Scilab Manual for
Digital Signal Processing Lab-1
by Prof Jeevan Reddy Koya
Electronics Engineering
Sreenidhi Institute Of Science And Technology[1]

Solutions provided by
Mr Sai Sugun L
Electronics Engineering
Sreenidihi Institute Of Science & Technology

August 2, 2025

[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

Power spectral density estimation using n-point dft

Scilab code Solution 1.1 1

```

//Power spectrum evaluation of a discrete sequence Using N-point DFT
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

N =16;  //Number of samples in given sequence
n =0:N-1;
delta_f = [0.06,0.01];//frequency separation
x1 = sin(2*%pi*0.315*n)+cos(2*%pi*(0.315+delta_f(1))*n);
```

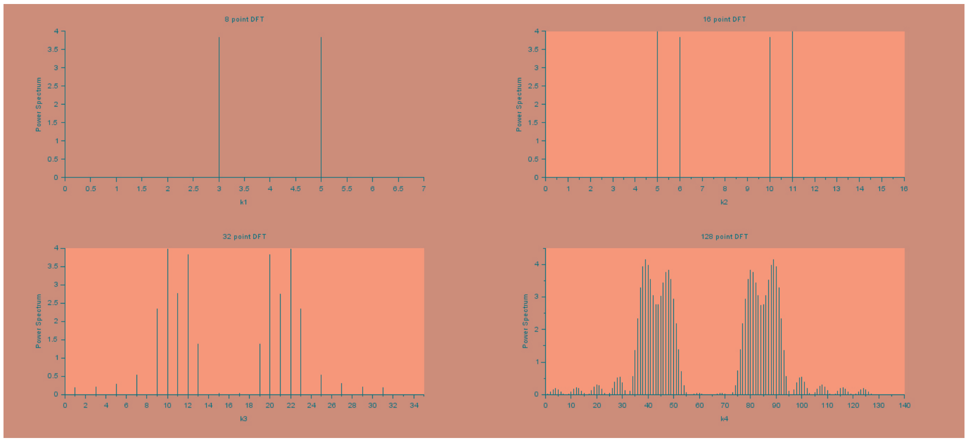

Figure 1.1: 1

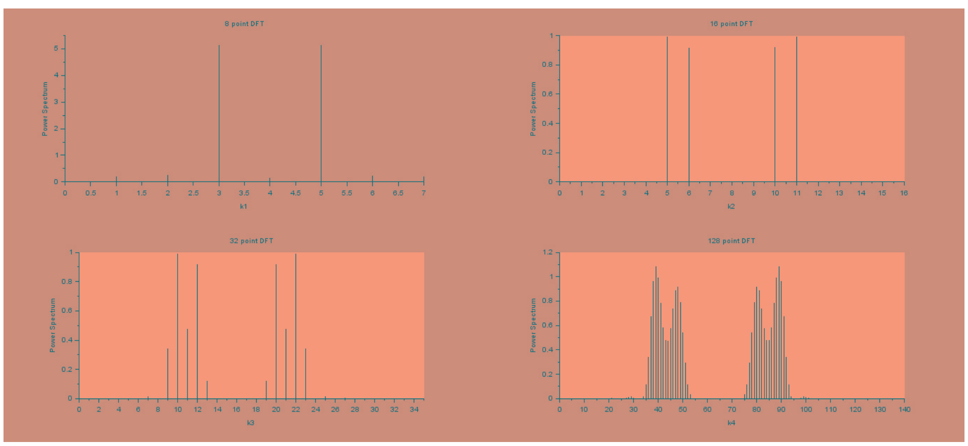

Figure 1.2: 1

```
x2 = sin(2*%pi*0.315*n)+cos(2*%pi*(0.315+delta_f(2))*n);
L = [8,16,32,128];
k1 = 0:L(1)-1;
k2 = 0:L(2)-1;
k3 = 0:L(3)-1;
k4 = 0:L(4)-1;
fk1 = k1./L(1);
fk2 = k2./L(2);
fk3 = k3./L(3);
fk4 = k4./L(4);
for i =1:length(fk1)
  Pxx1_fk1(i) = 0;
  Pxx2_fk1(i) = 0;
  for m = 1:N
    Pxx1_fk1(i)=Pxx1_fk1(i)+x1(m)*exp(-sqrt(-1)*2*%pi*(m-1)*fk1(i));
    Pxx2_fk1(i)=Pxx1_fk1(i)+x1(m)*exp(-sqrt(-1)*2*%pi*(m-1)*fk1(i));
  end
  Pxx1_fk1(i) = (Pxx1_fk1(i)^2)/N;
  Pxx2_fk1(i) = (Pxx2_fk1(i)^2)/N;
end
for i =1:length(fk2)
  Pxx1_fk2(i) = 0;
  Pxx2_fk2(i) = 0;
  for m = 1:N
    Pxx1_fk2(i)=Pxx1_fk2(i)+x1(m)*exp(-sqrt(-1)*2*%pi*(m-1)*fk2(i));
    Pxx2_fk2(i)=Pxx1_fk2(i)+x1(m)*exp(-sqrt(-1)*2*%pi*(m-1)*fk2(i));
  end
  Pxx1_fk2(i) = (Pxx1_fk2(i)^2)/N;
  Pxx2_fk2(i) = (Pxx1_fk2(i)^2)/N;
end
for i =1:length(fk3)
  Pxx1_fk3(i) = 0;
  Pxx2_fk3(i) = 0;
```

```
  for m = 1:N
    Pxx1_fk3(i) =Pxx1_fk3(i)+x1(m)*exp(-sqrt(-1)*2*%pi*(m-1)*fk3(i));
    Pxx2_fk3(i) =Pxx1_fk3(i)+x1(m)*exp(-sqrt(-1)*2*%pi*(m-1)*fk3(i));
