Scilab Manual for
Communication Systems
by Prof R.Senthilkumar, Assistant Professor
Electronics Engineering
Institute of Road and Transport Technology[1]

Solutions provided by
Mr R.Senthilkumar, Assistant Professor
Electronics Engineering
Institute of Road and Transport Technology

May 5, 2026

Contents

List of Experiments

Experiment: 1

Digital Modulation Scemes-ASK, PSK, FSK

Scilab code Solution 1.1 Digital Modulation Schemes

```scilab
1  //Caption:Waveforms of Different Digital Modulation
      techniques
2  //Digital Modulation Techniques
3  //To Plot the ASK, FSK and PSk Waveforms
4  clear;
5  clc;
6  close;
7  f = input('Enter the Analog Carrier Frequency in Hz'
      );
8  t = 0:1/512:1;
9  x = sin(2*%pi*f*t);
10 I = input('Enter the digital binary data');
11 //Generation of ASK Waveform
12 Xask = [];
13 for n = 1:length(I)
14   if((I(n)==1)&(n==1))
15     Xask = [x,Xask];
16   elseif((I(n)==0)&(n==1))
17     Xask = [zeros(1,length(x)),Xask];
```

```
18    elseif((I(n)==1)&(n~=1))
19      Xask = [Xask,x];
20    elseif((I(n)==0)&(n~=1))
21      Xask = [Xask,zeros(1,length(x))];
22    end
23 end
24 //Generation of FSK Waveform
25 Xfsk = [];
26 x1 = sin(2*%pi*f*t);
27 x2 = sin(2*%pi*(2*f)*t);
28 for n = 1:length(I)
29    if (I(n)==1)
30      Xfsk = [Xfsk,x2];
31    elseif (I(n)~=1)
32      Xfsk = [Xfsk,x1];
33    end
34 end
35 //Generation of PSK Waveform
36 Xpsk = [];
37 x1 = sin(2*%pi*f*t);
38 x2 = -sin(2*%pi*f*t);
39 for n = 1:length(I)
40    if (I(n)==1)
41      Xpsk = [Xpsk,x1];
42    elseif (I(n)~=1)
43      Xpsk = [Xpsk,x2];
44    end
45 end
46 figure
47 plot(t,x)
48 xtitle('Analog Carrier Signal for Digital Modulation
       ')
49 xgrid
50 figure
51 plot(Xask)
52 xtitle('Amplitude Shift Keying')
53 xgrid
54 figure
```

```
55  plot(Xfsk)
56  xtitle('Frequency  Shift  Keying')
57  xgrid
58  figure
59  plot(Xpsk)
60  xtitle('Phase  Shift  Keying')
61  xgrid
62  //Example
63  //Enter  the  Analog  Carrier  Frequency  2
64  //Enter  the  digital  binary  data[0,1,1,0,1,0,0,1]
```

Experiment: 2

Error Control Coding

Scilab code Solution 2.1 Convolutional Coding Technique

```
1  //Caption:Convolutional  Code  Generation
2  //Time  Domain  Approach
3  clear;
4  close;
5  clc;
6  g1 = input('Enter  the  input  Top  Adder  Sequence:=')
7  g2 = input('Enter  the  input  Bottom  Adder  Sequence:='
      )
8  m = input('Enter  the  message  sequence:=')
9  x1 = round(convol(g1,m));
10 x2 = round(convol(g2,m));
11 x1 = modulo(x1,2);
12 x2 = modulo(x2,2);
13 N = length(x1);
14 for i =1:length(x1)
15    x(i,:) =[x1(N-i+1),x2(N-i+1)];
16 end
17 x = string(x)
18 disp(x,'x=')
19 //Result
20 //Enter  the  input  Top  Adder  Sequence:=[1,1,1]
```

```
21  // Enter the input Bottom Adder Sequence:=[1,0,1]
22  // Enter the message sequence:=[1,1,0,0,1]
23  // x =
24  // ! 1   1   !
25  // !        !
26  // ! 1   0   !
27  // !        !
28  // ! 1   1   !
29  // !        !
30  // ! 1   1   !
31  // !        !
32  // ! 0   1   !
33  // !        !
34  // ! 0   1   !
35  // !        !
36  // ! 1   1   !
```

