Scilab Manual for
Digital Communication
by Prof Kalawati Patil
Others
Thakur College of Engineering & Technology[1]

Solutions provided by
Mr Sanjay Rawat
Others
Mumbai University/Thakur College of Engg. & Tech.

May 5, 2026

Contents

List of Experiments

Experiment: 1

Study and generate different Line Codes - 1 (Unipolar and Polar RZ and NRZ)

Scilab code Solution 1.1 Study and generate different Line Codes 1 Unipolar NRZ

```
1  //Note: Details of scilab software version and OS
      version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
      64 bit versions)
4  //Program Title: Study and generate different Line
      Codes − 1(Unipolar NRZ)
5
6  clear;
7  close;
8  clc;
9  clf;
10 x=[1 0 1 0 0 1 1 0]//Data Stream
11
12 //NRZ
```

```
13  z=0;//Starting value on x axis
14  for i=1:length(x)
15
16      t=[z:1:z+1]//Set of x cordinates for current bit
             duration
17      subplot(2,1,1)
18      a=gca();
19      a.data_bounds=[0,-1.5;length(x),1.5]
20      a.grid=[1,-1]
21      title('Data')
22      plot(t,x(i))//Plot current data bit
23
24      subplot(2,1,2)
25      a=gca();
26      a.data_bounds=[0,-1.5;length(x),1.5]
27      a.grid=[1,-1]
28      title('NRZ')
29      if(x(i)==0)
30          plot(t,0)//Plot 0 for current bit duration
31      else
32          plot(t,1)//Plot 1 for current bit duration
33      end
34
35      z=z+1//Increament starting value on x axis by
          one bit period
36  end
```

Scilab code Solution 1.2 Study and generate different Line Codes 1 Polar
NRZ

```
1  //Note: Details of scilab software version and OS
      version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
```

5

```scilab
     64 bit versions)
4 //Program Title: Study and generate different Line
     Codes - 1(Polar NRZ)
5
6 clear;
7 close;
8 clc;
9 x=[1 0 1 0 0 1 1 0]//Data Stream
10
11 //Polar NRZ
12 z=0;//Starting value on x axis
13 for i=1:length(x)
14     t=[z:1:z+1]//Set of x cordinates for current bit
             duration
15          subplot(2,1,1)
16          a=gca();
17          a.data_bounds=[0,-1.5;length(x),1.5]
18          a.grid=[1,-1]
19          title('Data')
20          plot(t,x(i))//Plot current data bit
21
22          subplot(2,1,2)
23          a=gca();
24          a.data_bounds=[0,-1.5;length(x),1.5]
25          a.grid=[1,-1]
26          title('Polar NRZ')
27          if(x(i)==0)
28              plot(t,-1)//Plot -1 for current bit
                     duration
29          else
30              plot(t,1)//Plot 1 for current bit
                     duration
31          end
32
33      z=z+1//Increament starting value on x axis by
             one bit period
34 end
```

Scilab code Solution 1.3 Study and generate different Line Codes 1 Unipolar RZ

```
1  //Note:  Details  of  scilab  software  version  and  OS
       version  used:
2  //Tested  on  OS:  Windows  7  SP1,  64  bit  and  Windows  XP
       SP3,  32  bit
3  //Scilab  version:  5.4.1  (Tested  on  both  32  bit  and
       64  bit  versions)
4  //Program  Title:  Study  and  generate  different  Line
       Codes  −  1(Unipolar  RZ)
5
6  clear;
7  close;
8  clc;
9  x=[1  0  1  0  0  1  1  0]//Data  Stream
10
11 //RZ
12 z=0;//Starting  value  on  x  axis
13 for  i=1:length(x)
14     t=[z:z+1]//Set  of  x  cordinates  for  current  bit
            duration
15          subplot(2,1,1)
16          a=gca();
17          a.data_bounds=[0,-1.5;length(x),1.5]
18          a.grid=[1,-1]
19          title('Data')
20          plot(t,x(i))//Plot  current  data  bit
21
22     t=[z:0.5:z+0.5]//Set  of  x  cordinates  for  first
            half  bit  duration
23          subplot(2,1,2)
24          a=gca();
25          a.data_bounds=[0,-1.5;length(x),1.5]
```

7

```
26          a.grid=[1,-1]
27          title('Polar RZ')
28          if(x(i)==0)
29              plot(t,0)//Plot 0 for first half bit
                    duration
30          else
31              plot(t,1)//Plot 1 for first half bit
                    duration
32          end
33      t=[z+0.5:0.5:z+1]//Set of x cordinates for
            second half bit duration
34      plot(t,0)//Plot 0 for second half bit duration
35
36      z=z+1;//Increament starting value on x axis by
            one bit period
37 end
```

Scilab code Solution 1.4 Study and generate different Line Codes 1 Polar RZ

```
1 //Note: Details of scilab software version and OS
        version used:
2 //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
        SP3, 32 bit
3 //Scilab version: 5.4.1 (Tested on both 32 bit and
        64 bit versions)
4 //Program Title: Study and generate different Line
      Codes - 1(Polar RZ)
5
6 clear;
7 close;
8 clc;
9 x=[1 0 1 0 0 1 1 0]//Data Stream
10
11 //Polar RZ
```

8

```
12  z=0;//Starting value on x axis
13  for i=1:length(x)
14      t=[z:z+1]//Set of x cordinates for current bit
             duration
15          subplot(2,1,1)
16          a=gca();
17          a.data_bounds=[0,-1.5;length(x),1.5]
18          a.grid=[1,-1]
19          title('Data')
20          plot(t,x(i))//Plot current data bit
21
22      t=[z:0.5:z+0.5]//Set of x cordinates for first
             half bit duration
23          subplot(2,1,2)
24          a=gca();
25          a.data_bounds=[0,-1.5;length(x),1.5]
26          a.grid=[1,-1]
27          title('Polar RZ')
28          if(x(i)==0)
29              plot(t,-1)//Plot -1 for first half bit
                     duration
30          else
31              plot(t,1)//Plot 1 for first half bit
                     duration
32          end
33
34      t=[z+0.5:0.5:z+1]//Set of x cordinates for
             second half bit duration
35      plot(t,0)//Plot 0 for second half bit duration
36
37      z=z+1;//Increament starting value on x axis by
             one bit period
38  end
```

Experiment: 2

Study and generate different Line Codes - 2(Bipolar, Manchestre and Quaternary)

