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

November 28, 2026

[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 Scilab Solutions 3

1 Power spectral density estimation using n-point dft 5

2 PSD estimation via window based technique 10

3 Direct Sequence Spread Spectrum(DS-BPSK) 23

4 Constellation Diagram For Binary PSK 28

5 Repetition Code 31

6 Continuous Time Fourier Series of Sine Signal 33

7 Spectrum of Signal (Frequency Response)-Blackmann Window 35

8 Comparision Of Different Power Spectrum Estimates 37

9 Continuous Time Fourier Transform Of An Exponential Signal 39

10 FIR Band Pass Filter - Remez Algorithm-LPF 43

List of Experiments

Solution 1.1	1	5
Solution 2.2	2	10
Solution 3.3	3	23
Solution 4.4	4	28
Solution 5.5	5	31
Solution 6.6	6	33
Solution 7.7	7	35
Solution 8.8	8	37
Solution 9.9	9	39
Solution 10.10	10	43
AP 1	RepetitionCode	46
AP 2	DSSS	49

List of Figures

1.1 1 6
1.2 1 6

2.1 2 11

3.1 3 24
3.2 3 24

4.1 4 29

7.1 7 36

9.1 9 40
9.2 9 40

10.1 10 44

Experiment: 1

Power spectral density estimation using n-point dft

Scilab code Solution 1.1 1

```
1
2 //Power spectrum evaluation of a discrete sequence
     Using N-point DFT
3 //OS:Windows 10
4 //Scilab 5.5.2
5
6 clear all;
7 clc;
8 close;
9
10 N =16;  //Number of samples in given sequence
11 n =0:N-1;
12 delta_f = [0.06,0.01];//frequency separation
13 x1 = sin(2*%pi*0.315*n)+cos(2*%pi*(0.315+delta_f(1))
     *n);
```