  end
  Pxx1_fk3(i) = (Pxx1_fk3(i)^2)/N;
  Pxx2_fk3(i) = (Pxx1_fk3(i)^2)/N;
end
for i =1:length(fk4)
  Pxx1_fk4(i) = 0;
  Pxx2_fk4(i) = 0;
  for m = 1:N
    Pxx1_fk4(i) =Pxx1_fk4(i)+x1(m)*exp(-sqrt(-1)*2*%pi*(m-1)*fk4(i));
    Pxx2_fk4(i) =Pxx1_fk4(i)+x1(m)*exp(-sqrt(-1)*2*%pi*(m-1)*fk4(i));
  end
  Pxx1_fk4(i) = (Pxx1_fk4(i)^2)/N;
  Pxx2_fk4(i) = (Pxx1_fk4(i)^2)/N;
end

figure(1)
subplot(2,2,1)
plot2d3('gnn',k1,abs(Pxx1_fk1))
xtitle('8 point DFT')
xlabel('k1')
ylabel('Power Spectrum')
subplot(2,2,2)
plot2d3('gnn',k2,abs(Pxx1_fk2))
xtitle('16 point DFT')
xlabel('k2')
ylabel('Power Spectrum')
subplot(2,2,3)
plot2d3('gnn',k3,abs(Pxx1_fk3))
xtitle('32 point DFT')
xlabel('k3')
ylabel('Power Spectrum')
```

```
subplot(2,2,4)
plot2d3('gnn',k4,abs(Pxx1_fk4))
xtitle('128 point DFT')
xlabel('k4')
ylabel('Power Spectrum')
figure(2)
xlabel('k1')
ylabel('Power Spectrum')
subplot(2,2,1)
plot2d3('gnn',k1,abs(Pxx2_fk1))
xtitle('8 point DFT')
xlabel('k1')
ylabel('Power Spectrum')
subplot(2,2,2)
plot2d3('gnn',k2,abs(Pxx2_fk2))
xtitle('16 point DFT')
xlabel('k2')
ylabel('Power Spectrum')
subplot(2,2,3)
plot2d3('gnn',k3,abs(Pxx2_fk3))
xtitle('32 point DFT')
xlabel('k3')
ylabel('Power Spectrum')
subplot(2,2,4)
plot2d3('gnn',k4,abs(Pxx2_fk4))
xtitle('128 point DFT')
xlabel('k4')
ylabel('Power Spectrum')
```

Experiment: 2

PSD estimation via window based technique

Scilab code Solution 2.2 2

```
//Determination of power spectrum of a signal using
   window based techniques
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

//With maximum normalized frequency f = 0.1

N = 61;
cfreq = [0.1 0];
[wft,wfm,fr]=wfir('lp',N,cfreq,'re',0);
disp(wft,'Time domain filter coefficients hd(n)=');
disp(wfm,'Frequency domain filter values Hd(w)=');
WFM_dB = 20*log10(wfm);//Frequency response in dB
```

