Scilab Manual for
Applied Mathematics for Communication Engineers
by Prof Prarthan Mehta
Others
Dharmsinh Desai University[1]

Solutions provided by
Prof PRARTHAN MEHTA
Others
DHARMSINH DESAI UNIVERSITY

June 20, 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 Experiments

Experiment: 1

TRUNCATION OF A SQUARE WAVE USING FOURIER SERIES EXPANTION

Scilab code Solution 1.1 Truncation of the Square Wave

```
v=1;                                        //Multiplier Voltage
w=15;                                       //Frequency
t=0:0.001:0.99;                             //Incremental Time Stamp
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
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
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
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
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
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
subplot(3,2,1)
    //Deviding the Figure Qindow in Six Subwindows to plot diffenrent components
plot(x1);
    //Plotting the First 2 components of the Fourier Series of a Square Wave
title('Square Wave Contructed with First 2 Components of Fourier Series');        //Assigning a Title of the Figure
xlabel('Samples');
    //Assigning an X-label of the figure
ylabel('Amplitude (V)');
    //Assigning of a Y-label of the figure
subplot(3,2,2)
plot(x2);
```

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

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

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

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

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

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

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

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

```
   //Assigning of a Y-label of the figure

subplot(3,2,5)
plot(x5);
   //Plotting the First 6 components of the Fourier Series of a Square Wave
title('Square Wave Contructed with First 6 Components of Fourier Series');      //Assigning a Title of the Figure
xlabel('Samples');
   //Assigning an X-label of the figure
ylabel('Amplitude (V)');
   //Assigning of a Y-label of the figure

subplot(3,2,6)
plot(x6);
   //Plotting the First 7 components of the Fourier Series of a Square Wave
title('Square Wave Contructed with First 7 Components of Fourier Series');      //Assigning a Title of the Figure
xlabel('Samples');
   //Assigning an X-label of the figure
ylabel('Amplitude (V)');
   //Assigning of a Y-label of the figure


///---Input Parameters---//
//v==> voltage
//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

```


clear


function [y, deriv] = fcn_nr(x)
     y = x^3 + 4*(x)^2 -10;
    deriv = 3*(x)^2 + 8*x;

endfunction

xp = -15:1:15;
n = length(xp);
```

```
for i=1:n
  yp(i) = fcn_nr(xp(i));
end
plot(xp,yp)
xlabel('x')
ylabel('y')
title('Plot of function')



x = input('Enter initial guess of root location: ');

itermax = 10;     // % max # of iterations
iter = 0;
errmax = 0.00001;    //  % convergence tolerance
error1 = 1;


while error1 > errmax & iter < itermax

  iter = iter + 1;
  [f fprime] = fcn_nr(x);
  if fprime == 0
     break;
  end;

  xnew = x - f / fprime;

  error1 = abs((xnew - x)/xnew) * 100; // % find change from previous

  x = xnew     // % set up for next iteration
disp('Iteration No.')
disp(iter)
disp('the estimated value of the variable x is=')
disp(x)
```

```
end

///----Input Parameters----///

//This code will work for different functions of the variable x. hence the user has to modify the function & its derivative accordingly
//Itermax==>maximum number of iteration for which the method has to estimate the value of the variable
//errmax==>maximun tolerable error
```

Experiment: 3

DFT & IDFT OF A GIVEN SEQUENCE

Scilab code Solution 3.1 DFT of a Given Sequence

```
clear
clc

x=input('enter the digital sequence');                    //Input
    the digital sequence
N=length(x);
    //Finding the length of the sequence for N-point
    DFT

for k=1:N
    y(k)=0;
    for n=1:N
        y(k)=y(k)+(x(n)*(exp(-%i*2*%pi*(k-1)*(n-1)/N
            )));                                  //Equation to
             find DFT
    end
end
```

```
disp('The DFT of the given sequence is:')                                    //Displaying the DFT
disp(y)


///--Input Parameters---//

//x==>digital sequence for which DFT has to be found
```

Scilab code Solution 3.2 IDFT of the given responce

```
clear
clc

x=input('Enter the sequence:');                                               //Input the DFT
N=length(x);                                                                   //FInding the length to get N-point IDFT
for n=1:N
    y(n)=0;
    for k=1:N
        y(n)=y(n)+(x(k)*(exp(%i*2*%pi*(k-1)*(n-1)/N)));                        //Equation to find the IDFT
    end
    y(n)=y(n)/N;
end
disp('The IDFT of the given sequence is:')                                   //Displaying the IDFT
disp(y)

```