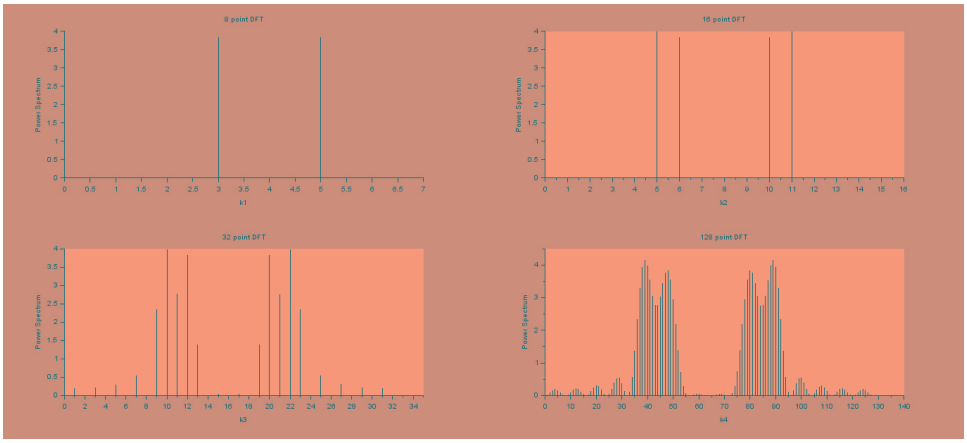

Figure 1.1: 1

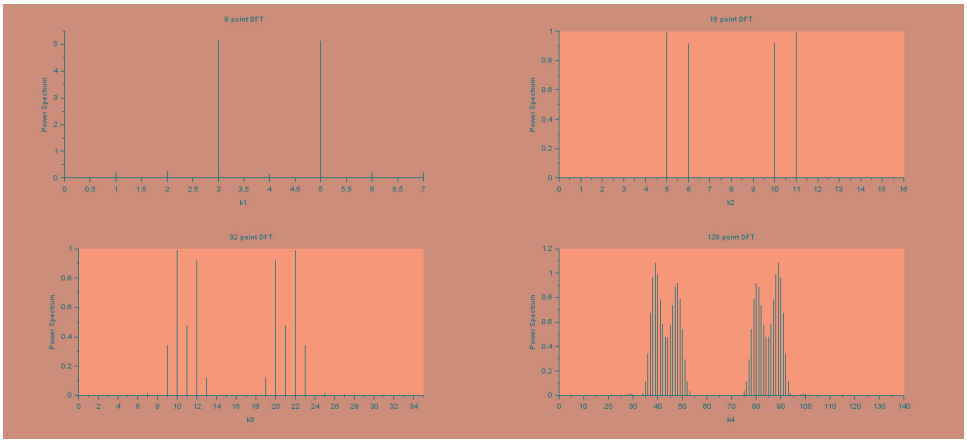

Figure 1.2: 1

```
14 x2 = sin(2*%pi*0.315*n)+cos(2*%pi*(0.315+delta_f(2))
      *n);
15 L = [8,16,32,128];
16 k1 = 0:L(1)-1;
17 k2 = 0:L(2)-1;
18 k3 = 0:L(3)-1;
19 k4 = 0:L(4)-1;
20 fk1 = k1./L(1);
21 fk2 = k2./L(2);
22 fk3 = k3./L(3);
23 fk4 = k4./L(4);
24 for i =1:length(fk1)
25   Pxx1_fk1(i) = 0;
26   Pxx2_fk1(i) = 0;
27   for m = 1:N
28     Pxx1_fk1(i)=Pxx1_fk1(i)+x1(m)*exp(-sqrt(-1)*2*
         %pi*(m-1)*fk1(i));
29     Pxx2_fk1(i)=Pxx1_fk1(i)+x1(m)*exp(-sqrt(-1)*2*
         %pi*(m-1)*fk1(i));
30   end
31   Pxx1_fk1(i) = (Pxx1_fk1(i)^2)/N;
32   Pxx2_fk1(i) = (Pxx2_fk1(i)^2)/N;
33 end
34 for i =1:length(fk2)
35   Pxx1_fk2(i) = 0;
36   Pxx2_fk2(i) = 0;
37   for m = 1:N
38     Pxx1_fk2(i)=Pxx1_fk2(i)+x1(m)*exp(-sqrt(-1)*2*
         %pi*(m-1)*fk2(i));
39     Pxx2_fk2(i)=Pxx1_fk2(i)+x1(m)*exp(-sqrt(-1)*2*
         %pi*(m-1)*fk2(i));
40   end
41   Pxx1_fk2(i) = (Pxx1_fk2(i)^2)/N;
42   Pxx2_fk2(i) = (Pxx1_fk2(i)^2)/N;
43 end
44 for i =1:length(fk3)
45   Pxx1_fk3(i) = 0;
46   Pxx2_fk3(i) = 0;
```

```
47    for m = 1:N
48      Pxx1_fk3(i) =Pxx1_fk3(i)+x1(m)*exp(-sqrt(-1)*2*
          %pi*(m-1)*fk3(i));
49      Pxx2_fk3(i) =Pxx1_fk3(i)+x1(m)*exp(-sqrt(-1)*2*
          %pi*(m-1)*fk3(i));
50    end
51    Pxx1_fk3(i) = (Pxx1_fk3(i)^2)/N;
52    Pxx2_fk3(i) = (Pxx1_fk3(i)^2)/N;
53  end
54  for i =1:length(fk4)
55    Pxx1_fk4(i) = 0;
56    Pxx2_fk4(i) = 0;
57    for m = 1:N
58      Pxx1_fk4(i) =Pxx1_fk4(i)+x1(m)*exp(-sqrt(-1)*2*
          %pi*(m-1)*fk4(i));
59      Pxx2_fk4(i) =Pxx1_fk4(i)+x1(m)*exp(-sqrt(-1)*2*
          %pi*(m-1)*fk4(i));
60    end
61    Pxx1_fk4(i) = (Pxx1_fk4(i)^2)/N;
62    Pxx2_fk4(i) = (Pxx1_fk4(i)^2)/N;
63  end
64
65  figure(1)
66  subplot(2,2,1)
67  plot2d3('gnn',k1,abs(Pxx1_fk1))
68  xtitle('8 point DFT')
69  xlabel('k1')
70  ylabel('Power Spectrum')
71  subplot(2,2,2)
72  plot2d3('gnn',k2,abs(Pxx1_fk2))
73  xtitle('16 point DFT')
74  xlabel('k2')
75  ylabel('Power Spectrum')
76  subplot(2,2,3)
77  plot2d3('gnn',k3,abs(Pxx1_fk3))
78  xtitle('32 point DFT')
79  xlabel('k3')
80  ylabel('Power Spectrum')
```

```
81  subplot(2,2,4)
82  plot2d3('gnn',k4,abs(Pxx1_fk4))
83  xtitle('128 point DFT')
84  xlabel('k4')
85  ylabel('Power Spectrum')
86  figure(2)
87  xlabel('k1')
88  ylabel('Power Spectrum')
89  subplot(2,2,1)
90  plot2d3('gnn',k1,abs(Pxx2_fk1))
91  xtitle('8 point DFT')
92  xlabel('k1')
93  ylabel('Power Spectrum')
94  subplot(2,2,2)
95  plot2d3('gnn',k2,abs(Pxx2_fk2))
96  xtitle('16 point DFT')
97  xlabel('k2')
98  ylabel('Power Spectrum')
99  subplot(2,2,3)
100 plot2d3('gnn',k3,abs(Pxx2_fk3))
101 xtitle('32 point DFT')
102 xlabel('k3')
103 ylabel('Power Spectrum')
104 subplot(2,2,4)
105 plot2d3('gnn',k4,abs(Pxx2_fk4))
106 xtitle('128 point DFT')
107 xlabel('k4')
108 ylabel('Power Spectrum')
```

Experiment: 2

PSD estimation via window based technique

Scilab code Solution 2.2 2

```
1 //Determination of power spectrum of a signal using
     window based techniques
2 //OS:Windows 10
3 //Scilab 5.5.2
4
5 clear all;
6 clc;
7 close;
8
9 //With maximum normalized frequency f = 0.1
10
11 N = 61;
12 cfreq = [0.1 0];
13 [wft,wfm,fr]=wfir('lp',N,cfreq,'re',0);
14 disp(wft,'Time domain filter coefficients hd(n)=');
15 disp(wfm,'Frequency domain filter values Hd(w)=');
16 WFM_dB = 20*log10(wfm);//Frequency response in dB
```