Scilab code Solution 2.1 Study and generate different Line Codes 2 Bipolar NRZ

```
1  //Note: Details of scilab software version and OS
       version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
        SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
       64 bit versions)
4  //Program Title: Study and generate different Line
       Codes - 2(Bipolar NRZ)
5
6  clear;
7  close;
8  clc;
9  x=[1 0 1 0 0 1 1 0]//Data stream
10
11 //Bipolar NRZ
12
```

```
13  z=0;//Starting point of plot on x-axis
14  ob=-1;//Initial o/p bit value
15
16  for i=1:1:length(x)
17
18      subplot(2,1,1)//Data Plot
19          a=gca();
20          a.data_bounds=[0,-1.5;length(x),1.5]
21          a.grid=[1,-1]
22          title('Data')
23
24          t=[z:1:z+1]//Plot range on x-axis (One bit
                 period)
25          plot(t,x(i))
26
27      subplot(2,1,2)//Bipolar Bipolar NRZ
28          a=gca();
29          a.data_bounds=[0,-1.5;length(x),1.5]
30          a.grid=[1,-1]
31          title('Bipolar NRZ')
32
33          if(x(i)==0)
34              t=[z:1:z+1]//Plot range on x-axis (One
                     bit period)
35              plot(t,0)//Plot zero
36          else
37              t=[z:1:z+1]//Plot range on x-axis (One
                     bit period)
38              ob=-ob//Invert previous o/p bit value
39              plot(t,ob)// Plot o/p bit
40          end
41
42      z=z+1//Move starting point of plot on x-axis by
             one bit period
43  end
```

Scilab code Solution 2.2 Study and generate different Line Codes 2 Bipolar RZ

```
1  //Note: Details of scilab software version and OS
       version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
       64 bit versions)
4  //Program Title: Study and generate different Line
       Codes - 2(Bipolar RZ)
5
6  clear;
7  close;
8  clc;
9  x=[1 0 1 0 0 1 1 0]//Data stream
10
11 //Bipolar RZ
12
13 z=0;//Starting point of plot on x-axis
14 ob=-1;//Initial o/p bit value
15
16 for i=1:1:length(x)
17
18     subplot(2,1,1)//Data Plot
19         a=gca();
20         a.data_bounds=[0,-1.5;length(x),1.5]
21         a.grid=[1,-1]
22         title('Data')
23
24         t=[z:1:z+1]//Plot range on x-axis (One bit
               period)
25         plot(t,x(i))
26
```

```
27      subplot(2,1,2)//Bipolar  Bipolar  RZ
28          a=gca();
29          a.data_bounds=[0,-1.5;length(x),1.5]
30          a.grid=[1,-1]
31          title('Bipolar  RZ')
32
33          if(x(i)==0)
34              t=[z:1:z+1]//Plot  range  on  x-axis  (One
                     bit  period)
35              plot(t,0)//Plot  zero
36          else
37              t=[z:0.5:z+0.5]//Plot  range  on  x-axis  (
                     first  half  bit  period)
38              ob=-ob//Invert  previous  o/p  bit  value
39              plot(t,ob)// Plot  o/p  bit
40              t=[z+0.5:0.5:z+1]//Plot  range  on  x-axis
                     (second  half  bit  period)
41              plot(t,0)
42          end
43
44      z=z+1//Move  starting  point  of  plot  on  x-axis  by
            one  bit  period
45  end
```

Scilab code Solution 2.3 Study and generate different Line Codes 2 Manchestre

```
1  //Note:  Details  of  scilab  software  version  and  OS
       version  used:
2  //Tested  on  OS:  Windows  7  SP1,  64  bit  and  Windows  XP
        SP3,  32  bit
3  //Scilab  version:  5.4.1  (Tested  on  both  32  bit  and
       64  bit  versions)
4  //Program  Title:  Study  and  generate  different  Line
       Codes  −  2(Manchestre)
5
```

```scilab
 6  clear;
 7  close;
 8  clc;
 9  x=[1 0 1 0 0 1 1 0]//Data Stream
10
11  //Manchester
12  z=0;//Starting value on x axis
13  for i=1:length(x)
14      t=[z:1:z+1]//Plot range on x-axis (One bit
            period)
15          subplot(2,1,1)
16          a=gca();
17          a.data_bounds=[0,-1.5;length(x),1.5]
18          a.grid=[1,-1]
19          title('Data')
20          plot(t,x(i))//Plot current data bit
21
22      t=[z:0.5:z+0.5]//Plot range on x-axis (first
            half bit period)
23          subplot(2,1,2)
24          a=gca();
25          a.data_bounds=[0,-1.5;length(x),1.5]
26          a.grid=[1,-1]
27          title('Manchester')
28          if(x(i)==0)
29              plot(t,1)//Plot 1 for first half bit
                    duration
30              t=[z+0.5:0.5:z+1]//Plot range on x-axis
                    (second half bit period)
31              plot(t,-1)//Plot 1 for second half bit
                    duration
32          else
33              plot(t,-1)//Plot -1 for first half bit
                    duration
34              t=[z+0.5:0.5:z+1]//Plot range on x-axis
                    (second half bit period)
35              plot(t,1)//Plot 1 for second half bit
                    duration
```

```
36              end
37          z=z+1;//Increament starting value on x axis by
                  one bit period
38      end
```

Scilab code Solution 2.4 Study and generate different Line Codes 2 Quaternary

```
1  //Note: Details of scilab software version and OS
       version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
        SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
        64 bit versions)
4  //Program Title: Study and generate different Line
       Codes - 2(Quaternary)
5
6  clear;
7  close;
8  clc;
9  x=[1 0 0 1 1 1 0 0]//Data Stream
10 //x=[0 0 0 1 1 0 1 1]//Data Stream
11 a=1;
12 //Polar NRZ
13 z=0;//Starting value on x axis
14 for i=1:2:length(x)
15     subplot(2,1,1)
16     g=gca();
17     g.data_bounds=[0,-1.5;length(x),1.5]
18     g.grid=[1,-1]
19     title('Data')
20         t=[z:1:z+1]//Plot range on x-axis (One bit
                period for current bit)
21         plot(t,x(i))//Plot curent bit
22         t=[z+1:1:z+2]//Plot range on x-axis (One bit
```

```
                      period for next bit)
23          plot(t,x(i+1))//Plot next bit
24
25
26      subplot(2,1,2)
27      g=gca();
28      g.data_bounds=[0,-2;length(x),2]
29      g.grid=[1,-1]
30      title('2B1Q (Quaternary)')
31          t=[z:2:z+2]//Plot range on x-axis (two bit
                 periods for current and next bit)
32          if((x(i)==0)&(x(i+1)==0))//Check current and
                 next bit combination
33              plot(t,-3/2*a)//if 00 then plot -3/2*a
34          elseif((x(i)==0)&(x(i+1)==1))//Check current
                 and next bit combination
35              plot(t,-1/2*a)//if 01 then plot -1/2*a
36          elseif((x(i)==1)&(x(i+1)==0))//Check current
                 and next bit combination
37              plot(t,1/2*a)//if 10 then plot 1/2*a
38          elseif((x(i)==1)&(x(i+1)==1))//Check current
                 and next bit combination
39              plot(t,3/2*a)//if 11 then plot 3/2*a
40          end
41      z=z+2//Increament starting value on x axis by
             two bits period
42  end
```