```
//--Input Parameters---//

//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

```

X=input('Enter the Input data Sequence X[n]:');
H=input('Enter the Response of the System H[n]:');

Y=conv(X,H);

disp('The response of the given system for the given inpur data sequence is Y[n]:');
disp(Y);


//---Input Variables--//
//X==>digital sequence X[n]
//H==>filter impulse response H[n]
```

Experiment: 5

NODE VOLTAGE & MESH CURRENT ANALYSIS

Scilab code Solution 5.1 Node Voltage

```
//number of loops in the networks=2

v1=input('Enter the voltage in loop-1 (in V):');
i2=input('Enter the current through loop-2 (in A):');
r2=input('Enter the resistance in series with V1 in loop-1 (in Ohm):');
r1=input('Enter the resistance between v2 & GND. (a common element bwetween loop-1 & 2) (in Ohm):');

//Teh node voltage at node-2 can be solved as
//temp1=(v2-v1)/r2;
//temp2=(v2/r1)'
//0=temp1+temp2+i2;
temp3=i2*r1*r2;
temp4=v1/r2;
temp5=temp4-temp3;
temp6=temp5*r1*r2;
v2=temp6/(r1+r2);
```

```
disp('Voltage at Node-2 V2=');
disp(v2)


//---Input Variables---//

//v1==>equivalent voltage in loop 1 --constant
//i2==>value of the current source in llop 2--constant
//r2==>value of the resitance in series with V1 in loop 1-- constant
//r1==>value of the resitance in the common branch in loop 1 & loop 2--constant
```

Scilab code Solution 5.2 Mesh Current

```
//number of loops in the networks=2

v1=input('Enter the voltage in loop-1 (in V):');
v2=input('Enter the voltage in loop-2 (in V):');
r1=input('Enter the resistance in series with V1 in loop-1 (in Ohm):');
r2=input('Enter the resistance between v2 & GND. (a common element bwetween loop-1 & 2) (in Ohm):');
r3=input('Enter the resistance in series with V2 in loop-2 (in Ohm):');

//The mesh current can be solved as
//v1=(r1+r2)*i1-r2*i2;
//-v2=-r2*i1+(r2+r3)*i2;

//[v]=[r][i];

//[i]=[v][R]; where [R]=inv([r]);
```

```
v=[v1;v2];

r=[r1+r2 -r2 ; -r2 r2+r3];

//i=[i1;i2];

R=inv(r);


i=R*v;

disp('the current through the loops=')
disp(i)


//--Input variables--//
//v1==>voltage in loop 1--constant
//v2==>voltage in loop 2--constant
//r1==>resitance in loop 1 -- constant
//r2==>resitance in the common branch of loop1 &
    loop 2 --constant
//r3==>resitance in loop 2 -- constant
```

Experiment: 6

STABILITY OF A FEEDBACK SYSTEM IN XCOS

This code can be downloaded from the website wwww.scilab.in This code

can be downloaded from the website wwww.scilab.in

Experiment: 7

IMPLEMENTATION OF CHANNEL CODING-LINEAR BLOCK CODE

Scilab code Solution 7.1 Linear Block Coder Decoder

```


//% this is a linear block code main script file%
clear

global P n k;

n=7;
k=4;
P=[1 1 0; 0 1 1; 1 0 1;1 1 1]; //% parity  matrix of size k*(n-k) to be
//                                      % selected so that the systematic generator
//                                      % matrix is linearly independent or full rank
//                                      % matrix

```

```
//% (n,k) linear block code where k - no. of input data bits and n-no. of o/p
//% data bits. code rate=k/n
//% x is an input vector containing k bits

//%This is an linear block encoding function file%
function y1=linblkcode(x);
global P n k;
n=7;
k=4;
P=[1 1 0; 0 1 1; 1 0 1;1 1 1];
//x=[0 1 1 0];

//G=[ ];     //       % Generator matrix k*n
G=[eye(k,k) P];

y1=zeros(1,n);
for i=1:k
    var(i,:)=x(1,i) & G(i,:);
    var(i,:)=bool2s(var(i,:));
    y1(1,:)=bitxor(var(i,:),y1(1,:));
end

endfunction


//%This is a linear block syndrome decoding function file%

function x1=linblkdecoder(y)
//% here y is recieved vector 7 bits long

//% (7,4) linear block code
global P n k;


//H=[ ]; //% PARITY CHECK MATRIX