check Appendix AP 1 for dependency:

```
xor.sce
```

Scilab code Solution 2.2 Hamming Error Control Technique

```
1   // Caption:Hamming Encoding
2   //H(7,4)
3   // Code Word Length = 7, Message Word length = 4,
        Parity bits =3
4   // clear;
5   close;
6   clc;
7   // Getting Message Word
8   m3 = input('Enter the 1 bit(MSb) of message word');
9   m2 = input('Enter the 2 bit of message word');
10  m1 = input('Enter the 3 bit of message word');
11  m0 = input('Enter the 4 bit(LSb) of message word');
12  // Generating Parity bits
```

```
13  for i = 1:(2^4)
14    b2(i) = xor(m0(i),xor(m3(i),m1(i)));
15    b1(i) = xor(m1(i),xor(m2(i),m3(i)));
16    b0(i) = xor(m0(i),xor(m1(i),m2(i)));
17    m(i,:) = [m3(i) m2(i) m1(i) m0(i)];
18    b(i,:) = [b2(i) b1(i) b0(i)];
19  end
20  C = [b m];
21  disp('
         -------------------------------------------------------------
         ')
22  for i = 1:2^4
23    disp(i)
24    disp(m(i,:),'Message Word')
25    disp(b(i,:),'Parity Bits')
26    disp(C(i,:),'CodeWord')
27    disp("   ");
28    disp("   ");
29  end
30  disp('
         -------------------------------------------------------------
         ')
31  //Input
32  //Enter the 1 bit(MSb) of message word
        [0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1];
33  //Enter the 2 bit of message word
        [0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1];
34  //Enter the 3 bit of message word
        [0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1];
35  //Enter the 4 bit(LSb) of message word
        [0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1];
36  //Output Result
37  //Message Word
38  //     0.     0.     0.     0.
39  //  Parity Bits
40  //     0.     0.     0.
41  //  CodeWord
42  //     0.     0.     0.     0.     0.     0.     0.
```

```
43  //  //  2.
44  //    Message  Word
45  //        0.       0.       0.       1.
46  //    Parity  Bits
47  //        1.       0.       1.
48  //    CodeWord
49  //        1.       0.       1.       0.       0.       0.       1.
50  //  //        3.
51  //    Message  Word
52  //        0.       0.       1.       0.
53  //    Parity  Bits
54  //        1.       1.       1.
55  //    CodeWord
56  //        1.       1.       1.       0.       0.       1.       0.
57  //  //    4.
58  //    Message  Word
59  //        0.       0.       1.       1.
60  //    Parity  Bits
61  //        0.       1.       0.
62  //    CodeWord
63  //        0.       1.       0.       0.       0.       1.       1.
64  //  //      5.
65  //    Message  Word
66  //        0.       1.       0.       0.
67  //    Parity  Bits
68  //        0.       1.       1.
69  //    CodeWord
70  //        0.       1.       1.       0.       1.       0.       0.
71  //  //   6.
72  //  Message  Word
73  //        0.       1.       0.       1.
74  //  Parity  Bits
75  //        1.       1.       0.
76  //  CodeWord
77  //        1.       1.       0.       0.       1.       0.       1.
78  //  //   7.
79  //    Message  Word
80  //        0.       1.       1.       0.
```

```
81  //    Parity  Bits
82  //      1.      0.      0.
83  //    CodeWord
84  //      1.      0.      0.      0.      1.      1.      0.
85  //  //      8.
86  //    Message  Word
87  //      0.      1.      1.      1.
88  //    Parity  Bits
89  //      0.      0.      1.
90  //    CodeWord
91  //      0.      0.      1.      0.      1.      1.      1.
92  //  //      9.
93  //    Message  Word
94  //      1.      0.      0.      0.
95  //    Parity  Bits
96  //      1.      1.      0.
97  //    CodeWord
98  //      1.      1.      0.      1.      0.      0.      0.
99  //  //      10.
100 //    Message  Word
101 //      1.      0.      0.      1.
102 //    Parity  Bits
103 //      0.      1.      1.
104 //    CodeWord
105 //      0.      1.      1.      1.      0.      0.      1.
106 //  //      11.
107 //    Message  Word
108 //      1.      0.      1.      0.
109 //    Parity  Bits
110 //      0.      0.      1.
111 //    CodeWord
112 //      0.      0.      1.      1.      0.      1.      0.
113 //  //      12.
114 //    Message  Word
115 //      1.      0.      1.      1.
116 //    Parity  Bits
117 //      1.      0.      0.
118 //    CodeWord
```

```
119  //      1.      0.      0.      1.      0.      1.      1.
120  //  //      13.
121  //   Message  Word
122  //      1.      1.      0.      0.
123  //   Parity  Bits
124  //      1.      0.      1.
125  //   CodeWord
126  //      1.      0.      1.      1.      1.      0.      0.
127  //  //      14.
128  //   Message  Word
129  //      1.      1.      0.      1.
130  //   Parity  Bits
131  //      0.      0.      0.
132  //   CodeWord
133  //      0.      0.      0.      1.      1.      0.      1.
134  //  //      15.
135  //   Message  Word
136  //      1.      1.      1.      0.
137  //   Parity  Bits
138  //      0.      1.      0.
139  //   CodeWord
140  //      0.      1.      0.      1.      1.      1.      0.
141  //  //      16.
142  //   Message  Word
143  //      1.      1.      1.      1.
144  //   Parity  Bits
145  //      1.      1.      1.
146  //   CodeWord
147  //      1.      1.      1.      1.      1.      1.      1.
```

Appendix

Scilab code AP 1 `1 function [value]=xor(A, B)`
```
2   if(A==B)
3      value = 0;
4   else
5      value = 1;
6   end
7 endfunction
```
xor logic function