Experiment: 3

Study Carrier Modulation Techniques using BASK, BPSK and BFSK

Scilab code Solution 3.1 Study Carrier Modulation Techniques using BASK

```
1  //Note: Details of scilab software version and OS
      version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
      64 bit versions)
4  //Program Title: Study Carrier Modulation Techniques
       using BASK
5
6  clear;
7  clc;
8  close;
9  t = 0:0.01:1; // One symbol period
10 f=2; // Carrier cycles per symbol period
11 I=[0,0,1,1,0,1,0,1]; //data stream
12
13 //Generation of ASK Waveform
```

```
14
15  z =0;
16  for n =1: length ( I )
17      subplot (3 ,1 ,1)  // Carrier  Plot
18          a=gca ();
19          a . data_bounds =[0 , -1.5; length ( I ) ,1.5];  // set
                 the  boundary  values  for  the  x-y
                 coordinates .
20          a . x_location =" bottom ";
21          a . grid =[1 , -1];
22          title ('Carrier ')
23              plot (( t+z ), sin (2*% pi*f*t ));
24
25      subplot (3 ,1 ,2)  // Data  Plot
26          a=gca ();
27          a . data_bounds =[0 , -1.5; length ( I ) ,1.5];  // set
                 the  boundary  values  for  the  x-y
                 coordinates .
28          a . x_location =" bottom ";
29          a . grid =[1 , -1];
30          title ('Data ')
31              plot (( t+z ), I ( n ));
32
33      subplot (3 ,1 ,3)  // ASK  Waveform  Plot
34          a=gca ();
35          a . data_bounds =[0 , -1.5; length ( I ) ,1.5];  // set
                 the  boundary  values  for  the  x-y
                 coordinates .
36          a . x_location =" bottom ";
37          a . grid =[1 , -1];
38          title ('ASK  Waveform ')
39              plot (( t+z ), ( sin (2*% pi*f*t )) *( I ( n )));
40      z=z +1;
41  end
```

Scilab code Solution 3.2 Study Carrier Modulation Techniques using BFSK

```
1 //Note: Details of scilab software version and OS
        version used:
2 //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
        SP3, 32 bit
3 //Scilab version: 5.4.1 (Tested on both 32 bit and
       64 bit versions)
4 //Program Title: Study Carrier Modulation Techniques
        using BFSK
5
6 clear;
7 clc;
8 close;
9 t = 0:0.01:1; // One symbol period
10 f1=2; // Carrier cycles per symbol period
11 f2=4; // Carrier cycles per symbol period
12 I=[0,0,1,1,0,1,0,1]; //data stream
13
14 //Generation of FSK Waveform
15
16 z=0;
17 for n=1:length(I)
18     subplot(4,1,1) //Carrier 1 Plot
19         a=gca();
20         a.data_bounds=[0,-1.5;length(I),1.5]; //set
                the boundary values for the x-y
                coordinates.
21         a.x_location="bottom";
22         a.grid=[1,-1];
23         title('Carrier 1')
24             plot((t+z),sin(2*%pi*f1*t));
25
26     subplot(4,1,2) //Carrier 2 Plot
27         a=gca();
28         a.data_bounds=[0,-1.5;length(I),1.5]; //set
                the boundary values for the x-y
                coordinates.
```

```
29            a.x_location="bottom";
30            a.grid=[1,-1];
31            title('Carrier 2')
32                plot((t+z),sin(2*%pi*f2*t));
33
34      subplot(4,1,3) //Data Plot
35          a=gca();
36          a.data_bounds=[0,-1.5;length(I),1.5]; //set
                the boundary values for the x-y
                coordinates.
37          a.x_location="bottom";
38          a.grid=[1,-1];
39          title('Data')
40              plot((t+z),I(n));
41
42      subplot(4,1,4) //FSK Waveform Plot
43          a=gca();
44          a.data_bounds=[0,-1.5;length(I),1.5]; //set
                the boundary values for the x-y
                coordinates.
45          a.x_location="bottom";
46          a.grid=[1,-1];
47          title('FSK Waveform')
48              if (I(n)==0)
49                  plot((t+z),sin(2*%pi*f1*t));
50              elseif (I(n)==1)
51                  plot((t+z),sin(2*%pi*f2*t));
52              end
53      z=z+1;
54 end
```

Scilab code Solution 3.3 Study Carrier Modulation Techniques using BPSK

```
1 //Note: Details of scilab software version and OS
      version used:
```

```
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
      64 bit versions)
4  //Program Title: Study Carrier Modulation Techniques
       using BPSK
5
6  clear;
7  clc;
8  close;
9  t = 0:0.01:1; // One symbol period
10 f=2; // Carrier cycles per symbol period
11 I=[0,0,1,1,0,1,0,1]; //data stream
12
13 //Generation of PSK Waveform
14
15 z=0;
16 for n=1:length(I)
17     subplot(3,1,1) //Carrier Plot
18         a=gca();
19         a.data_bounds=[0,-1.5;length(I),1.5]; //set
               the boundary values for the x-y
               coordinates.
20         a.x_location="bottom";
21         a.grid=[1,-1];
22         title('Carrier')
23             plot((t+z),sin(2*%pi*f*t));
24
25     subplot(3,1,2) //Data Plot
26         a=gca();
27         a.data_bounds=[0,-1.5;length(I),1.5]; //set
               the boundary values for the x-y
               coordinates.
28         a.x_location="bottom";
29         a.grid=[1,-1];
30         title('Data')
31             plot((t+z),I(n));
32
```

```
33      subplot(3,1,3) //PSK Waveform Plot
34          a=gca();
35          a.data_bounds=[0,-1.5;length(I),1.5]; //set
               the boundary values for the x-y
               coordinates.
36          a.x_location="bottom";
37          a.grid=[1,-1];
38          title('PSK Waveform')
39              if (I(n)==1)
40                  plot((t+z),sin(2*%pi*f*t));
41              elseif (I(n)==0)
42                  plot((t+z),sin((2*%pi*f*t)+%pi));
43              end
44      z=z+1;
45  end
```