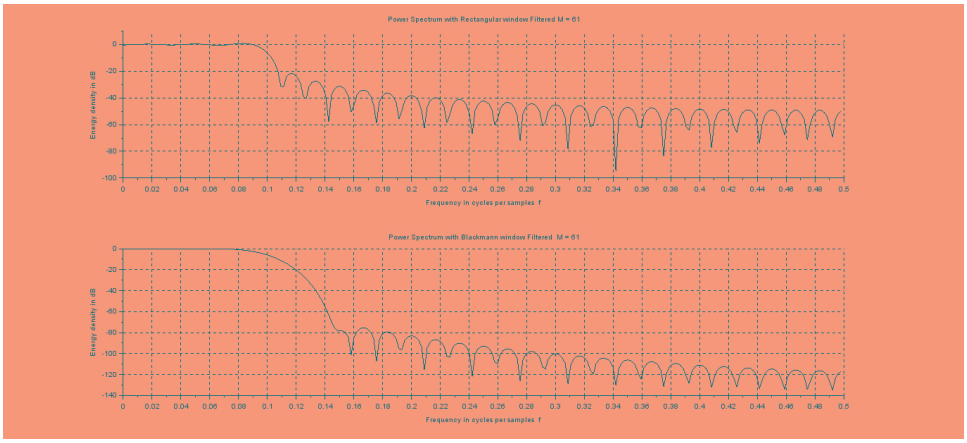

Figure 2.1: 2

```
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
```

```
34
35 // Time domain filter coefficients hd(n)=
36 //
37 //
38 //          column 1 to 6
39 //
40 //    0.   -0.0064517   -0.0108118   -0.0112122
      -0.0071961    0.
41 //
42 //          column 7 to 11
43 //
44 //    0.0077957    0.0131622    0.0137605    0.0089094
        0.
45 //
46 //          column 12 to 16
47 //
48 //   -0.0098473   -0.0168184   -0.0178077   -0.0116936
        0.
49 //
50 //          column 17 to 21
51 //
52 //    0.0133641    0.023287    0.0252276    0.0170089
      0.
53 //
54 //          column 22 to 26
55 //
56 //   -0.0207887   -0.0378413   -0.0432472   -0.031183
      0.
57 //
58 //          column 27 to 31
59 //
60 //    0.0467745    0.1009102    0.1513653    0.1870979
        0.2
61 //
62 //          column 32 to 36
63 //
64 //    0.1870979    0.1513653    0.1009102    0.0467745
        0.
```

```
65 //
66 //          column 37 to 41
67 //
68 //   -0.031183   -0.0432472   -0.0378413   -0.0207887
       0.
69 //
70 //          column 42 to 46
71 //
72 //    0.0170089    0.0252276    0.023287    0.0133641
       0.
73 //
74 //          column 47 to 51
75 //
76 //   -0.0116936   -0.0178077   -0.0168184   -0.0098473
         0.
77 //
78 //          column 52 to 56
79 //
80 //    0.0089094    0.0137605    0.0131622    0.0077957
         0.
81 //
82 //          column 57 to 61
83 //
84 //   -0.0071961   -0.0112122   -0.0108118   -0.0064517
         0.
85 //
86 // Frequency domain filter values Hd(w)=
87 //
88 //
89 //          column 1 to 4
90 //
91 //    0.9675288    0.9697174    0.9759947    0.9855327
92 //
93 //          column 5 to 8
94 //
95 //    0.9970705    1.0090769    1.0199494    1.0282222
96 //
97 //          column 9 to 12
```

```
98  //
99  //    1.0327597   1.0329081   1.0285865   1.0203056
100 //
101 //          column 13 to 16
102 //
103 //    1.0091095   0.9964485   0.9839967   0.9734374
104 //
105 //          column 17 to 20
106 //
107 //    0.9662427   0.9634759   0.9656411   0.9726015
108 //
109 //          column 21 to 24
110 //
111 //    0.9835771   0.9972258   1.0117987   1.0253531
112 //
113 //          column 25 to 28
114 //
115 //    1.0359982   1.0421436   1.0427207   1.0373466
116 //
117 //          column 29 to 32
118 //
119 //    1.0264107   1.0110667   0.9931303   0.9748874
120 //
121 //          column 33 to 36
122 //
123 //    0.9588338   0.9473719   0.9424999   0.9455261
124 //
125 //          column 37 to 40
126 //
127 //    0.9568447   0.9757996   1.0006553   1.0286838
128 //
129 //          column 41 to 44
130 //
131 //    1.0563623   1.0796646   1.0944208   1.0967084
132 //
133 //          column 45 to 49
134 //
135 //    1.083237   1.0516873   1.0009693   0.9313716
```

```
      0.844588
136 //
137 //          column 50 to 54
138 //
139 //   0.7436142   0.6325263   0.51616   0.3997228
      0.2883759
140 //
141 //          column 55 to 58
142 //
143 //   0.1868241   0.0989556   0.0275603   0.0258446
144 //
145 //          column 59 to 62
146 //
147 //   0.0610721   0.0792022   0.082409   0.0737002
148 //
149 //          column 63 to 66
150 //
151 //   0.0565962   0.0347843   0.0117814   0.009362
152 //
153 //          column 67 to 70
154 //
155 //   0.0262894   0.0374811   0.0423011   0.0409531
156 //
157 //          column 71 to 74
158 //
159 //   0.0343542   0.0239471   0.0114751   0.0012542
160 //
161 //          column 75 to 78
162 //
163 //   0.0125883   0.0212119   0.0262721   0.0274398
164 //
165 //          column 79 to 82
166 //
167 //   0.0249026   0.0192981   0.0115978   0.0029604
168 //
169 //          column 83 to 87
170 //
171 //   0.0054265   0.0124965   0.017429   0.019734
```

```
    0.0192912
172 //
173 //          column 88 to 92
174 //
175 //   0.01634   0.0114257   0.0053107   0.0011358
    0.0070546
176 //
177 //          column 93 to 96
178 //
179 //   0.0117055   0.0145548   0.0153314   0.0140473
180 //
181 //          column 97 to 100
182 //
183 //   0.0109805   0.0066249   0.0016159   0.0033586
184 //
185 //          column 101 to 104
186 //
187 //   0.0076495   0.0107281   0.0122486   0.0120842
188 //
189 //          column 105 to 108
190 //
191 //   0.0103331   0.0072958   0.0034271   0.0007296
192 //
193 //          column 109 to 112
194 //
195 //   0.0046165   0.0077345   0.0097054   0.0103164
196 //
197 //          column 113 to 116
198 //
199 //   0.0095408   0.0075343   0.0046078   0.0011806
200 //
201 //          column 117 to 120
202 //
203 //   0.0022776   0.0053107   0.0075345   0.0086846
204 //
205 //          column 121 to 124
206 //
207 //   0.0086466   0.0074659   0.0053365   0.0025704
```

```
208 //
209 //          column 125 to 128
210 //
