

Scilab Manual for  
Digital Signal Processing  
by Mr Rajesh B Raut  
Electronics Engineering  
Shri Ramdeobaba College of Engg & Mgmt<sup>1</sup>

Solutions provided by  
Mr. Vipul S Lande  
Electronics Engineering  
Shri Ramdeobaba College of Engg & Mgmt, Nagpur

June 23, 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	4
1 Basic operations on sequences and plotting them in continuous/discrete form.	6
2 Analyzing the effect of Sampling of continuous time signal to avoid aliasing.	13
3 Convolving two sequences in discrete time domain in discrete frequency domain.	15
4 Evaluating circular convolution using DFT-IDFT method.	18
5 Designing of FIR filters for low pass, high pass and band reject response.	21
6 Designing of Butterworth IIR filters and its realisation to filter out noise from a given signal.	27
7 Designing of Chebychev/inverse Chebychev/elliptical filter from a given transfer function.	29
8 Application of sound effect on wave file like Flanging, Echo and Equiliser.	31
9 Polyphase decomposition by decimation method for a given decimation factor.	34
10 Spectral estimation of a sequence using Periodogram method.	36

11 Real-time data acquisition and plotting through external hardware interfaced through serial port.	38
--	----

# List of Experiments

Solution 1.0	Basic operations on sequences and plotting them in continuous form . . . . .	6
Solution 1.1	Basic operations on sequences and plotting them in discrete form . . . . .	8
Solution 1.2	Basic operations on sequences Delaying and Advancing . . . . .	10
Solution 2.0	Analyzing the effect of sampling of continuous time signal to avoid aliasing . . . . .	13
Solution 3.0	Convolving two sequences in discrete time domain	15
Solution 4.0	Evaluating circular convolution using DFT IDFT method . . . . .	18
Solution 5.1	Design an Low Pass FIR Filter . . . . .	21
Solution 5.2	To Design an High Pass FIR Filter . . . . .	22
Solution 5.3	To Design Band Pass FIR Filter . . . . .	23
Solution 5.4	To Design Band Stop FIR Filter . . . . .	25
Solution 6.0	Designing of butterworth IIR filters and its realization to filter our noise from a given signal . . . . .	27
Solution 7.0	Designing of chebychev inverse chebyshev elliptical filter from a given transfer function . . . . .	29
Solution 8.0	Application of sound effect on wave file like Flanging Echo and Equalizer . . . . .	31
Solution 9.0	Polyphase decomposition by decimation method for a given decimation factor . . . . .	34
Solution 10.0	Spectral estimation of a sequence using Periodogram method . . . . .	36
Solution 11.0	Real time Data acquisition and plotting through external hardware interfaced through serial port .	38

# List of Figures

11.1 Real time Data acquisition and plotting through external hardware interfaced through serial port . . . . .	39
---	----

# Experiment: 1

## Basic operations on sequences and plotting them in continuous/discrete form.

Scilab code Solution 1.0 Basic operations on sequences and plotting them  
in continuous form

```
1 //Experiment -1.1
2 //Basic operations on sequences and plotting them in
   continuous form.
3
4 clear;
5 clc;
6 close;
7
8 //unit step
9 N=input('enter time of unit step sequence: ');//N=20
   (only integer value)
10 n=0:0.5:N-1;
11 NL=length(n)
12 y=ones(1,NL);
13 subplot(2,2,1);
14 plot(n,y);
```

```

15 title('UNIT STEP');
16 xlabel('Time—>'); ylabel('AMPL—>');
17
18 //ramp
19 N=input('enter time of ramp sequence: ');//N=20 (
    only intiger value)
20 n=0:0.5:N-1;
21 subplot(2,2,2);
22 plot(n,n);
23 title('RAMP');
24 xlabel('Time—>'); ylabel('AMPL—>');
25
26 //exponential exp(a*t)
27 N=input('enter time of expo. sequence: ');//N=20 (
    only intiger value)
28 n=0:0.5:N-1;
29 a=input('enter value of a= ');//a=0.03
30 y=exp(a*n);
31 subplot(2,2,3);
32 plot(n,y);
33 title('EXPONENTIAL');
34 xlabel('Time—>'); ylabel('AMPL—>');
35
36 //sine wave
37 f=input('enter frequency of sine sequence: ');//f=5
    (only intiger value)
38 t=0:0.001:1;
39 y=sin(2*%pi*f*t);
40 figure(1);
41 subplot(3,2,1);
42 plot(t,y);
43 title('SINE');
44 xlabel('Time—>'); ylabel('AMPL—>');
45
46 //cosine wave
47 f2=input('enter frequency of cosine sequence: ');//f
    =5 (only intiger value)
48 t=0:0.001:1;

