

Scilab Manual for  
Digital Signal Processing Lab  
by Dr R Kumaraswamy  
Electronics Engineering  
Siddaganga Institute Of Technology<sup>1</sup>

Solutions provided by  
Dr R Kumaraswamy  
Electronics Engineering  
Siddaganga Institute Of Technology

June 15, 2026

<sup>1</sup>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 Scilab Solutions	3
1 Discrete-time signals	6
2 Verification of Sampling Theorem	11
3 Impulse response of the LTI system	14
4 Frequency response of the LTI system	16
5 Linear and Circular convolution	18
6 Spectral analysis using DFT	25
7 FIR filter design	30
8 Design of Hilbert transformer using FIR filter	35
9 Design of digital differentiator using FIR filter	38
10 Design of IIR filter	41
11 Application of IIR filter	46
12 Design of Notch filter	51
13 Design of Resonator	54

# List of Experiments

Solution 1.1	Representation of discrete time signals . . . . .	6
Solution 2.1	To verify Sampling theorem in Time domain . . . .	11
Solution 3.1	To determine the impulse response of a system given a difference equation . . . . .	14
Solution 4.1	To plot the frequency response of a Digital system	16
Solution 5.1	To determine linear convolution . . . . .	18
Solution 5.2	Circular convolution in time domain and using DFT relations . . . . .	20
Solution 6.1	To demonstrate spectral leakage . . . . .	25
Solution 6.2	To demonstrate effects of zeropadding and zero in- sertion on the spectrum . . . . .	27
Solution 7.1	Design of FIR filter using Windowing method . . .	30
Solution 8.1	Design of a digital Hilbert Transformer using FIR filter . . . . .	35
Solution 9.1	Design of Digital Differentiator using a FIR filter	38
Solution 10.1	Design of digital Butterworth lowpass filter . . . .	41
Solution 10.2	Design of Digital Chebyshev lowpass filter . . . .	43
Solution 11.1	To design a digital IIR Butterworth filter to sup- press noise . . . . .	46
Solution 12.1	Suppression of noise at a given frequency using Notch filter . . . . .	51
Solution 13.1	Design of a Notch filter to filter noise at a given frequency . . . . .	54

# List of Figures

1.1	Representation of discrete time signals . . . . .	9
1.2	Representation of discrete time signals . . . . .	10
2.1	To verify Sampling theorem in Time domain . . . . .	13
3.1	To determine the impulse response of a system given a difference equation . . . . .	15
4.1	To plot the frequency response of a Digital system . . . . .	17
5.1	To determine linear convolution . . . . .	20
6.1	To demonstrate spectral leakage . . . . .	27
6.2	To demonstrate effects of zeropadding and zero insertion on the spectrum . . . . .	29
6.3	To demonstrate effects of zeropadding and zero insertion on the spectrum . . . . .	29
7.1	Design of FIR filter using Windowing method . . . . .	33
7.2	Design of FIR filter using Windowing method . . . . .	34
8.1	Design of a digital Hilbert Transformer using FIR filter . . . . .	37
9.1	Design of Digital Differentiator using a FIR filter . . . . .	40
10.1	Design of digital Butterworth lowpass filter . . . . .	43
10.2	Design of Digital Chebyshev lowpass filter . . . . .	45
11.1	To design a digital IIR Butterworth filter to suppress noise . . . . .	49
11.2	To design a digital IIR Butterworth filter to suppress noise . . . . .	50
12.1	Suppression of noise at a given frequency using Notch filter . . . . .	53

13.1 Design of a Notch filter to filter noise at a given frequency . 55

# Experiment: 1

## Discrete-time signals

Scilab code Solution 1.1 Representation of discrete time signals

```
1
2 //scilab 5.5.2 ,OS: Ubuntu 14.04
3 //Generation of signals
4
5 //Unit Sample Sequence
6 clear ;clc ;close ;
7 L = 4; //length= 2*L+1
8 n = -L:L; // Time index
   vector
9 x = [zeros(1,L),1,zeros(1,L) ] ;
10 figure (1);
11 subplot(421),plot2d3(n,x),xlabel('Unit Sample
   sequence', 'n', 'x_1[n]');
12
13 //Unit step function
14 //clear ;clc ;close ;
15 n1=0:5
16 x1=[ones(1,6)];
17 subplot(422),plot2d3(n1,x1),xlabel('Unit Step
   sequence', 'n', 'x_2[n]')
18 //figure(1); plot2d3(n,x);
```

```

19 //xtitle('Discrete Unit Step Sequence','n','x[n]') ;
20
21 //Unit ramp function
22 //clear ;clc ;close ;
23 L = 4; // Length of the
    sequence
24 n2= -L : L;
25 x2= [zeros(1,L ),0:L ];
26 ,subplot(423),plot2d3(n2,x2),xtitle('Unit Ramp
    sequence','n','x_2[n]')
27 //plot2d3(n,x);
28 //xtitle(' Discrete Unit Ramp Sequence','n','x[n]')
    ;
29
30 //Discrete time Exponential signal
31 //clear ;clc ;close ;
32 a =0.5; //For decreasing a<1 and For increasing
    exponential a>1
33 n3 = 0:10;
34 x3 = (a).^n3 ;
35 subplot(424),plot2d3(n3,x3),xtitle('Exponential
    Sequence','n','x_3[n]')
36 //plot2d3(n,x);xtitle('Exponentially Decreasing
    Signal ','n','x[n]');
37
38
39
40 //Sinusoidal signal
41 //clc;clear;
42 fm=100;// Frequency 100 Hz or input('Enter the input
    signal frequency:'); //100
43 k=3;// Number of cycles:3 or input('Enter the number
    of Cycles of input signal:'); //3
44 A=1; // Unit amplitude or input('Enter the amplitude
    of input signal:'); //5
45 tm=0:1/(fm*k):k/fm;
46 x4=A*cos(2*%pi*fm*tm);
47 subplot(425),plot2d3(tm,x4),xtitle('Sinusoidal

```

```

        Signal ', 'n', 'x_4[n]')
48 //figure(1);plot2d3(tm,x);
49 //title('Graphical Representation of Sinusoidal
    Signal');
50 //xlabel('Time');ylabel('Amplitude');
51 //xgrid(1)
52
53 //Square wave
54 //clc;clear;
55 t=(0:0.1:4*%pi)';
56 x5=4*%pi*squarewave(t);
57 subplot(426),plot2d3(t,x5),xtitle('Square wave', 'n',
    'x_5[n]')
58
59
60 //Triangular wave
61 //clear;clc;
62 A=5//input('enter the amplitude:'); //5
63 K= 2//input('enter number of cycles:'); //2
64 x6 = [0:A A-1:-1:1];
65 x7=x6;
66 for i=1:K-1
67 x7=[x7 x6];
68 end
69 n7=0:length(x7)-1; // Index of the sequence
70 subplot(427),plot2d3(n7,x7);xtitle('Triangular wave'
    , 'time', 'amplitude');
71
72 //Sawtooth wave
73 //clc;clear;
74 A=5//input('enter the amplitude:'); //5
75 K=2; //input('enter number of cycles:'); //2
76 x8 = [0:A];
77 x9=x8;
78 for i=1:K-1
79 x9=[x9 x8];
80 end
81 n9=0:length(x9)-1;

```

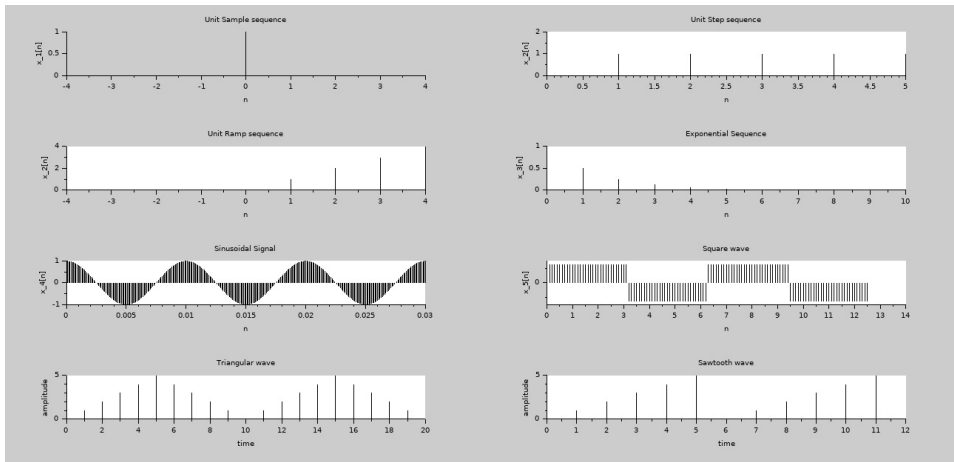


Figure 1.1: Representation of discrete time signals

```

82 subplot(428), plot2d3(n9, x9); xtitle('Sawtooth wave', '
    time', 'amplitude');
83
84 // Complex valued signals
85 clc; clear;
86 n= [-10:1:10];
87 a=-0.1+0.3*%i;
88 x=exp(a*n);
89 figure(2);
90 subplot(221), plot2d3(n, real(x)); xtitle('Complex
    valued signal', 'n', 'Real part');
91 subplot(223), plot2d3(n, imag(x)); xtitle('Imaginary',
    'n');
92 subplot(222), plot2d3(n, abs(x)); xtitle('Magnitude
    part', 'n');
93 theta=(180/%pi)*atan(imag(x), real(x));
94 subplot(224), plot2d3(n, theta); xtitle('Phase part', '
    n');