211 //    0.0004471    0.0033099    0.0056452    0.0071613
212 //
213 //          column 129 to 132
214 //
215 //    0.0076831    0.0071717    0.0057252    0.0035621
216 //
217 //          column 133 to 136
218 //
219 //    0.0009886    0.0016436    0.0039851    0.0057344
220 //
221 //          column 137 to 140
222 //
223 //    0.0066764    0.0067081    0.0058494    0.0042369
224 //
225 //          column 141 to 145
226 //
227 //    0.0021034    0.0002558    0.0025229    0.0044
        0.0056481
228 //
229 //          column 146 to 150
230 //
231 //    0.0061171    0.005763    0.004652    0.0029484
        0.0008913
232 //
233 //          column 151 to 154
234 //
235 //    0.0012388    0.0031582    0.0046169    0.0054315
236 //
237 //          column 155 to 158
238 //
239 //    0.0055074    0.0048499    0.003561    0.0018246
240 //
241 //          column 159 to 162
242 //
243 //    0.0001203    0.0020115    0.0035992    0.0046784
```

```
244 //
245 //          column 163 to 167
246 //
247 //   0.0051154   0.004864   0.0039702   0.002564
      0.0008414
248 //
249 //          column 168 to 171
250 //
251 //   0.0009635   0.0026096   0.0038807   0.0046144
252 //
253 //          column 172 to 175
254 //
255 //   0.004722   0.0041992   0.0031255   0.0016524
256 //
257 //          column 176 to 179
258 //
259 //   0.0000183   0.0016613   0.0030588   0.0040284
260 //
261 //          column 180 to 183
262 //
263 //   0.0044474   0.004268   0.0035224   0.0023176
264 //
265 //          column 184 to 187
266 //
267 //   0.0008202   0.0007666   0.0022306   0.0033785
268 //
269 //          column 188 to 191
270 //
271 //   0.0040614   0.0041944   0.0037665   0.0028415
272 //
273 //          column 192 to 195
274 //
275 //   0.0015484   0.0000637   0.0014125   0.0026839
276 //
277 //          column 196 to 199
278 //
279 //   0.0035833   0.0039949   0.0038691   0.0032282
280 //
```

```
281 //          column 200 to 203
282 //
283 //    0.0021631    0.0008199    0.0006196    0.0019626
284 //
285 //          column 204 to 207
286 //
287 //    0.003031    0.0036848    0.0038407    0.0034824
288 //
289 //          column 208 to 211
290 //
291 //    0.0026621    0.0014932    0.0001345    0.0012312
292 //
293 //          column 212 to 215
294 //
295 //    0.0024216    0.0032793    0.0036924    0.003609
296 //
297 //          column 216 to 219
298 //
299 //    0.0030439    0.002076    0.0008371    0.0005054
300 //
301 //          column 220 to 223
302 //
303 //    0.0017717    0.0027931    0.0034349    0.0036136
304 //
305 //          column 224 to 228
306 //
307 //    0.003308    0.0025617    0.0014769    0.0002001
        0.001097
308 //
309 //          column 229 to 233
310 //
311 //    0.002241    0.0030797    0.0035026    0.003455
        0.0029452
312 //
313 //          column 234 to 237
314 //
315 //    0.0020432    0.0008713    0.000413    0.0016374
316 //
```

```
317 //          column 238 to 241
318 //
319 //    0.0026386    0.0032833    0.0034864    0.0032221
320 //
321 //          column 242 to 245
322 //
323 //    0.0025269    0.0014951    0.0002654    0.0009971
324 //
325 //          column 246 to 249
326 //
327 //    0.0021236    0.0029635    0.0034049    0.0033894
328 //
329 //          column 250 to 253
330 //
331 //    0.0029197    0.0020592    0.0009236    0.0003348
332 //
333 //          column 254 to 256
334 //
335 //    0.0015476    0.0025522    0.0032144
336 //
337 // Blackmann window based Filter output h(n)=
338 //
339 //   -7.725D-21
340 //   -0.0000259
341 //   -0.0000994
342 //   -0.0001879
343 //   -0.0001942
344 //    3.135D-19
345 //    0.0004427
346 //    0.0010144
347 //    0.0013951
348 //    0.0011582
349 //   -1.272D-18
350 //   -0.001977
351 //   -0.0040862
352 //   -0.005155
353 //   -0.0039758
354 //    3.072D-18
```

```
355 //     0.0060255
356 //     0.0118714
357 //     0.0143756
358 //     0.0107156
359 //    -5.373D-18
360 //    -0.0155126
361 //    -0.0302706
362 //    -0.0367268
363 //    -0.0278468
364 //     7.253D-18
365 //     0.0449152
366 //     0.0991097
367 //     0.1506861
368 //     0.1870979
369 //     0.1991026
370 //     0.1837596
371 //     0.1453485
372 //     0.0938771
373 //     0.0417702
374 //     6.621D-18
375 //    -0.0249443
376 //    -0.0322712
377 //    -0.0260792
378 //    -0.0130968
379 //    -4.443D-18
380 //     0.0086709
381 //     0.0113744
382 //     0.0091754
383 //     0.0045438
384 //     2.257D-18
385 //    -0.0028411
386 //    -0.0035753
387 //    -0.0027431
388 //    -0.0012801
389 //    -7.904D-19
390 //     0.0006867
391 //     0.0007815
392 //     0.0005293
```

```
393 //     0.0002104
394 //     1.306D-19
395 //    -0.0000662
396 //    -0.000045
397 //    -0.0000107
398 //     8.953D-20
399 //    -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