```

```

49 y2=cos(2*%pi*f2*t);
50 subplot(3,2,2);
51 plot(t,y2);
52 title('COSINE');
53 xlabel('Time-->'); ylabel('AMPL-->');
54
55 //sine + cosine wave
56 y3=cos(2*%pi*f2*t)+sin(2*%pi*f*t);
57 subplot(3,2,3);
58 plot(t,y3);
59 title('SINE+COSINE');
60 xlabel('Time-->'); ylabel('AMPL-->');
61
62 //sine x sine wave
63 y4=y.*y2;
64 subplot(3,2,4);
65 plot(t,y4);
66 title('SINE x COSINE');
67 xlabel('Time-->'); ylabel('AMPL-->');

```

---

**Scilab code Solution 1.1** Basic operations on sequences and plotting them in discrete form

```

1 //Experiment -1.2
2 //Basic operations on sequences and plotting them in
   discrete form.
3
4 clear;
5 clc;
6 close;
7
8 //impulse
9 N=-3:1:3;
10 y=[zeros(1,3), ones(1,1), zeros(1,3)];
11 subplot(2,2,1);

```

```

12 plot2d3(N,y);
13 title('IMPULSE');
14 xlabel('N'); ylabel('AMPL—>');
15
16 //unit step
17 N=input('enter length of unit step sequence: '); //N
    =20 (only intiger value)
18 n=0:1:N-1;
19 y=ones(1,N);
20 subplot(2,2,2);
21 plot2d3(n,y);
22 title('UNIT STEP');
23 xlabel('N'); ylabel('AMPL—>');
24
25 //ramp
26 N=input('enter length of ramp sequence: ');//N=20 (
    only intiger value)
27 n=0:1:N-1;
28 subplot(2,2,3);
29 plot2d3(n,n);
30 title('RAMP');
31 xlabel('N'); ylabel('AMPL—>');
32
33 //exponential exp(a*t)
34 N=input('enter length of expo. sequence: ');//N=20 (
    only intiger value)
35 n=0:1:N-1;
36 a=input('enter value of a= ');//a=0.03
37 y=exp(a*n);
38 subplot(2,2,4);
39 plot2d3(n,y);
40 title('EXPONENTIAL');
41 xlabel('N'); ylabel('AMPL—>');
42
43 //sine wave
44 N=input('enter length of sine sequence: ');//N=60 (
    only intiger value)
45 n=0:1:N-1;

```

```

46 y=sin(0.3*%pi*n);
47 figure(1);
48 subplot(3,1,1);
49 plot2d3(n,y);
50 title('SINE');
51 xlabel('N'); ylabel('AMPL—>');
52
53 //cosine wave
54 N=input('enter length of cosine sequence: ');//N=60
    (only intiger value)
55 n=0:1:N-1;
56 y=cos(0.2*%pi*n);
57 subplot(3,1,2);
58 plot2d3(n,y);
59 title('COSINE');
60 xlabel('N'); ylabel('AMPL—>');
61
62 //sine + cosine wave
63 N=input('enter length of cosine/sine sequence: ');//
    N=80 (only intiger value)
64 n=0:1:N-1;
65 y=cos(0.2*%pi*n)+sin(0.3*%pi*n);
66 subplot(3,1,3);
67 plot2d3(n,y);
68 title('SINE+COSINE');
69 xlabel('N'); ylabel('AMPL—>');

```

---

**Scilab code Solution 1.2** Basic operations on sequences Delaying and Advancing

```

1 //Experiment –1.3
2 //Basic operations on sequences and plotting them in
    continuous form.
3 clc;
4 clear;

```

```

5 close;
6 x = input('Enter the input sequence:=')//x =[1 2 3
    -2 3]
7 m = length(x);
8 lx = input('Enter the starting index of original
    signal:=')//lx = -2
9 hx = lx+m-1;
10 n = lx:1:hx;
11 subplot(3,1,1)
12 a = gca();
13 a.x_location = "origin";
14 a.y_location = "origin";
15 a.data_bounds = [-10,0;10,10];
16 plot2d3('gnn',n,x);
17 xlabel('n====>')
18 ylabel('Amplitdue——>')
19 title('Original Sequence')
20 //
21 d = input('Enter the delay:=')// d = 1
22 n = lx+d:1:hx+d;
23 subplot(3,1,2)
24 a = gca();
25 a.x_location = "origin";
26 a.y_location = "origin";
27 a.data_bounds = [-10,0;10,10];
28 plot2d3('gnn',n,x)
29 xlabel('n====>')
30 ylabel('Amplitude——>')
31 title('Delayed Sequence')
32 //
33 a = input('Enter the advance:=')// a = 2
34 n = lx-a:1:hx-a;
35 subplot(3,1,3)
36 a = gca();
37 a.x_location = "origin";
38 a.y_location = "origin";
39 a.data_bounds = [-10,0;10,10];
40 plot2d3('gnn',n,x)