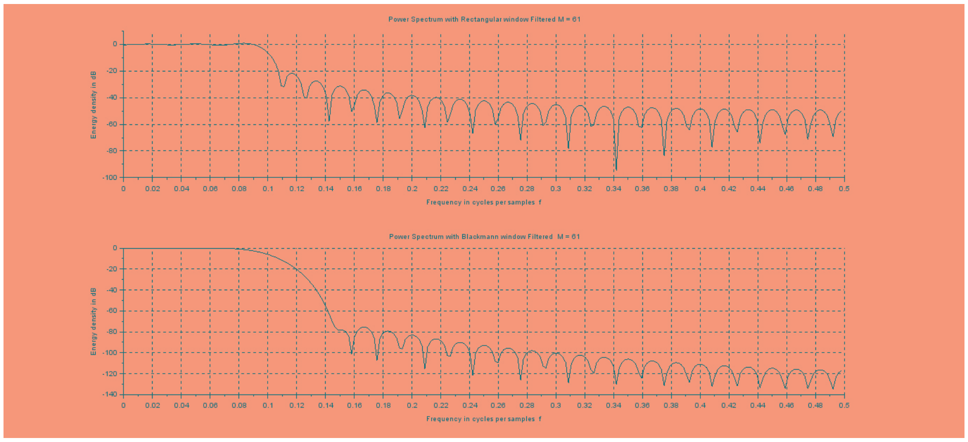

Figure 2.1: 2

```
for n = 1:N
 h_balckmann(n)=0.42-0.5*cos(2*%pi*n/(N-1))+0.08*cos
    (4*%pi*n/(N-1));
end
wft_blmn = wft'.*h_balckmann;
disp(wft_blmn,'Blackmann window based Filter output
   h(n)=')
wfm_blmn = frmag(wft_blmn,length(fr));
WFM_blmn_dB =20*log10(wfm_blmn);
subplot(2,1,1)
plot2d(fr,WFM_dB)
xgrid(1)
xtitle('Power Spectrum with Rectangular window
   Filtered M = 61','Frequency in cycles per samples
     f','Energy density in dB')
subplot(2,1,2)
plot2d(fr,WFM_blmn_dB)
xgrid(1)
xtitle('Power Spectrum with Blackmann window
   Filtered  M = 61','Frequency in cycles per
   samples  f','Energy density in dB')

//Output
```

```

// Time domain filter coefficients hd(n)=
//
//
//          column 1 to 6
//
//    0.   -0.0064517   -0.0108118   -0.0112122   -0.0071961    0.
//
//          column 7 to 11
//
//    0.0077957    0.0131622    0.0137605    0.0089094    0.
//
//          column 12 to 16
//
//   -0.0098473   -0.0168184   -0.0178077   -0.0116936    0.
//
//          column 17 to 21
//
//    0.0133641    0.023287    0.0252276    0.0170089    0.
//
//          column 22 to 26
//
//   -0.0207887   -0.0378413   -0.0432472   -0.031183    0.
//
//          column 27 to 31
//
//    0.0467745    0.1009102    0.1513653    0.1870979    0.2
//
//          column 32 to 36
//
//    0.1870979    0.1513653    0.1009102    0.0467745    0.
```

```
//
//          column 37 to 41
//
//   -0.031183   -0.0432472   -0.0378413   -0.0207887   0.
//
//          column 42 to 46
//
//    0.0170089    0.0252276    0.023287    0.0133641    0.
//
//          column 47 to 51
//
//   -0.0116936   -0.0178077   -0.0168184   -0.0098473    0.
//
//          column 52 to 56
//
//    0.0089094    0.0137605    0.0131622    0.0077957    0.
//
//          column 57 to 61
//
//   -0.0071961   -0.0112122   -0.0108118   -0.0064517    0.
//
// Frequency domain filter values Hd(w)=
//
//
//          column 1 to 4
//
//    0.9675288    0.9697174    0.9759947    0.9855327
//
//          column 5 to 8
//
//    0.9970705    1.0090769    1.0199494    1.0282222
//
//          column 9 to 12
```

```
//
//    1.0327597    1.0329081    1.0285865    1.0203056
//
//          column 13 to 16
//
//    1.0091095    0.9964485    0.9839967    0.9734374
//
//          column 17 to 20
//
//    0.9662427    0.9634759    0.9656411    0.9726015
//
//          column 21 to 24
//
//    0.9835771    0.9972258    1.0117987    1.0253531
//
//          column 25 to 28
//
//    1.0359982    1.0421436    1.0427207    1.0373466
//
//          column 29 to 32
//
//    1.0264107    1.0110667    0.9931303    0.9748874
//
//          column 33 to 36
//
//    0.9588338    0.9473719    0.9424999    0.9455261
//
//          column 37 to 40
//
//    0.9568447    0.9757996    1.0006553    1.0286838
//
//          column 41 to 44
//
//    1.0563623    1.0796646    1.0944208    1.0967084
//
//          column 45 to 49
//
//    1.083237    1.0516873    1.0009693    0.9313716
```

```
    0.844588
//
//         column 50 to 54
//
//    0.7436142    0.6325263    0.51616    0.3997228    0.2883759
//
//         column 55 to 58
//
//    0.1868241    0.0989556    0.0275603    0.0258446
//
//         column 59 to 62
//
//    0.0610721    0.0792022    0.082409    0.0737002
//
//         column 63 to 66
//
//    0.0565962    0.0347843    0.0117814    0.009362
//
//         column 67 to 70
//
//    0.0262894    0.0374811    0.0423011    0.0409531
//
//         column 71 to 74
//
//    0.0343542    0.0239471    0.0114751    0.0012542
//
//         column 75 to 78
//
//    0.0125883    0.0212119    0.0262721    0.0274398
//
//         column 79 to 82
//
//    0.0249026    0.0192981    0.0115978    0.0029604
//
//         column 83 to 87
//
//    0.0054265    0.0124965    0.017429    0.019734
```

```
    0.0192912
//
//          column 88 to 92
//
//    0.01634    0.0114257    0.0053107    0.0011358
    0.0070546
//
//          column 93 to 96
//
//    0.0117055    0.0145548    0.0153314    0.0140473
//
//          column 97 to 100
//
//    0.0109805    0.0066249    0.0016159    0.0033586
//
//          column 101 to 104
//
//    0.0076495    0.0107281    0.0122486    0.0120842
//
//          column 105 to 108
//
//    0.0103331    0.0072958    0.0034271    0.0007296
//
//          column 109 to 112
//
//    0.0046165    0.0077345    0.0097054    0.0103164