```

---

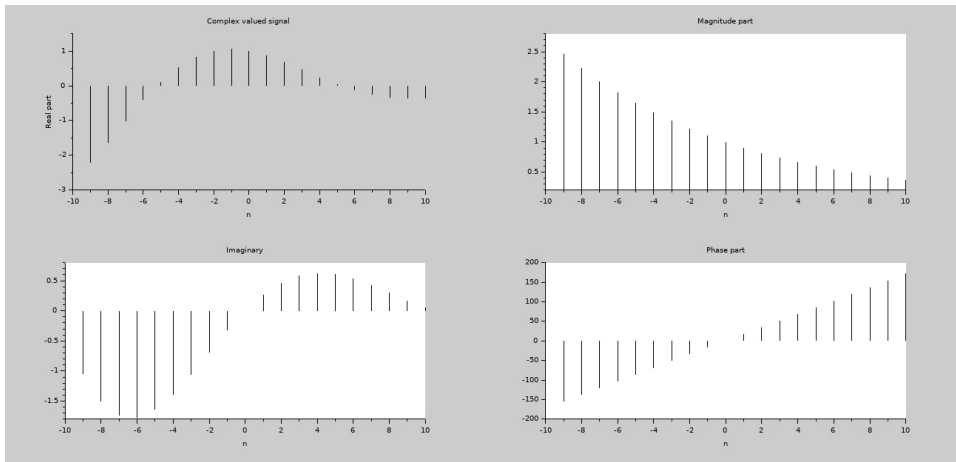


Figure 1.2: Representation of discrete time signals

## Experiment: 2

# Verification of Sampling Theorem

Scilab code Solution 2.1 To verify Sampling theorem in Time domain

```
1
2 //scilab 5.5.2 , OS: Ubuntu 14.04
3 //Sampling
4 clc; clear;
5 fm=100; //input('Enter the input signal frequency:')
   ; //100
6 k=4; //input('Enter the number of Cycles of input
   signal:'); //2
7 A=1; //input('Enter the amplitude of input signal:');
   //3
8 tm=0:1/(fm*fm):k/fm;
9 x=A*cos(2*%pi*fm*tm);
10 figure(1);
11 subplot(411), plot(tm,x);
12 title('ORIGINAL SIGNAL'); xlabel('Time'); ylabel('
   Amplitude');
13 xgrid(1)
14
15 //Sampling Rate(Nyquist Rate)=2*fm
```

```

16 fnyq=2*fm;
17
18 // UNDER SAMPLING
19 fs=(3/4)*fnyq;
20 n=0:1/fs:k/fm;
21 xn=A*cos(2*%pi*fm*n);
22 //figure(2);
23 subplot(412),plot2d3('gmn',n,xn);
24 plot(n,xn,'r');
25 title('Under Sampling');
26 xlabel('Time');
27 ylabel('Amplitude');
28 legend('Sampled Signal', 'Reconstructed Signal');
29 xgrid(1)
30 //NYQUIST SAMPLING
31 fs=fnyq;
32 n=0:1/fs:k/fm;
33 xn=A*cos(2*%pi*fm*n);
34 //figure(3);
35 subplot(413),
36 plot2d3('gmn',n,xn);
37 plot(n,xn,'r');
38 title('Nyquist Sampling');
39 xlabel('Time');
40 ylabel('Amplitude');
41 legend('Sampled Signal', 'Reconstructed Signal');
42 xgrid(1)
43 //OVER SAMPLING
44 fs=fnyq*10;
45 n=0:1/fs:k/fm;
46 xn=A*cos(2*%pi*fm*n);
47 //figure(4);
48 subplot(414)
49 plot2d3('gmn',n,xn);
50 plot(n,xn,'r');
51 title('Over Sampling');
52 xlabel('Time');
53 ylabel('Amplitude');

```

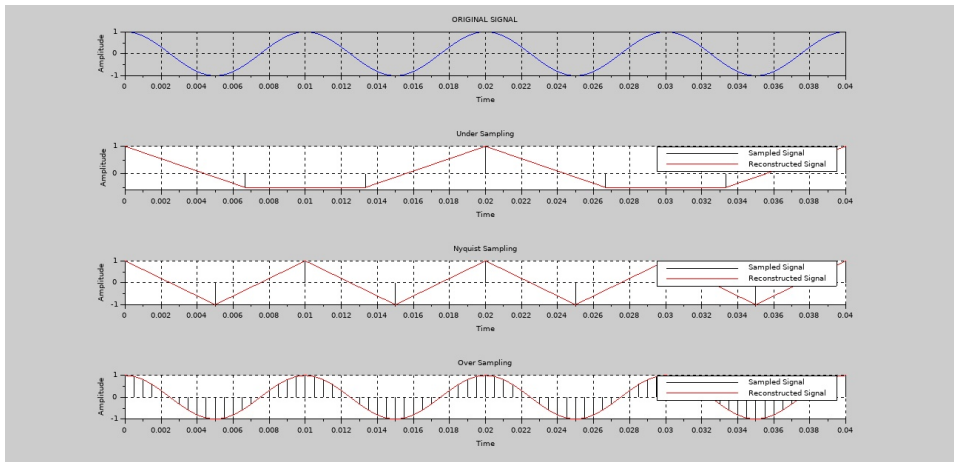


Figure 2.1: To verify Sampling theorem in Time domain

```

54 legend('Sampled Signal', 'Reconstructed Signal');
55 xgrid(1)
56 //Result
57 // Observing plots

```

---

## Experiment: 3

# Impulse response of the LTI system

**Scilab code Solution 3.1** To determine the impulse response of a system given a difference equation

```
1 //scilab 5.5.2 , OS: Ubuntu 14.04
2 //To determine the impulse response of a LTI system ,
   given the difference equation  $y[n]=b_2 x[n-2]+b_1$ 
    $x[n-1]+ b_0x[n] +a(1)y[n-1]$ 
3 clear all;clc;close;
4 b=input('Enter the coefficients of input x[n]= ');//
   [1]
5 a=input('Enter the coefficients of output y[n]= ');
   //[1 -1 0.9]
6 x=[1 zeros(1,9)];//generate impulse sequence of
   length 10
7 n=0:9;
8 h=filter(b,a,x);
9 figure; plot2d3(n,h),
10 xtitle('Impulse response h[n]','Time index n', 'h[n]
   ',' ');
11 //Example:  $y[n]-y[n-1]+0.9y[n-2]=x[n]$ ; a=[1] b=[1 -1
   0.9]
```

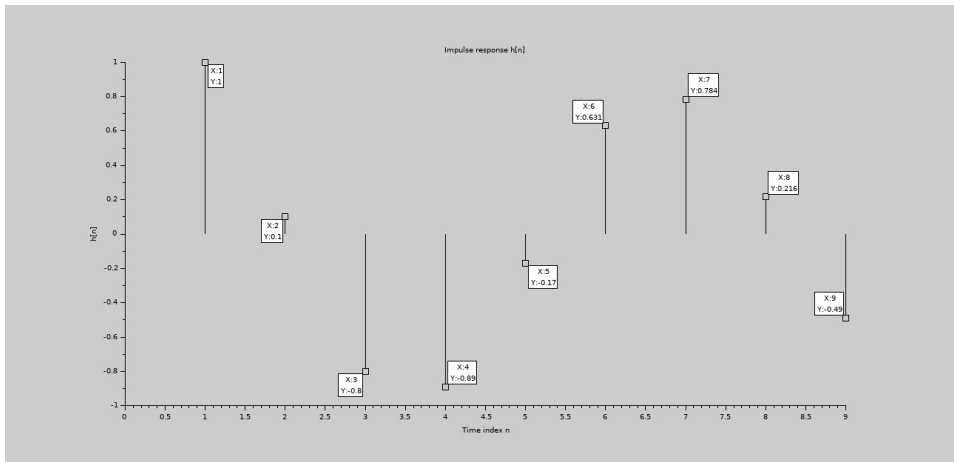


Figure 3.1: To determine the impulse response of a system given a difference equation

- ```

12 //n determines the length of the impulse response
    required
13 //Result:10 samples of h[n
    ]=[1,1,0.1,-0.8,-0.89,-0.17,0.631,0.784,
    0.216,-0.4895]

```
-

## Experiment: 4

# Frequency response of the LTI system

**Scilab code Solution 4.1** To plot the frequency response of a Digital system

```
1 //scilab 5.5.2 , OS: Ubuntu 14.04
2 //To determine the frequency response of a discrete-
   time system from its difference equation
3
4 //Design steps: Given  $y[n] = -a_2 y[n-2] - a_1 y[n-1] + b_0 x[n] + b_1 x[n-1] + b_2 x[n-2]$ 
5 //1. System function  $H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$ 
6 //2. Put  $z = e^{j\omega}$  to get the frequency response
7 //Design example: Plot the magnitude and phase
   response of the system represented by
8 // $6y[n] + 5y[n-1] + y[n-2] = 18x[n] + 8x[n-1]$ 
9
10
11 clear;clc;
12 close;
13 b=input('Enter the coefficients of x[n] ');//[1 -1]
14 a=input('Enter the coefficients of y[n] ');//[1 -0.5]
```