```
1
2 //Direct Sequence Spread Spectrum(DS-BPSK)
3 //OS:Windows 10
4 //Scilab 5.5.2
5
6 clear all;
7 clc;
8 close;
9
10 function[st,mt]= DS_Spread_Spectrum(bt,ct_polar)
```

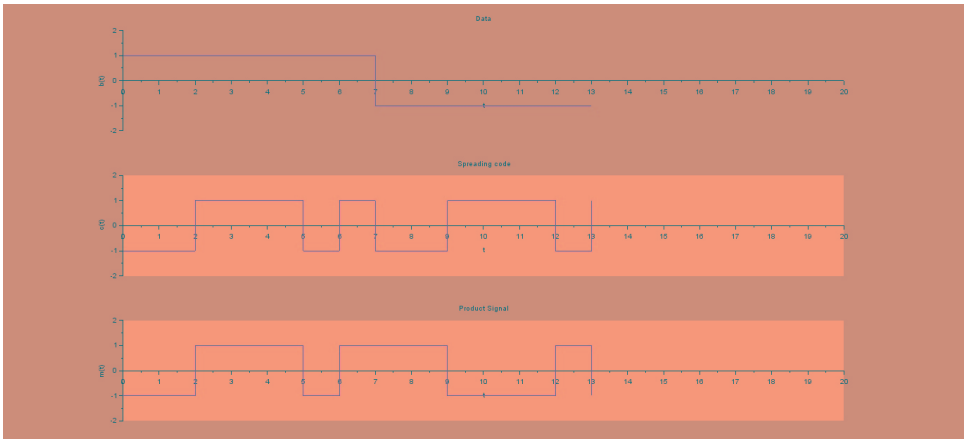

Figure 3.1: 3

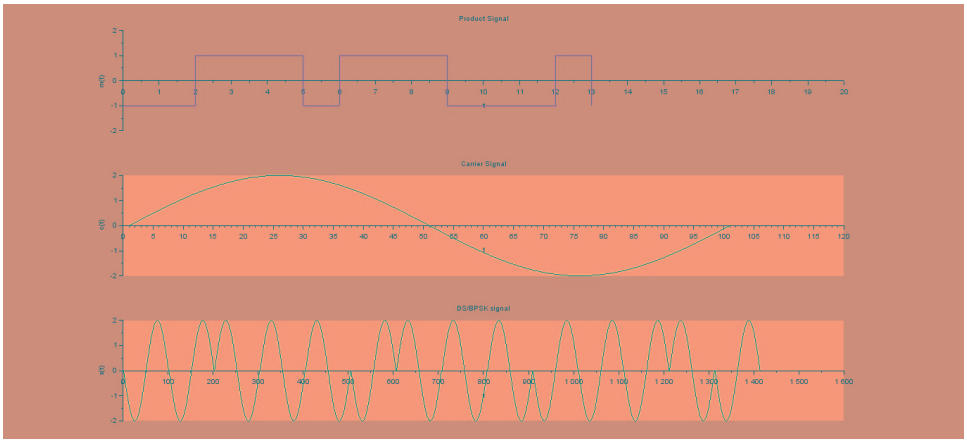

Figure 3.2: 3

```
11 //Generation of waveforms in DS/BPSK spread spectrum
       transmitter
12 //bt: Input Data Sequence (bipolar format)
13 //ct_polar: Spreading code (bipolar format)
14 Ft = 0:0.01:1;
15 //bt = [1*ones(1,N) -1*ones(1,N)];
16 t = 0:length(bt)-1;
17 //ct_polar = [-1,-1,1,1,1,-1,1,-1,-1,1,1,1,-1,1];
18 mt = bt.*ct_polar;
19 Carrier = 2*sin(Ft*2*%pi);
20 st = [];
21 for i = 1:length(mt)
22   st = [st mt(i)*Carrier];
23 end
24
25 figure
26 subplot(3,1,1)
27 a =gca();
28 a.x_location = "origin";
29 a.y_location = "origin";
30 a.data_bounds = [0,-2;20,2];
31 plot2d2(t,bt,5)
32 xlabel('

      t')
33 ylabel('

      b(t)')
34 title('Data')
35 subplot(3,1,2)
36 a =gca();
37 a.x_location = "origin";
38 a.y_location = "origin";
39 a.data_bounds = [0,-2;20,2];
40 plot2d2(t,ct_polar,5)
41 xlabel('

      t')
```

```
42 ylabel('
      c(t)')
43 title('Spreading code')
44 subplot(3,1,3)
45 a =gca();
46 a.x_location = "origin";
47 a.y_location = "origin";
48 a.data_bounds = [0,-2;20,2];
49 plot2d2(t,mt,5)
50 xlabel('
      t')
51 ylabel('
      m(t)')
52 title('Product Signal')
53
54 figure
55 subplot(3,1,1)
56 a =gca();
57 a.x_location = "origin";
58 a.y_location = "origin";
59 a.data_bounds = [0,-2;20,2];
60 plot2d2(t,mt,5)
61 xlabel('
      t')
62 ylabel('
      m(t)')
63 title('Product Signal')
64 subplot(3,1,2)
65 a =gca();
66 a.x_location = "origin";
67 a.y_location = "origin";
68 a.data_bounds = [0,-2;20,2];
69 plot(Carrier)
```

