Scilab Manual for
Digital Signal Processing
by Dr R.senthilkumar
Electronics and Telecommunication Engineering
Government College Of Engineering Erode[1]

Solutions provided by
Dr R.senthilkumar
Electronics and Telecommunication Engineering
Government College Of Engineering Erode

November 24, 2024

[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

Generation of elementary Discrete-Time sequences

Scilab code Solution 1.1 Unit Sample Sequence

```
//1.GENERATION OF ELEMENTARY DISCRETE TIME SEQUENCE
//1.1 Caption:Unit Sample Sequence
clear;
clc;
close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),1,zeros(1,L)];
b = gca();
b.y_location = "middle";
plot2d3('gnn',n,x)
a=gce();
a.children(1).thickness =4;
xtitle('Graphical Representation of Unit Sample
   Sequence','n','x[n]');
```

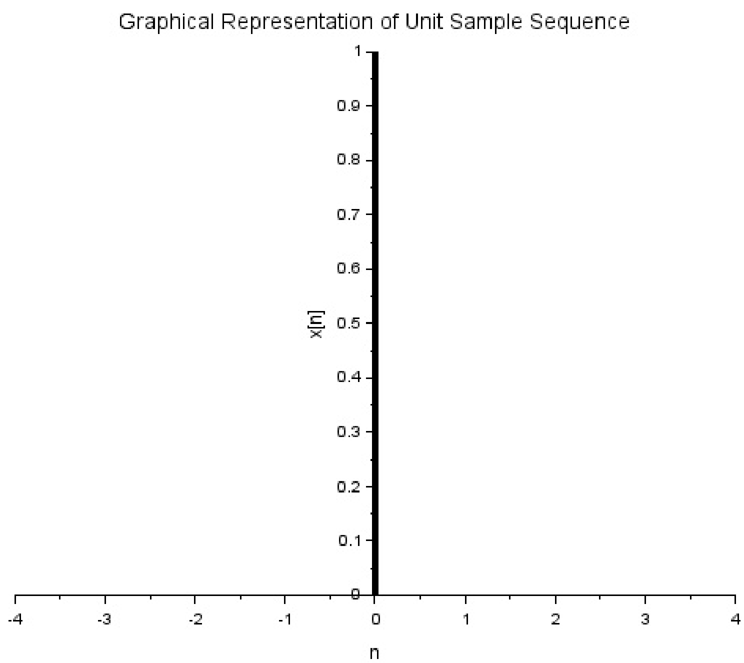

Figure 1.1: Unit Sample Sequence

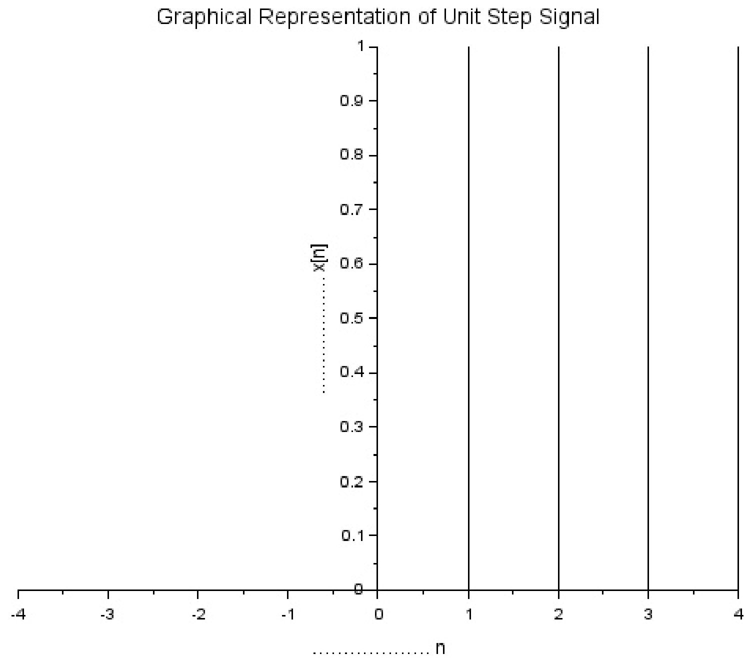

Figure 1.2: Unit Step Sequence

Scilab code Solution 1.2 Unit Step Sequence

```
//1.GENERATION OF ELEMENTARY DISCRETE TIME SEQUENCE
//1.2 Caption: Unit Step Sequence
clear;
clc;
close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),ones(1,L+1)];
```

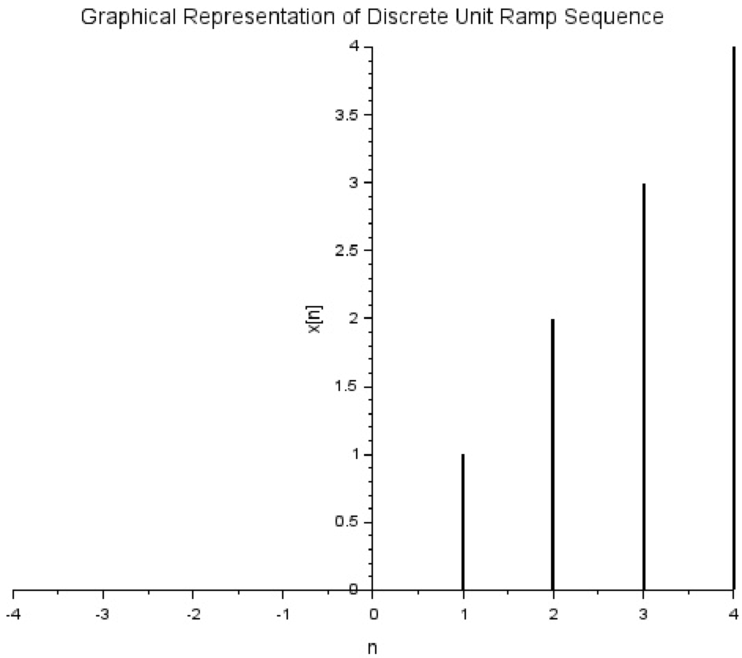

Figure 1.3: Discrete Ramp Sequence

```
a=gca();
a.y_location = "middle";
plot2d3('gnn',n,x)
title('Graphical Representation of Unit Step Signal'
   )
xlabel('                    . n');
ylabel('                    . x[n]');
```