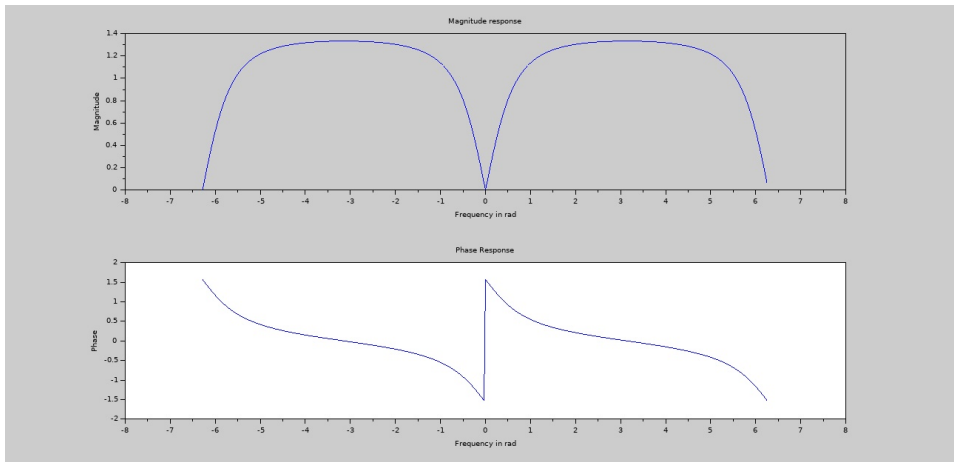


Figure 4.1: To plot the frequency response of a Digital system

```

15 //b=[18, 8];
16 //a=[6 5 1];
17 m= 0: length(b)-1; p=0:length(a)-1;
18 w=-2*%pi:%pi/100:2*%pi;//Plot over a interval of 4pi
    to observe periodicity
19 num = b* exp(-%i*m'*w);
20 den = a*exp(-%i*p'*w);
21 H= num./den;
22 magH = abs(H); angH= atan(imag(H),real(H));
23 figure;
24 subplot(211), plot(w, magH);
25 xtitle('Magnitude response', 'Frequency in rad', '
    Magnitude');
26 subplot(212), plot(w, angH);
27 xtitle('Phase Response', 'Frequency in rad', 'Phase');
28 //Expected result
29 //H = [5,3.5802695 - 1.3881467i,2.6 - i,2.253303 -
    0.4785341i,2.1666667,2.253303 + 0.4785341i,2.6 +
    i,3.5802695 + 1.3881467i,5]

```

---

# Experiment: 5

## Linear and Circular convolution

Scilab code Solution 5.1 To determine linear convolution

```
1 //scilab 5.5.2 , OS: Ubuntu 14.04
2 // Linear Convolution in time and frequency domain
3
4 clc ;clear all;close ;
5
6 x=[1 2 3 4];//input ('enter the input sequence
   values x(n)= '); // [1 2 3 4]
7 h=[1 -1 0 -1];//input('enter the impulse sequence
   values h(n) = ');..// [1 -1 0 -1]
8
9 L1 = length(x);
10 L2 = length(h);
11
12 //Method 1 Using Direct Convolution Sum Formula
13 for i = 1: L1 +L2 -1
14   conv_sum = 0;
15   for j = 1: i
16     if ((( i - j +1) <= L2 ) &( j <= L1 ) )
17       conv_sum = conv_sum + x ( j ) * h ( i -j +1) ;
18     end ;
19   y(i) = conv_sum ;
```

```

20 end ;
21 end ;
22
23 disp(y, ' Convolution Sum using Direct Formula Method
    = ' )
24
25 //Method 2 Using In built Function
26 f = convol(x,h)
27 disp(f, ' Convolution Sum Result using Inbuilt
    Function = ' )
28
29 //Method 3 Using frequency Domain multiplication
30 N = L1 +L2 -1; //
    Linear convolution output length
31 x = [ x zeros(1 ,N - L1 ) ];
32 h = [ h zeros(1 ,N - L2 ) ];
33 f1 = fft(x)
34 f2 = fft(h)
35 f3 = f1.* f2 ; //
    Multiplication in frequency domain
36 f4 = ifft(f3)
37 disp (f4 , 'Convolution Sum Result DFT and IDFT
    method = ' )
38
39 //To plot input , impulse and output signals .
40 subplot (5,1,1) ;plot2d3(x);xtitle('Input signal x '
    , 'n' , 'x[n] ');
41 subplot(5,1,2) ;plot2d3(h);xtitle('Impulse signal h'
    , 'n' , 'h[n] ');
42 subplot(5,1,3) ;plot2d3(y);xtitle('Liner Convolution
    using formula', 'n' , 'y1[n] ');
43 subplot(5,1,4) ;plot2d3(f);xtitle('Linear
    Convolution using Inbuilt function', 'n' , 'y2[n] ');
44 subplot(5,1,5) ;plot2d3(f);xtitle('Linear
    Convolution using DFT method', 'n' , 'y3[n] ');
45
46 // Expected result
47 //1.    1.    1.    0. - 6. - 3. - 4.

```

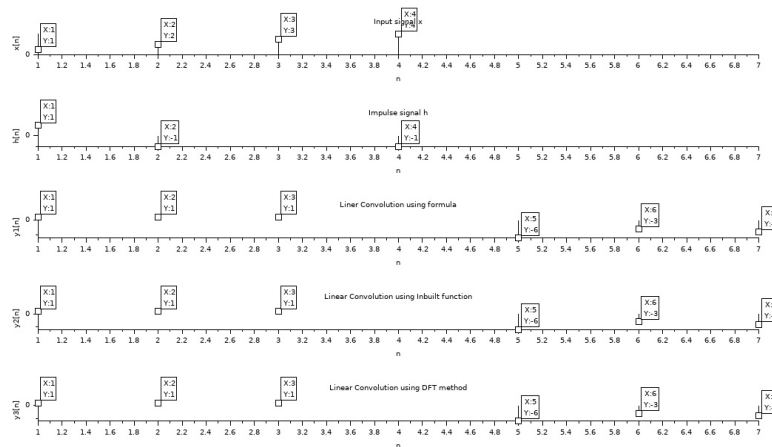


Figure 5.1: To determine linear convolution

### Scilab code Solution 5.2 Circular convolution in time domain and using DFT relations

```

1
2 //scilab 5.5.2 , OS: Ubuntu 14.04
3 // Circular convolution of given discrete sequences
  in time domain (Matrix method)
4 clear;clc;
5 x1=input('enter the first sequence values x1(n)= ');
  ; // [1 2 3 4]
6 x2=input('enter the second sequence values x2(n) = ');
  ); // [1 -1 0 -1]
7 L1 = length(x1); //length of
  first sequence
8 L2 = length(x2); //length of
  second sequence
9

```

```

10 if (L1 >L2) //To make
    length of x1 and x2 are Equal
11     for i = L2+1:L1
12         x2(i) = 0;
13     end
14 elseif (L2>L1)
15     for i = L1+1:L2
16         x1(i) = 0;
17     end
18 end
19
20 N = length(x1);
21 x3 = zeros(1,N); //x3 =
    Circular convolution result
22 a(1) = x2(1);
23 for j = 2:N
24     a(j) = x2(N-j+2);
25 end
26 for i =1:N
27     x3(1) = x3(1)+x1(i)*a(i);
28 end
29 X(1,:) =a;
30
31 //Calculation of circular convolution
32 for k = 2:N
33     for j =2:N
34         x2(j) = a(j-1);
35     end
36     x2(1) = a(N);
37     X(k,:) = x2;
38     for i = 1:N
39         a(i) = x2(i);
40         x3(k) = x3(k)+x1(i)*a(i);
41     end
42 end
43 disp(X, 'Circular Convolution Matrix x2[n]= ')
44 disp(x3, 'Circular Convolution Result x3[n] = ')
45 // Expected result

```

```

46 //Circular Convolution Matrix x2[n]=
47
48 //      1.  - 1.    0.  - 1.
49 //     - 1.    1.  - 1.    0.
50 //      0.  - 1.    1.  - 1.
51 //     - 1.    0.  - 1.    1.
52
53 //Circular Convolution Result x3[n] =
54
55 // -5.  -2.  -3.    0.
56
57 // Circular Convolution in frequency domain (DFT-
    IDFT method)
58 clear all;clc;close;
59 x1=input('enter the first sequence values x1(n)= ');
    ; // [1 2 3 4]
60 x2=input('enter the second sequence values x2(n) = ');
    // [1 -1 0 -1]
61 L=input('enter the length of the sequence values L=
    '); //4
62
63 //Computing DFT
64 X1 = fft(x1,-1); // -1 for direct
    FFT
65 X2 = fft(x2,-1);
66 disp(X1,'DFT of x1[n] is X1(k)=')
67 disp(X2,'DFT of x2[n] is X2(k)=')
68
69 //Multiplication of 2 DFTs
70 X3 = X1.*X2;
71 disp(X3,'DFT of x3[n] is X3(k)=')
72 x3 =(fft(X3,1)) //
    Circular Convolution Result ,1 for IFFT
73 disp(x3,'Circular Convolution x3[n]=')
74 //// Expected result
75 //DFT of x1[n] is X1(k)=      10.  - 2. + 2.i  - 2.
    - 2.  - 2.i
76