```
70 xlabel('
      t')
71 ylabel('
      c(t)')
72 title('Carrier Signal')
73 subplot(3,1,3)
74 a =gca();
75 a.x_location = "origin";
76 a.y_location = "origin";
77 a.data_bounds = [0,-2;20,2];
78 plot(st)
79 xlabel('
      t')
80 ylabel('
      s(t)')
81 title('DS/BPSK signal')
82 endfunction
83
84 bt = [1,1,1,1,1,1,1,-1,-1,-1,-1,-1,-1,-1]
85 ct_polar = [-1,-1,1,1,1,-1,1,-1,-1,1,1,1,-1,1]
86 [st,mt]= DS_Spread_Spectrum(bt,ct_polar)
```

Experiment: 4

Constellation Diagram For Binary PSK

Scilab code Solution 4.4 4

```
1  //Constellation Diagram For Binary PSK
2  //OS:Windows 10
3  //Scilab 5.5.2
4
5  clear all;
6  clc;
7  close;
8
9  function[y]= Constellation_BPSK()
10 M =2;
11 i = 1:M;
12 y = cos(2*%pi+(i-1)*%pi);
13 annot = dec2bin([length(y)-1:-1:0],log2(M));
14 disp(y,'coordinates of message points')
15 disp(annot,'Message points')
16 figure;
17 a =gca();
```

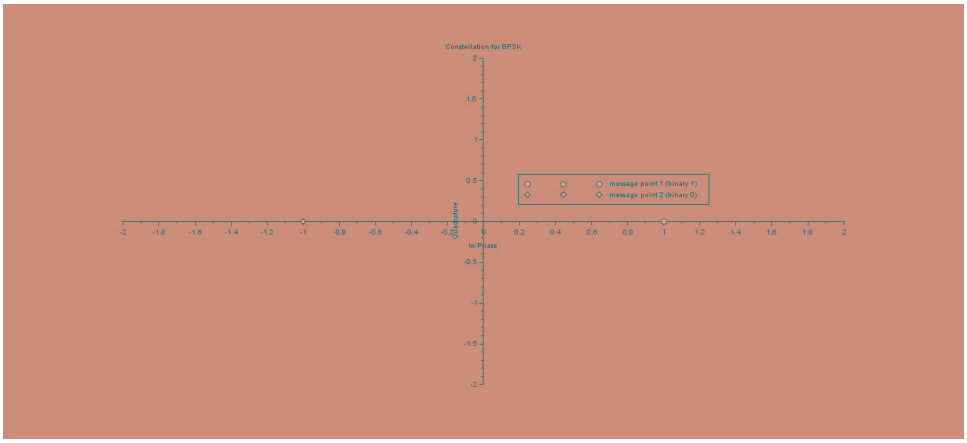

Figure 4.1: 4

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

      In-Phase');
24 ylabel('

      Quadrature');
25 title('Constellation for BPSK')
26 legend(['message point 1 (binary 1)';'message point
      2 (binary 0)'],5)
27 endfunction
28
29 Constellation_BPSK()
30
31 //Output
32 //coordinates of message points
33 //
34 //    1.   -1.
35 //
```

```
36 // Message points
37 //
38 //!1   0   !
```

Experiment: 5

Repetition Code

check Appendix AP 1 for dependency:

RepetitionCode.sci

Scilab code Solution 5.5 5

```
1  //Repetition Code
2  //OS:Windows 10
3  //Scilab 5.5.2
4
5  clear all;
6  clc;
7  close;
8
9  function [G,H,x]=RepetitionCode(n,k,m)
10 //Repetition Codes
11 //n =block of identical 'n' bits
12 //k =1 one bit
13 //m = 1;// bit value = 1
14 I = eye(n-k,n-k);//Identity matrix
15 P = ones(1,n-k);//coefficient matrix
16 H = [I P'];//parity-check matrix
17 G = [P 1];//generator matrix
```

```
18 x = m.*G; //code word
19 disp(G,'generator matrix');
20 disp(H,'parity-check matrix');
21 disp(x,'code word for binary one input');
22 endfunction
23
24 n=5;
25 k=1;
26 m=1;
27 [G,H,x]=RepetitionCode(n,k,m)
28
29 //Output
30 // generator matrix
31 //
32 //    1.    1.    1.    1.    1.
33 //
34 // parity-check matrix
35 //
36 //    1.    0.    0.    0.    1.
37 //    0.    1.    0.    0.    1.
38 //    0.    0.    1.    0.    1.
39 //    0.    0.    0.    1.    1.
40 //
41 // code word for binary one input
42 //
43 //    1.    1.    1.    1.    1.
```

Experiment: 6

Continuous Time Fourier Series of Sine Signal

Scilab code Solution 6.6 6

```
1  //Continuous Time Fourier Series of Sine Signal
2  //OS:Windows 10
3  //Scilab 5.5.2
4
5  clear;
6  clc;
7  close;
8
9  //periodic sine signal x(t) = sin(Wot)
10
11 t = 0:0.01:1;
12 T = 1;
13 Wo = 2*%pi/T;
14 xt = sin(Wo*t);
15 for k =0:5
16   C(k+1,:) = exp(-sqrt(-1)*Wo*t.*k);
17   a(k+1) = xt*C(k+1,:)'/length(t); //fourier series
       is done
18   if(abs(a(k+1))<=0.01)
```

```
19      a(k+1)=0;
20    end
21  end
22  a =a';
23  ak = [-a($:-1:1),a(2:$)];
24  disp(ak,'Continuous Time Fourier Series Coefficients
        are:')
25
26  //Output
27  // Continuous Time Fourier Series Coefficients are:
28  //
29  //
30  //           column 1 to 9
31  //
32  //    0.    0.    0.    0.    0.4950495i    0.   -0.4950495
       i    0.    0.
33  //
34  //           column 10 to 11
35  //
36  //    0.    0.
```