Scilab code Solution 1.3 Discrete Ramp Sequence

```
//1.GENERATION OF ELEMENTARY DISCRETE TIME SEQUENCE
```

```
//1.3 Caption: Discrete Ramp Sequence
clear;
clc;
close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),0:L];
b = gca();
b.y_location = 'middle';
plot2d3('gnn',n,x)
a=gce();
a.children(1).thickness =2;
xtitle('Graphical Representation of Discrete Unit
    Ramp Sequence','n','x[n]');
```

Scilab code Solution 1.4 Exponent Increasing Signal

```
//1.GENERATION OF ELEMENTARY DISCRETE TIME SEQUENCE
//1.4 Caption:Exponentially Increasing Signal
clear;
clc;
close;
a =1.5;
n =1:10;
x = (a)^n;
a=gca();
a.thickness = 2;
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Exponentially
    Increasing Signal','n','x[n]');
```

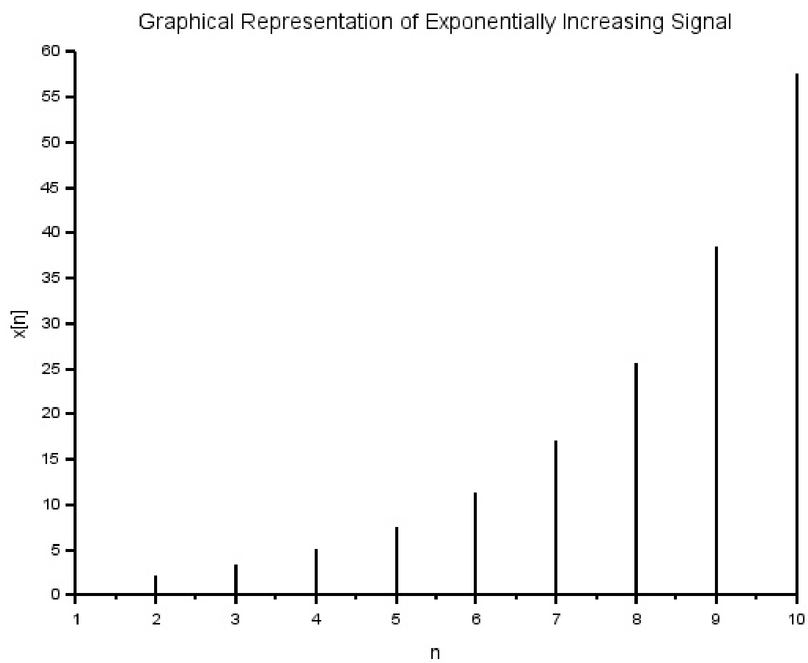

Figure 1.4: Exponent Increasing Signal

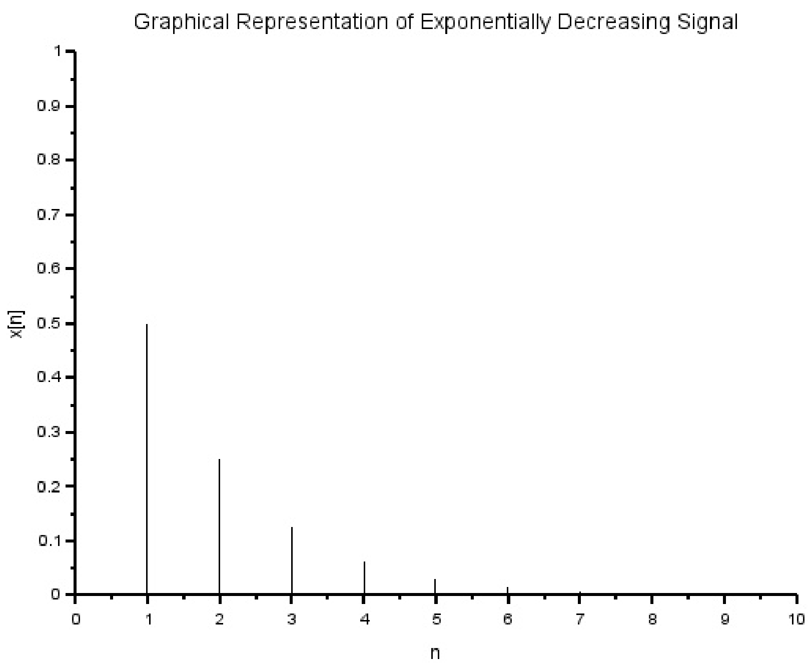

Figure 1.5: Exponential Decreasing Signal

Scilab code Solution 1.5 Exponential Decreasing Signal

```
//1.GENERATION OF ELEMENTARY DISCRETE TIME SEQUENCE
//Caption: Exponentially Decreasing Signal
clear;
clc;
close;
a =0.5;
n = 0:10;
x = (a)^n;
a=gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x)
a.thickness = 2;
xtitle('Graphical Representation of Exponentially Decreasing Signal','n','x[n]');
```