```

```
41 xlabel('n====>')
42 ylabel('Amplitude——>')
43 title('Advanced Sequence')
```

---

## Experiment: 2

# Analyzing the effect of Sampling of continuous time signal to avoid aliasing.

Scilab code Solution 2.0 Analyzing the effect of sampling of continuous time signal to avoid aliasing

```
1 //Experiment – 2
2 //Analyzing the effect of sampling of continuous
   time signal to avoid aliasing.
3
4 clear;
5 clc;
6 close;
7
8
9 // Aliasing
10
11 f = 60; // signal freq. in Hz
12 tmin = -0.05;
13 tmax = 0.05;
14 t = linspace(tmin, tmax, 400);
15 alpha = 1;
```

```

16 x_c = cos(alpha * 2 * %pi * f * t);
17 figure;
18 plot(t, x_c)
19
20 Fs = 200 // for 400, 200, 120, 70, 40 Hz
21
22 T = 1/Fs;
23 nmin = ceil(tmin/T);
24 nmax = floor(tmax/T);
25 n = nmin:nmax;
26 x_s = cos(2*%pi*f*n*T);
27 figure;
28 plot(t, x_c)
29 mtlb_hold("on")
30 plot(n*T, x_s, '.')
31 mtlb_hold("off")
32
33 figure;
34 plot(n*T, x_s, '-. ');
35
36 //viewing spectrum of signals
37 t=soundsec(0.5);
38 s1=sin(2*%pi*60*t); // signal frequency=60Hz
39 s2=sin(2*%pi*200*t); // sampling frequency=200Hz
40 analyze(s1,10,600,22050)
41 analyze(s2,10,600,22050)
42 title('Spectrum of signals')

```

---

## Experiment: 3

# Convolving two sequences in discrete time domain in discrete frequency domain.

Scilab code Solution 3.0 Convolving two sequences in discrete time domain

```
1 //Experiment - 3
2 //Convolving two sequences in discrete time domain.
3
4
5 clc;
6 clear ;
7 close ;
8 x=input('enter i/p x(n):'); // x=[1 2 3 4 5 6]
9 m=length(x);
10 h=input('enter i/p h(n):'); // h=[1 -1 1]
11 n=length(h);
12
13 subplot(2,2,1),
14 p=0:1:m-1;
15
16 a = gca ();
```

```

17 a.x_location = "origin";
18 a.y_location = "origin";
19 plot2d3('gmn',p,x)
20
21 title('i/p sequence x(n) is:');
22 xlabel('—>n');
23
24
25 x=[x,zeros(1,n)];
26
27 subplot(2,2,2),
28 q=0:1:n-1;
29
30 a = gca ();
31 a.x_location = "origin";
32 a.y_location = "origin";
33 plot2d3('gmn',q,h)
34 title('i/p sequence h(n) is:');
35 xlabel('—>n');
36
37
38 h=[h,zeros(1,m)];
39
40 disp('convolution of x(n) & h(n) is y(n):');
41 y=zeros(1,m+n-1);
42 for i=1:m+n-1
43     y(i)=0;
44     for j=1:m+n-1
45         if(j<i+1)
46             y(i)=y(i)+x(j)*h(i-j+1);
47         end
48     end
49 end
50 y
51 subplot(2,2,3)
52 r=0:1:m+n-2;
53
54 a = gca ();

```

```
55 a.x_location = "origin";
56 a.y_location = "origin";
57 plot2d3('gnn',r,y)
58
59
60 title('convolution of x(n) & h(n) is :');
61 xlabel('—>n');
```

