Coefficients ,h(n)=")
13 //Filter Coefficients ,h(n)=
14
15 //   -0.1145625
16 //    0.0792797
17 //    0.3209971
18 //    0.4285714
19 //    0.3209971
```

```
20 //     0.0792797
21 //    -0.1145625
```

Experiment: 9

Design of FIR filter using frequency sampling method.

Scilab code Solution 9.1 Design of FIR LPF using frequency sampling method

```
1  //Exp 9. FIR LPF using frequency Sampling Method
2  //O.S. Windows 10;
3  // Scilab 6.0.0.
4  clc ;
5  clear ;
6  N =15;
7  U=1;
8  for n=0+U:1:N-1+U
9  h(n)=(1+cos(2*%pi*(7-n)/N))/N;
10 end
11 [hz,f]=frmag(h,256);
12 hz_dB=20*log10(hz)./max(hz);
13 figure;
14 plot(2*f,hz_dB);
15 a=gca();
16 xlabel('Frequency wpi');
```

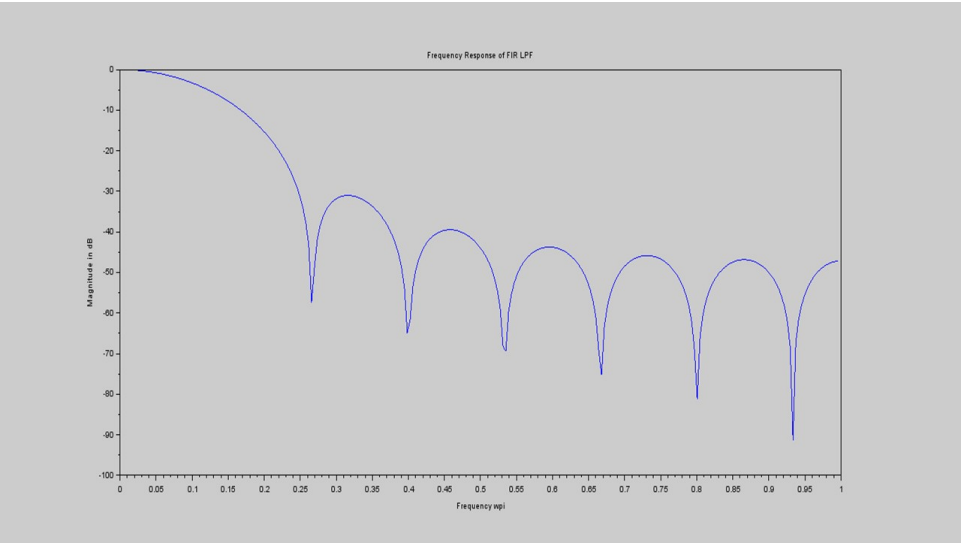

Figure 9.1: Design of FIR LPF using frequency sampling method

```
17 ylabel('Magnitude in dB') ;
18 title ('Frequency Response of FIR LPF');
```

Experiment: 10

Design of FIR filter using windowing technique.

Scilab code Solution 10.1 FIR Filter using rectangular window

```
1 //Expt. 10 Design of FIR filter (Band Pass) using
     windowing technique (Kaiser Window)
2 // O.S. Windows 10
3 // Scilab 6.0.0.
4 clear;
5 clc ;
6 close ;
7 wsf=200*%pi;//rad/sec
8 ws1=20*%pi;//rad/sec
9 ws2=80*%pi;//rad/sec
10 wp1=40*%pi;//rad/sec
11 wp2=60*%pi;//rad/sec
12 as=30//dB
13 ap=0.5//dB
14 B=min(wp1-ws1,ws2-wp2);
15 wc1=wp1-B/2;
16 wc2=wp2+B/2;
```

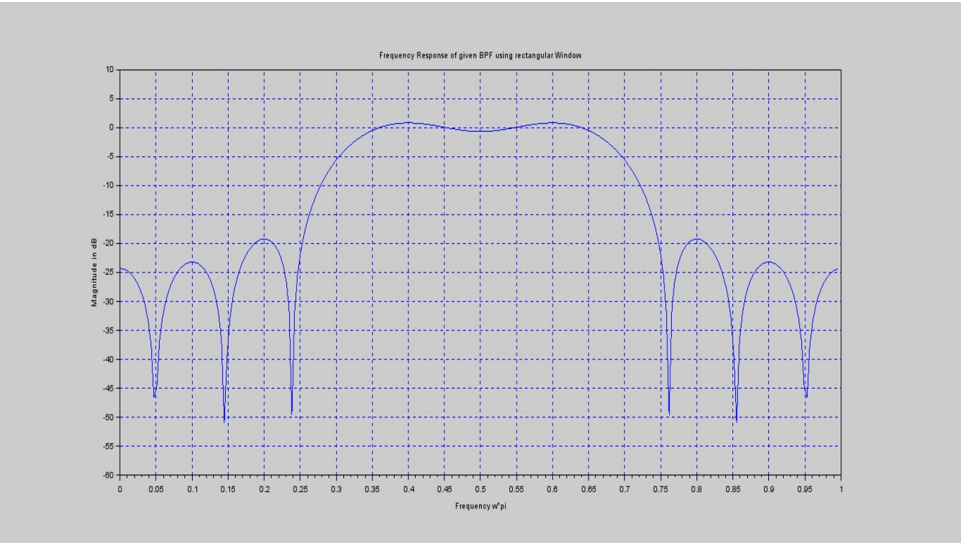

Figure 10.1: FIR Filter using rectangular window