Experiment: 2

Linear and Circular convolutions

Scilab code Solution 2.1 Linear Convolution

```
//Caption:Program for Linear Convolution
clc;
clear all;
close ;
x = input('enter x seq');
h = input('enter h seq');
m = length(x);
n = length(h);
//Method 1 Using Direct Convolution Sum Formula
for i = 1:n+m-1
    conv_sum = 0;
    for j = 1:i
        if (((i-j+1) <= n)&(j <= m))
            conv_sum = conv_sum + x(j)*h(i-j+1);
        end;
        y(i) = conv_sum;
    end;
```

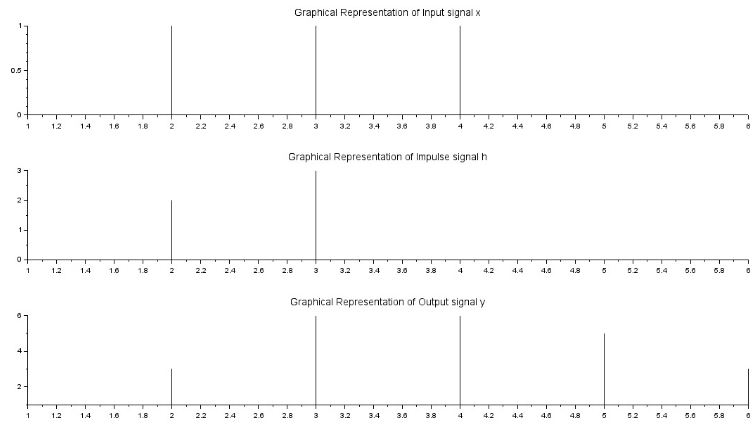

Figure 2.1: Linear Convolution

```
end;
disp(y','Convolution Sum using Direct Formula Method =')
//Method 2 Using Inbuilt Function
f = convol(x,h)
disp(f,'Convolution Sum Result using Inbuilt Funtion =')
//Method 3 Using frequency Domain multiplication
N = n+m-1;
x = [x zeros(1,N-m)];
h = [h zeros(1,N-n)];
f1 = fft(x)
f2 = fft(h)
f3 = f1.*f2;    // freq domain multiplication
f4 = ifft(f3)
disp(f4,'Convolution Sum Result DFT - IDFT method =')
//f4 =  real(f4)
subplot(3,1,1);
plot2d3('gnn',x)
xtitle('Graphical Representation of Input signal x');
subplot(3,1,2);
```

```
plot2d3('gnn',h)
xtitle('Graphical Representation of Impulse signal h
    ');
subplot(3,1,3);
plot2d3('gnn',y)
xtitle('Graphical Representation of Output signal y'
    );


//OUTPUT Test case

//enter x seq [1,1,1,1]
//enter h seq [1,2,3]

//Result
//   1.    3.    6.    6.    5.    3.
//  "Convolution Sum using Direct Formula Method ="
//    1.    3.    6.    6.    5.    3.
//  "Convolution Sum Result using Inbuilt Funtion ="
//    1.    3.    6.    6.    5.    3.
//  "Convolution Sum Result DFT - IDFT method ="
```

Scilab code Solution 2.2 Circular Convolution

```
//Caption: Program to find the Cicrcular Convolution
     of given discrete sequences
clear all;
clc;
x1 = input('Enter the first sequence x1[n]=')
x2 = input('Enter the second sequence x2[n]=')
//x1 = [1,3,5,7];
//x2 = [2,4,6,8];

//Method 1: Circular Convolution using built-in fft
    function
```

```
X1 = fft(x1,-1); //fft of x1
X2 = fft(x2,-1); //fft of x2
X3 = X1.*X2; //X1(k)*X2(K)
x3 = fft(X3,1); //ifft of X3(K)
disp('Circular Convolution using built-in fft
    function:')
disp(x3)



//Method 2: Circular Convolution of two sequences
    using direct formula
m = length(x1)
n = length(x2)
a = zeros(1,max(m,n))
if (m >n)
   for i = n+1:m
     x2(i) = 0;
   end
elseif (n>m)
  for i = m+1:n
    x1(i) = 0;
  end
end
N = length(x1)
x3 = zeros(1,N);
a(1) = x2(1);
for j = 2:N
  a(j) = x2(N-j+2);
end
for i =1:N
  x3(1) = x3(1)+x1(i)*a(i);
end
for k = 2:N
 for j =2:N
   x2(j) = a(j-1);
 end
   x2(1) = a(N);
```

```
   x2
 for i = 1:N
    a(i) = x2(i);
    x3(k) = x3(k)+x1(i)*a(i);
 end
end
disp('Circular Convolution using Direct formula method:')
disp('Circular Convolution Result x3 = ')
disp(x3)

//OUTPUT Test case
//Enter the first sequence x1[n]=[1,2,3,4]
//
//Enter the second sequence x2[n]=[1,1,1,1]
//
//
//  "Circular Convolution using built-in fft function:"
//
//    10.    10.    10.    10.
//
//  "Circular Convolution using Direct formula method:"
//
//  "Circular Convolution Result x3 = "
//
//    10.    10.    10.    10.
```

Experiment: 3

Auto correlation and Cross Correlation

Scilab code Solution 3.1 Auto Correlation

```
//Caption: Progrem to find the Autocorrelation of a
   given Input Sequence
clear all;
clc;
x = input('Enter the given discrete time sequence');
L = length(x);
h = zeros(1,L);
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('Auto Correlation Result is')
disp(Rxx)


//OUTPUT Test case
//Enter the given discrete time sequence [1,2,3,4]
//Result
//"Auto Correlation Result is"
//4.    11.    20.    30.    20.    11.    4.
```

