Scilab Manual for
Digital Signal Processing
by Mr Kaustubh Shivaji Sagale
Electronics Engineering
S.S.V.P.S's B.S.D. C.O.E.[1]

Solutions provided by
Mr R. Senthilkumar
Electrical Engineering
IRTT

July 16, 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

Experiment: 1

Basic operations on sequences of equal and unequal lengths.

Scilab code Solution 1.1 Program for addition of two sequences of unequal lengths

```
//Caption:Program for addition of two sequences of unequal lengths
clc;
clear;
close;
x = input('Enter Seq.=');
y = input('Enter Seq.=');
m = length(x);
n = length(y);
if m>n then
    y = [y,zeros(1,m-n)];
elseif n>m
    x = [x,zeros(1,n-m)];
end
z = x+y;
disp(z,'Addition result of two unequal length sequences:=');
//Example
```

```
//Enter Seq.=[1,2,3]
//
//Enter Seq.=[1,1,1,1]
//
// Addition result of two unequal length sequences:=
//
//     2.     3.     4.     1.
```

Scilab code Solution 1.2 Program for multiplication of two sequences of unequal lengths

```
//Caption:Program for multiplication of two
   sequences of unequal lengths
clc;
clear;
close;
x = input('Enter Seq.=');
y = input('Enter Seq.=');
m = length(x);
n = length(y);
if m>n then
    y = [y,zeros(1,m-n)];
elseif n>m
    x = [x,zeros(1,n-m)];
end
z = x.*y;
disp(z,'Multiplication result of two unequal length
   sequences:=');
//Example
//Enter Seq.=[1,2,3]
//
//Enter Seq.=[1,1,1,1]
//
// Multiplication result of two unequal length
   sequences:=
```

```
//
//      1.      2.      3.      0.
```

Scilab code Solution 1.3 Program to Demonstrate the signal Folding

```
//Caption: Program to Demonstrate the signal Folding
clc;
clear;
x = input('Enter the input sequence:=');
m = length(x);
lx = input('Enter the starting point of original
    signal=');
hx = lx+m-1;
n = lx:1:hx;
subplot(2,1,1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-5,0;5,5];
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude-->')
title('Original Sequence')
subplot(2,1,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-5,0;5,5];
plot2d3(-n,x)
xlabel('n===>')
ylabel('Amplitude-->')
title('Folded Sequence')
```

Scilab code Solution 1.4 Program to demonstrate the Amplitude and Time Scaling of a signal

```
//Caption: Program to demonstrate the Amplitude & Time Scaling of a signal
clc;
clear;
x = input('Enter input Sequence:=');
m = length(x);
lx = input('Enter starting point of original signal:=')
hx = lx+m-1;
n = lx:1:hx;
subplot(2,2,1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x)
xlabel('n====>')
ylabel('Amplitude---->')
title('orginal sequence')
//Amplitude Scaling
a = input('Amplitude Scaling Factor:=')
y =a*x;
subplot(2,2,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,y)
xlabel('n====>')
ylabel('Amplitdue---->')
title('Amplitude Scaled Sequence')
//Time Scaling-Compression
C = input('Enter Compression factor-Time Scaling factor')
n = lx/C:1/C:hx/C;
```

```
subplot(2,2,3)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude--->')
title('Compressed Sequence')
//Time Scaling-Expansion
d = input('Enter Extension factor-Time Scaling factor')
n = lx*d:d:hx*d;
subplot(2,2,4)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude--->')
title('Extended Sequence')
//Example
//Enter input Sequence:=[1,2,3,4,5]
//
//Enter starting point of original signal:= 2
//
//Amplitude Scaling Factor:= 2
//
//Enter Compression factor-Time Scaling factor 2
//
//Enter Extension factor-Time Scaling factor 2
```

Scilab code Solution 1.5 Program to demonstrate the shifting of the discrete time signal

```
//Caption:Program to demonstrate the shifting of the discrete time signal
clc;
clear;
close;
x = input('Enter the input sequence:=')
m = length(x);
lx = input('Enter the starting point of original signal:=')
hx = lx+m-1;
n = lx:1:hx;
subplot(3,1,1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x);
xlabel('n===>')
ylabel('Amplitdue---->')
title('Original Sequence')
//
d = input('Enter the delay:=')
n = lx+d:1:hx+d;
subplot(3,1,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude---->')
title('Delayed Sequence')
//
a = input('Enter the advance:=')
n = lx-a:1:hx-a;
subplot(3,1,3)
a = gca();
a.x_location = "origin";
```

```
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude---->')
title('Advanced Sequence')
//Example
//Enter the input sequence:=[1,2,3,4,5]
//
//Enter the starting point of original signal:=0
//
//Enter the delay:=2
//
//Enter the advance:=3
```