//
//          column 113 to 116
//
//    0.0095408    0.0075343    0.0046078    0.0011806
//
//          column 117 to 120
//
//    0.0022776    0.0053107    0.0075345    0.0086846
//
//          column 121 to 124
//
//    0.0086466    0.0074659    0.0053365    0.0025704
```

```
//
//          column 125 to 128
//
//    0.0004471    0.0033099    0.0056452    0.0071613
//
//          column 129 to 132
//
//    0.0076831    0.0071717    0.0057252    0.0035621
//
//          column 133 to 136
//
//    0.0009886    0.0016436    0.0039851    0.0057344
//
//          column 137 to 140
//
//    0.0066764    0.0067081    0.0058494    0.0042369
//
//          column 141 to 145
//
//    0.0021034    0.0002558    0.0025229    0.0044    0.0056481
//
//          column 146 to 150
//
//    0.0061171    0.005763    0.004652    0.0029484    0.0008913
//
//          column 151 to 154
//
//    0.0012388    0.0031582    0.0046169    0.0054315
//
//          column 155 to 158
//
//    0.0055074    0.0048499    0.003561    0.0018246
//
//          column 159 to 162
//
//    0.0001203    0.0020115    0.0035992    0.0046784
```

```
//
//          column 163 to 167
//
//    0.0051154    0.004864    0.0039702    0.002564
    0.0008414
//
//          column 168 to 171
//
//    0.0009635    0.0026096    0.0038807    0.0046144
//
//          column 172 to 175
//
//    0.004722    0.0041992    0.0031255    0.0016524
//
//          column 176 to 179
//
//    0.0000183    0.0016613    0.0030588    0.0040284
//
//          column 180 to 183
//
//    0.0044474    0.004268    0.0035224    0.0023176
//
//          column 184 to 187
//
//    0.0008202    0.0007666    0.0022306    0.0033785
//
//          column 188 to 191
//
//    0.0040614    0.0041944    0.0037665    0.0028415
//
//          column 192 to 195
//
//    0.0015484    0.0000637    0.0014125    0.0026839
//
//          column 196 to 199
//
//    0.0035833    0.0039949    0.0038691    0.0032282
//
```

```
//          column 200 to 203
//
//    0.0021631    0.0008199    0.0006196    0.0019626
//
//          column 204 to 207
//
//    0.003031    0.0036848    0.0038407    0.0034824
//
//          column 208 to 211
//
//    0.0026621    0.0014932    0.0001345    0.0012312
//
//          column 212 to 215
//
//    0.0024216    0.0032793    0.0036924    0.003609
//
//          column 216 to 219
//
//    0.0030439    0.002076    0.0008371    0.0005054
//
//          column 220 to 223
//
//    0.0017717    0.0027931    0.0034349    0.0036136
//
//          column 224 to 228
//
//    0.003308    0.0025617    0.0014769    0.0002001    0.001097
//
//          column 229 to 233
//
//    0.002241    0.0030797    0.0035026    0.003455    0.0029452
//
//          column 234 to 237
//
//    0.0020432    0.0008713    0.000413    0.0016374
//
```

```
//           column 238 to 241
//
//    0.0026386    0.0032833    0.0034864    0.0032221
//
//           column 242 to 245
//
//    0.0025269    0.0014951    0.0002654    0.0009971
//
//           column 246 to 249
//
//    0.0021236    0.0029635    0.0034049    0.0033894
//
//           column 250 to 253
//
//    0.0029197    0.0020592    0.0009236    0.0003348
//
//           column 254 to 256
//
//    0.0015476    0.0025522    0.0032144
//
// Blackmann window based Filter output h(n)=
//
//   -7.725D-21
//   -0.0000259
//   -0.0000994
//   -0.0001879
//   -0.0001942
//    3.135D-19
//    0.0004427
//    0.0010144
//    0.0013951
//    0.0011582
//   -1.272D-18
//   -0.001977
//   -0.0040862
//   -0.005155
//   -0.0039758
//    3.072D-18
```

```
//     0.0060255
//     0.0118714
//     0.0143756
//     0.0107156
//    -5.373D-18
//    -0.0155126
//    -0.0302706
//    -0.0367268
//    -0.0278468
//     7.253D-18
//     0.0449152
//     0.0991097
//     0.1506861
//     0.1870979
//     0.1991026
//     0.1837596
//     0.1453485
//     0.0938771
//     0.0417702
//     6.621D-18
//    -0.0249443
//    -0.0322712
//    -0.0260792
//    -0.0130968
//    -4.443D-18
//     0.0086709
//     0.0113744
//     0.0091754
//     0.0045438
//     2.257D-18
//    -0.0028411
//    -0.0035753
//    -0.0027431
//    -0.0012801
//    -7.904D-19
//     0.0006867
//     0.0007815
//     0.0005293
```

```
//      0.0002104
//      1.306D-19
//     -0.0000662
//     -0.000045
//     -0.0000107
//      8.953D-20
//     -7.725D-21
```

