Scilab Textbook Companion for Wireless Communications Principles and Practices
by T. S. Rappaport[1]

Created by
Priyanka Gavadu Patil
Wireless communication
Others
Pillai HOC College Of Engineering & Technology
College Teacher
None
Cross-Checked by
Spandana

July 31, 2019

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: Wireless Communications Principles and Practices

Author: T. S. Rappaport

Publisher: Pearson, New Delhi

Edition: 2

Year: 2002

ISBN: 81-7808-648-4

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes

List of Figures

Chapter 2

Appendix B Noise figure calculations for link budgets

Scilab code Exa 2.1 `To determine SNR at the detector output stage`

```
// Example no B.1
// To determine SNR at the detector output stage
// Page no. 613

clc;
clear all;

// Given data
SNRin=20;                                              //
    SNR at the receiver antenna input terminal in dB
F=6;                                                   //
    Noise figure in dB

// SNR at the detector output stage
SNRout=SNRin-F;                                       //
   SNR at the detector output stage in dB

```

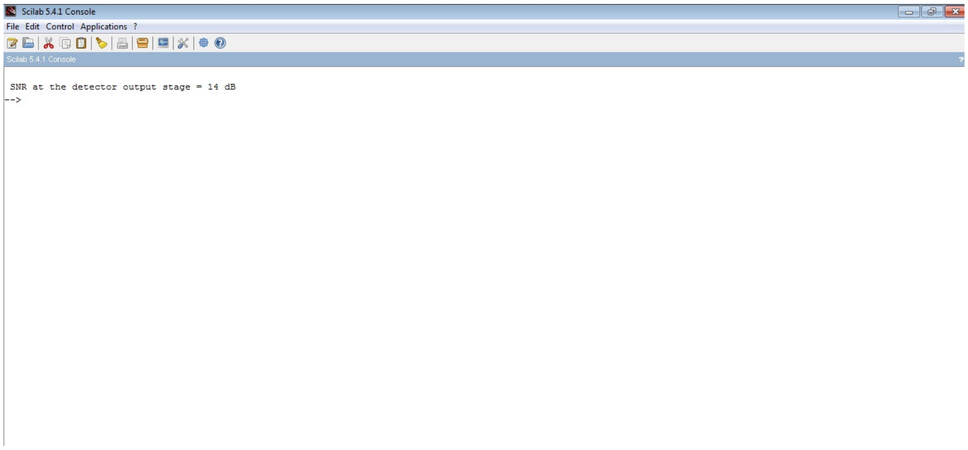

Figure 2.1: To determine SNR at the detector output stage

```
// Displaying the result in command window
printf('\n SNR at the detector output stage = %0.0f dB',SNRout);
```

Scilab code Exa 2.2 `To compute noise figure of mobile receiver system`

```
//Example no B.2
//To compute noise figure of mobile receiver system
//Page no. 613

clc;
clear all;

//Given data
F1=3;                                          // Coaxial cable loss in dB
F1=10^(F1/10);                                 // Coaxial cable loss
```

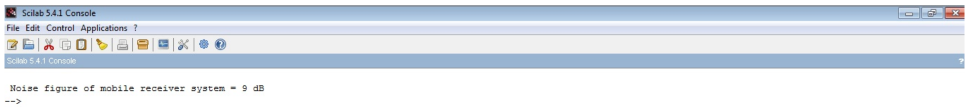

Figure 2.2: To compute noise figure of mobile receiver system

```
F2=6;                                //Noise figure of phone in dB
F2=10^(F2/10);                       //Noise figure of phone

Fsys=F1+((F2-1)/0.5);                //Noise figure of mobile receiver system
Fsys=10*log10(Fsys);                 //Noise figure of mobile receiver system in dB

// Displaying the result in command window
printf('\n Noise figure of mobile receiver system = %0.0f dB',Fsys);
```

Scilab code Exa 2.3 To determine average output thermal noise power

```
// Example no B.3
// To determine average output thermal noise power
// Page no. 614
```

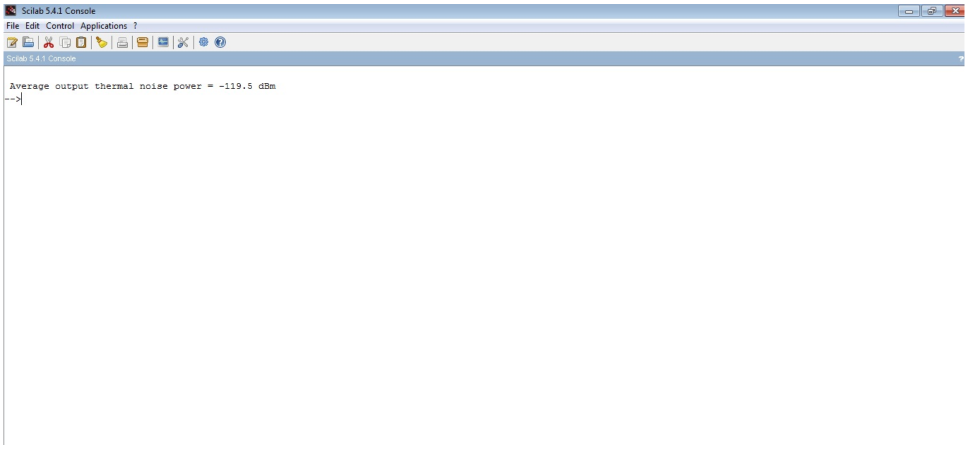

Figure 2.3: To determine average output thermal noise power

```

clc;
clear all;

// Given data
T0=300;
    // Ambient room temperature in K
Fsys=8;
    // Noise figure of the system
Tant=290;
    // Effective temperature of antenna in K
K=1.38*10^-23;
    // Boltzmann's constant in J/K
B=30000;
    // Effective bandwidth in Hz

Te=(Fsys-1)*T0;
```

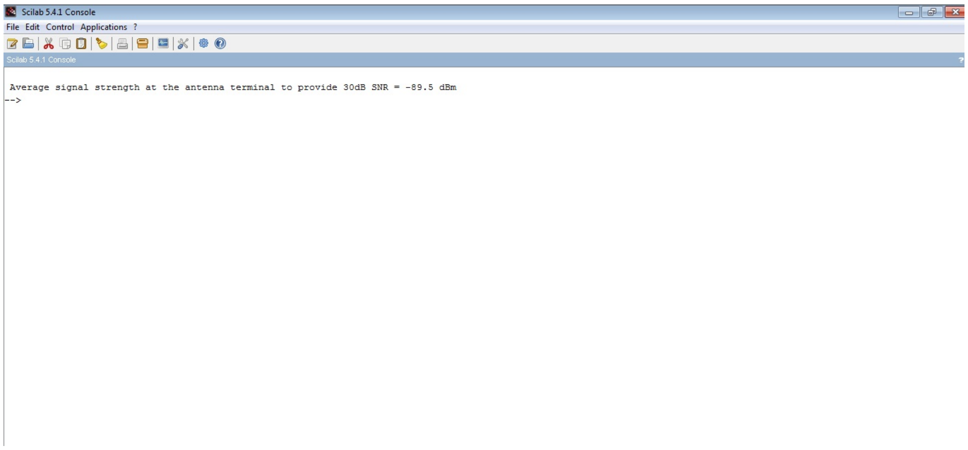

Figure 2.4: To determine average signal strength at the antenna terminal

```
    // Effective noise temperature in K
Ttotal=Tant+Te;
    // Overall system noise temperature in K

// To determine average output thermal noise power
Pn=(1+(Ttotal/T0))*K*T0*B;                                //
      Average output thermal noise power in W
Pn=10*log10(Pn/(10^-3));
    // Average output thermal noise power in dBm

// Displaying the result in command window
printf('\n Average output thermal noise power = %0.1
    f dBm',Pn);
```