Scilab code Solution 3.2 Cross Correlation

```
//Caption: Program to find the Cross-correlation of two Sequences
clear all;
clc;
x = [1,2,1,1];
L = length(x);
h1 = [1,1,2,1];
disp('The input sequence 1 =')
disp(x)
disp('The input sequence 2=')
disp(h1)

//Cross Correlation using Built-in function xcorr()

Y  = xcorr(x,h1)
disp('Cross Correlation using built-in function
```

```
xcorr():')
disp(Y)

//Cross Correlation using Direct Formula Method
for i = 1:L
  h(L-i+1) = h1(i);
end
N = 2*L-1;
Rxy = 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)
      Rxy(n) = Rxy(n)+x(n-k+1)*h(k);
    end
  end
end

disp('Cross Correlation Result using direct formula method is')
disp(Rxy)


//OUTPUT Test Case
//Result
//"The input sequence 1 ="
//
//    1.    2.    1.    1.
//
//   "The input sequence 2="
//
//    1.    1.    2.    1.
```

```
//
//  "Cross Correlation using built-in function xcorr
    ():"
//
//    1.    4.    6.    6.    5.    2.    1.
//
//  "Cross Correlation Result using direct formula
   method is"
//
//    1.    4.    6.    6.    5.    2.    1.
```

Experiment: 4

Frequency Analysis using DFT

Scilab code Solution 4.0 Spectrum Using DFT

```
//Caption:Discrete Periodic Spectrum Plot of N-point
    Sequece
clear all;
clc;
x=[0,1,2,3];
//Computing DFT and IDFT
X=fft(x,-1);
disp('The DFT of given sequence x[n] is X(k)=')
disp(X)
Phase=atan(imag(X),real(X));
figure
subplot(2,1,1)
a=gca();
a.data_bounds=[0,0;5,6];
a.x_location='origin'
a.y_location='origin'
plot2d3('gnn',[0:length(x)-1],abs(X))
poly1=a.children(1).children(1);
poly1.thickness=2;
```

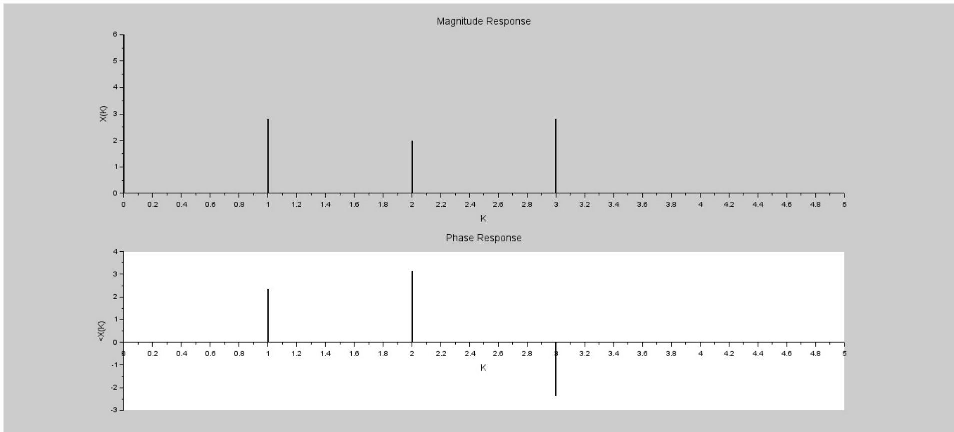

Figure 4.1: Spectrum Using DFT

```
xtitle('Magnitude Response','K','X(K)');
subplot(2,1,2)
a=gca();
a.data_bounds=[0,0;5,2];
a.x_location='origin'
a.y_location='origin'
plot2d3('gnn',[0:length(x)-1],Phase)
poly1=a.children(1).children(1);
poly1.thickness=2;
xtitle('Phase Response','K','<X(K)');
disp('The Magnitude Response is |X(k)|=')
disp(abs(X))
disp('The Phase Response is <X(K)=')
disp(Phase)


//OUTPUT Test case
//Result
//"The DFT of given sequence x[n] is X(k)="
//
//    6. + 0.i   -2. + 2.i   -2. + 0.i   -2. - 2.i
//
//   "The Magnitude Response is |X(k)|="
```

```
//
//    6.    2.8284271    2.    2.8284271
//
//  "The Phase Response is <X(K)="
//
//    0.    2.3561945    3.1415927   -2.3561945
```

Experiment: 5

Design of FIR filters (LPF/HPF/BPF/BSF) and demonstrates the filtering operation

Scilab code Solution 5.1 FIR LPF

```
//Caption: Program to Design FIR Low Pass Filter
clc;
close;
M = input('Enter the Odd Filter Length =');    //Filter length
Wc = input('Enter the Digital Cutoff frequency =');    //Digital Cutoff frequency
Tuo = (M-1)/2     //Center Value
for n = 1:M
    if (n == Tuo+1)
      hd(n) = Wc/%pi;
    else
      hd(n) =  sin(Wc*((n-1)-Tuo))/(((n-1)-Tuo)*%pi)
```

