

Scilab Manual for  
Applied Mathematics for Communication  
Engineers  
by Prof Prarthan Mehta  
Others  
Dharmsinh Desai University<sup>1</sup>

Solutions provided by  
Prof PRARTHAN MEHTA  
Others  
DHARMSINH DESAI UNIVERSITY

June 30, 2026

<sup>1</sup>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 TRUNCATION OF A SQUARE WAVE USING FOURIER SERIES EXPANTION	4
2 ESTIMATION OF THE ROOTS(ZEROS) FOR A REAL VALUED FUNCTION USING NEWTON-RAPHSON METHOD	8
3 DFT & IDFT OF A GIVEN SEQUENCE	11
4 SYSTEM RESPONSE USING CONVOLUTION	14
5 NODE VOLTAGE & MESH CURRENT ANALYSIS	15
6 STABILITY OF A FEEDBACK SYSTEM IN XCOS	18
7 IMPLEMENTATION OF CHANNEL CODING-LINEAR BLOCK CODE	19
8 FILTER DESIGN USING LAPLACE TRANSFORM	24
9 SIGNAL MODULATION	26
10 APPLICATION OF LINEAR ALGEBRA IN DIGITAL COMMUNICATION	28

# List of Experiments

Solution 1.1	Truncation of the Square Wave . . . . .	4
Solution 2.1	Estimation of Roots . . . . .	8
Solution 3.1	DFT of a Given Sequence . . . . .	11
Solution 3.2	IDFT of the given response . . . . .	12
Solution 4.1	System Response using Convolution . . . . .	14
Solution 5.1	Node Voltage . . . . .	15
Solution 5.2	Mesh Current . . . . .	16
Solution 7.1	Linear Block Coder Decoder . . . . .	19
Solution 8.1	Filter Design . . . . .	24
Solution 9.1	BPSK . . . . .	26
Solution 10.1	Rotation of a Vector . . . . .	28

# Experiment: 1

## TRUNCATION OF A SQUARE WAVE USING FOURIER SERIES EXPANTION

Scilab code Solution 1.1 Truncation of the Square Wave

```
1 v=1;
    //Multiplier Voltage
2 w=15;
    //Frequency
3 t=0:0.001:0.99;
    //Incremental Time Stamp
4 x1=v*(cos(w*t)-cos(3*w*t))/3)/%pi;
    //
    First 2 Components of the Fourier Series of a
    Square Wave with Frequency W rad/sec
5 x2=v*(cos(w*t)-cos(3*w*t))/3+cos(5*w*t)/5)/%pi;
    //First 3
```

```

        Components of the Fourier Series of a Square Wave
        with Frequency W rad/sec
6  x3=v*(cos(w*t)-cos(3*w*t)/3+cos(5*w*t)/5-cos(7*w*t)
    /7)/%pi;                                //First 4 Components
        of the Fourier Series of a Square Wave with
        Frequency W rad/sec
7  x4=v*(cos(w*t)-cos(3*w*t)/3+cos(5*w*t)/5-cos(7*w*t)
    /7+cos(9*w*t)/9)/%pi;                    //First 5 Components
        of the Fourier Series of a Square Wave with
        Frequency W rad/sec
8  x5=v*(cos(w*t)-cos(3*w*t)/3+cos(5*w*t)/5-cos(7*w*t)
    /7+cos(9*w*t)/9-cos(11*w*t)/11)/%pi;    //
        First 6 Components of the Fourier Series of a
        Square Wave with Frequency W rad/sec
9  x6=v*(cos(w*t)-cos(3*w*t)/3+cos(5*w*t)/5-cos(7*w*t)
    /7+cos(9*w*t)/9-cos(11*w*t)/11+cos(13*w*t)/13)/
    %pi;    //First 7 Components of the Fourier
        Series of a Square Wave
10 subplot(3,2,1)

        //Deviding the Figure Qindow in Six Subwindows to
        plot diffenrent components
11 plot(x1);

        //Plotting the First 2 components of the Fourier
        Series of a Square Wave
12 title('Square Wave Constructed with First 2
        Components of Fourier Series');    //Assigning
        a Title of the Figure
13 xlabel('Samples');

        //Assigning an X-label of the figure
14 ylabel('Amplitude (V)');

        //Assigning of a Y-label of the figure
15 subplot(3,2,2)
16 plot(x2);