Experiment: 3

Direct Sequence Spread Spectrum(DS-BPSK)

check Appendix AP 2 for dependency:

DS_Spread_Spectrum.sci

Scilab code Solution 3.3 3

```

//Direct Sequence Spread Spectrum(DS-BPSK)
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

function[st,mt]= DS_Spread_Spectrum(bt,ct_polar)
```

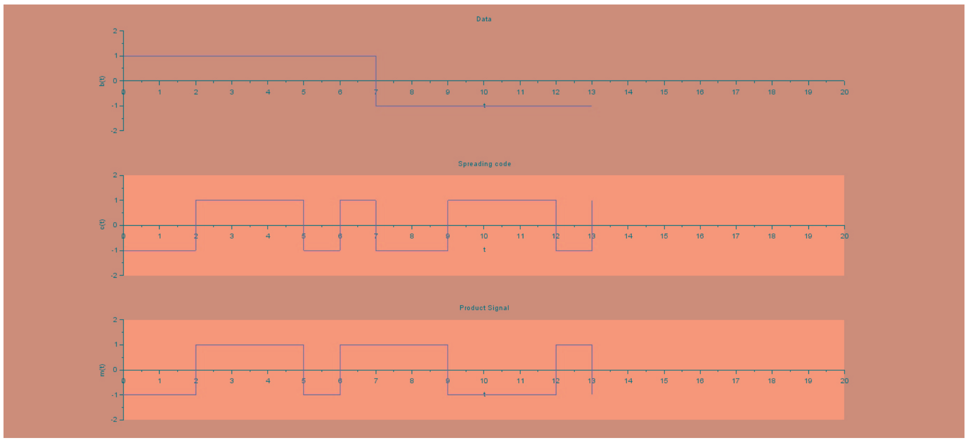

Figure 3.1: 3

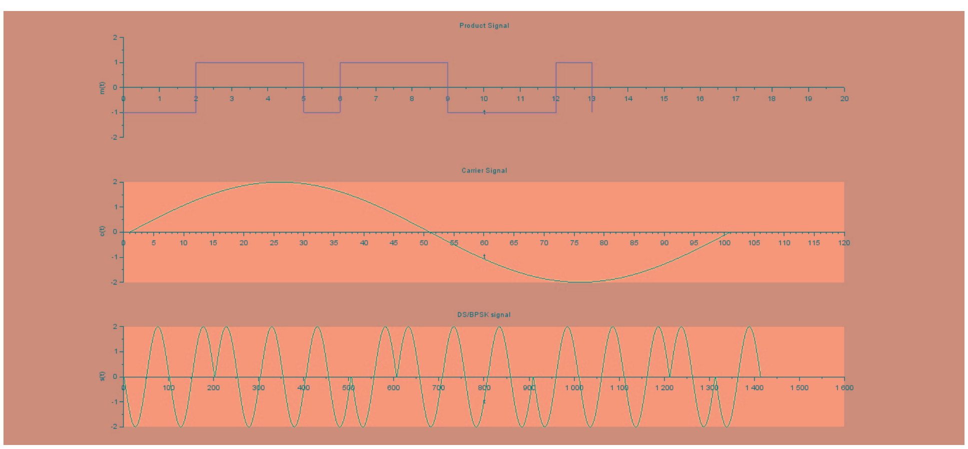

Figure 3.2: 3

```
//Generation of waveforms in DS/BPSK spread spectrum
    transmitter
//bt: Input Data Sequence (bipolar format)
//ct_polar: Spreading code (bipolar format)
Ft = 0:0.01:1;
//bt = [1*ones(1,N) -1*ones(1,N)];
t = 0:length(bt)-1;
//ct_polar = [-1,-1,1,1,1,-1,1,-1,-1,1,1,1,-1,1];
mt = bt.*ct_polar;
Carrier = 2*sin(Ft*2*%pi);
st = [];
for i = 1:length(mt)
  st = [st mt(i)*Carrier];
end

figure
subplot(3,1,1)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot2d2(t,bt,5)
xlabel('

   t')
ylabel('

   b(t)')
title('Data')
subplot(3,1,2)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot2d2(t,ct_polar,5)
xlabel('

   t')
```

```
ylabel('

    c(t)')
title('Spreading code')
subplot(3,1,3)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot2d2(t,mt,5)
xlabel('

    t')
ylabel('

    m(t)')
title('Product Signal')

figure
subplot(3,1,1)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot2d2(t,mt,5)
xlabel('

    t')
ylabel('

    m(t)')
title('Product Signal')
subplot(3,1,2)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot(Carrier)
```

```
xlabel('

    t')
ylabel('

    c(t)')
title('Carrier Signal')
subplot(3,1,3)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot(st)
xlabel('

    t')
ylabel('

    s(t)')
title('DS/BPSK signal')
endfunction

bt = [1,1,1,1,1,1,1,-1,-1,-1,-1,-1,-1,-1]
ct_polar = [-1,-1,1,1,1,-1,1,-1,-1,1,1,1,-1,1]
[st,mt]= DS_Spread_Spectrum(bt,ct_polar)
```

Experiment: 4

Constellation Diagram For Binary PSK

Scilab code Solution 4.4 4

```
//Constellation Diagram For Binary PSK
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

function[y]= Constellation_BPSK()
M =2;
i = 1:M;
y = cos(2*%pi+(i-1)*%pi);
annot = dec2bin([length(y)-1:-1:0],log2(M));
disp(y,'coordinates of message points')
disp(annot,'Message points')
figure;
a =gca();
```

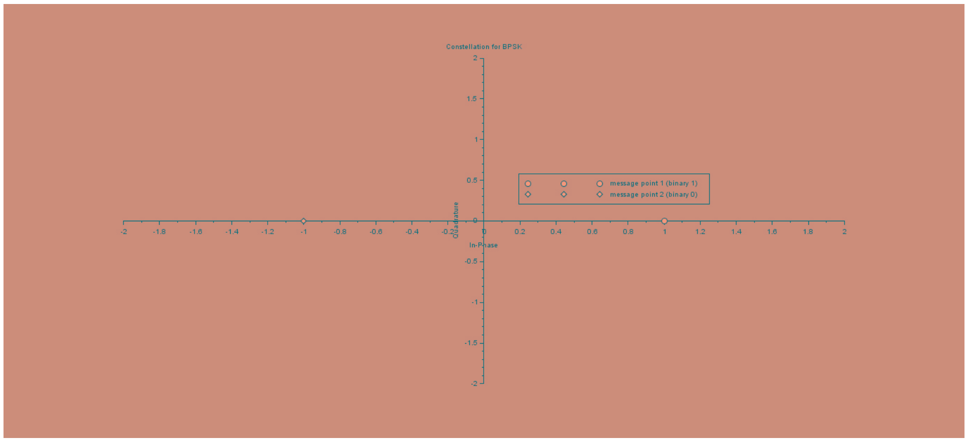

Figure 4.1: 4

```
a.data_bounds = [-2,-2;2,2];
a.x_location = "origin";
a.y_location = "origin";
plot2d(real(y(1)),imag(y(1)),-9)
plot2d(real(y(2)),imag(y(2)),-5)
xlabel('

    In-Phase');
ylabel('

    Quadrature');
title('Constellation for BPSK')
legend(['message point 1 (binary 1)';'message point
    2 (binary 0)'],5)
endfunction

Constellation_BPSK()

//Output
//coordinates of message points
//
//    1.   -1.
//
```

```
// Message points
//
//!1   0   !
```