```

```

77 //DFT of x1[n] is X2(k)=      - 1.      1.      3.      1.
78
79 // DFT of x3[n] is X3(k)=      - 10.      - 2. + 2.i      -
      6.      - 2.      - 2.i
80
81 //Circular Convolution x3[n]=      -5.      -2.
      -3.      0.
82
83 ////Performing Linear Convolution using Circular
      Convolution
84 clear;clc;
85 x=input ('enter the input sequence values x(n)= ');
      // [1 2 3 4]
86 h=input('enter the impulse sequence values h(n) = ')
      ; // [1 -1 0 -1]
87 N1 = length(x); //Length of input signal
88 N2 = length(h); //Length of impulse response
89
90 N = N1+N2-1 // Length of
      output response
91 disp(N,'Length of Output Response y(n)')
92
93 //Padding zeros to Make Length of 'h' and 'x' equal
      to length of output response 'y'
94
95 h1 = [h,zeros(1,N-N2)];
96 x1 = [x,zeros(1,N-N1)];
97
98 H = fft(h1,-1);
99 X = fft(x1,-1);
100 //Multiplication of 2 DFTs
101 Y = X.*H
102 y =(fft(Y,1)) //Linear Convolution Result
103
104 disp(X,'DFT of i/p X(k)=')
105 disp(H,'DFT of impulse sequence H(k)=')
106 disp(Y,'DFT of Linear Filter o/p Y(k)=')
107 disp(y,'Linear Convolution result y[n]=')

```

```

108
109 //Expected output
110 //Length of Output Response y(n)      7.
111
112 //DFT of i/p X(k)=      10. - 2.0244587 -
      6.2239817i ,      0.3460107 + 2.4791213i ,
      0.1784479 - 2.4219847i ,      0.1784479 +
      2.4219847i ,      0.3460107 - 2.4791213i , -
      2.0244587 + 6.2239817i ,
113 //DFT of impulse sequence H(k)=      - 1.
      1.2774791 + 1.2157152i , ,      0.5990311 +
      0.1930964i ,      2.1234898 + 1.4088117i ,
      2.1234898 - 1.4088117i ,      0.5990311 -
      0.1930964i ,      1.2774791 - 1.2157152i ,
114 //DFT of Linear Filter o/p Y(k)=      - 10.
      4.9803857 - 10.412171i , - 0.2714383 +
      1.5518843i ,      3.7910526 - 4.8916602i ,
      3.7910526 + 4.8916602i , - 0.2714383 -
      1.5518843i ,      4.9803857 + 10.412171i ,
115 //Linear Convolution result y[n]=      1.      1.
      1.      0.      -6.      -3.      -4.

```

---

# Experiment: 6

## Spectral analysis using DFT

Scilab code Solution 6.1 To demonstrate spectral leakage

```
1
2 //scilab 5.5.2 , OS: Ubuntu 14.04
3 // Spectral Leakage
4 //Check the result for the following cases
5 //case(1): fm=10;fs=125;m=1;m=number of cycles
6 //case(2): fm=10;fs=125;m=2;
7 //case(3): fm=200;fs=10000;m=2.5;
8 //case(4): fm=75;fs=250;m=3;
9
10 clc;clear;close;
11 //fm=input('Enter the frequency of the input signal
    ');//message frequency in Hz
12 //fs=input('Enter the sampling frequency');//
    sampling frequency in Hz
13 //m=input('Enter the number of cycles of the input
    signal');// Number of cycles
14 //Case2:No spectral leakage
15 fm=10;fs=125;m=2;//Oversampling and integer number
    of cycles
16 t=0.0001:1/fs:m/fm;
17 x=3*cos(2*%pi*fm*t); //signal
```

```

18 N=(m*fs/fm); //should be non-
    integer to obtain spectral leakage
19 for k=1:N
20   X1(k)=0;
21   for n=1:length(x)
22     X1(k)=X1(k)+x(n).*exp((-i).*2.*pi.*(n-1).*(k-1)
        ./N);
23   end
24 end
25 k=0:N-1
26 f=k*fs/N; //frequency axis in Hz
27 figure(1), subplot(221), plot2d3(t,x), xlabel('time'),
    ylabel('x(n)'), title('No leakage: m=2, f=10 and
    Fs=125 Hz'), subplot(223), plot2d3(f,abs(X1)),
    xlabel('freq in Hz'), ylabel('Mag'); //Case 3:
    Spectral leakage
28 fm=10; fs=125; m=2.5; //Oversampling and integer
    number of cycles
29 t=0.0001:1/fs:m/fm;
30 x=3*cos(2*pi*fm*t); //signal
31 N=(m*fs/fm); //should be non-
    integer to obtain spectral leakage
32 for k=1:N
33   X1(k)=0;
34   for n=1:length(x)
35     X1(k)=X1(k)+x(n).*exp((-i).*2.*pi.*(n-1).*(k-1)
        ./N);
36   end
37 end
38 k=0:N-1
39 f=k*fs/N; //frequency axis in Hz
40 figure(1), subplot(222), plot2d3(t,x), xlabel('time'),
    ylabel('x(n)'), title('Spectral leakage: m=2.5, f
    =10 and Fs=125 Hz'), subplot(224), plot2d3(f,abs(X1
    )), xlabel('freq in Hz'), ylabel('Mag')

```

---

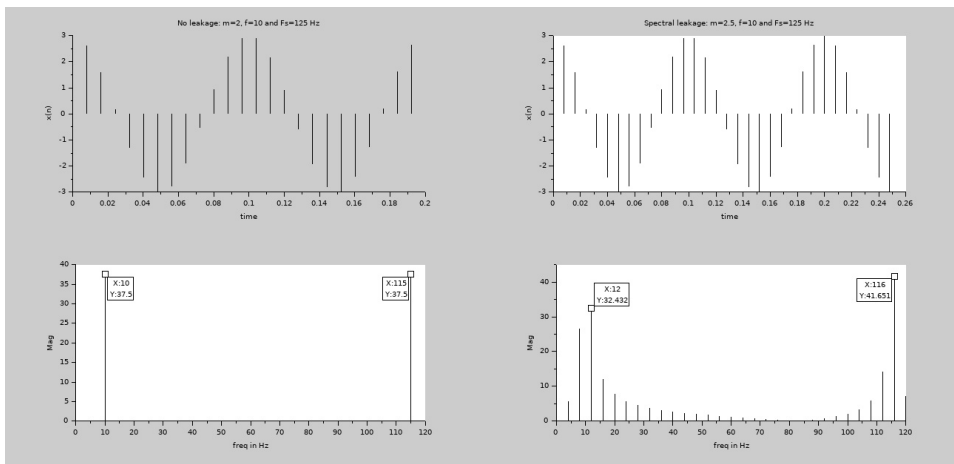


Figure 6.1: To demonstrate spectral leakage

**Scilab code Solution 6.2** To demonstrate effects of zeropadding and zero insertion on the spectrum

```

1 //scilab 5.5.2 , OS: Ubuntu 14.04
2 //Effect of zero padding and interpolation
3 // Effect of Zero padding
4 clc;clear;close;
5 x= input ('enter the input sequence values x(n)= ');
   // [1 2 3 4]
6 k= input ('enter the number of zeros to be padded= ');
   // 1020 (For 1024 point DFT)
7 N=length(x);
8 x_pad=[x zeros(1,k)];
9 N1=length(x_pad);
10 f=0:N-1;
11 f1=0:N1-1;
12 X=abs(fft(x));

```

```

13 X_pad= abs(fft(x_pad));
14 figure(1);
15 subplot(221),plot2d3(x),title(' Original sequence '),
    subplot(223),plot2d3 (f,X), title('Spectrum of
    Original sequence ');
16 subplot(222),plot2d3(x_pad),title('Zero-padded
    sequence '),subplot(224), plot2d3 (f1,X_pad),title
    ('Spectrum of Zero-padded sequence ')
17 //// Effect of inserting zeros in between samples (
    Interpolation)
18 x= input ('enter the input sequence values x(n)= ');
    //[1 2 3 4]
19 k= input ('enter the number of zeros to be inserted=
    ');//2 (Vary and observe effect of zero
    interpolation)
20 x_mod=[];
21 N=length(x);
22 //
23 for i= 1: N
24 x_mod=[x_mod, x(i), zeros(1,k)];
25 end
26 N1=length(x_mod);
27 f=0:N-1;
28 f1=0:N1-1;
29 X=abs(fft(x));
30 X_mod= abs(fft(x_mod));
31 figure(2);subplot(221),plot2d3(x),title(' Original
    sequence '),subplot(223),plot2d3 (f,X), title('
    Spectrum of Original sequence ');
32 subplot(222),plot2d3(x_mod),title('Zero-interpolated
    sequence '),subplot(224), plot2d3 (f1,X_mod),
    title('Spectrum of Zero-inserted sequence ')

```

---

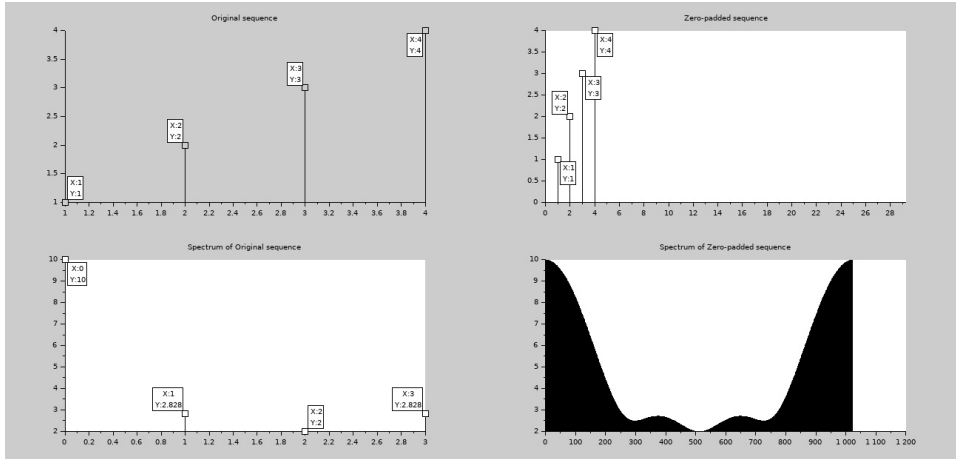


Figure 6.2: To demonstrate effects of zeropadding and zero insertion on the spectrum