Experiment: 7

Spectrum of Signal (Frequency Response)-Blackmann Window

Scilab code Solution 7.7 7

```
1 //SPECTRUM OF SIGNAL (FREQUENCY RESPONSE)- BLACKMANN
      WINDOW
2 //OS: Windows 10
3 //Scilab 5.5.2
4
5 clear all;
6 clc;
7 close;
8
9 //With maximum normalized frequency f = 0.4
10
11 N = 11;
12 cfreq = [0.4 0];
13 [wft,wfm,fr]=wfir('lp',N,cfreq,'re',0);
14 wft;                  // Time domain filter
      coefficients
15 wfm;                  // Frequency domain filter
```

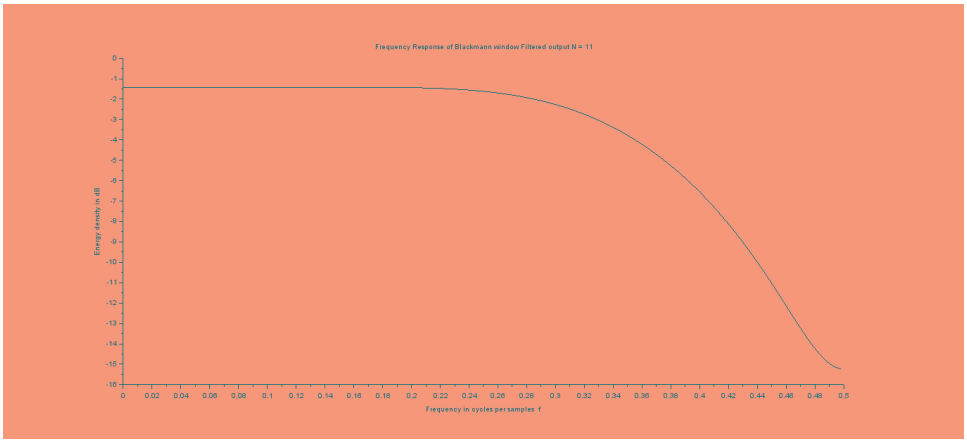

Figure 7.1: 7

```
    values
16  fr;                    // Frequency sample points
17  for n = 1:N
18    h_blackmann(n)=0.42-0.5*cos(2*%pi*n/(N-1))+0.08*
        cos(4*%pi*n/(N-1));
19    wft_blmn(n) = wft(n)*h_blackmann(n);
20  end
21  wfm_blmn = frmag(wft_blmn,length(fr));
22  WFM_blmn_dB =20*log10(wfm_blmn);
23  plot2d(fr,WFM_blmn_dB)
24  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

```
1  //COMPARISON OF DIFFERENT POWER SPECTRUM ESTIMATES
2  //OS:Windows 10
3  //Scilab 5.5.2
4
5  clear all;
6  clc;
7  close;
8
9  Q = 10;  //Quality factor
10 N = 1000; //Length of the sample sequence
11 //Bartlett Method
12 F_Bartlett = Q/(1.11*N);
13 disp(F_Bartlett,'Frequency Resolution of Bartlett
       Power Spectrum Estimation')
14 //Welch Method
15 F_Welch = Q/(1.39*N);
16 disp(F_Welch,'Frequency Resolution of Welch Power
       Spectrum Estimation')
17 //Blackmann-Tukey Method
```

```
18 F_Blackmann_Tukey  = Q/(2.34*N);
19 disp(F_Blackmann_Tukey,'Frequency Resolution of
      Blackmann Tukey Power Spectrum Estimation')
20
21
22 //Output
23 // Frequency Resolution of Bartlett Power Spectrum
      Estimation
24 //
25 //   0.009009
26 //
27 // Frequency Resolution of Welch Power Spectrum
      Estimation
28 //
29 //   0.0071942
30 //
31 // Frequency Resolution of Blackmann Tukey Power
      Spectrum Esti
32 // mation
33 //   0.0042735
```

Experiment: 9

Continuous Time Fourier Transform Of An Exponential Signal

Scilab code Solution 9.9 9