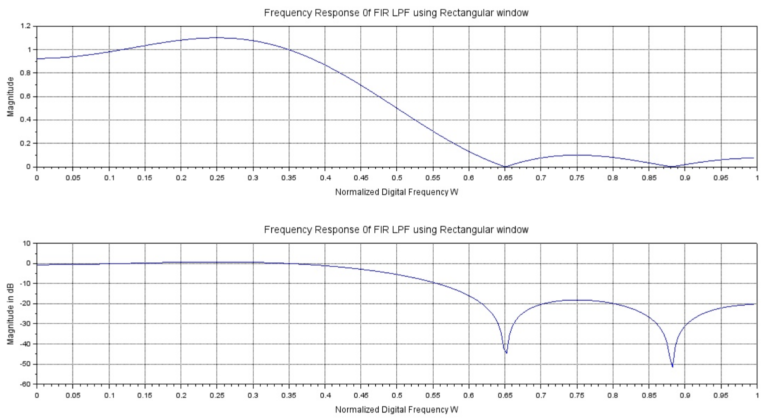

Figure 5.1: FIR LPF

```
            ;
    end
end
//Rectangular Window
for n = 1:M
  W(n) = 1;
end
//Windowing Fitler Coefficients
h = hd.*W;
disp('Filter Coefficients are')
disp(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 LPF using
   Rectangular window')
xgrid(1)
subplot(2,1,2)
plot(2*fr,hzm_dB)
```

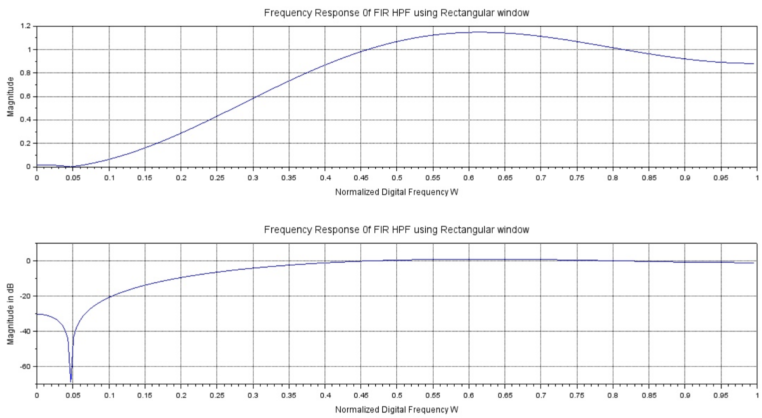

Figure 5.2: FIR HPF

```
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response 0f FIR LPF using
    Rectangular window')
xgrid(1)

//OUTPUT Test case
//Enter the Odd Filter Length = 7
//Enter the Digital Cutoff frequency = %pi/2
//Result
// "Filter Coefficients are"
//   -0.1061033
//    1.949D-17
//    0.3183099
//    0.5
//    0.3183099
//    1.949D-17
//   -0.1061033
```

Scilab code Solution 5.2 FIR HPF

```
//Caption: Program to Design FIR High Pass Filter
clear;
clc;
close;
M = input('Enter the Odd Filter Length =');
                //Filter length
Wc = input('Enter the Digital Cutoff frequency =');
    //Digital Cutoff frequency
Tuo = (M-1)/2     //Center Value
for n = 1:M
    if (n == Tuo+1)
      hd(n) = 1-Wc/%pi;
    else
      hd(n) = (sin(%pi*((n-1)-Tuo)) -sin(Wc*((n-1)-
         Tuo)))/(((n-1)-Tuo)*%pi);
    end
end
//Rectangular Window
for n = 1:M
  W(n) = 1;
end
//Windowing Fitler Coefficients
h = hd.*W;
disp('Filter Coefficients are')
disp(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 HPF using
   Rectangular window')
xgrid(1)
subplot(2,1,2)
plot(2*fr,hzm_dB)
```

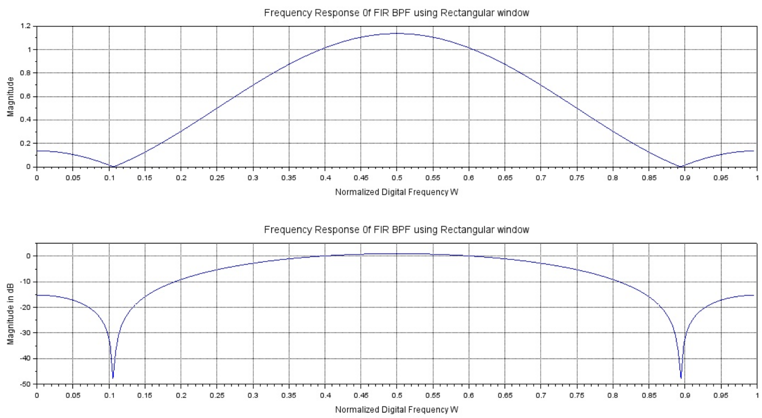

Figure 5.3: FIR BPF

```
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response 0f FIR HPF using
    Rectangular window')
xgrid(1)

//OUTPUT Test case
//Enter the Odd Filter Length = 5
//Enter the Digital Cutoff frequency = %pi/4
//"Filter Coefficients are"
//   -0.1591549
//   -0.2250791
//    0.75
//   -0.2250791
//   -0.1591549
```