Scilab code Solution 1.6 Program to Generate Sine Waveform for one cycle

```
//Caption: Program to Generate Sine Waveform for one cycle
clc;
clear;
close;
t = 0:0.01:1;
f = 1;//frequency = 1Hz (i.e) one cycle
y = sin(2*%pi*f*t);
plot(2*%pi*t,y)
xlabel('time---->')
ylabel('Amplitude----->')
title('Sinusoidal Waveform')
xgrid(1)
```

Experiment: 2

Sampling of continuous time signal and aliasing effect.

Scilab code Solution 2.1 Program to demonstrate the Aliasing Effect

```
//Caption: Program to demonstrate the Aliasing Effect
clc;
clear;
close;
f = input('Enter the frequency of Continuous Time Signal:=')
t = 0:0.00001:1/f;
xt = 3*sin(2*%pi*f*t);
subplot(2,2,1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot(t,xt)
title('Continuous Time Signal')
fs = input('Enter the Sampling frequency Fs=');//fs = 3000Hz
fd = f/fs;
n = 0:0.01:1/fd;
```

```
xn =  3*sin(2*%pi*fd*n);
subplot(2,2,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,xn)
title('Discrete Time Signal')
x1 = 3300;
x1n = 3*sin(2*%pi*(x1/fs)*n);
subplot(2,2,3)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x1n)
title('Samples of 3300 Hz signal with fs = 3000 Hz')
x2 = 30300;
x2n = 3*sin(2*%pi*(x2/fs)*n);
subplot(2,2,4)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x2n)
title('Samples of 30300 Hz signal with fs = 3000 Hz')
//Example
//Enter the frequency of Continuous Time Signal:= 1500
//
//Enter the Sampling frequency Fs= 3000
```

Experiment: 3

Convolution of two sequence Impulse response.

Scilab code Solution 3.1 Program to Compute the Convolution of Two Sequences

```
//Caption: Program to Compute the Convolution of Two
    Sequences
clc;
clear;
close;
x = input('Enter the input Sequence:=');
m = length(x);
lx = input('Enter the lower index of input sequence
   :=')
hx = lx+m-1;
n = lx:1:hx;
h = input('Enter impulse response sequence:=')
l = length(h);
lh = input('Enter the lower index of impulse
   response:=')
hh = lh+l-1;
g = lh:1:hh;
nx = lx+lh;
```

```
nh = nx+m+l-2;
y = convol(x,h)
r = nx:nh;
subplot(3,1,1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude--->')
title('Input Sequence x[n]')
subplot(3,1,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',g,h)
xlabel('n===>')
ylabel('Amplitude---->')
title('Impulse Response Sequence h[n]=')
subplot(3,1,3)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',r,y)
xlabel('n===>')
ylabel('Amplitude---->')
title('Output Response Sequence y[n]=')
//Example
//Enter the input Sequence:=[1,2,3,1]
//
//Enter the lower index of input sequence:=0
//
//Enter impulse response sequence:=[1,2,1,-1]
//
//Enter the lower index of impulse response:=-1
//
//
//-->y
```

```
// y =
//
//      1.    4.    8.    8.    3.  - 2.  - 1.
//
```

Experiment: 4

Spectrum of signals using DFT.