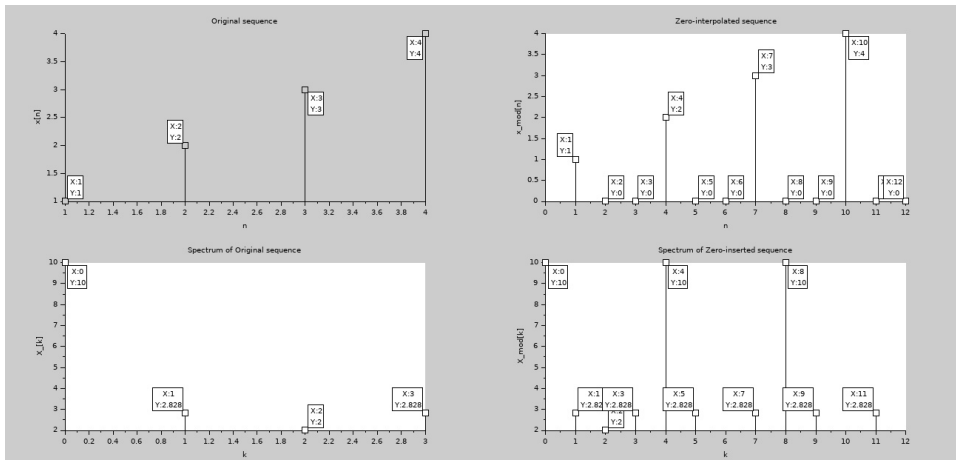


Figure 6.3: To demonstrate effects of zeropadding and zero insertion on the spectrum

# Experiment: 7

## FIR filter design

Scilab code Solution 7.1 Design of FIR filter using Windowing method

```
1
2 //scilab 5.5.2 , OS: Ubuntu 14.04
3 //Design of FIR filters using windowing
4 // Design a digital FIR low pass filter with
  following specifications.
5 //a) Pass band cut-off frequency      :wp=
  -----radians
6 //b) Pass band ripple                  :rp=-----dB
7 //c) Stop band cut-off frequency      :ws=
  -----radians
8 //d) Stop band attenuation             :rs=
  -----dB
9 //Choose an appropriate window function and
  determine impulse response and provide a plot of
  frequency response of the designed filter.
10
11 //Design example:
12 //Design a digital FIR low pass filter with
  following specifications.
13 //a) Pass band cut-off frequency      :0.3 rad
14 //b) Pass band ripple                  :0.25 dB
```

```

15 //c) Stop band cut-off frequency      :0.45   rad
16 //d) Stop band attenuation           : 50 dB
17 clc;
18 clear;
19 close;
20 wp=input('enter the pass band edge in rad');
21 ws=input('enter the stop band edge in rad');
22 rs=input('enter the stop band ripple in dB');
23 freq_points=1024;
24 freq_divs=(freq_points/2)-1;
25 k=4; //Hamming window (decided based on stop band
      attenuation)
26 trw=ws-wp;
27 N=(k*2*%pi/trw);
28 N=ceil(N);
29 remainder=N-fix(N./2).*2
30 if remainder==0
31     N=N+1;
32 end
33
34 wc=wp;
35 aph=(N-1)/2;
36 for n=0:N-1
37     if n==aph
38         hdn_minusalph(n+1)=wc/%pi;
39
40     else
41         hdn_minusalph(n+1)= sin(wc.*(n-aph))./(%pi.*(n-
      aph));
42
43 end
44 end
45 n=0:N-1;
46 wndw=window('hm',N);
47
48 hn=hdn_minusalph.*wndw';
49 figure(1); subplot(311); plot2d3(n,wndw); xlabel('n');
      ylabel('wndw'); title('Hamming Window function');

```

```

50 subplot(312); plot2d3(n,hdn_minusalph);xlabel('n');
    ylabel('hdn_minusalph');title('Impulse response
    of IIR filter');
51 subplot(313); plot2d3(n,hn);xlabel('n');ylabel('hn')
    ;title('Impulse response of FIR filter');
52 //omega=0:%pi/freq_divs:%pi;
53 h=[hn' zeros(1,freq_points-length(hn))];; //For a
    1024 point DFT
54 H=fft(h);
55 H_mag=20*log10(abs(H));
56 H_ang=atan(imag(H),real(H));
57 H_phase=unwrap(H_ang);
58 w=(0:freq_divs)./(freq_points);
59 w1=w*pi;
60 figure(2);subplot(211),plot2d(w1,H_mag(1:512));
61 xtitle('Magnitude response','w (rad)','Magnitude(dB)
    ');
62 subplot(212),plot2d(w1,H_phase(1:512));
63 xtitle('Phase Response','w (rad)','Phase (rad)');
64
65
66
67 //Problems:
68
69 //1. Design a digital FIR low pass filter with
    following specifications.
70 //a) Pass band cut-off frequency :0.4 rad
71 //b) Pass band ripple :0.25 dB
72 //c) Stop band cut-off frequency :0.6 rad
73 //d) Stop band attenuation : 44 dB
74
75 //2. Design a digital FIR low pass filter with
    following specifications.
76 //a) Pass band cut-off frequency :0.25 rad
77 //b) Pass band ripple :0.25 dB
78 //c) Stop band cut-off frequency :0.3 rad
79 //d) Stop band attenuation : 50 dB

```

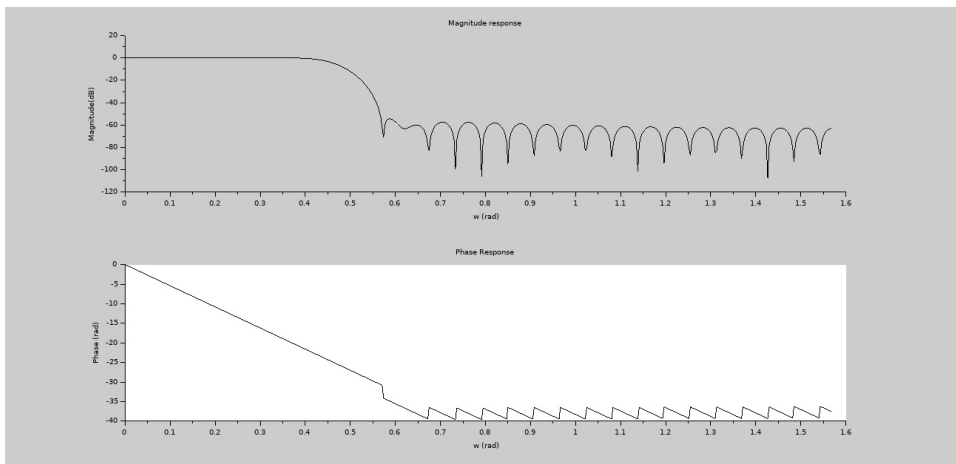


Figure 7.1: Design of FIR filter using Windowing method

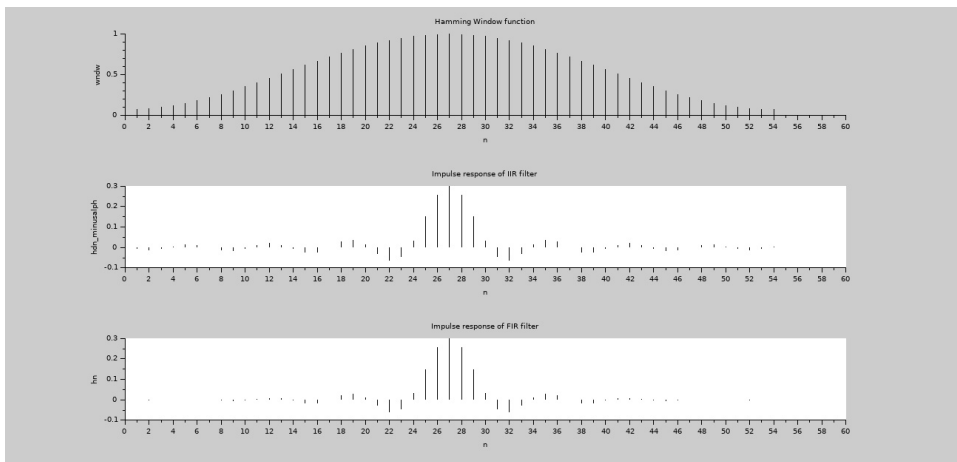


Figure 7.2: Design of FIR filter using Windowing method

## Experiment: 8

# Design of Hilbert transformer using FIR filter

Scilab code Solution 8.1 Design of a digital Hilbert Transformer using FIR filter

```
1 //scilab 5.5.2 , OS: Ubuntu 14.04
2 //Design a differentiator using a Hamming window of
  length N=21. Plot the time and frequency domain
  responses.
3 //Design a length-25 digital Hilbert transformer
  using a Hann window.
4
5 //Design of Hilbert transformer
6 //The ideal frequency response of a linear phase
  Hilbert transformer is given by
7 //Hd(ejw) = -j e(-j w ) , 0 < w < pi
8 //           j e(-j w ) , -pi < w < 0
9
10 //The ideal impulse response is given by
11
12 //hd(n- )= 2/pi (sin2 pi( n )/2) / ( n
  ),
13 //           0, n=
```

```

14
15
16 //Scilab Program
17 //Inputs: Window length and type of window
18 clc; clear; close;
19
20 N = 41; //input("enter the window length"); //55
21 freq_points=1024;
22 windowfn = window('hm',N); // Hamming window ()Window
    type can be changed here)
23 m = 0:N-1;
24 aph = (N-1)/2;
25 for n=0:N-1
26     if n==aph
27         hd(n+1)=0;
28
29     else
30         hd(n+1)=(2/%pi)*((sin((%pi/2)*(n-aph)).^2) ./ (n-
            aph));
31
32 end
33 end
34 n=0:N-1;
35 hn = hd.*windowfn';
36
37 omega=-%pi:2*%pi/(freq_points-1):%pi;
38
39 z=%z;
40 den1=real(z^(N-1));
41 num=0;
42 for n=0:N-1
43     num=num+(hn(n+1).*z^(N-n-1));
44 end
45 num1=real(num);
46 Hz=num1./den1;
47 w=exp(%i*omega);
48 rep=freq(Hz("num"),Hz("den"),w);
49 magH=abs(rep);