Scilab code Solution 5.3 FIR BPF

```
//Caption: Program to Design FIR Band Pass Filter
```

```
clear;
clc;
close;
M = input('Enter the Odd Filter Length =');
                //Filter length
//Digital Cutoff frequency [Lower Cutoff, Upper
   Cutoff]
Wc = input('Enter the Digital Cutoff frequency =');
Wc2 = Wc(2)
Wc1 = Wc(1)
Tuo = (M-1)/2      //Center Value
hd = zeros(1,M);
W = zeros(1,M);
for n = 1:11
  if (n == Tuo+1)
     hd(n) = (Wc2-Wc1)/%pi;
  else
        n
     hd(n) = (sin(Wc2*((n-1)-Tuo)) -sin(Wc1*((n-1)-
        Tuo)))/(((n-1)-Tuo)*%pi);
  end
  if(abs(hd(n))<(0.00001))
    hd(n)=0;
  end
end
hd;
//Rectangular Window
for n = 1:M
  W(n) = 1;
end
//Windowing Fitler Coefficients
h = hd.*W;
disp('Filter Coefficients are')
disp(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')
xgrid(1)
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')
xgrid(1)


//OUTPUT Test case
//Enter the Odd Filter Length = 11
//Enter the Digital Cutoff frequency = [%pi/4,3*%pi
   /4]
//Result
//   "Filter Coefficients are"
// 0.    0.    0.   -0.3183099    0.    0.5    0.
   -0.3183099    0.    0.    0.
```

Scilab code Solution 5.4 FIR BSF

```
//Caption: Program to Design FIR Band Reject Filter
clear ;
clc;
close;
M = input('Enter the Odd Filter Length =');
                //Filter length
//Digital Cutoff frequency [Lower Cutoff, Upper
   Cutoff]
```

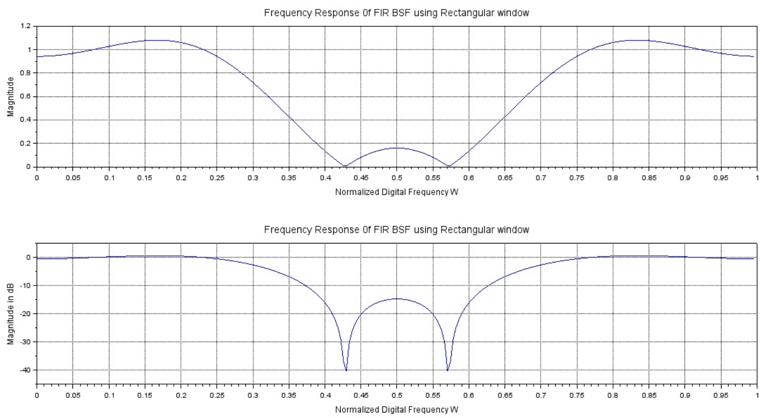

Figure 5.4: FIR BSF

```
Wc = input('Enter the 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:M
 if (n == Tuo+1)
 hd(n) = 1-((Wc2-Wc1)/%pi);
 else
 hd(n)=(sin(%pi*((n-1)-Tuo))-sin(Wc2*((n-1)-Tuo))+
    sin(Wc1*((n-1)-Tuo)))/(((n-1)-Tuo)*%pi);
 end
  if(abs(hd(n))<(0.00001))
    hd(n)=0;
  end
end

//Rectangular Window
for n = 1:M
  W(n) = 1;
end
//Windowing Fitler Coefficients
```

```
h = hd.*W;
disp('Filter Coefficients are')
disp(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 BSF using Rectangular window')
xgrid(1)
subplot(2,1,2)
plot(2*fr,hzm_dB)
xlabel('Normalized Digital Frequency W');
ylabel('Magnitude in dB');
title('Frequency Response 0f FIR BSF using Rectangular window')
xgrid(1)

//OUTPUT Test case
//Enter the Odd Filter Length = 11
//Enter the Digital Cutoff frequency = [%pi/3,2*%pi/3]
//Result
//  "Filter Coefficients are"
// 0.   -0.1378322    0.    0.2756644    0.    0.6666667    0.    0.2756644    0.   -0.1378322    0.
```

Experiment: 6

Design of Butterworth and Chebyshev IIR filters (LPF/HPF/BPF/BSF) and demonstrate the filtering operations

Scilab code Solution 6.1 Butterworth IIR LPF

```
//IIR Butterworth LPF Filter
clear;
clc;
n=3;
ftype='lp';
fdesign = 'butt';
frq=[0.15,0.25];
delta=[0.08,0.02]
hz=iir(n,ftype,fdesign,frq);
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm')
```

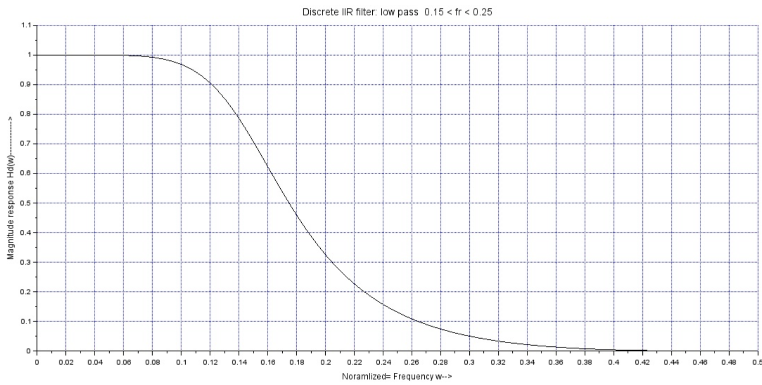

Figure 6.1: Butterworth IIR LPF

```
xtitle('Discrete IIR filter: low pass  0.15 < fr <
    0.25 ',' ',' ');
xlabel('Noramlized= Frequency w-->')
ylabel('Magnitude response Hd(w)---------------->')
xgrid(2)
```