```
1 //CONTINUOUS TIME FOURIER TRANSFORM OF A EXPONENTIAL
      SIGNAL
2 //OS:Windows 10
3 //Scilab 5.5.2
4
5 clear all;
6 clc;
7 close;
8
9 //Continuous Time Exponential Signal x(t)= exp(-A*t)
     u(t), A>0
10
11 // Analog Signal
```

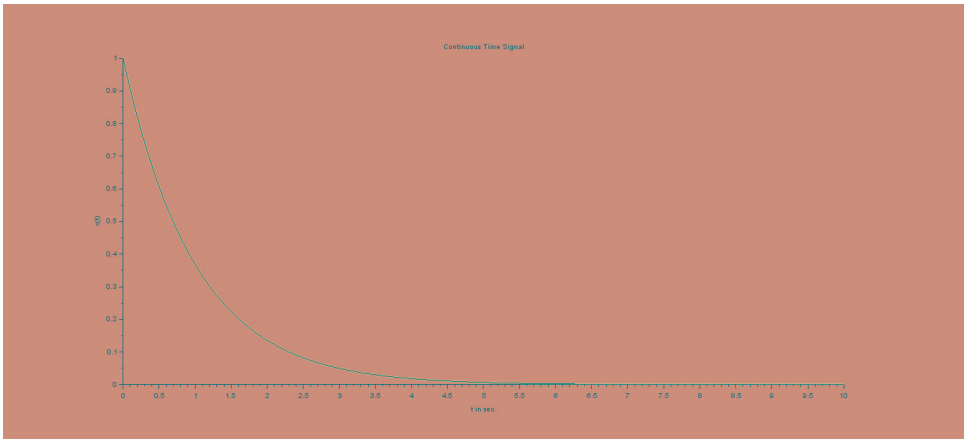

Figure 9.1: 9

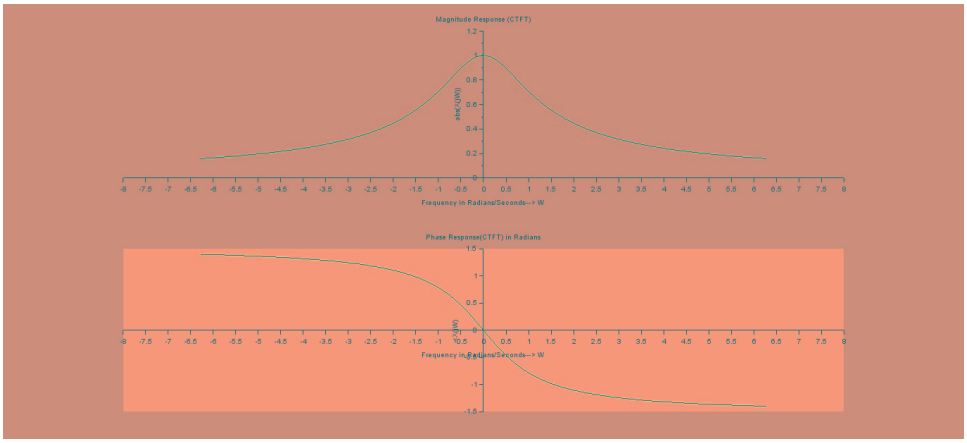

Figure 9.2: 9

```
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)')

```

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

Experiment: 10

FIR Band Pass Filter - Remez Algorithm-LPF

Scilab code Solution 10.10 10

```
1  //FIR BAND PASS FILTER - REMEZ ALGORITHM -FOR LPF
2  //OS: Windows 10
3  //Scilab 5.5.2
4
5  //Band Pass Filter of length M = 16
6  //Lower Cutoff frequency fp = 0.2 and Upper Cutoff
       frequency fs = 0.3
7
8  clear all;
9  clc;
10 close;
11 hn = 0;
12 hm = 0;
13 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
14 [hm,fr]=frmag(hn,256);
```

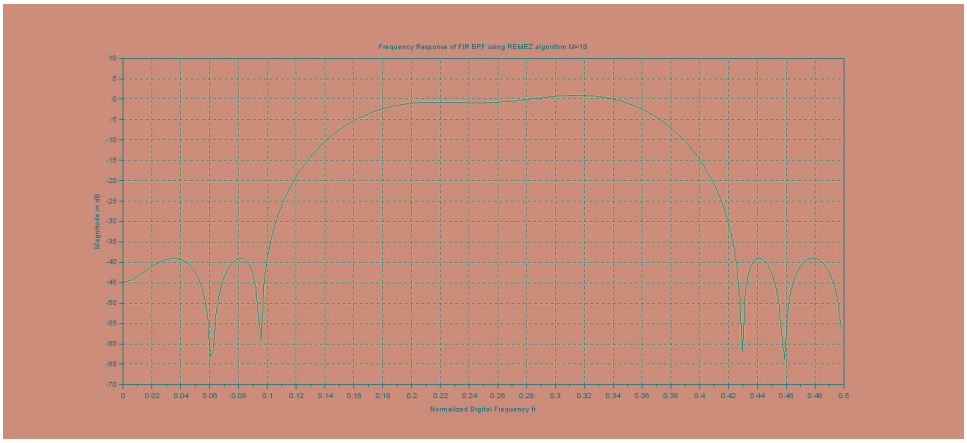

Figure 10.1: 10

```
15 disp(hn,'The Filter Coefficients are:')
16 figure
17 plot(.5*(0:255)/256,20*log10(frmag(hn,256)));
18 a = gca();
19 xlabel('Normalized Digital Frequency fr');
20 ylabel('Magnitude in dB');
21 title('Frequency Response of FIR BPF using REMEZ
      algorithm M=16')
22 xgrid(2)
23
24 //Output
25 //
26 // The Filter Coefficients are:
27 //
28 //   -0.0395487
29 //    0.0232284
30 //    0.0480681
31 //   -0.0218794
32 //    0.0975735
33 //   -0.1012773
34 //   -0.2880134
35 //    0.2847346
36 //    0.2847346
```

```
37 //    -0.2880134
38 //    -0.1012773
39 //     0.0975735
40 //    -0.0218794
41 //     0.0480681
42 //     0.0232284
43 //    -0.0395487
44 //
```

Appendix

Scilab code AP 11

```
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 12

```
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)')
38 title('Spreading code')
39 subplot(3,1,3)
40 a =gca();
41 a.x_location = "origin";
42 a.y_location = "origin";
43 a.data_bounds = [0,-2;20,2];
44 plot2d2(t,mt,5)
45 xlabel('

      t')
46 ylabel('

      m(t)')
47 title('Product Signal')
48
49 figure
50 subplot(3,1,1)
51 a =gca();
52 a.x_location = "origin";
53 a.y_location = "origin";
54 a.data_bounds = [0,-2;20,2];
55 plot2d2(t,mt,5)
56 xlabel('

      t')
57 ylabel('

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

```
      t')
66 ylabel('

      c(t)')
67 title('Carrier Signal')
68 subplot(3,1,3)
69 a =gca();
70 a.x_location = "origin";
71 a.y_location = "origin";
72 a.data_bounds = [0,-2;20,2];
73 plot(st)
74 xlabel('

      t')
75 ylabel('

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

DSSS