Experiment: 5

Repetition Code

check Appendix AP 1 for dependency:

`RepetitionCode.sci`

Scilab code Solution 5.5 5

```
//Repetition Code
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

function [G,H,x]=RepetitionCode(n,k,m)
//Repetition Codes
//n =block of identical 'n' bits
//k =1 one bit
//m = 1;// bit value = 1
I = eye(n-k,n-k);//Identity matrix
P = ones(1,n-k);//coefficient matrix
H = [I P'];//parity-check matrix
G = [P 1];//generator matrix
```

```
x = m.*G; //code word
disp(G,'generator matrix');
disp(H,'parity-check matrix');
disp(x,'code word for binary one input');
endfunction

n=5;
k=1;
m=1;
[G,H,x]=RepetitionCode(n,k,m)

//Output
// generator matrix
//
//    1.    1.    1.    1.    1.
//
// parity-check matrix
//
//    1.    0.    0.    0.    1.
//    0.    1.    0.    0.    1.
//    0.    0.    1.    0.    1.
//    0.    0.    0.    1.    1.
//
// code word for binary one input
//
//    1.    1.    1.    1.    1.
```

Experiment: 6

Continuous Time Fourier Series of Sine Signal

Scilab code Solution 6.6 6

```
//Continuous Time Fourier Series of Sine Signal
//OS:Windows 10
//Scilab 5.5.2

clear;
clc;
close;

//periodic sine signal x(t) = sin(Wot)

t = 0:0.01:1;
T = 1;
Wo = 2*%pi/T;
xt = sin(Wo*t);
for k =0:5
  C(k+1,:) = exp(-sqrt(-1)*Wo*t.*k);
  a(k+1) = xt*C(k+1,:)'/length(t); //fourier series
     is done
  if(abs(a(k+1))<=0.01)
```

```
    a(k+1)=0;
  end
end
a =a';
ak = [-a($:-1:1),a(2:$)];
disp(ak,'Continuous Time Fourier Series Coefficients
    are:')

//Output
// Continuous Time Fourier Series Coefficients are:
//
//
//          column 1 to 9
//
//    0.    0.    0.    0.    0.4950495i    0.    -0.4950495
   i    0.    0.
//
//          column 10 to 11
//
//    0.    0.
```

Experiment: 7

Spectrum of Signal (Frequency Response)-Blackmann Window

Scilab code Solution 7.7 7

```
//SPECTRUM OF SIGNAL (FREQUENCY RESPONSE)- BLACKMANN
    WINDOW
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

//With maximum normalized frequency f = 0.4

N = 11;
cfreq = [0.4 0];
[wft,wfm,fr]=wfir('lp',N,cfreq,'re',0);
wft;                    // Time domain filter
   coefficients
wfm;                    // Frequency domain filter
```

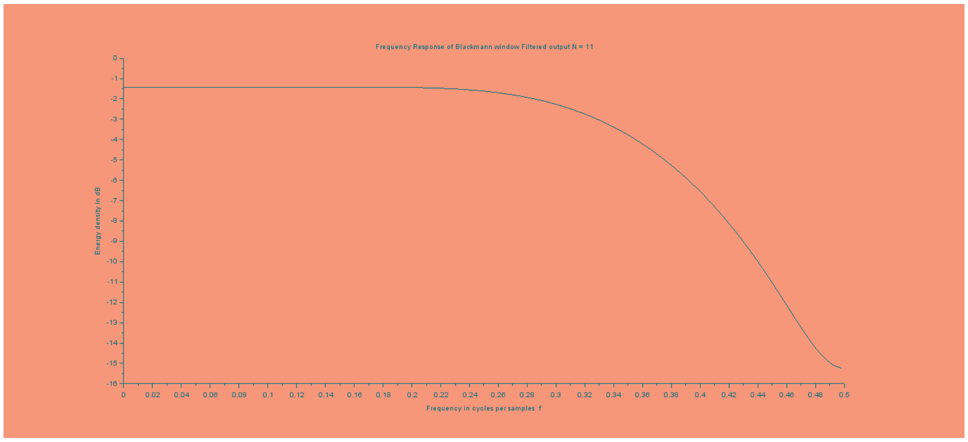

Figure 7.1: 7

```
   values
fr;                    // Frequency sample points
for n = 1:N
  h_blackmann(n)=0.42-0.5*cos(2*%pi*n/(N-1))+0.08*
     cos(4*%pi*n/(N-1));
  wft_blmn(n) = wft(n)*h_blackmann(n);
end
wfm_blmn = frmag(wft_blmn,length(fr));
WFM_blmn_dB =20*log10(wfm_blmn);
plot2d(fr,WFM_blmn_dB)
xtitle('Frequency Response of Blackmann window
   Filtered output N = 11','Frequency in cycles per
   samples  f','Energy density in dB')
```