```

```

    //Plotting the First 3 components of the Fourier
    Series of a Square Wave
17 title('Square Wave Constructed with First 3
    Components of Fourier Series'); //Assigning
    a Title of the Figure
18 xlabel('Samples');

    //Assigning an X-label of the figure
19 ylabel('Amplitude (V)');

    //Assigning of a Y-label of the figure
20 subplot(3,2,3)
21 plot(x3);

    //Plotting the First 4 components of the Fourier
    Series of a Square Wave
22 title('Square Wave Constructed with First 4
    Components of Fourier Series'); //Assigning
    a Title of the Figure
23 xlabel('Samples');

    //Assigning an X-label of the figure
24 ylabel('Amplitude (V)');

    //Assigning of a Y-label of the figure
25 subplot(3,2,4)
26 plot(x4);

    //Plotting the First 5 components of the Fourier
    Series of a Square Wave
27 title('Square Wave Constructed with First 5
    Components of Fourier Series'); //Assigning
    a Title of the Figure
28 xlabel('Samples');

    //Assigning an X-label of the figure
29 ylabel('Amplitude (V)');

```

```

    //Assigning of a Y-label of the figure
30
31 subplot(3,2,5)
32 plot(x5);

    //Plotting the First 6 components of the Fourier
    Series of a Square Wave
33 title('Square Wave Constructed with First 6
    Components of Fourier Series'); //Assigning
    a Title of the Figure
34 xlabel('Samples');

    //Assigning an X-label of the figure
35 ylabel('Amplitude (V)');

    //Assigning of a Y-label of the figure
36
37 subplot(3,2,6)
38 plot(x6);

    //Plotting the First 7 components of the Fourier
    Series of a Square Wave
39 title('Square Wave Constructed with First 7
    Components of Fourier Series'); //Assigning
    a Title of the Figure
40 xlabel('Samples');

    //Assigning an X-label of the figure
41 ylabel('Amplitude (V)');

    //Assigning of a Y-label of the figure
42
43
44 ///---Input Parameters---//
45 //v==> voltage
46 //w==>Frequency

```

---

## Experiment: 2

# ESTIMATION OF THE ROOTS(ZEROS) FOR A REAL VALUED FUNCTION USING NEWTON-RAPHSON METHOD

Scilab code Solution 2.1 Estimation or Roots

```
1
2
3 clear
4
5
6 function [y, deriv] = fcn_nr(x)
7     y = x^3 + 4*(x)^2 -10;
8     deriv = 3*(x)^2 + 8*x;
9
10 endfunction
11
12 xp = -15:1:15;
13 n = length(xp);
```

```

14 for i=1:n
15     yp(i) = fcn_nr(xp(i));
16 end
17 plot(xp,yp)
18 xlabel('x')
19 ylabel('y')
20 title('Plot of function')
21
22
23
24 x = input('Enter initial guess of root location: ');
25
26 itermax = 10;        // % max # of iterations
27 iter = 0;
28 errmax = 0.00001;   // % convergence tolerance
29 error1 = 1;
30
31
32 while error1 > errmax & iter < itermax
33
34     iter = iter + 1;
35     [f fprime] = fcn_nr(x);
36     if fprime == 0
37         break;
38     end;
39
40     xnew = x - f / fprime;
41
42     error1 = abs((xnew - x)/xnew) * 100; // % find
        change from previous
43
44     x = xnew        // % set up for next iteration
45 disp('Iteration No. ')
46 disp(iter)
47 disp('the estimated value of the variable x is=')
48 disp(x)
49
50