Experiment: 4

Study and generate OQPSK waveforms

Scilab code Solution 4.1 Study and generate offset QPSK waveforms

```
1  //Note: Details of scilab software version and OS
       version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
       64 bit versions)
4  //Program Title: Study and generate Offset QPSK (
      OQPSK) waveforms
5
6  clear;
7  clc;
8  close;
9  T=2;// One symbol period
10 t = 0:0.01:T/2;//Sampling Matrix for half symbol
       period
11 f=1; // Carrier frequency (cycles per bit period)
12 //I=[0 0 1 1 0 0 1 1]; //data stream
13 I=[0 0 0 1 1 0 1 1]; //data stream giving dibit
       equivalent to 0,1,2,3
```

```scilab
14  //I=[1 1 0 0 0 1 1 1]; //data stream
15
16  //Polar NRZ Converter
17  I_PNRZ = [] //empty matrix for Polar NRZ data
18          for n = 1:length(I)
19                  if I(n)== 0 then
20                          I_PNRZ = [I_PNRZ, -1]
21                  else
22                          I_PNRZ = [I_PNRZ, 1]
23                  end
24          end
25
26  I_Carrier = sqrt(2/T)*cos(2*%pi*f*t); //In phase
        carrier
27  Q_Carrier = sqrt(2/T)*sin(2*%pi*f*t); //Quadrature
        phase carrier
28
29  //Generation of OQPSK Waveform
30  z=0;//Starting point of plot on x-axis
31      subplot(3,1,1) //I-PSK Plot
32          a=gca();
33          a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
34          a.x_location="origin";
35          a.grid=[1,1];
36          title('I-Data and I-PSK')
37              plot((t+z),I_Carrier*I_PNRZ(1));//
                    I_Carrier * First bit (I Balance
                    Modulator)
38              plot((t+z),I_PNRZ(1),'r');//First bit
                    Data for reference
39  //xpause(2000000);//Delay for observation
40  z=z+1;//Move starting point of plot on x-axis by 1
        bit (half symbol) period
41  for n=2:1:length(I_PNRZ)
42      if modulo(n,2)==0 then//Check for odd-even bit
43          I_Bit=I_PNRZ(n-1)//set I bit as previous bit
44          Q_Bit=I_PNRZ(n)//set Q bit as current bit
45      else
```

24

```scilab
46              I_Bit=I_PNRZ(n)//set I bit as current bit
47              Q_Bit=I_PNRZ(n-1)//set Q bit as previous bit
48          end
49
50          subplot(3,1,1) //I-PSK Plot
51              a=gca();
52              a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
53              a.x_location="origin";
54              a.grid=[1,1];
55              title('I-Data and I-PSK')
56                  plot((t+z),I_Carrier*I_Bit);//I_Carrier
                        * Even bit (I Balance Modulator)
57                  plot((t+z),I_Bit,'r');//I Data for
                        reference
58
59          subplot(3,1,2) //Q-PSK Plot
60              a=gca();
61              a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
62              a.x_location="origin";
63              a.grid=[1,1];
64              title('Q-Data and Q-PSK')
65                  plot((t+z),Q_Carrier*Q_Bit);//Q_Carrier
                        * Odd bit (Q Balance Modulator)
66                  plot((t+z),Q_Bit,'r');//Q Data for
                        reference
67
68          subplot(3,1,3) //QPSK Plot
69              a=gca();
70              a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
71              a.x_location="Origin";
72              a.grid=[1,1];
73              title('OQPSK and I-Carrier')
74                  plot((t+z),(I_Carrier*I_Bit)+(Q_Carrier*
                        Q_Bit));//I-PSK + Q-PSK (Adder)
75                  plot((t+z),I_Carrier,'r');//I Carrier
                            for reference
76  z=z+1;//Move starting point of plot on x-axis by 1
        bit (half symbol) period
```

```
77  //xpause(2000000);//Delay for observation
78  end
```

Experiment: 5

Study and generate NON OQPSK waveforms

Scilab code Solution 5.1 Study and generate Non offset QPSK waveforms

```
1  //Note: Details of scilab software version and OS
       version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
      64 bit versions)
4  //Program Title: Study and generate Non-Offset QPSK
      waveforms
5
6  clear;
7  clc;
8  close;
9  T=2; //One Symbol period
10 t = 0:0.01:T; // Sampling Matrix for one symbol
       period
11 f=1; // Carrier frequency (cycles per bit period)
12 I=[0 0 0 1 1 0 1 1]; //data stream giving dibit
       equivalent to 0,1,2,3
13 //I=[0 1 1 0 1 0 0 0]; //data stream Simon Hykin Ex
```

```scilab
14
15  //Polar NRZ Converter
16  I_PNRZ = [] //empty matrix for Polar NRZ data
17          for n = 1:length(I)
18              if I(n)== 0 then
19                  I_PNRZ = [I_PNRZ, -1]
20              else
21                  I_PNRZ = [I_PNRZ, 1]
22              end
23          end
24
25  I_Carrier = sqrt(2/T)*cos(2*%pi*f*t); // In phase
        carrier
26  Q_Carrier = sqrt(2/T)*sin(2*%pi*f*t); // Quadrature
        phase carrier
27
28  //Generation of QPSK Waveform
29
30  z=0;//Starting point of plot on x-axis
31  for n=1:2:length(I_PNRZ)
32      I_Bit=I_PNRZ(n)
33      Q_Bit=I_PNRZ(n+1)
34      subplot(3,1,1) //I-PSK Plot
35          a=gca();
36          a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
37          a.x_location="origin";
38          a.grid=[1,1];
39          title('I-Data and I-PSK')
40              plot((t+z),I_Carrier*I_Bit);//I_Carrier
                    * Even bit (I Balance Modulator)
41              plot((t+z),I_Bit,'r');//I Data for
                    reference
42
43      subplot(3,1,2) //Q-PSK Plot
44          a=gca();
45          a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
46          a.x_location="origin";
```

```
47          a.grid=[1,1];
48          title('Q-Data and Q-PSK')
49              plot((t+z),Q_Carrier*Q_Bit);//Q_Carrier
                    * Odd bit (Q Balance Modulator)
50              plot((t+z),Q_Bit,'r');//Q Data for
                    reference
51
52      subplot(3,1,3) //QPSK Plot
53          a=gca();
54          a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
55          a.x_location="origin";
56          a.grid=[1,1];
57          title('QPSK and I-Carrier')
58              plot((t+z),(I_Carrier*I_Bit)+(Q_Carrier*
                    Q_Bit));//I-PSK + Q-PSK (Adder)
59              plot((t+z),I_Carrier,'r');//I Carrier
                    for reference
60 z=z+2;//Move starting point of plot on x-axis by 2
    bits (1 symbol) period
61 end
```