Scilab code Exa 2.4 To determine average signal strength at the antenna terminal

```
// Example no B.4
// To determine average signal strength at the
    antenna terminal
// Page no. 614

clc;
clear all;

// Given data
Pn=-119.5;
    // Average output thermal noise power in dBm
SNR=30;
    // SNR at the receiver output in dB

// To determine average signal strength at the
    antenna terminal to provide 30dB SNR
Ps=SNR+Pn;
    // Average signal strength at the antenna
    terminal

// Displaying the result in command window
printf('\n Average signal strength at the antenna
    terminal to provide 30dB SNR = %0.1f dBm',Ps);
```

Chapter 3

The cellular concept system design fundamentals

Scilab code Exa 3.1 `To compute the number of channels available per cell for four`

```
// Example 3.1
// To compute the number of channels available per
   cell for a)four-cell reuse system a)seven-cell
   reuse system a)12-cell reuse system
// Page No.61

clc;
clear;

// Given data
B=33*10^6;

   // Total bandwidth allocated to particular FDD
   system in Hz
Bc=25*10^3;

   // Bandwidth per channel in Hz
```

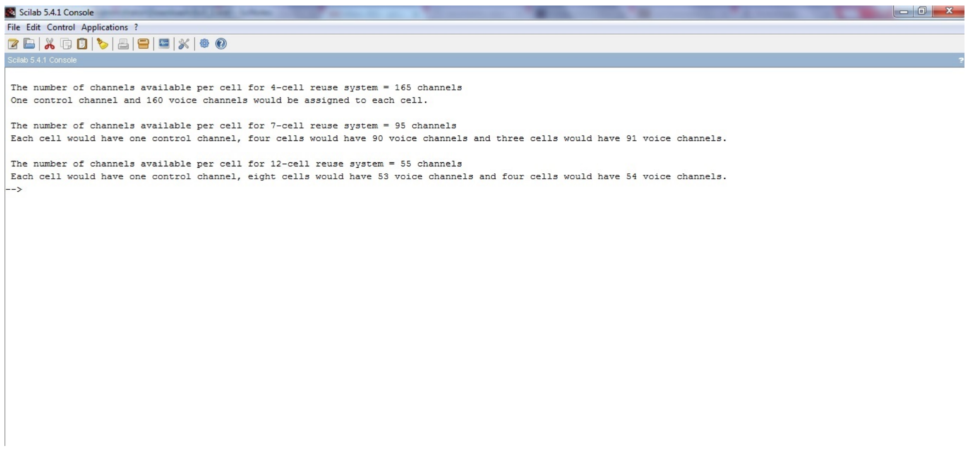

Figure 3.1: To compute the number of channels available per cell for four cell reuse system and seven cell reuse system and twelve cell reuse system

```
Nc=2;
    // Number of simplex channels
Bc=Bc*Nc;
    // Channel bandwidth in Hz

Ntotal=B/Bc;
                                                        //
     Total number of channels

//a) To compute the number of channels available per
     cell for four-cell reuse system
N=4;
    // frequency reuse factor
chpercell=Ntotal/N;
                                                  // number
     of channels available per cell for four-cell
    reuse system

// Displaying the result in command window
```

```
printf('\n The number of channels available per cell for 4-cell reuse system = %0.0f channels', chpercell);
printf('\n One control channel and 160 voice channels would be assigned to each cell.');

// b) To compute the number of channels available per cell for seven-cell reuse system
N=7;                                                   // frequency reuse factor
chpercell=ceil(Ntotal/N);                              // number of channels available per cell for seven-cell reuse system

// Answer is varrying due to round-off error

// Displaying the result in command window
printf('\n \n The number of channels available per cell for 7-cell reuse system = %0.0f channels', chpercell);
printf('\n Each cell would have one control channel, four cells would have 90 voice channels and three cells would have 91 voice channels.');

// c) To compute the number of channels available per cell for 12-cell reuse system
N=12;                                                  // frequency reuse factor
chpercell=Ntotal/N;                                    // number of channels available per cell for seven-cell reuse system

// Displaying the result in command window
printf('\n \n The number of channels available per
```

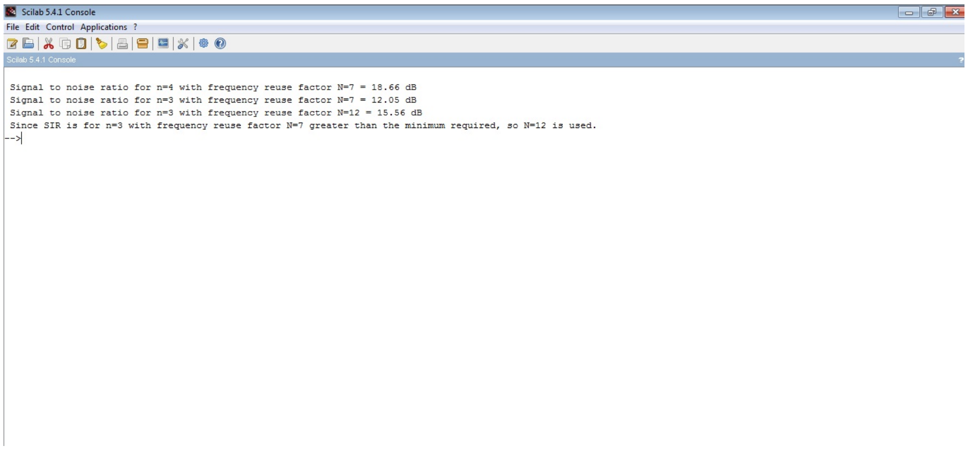

Figure 3.2: To find frequency reuse factor for path loss exponent 4 and 3

```
    cell for 12-cell reuse system = %0.0f channels',
    chpercell);
printf('\n Each cell would have one control channel,
    eight cells would have 53 voice channels and
    four cells would have 54 voice channels.');
```