---

## Experiment: 4

# Evaluating circular convolution using DFT-IDFT method.

Scilab code Solution 4.0 Evaluating circular convolution using DFT IDFT method

```
1 //Experiment - 4
2 //Evaluating circular convolution using DFT-IDFT
  method.
3
4 clear ;
5 clc;
6 close ;
7 x=input('enter 1st seq: ');// x=[1 2 3 4 5 6]
8 h=input('enter 2nd seq: ');// h=[1 -1 1]
9
10
11 dif =abs( max(size(x))- max(size(h)));
12 if(dif>0)
13     h(max(size(h))+dif) = 0;
14 else
15     x(max(size(x))+dif) = 0;
16 end
17
```

```

18
19 X=fft(x,-1);
20 H=fft(h,-1);
21 Y=X.*H;
22 y=ifft(Y);
23
24 subplot(3,2,1);
25 n=0:1:length(x)-1
26 plot2d3(n,x);
27 ylabel('amplitude——>');
28 xlabel('x(n)      n——>');
29
30 subplot(3,2,2);
31 plot2d3(n,abs(X));
32 ylabel('amplitude——>');
33 xlabel('X(k)      k——>');
34
35 subplot(3,2,3);
36 n=0:1:length(h)-1
37 plot2d3(n,h);
38 ylabel('amplitude——>');
39 xlabel('h(n)      n——>');
40
41 subplot(3,2,4);
42 plot2d3(n,abs(H));
43 ylabel('amplitude——>');
44 xlabel('H(k)      k——>');
45
46 subplot(3,2,5);
47 n=0:1:length(y)-1
48 plot2d3(n,y);
49 ylabel('amplitude——>');
50 xlabel('y(n)      n——>');
51
52 subplot(3,2,6);
53 plot2d3(n,abs(Y));
54 ylabel('amplitude——>');
55 xlabel('Y(k)      k——>');

```

56

57 `disp(y, 'the circular conv. resultant signal is');`

---

## Experiment: 5

# Designing of FIR filters for low pass, high pass and band reject response.

Scilab code Solution 5.1 Design an Low Pass FIR Filter

```
1 //Experiment - 5
2 //Designing of FIR filters for low pass, high pass,
   band pass and band reject response.
3
4 // To Design an Low Pass FIR Filter
5 //Filter Length =5, Order = 4
6 //Window = Rectangular Window
7
8 clc;
9 clear;
10 xdel(winsid());
11 fc = input("Enter Analog cutoff freq. in Hz=")//250
12 fs = input("Enter Analog sampling freq. in Hz=")//
   2000
13 M = input("Enter order of filter =")//4
14 w = (2*%pi)*(fc/fs);
15 disp(w, 'Digital cutoff frequency in radians.cycles/
```

```

    samples ');
16 wc = w/%pi;
17 disp(wc, 'Normalized digital cutoff frequency in
    cycles/samples ');
18 [wft,wfm,fr]=wfirm('lp',M+1,[wc/2,0], 're',[0,0]);
19 disp(wft, 'Impulse Response of LPF FIR Filter:h[n]=')
    ;
20 //Plotting the Magnitude Response of LPF FIR Filter
21 subplot(2,1,1)
22 plot(2*fr,wfm)
23 xlabel('Normalized Digital Frequency w—>')
24 ylabel('Magnitude |H(w)|=')
25 title('Magnitude Response of FIR LPF')
26 xgrid(1)
27 subplot(2,1,2)
28 plot(fr*fs,wfm)
29 xlabel('Analog Frequency in Hz f —>')
30 ylabel('Magnitude |H(w)|=')
31 title('Magnitude Response of FIR LPF')
32 xgrid(1)

```

---

### Scilab code Solution 5.2 To Design an High Pass FIR Filter

```

1 //Experiment – 5
2 //Designing of FIR filters for low pass, high pass,
    band pass and band reject response.
3
4 // To Design an High Pass FIR Filter
5 //Filter Length =5, Order = 4
6 //Window = Rectangular Window
7
8 clc;
9 clear;
10 xdel(winsid());
11 fc = input("Enter Analog cutoff freq. in Hz=")//250

```

```

12 fs = input("Enter Analog sampling freq. in Hz=")//
    2000
13 M = input("Enter order of filter =")//4
14 w = (2*%pi)*(fc/fs);
15 disp(w, 'Digital cutoff frequency in radians.cycles/
    samples');
16 wc = w/%pi;
17 disp(wc, 'Normalized digital cutoff frequency in
    cycles/samples');
18 [wft,wfm,fr]=wfir('hp',M+1,[wc/2,0], 're',[0,0]);
19 disp(wft, 'Impulse Response of HPF FIR Filter:h[n]=')
    ;
20 //Plotting the Magnitude Response of HPF FIR Filter
21 subplot(2,1,1)
22 plot(2*fr,wfm)
23 xlabel('Normalized Digital Frequency w—>')
24 ylabel('Magnitude |H(w)|=')
25 title('Magnitude Response of FIR HPF')
26 xgrid(1)
27 subplot(2,1,2)
28 plot(fr*fs,wfm)
29 xlabel('Analog Frequency in Hz f —>')
30 ylabel('Magnitude |H(w)|=')
31 title('Magnitude Response of FIR HPF')
32 xgrid(1)

```

---

### Scilab code Solution 5.3 To Design Band Pass FIR Filter

```

1 //Experiment – 5
2 //Designing of FIR filters for low pass, high pass,
    band pass and band reject response.
3
4 // To Design an Band Pass FIR Filter
5 //Filter Length =5, Order = 4
6 //Window = Rectangular Window