Experiment: 6

Study and generate 8-QAM waveforms

Scilab code Solution 6.1 Study and generate 8QAM waveforms

```
1  //Note: Details of scilab software version and OS
      version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
      64 bit versions)
4  //Program Title: Study and generate 8-QAM waveforms
5
6  clear;
7  clc;
8  close;
9  T=3; //One Symbol period
10 t = 0:0.01:T; // Sampling Matrix for one symbol
      period
11 f=1/T; // Carrier frequency (cycles per bit period)
12 I=[0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1];
      //data stream giving tribits equivalent to
    0,1,2,3,4,5,6,7
13
```

```scilab
14  //Polar NRZ Converter
15  I_PNRZ = []  //empty matrix for Polar NRZ data
16        for n = 1:length(I)
17             if I(n)== 0 then
18                  I_PNRZ = [I_PNRZ, -1]
19             else
20                  I_PNRZ = [I_PNRZ, 1]
21             end
22        end
23
24  I_Carrier = sqrt(2/T)*cos(2*%pi*f*t); // In phase
        carrier
25  Q_Carrier = sqrt(2/T)*sin(2*%pi*f*t); // Quadrature
        phase carrier
26
27  //Generation of 8-QAM Waveform
28
29  z=0;//Starting point of plot on x-axis
30  for n=1:3:length(I_PNRZ)
31      Q_Bit=I_PNRZ(n)//Set Q Bit Value
32      I_Bit=I_PNRZ(n+1)//Set I Bit Value
33      C_Bit=I_PNRZ(n+2)//Set C Bit Value
34          if C_Bit==-1 then //Set PAM, Product of C
                with I or Q
35              QC=0.5*Q_Bit//Set half amplitude
36              IC=0.5*I_Bit//Set half amplitude
37          else
38              QC=Q_Bit//Set full amplitude
39              IC=I_Bit//Set full amplitude
40          end
41
42      subplot(3,1,1) //QC Plot
43          a=gca();
44          a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
45          a.x_location="origin";
46          a.grid=[1,1];
47          title('Q-PAM')
48              plot((t+z),Q_Carrier*QC);//Q_Carrier * Q
```

```
                    -PAM (Q Balance Modulator)
49              plot((t+z),QC,'r');//Q-PAM Output
50
51      subplot(3,1,2) //IC Plot
52          a=gca();
53          a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
54          a.x_location="origin";
55          a.grid=[1,1];
56          title('I-PAM')
57              plot((t+z),I_Carrier*IC);//I_Carrier * I
                    -PAM (I Balance Modulator)
58              plot((t+z),IC,'r');//I-PAM Output
59
60      subplot(3,1,3) //8-QAM Plot
61          a=gca();
62          a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
63          a.x_location="origin";
64          a.grid=[1,1];
65          title('8-QAM')
66              plot((t+z),(I_Carrier*IC)+(Q_Carrier*QC)
                    );//I-PAM + Q-PAM (Adder)
67              plot((t+z),I_Carrier,'r');//I Carrier
                    for reference
68              plot(((t/3)+z),Q_Bit,'c');//Q Bit for
                    reference
69              plot(((t/3)+1+z),I_Bit,'b');//I Bit for
                    reference
70              plot(((t/3)+2+z),C_Bit,'m');//C Bit for
                    reference
71
72  z=z+3;//Move starting point of plot on x-axis by 3
        bits (1 symbol) period
73  end
```

Experiment: 7

Study and generate MSK waveforms (PSK Approach)

Scilab code Solution 7.1 Study and generate MSK waveforms with PSK Approach

```
1  //Note: Details of scilab software version and OS
       version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
       64 bit versions)
4  //Program Title: Study and generate MSK waveforms (
       PSK Approach)
5
6  clear;
7  clc;
8  close;
9  h=1/2;
10 T=1;// One symbol period
11 t = 0:0.01:T; // One symbol period vector
12 f=1; // Carrier cycles per symbol period "t"
13 //I=[0 1 0 1 0 1 0 1]; //data stream in binary to
       test worst case
```

```scilab
14  I=[0 0 0 1 1 0 1 1]; //data stream giving dibits
        equivalent to 0,1,2,3
15
16  //PNRZ Converter, converts data to PNRZ (Bi-Polar
        Signal)
17      I_PNRZ = [] //empty matrix for PNRZ data
18          for n = 1:length(I)
19              if I(n)== 0 then
20                  I_PNRZ = [I_PNRZ, -1]
21              else
22                  I_PNRZ = [I_PNRZ, 1]
23              end
24          end
25
26  //Generation of MSK Waveform using PSK approach
27
28  theta=0;//Initial phase in radians
29
30  z=0;//Starting point of plot on x-axis
31
32  for n=1:1:length(I_PNRZ)
33      subplot(3,1,1) //Data Plot
34          a=gca();
35          a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
                //set the boundary values for the x-y
                coordinates.
36          a.x_location="origin";
37          a.grid=[1,-1];
38          title('Data')
39              plot((t+z),I_PNRZ(n));
40
41      subplot(3,1,2) //MSK Plot
42          a=gca();
43          a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
                //set the boundary values for the x-y
                coordinates.
44          a.x_location="origin";
45          a.grid=[1,-1];
```

```
46          title('MSK')
47              theta_change = theta + ((I_PNRZ(n))*((
                   %pi*h*t)/T));//Phase variation over a
                    bit period
48              plot((t+z),sqrt(2/T)*cos(2*%pi*f*t +
                   theta_change)); // MSK Plot
49              plot((t+z),sqrt(2/T)*cos(2*%pi*f*t),'r')
                   ; // Carrier for reference
50
51      subplot(3,1,3) // Plot for MSK Phase variation
           wrt Carrier
52          a=gca();
53          a.x_location="bottom";
54          a.grid=[1,1];
55          title('MSK Phase variation wrt Carrier')
56          theta_degrees = theta_change*(180/%pi);//
               converts radians to degrees
57              plot((t+z),theta_degrees);// plote phase
                   variation for a bit period
58
59  theta=theta_change(length(theta_change));//Stores
        last value of phase to theta
60  z=z+1;//Move starting point of plot on x-axis by 1
        bit period
61  //xpause(2000000);    //Delay for observation
62  end
```

Experiment: 8

Study and generate MSK waveforms (FSK Approach)