```

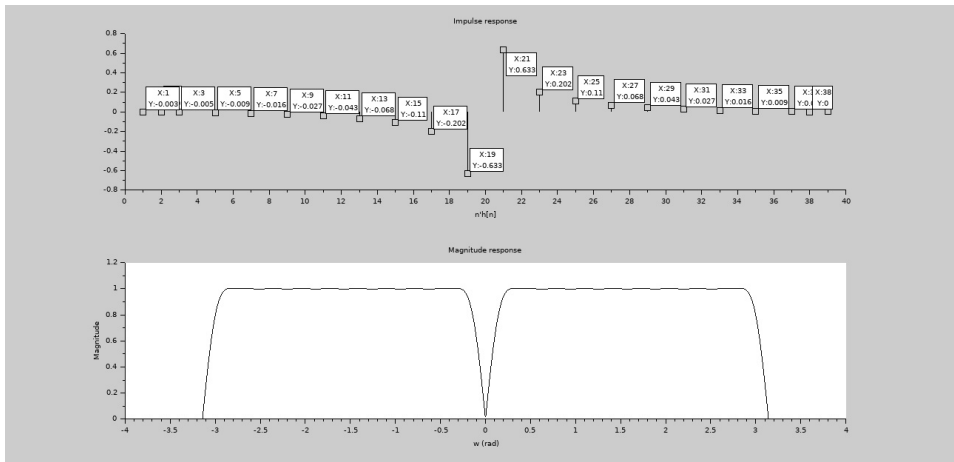


Figure 8.1: Design of a digital Hilbert Transformer using FIR filter

```

50 figure; subplot(211), plot2d3(m, hn), xtitle('Impulse
    response', 'n', 'h[n]')
51 , subplot(212), plot2d(omega, magH);
52 xtitle('Magnitude response', 'w (rad)', 'Magnitude');
53 //Expected result
54 //Magnitude response graph

```

---

## Experiment: 9

# Design of digital differentiator using FIR filter

Scilab code Solution 9.1 Design of Digital Differentiator using a FIR filter

```
1
2 11//scilab 5.5.2 , OS: Ubuntu 14.04
3 //Design a differentiator using a Hamming window of
  length N=21. Plot the time and frequency domain
  response
4 //Inputs: Window length and Type of window
5 //The frequency response of a linear-phase ideal
  differentiator is given by
6 //Hd(ejw) = j , 0 < w <
7 //          -jw, -π < w < 0
8 //The ideal impulse response of a digital
  differentiator shifted by π with linear phase is
  given by
9 //hd( n ) = cos( n ) / ( n - ), n
10 //          0, n =
11
12 //Scilab Program:
```

```

13 clc; clear; close;
14 N = 41; // input("enter the window length"); //55
15 freq_points=1024;
16 windowfn = window('hm',N); //Hamming wuindow (Try with
    different windows)
17 m = 0:N-1;
18 aph = (N-1)/2;
19 for n=0:N-1
20     if n==aph
21         hd(n+1)=0;
22
23     else
24         hd(n+1)= cos(%pi*(n-aph)) ./ (n-aph);
25
26 end
27 end
28 n=0:N-1;
29 hn = hd.*windowfn';
30
31 omega=-%pi:2*%pi/(freq_points-1):%pi;
32
33 z=%z;
34 den1=real(z^(N-1));
35 num=0;
36 for n=0:N-1
37     num=num+(hn(n+1).*z^(N-n-1));
38 end
39 num1=real(num);
40 Hz=num1./den1;
41 w=exp(%i*omega);
42 rep=freq(Hz("num"),Hz("den"),w);
43 magH=abs(rep);
44 figure; subplot(211), plot2d3(m, hn), xtitle('Impulse
    response', 'n', 'h[n]'), subplot(212), plot2d(omega,
    magH);
45 xtitle('Magnitude response', 'w (rad)', 'Magnitude');
46 //Expected result
47 //Magnitude response graph

```

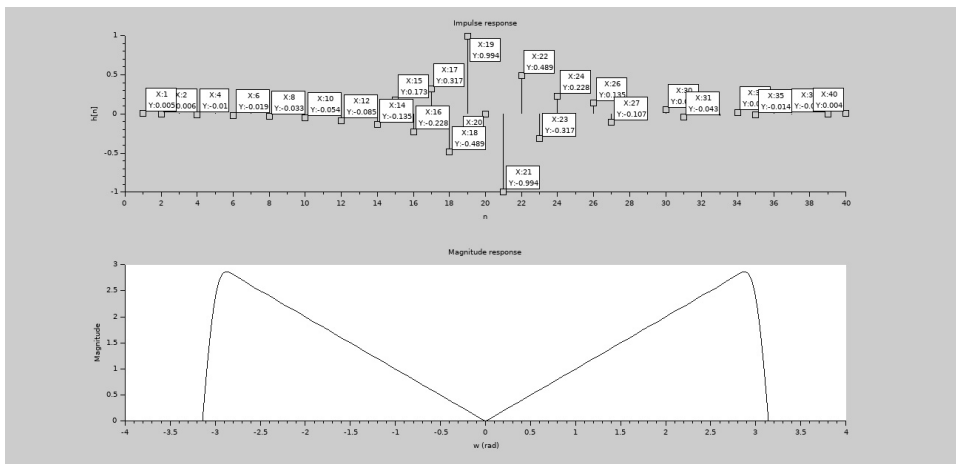


Figure 9.1: Design of Digital Differentiator using a FIR filter

# Experiment: 10

## Design of IIR filter

Scilab code Solution 10.1 Design of digital Butterworth lowpass filter

```
1 //scilab 5.5.2 , OS: Ubuntu 14.04
2 //Program To Design the Digital Butterworth IIR
  Filter
3 //Design a digital IIR low pass filter with
  following specifications.
4 //a) Pass band cut-off frequency      :1000 Hz
5 //b) Pass band ripple                  : -1 dB
6 //c) Stop band cut-off frequency      :3000 Hz
7 //d) Stop band attenuation            : -15 dB
8 //Sampling frequency: 15000 Hz
9
10 clear all;clc;close;
11 f1=1000;//input('Enter the pass band edge(Hz)= ');
12 f2=3000;//input('Enter the stop band edge(Hz)= ');
13 k1=-1;//input('Enter the pass band attenuation(dB)=
  ');
14 k2=-15;//input('Enter the stop band attenuation(dB)=
  ');
15 fs=10000;//input('Enter the sampling rate(Hz)= ');
16
17 //Digital filter specifications(rad)
```

```

18 w1=2*%pi*f1*1/fs;
19 w2=2*%pi*f2*1/fs;
20
21 //Pre warping
22 o1=2*fs*tan(w1/2)
23 o2=2*fs*tan(w2/2)
24
25 //Design of analog filter
26 n=log10(((10.^(-k1/10))-1)/((10.^(-k2/10))-1))./(2*
    log10(o1/o2));
27 n=round(n);
28 wn= o2./((10.^(-k2/10)-1).^1/(2*n));
29
30 // [h, poles , zeros , gain]=analpf(n, 'butt ', [0 0], wn)hb.
    dt = 'c';
31 // [fr , hr]=repfreq (hb, fmin , fmax)
32
33 h=buttmag(n, wn, 1:2*%pi*fs);
34 mag=20*log10(h)';
35
36
37 //Converting analog to digital filter
38 hz=iir(n, 'lp', 'butt', 0.25, [])
39 //g*poly(z, 'z')/poly(p, 'z')
40
41 [hzm, fr]=frmag(hz, 256);
42 magz=20*log10(hzm)';
43
44 subplot(2,1,1), plot2d((1:2*%pi*fs)', mag), xtitle('
    Analog IIR filter: lowpass', 'Analog frequency in
    rads/sec', 'dB', ' '); subplot(2,1,2), plot2d(fr,
    magz); xtitle('Digital IIR filter: lowpass 0 < fr
    < 0.5', 'frequency', 'dB', ' ');
45
46 //note: Use zoom/axis commands to verify the design.