```

```

7
8 clc;
9 clear;
10 xdel(winsid());
11 fc1 = input("Enter Analog lower cutoff freq. in Hz="
    )//250
12 fc2 = input("Enter Analog higher cutoff freq. in Hz="
    ")//600
13 fs = input("Enter Analog sampling freq. in Hz=")//
    2000
14 M = input("Enter order of filter =")//4
15 w1 = (2*%pi)*(fc1/fs);
16 w2 = (2*%pi)*(fc2/fs);
17 disp(w1, 'Digital lower cutoff frequency in radians.
    cycles/samples');
18 disp(w2, 'Digital higher cutoff frequency in radians.
    cycles/samples');
19 wc1 = w1/%pi;
20 wc2 = w2/%pi;
21 disp(wc1, 'Normalized digital lower cutoff frequency
    in cycles/samples');
22 disp(wc2, 'Normalized digital higher cutoff frequency
    in cycles/samples');
23 [wft,wfm,fr]=wfir('bp',M+1,[wc1/2,wc2/2], 're',[0,0])
    ;
24 disp(wft, 'Impulse Response of BPF FIR Filter:h[n]=')
    ;
25 //Plotting the Magnitude Response of HPF FIR Filter
26 subplot(2,1,1)
27 plot(2*fr,wfm)
28 xlabel('Normalized Digital Frequency w—>')
29 ylabel('Magnitude |H(w)|=')
30 title('Magnitude Response of FIR BPF')
31 xgrid(1)
32 subplot(2,1,2)
33 plot(fr*fs,wfm)
34 xlabel('Analog Frequency in Hz f —>')
35 ylabel('Magnitude |H(w)|=')

```

```
36 title('Magnitude Response of FIR BPF')
37 xgrid(1)
```

---

#### Scilab code Solution 5.4 To Design Band Stop FIR Filter

```
1 //Experiment – 5
2 //Designing of FIR filters for low pass, high pass,
   band pass and band reject response.
3
4 // To Design an Band Stop FIR Filter
5 //Filter Length =5, Order = 4
6 //Window = Rectangular Window
7
8 clc;
9 clear;
10 xdel(winsid());
11 fc1 = input("Enter Analog lower cutoff freq. in Hz="
   )//250
12 fc2 = input("Enter Analog higher cutoff freq. in Hz="
   )//600
13 fs = input("Enter Analog sampling freq. in Hz=")//
   2000
14 M = input("Enter order of filter =")//4
15 w1 = (2*%pi)*(fc1/fs);
16 w2 = (2*%pi)*(fc2/fs);
17 disp(w1, 'Digital lower cutoff frequency in radians.
   cycles/samples');
18 disp(w2, 'Digital higher cutoff frequency in radians.
   cycles/samples');
19 wc1 = w1/%pi;
20 wc2 = w2/%pi;
21 disp(wc1, 'Normalized digital lower cutoff frequency
   in cycles/samples');
22 disp(wc2, 'Normalized digital higher cutoff frequency
   in cycles/samples');
```

```

23 [wft,wfm,fr]=wfirm('sb',M+1,[wc1/2,wc2/2],'re',[0,0])
    ;
24 disp(wft,'Impulse Response of BSF FIR Filter:h[n]=')
    ;
25 //Plotting the Magnitude Response of HPF FIR Filter
26 subplot(2,1,1)
27 plot(2*fr,wfm)
28 xlabel('Normalized Digital Frequency  $w \longrightarrow$ ')
29 ylabel('Magnitude  $|H(w)|=$ ')
30 title('Magnitude Response of FIR BSF')
31 xgrid(1)
32 subplot(2,1,2)
33 plot(fr*fs,wfm)
34 xlabel('Analog Frequency in Hz  $f \longrightarrow$ ')
35 ylabel('Magnitude  $|H(w)|=$ ')
36 title('Magnitude Response of FIR BSF')
37 xgrid(1)

```

---

## Experiment: 6

# Designing of Butterworth IIR filters and its realisation to filter out noise from a given signal.

Scilab code Solution 6.0 Designing of butterworth IIR filters and its realization to filter our noise from a given signal

```
1 //Experiment - 6
2 //Designing of butterworth IIR filters and its
   realization to filter our noise from a given
   signal.
3
4
5
6 t = 0:0.001:4;
7 signal = sin(2*pi*20*t);
8 noise = sin(2*pi*50*t);
9 compound = signal + noise;
10
11 playsnd(compound);
12
```

```
13 myfilter = iir(15, 'lp', 'butt', [0.025 0], [0 0]); //  
    butt, cheb1/2, ellip  
14  
15 output = flts(compound, myfilter);  
16  
17 halt('Press Enter');  
18  
19 playsnd(output);  
20  
21 // Sound difference in filtered sound  
22 xsetech([0,0,1,1/2]);  
23 plot2d(t, compound);  
24 xtitle('input signal');  
25 xsetech([0,1/2,1,1/2]);  
26 plot2d(t, output);  
27 xtitle('output signal');
```