Scilab code Solution 6.2 Butterworth IIR HPF

```
//IIR Butterworth HPF Filter

clear;
clc;
n=3;
ftype='hp';
fdesign = 'butt';
frq=[0.15,0.25];
delta=[0.08,0.02]

hz=iir(n,ftype,fdesign,frq);
```

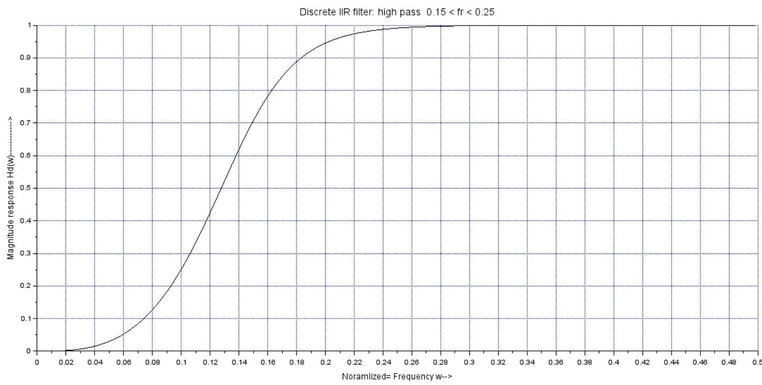

Figure 6.2: Butterworth IIR HPF

```
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm')
xtitle('Discrete IIR filter: high pass  0.15 < fr <
    0.25 ',' ',' ');
xlabel('Noramlized= Frequency w-->')
ylabel('Magnitude response Hd(w)------------>')
xgrid(2)
```

Scilab code Solution 6.3 Butterworth IIR BPF

```
//IIR Butterworth BPF Filter
clear all;
clc;
n=3;
ftype='bp';
fdesign = 'butt';
frq=[0.15,0.25];
delta=[0.08,0.02]
hz=iir(n,ftype,fdesign,frq);
```

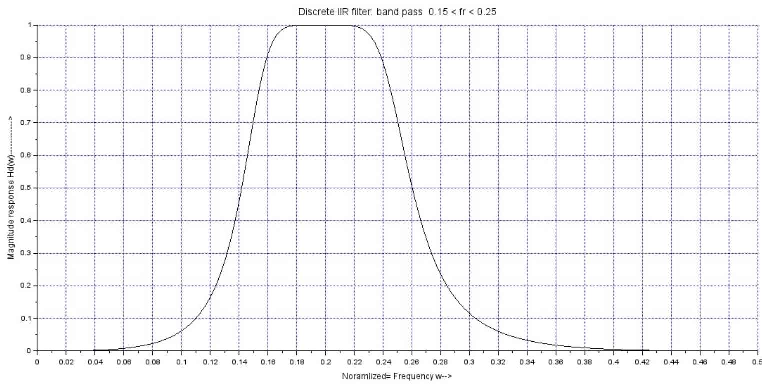

Figure 6.3: Butterworth IIR BPF

```
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm')
xtitle('Discrete IIR filter: band pass  0.15 < fr <
    0.25 ',' ',' ');
xlabel('Noramlized= Frequency w-->')
ylabel('Magnitude response Hd(w)------------->')
xgrid(2)
```

Scilab code Solution 6.4 Butterworth IIR BSF

```
//IIR Butterworth BSF Filter
clear;
clc;
n=3;
ftype='sb';
fdesign = 'butt';
frq=[0.15,0.25];
delta=[0.08,0.02]
hz=iir(n,ftype,fdesign,frq);
```

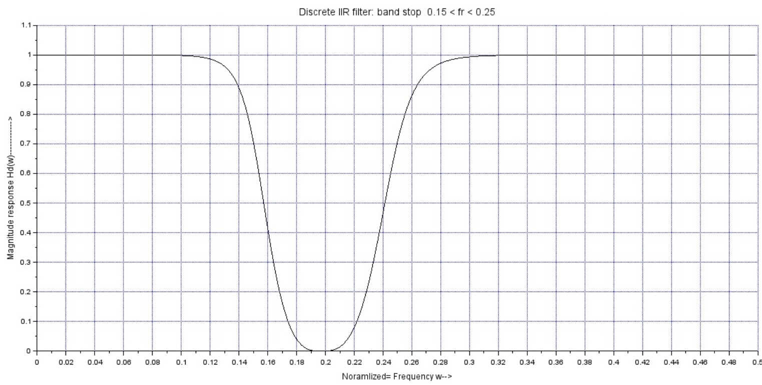

Figure 6.4: Butterworth IIR BSF

```
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm')
xtitle('Discrete IIR filter: band stop  0.15 < fr <
   0.25 ',' ',' ');
xlabel('Noramlized= Frequency w-->')
ylabel('Magnitude response Hd(w)---------------->')
xgrid(2)
```

Scilab code Solution 6.5 Chebyshev IIR LPF

```
//IIR Chebyshev LPF Filter
clear;
clc;
n=3;
ftype='lp';
fdesign = 'cheb1';
frq=[0.15,0.25];
delta=[0.08,0.02]
hz=iir(n,ftype,fdesign,frq);
```