```

---

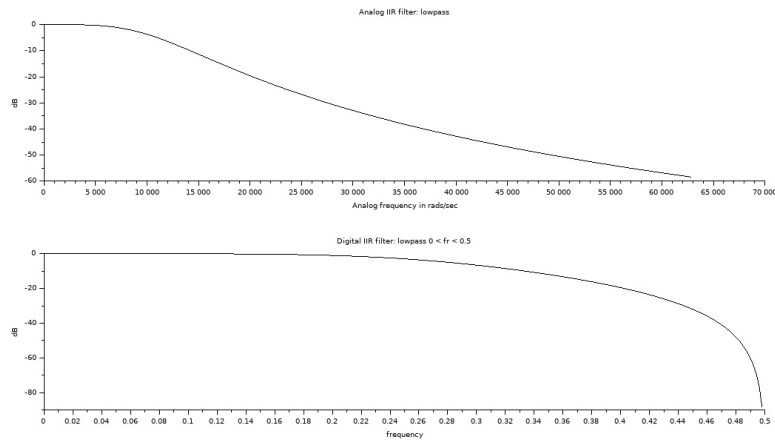


Figure 10.1: Design of digital Butterworth lowpass filter

### Scilab code Solution 10.2 Design of Digital Chebyshev lowpass filter

```

1 //scilab 5.5.2 , OS: Ubuntu 14.04
2 //Program To Design the Digital Chebyshev IIR Filter
3 ////Design example:
4 //Design a digital IIR low pass filter with
   following specifications.
5 //a) Pass band cut-off frequency      :1000 Hz
6 //b) Pass band ripple                  : -1 dB
7 //c) Stop band cut-off frequency      :3000 rad
8 //d) Stop band attenuation            : -15 dB
9 //Sampling frequency: 15000 Hz
10
11 clear all;clc;close;
12 f1=1000;//input('Enter the pass band edge(Hz)= ');
13 f2=3000;//input('Enter the stop band edge(Hz)= ');
14 rp=-1;//input('Enter the pass band ripple(dB)= ');

```

```

15 rs=-15; //input('Enter the stop band attenuation (dB)=
    ');
16 fs=10000; //input('Enter the sampling rate(Hz)= ');
17 //Digital filter specifications(rad)
18 w1=2*pi*f1*1/fs
19 w2=2*pi*f2*1/fs
20 //Pre warping
21 o1=2*fs*tan(w1/2)
22 o2=2*fs*tan(w2/2)
23 or=o2/o1; //Stop-band edge of normalized lowpass
    filter
24 A2 =10.^(-rs/10);
25 A=sqrt(A2);
26 epsilon2 = (10.^(-rp/10)-1);
27 epsilon=sqrt(epsilon2)
28 g=((A2-1).^0.5./epsilon)
29
30 N = (acosh(g))/(acosh(or))
31 N = ceil(N)
32 oc=o1;
33 // [pols ,gn] = zpch1(N,epsilon ,o1)
34 //Hs = poly(gn,'s','coeff')/real(poly(pols,'s'))
35 h=cheb1mag(N,oc,epsilon,1:2*pi*fs);
36 mag=20*log10(h)';
37 //plot2d((1:1000)',mag,[2],"011"," ",[ymax,ymin,fmax
    ,fmin])
38 //gain=20*log10(abs(h_s)); %Verify the specification
    [k1,k2] at prewarped frequencies
39 //subplot(211);
40 //plot(omega,gain);
41 //xlabel('frequency in rad/sec ');
42 //Converting analog to digital filter
43 fc=w1/(2*pi);
44 delta1=(1-(1./A2));
45 //1-ripple in passband
46 hz=iir(N,'lp','cheb1',[fc],[delta1 0]);
47 //for cheb1 filters 1-delta(1)<ripple<1 in passband
48 //g*poly(z,'z')/poly(p,'z')

```

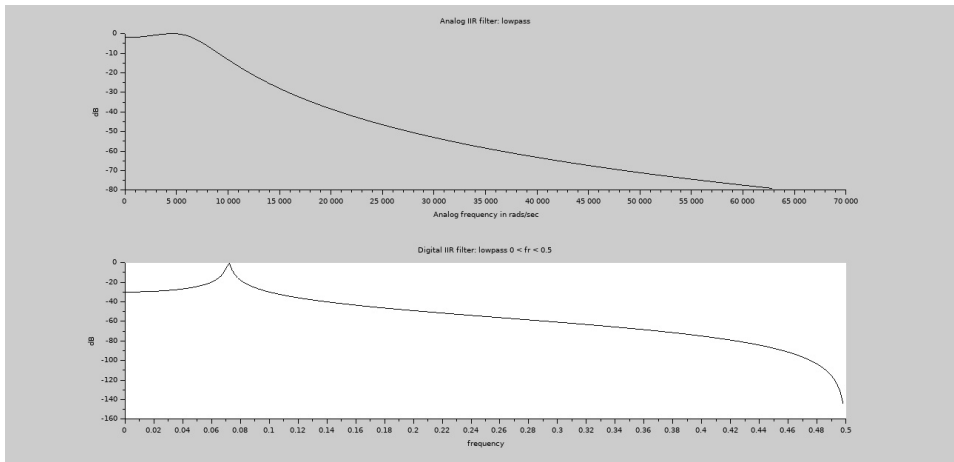


Figure 10.2: Design of Digital Chebyshev lowpass filter

```

49 [hzm,fr]=frmag(hz,256);
50 magz=20*log10(hzm)';
51 figure(1);subplot(2,1,1),plot2d((1:2*%pi*fs)',mag),
    xtitle('Analog IIR filter: lowpass','Analog
    frequency in rads/sec','dB','');subplot(2,1,2),
    plot2d(fr,magz);xtitle('Digital IIR filter:
    lowpass 0 < fr < 0.5','frequency','dB','');

```

---

# Experiment: 11

## Application of IIR filter

**Scilab code Solution 11.1** To design a digital IIR Butterworth filter to suppress noise

```
1 //scilab 5.5.2 , OS: Ubuntu 14.04
2 // This program will suppress noise at f=4000 Hz
   using Butterworth prototype
3 //pass band edge=f1=1500Hz
4 //stop band edge=f2=2000 Hz
5 //sampling rate =Fs=10000 Hz = 1/Ts
6 //passband attenuation = -1db
7 //stop attenuation = -3 db
8
9 clear all;clc;close;
10 f1=input('Enter the pass band edge(Hz)= ');
11 f2=input('Enter the stop band edge(Hz)= ');
12 k1=input('Enter the pass band attenuation(dB)= ');
13 k2=input('Enter the stop band attenuation(dB)= ');
14 fs=input('Enter the sampling rate(Hz)= ');
15
16 signal_fo=1000;
17 noise_fo=4000;
18
19 //Digital filter specifications(rad)
```

```

20 w1=2*%pi*f1*1/fs;
21 w2=2*%pi*f2*1/fs;
22
23 //Pre warping
24 o1=2*fs*tan(w1/2)
25 o2=2*fs*tan(w2/2)
26
27 //Design of analog filter
28 n=log10(((10.^(-k1/10))-1)/((10.^(-k2/10))-1))./(2*
    log10(o1/o2));
29 n=round(n);
30 wn= o2./((10.^(-k2/10)-1).^1/(2*n));
31
32 // [h, poles , zeros , gain]=analpf(n, 'butt ', [0 0], wn)hb.
    dt = 'c';
33 // [fr , hr]=repfreq (hb, fmin , fmax)
34
35 h=buttmag(n, wn, 1:2*%pi*fs);
36 mag=20*log10(h)';
37 //plot2d((1:2*%pi*fs)', mag)
38 //xtitle('Analog IIR filter: lowpass ', 'Analog
    frequency in rads/sec ', 'dB', ' ');
39
40 //Converting analog to digital filter
41 hz=iir(n, 'lp ', 'butt ', 0.25, [])
42 //g*poly(z, 'z')/poly(p, 'z')
43
44 [hzm, fr]=frmag(hz, 256);
45 magz=20*log10(hzm)';
46 fr1=fr*fs;
47 //figure; plot2d(fr1', magz'); xtitle('Digital IIR
    filter: lowpass 0 < fr < 0.5 ', 'frequency ', 'dB', '
    ');
48
49 ///////note: Use zoom/axis commands to verify the
    design.
50 //These coefficients are to be read from variable hz
    (line 41, output of iir function)

```

```

51 num=[0.2928 0.5858 0.2928];
52 den=[1 0 0.1716]; // In negative powers of z
53
54 //Signal generation (sine wave of frequency 1000 Hz)
    of length 1 second
55 t=0:1/fs:10/signal_fo; //10 cycles of input
56 original_signal=sin(2*%pi*signal_fo*t);
57
58 //Noise generation (sine wave of frequency 4000 Hz)
    of length 1 second
59 t=0:1/fs:10/signal_fo;
60 noise=sin(2*%pi*noise_fo*t);
61
62 noisy_signal=original_signal+noise;
63
64 filter_output=filter(num,den,noisy_signal);
65
66 //Plot original, noisy and filtered outputs
67
68 figure;subplot(3,1,1), plot2d(t,original_signal),
    xtitle('Original_signal','t','x(t)'),
69 subplot(3,1,2), plot2d(t,noisy_signal),xtitle('
    Noisy_signal','t','n(t)'),
70 subplot(3,1,3), plot2d(t,filter_output),xtitle('
    Filtered_signal','t','y(t)'),
71 l1=length(original_signal);
72 l2=length(noisy_signal);
73 N=512;
74 x=[original_signal zeros(1,N-l1)]; //To make it of
    length 512
75 n=[noisy_signal zeros(1,N-l1)];
76 y=[filter_output zeros(1,N-l1)];
77 X=fft(x);
78 N=fft(n);
79 Y=fft(y);
80 f=(0:511)*fs;
81 figure;
82 subplot(3,1,1), plot2d(f,abs(X)),xtitle('