Scilab code Exa 3.2 To find frequency reuse factor for path loss exponent 4 and 3

```
// Example 3.2
// To find frequency reuse factor for path loss
    exponent (n) a)n=4 b)n=3
// Page No.72

clc;
clear;

// Given data
SIdB=15;                              // Signal to
    interference(dB)
```

```
io=6;                          // Number of cochannel cell

// For n=4
n1=4;                          // Path loss exponent
N1=7;                          // First consideration: frequency reuse factor N=7
DR1=sqrt(3*N1);                // Co-channel reuse ratio
si1=(1/io)*(DR1)^n1;           // Signal to interference
sidB1=10*log10(si1);           // Signal to interference(dB)

// For n=3
n2=3;                          // Path loss exmponent
si=(1/io)*(DR1)^n2;            // Signal to interference for first consideration: frequency reuse factor N=7
sidB=10*log10(si);             // Signal to interference(dB)

N2=12;                         // second consideration: frequency reuse factor N=12 since sidB<SIdB
DR2=sqrt(3*N2);                // Co-channel reuse ratio
si2=(1/io)*(DR2)^n2;           // Signal to interference
sidB2=10*log10(si2);           // Signal to interference(dB)

// Displaying the result in command window
printf('\n Signal to noise ratio for n=4 with frequency reuse factor N=7 = %0.2f dB',sidB1);
printf('\n Signal to noise ratio for n=3 with frequency reuse factor N=7 = %0.2f dB',sidB);
printf('\n Signal to noise ratio for n=3 with frequency reuse factor N=12 = %0.2f dB',sidB2);
```

Figure 3.3: To find number of users for Number of channels 1 and 5 and 10 and 20 and 100

```
printf('\n Since SIR is for n=3 with frequency reuse
    factor N=7 greater than the minimum required, so
    N=12 is used.');
```

Scilab code Exa 3.4 To find number of users for Number of channels 1 and 5 and 10

```
// Example 3.4
// To find number of users for Number of channels (C
   ) a)C=1 b)C=5 c)C=10 d)C=20 e)C=100
// Page No.80

clc;
clear;

// Given data
GOS=0.005;                //Grade of Service
Au=0.1;                   // Traffic intensity per user

```

```
// a)To find number of users for C=1
C1=1;                    // Number of channels
A1=0.005;                // Total traffic intensity from Erlangs B chart
U1=(A1/Au);              // Number of users
U1=1;                    // Since one user could be supported on one channel

// b)To find number of users for C=5
C2=5;                    // Number of channels
A2=1.13;                 // Total traffic intensity from Erlangs B chart
U2=round(A2/Au);         // Number of users

// c)To find number of users for C=10
C3=10;                   // Number of channels
A3=3.96;                 // Total traffic intensity from Erlangs B chart
U3=round(A3/Au);         // Number of users

// Answer is varrying due to round off error

// d)To find number of users for C=20
C4=20;                   // Number of channels
A4=11.10;                // Total traffic intensity from Erlangs B chart
U4=round(A4/Au);         // Number of users

// Answer is varrying due to round off error

// e)To find number of users for C=100
C5=100;                  // Number of channels
A5=80.9;                 // Total traffic intensity from Erlangs B chart
U5=round(A5/Au);         // Number of users

// Displaying the result in command window
printf('\n Total number of users for 1 channel = %0
```

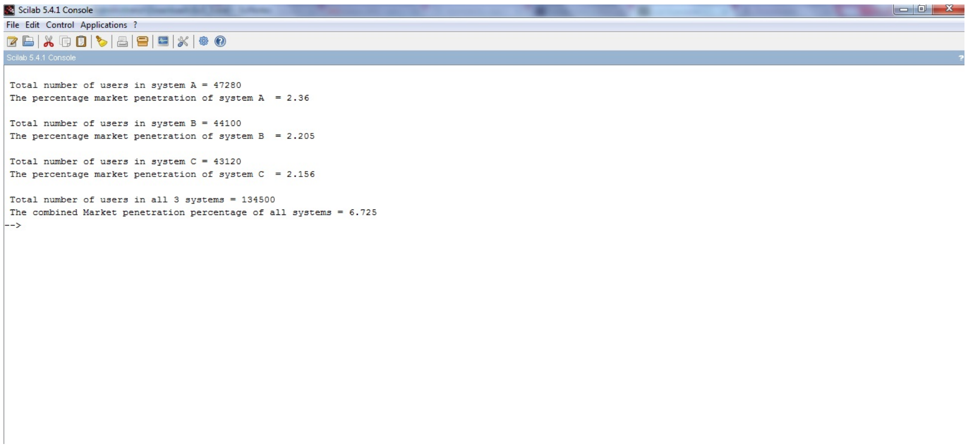

Figure 3.4: To find number of users for system A and system B and system C

```
.0f',U1);
printf('\n Total number of users for 5 channel = %0.0f',U2);
printf('\n Total number of users for 10 channel = %0.0f',U3);
printf('\n Total number of users for 20 channel = %0.0f',U4);
printf('\n Total number of users for 100 channel = %0.0f',U5);
```

Scilab code Exa 3.5 To find number of users for system A and system B and system C

```
// Example 3.5
// To find number of users for a)system A b)system B c)system C
// Page No.83

clc;
```

```
clear;

// Given data
GOS=0.02;                                   // Grade of Service (Probability of bloacking)
lamda=2;                                    // Average calls per hour
H=(3/60);                                   // Call duration in seconds

Au=lamda*H;                                 // Traffic intensity per user

// a)To find number of users for System A
C1=19;                                     // Number of channels used
A1=12;                                     // Traffic intensity from Erlang B chart
U1=round(A1/Au);                           // Number of users per cell
cells1=394;
TU1=U1*cells1;                             // Total number of users
MP1=TU1/(2*10^6)*100;                      // Market penetration percentage

// b)To find number of users for System B
C2=57;                                    // No. of channels used
A2=45;                                    // Traffic intensity from Erlang B chart
U2=round(A2/Au);                          // Number of users per cell
cells2=98;
TU2=U2*cells2;                            // Total no. of users
MP2=TU2/(2*10^6)*100;                     // Market penetration percentage
```

```
// c)To find number of users for System C
C3=100;                          // Number of channels used
A3=88;                           // traffic intensity from Erlang B chart
U3=round(A3/Au);                 // Number of users per cell
cells3=49;
TU3=U3*cells3;                   // Total no. of users
MP3=TU3/(2*10^6)*100;            // Market penetration percentage

TU=TU1+TU2+TU3;                  // Total number of users in all 3 systems
MP=TU/(2*10^6)*100;              // Combined Market penetration percentage

// Displaying the result in command window
printf('\n Total number of users in system A = %0.0f',TU1);
printf('\n The percentage market penetration of system A  = %0.2f',MP1);
printf('\n \n Total number of users in system B = %0.0f',TU2);
printf('\n The percentage market penetration of system B  = %0.3f',MP2);
printf('\n \n Total number of users in system C = %0.0f',TU3);
printf('\n The percentage market penetration of system C  = %0.3f',MP3);
printf('\n \n Total number of users in all 3 systems = %0.0f',TU);
printf('\n The combined Market penetration percentage of all systems = %0.3f',MP);
```