Experiment: 8

Comparision Of Different Power Spectrum Estimates

Scilab code Solution 8.8 8

```
//COMPARISON OF DIFFERENT POWER SPECTRUM ESTIMATES
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

Q = 10;  //Quality factor
N = 1000; //Length of the sample sequence
//Bartlett Method
F_Bartlett = Q/(1.11*N);
disp(F_Bartlett,'Frequency Resolution of Bartlett
   Power Spectrum Estimation')
//Welch Method
F_Welch = Q/(1.39*N);
disp(F_Welch,'Frequency Resolution of Welch Power
   Spectrum Estimation')
//Blackmann-Tukey Method
```

```
F_Blackmann_Tukey  = Q/(2.34*N);
disp(F_Blackmann_Tukey,'Frequency Resolution of Blackmann Tukey Power Spectrum Estimation')


//Output
// Frequency Resolution of Bartlett Power Spectrum Estimation
//
//    0.009009
//
// Frequency Resolution of Welch Power Spectrum Estimation
//
//    0.0071942
//
// Frequency Resolution of Blackmann Tukey Power Spectrum Esti
// mation
//    0.0042735
```

Experiment: 9

Continuous Time Fourier Transform Of An Exponential Signal

Scilab code Solution 9.9 9

```
//CONTINUOUS TIME FOURIER TRANSFORM OF A EXPONENTIAL SIGNAL
//OS:Windows 10
//Scilab 5.5.2

clear all;
clc;
close;

//Continuous Time Exponential Signal x(t)= exp(-A*t)u(t), A>0

// Analog Signal
```

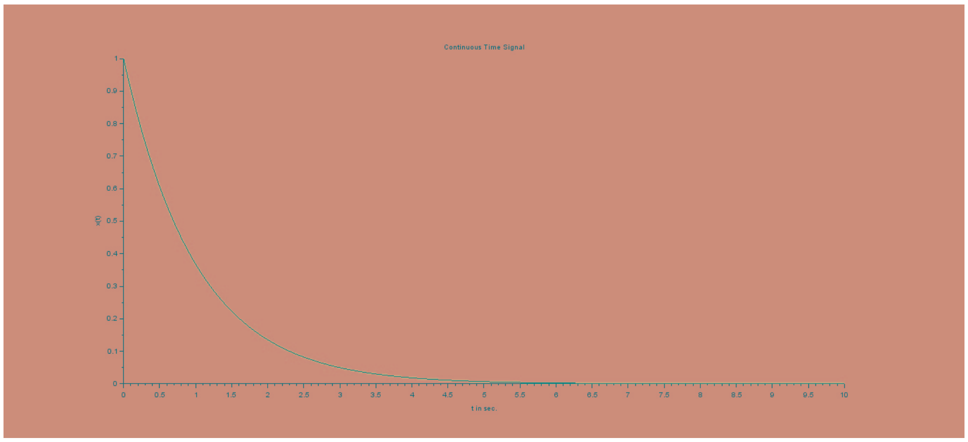

Figure 9.1: 9

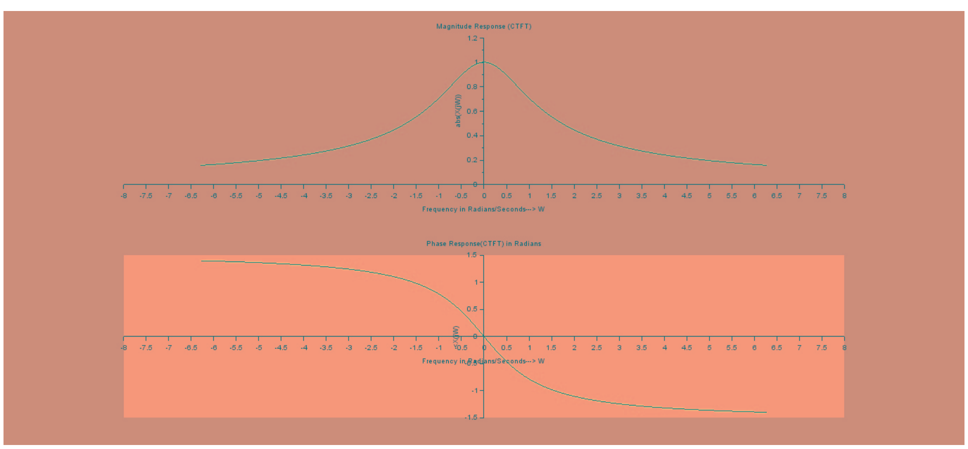

Figure 9.2: 9

```
A =1;     //Amplitude
Dt = 0.005;
t = 0:Dt:10;
xt = exp(-A*t);

// Continuous-time Fourier Transform
Wmax = 2*%pi*1;         //Analog Frequency = 1Hz
K = 4;
k = 0:(K/1000):K;
W = k*Wmax/K;
XW = xt* exp(-sqrt(-1)*t'*W) * Dt;
XW_Mag = abs(XW);
W = [-mtlb_fliplr(W), W(2:1001)]; // Omega from -
    Wmax to Wmax
XW_Mag = [mtlb_fliplr(XW_Mag), XW_Mag(2:1001)];
[XW_Phase,db] = phasemag(XW);
XW_Phase = [-mtlb_fliplr(XW_Phase),XW_Phase(2:1001)
    ];

//Plotting Continuous Time Signal
figure
a = gca();
a.y_location = "origin";
plot(t,xt);
xlabel('t in sec.');
ylabel('x(t)')
title('Continuous Time Signal')

//Plotting Magnitude Response of CTS
figure
subplot(2,1,1);
a = gca();
a.y_location = "origin";
plot(W,XW_Mag);
xlabel('Frequency in Radians/Seconds---> W');
ylabel('abs(X(jW))')
title('Magnitude Response (CTFT)')