```

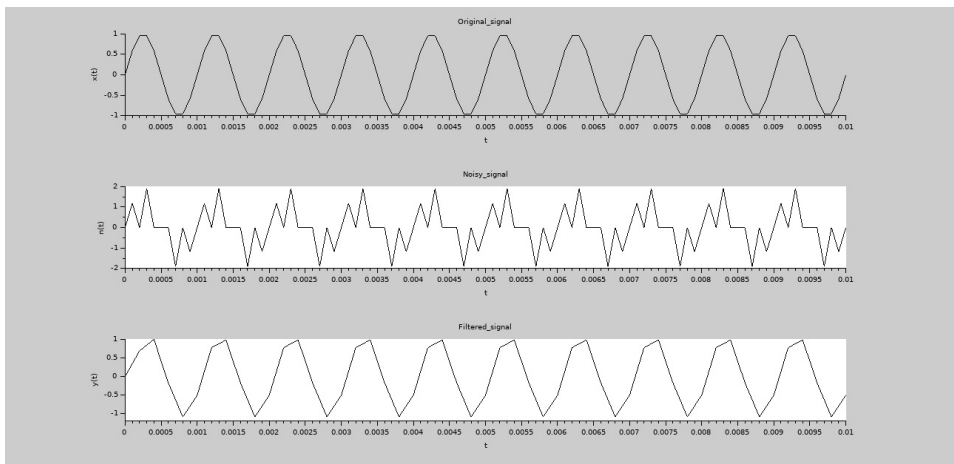


Figure 11.1: To design a digital IIR Butterworth filter to suppress noise

```

Original_signal', 'F', 'X(f) '),
83 subplot(3,1,2), plot2d(f,abs(N)), xtitle('
   Noisy_signal', 'F', 'N(f) '),
84 subplot(3,1,3), plot2d(f,abs(Y)), xtitle('
   Filtered_signal', 'F', 'Y(f) ');

```

---

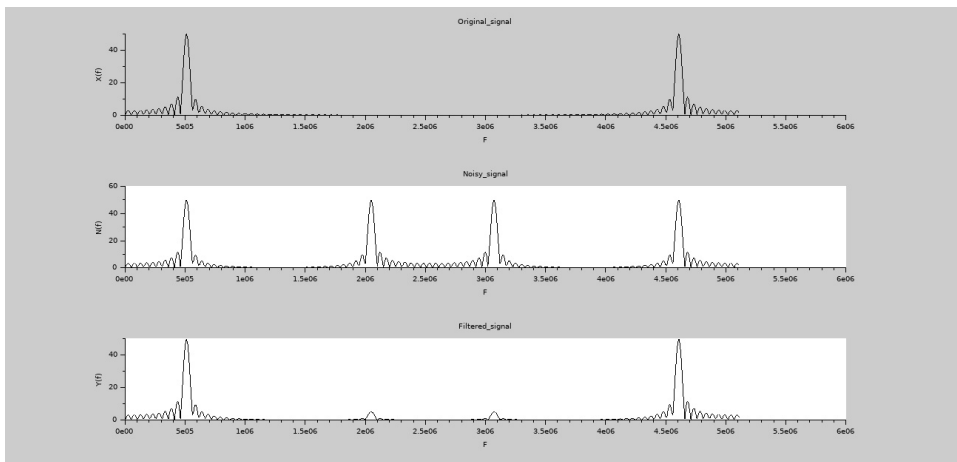


Figure 11.2: To design a digital IIR Butterworth filter to suppress noise

# Experiment: 12

## Design of Notch filter

**Scilab code Solution 12.1** Suppression of noise at a given frequency using Notch filter

```
1
2 //scilab 5.5.2 , OS: Ubuntu 14.04
3 //Program To Design a simple notch filter and verify
4 // Design a simple notch filter to stop a
   disturbance with frequency F_0=3.5 kHz and a
   sampling frequency F_s=8 kHz.
5 //Also, verify the notch filter operation by adding
   a sinewave of F_0 Hz to a speech signal, filter
   and verify.
6
7 //Scilab Program:
8 clc; clear; close;
9 f=3500; //input("Enter the frequency in Hz");
   //3500
10 fs=8000; //input("Enter the sampling rate");
   //8000
11 r=0.98; //input("Enter the radius of the pole in the
   z-plane"); //0.98
12 w=2*%pi*f/fs;
13 z1=exp(%i*w);
```

```

14 z2=exp(-%i*w);
15 p1=r*exp(%i*w);
16 p2=r*exp(-%i*w);
17 z=%z;
18 num1=(real((z-z1)*(z-z2)))
19 den1=(real(((z-p1)*(z-p2))))
20 Hz=num1./den1
21 //figure(1);plzr(Hz);zgrid()
22 [h1 fr]=frmag(Hz,512)
23 figure(1);plot2d(fr*fs,h1);xlabel('Magnitude
    response','frequency in Hz','Mag');
24
25 //Noise generation
26
27 original_signal=wavread('home/hyrkswamy/kswamy/
    Coursework/SAP/wav/mask.wav');
28 t=0:1/fs:(length(original_signal)-1)/fs;
29 noise=sin(2*pi*f*t);
30 noisy_signal=original_signal+noise;
31
32 filter_output=filter(num1,den1,noisy_signal);
33
34 //Play back the original, noisy and filtered outputs
35 playsnd(original_signal,fs);
36 pause;
37 playsnd(noisy_signal,fs);
38 pause;
39 playsnd(filter_output,fs);

```

---

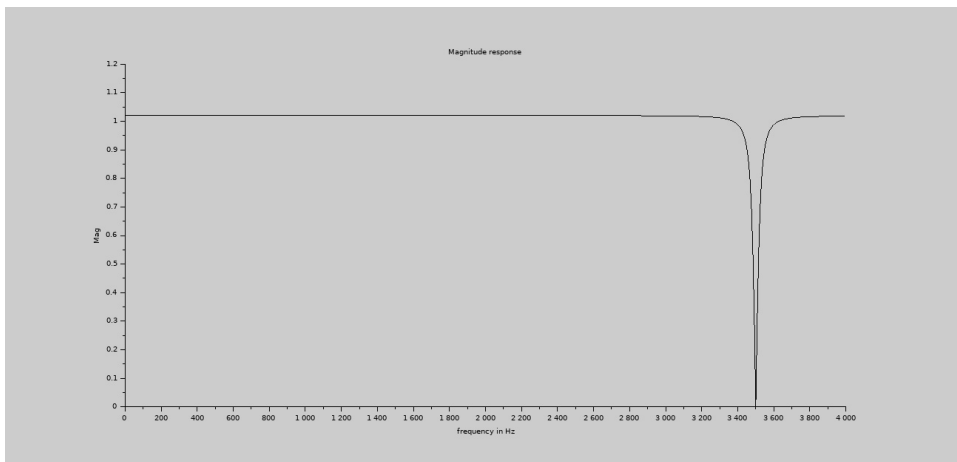


Figure 12.1: Suppression of noise at a given frequency using Notch filter

# Experiment: 13

## Design of Resonator

**Scilab code Solution 13.1** Design of a Notch filter to filter noise at a given frequency

```
1
2 //scilab 5.5.2 , OS: Ubuntu 14.04
3 //Design a digital resonator that resonates at 1000
  Hz. Assume Fs=8000 Hz.
4 // Calculate the pole location
5 //w=2*pi*f/fs;
6 //Complex conjugate pair of poles at w=pi/4 and -pi
  /4
7 //Assume radius=0.98 (near to unit circle but inside
  the unit circle)
8
9 //Scilab Program:
10 clc;
11 clear;
12 close;
13 f=1000;//input("Enter the frequency in Hz");
  //1000
14 fs=8000;//input("Enter the sampling rate");
  //8000
15 r=0.98;//input("Enter the radius of the pole in the
```

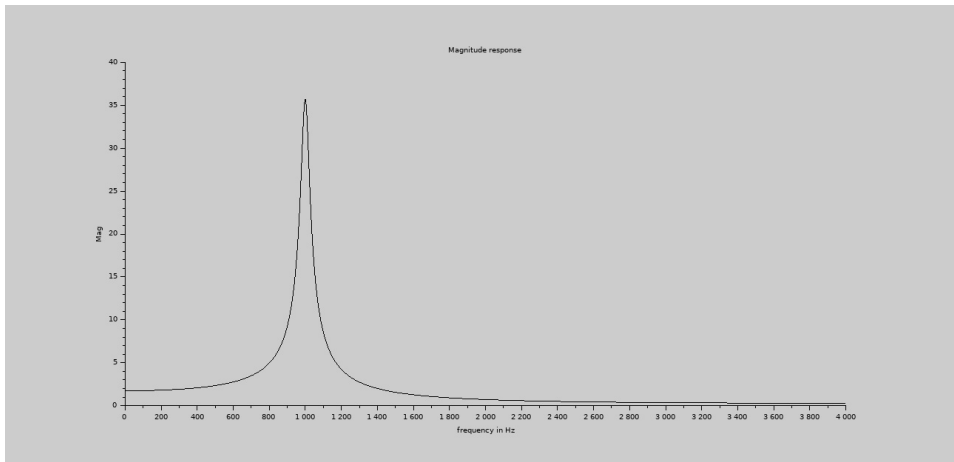


Figure 13.1: Design of a Notch filter to filter noise at a given frequency

```

    z-plane"); //0.98
16 w=2*pi*f/fs;
17 pole1=r*exp(i*w);
18 pole2=r*exp(-i*w);
19
20 z=z;
21
22 num1=real(z^(2));
23 den1=real(z^(2)-1.3859293*z+0.9604);
24 Hz=num1./den1;
25 //figure;
26 //plzr(Hz);
27 [h1 fr]=frmag(Hz,1024);
28 figure;
29 plot2d(fr*fs,h1);
30 xtitle('Magnitude response','frequency in Hz','Mag')
    ;

```

---