```
17 wc1=wc1*2*%pi/wsf;
18 wc2=wc2*2*%pi/wsf;
19 delta1=10^(-0.05*as);
20 delta2=(10^(0.05*as)-1)/(10^(0.05*as)+1);
21 delta=min(delta1,delta2);
22 alphas=-20*log10(delta);
23 alpha=0.5842*(alphas-21)^0.4+0.07886*(alphas-21)
24 D=(alphas-7.95)/14.36;
25 N1=wsf*D/B+1;
26 N=ceil(N1);
27 U=ceil(N/2);
28 win_l=window('re',N,alpha);
29 for n=-floor(N/2)+U:1:floor(N/2)+U
30 if n==ceil(N/2);
31 hd(n)=0.4;
32 else
33 hd(n)=(sin(0.7*%pi*(n-U))-sin(0.3*%pi*(n-U)))/(%pi*(
      n-U));
34 end
35 h(n)=hd(n)*win_l(n);
```

```
end
[hzm ,fr ]= frmag (h ,256) ;
hzm_dB = 20* log10 (hzm)./ max ( hzm );
figure
plot (2*fr , hzm_dB )
a= gca ();
xlabel ('Frequency w*pi');
ylabel ('Magnitude in dB');
title ('Frequency Response of given BPF using
   rectangular Window');
xgrid (2);
disp(h,"Filter Coefficients ,h(n)=");
```

Experiment: 11

Design of IIR filter using impulse invariant technique.

Scilab code Solution 11.1 Design of IIR Filter using Impulse Invarient technique

```
1  //Expt.11:To Design the Filter using Impulse
       Invarient Method
2  // O.S. Windows 10
3  //Scilab: 6.0.0
4  clear;
5  clc ;
6  close ;
7  s=%s;
8  T=0.2;
9  HS=10/(s^2+7*s+10);
10 elts=pfss(HS);
11 disp(elts,'Factorized HS = ');
12 //Outputs:
13 //Factorized HS =
14 //              (1)
15 //   3.3333333
16 //  ------------
17 //      2 + s
```

```
18 // (2)
19 //   -3.3333333
20 // ------------
21 //    5 + s
22
23 //The poles comes out to be at -5 and -2
24 p1=-5;
25 p2=-2;
26 z=%z;
27 HZ=T*((-3.33/(1-%e^(p1*T)*z^(-1)))+(3.33/(1-%e^(p2*T
      )*z^(-1))))
28 disp(HZ,'HZ = ');
29 //Result:
30 //HZ =
31 //           0.2014254z
32 //    ---------------------------------
33 //                                  2
34 //    0.2465970 - 1.0381995z + z
```

Experiment: 12

Design of IIR filters using Bilinear transformation/Butterworth Technique.

Scilab code Solution 12.1 IIR filter design using Bilinear Transformation Technique

```
1 //Expt 12 Design of IIR filters using Bilinear
    transformation/Butterworth Technique.
2 //To Find out Bilinear Transformation of HS=2/((s+1)
    *(s+2)*(s+3))
3 // O.S. Windows 10;
4 //Scilab 6.0.0
5 clear;
6 clc ;
7 close ;
8 s=%s;
9 z=%z;
10 HS=2/((s+1)*(s+2)*(s+3));
11 T=1;
12 HZ=horner(HS,(2/T)*(z-1)/(z+1));
```

```
13 disp(HZ,'H(z) =');
14
15 //H(z) =
16
17 //..  .       2      3
18 //    2 + 6z + 6z + 2z
19 //    ------------------
20 //                2      3
21 //     -4z - 8z + 60z
```

Experiment: 13

Design of IIR Filters Chebyshev

Scilab code Solution 13.1 To Design an analog Chebyshev Filter with Given Specifications

```
1  //Expt 13 To Design an analog Chebyshev Filter with
       Given Specifications
2  // O.S. Windows 10;
3  //Scilab 6.0.0
4  clear;
5  clc ;
6  //
7  os=2;
8  op=1;
9  ap=3;//db
10 as=16;//db
11 e1=1/sqrt(2);
12 l1=0.1;
13 epsilon=sqrt(1/(e1^2)-1);
14 lambda=sqrt(1/(l1^2)-1);
15 N=acosh(lambda/epsilon)/acosh(os/op);
16 disp(ceil(N),'Order of the filter , N =');
17
```

```
18 // Result:
19 //Order of the filter, N =
20
21 //    3.
```