```

```
51 end
52
53 ///-----Input Parameters-----///
54
55 //This code will work for different functions of the
    variable x. hence the user has to modify the
    function & its derivative accordingly
56 //Itermax==>maximum number of iteration for which
    the method has to estimate the value of the
    variable
57 //errmax==>maximun tolerable error
```

---



```

13 disp('The DFT of the given sequence is:')
//
    Displaying the DFT
14 disp(y)
15
16
17 ///--Input Parameters---//
18
19 //x==>digital sequence for which DFT has to be found

```

---

### Scilab code Solution 3.2 IDFT of the given response

```

1 clear
2 clc
3
4 x=input('Enter the sequence:');

    //Input the DFT
5 N=length(x);

    //Finding the length to get N-point IDFT
6 for n=1:N
7     y(n)=0;
8     for k=1:N
9         y(n)=y(n)+(x(k)*(exp(%i*2*pi*(k-1)*(n-1)/N)
            )); //Equation to
            find the IDFT
10    end
11    y(n)=y(n)/N;
12 end
13 disp('The IDFT of the given sequence is:')
//Displaying
    the IDFT
14 disp(y)
15

```

```
16
17 //—Input Parameters---//
18
19 //x==>Frequency domain digital representation of the
    sequence (DFT) for which IDFT has to be found
```

---

## Experiment: 4

# SYSTEM RESPONSE USING CONVOLUTION

Scilab code Solution 4.1 System Response using Convolution

```
1
2 X=input('Enter the Input data Sequence X[n]: ');
3 H=input('Enter the Response of the System H[n]: ');
4
5 Y=conv(X,H);
6
7 disp('The response of the given system for the given
      input data sequence is Y[n]: ');
8 disp(Y);
9
10
11 //—Input Variables—//
12 //X==>digital sequence X[n]
13 //H==>filter impulse response H[n]
```

---

## Experiment: 5

# NODE VOLTAGE & MESH CURRENT ANALYSIS

Scilab code Solution 5.1 Node Voltage

```
1 //number of loops in the networks=2
2
3 v1=input('Enter the voltage in loop-1 (in V):');
4 i2=input('Enter the current through loop-2 (in A):')
5 ;
6 r2=input('Enter the resistance in series with V1 in
7 loop-1 (in Ohm):');
8 r1=input('Enter the resistance between v2 & GND. (a
9 common element bwtween loop-1 & 2) (in Ohm):');
10
11 //Teh node voltage at node-2 can be solved as
12 //temp1=(v2-v1)/r2;
13 //temp2=(v2/r1)
14 //0=temp1+temp2+i2;
15 temp3=i2*r1*r2;
16 temp4=v1/r2;
17 temp5=temp4-temp3;
18 temp6=temp5*r1*r2;
19 v2=temp6/(r1+r2);
```

```

17
18 disp('Voltage at Node-2 V2=');
19 disp(v2)
20
21
22 //---Input Variables---//
23
24 //v1==>equivalent voltage in loop 1 ---constant
25 //i2==>value of the current source in llop 2--
    constant
26 //r2==>value of the resitance in series with V1 in
    loop 1-- constant
27 //r1==>value of the resitance in the common branch
    in loop 1 & loop 2--constant

```

---

### Scilab code Solution 5.2 Mesh Current

```

1 //number of loops in the networks=2
2
3 v1=input('Enter the voltage in loop-1 (in V):');
4 v2=input('Enter the voltage in loop-2 (in V):');
5 r1=input('Enter the resistance in series with V1 in
    loop-1 (in Ohm):');
6 r2=input('Enter the resistance between v2 & GND. (a
    common element bwtween loop-1 & 2) (in Ohm):');
7 r3=input('Enter the resistance in series with V2 in
    loop-2 (in Ohm):');
8
9 //The mesh current can be solved as
10 //v1=(r1+r2)*i1-r2*i2;
11 //-v2=-r2*i1+(r2+r3)*i2;
12
13 //[v]=[r][i];
14
15 //[i]=[v][R]; where [R]=inv([r]);