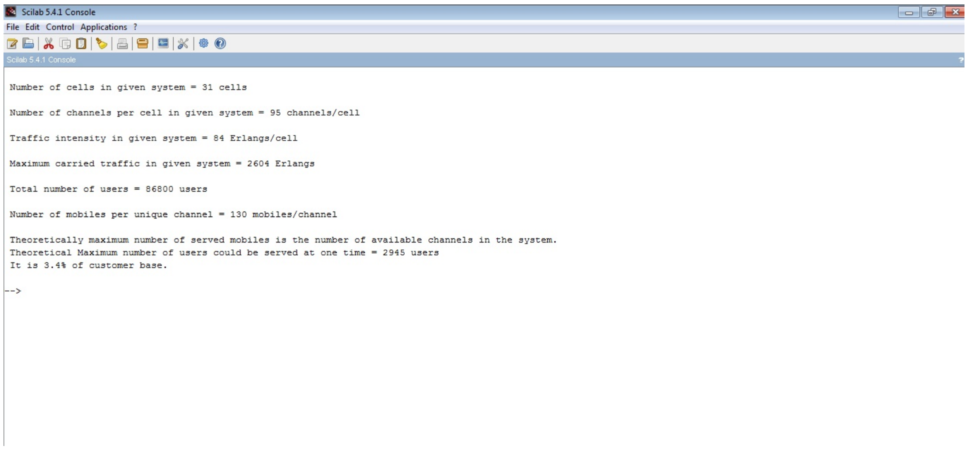

Figure 3.5: To find Number of cells in given area Number of channels per cell Traffic intensity per cell Maximum carried traffic Total no of users for 2percent GOS and Number of mobiles per unique channel and Maximum number of users could be served at one time

Scilab code Exa 3.6 `To find Number of cells in given area Number of channels per c`

```
// Example 3.6
// To find a)Number of cells in given area b)Number
   of channels/cell c)Traffic intensity per cell d)
   Maximum carried traffic e)Total number of users
   for 2% GOS f) Number of mobiles per unique
   channel   g)Maximum number of users could be
   served at one time
// Page No.84

clc;
clear;

// Given data
```

```
Area=1300;                                  // Total coverage area in m^2
R=4;                                        // Radius of cell in m
N=7;                                        // Frequecy reuse factor
S=40*10^6;                                  // Allocated spectrum in Hz
Ch=60*10^3;                                 // Channel width in Hz

// a)Number of cells
CA=2.5981*R^2;                              // Area of hexagonal cell in m^2
Nc=round(Area/CA);                          // Number of cells

// Displaying the result in command window
printf('\n Number of cells in given system = %0.0f cells',Nc);

// b)Number of channels/cell
C1=round(S/(Ch*N));                          // Number of channels

// Displaying the result in command window
printf('\n \n Number of channels per cell in given system = %0.0f channels/cell',C1);

// c) Traffic intensity per cell
C1=95;                                       // Number of channels from b)
GOS=0.02;                                    // Grade of service
A=84;                                        // Traffic intensity from Erlang B chart

// Displaying the result in command window
```

```
printf('\n \n Traffic intensity in given system = %0.0f Erlangs/cell',A);

// d)Maximum carried traffic
traffic=Nc*A;                              // Maximum carried traffic

// Displaying the result in command window
printf('\n \n Maximum carried traffic in given system = %0.0f Erlangs',traffic);

// e)Total number of users for 2% GOS
trafficperuser=0.03;                       // Given traffic per user
U=traffic/trafficperuser;                  // Total number of users

// Displaying the result in command window
printf('\n \n Total number of users = %0.0f users',U);

// f) Number of mobiles per unique channel
C=666;                                     // Number of channels
mobilesperchannel=round(U/C);              // Number of mobiles per unique channel

// Displaying the result in command window
printf('\n \n Number of mobiles per unique channel = %0.0f mobiles/channel',mobilesperchannel);

// g)Maximum number of users could be served at one time
printf('\n \n Theoretically maximum number of served mobiles is the number of available channels in the system.')
C=C1*Nc;                                   // Maximum number of users could be served at one time
```

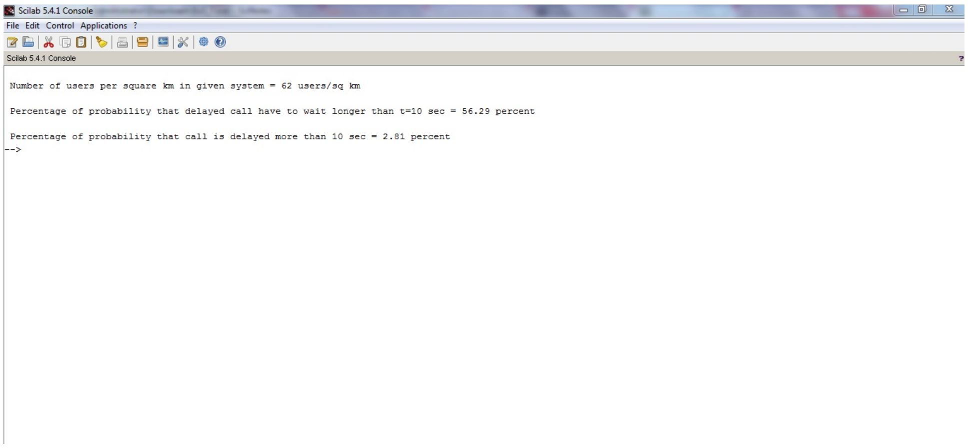

Figure 3.6: To find number of users per square km probability that delayed call have to wait longer than 10sec and probability that call is delayed more than 10 sec

```

// Displaying the result in command window
printf('\n Theoretical Maximum number of users could
    be served at one time = %0.0f users',C);
disp('It is 3.4% of customer base.');
```

Scilab code Exa 3.7 `To find number of users per square km probability that delayed`

```
// Example 3.7
// To find a)number of users per square km b)
   probability that delayed call have to wait longer
    than t=10sec c)probability that call is delayed
   more than 10 sec
// Page 85

clc;
clear;
```

```

// Given data
R=1.387;                                    // Radius of cell in m
Area=2.598*R^2;                             // Area of hexagonal cell in m^2
cellpercluster=4;                           // Number of cells/cluster
channels=60;                                // Number of channels

channelspercell=channels/cellpercluster;    // Number of channels per cell

// a)To find number of users per square km
A=0.029;                                    // Traffic intensity per user
delayprob=0.05;                             // Grade of service
traffic=9;                                  // Traffic intensity from Erlang chart C
U1=traffic/A;                               // Total number of users in 5sq.km.
U=round(U1/Area);                           // Number of users per square km

// Displaying the result in command window
printf('\n Number of users per square km in given system = %0.0f users/sq km',U);

// b)To find the probability that delayed call have to wait longer than t=10sec
lambda=1;                                   // Holding time
H1=A/lambda;                                // Duration of call
H=H1*3600;                                  // Duration of call in second
```