Scilab code Solution 4.1 Program to find the spectral information of discrete time signal

```
//Caption: Program to find the spectral information
   of discrete time  signal
clc;
close;
clear;
xn = input('Enter the real input discrete sequence x
   [n]=');
N = length(xn);
XK = zeros(1,N);
IXK = zeros(1,N);
//Code block to find the DFT of the Sequence
for K = 0:N-1
    for n = 0:N-1
        XK(K+1) = XK(K+1)+xn(n+1)*exp(-%i*2*%pi*K*n/
           N);
    end
end
[phase,db] = phasemag(XK)
disp(XK,'Discrete Fourier Transform X(k)=')
disp(abs(XK),'Magnitude Spectral Samples=')
```

```
disp(phase,'Phase Spectral Samples=')
n = 0:N-1;
K = 0:N-1;
subplot(2,2,1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,xn)
xlabel('Time Index n------->')
ylabel('Amplitude xn------->')
title('Discrete Input Sequence')
subplot(2,2,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',K,abs(XK))
xlabel('Frequency Sample Index K------->')
ylabel('|X(K)|------->')
title('Magnitude Spectrum')
subplot(2,2,3)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',K,phase)
xlabel('Frequency Sample Index K------->')
ylabel('<X(K) in radians------->')
title('Phase Spectrum')
//Code block to find the IDFT of the sequence
for n = 0:N-1
    for K = 0:N-1
        IXK(n+1) = IXK(n+1)+XK(K+1)*exp(%i*2*%pi*K*n
           /N);
    end
end
IXK = IXK/N;
ixn = real(IXK);
subplot(2,2,4)
a = gca();
```

```
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',[0:N-1],ixn)
xlabel('Discrete Time Index n --->')
ylabel('Amplitude x[n]---->')
title('IDFT sequence')
//Example
//
//Enter the real input discrete sequence x[n]=[1,2,3,4]
//
// Discrete Fourier Transform X(k)=
//
//     10.   - 2. + 2.i   - 2. - 9.797D-16i   - 2. - 2.i
//
// Magnitude Spectral Samples=
//
//     10.    2.8284271    2.    2.8284271
//
// Phase Spectral Samples=
//
//     0.    135.    180.    225.
//
```

Experiment: 5

Designing of FIR Filter.

Scilab code Solution 5.1 To Design an Low Pass FIR Filter

```
//Caption: To Design an Low Pass FIR Filter
//Filter Length =5, Order = 4
//Window = Rectangular Window
//Ana
clc;
clear;
xdel(winsid());
fc = input("Enter Analog cutoff freq. in Hz=")
fs = input("Enter Analog sampling freq. in Hz=")
M = input("Enter order of filter =")
w = (2*%pi)*(fc/fs);
disp(w,'Digital cutoff frequency in radians.cycles/samples');
wc = w/%pi;
disp(wc,'Normalized digital cutoff frequency in cycles/samples');
[wft,wfm,fr]=wfir('lp',M+1,[wc/2,0],'re',[0,0]);
disp(wft,'Impulse Response of LPF FIR Filter:h[n]=');
//Plotting the Magnitude Response of LPF FIR Filter
subplot(2,1,1)
```

```
plot(2*fr,wfm)
xlabel('Normalized Digital Frequency w——>')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of FIR LPF')
xgrid(1)
subplot(2,1,2)
plot(fr*fs,wfm)
xlabel('Analog Frequency in Hz f ——>')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of FIR LPF')
xgrid(1)
//Example
//Enter Analog cutoff freq. in Hz= 250
//
//Enter Analog sampling freq. in Hz= 2000
//
//Enter order of filter = 4
//
// Digital cutoff frequency in radians.cycles/
   samples
//
//    0.7853982
//
// Normalized digital cutoff frequency in cycles/
   samples
//
//    0.25
//
// Impulse Response of LPF FIR Filter:h[n]=
//
//    0.1591549    0.2250791    0.25    0.2250791
      0.1591549
```

Experiment: 6

Designing of IIR Filter.

Scilab code Solution 6.1 To Design Digital IIR Butterworth LPF

```
//Caption: To Design Digital IIR Butterworth LPF
//Analog cutoff freq = 1000 Hz, Sampling Freq = 10000 samples/sec
//Order of IIR filter N = 2
clc;
clear;
xdel(winsid());
fc = input('Enter cutoff freq in Hz fc =')
fs = input('Enter sampling freq in Hz fs =')
N = input('Enter order of Butterworth filterN =')
Fp = 2*fc/fs; //Pass band edge frequency in cycles/samples
[Hz]=iir(N,'lp','butt',[Fp/2,0],[0,0]) //digital IIR Butterworth Filter
[Hw,w] = frmag(Hz,256);
subplot(2,1,1)
plot(2*w,abs(Hw));
xlabel('Normalized Digital Frequency w--->')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of IIR LPF')
xgrid(1)
```

```
subplot(2,1,2)
plot(2*w*fs,abs(Hw));
xlabel('Analog Frequency in Hz f -->')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of IIR LPF')
xgrid(1)
//Example
//
//Enter cutoff freq in Hz fc =1000
//
//Enter sampling freq in Hz fs =10000
//
//Enter order of Butterworth filterN = 2
// -->Hz
// Hz  =
//
//                                            2
//      0.0674553 + 0.1349105z + 0.0674553z
//      -----------------------------------
//                                       2
//          0.4128016 - 1.1429805z + z
```