```

```

16
17
18 v=[v1;v2];
19
20 r=[r1+r2 -r2 ; -r2 r2+r3];
21
22 //i=[i1;i2];
23
24 R=inv(r);
25
26
27 i=R*v;
28
29 disp('the current through the loops=')
30 disp(i)
31
32
33 //—Input variables—//
34 //v1==>voltage in loop 1—constant
35 //v2==>voltage in loop 2—constant
36 //r1==>resistance in loop 1 — constant
37 //r2==>resistance in the common branch of loop1 &
    loop 2 —constant
38 //r3==>resistance in loop 2 — constant

```

---

## **Experiment: 6**

# **STABILITY OF A FEEDBACK SYSTEM IN XCOS**

This code can be downloaded from the website [www.scilab.in](http://www.scilab.in) This code

can be downloaded from the website [www.scilab.in](http://www.scilab.in)

## Experiment: 7

# IMPLEMENTATION OF CHANNEL CODING-LINEAR BLOCK CODE

Scilab code Solution 7.1 Linear Block Coder Decoder

```
1
2
3 %% this is a linear block code main script file%
4 clear
5
6 global P n k;
7
8 n=7;
9 k=4;
10 P=[1 1 0; 0 1 1; 1 0 1;1 1 1]; %% parity matrix of
    size k*(n-k) to be
11 // % selected so that
    the systematic generator
12 // % matrix is
    linearly independent or full rank
13 // % matrix
14
```

```

15 //%(n,k) linear block code where k – no. of input
    data bits and n=no. of o/p
16 //% data bits. code rate=k/n
17 //% x is an input vector containing k bits
18
19 //%This is an linear block encoding function file%
20 function y1=linblkcode(x);
21 global P n k;
22 n=7;
23 k=4;
24 P=[1 1 0; 0 1 1; 1 0 1;1 1 1];
25 //x=[0 1 1 0];
26
27 //G=[ ]; // % Generator matrix k*n
28 G=[eye(k,k) P];
29
30 y1=zeros(1,n);
31 for i=1:k
32     var(i,:)=x(1,i) & G(i,:);
33     var(i,:)=bool2s(var(i,:));
34     y1(1,:)=bitxor(var(i,:),y1(1,:));
35 end
36
37 endfunction
38
39
40 //%This is a linear block syndrome decoding function
    file%
41
42 function x1=linblkdecoder(y)
43 //% here y is recieved vector 7 bits long
44
45 //% (7,4) linear block code
46 global P n k;
47
48
49 //H=[ ]; //% PARITY CHECK MATRIX
50

```

```

51 H=[P' eye((n-k),(n-k))];
52 Ht=H'; %%transpose of H
53
54 S=zeros(1,n-k); %%syndrome of recieved vector x
55 for i=1:n-k
56     S(i)=y(1) & Ht(1,i);
57     S(i)=bool2s(S(i));
58     for j=2:n
59
60         S(i)=bitxor(S(i), bool2s((y(j) & Ht(j,i))));
61     end
62 end
63
64
65
66 %%***SYNDROME LOOK UP TABLE*****
67
68 %*****
69 %%
70 if S==[0 0 0]
71     e=[0 0 0 0 0 0 0];
72     z=bitxor(y,e);
73 end
74
75 if S==[0 0 1]
76     e=[0 0 0 0 0 0 1];
77     z=bitxor(y,e);
78 end
79 if S==[0 1 0]
80     e=[0 0 0 0 0 1 0];
81     z=bitxor(y,e);
82 end
83 if S==[1 0 0]
84     e=[0 0 0 0 1 0 0];
85     z=bitxor(y,e);
86 end
87 if S==[1 1 1]
88     e=[0 0 0 1 0 0 0];