```

```
//Plotting Phase Reponse of CTS
subplot(2,1,2);
a = gca();
a.y_location = "origin";
a.x_location = "origin";
plot(W,XW_Phase*%pi/180);
xlabel('                              Frequency in Radians/Seconds---> W');
ylabel('                                                            <X(jW)')
title('Phase Response(CTFT) in Radians')
```

Experiment: 10

FIR Band Pass Filter - Remez Algorithm-LPF

Scilab code Solution 10.10 10

```
//FIR BAND PASS FILTER - REMEZ ALGORITHM -FOR LPF
//OS:Windows 10
//Scilab 5.5.2

//Band Pass Filter of length M = 16
//Lower Cutoff frequency fp = 0.2 and Upper Cutoff
    frequency fs = 0.3

clear all;
clc;
close;
hn = 0;
hm = 0;
hn=eqfir(16,[0 .1;.2 .35;.425 .5],[0 1 0],[10 1 10])
    ;//number of cosine functions,pass band magnitude
    &stop band,weighting function
[hm,fr]=frmag(hn,256);
```

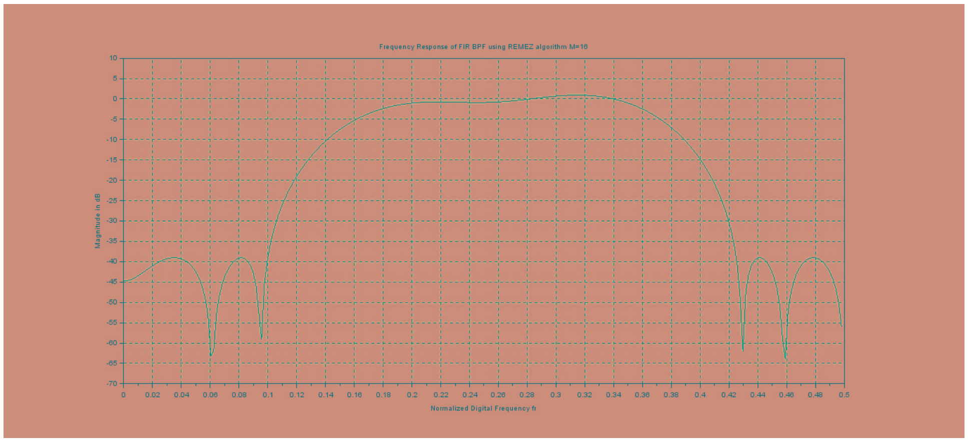

Figure 10.1: 10

```
disp(hn,'The Filter Coefficients are:')
figure
plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
a = gca();
xlabel('Normalized Digital Frequency fr');
ylabel('Magnitude in dB');
title('Frequency Response of FIR BPF using REMEZ
   algorithm M=16')
xgrid(2)

//Output
//
// The Filter Coefficients are:
//
//   -0.0395487
//    0.0232284
//    0.0480681
//   -0.0218794
//    0.0975735
//   -0.1012773
//   -0.2880134
//    0.2847346
//    0.2847346
```

```
//     -0.2880134
//     -0.1012773
//      0.0975735
//     -0.0218794
//      0.0480681
//      0.0232284
//     -0.0395487
//
```

Appendix

Scilab code AP 1

```
clear all;
clc;
close;

function [G,H,x]=RepetitionCode(n,k,m)
//Repetition Codes
//n =block of identical 'n' bits
//k =1 one bit
//m = 1;// bit value = 1
I = eye(n-k,n-k);//Identity matrix
P = ones(1,n-k);//coefficient matrix
H = [I P'];//parity-check matrix
G = [P 1];//generator matrix
x = m.*G; //code word
disp(G,'generator matrix');
disp(H,'parity-check matrix');
disp(x,'code word for binary one input');
endfunction
```

RepetitionCode

Scilab code AP 2

```
clear all;
clc;
close;

function[st,mt]= DS_Spread_Spectrum(bt,ct_polar)
//Generation of waveforms in DS/BPSK spread spectrum
```

```
      transmitter
//bt: Input Data Sequence (bipolar format)
//ct_polar: Spreading code (bipolar format)
Ft = 0:0.01:1;
//bt = [1*ones(1,N) -1*ones(1,N)];
t = 0:length(bt)-1;
//ct_polar = [-1,-1,1,1,1,-1,1,-1,-1,1,1,1,-1,1];
mt = bt.*ct_polar;
Carrier = 2*sin(Ft*2*%pi);
st = [];
for i = 1:length(mt)
  st = [st mt(i)*Carrier];
end

figure
subplot(3,1,1)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot2d2(t,bt,5)
xlabel('

    t')
ylabel('

    b(t)')
title('Data')
subplot(3,1,2)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot2d2(t,ct_polar,5)
xlabel('

    t')
ylabel('
```

```
    c(t)')
title('Spreading code')
subplot(3,1,3)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot2d2(t,mt,5)
xlabel('

    t')
ylabel('

    m(t)')
title('Product Signal')

figure
subplot(3,1,1)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot2d2(t,mt,5)
xlabel('

    t')
ylabel('

    m(t)')
title('Product Signal')
subplot(3,1,2)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot(Carrier)
xlabel('
```

```
    t')
ylabel('

    c(t)')
title('Carrier Signal')
subplot(3,1,3)
a =gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [0,-2;20,2];
plot(st)
xlabel('

    t')
ylabel('

    s(t)')
title('DS/BPSK signal')
endfunction
```

DSSS