Scilab code Solution 8.1 Study and generate MSK waveforms with FSK Approach

```scilab
1  //Note:  Details  of  scilab  software  version  and  OS
       version  used :
2  //Tested  on  OS:  Windows  7  SP1 ,  64  bit  and  Windows  XP
        SP3 ,  32  bit
3  //Scilab  version :  5.4.1  (Tested  on  both  32  bit  and
       64  bit  versions )
4  //Program  Title :  Study  and  generate  MSK  waveforms  (
       FSK  Approach )
5
6  clear ;
7  clc ;
8  close ;
9  h =1/2;
10 T =1;//  One  symbol  period
11 t  =  0:0.01: T;  //  One  symbol  period  vector
12 f =1;  //  Carrier  cycles  per  symbol  period  "t"
13 //I =[0  1  0  1  0  1  0  1];  //data  stream  in  binary  to
       test  worst  case
```

```scilab
14 I=[0 0 0 1 1 0 1 1]; //data stream giving dibits
      equivalent to 0,1,2,3
15
16 //PNRZ Converter, converts data to PNRZ (Bi-Polar
     Signal)
17     I_PNRZ = [] //empty matrix for PNRZ data
18         for n = 1:length(I)
19             if I(n)== 0 then
20                 I_PNRZ = [I_PNRZ, -1]
21             else
22                 I_PNRZ = [I_PNRZ, 1]
23             end
24         end
25
26 //Generation of MSK Waveform using FSK approach
27
28 bitchange=0;//Initial bit state (before first bit of
        sequence)
29 theta=0;//Initial phase state in radians (before
     first bit of sequence)
30 theta_degrees=[0,0];//Initial phase state in degrees
      (first element = start value, second element =
     last value)
31
32 z=0;//Starting point of plot on x-axis
33 for n=1:1:length(I_PNRZ)
34     subplot(3,1,1) //Data Plot
35         a=gca();
36         a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
             //set the boundary values for the x-y
             coordinates.
37         a.x_location="origin";
38         a.grid=[1,-1];
39         title('Data')
40             plot((t+z),I_PNRZ(n));
41
42     subplot(3,1,2) //MSK Plot
43         a=gca();
```

```scilab
44          a.data_bounds =[0 , -1.5; length ( I_PNRZ ) ,1.5];
               // set  the  boundary  values  for  the  x-y
               coordinates .
45          a.x_location =" origin ";
46          a.grid =[1 , -1];
47          title ( 'MSK ')
48          fm = f + ( I_PNRZ (n)*(h/(2*T)));// Generating
               two  frequencies  corresponding  to
49                                          // binary  0  (-1
                                               in  PNRZ)and
                                               binary  1  (1
                                               in  PNRZ)
50                                          // (0  ->  fc  -  h
                                               /2T)
51                                          // (1  ->  fc  + h
                                               /2T)
52              plot ((t+z),sqrt (2/T)* cos (2*%pi*fm*t +
                  theta )); // MSK  Plot
53              plot ((t+z),sqrt (2/T)* cos (2*%pi*f*t) ,'r')
                  ; // Carrier  for  reference
54
55      subplot (3 ,1 ,3) // Plot  for  MSK  Phase  variation
           wrt  Carrier
56          a=gca ();
57          a.x_location =" bottom ";
58          a.grid =[1 ,1];
59          title ( 'MSK  Phase  variation  wrt  Carrier ')
60          bitchange = bitchange + I_PNRZ (n); // Bit  State
               value ( cumulative )
61          theta = bitchange *(( %pi*h)/T); // Phase  state
               at  the  end  of  bit  period ,  in  radians
62          theta_degrees (2)= theta *180/% pi ; // Phase  state
               at  the  end  of  bit  period ,  in  degrees
63              plot ([z n], theta_degrees ); // plote  phase
                  variation  for  a  bit  period
64
65 theta_degrees (1)= theta_degrees (2); // Copy  end  phase
     value  to  start  phase  value  for  next  cycle
```

```
66  z=z+1;//Move starting point of plot on x-axis by 1
       bit period
67  //xpause(2000000); //Delay for observation
68  end
```

Experiment: 9

To calculate all Codewords, error detection and correction capability of given LBC

Scilab code Solution 9.1 Linear Block Codes

```
1  //Note: Details of scilab software version and OS
       version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
       64 bit versions)
4  //Program Title: Linear Block Codes (7,4)
5
6  clc;
7  clear;
8
9  k = 4; //Information message length
10 n = 7; //Coded word length
11
12 P = [1 1 0;0 1 1 ;1 1 1;1 0 1]//Parity Matrix
13 disp(P,'Parity Matrix P')
14
```

```scilab
15  G = [P eye(k,k)]//Generator Matrix to create code
        word in P1P2P3D1D2D3D4 format
16  disp(G,'Generator Matrix G')
17
18  H=[eye(n-k,n-k);P]'//Parity Check Matrix
19  disp(H,'Parity Check Matrix H')
20
21  //All_M = All 16 posibilities for Information
        Message Matrix
22  All_M = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;
23  0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;
24  1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;
25  1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1]
26
27  //Calculate all 16 posibile codewords
28  CodedMat=All_M*G;
29  CodedMat = modulo(CodedMat,2);//Convert generated
        code into binary
30  disp(CodedMat,'Codewords Matrix')
31
32  //Calculate Hamming Distances
33  HamDist=sum(CodedMat,'c')//Sum over the rows of
        CodedMat(column of values)
34  disp(HamDist,'Hamming Distances');
35
36  //Find Minimum non-zero Hamming Distance
37  [row,col]=find(HamDist==0);//find elements that are
        zero
38  HamDist(row,:)=[];//Remove all rows that are zero (
        replace by null)
39  MinHamDist=min(HamDist)//Find Minimum non-zero
        Hamming Distance
40  disp(MinHamDist,'Minimum Non-Zero Hamming Distance')
41
42  //Calculate Error Detection Capability
43  ErrDetCap=MinHamDist-1;
44  disp(ErrDetCap,'Error Detection Capability');
45
```

```scilab
46  //Calculate Error Correction Capability
47  ErrCorCap=(MinHamDist-1)/2;
48  disp(ErrCorCap,'Error Correction Capability');
49
50  //Generate random message
51  RandMessage=modulo(round(16*rand()),16)+1//Get
        random number between 1 to 16
52
53  M=All_M(RandMessage,:)//Select a random row from
        Message Matrix All_M as Information Message
54  disp(M,'Information Message M')
55
56  C = CodedMat(RandMessage,:)//Select a random row
        from Coded Matrix CodedMat as Coded Message
57  disp(C,'Coded Message C')
58
59  //Transmit random message
60  R=C//Create recieved code word
61
62  //Generate error at random bit position
63
64  ErrPos=modulo(round(8*rand()),8)//Get random number
        between 0 to 7
65
66  if ErrPos==0 then
67      //Do nothing, as '0' means no error
68  else
69      if R(ErrPos)==0 then
70          R(ErrPos)=1//Invert bit at Erroneous Bit
                Position
71      else
72          R(ErrPos)=0//Invert bit at Erroneous Bit
                Position
73      end
74  end
75
76  disp(R,'Recieved Code word R')
77
```

```scilab
78  //Error Correction
79
80  S=R*H'//Find Syndrome Matrix
81  S = modulo(S,2);//Convert Syndrome Matrix into
        binary
82  disp(S,'Syndrome Matrix R*H(transpose)')
83
84  if S==[0 0 0]  then //[0 0 0] indicates no error
85      disp(R,'Recieved Code without error')
86      disp(R(4:7),'Recieved Information Message')//
            Extract and display  Message from code word
87  else
88      //Find erroneous bit position
89      //Here we find colomn within H matrix with
            pattern simmilar to Syndrome Matrix
90      //The position number of that colomn is
            equivalent to erroneous bit position
91
92      ErrPos=1//Initiallize erroneous bit position
93      d=[H(:,ErrPos)]'//Transpose of first coloumn of
         H matrix
94                        //(Transpose is used to convert
                             colomn to row as syndrome is
                             in row format)
95
96          while ((d(1)<>S(1))|(d(2)<>S(2))|(d(3)<>S(3)
              )) do //Check element wise inequallity
              for any element (OR condition)
97              ErrPos=ErrPos+1//Increament erroneous
                  bit position (Point to next colomn)
98              d=[H(:,ErrPos)]'//Transpose of next
                  coloumn of H matrix
99          end
100
101          disp(ErrPos,'Erroneous Bit Position')
102
103          //Error correction
104          if R(ErrPos)==0 then
```

```
105                        R(ErrPos)=1//Invert  bit  at  Erroneous
                                  Bit  Position
106                        disp(R,'Recieved  Code  with  error
                              corrected ')
107                        disp(R(4:7),'Recieved  Information
                              Message ')//Extract  and  display
                              Message  from  code  word
108            else
109                R(ErrPos)=0//Invert  bit  at  Erroneous  Bit
                          Position
110                disp(R,'Recieved  Code  with  error
                      corrected ')
111                disp(R(4:7),'Recieved  Information
                      Message ')//Extract  and  display
                      Message  from  code  word
112            end
113  end
```