```

```

89     z=bitxor(y,e);
90 end
91 if S==[1 0 1]
92     e=[0 0 1 0 0 0 0];
93     z=bitxor(y,e);
94 end
95 if S==[0 1 1]
96     e=[0 1 0 0 0 0 0];
97     z=bitxor(y,e);
98 end
99 if S==[1 1 0]
100    e=[1 0 0 0 0 0 0];
101    z=bitxor(y,e);
102 end
103
104
105
106
107 x1=z(1,1:k);
108 endfunction
109
110
111 x=[0 1 1 0]; // % input bits to the
    encoder of size 1* k
112 y1=linblkcode(x);// // % y1 is the
    output of linear block encoder
113
114
115 e1=[1 0 0 0 0 0 0]; // % intentionally
    error introduced after
116 // % encoding and
    before sending to decoder (in
117 // % this case pls
    introduce only one bit error)
118 y=bitxor(y1,e1);// % input that will
    be made available to linear
119 // % block decoder
120

```

```

121 x1=linblkdecoder(y) //           % x1 is the output
    of the linear block decoder
122           //           % which will be
    same as x provided that
    here
123           //           % have introduced
    only one bit error

124
125 disp('The information signal=')
126 disp(x)
127 disp('The transmitted encoded signal=')
128 disp(y1)
129 disp('The recieved signal=')
130 disp(y)
131 disp('The denoded signal=')
132 disp(x1)
133
134
135 //—Input variables —//
136
137 //(n,k)==>constants according to the design of
    linear block code
138 //P==>parity matrix of size k*(n-k) to be selected
    so that the systematic generator matrix is
    linearly independent or full rank matrix
139 //x==>information signal—vector

```

---

# Experiment: 8

## FILTER DESIGN USING LAPLACE TRANSFORM

Scilab code Solution 8.1 Filter Design

```
1
2
3
4
5 x=input('Enter the input data sequence');
6 b=input('Enter the co-effitients for the numerator
    polynomial:');
7 a=input('Enter the co-effitients for the denominator
    polynomial:');
8
9 y=filter(b,a,x);
10
11
12 disp('The input signal is=')
13 disp(x)
14 disp('The filtered signal is=')
15 disp(y)
16 //—Input Variables—//
17
```

```
18 //x==>input digital sequence—vector
19 //b==>vecotr
20 //a==>vector
```

---

# Experiment: 9

## SIGNAL MODULATION

Scilab code Solution 9.1 BPSK

```
1 function out_a=BPSK_a(bit,no_shift,shift)
2 if bit == 1
3     out_a = no_shift
4 else
5     out_a = shift
6 end
7
8 endfunction
9
10 t=0:0.001:1;
11 bit_1=0;
12
13 no_shift1=sin(2*%pi*10*t);
14 shift1=sin(2*%pi*10*t+(%pi));
15
16
17
18 out=BPSK_a(bit_1,no_shift1,shift1);
19 plot(t,out)
20
21 t1=1:0.001:2;
```

```
22 bit_2=1;
23
24 out1=BPSK_a(bit_2,no_shift1,shift1)    ;
25 plot(t1,out1)
26
27 xlabel('time')
28 ylabel('Amplitude')
29 //—Input Variables
30 //bit_1 & bit_2==> digital logic 1 or 0
```

---

## Experiment: 10

# APPLICATION OF LINEAR ALGEBRA IN DIGITAL COMMUNICATION

Scilab code Solution 10.1 Rotation of a Vector

```
1 r1=[1 0];
2 r2=[0 1];
3 theta=input("Enter the theta (angle for rotation in
   radian):")
4 rotat=[cos(theta) sin(theta); -sin(theta) cos(theta)
   ];
5
6 R=[r1 ; r2];
7
8 new_R=R*rotat;
9
10 rotated_r1=new_R(1,:);
11 rotated_r2=new_R(2,:);
12
13 disp('The vector r1==')
14 disp(r1)
15 disp('The rotated vector r1=')
```

```
16 disp(rotated_r1)
17
18 disp('The vector r2=')
19 disp(r2)
20
21 disp('The rotated vector r2=')
22 disp(rotated_r2)
23
24
25 //—Input variables —//
26
27 //thera==>constant
```

---