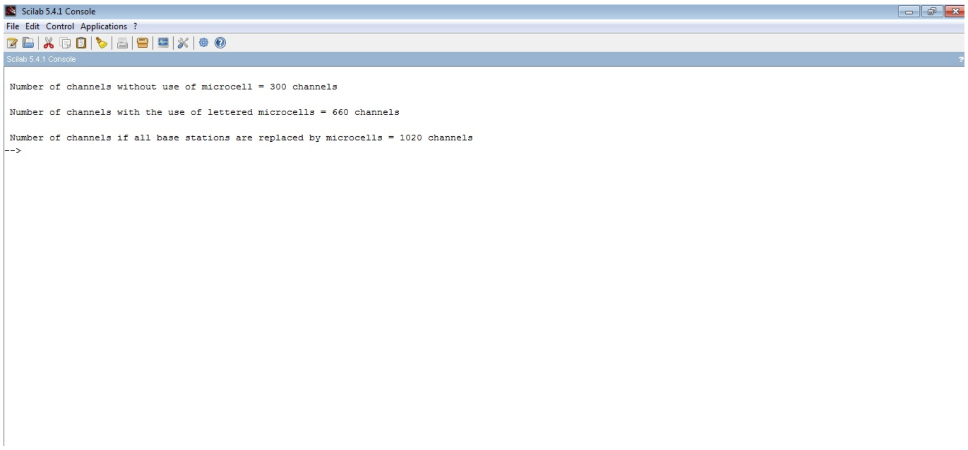

Figure 3.7: To find number of channels in 3 km by 3 km square centered around A in given Figure for without use of microcell with the use of lettered microcells and all base stations are replaced by microcells

```
t=10;
Pr=exp(-(channelspercell-traffic)*t/H)*100; // probability that delayed call have to wait longer than t=10sec.

// Displaying the result in command window
printf('\n \n Percentage of probability that delayed call have to wait longer than t=10 sec = %0.2f percent',Pr);

// c)To find the probability that call is delayed more than 10 sec
Pr10=delayprob*Pr;                                  // probability that call is delayed more than 10 sec

// Displaying the result in command window
printf('\n \n Percentage of probability that call is delayed more than 10 sec = %0.2f percent',Pr10);
```

Scilab code Exa 3.8 `To find number of channels in 3 km by 3 km square centered aro`

```
// Example 3.8
// To find number of channels in 3 km by 3 km square centered around A in Figure 3.9 for a)without use of microcell b)with the use of lettered microcells c)all base stations are replaced by microcells
// Page 89

clc;
clear;

// Given data
R=1;                    // Cell radius in km
r=0.5;                  // Micro-cell radius in km
Nc=60;                  // Number of channels in base station

// a)To find number of channels without use of microcell
Nb1=5;                  // Number of base stations in given area
N1=Nb1*Nc;              // Number of channels without use of microcell

// b)To find number of channels with the use of
```

```
    lettered microcells
Nb2=6;
    // Number of lettered microcells
Nb2=Nb1+Nb2;
    // Total number of base stations in given area
N2=Nb2*Nc;
    // Number of channels with the use of lettered
    microcells

// c)To find number of channels if all base stations
     are replaced by microcells
Nb3=12;
    // Number of all the microcells
Nb3=Nb1+Nb3;
    // Total number of base stations in given area
N3=Nb3*Nc;
    // Number of channels if all base stations are
    replaced by microcells

// Displaying the result in command window
printf('\n Number of channels without use of
    microcell = %0.0f channels',N1);
printf('\n \n Number of channels with the use of
    lettered microcells = %0.0f channels',N2);
printf('\n \n Number of channels if all base
    stations are replaced by microcells = %0.0f
    channels',N3);
```

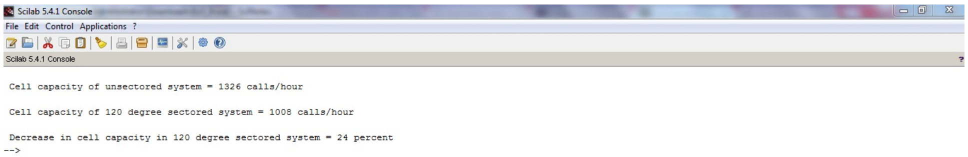

Figure 3.8: To analyze trunking efficiency capacity of sectoring and unsectoring

Scilab code Exa 3.9 `To analyze trunking efficiency capacity of sectoring and unsec`

```
// Example 3.9
// To analyze trunking efficiency capacity of
   sectoring and unsectoring
// Page 92

clc;
clear all;

// Given data
H=2/60;

   // Average call duration in hour
GOS=0.01;

   // Probability of blocking

// Unsectored system
C1=57;

   // Number of traffic channels per cell in
```

```
unsectored system
A=44.2;        // Carried traffic in unsectored system
calls1=1326;   // Number of calls per hour in unsectored system from Erlangs B table

// 120 degree sectored system
C2=C1/3;       // Number of traffic channels per antenna sector in 120 degree sectored system
calls2=336;    // Number of calls per hour in 120 degree sectored system from Erlangs B table
Ns1=3;         // Number of sectors
capacity=Ns1*calls2;          // Cell capacity or number of calls handled by system per hour

dif=calls1-capacity;          // decrease in cell capacity in 120 degree sectored system
percentdif=(dif/calls1)*100;  // decrease in cell capacity in 120 degree sectored system in percentage

// Displaying the result in command window
printf('\n Cell capacity of unsectored system = %0.0f calls/hour',calls1);
```

```
printf('\n \n Cell capacity of 120 degree sectored
    system = %0.0f calls/hour',capacity);
printf('\n \n Decrease in cell capacity in 120
    degree sectored system = %0.0f percent',
    percentdif);
```

Chapter 4

Mobile radio propagation large scale path loss

Scilab code Exa 4.1 To find far field distance for antenna with maximum dimensions

```
// Example 4.1
// To find far field distance for antenna with
   maximum dimensions and operating frequency
// Page No.109

clc;
clear all;

// Given data
D=1;                    // Maximum dimension in m
f=900*10^6;             // Operating frequency in Hz
C=3*10^8;               // Speed of light in m/sec

lambda=C/f;             // Carrier wavelength in m

// To find far field distance
df=(2*D^2)/lambda;      //Far field distance
```

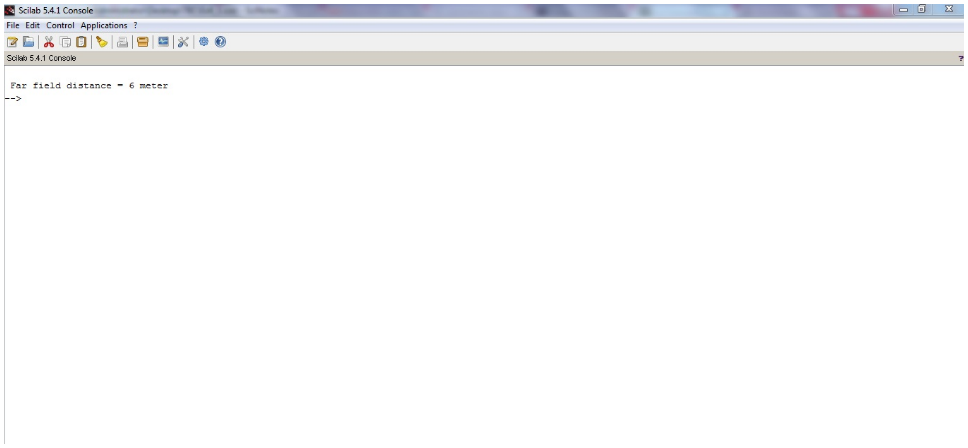

Figure 4.1: To find far field distance for antenna with maximum dimensions and operating frequency

```

//Displaying the result in command window
printf('\n Far field distance = %0.0f meter',df);
```