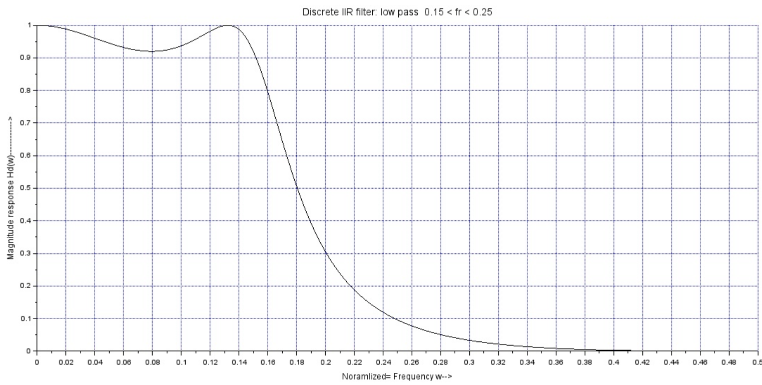

Figure 6.5: Chebyshev IIR LPF

```
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm')
xtitle('Discrete IIR filter: low pass  0.15 < fr <
   0.25 ',' ',' ');
xlabel('Noramlized= Frequency w-->')
ylabel('Magnitude response Hd(w)--------->')
xgrid(2)
```

Scilab code Solution 6.6 Chebyshev IIR HPF

```
//IIR Chebyshev HPF Filter

clear;
clc;
n=3;
ftype='hp';
fdesign = 'cheb1';
frq=[0.15,0.25];
delta=[0.08,0.02]
```

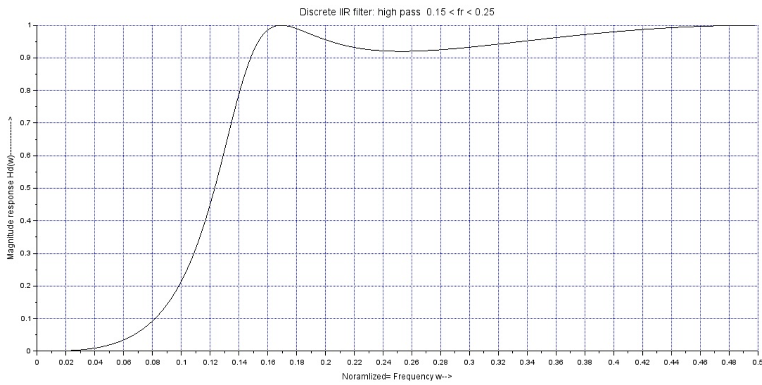

Figure 6.6: Chebyshev IIR HPF

```

hz=iir(n,ftype,fdesign,frq);
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm')
xtitle('Discrete IIR filter: high pass  0.15 < fr <
   0.25 ',' ',' ');
xlabel('Noramlized= Frequency w—>')
ylabel('Magnitude response Hd(w)—————————>')
xgrid(2)
```

Scilab code Solution 6.7 Chebyshev IIR BPF

```
//IIR Chebyshev BPF Filter
clear all;
clc;
n=3;
ftype='bp';
fdesign = 'cheb1';
frq=[0.15,0.25];
```

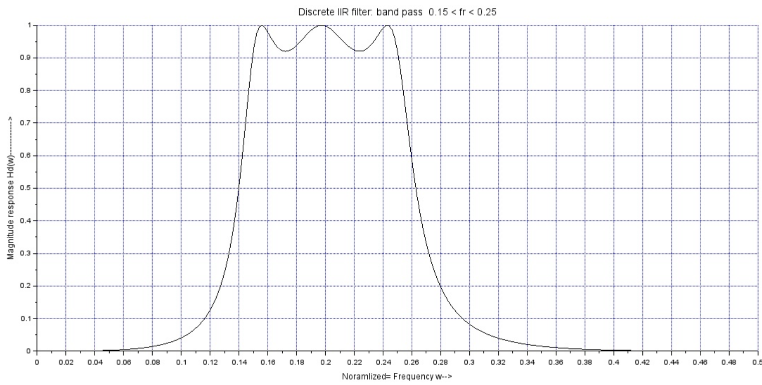

Figure 6.7: Chebyshev IIR BPF

```
delta=[0.08,0.02]
hz=iir(n,ftype,fdesign,frq);
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm')
xtitle('Discrete IIR filter: band pass  0.15 < fr <
    0.25 ',' ',' ');
xlabel('Noramlized= Frequency w-->')
ylabel('Magnitude response Hd(w)--------------->')
xgrid(2)
```

Scilab code Solution 6.8 Chebychev IIR BSF

```
//IIR Chebyshev BSF Filter
clear;
clc;
n=3;
ftype='sb';
fdesign = 'cheb1';
frq=[0.15,0.25];
```

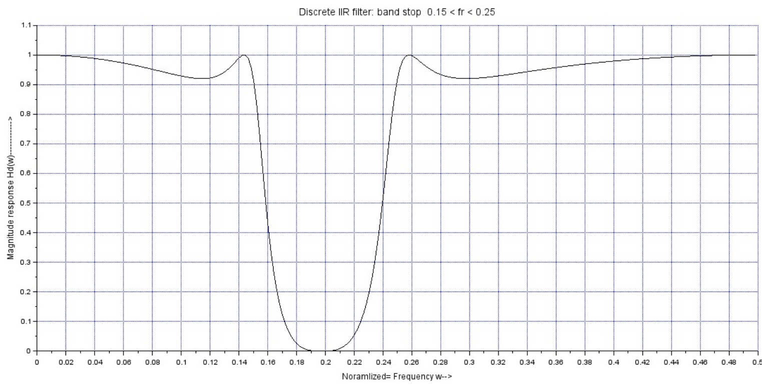

Figure 6.8: Chebychev IIR BSF

```
delta=[0.08,0.02]
hz=iir(n,ftype,fdesign,frq);
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm')
xtitle('Discrete IIR filter: band stop  0.15 < fr < 0.25 ',' ',' ');
xlabel('Noramlized= Frequency w-->')
ylabel('Magnitude response Hd(w)--------------->')
xgrid(2)
```