Experiment: 10

To encode Cyclic code and calculate Syndrome for the given generator polynomial

Scilab code Solution 10.1 Cyclic Codes

```
1  //Note:  Details  of  scilab  software  version  and OS
      version  used :
2  //Tested  on OS:  Windows  7  SP1,  64  bit  and  Windows  XP
      SP3,  32  bit
3  //Scilab  version :  5.4.1  (Tested  on  both  32  bit  and
      64  bit  versions)
4  //Program  Title :  Cyclic  Codes  (7,4)
5
6  clc;
7  clear;
8  k = 4;  //Information  Message  Length
9  n = 7;  //Codeword  Length
10
11 //Generator  Polynomial
12 x=poly(0, 'x');
13 GenPoly=1+x+x^3;
14 disp(GenPoly, 'The  Generator  Polynomial');
```

```scilab
15
16  //Generating Random Message
17
18  //All_M = All 16 posibilities for Information
        Message Matrix
19  All_M = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;
20  0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;
21  1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;
22  1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1]
23
24  RandMessage=modulo(round(16*rand()),16)+1//Get
        random number between 1 to 16
25
26  M=All_M(RandMessage,:)//Select a random row from
        Message Matrix All_M as Information Message
27  disp(M,'Information Message M')
28
29  //Message Polynomial
30  MesPoly=(M(1)*1) + (M(2)*(x^1)) + (M(3)*(x^2)) + (M
        (4)*(x^3));
31  disp(MesPoly,'Message Polynomial u(x)');
32
33  //Encoding
    _____

34
35  //Generating Codeword Polynomial
36  p=(x^(n-k))*(MesPoly);//Step 1 - multiply MesPoly by
        x^(n-k), [x^(n-k)*u(x)]
37  [RemPoly,q]=pdiv(p,GenPoly);//Step 2 - divide above
        product by GenPoly, g(x)(Polynomial Division)
38  RemPoly=modulo(RemPoly,2);//Convert Remainder
        Polynomial to binary to get parity check
        polynomial, b(x)
39  disp(RemPoly,'Remainder Polynomial b(x)');
40  CodePoly=RemPoly+(MesPoly*(x^(n-k)));//Step 3 - add
        (x^(n-k)*u(x)) and b(x) to get Codeword
        Polynomial
```

```scilab
41 disp(CodePoly,'Codeword Polynomial c(x)');
42
43 //Finding Coefficients of Codeword Polynomial
44 CodePolyCoeff=coeff(CodePoly);
45     //Removal of - signs from Coefficients of
              Codeword Polynomial
46     for i=1:length(CodePolyCoeff)
47         if (CodePolyCoeff(i)==-1) then
48             CodePolyCoeff(i)=1;
49         end
50     end
51 disp(CodePolyCoeff,'Coefficients of Codeword
    Polynomial');
52
53 //Generating 7 bit Codeword from Coefficients of
     Codeword Polynomial
54 C=CodePolyCoeff;
55 if length(C)<7 then
56     C(1,7)=0;//Assigning a value outside arrey
          dimension will automatically
57             //pad additional zeros to resize the
                arrey / vector
58 end
59 disp(C,'Generated Codeword');
60
61 //Transmition
    _____

62 R=C//Create recieved code word
63
64     //Generate error at random bit position
65     ErrPos=modulo(round(8*rand()),8)//Get random
          number between 0 to 7
66     //ErrPos=0 //for testing
67     if ErrPos==0 then
68         //Do nothing, as '0' means no error
69     else
70         if R(ErrPos)==0 then
```

47

```
71                 R(ErrPos)=1//Invert  bit  at  Erroneous  Bit
                       Position
72           else
73                 R(ErrPos)=0//Invert  bit  at  Erroneous  Bit
                       Position
74           end
75        end
76
77  //Reception  and  Decoding
     _____

78
79  disp(R,'Recieved  Code  word  R')
80
81  //Received  Polynomial
82  RecPoly=(R(1)*1) + (R(2)*(x^1)) + (R(3)*(x^2)) + (R
      (4)*(x^3)) + (R(5)*(x^4)) + (R(6)*(x^5)) + (R(7)
      *(x^6));
83  disp(RecPoly,'Received  Polynomial  u(x)');
84
85  //Syndrome  Polynomial
86  [SynPoly,q]=pdiv(RecPoly,GenPoly);
87  SynPoly=modulo(SynPoly,2)
88  disp(SynPoly,'Syndrome  Polynomial')
89
90  //Finding  Coefficients  of  Syndrome  Polynomial
91  SynPolyCoeff=coeff(SynPoly);
92      //Removal  of − signs  from  Coefficients  of
           Syndrome  Polynomial
93      for  i=1:length(SynPolyCoeff)
94          if  (SynPolyCoeff(i)==-1)  then
95              SynPolyCoeff(i)=1;
96          end
97      end
98  disp(SynPolyCoeff,'Coefficients  of  Syndrome
      Polynomial');
99
100 //Generating  3  bit  Syndrome  from  Coefficients  of
```