Scilab code Exa 4.2 `To find transmitter power in dBm and Transmitter power in dBW`

```
// Example 4.2
// To find a)transmitter power in dBm b)Transmitter power in dBW and the received power of antenna in dBm at free space distance of 100m from antenna and 10km
// Page No.109

clc;
clear all;

// Given data
```

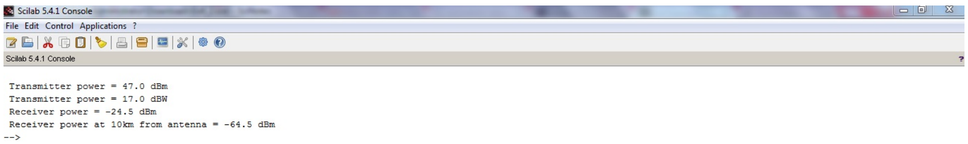

Figure 4.2: To find transmitter power in dBm and Transmitter power in dBW and the received power of antenna in dBm at free space distance of 100m from antenna and 10km

```
Pt=50;                  // Transmitter power in W
fc=900*10^6;            // Carrier frequency in Hz
C=3*10^8;               // Speed of light in m/s

//a) Transmitter power in dBm
PtdBm=round(10*log10(Pt/(1*10^(-3))));    // Transmitter power in dBm

// Displaying the result in command window
printf('\n Transmitter power = %0.1f dBm',PtdBm);

//b) Transmitter power in dBW
PtdBW=round(10*log10(Pt/1));     // Transmitter power in dBW

// Displaying the result in command window
printf('\n Transmitter power = %0.1f dBW',PtdBW);

```

```
// To find receiver power at 100m
Gt=1;                                            //Transmitter gain
Gr=1;                                            //Receiver gain
d=100;                                           //Free space distance from antenna in m
L=1;                                             //System loss factor since no loss in system
lambda=C/fc;                                     //Carrier wavelength in m
Pr=(Pt*Gt*Gr*lambda^2)/((4*%pi)^2*d^2*L);        //Receiver power in W
PrdBm=10*log10(Pr/10^(-3));                      //Receiver power in dBm

//Displaying the result in command window
printf('\n Receiver power = %0.1f dBm',PrdBm);

//For Pr(10km)
d0=100;                                          //Reference distance
d=10000;                                         //Free space distance from antenna
Pr10km=PrdBm+20*log10(d0/d);                     //Received power at 10km from antenna in dBm

//Displaying the result in command window
printf('\n Receiver power at 10km from antenna = %0.1f dBm',Pr10km);
```

Scilab code Exa 4.3 To find power at receiver Magnitude of E field at receiver and

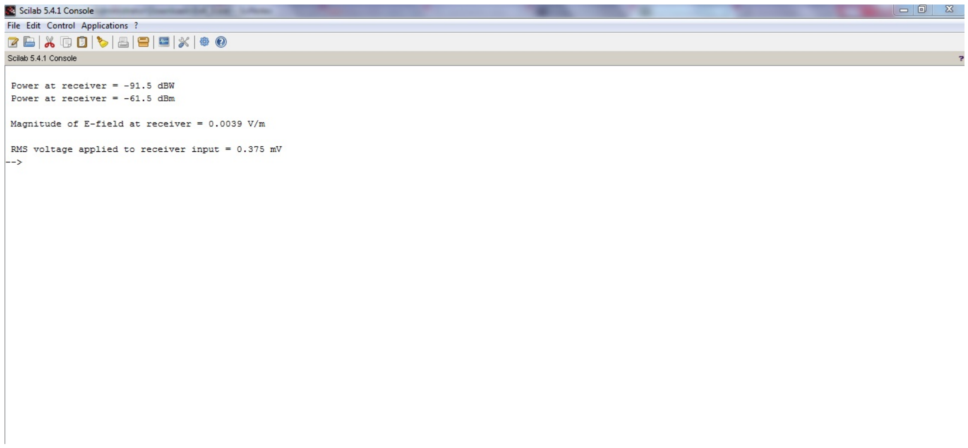

Figure 4.3: To find power at receiver Magnitude of E field at receiver and rms voltage applied to receiver input

```
// Example 4.3
// To find a)power at receiver b)magnitude of E-
   field at receiver c)rms voltage applied to
   receiver input
// Page no. 112

clc;
clear all;

// Given data
Pt=50;
   // Transmitter power in Watt
fc=900*10^6;
   // Carrier frequency in Hz
Gt=1;
   // Transmitter antenna gain
Gr=2;
   // Receiver antenna gain
```

```
Rant=50; // Receiver antenna resistance in ohm

// a)Power at receiver
d=10*10^3; // Distance from antenna in meter
lambda=(3*10^8)/fc; // Carrier wavelength in meter
Prd1=10*log10((Pt*Gt*Gr*lambda^2)/((4*%pi)^2*d^2)); // Power at transmitter in dBW
Prd=10*log10(((Pt*Gt*Gr*lambda^2)/((4*%pi)^2*d^2))/(10^-3)); // Power at transmitter in dBm

// Displaying the result in command window
printf('\n Power at receiver = %0.1f dBW',Prd1);
printf('\n Power at receiver = %0.1f dBm',Prd);

// b)Magnitude of E-field at receiver
Ae=(Gr*lambda^2)/(4*%pi); // Aperture gain
Pr=10^(Prd1/10); // Receiver power in W
E=sqrt((Pr*120*%pi)/Ae); // Magnitude of E-field at receiver

// Displaying the result in command window
printf('\n \n Magnitude of E-field at receiver = %0.4f V/m',E);

// c)rms voltage applied to receiver input
```

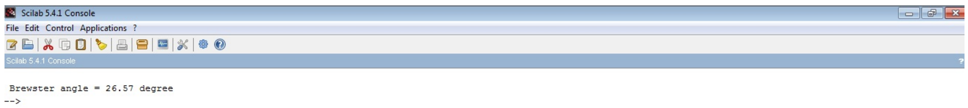

Figure 4.4: To calculate the Brewster angle

```
Vant=sqrt(Pr*4*Rant)*10^3;
    // rms voltage applied to receiver input
//Answer is varrying due to round-off error

//Displaying the result in command window
printf('\n \n RMS voltage applied to receiver input
   = %0.3f mV',Vant);
```

Scilab code Exa 4.5 To calculate the Brewster angle

```
// Example no. 4.5
// To calculate the Brewster angle
// Page no. 119

clc;
clear all;

// Given data
```