```

```
H=[P' eye((n-k),(n-k))];
Ht=H'; //%transpose of H

S=zeros(1,n-k); //%syndrome of recieved vector x
for i=1:n-k
    S(i)=y(1) & Ht(1,i);
    S(i)=bool2s(S(i));
    for j=2:n

        S(i)=bitxor(S(i), bool2s((y(j) & Ht(j,i))));
    end
end



//%%****SYNDROME LOOK UP TABLE************

//%%************************************
//%
if S==[0 0 0]
   e=[0 0 0 0 0 0 0];
    z=bitxor(y,e);
end

if S==[0 0 1]
   e=[0 0 0 0 0 0 1];
    z=bitxor(y,e);
end
if S==[0 1 0]
   e=[0 0 0 0 0 1 0];
    z=bitxor(y,e);
end
if S==[1 0 0]
   e=[0 0 0 0 1 0 0];
    z=bitxor(y,e);
end
if S==[1 1 1]
   e=[0 0 0 1 0 0 0];
```

```
    z=bitxor(y,e);
end
if S==[1 0 1]
   e=[0 0 1 0 0 0 0];
    z=bitxor(y,e);
end
if S==[0 1 1]
   e=[0 1 0 0 0 0 0];
    z=bitxor(y,e);
end
if S==[1 1 0]
   e=[1 0 0 0 0 0 0];
    z=bitxor(y,e);
end




x1=z(1,1:k);
endfunction


x=[0 1 1 0];   //                % input bits to the encoder of size 1* k
y1=linblkcode(x);//           //   % y1 is the output of linear block encoder


e1=[1 0 0 0 0 0 0];           // % intentionally error introduced after
//                               % encoding and before sending to decoder (in
  //                             % this case pls introduce only one bit error)
y=bitxor(y1,e1);//               % input that will be made available to linear
                  //             % block decoder

```

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

disp('The information signal=')
disp(x)
disp('The transmitted encoded signal=')
disp(y1)
disp('The recieved signal=')
disp(y)
disp('The denoded signal=')
disp(x1)


//--Input variables--//

//(n,k)==>constants according to the design of linear block code
//P==>parity  matrix of size k*(n-k) to be selected so that the systematic generator matrix is linearly independent or full rank matrix
//x==>information signal--vector
```

Experiment: 8

FILTER DESIGN USING LAPLACE TRANSFORM

Scilab code Solution 8.1 Filter Design

```
x=input('Enter the input data sequence');
b=input('Enter the co-effitients for the numerator
    polynomial:');
a=input('Enter the co-effitients for the denominator
     polynomial:');

y=filter(b,a,x);


disp('The input signal is=')
disp(x)
disp('The filtered signal is=')
disp(y)
//--Input Variables--//

```

```
//x==>input digital sequence--vector
//b==>vecotr
//a==>vector
```

Experiment: 9

SIGNAL MODULATION

Scilab code Solution 9.1 BPSK

```
function out_a=BPSK_a(bit,no_shift,shift)
if bit == 1
    out_a = no_shift
else
    out_a = shift
end

endfunction

t=0:0.001:1;
bit_1=0;

no_shift1=sin(2*%pi*10*t);
shift1=sin(2*%pi*10*t+(%pi));



out=BPSK_a(bit_1,no_shift1,shift1);
plot(t,out)

t1=1:0.001:2;
```

```
bit_2=1;

out1=BPSK_a(bit_2,no_shift1,shift1)    ;
plot(t1,out1)

xlabel('time')
ylabel('Amplitude')
//--Input Variables
//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

```
r1=[1 0];
r2=[0 1];
theta=input("Enter the theta (angle for rotation in radian):")
rotat=[cos(theta) sin(theta); -sin(theta) cos(theta)];

R=[r1 ; r2];

new_R=R*rotat;

rotated_r1=new_R(1,:);
rotated_r2=new_R(2,:);

disp('The vector r1==')
disp(r1)
disp('The rotated vector r1=')
```

```
disp(rotated_r1)

disp('The vector r2=')
disp(r2)

disp('The rotated vector r2=')
disp(rotated_r2)


//--Input variables--//

//thera==>constant
```