48

```scilab
        Syndrome  Polynomial
101 if  length ( SynPolyCoeff ) <3 then
102      SynPolyCoeff (1 ,3) =0;// Assigning  a  value  outside
           arrey  dimension  will  automatically
103              // pad  additional  zeros  to  resize  the
                  arrey  /  vector
104 end
105 disp ( SynPolyCoeff , 'Syndrome ');
106
107
108
109          // Create  H  ( Parity  check  matrix )  as  error
               lookup  table
110          P  =  [1  1  0;0  1  1  ;1  1  1;1  0  1] // Parity
               Matrix
111          H =[ eye (n -k ,n -k);P] '// Parity  Check  Matrix
112          // disp (H, 'Parity  Check  Matrix  H')  // for
               testing
113
114 if  SynPolyCoeff ==[0  0  0]   then  //[0  0  0]  indicates
      no  error
115      disp (R, 'Recieved  Code  without  error ')
116      disp (R(4:7) , 'Recieved  Information  Message ') //
           Extract  and  display   Message  from  code  word
117 else
118      // Find  erroneous  bit  position
119      // Here  we  find  colomn  within  H  matrix  with
            pattern  simmilar  to  Syndrome  Matrix
120      // The  position  number  of  that  colomn  is
            equivalent  to  erroneous  bit  position
121
122      ErrPos =1 // Initiallize  erroneous  bit  position
123      d =[ H (: , ErrPos )] '// Transpose  of  first  coloumn  of
            H  matrix
124                          //( Transpose  is  used  to  convert
                              colomn  to  row  as  syndrome  is
                              in  row  format )
125
```

```
126            while ((d(1)<>SynPolyCoeff(1))|(d(2)<>
               SynPolyCoeff(2))|(d(3)<>SynPolyCoeff(3)))
                do //Check element wise inequallity for
               any element (OR condition)
127             ErrPos=ErrPos+1//Increament erroneous
                   bit position (Point to next colomn)
128             d=[H(:,ErrPos)]'//Transpose of next
                   coloumn of H matrix
129            end
130
131            disp(ErrPos,'Erroneous Bit Position')
132
133            //Error correction
134            if R(ErrPos)==0 then
135                R(ErrPos)=1//Invert bit at Erroneous Bit
                       Position
136            else
137                R(ErrPos)=0//Invert bit at Erroneous Bit
                       Position
138            end
139            disp(R,'Recieved Code with error corrected')
140            disp(R(4:7),'Recieved Information Message')
                   //Extract and display  Message from code
                   word
141 end
142 disp(M,'Information Message M that was sent ...')
```

Experiment: 11

To encode and decode Hamming code

Scilab code Solution 11.1 Hamming Codes

```
1  //Note: Details of scilab software version and OS
       version used:
2  //Tested on OS: Windows 7 SP1, 64 bit and Windows XP
       SP3, 32 bit
3  //Scilab version: 5.4.1 (Tested on both 32 bit and
       64 bit versions)
4  //Program Title: Hamming Codes (7,4)
5
6  clc;
7  clear;
8
9  k = 4; //Information message matrix length
10 n = 7; //Coded word length
11
12 P = [1 1 0;0 1 1 ;1 1 1;1 0 1]//Parity Matrix
13 disp(P,'Parity Matrix P')
14
15 G = [P eye(k,k)]//Generator Matrix to create code
       word in P1P2P3D1D2D3D4 format
```

```
16  G(:,[3 4])=G(:,[4 3])//Swap colomn 3 and 4 of G to
        create code word in P1P2D1P3D2D3D4 format
17  disp(G,'Generator Matrix G')
18
19  H=[eye(n-k,n-k);P]'//Parity Check Matrix
20  H(:,[3 4])=H(:,[4 3])//Swap colomn 3 and 4 of H to
        satisfy GH'=0
21  disp(H,'Parity Check Matrix H')
22
23  //disp(modulo(G*H',2),'GH')//Check if the condition
        GH'=0 satisfy (for testing)
24
25  //M = [1 1 0 1]//Information Message Matrix for
        testing
26
27  //Generate random message
28  //All_M = All 16 posibilities for Information
        Message Matrix
29  All_M = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;
30  0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;
31  1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;
32  1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1]
33  RandMessage=modulo(round(16*rand()),16)+1//Get
        random number between 1 to 16
34  M=All_M(RandMessage,:)//Select a random row from 1
        to 16 as Information Message
35
36  disp(M,'Information Message M')
37
38  C = M*G;//Generate code word
39  C = modulo(C,2);//Convert generated code into binary
40  disp(C,'Code word of (7,4) Hamming code M*G')
41
42  R=C//Create recieved code word
43
44  //Generate error at random bit position
45
46  ErrPos=modulo(round(8*rand()),8)//Get random number
```

```
        between 0 to 7
47
48  if ErrPos==0 then
49      //Do nothing, as '0' means no error
50  else
51      if R(ErrPos)==0 then
52          R(ErrPos)=1//Invert bit at Erroneous Bit
                  Position
53      else
54          R(ErrPos)=0//Invert bit at Erroneous Bit
                  Position
55      end
56  end
57
58  disp(R,'Recieved Code word R')
59
60  //Error Correction
61
62  S=R*H'//Find Syndrome Matrix
63  S = modulo(S,2);//Convert Syndrome Matrix into
        binary
64  disp(S,'Syndrome Matrix R*H(transpose)')
65
66  if S==[0 0 0]   then  //[0 0 0] indicates no error
67      disp(R,'Recieved Code without error')
68      disp([R(3) R(5:7)],'Recieved Information Message
            ')//Extract and display  Message from code
            word
69  else
70      //Find erroneous bit position
71      //Here we find colomn within H matrix with
            pattern simmilar to Syndrome Matrix
72      //The position number of that colomn is
            equivalent to erroneous bit position
73
74      ErrPos=1//Initiallize erroneous bit position
75      d=[H(:,ErrPos)]'//Transpose of first coloumn of
            H matrix
```

```scilab
76                          //(Transpose is used to convert
                                colomn to row as syndrome is
                                in row format)
77
78          while ((d(1)<>S(1))|(d(2)<>S(2))|(d(3)<>S(3)
               )) do //Check element wise inequallity
               for any element (OR condition)
79             ErrPos=ErrPos+1//Increament erroneous
                  bit position (Point to next colomn)
80             d=[H(:,ErrPos)]'//Transpose of next
                  coloumn of H matrix
81          end
82
83          disp(ErrPos,'Erroneous Bit Position')
84
85          //Error correction
86          if R(ErrPos)==0 then
87                  R(ErrPos)=1//Invert bit at Erroneous
                        Bit Position
88                  disp(R,'Recieved Code with error
                        corrected')
89                  disp([R(3) R(5:7)],'Recieved
                        Information Message')//Extract
                        and display  Message from code
                        word
90          else
91              R(ErrPos)=0//Invert bit at Erroneous Bit
                  Position
92              disp(R,'Recieved Code with error
                  corrected')
93              disp([R(3) R(5:7)],'Recieved Information
                   Message')//Extract and display
                  Message from code word
94          end
95   end
```