Figure 4.5: To find the length and effective aperture of receiving antenna and the received power at mobile

```
Er=4;                                   //
   Permittivity
x=sqrt((Er-1)/(Er^2-1));                // Sine of
   brewster angle
theta=asind(x);                         // Brewster
   angle
//Answer is varrying due to round off error

// Displaying the result in command window
printf('\n Brewster angle = %0.2f degree',theta);
```

Scilab code Exa 4.6 `To find the length and effective aperture of receiving antenna`

```
// Example no 4.6
// To find a)the length and effective aperture of
   receiving antenna b)the received power at mobile
// Page no. 125

```

```
clc;
clear;

// Given data
d=5*10^3;                                   // distance of mobile from base station in m
E0=1*10^-3;                                 // E-field at 1Km from transmitter in V/m
d0=1*10^3;                                  // Distance from transmitter in m
f=900*10^6;                                 // Carrier frequency used for the system in Hz
c=3*10^8;                                   // Speed of ligth in m/s
gain=2.55;                                  // Gain of receiving antenna in dB
G=10^(gain/10);                             // Gain of receiving antenna

// a)To find the length and effective aperture of receiving antenna
lambda=c/f;                                 // Wavelength
L=lambda/4;                                 // Length of antenna
Ae=(G*lambda^2)/(4*%pi);                    // Effective aperture of receiving antenna

```

```
// Displaying the result in command window
printf('\n Length of antenna = %0.4f m',L);
printf(' = %0.2f cm',L*10^2);
printf('\n Effective aperture of receiving antenna = %0.3f m^2',Ae);

// b)To find the received power at mobile
// Given data
ht=50;                                  // Heigth of transmitting antenna
hr=1.5;                                 // Heigth of receiving antenna
ERd=(2*E0*d0*2*%pi*ht*hr)/(d^2*lambda);           // Electic field at distance d in V/m
Prd=((ERd^2/377)*Ae);                                    // The received power at mobile in W
PrddB=10*log10(Prd);                                      // The received power at mobile in dBW
PrddBm=10*log10(Prd/10^-3);                          // The received power at mobile in dBm
Prd=((ERd^2/377)*Ae)*10^13;                          // The received power at mobile in 10^-13W

// Displaying the result in command window
printf('\n \n The received power at mobile = %0.1f X 10^-13 W',Prd);
printf(' = %0.2f dBW',PrddB);
printf(' = %0.2f dBm',PrddBm);
```

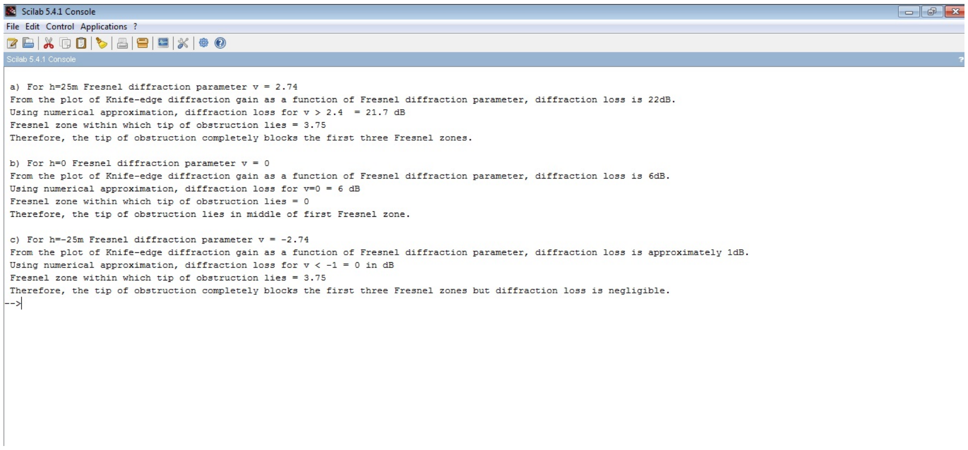

Figure 4.6: To compute diffraction loss and identify Fresnel zone within which tip of obstruction lies for height 25 m 0 m and negative 25m

Scilab code Exa 4.7 `To compute diffraction loss and identify Fresnel zone within w`

```
// Example no 4.7
// To compute diffraction loss and identify Fresnel
   zone within which tip of obstruction lies for a)h
   =25m b)h=0 c)h=-25m
// Page no. 132

clc;
clear;

// Given data
lambda=1/3;
   // Wavelength in meter
d1=1*10^3;
   // Distance between transmitter and obstructing
   screen in m
```

```
d2=1*10^3;    // Distance between receiver and obstructing screen in m

// a) For h=25m
h=25;    // Effective heigth of obstruction screen in m
v=h*sqrt((2*(d1+d2))/(lambda*d1*d2));    // Fresnel diffraction parameter
printf('\n a) For h=25m Fresnel diffraction parameter v = %0.2f',v);
printf('\n From the plot of Knife-edge diffraction gain as a function of Fresnel diffraction parameter, diffraction loss is 22dB.');
Gd=-20*log10(0.225/v);                                    // Diffraction loss for v>2.4 in dB
printf('\n Using numerical approximation, diffraction loss for v > 2.4  = %0.1f dB',Gd);
delta=(h^2/2)*((d1+d2)/(d1*d2));    // Path length difference between direct and diffracted rays
n=(2*delta)/lambda;    // Number of Fresnel zones in which the obstruction lies
printf('\n Fresnel zone within which tip of obstruction lies = %0.2f',n);
printf('\n Therefore, the tip of obstruction completely blocks the first three Fresnel zones.');

// b) For h=0
h=0;    // Effective heigth of obstruction screen in m
v=h*sqrt((2*(d1+d2))/(lambda*d1*d2));    // Fresnel diffraction parameter
printf('\n \n b) For h=0 Fresnel diffraction parameter v = %0.0f',v);
```

```
printf('\n From the plot of Knife-edge diffraction gain as a function of Fresnel diffraction parameter, diffraction loss is 6dB.');
Gd=-20*log10(0.5-0.62*v); // Diffraction loss for v=0 in dB
printf('\n Using numerical approximation, diffraction loss for v=0 = %0.0f dB',Gd);
delta=(h^2/2)*((d1+d2)/(d1*d2)); // Path length difference between direct and diffracted rays
n=(2*delta)/lambda; // Number of Fresnel zones in which the obstruction lies
printf('\n Fresnel zone within which tip of obstruction lies = %0.0f',n);
printf('\n Therefore, the tip of obstruction lies in middle of first Fresnel zone.');

// c) For h=-25m
h=-25; // Effective heigth of obstruction screen in m
v=h*sqrt((2*(d1+d2))/(lambda*d1*d2)); // Fresnel diffraction parameter
printf('\n \n c) For h=-25m Fresnel diffraction parameter v = %0.2f',v);
printf('\n From the plot of Knife-edge diffraction gain as a function of Fresnel diffraction parameter, diffraction loss is approximately 1dB.');
Gd=0; // Diffraction loss for v<-1 in dB
printf('\n Using numerical approximation, diffraction loss for v < -1 = %0.0f in dB',Gd);
delta=(h^2/2)*((d1+d2)/(d1*d2)); // Path length difference between direct and diffracted rays
```