---

## Experiment: 7

# Designing of Chebychev/inverse Chebychev/elliptical filter from a given transfer function.

Scilab code Solution 7.0 Designing of chebychev inverse chebyshev elliptical filter from a given transfer function

```
1 //Experiment - 7
2 //Designing of chebychev/inverse chebyshev/
   elliptical filter from a given transfer function.
3
4 clc;
5 clear all;
6
7 cheblpf = iir(2, 'lp', 'cheb1', [0.3 0], [0.1 0]);
8 [hzm, fr]=frmag(cheblpf, 256);
9 figure;
10 plot2d(fr', hzm')
11 title('Chebysheve Lowpass Filter')
12
13
```

```

14
15 [cells, fact, zzeros, zpoles]=eqiir('lp', 'ellip', [%pi
      *3/8, %pi*3.5/8], 0.1, 0.01);
16 h=fact*poly(zzeros, 'z')/poly(zpoles, 'z');
17 [hzm, fr]=frmag(h, 256);
18 figure;
19 plot2d(fr, hzm)
20 title('Elliptical Lowpass Filter')
21
22
23
24 [valcoeff, filtamp, filtfreq] = wfir('lp', 20, [.2
      0], 'hm', [0 0]);
25 figure;
26 plot2d(filtfreq, filtamp)
27 title('FIR-Hamming window Lowpass Filter')
28
29
30 hn=eqfir(33, [0 .2;.25 .35;.4 .5], [0 1 0], [1 1 1]);
31 [hm, fr]=frmag(hn, 256);
32 figure;
33 plot2d(fr, hm)
34 title('FIR-Multiband Filter response')

```

---

## Experiment: 8

# Application of sound effect on wave file like Flanging, Echo and Equaliser.

**Scilab code Solution 8.0** Application of sound effect on wave file like Flanging Echo and Equalizer

```
1
2 //Experiment – 8
3 //Application of sound effect on wave file like
   Flanging , Echo and Equalizer .
4
5
6 // The "dsp01.wav" file should be placed in current
   working directory
7 // The code can also be executed with any other
   small .wav file of duration 3–5 seconds.
8 clc;
9 clear;
10 clear all;
11
12 [y,fs] = wavread('dsp01.wav');
13 playsnd(y,fs);
```

```

14
15 halt('Original');
16
17 //Flanging
18 z = 0;
19 n = 1:length(y);
20 z(n) = y(n);
21 m = 11:length(y);
22 x(m) = z(m) + 0.8*z(m-10).*cos(2*%pi*m/fs);
23 playsnd(x,fs);
24 halt('Flange');
25
26
27 //Echo
28 clc;
29 clear all;
30 a = 0.5;
31 [l,fs] = wavread('dsp01.wav');
32 len = length(l);
33 delay = 0.4;
34 D = ceil(fs*delay);
35 m = zeros(max(size(l)));
36 for i = D+1:len
37     m(i) = l(i) + a*l(i-D);
38 end
39
40 playsnd(m,fs);
41
42
43 halt('Echo');
44
45 //Equalizer
46 //For Low frequencies
47 clc;
48 clear all;
49
50 [h,fs] = wavread('dsp01.wav');
51 playsnd(h,fs);

```

```

52 halt('Original');
53
54 j = iir(3, 'bp', 'butt', [.01 .05], [0.03 0.03]);
55
56 k = flts(h(1,:), j);
57 playsnd(k, fs);
58 halt('Low');
59
60 // For Mid frequencies
61 clc;
62 clear all;
63
64 [g, fs] = wavread('dsp01.wav');
65 playsnd(g, fs);
66 halt('Original');
67
68 j = iir(3, 'bp', 'ellip', [.05 .10], [.08 .03]);
69
70 k = flts(g(2,:), j);
71 sound(k, fs);
72
73 halt('Mid');
74
75 //For High frequencies
76 clc;
77 clear all;
78
79 [h, fs] = wavread('dsp01.wav');
80 playsnd(h, fs);
81 halt('Original');
82
83 j = iir(3, 'bp', 'butt', [.15 .25], [0.03 0.03]);
84
85 k = flts(h(1,:), j);
86 playsnd(k, fs);
87
88 halt('High');

```

---

## Experiment: 9

# Polyphase decomposition by decimation method for a given decimation factor.

Scilab code Solution 9.0 Polyphase decomposition by decimation method for a given decimation factor

```
1 //Experiment - 9
2 //Polyphase decomposition by decimation method for a
   given decimation factor.
3
4 clear;
5 clc;
6 close;
7 M =input('Enter the filter length(M)= '); //Filter
   Length = 30
8 D =input('Decimation Factor(D) ='); //Decimation
   Factor = 2
9 Ws =input('Enter stopband frequency(Ws)= '); //
   Stopband Edge Frequency = 0.31
10 Wc = %pi/2; //Cutoff Frequency
11 Wp = Wc/(2*%pi); //Passband Edge Frequency
12
```

```
13 hn=eqfir(M,[0 Wp;Ws .5],[1 0],[2 1]);
14 disp(hn,'The LPF Filter Coefficients are:')
15 //Obtaining Polyphase Filter Coefficients from hn
16 p = zeros(D,M/D);
17 for k = 1:D
18     for n = 1:(length(hn)/D)
19         p(k,n) = hn(D*(n-1)+k);
20     end
21 end
22 disp(D,'For the given Decimation factor = ')
23 disp(p,'The Polyphase Decimator are:')
```

---

## Experiment: 10

# Spectral estimation of a sequence using Periodogram method.

Scilab code Solution 10.0 Spectral estimation of a sequence using Periodogram method

```
1 //Experiment – 10
2 //Spectral estimation of a sequence using
   Periodogram method.
3
4 clear;
5 clc;
6 close;
7 x=input('Input the signal x(n) = ');// x=[0.1 2.6
   -1.1 3.2 2 -8 1.2 6]
8 N =length(x);
9 X = dft(x,-1);
10 Pxx = (1/N)*(abs(X).^2); //Periodogram Estimate
11 disp(X, 'DFT of x(n) is X(k)=')
12 disp(Pxx, 'Peridogram of x(n) is Pxx(k/N)=')
13 figure(1)
14 a = gca();
```

```
15 a.data_bounds = [0,0;8,11];
16 plot2d3('gmn',[1:N],Pxx)
17 a.foreground = 5;
18 a.font_color = 5;
19 a.font_style = 5;
20 title('Peridogram Estimate')
21 xlabel('Discrete Frequency Variable K ——>')
22 ylabel('Periodogram Pxx (k /N) ——>')
```

---

## Experiment: 11

# Real-time data acquisition and plotting through external hardware interfaced through serial port.

**Scilab code Solution 11.0** Real time Data acquisition and plotting through external hardware interfaced through serial port

```
1 //Experiment – 11
2 //Real time Data acquisition and plotting through
   external hardware interfaced through serial port.
3
4 // This code requires serial toolbox installed on
   scilab.
5 // The code will execute only when Data acquisition
   hardware is connected on serial port(COM) of
   computer.
6
7 clear;
8 clc;
```

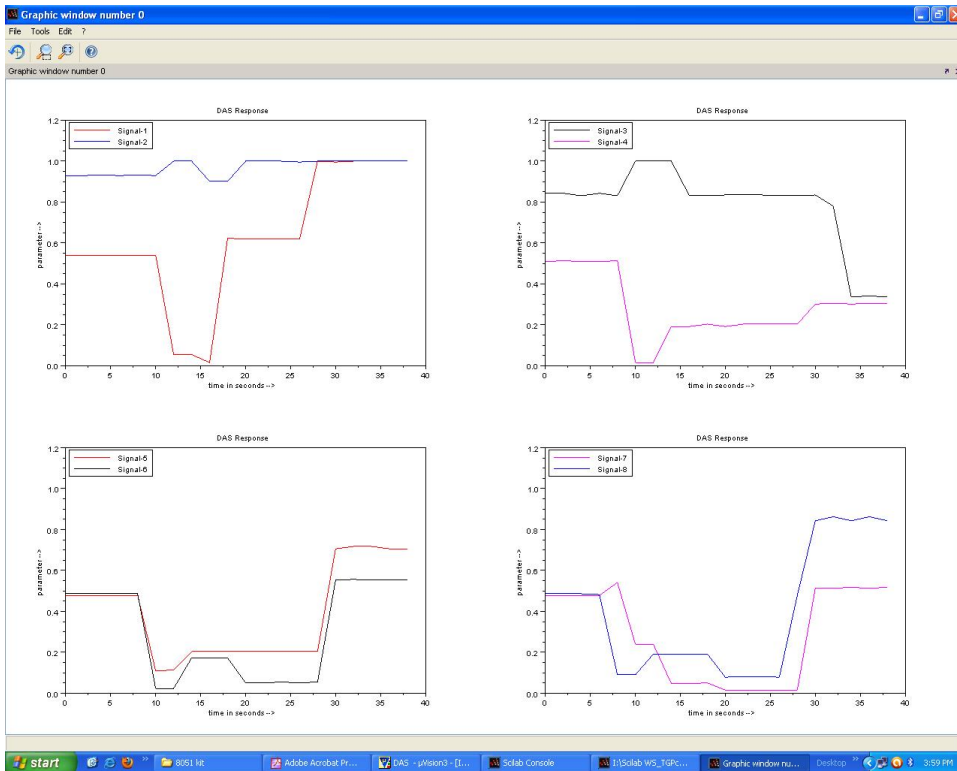


Figure 11.1: Real time Data acquisition and plotting through external hardware interfaced through serial port

```

 9  close;
10
11  //s = openserial(1,"9600,n,8,1","binary","dtrdsr");
12  s = openserial(1,"9600,n,8,1","binary","none");
13  ns=input('Samples to be plotted: ');// 50
14  ts=input('Sampling time(sec): ');//2 sec
15  xpause(20000);
16
17  signal1=zeros(1,ns);
18  signal2=zeros(1,ns);
19  signal3=zeros(1,ns);
20  signal4=zeros(1,ns);
21  signal5=zeros(1,ns);
22  signal6=zeros(1,ns);
23  signal7=zeros(1,ns);
24  signal8=zeros(1,ns);
25
26  t=0:1*ts:ts*ns-1;
27  j=0;
28  clf()
29
30      result = writeserial(s,"OK");//
31      buf = readserial(s, [17]);
32      xpause(200000);
33
34  for i=1:1:ns,
35      result = writeserial(s,"OK");//
36      // xpause(6000);//66941usec after tx will begin
37      buf = readserial(s, [17]);
38      rbuf = str2code(buf);
39
40      signal1(i)=abs((1/255)*(rbuf(2)-100));
41      signal2(i)=abs((1/255)*(rbuf(4)-100));
42      subplot(2,2,1)
43      plot(t(i-j:i),signal1(i-j:i),'-r',t(i-j:i),
44           signal2(i-j:i),'-b')
45      a=gca();
46      a.data_bounds=[0 -0.1;ts*ns-1 1.2];

```

```

46     h = legend('Signal-1', 'Signal-2', 2);
47     xlabel('time in seconds —>')
48     ylabel('parameter —>')
49     title('DAS Response')
50     set(gca(), "auto_clear", "off")
51
52     signal3(i)=abs((1/255)*(rbuf(6)-100));
53     signal4(i)=abs((1/255)*(rbuf(8)-100));
54     subplot(2,2,2)
55     plot(t(i-j:i), signal3(i-j:i), '-k', t(i-j:i),
56         signal4(i-j:i), '-m')
57     a=gca();
58     a.data_bounds=[0 -0.1;ts*ns-1 1.2];
59     h = legend('Signal-3', 'Signal-4', 2);
60     xlabel('time in seconds —>')
61     ylabel('parameter —>')
62     title('DAS Response')
63     set(gca(), "auto_clear", "off")
64
65     signal5(i)=abs((1/255)*(rbuf(10)-100));
66     signal6(i)=abs((1/255)*(rbuf(12)-100));
67     subplot(2,2,3)
68     plot(t(i-j:i), signal5(i-j:i), '-r', t(i-j:i),
69         signal6(i-j:i), '-k')
70     a=gca();
71     a.data_bounds=[0 -0.1;ts*ns-1 1.2];
72     h = legend('Signal-5', 'Signal-6', 2);
73     xlabel('time in seconds —>')
74     ylabel('parameter —>')
75     title('DAS Response')
76     set(gca(), "auto_clear", "off")
77
78     signal7(i)=abs((1/255)*(rbuf(14)-100));
79     signal8(i)=abs((1/255)*(rbuf(16)-100));
80     subplot(2,2,4)
81     plot(t(i-j:i), signal7(i-j:i), '-m', t(i-j:i),
82         signal8(i-j:i), '-b')
83     a=gca();

```

```
81     a.data_bounds=[0 -0.1;ts*ns-1 1.2];
82     h = legend('Signal-7','Signal-8',2);
83     xlabel('time in seconds —>')
84     ylabel('parameter —>')
85     title('DAS Response')
86     set(gca(),"auto_clear","off")
87     sleep(ts*1000); //wait for ts sec.
88     j=j+1;
89     clear buf
90 end
91 closeserial(s);
92 clear all
93 clc
94 disp("Required Data Plotted")
```

---