Figure 4.7: To determine the loss due to knife edge diffraction and the heigth of obstacle required to induce 6dB diffraction loss

```
n=(2*delta)/lambda;
    // Number of Fresnel zones in which the
    obstruction lies
printf('\n Fresnel zone within which tip of
    obstruction lies = %0.2f',n);
printf('\n Therefore, the tip of obstruction
    completely blocks the first three Fresnel zones
    but diffraction loss is negligible.');
```

Scilab code Exa 4.8 `To determine the loss due to knife edge diffraction and the he`

```
// Example no 4.8
// To determine a)the loss due to knife-edge
    diffraction b)the heigth of obstacle required to
    induce 6dB diffraction loss
// Page no. 133

clc;
```

```
clear;

// Given data
f=900*10^6;  // Operating frequency in Hz
c=3*10^8;  // Speed of ligth in m/s
hr=25;  // Heigth of receiver in m
ht=50;  // Heigth of transmitter in m
h=100;  // Heigth of obstruction in m
d1=10*10^3;  // Distance between transmitter and obstruction in m
d2=2*10^3;  // Distance between receiver and obstruction in m

// a)To determine the loss due to knife-edge diffraction
lambda=c/f;  // Operating wavelength in m
ht=ht-hr;  // Hegth of transmitter after subtracting smallest heigth (hr)
h=h-hr;  // Heigth of obstruction after subtracting
```

```
smallest heigth (hr)
bet=atan((h-ht)/d1);                                  // From geometry of environment in rad
gamm=atan(h/d2);                                      // From geometry of environment in rad
alpha=bet+gamm;                                       // From geometry of environment in rad
v=alpha*sqrt((2*d1*d2)/(lambda*(d1+d2)));             // Fresnel diffraction parameter

// the loss due to knife-edge diffraction
Gd=-20*log10(0.225/v);                                // Diffraction loss for v>2.4 in dB

// Displaying the result in command window
printf('\n The loss due to knife-edge diffraction = %0.1f dB',Gd);

// b)To determine the heigth of obstacle required to induce 6dB diffraction loss
Gd=6;                                                 // Diffraction loss in dB
v=0;                                                  // Fresnel diffraction parameter from the plot of Knife-edge diffraction gain as a function of Fresnel diffraction parameter
// v=0 is possible only if alpha=0. Therefore bet=-gamm
// By considering this situation, the geometry of environment provides (h/d2)=(ht/(d1+d2))
h=(ht*d2)/(d1+d2);                                    // the
```

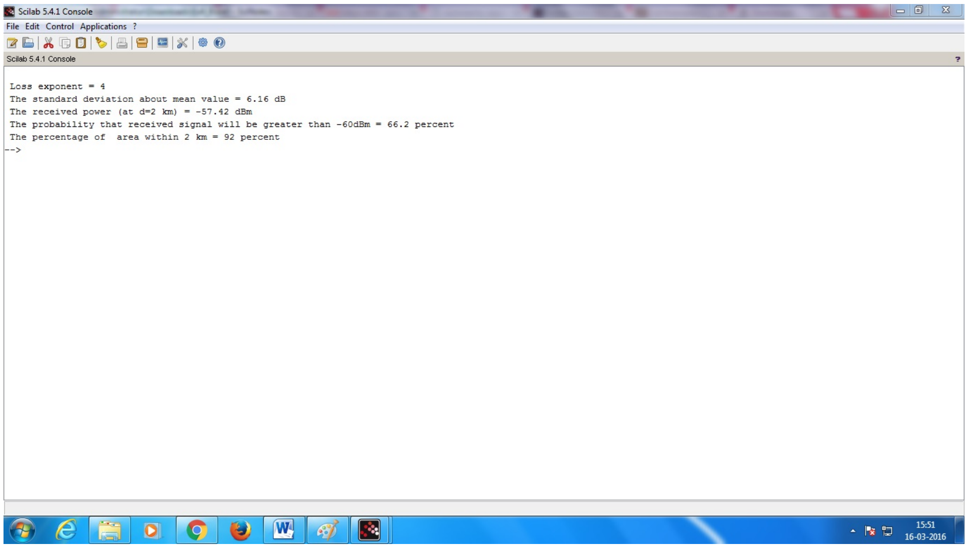

Figure 4.8: To find The minimum mean square error The standard deviation about mean value The received power at 2 km The likelihood that the received signal level at 2 km The percentage of area within 2 km

```
   heigth of obstacle required to induce 6dB
   diffraction loss

// Displaying the result in command window
printf('\n The heigth of obstacle required to induce
    6dB diffraction loss = %0.2f m',h);
```

Scilab code Exa 4.9 To find The minimum mean square error The standard deviation a

```
// Example no 4.9
// To find a)the minimum mean square error b)the
   standard deviation about mean value c)received
   power at d=2 km d)the likelihood that the
   received signal level at 2 km e) the percentage
```

```
    of  area within 2 km
// Page no. 143

clc;
clear all;

// Given data
d0=100;                                                     // First receiver distance in meter
d1=200;                                                     // Second receiver distance in meter
d2=1000;                                                    // Third receiver distance in meter
d3=3000;                                                    // Fourth receiver distance in meter
p0=0;                                                       // Receved power of first receiver in dBm
p1=-20;                                                     // Receved power of second receiver in dBm
p2=-35;                                                     // Receved power of third receiver in dBm
p3=-70;                                                     // Receved power of forth receiver in dBm

// a)To find the minimum mean square error
n=2887.8/654.306;                                           // Loss exponent after differentiating and equating the squared error function with zero

// Displaying the result in command window
printf('\n Loss exponent = %0.0f',n);

// b)To find the standard deviation about mean value
P0=-10*n*log10(d0/100);                                     // The estimate of p0 with path loss model
P1=-10*n*log10(d1/100);                                     // The estimate of p1 with path loss model
P2=-10*n*log10(d2/100);                                     //
```

```
The estimate of p2 with path loss model
P3=-10*n*log10(d3/100);                              // The estimate of p3 with path loss model
J=(p0-P0)^2+(p1-P1)^2+(p2-P2)^2+(p3-P3)^2;      // Sum of squared error
SD=sqrt(J/4);                                        // The standard deviation about mean value

// Displaying the result in command window
printf('\n The standard deviation about mean value = %0.2f dB',SD);
// The decimal point is not given in the answer given in book.

// c)To find received power at d=2 km
d=2000;                                               // The distance of receiver
P=-10*n*log10(d/100);                                 // The estimate of p2 with path loss model

// Displaying the result in command window
printf('\n The received power (at d=2 km) = %0.2f dBm',P);
// Answer is varying due to round off error

// d)To find the likelihood that the received signal level at 2 km
gam=-60;                                               // The received power at 2km will be greater than this power
z=(gam-P)/SD;
Pr=(1/2)*(1-erf(z/sqrt(2)));                           // The probability that received signal will be greater than -60dBm

// Displaying the result in command window
printf('\n The probability that received signal will be greater than -60dBm = %0.1f percent',Pr*100);
```

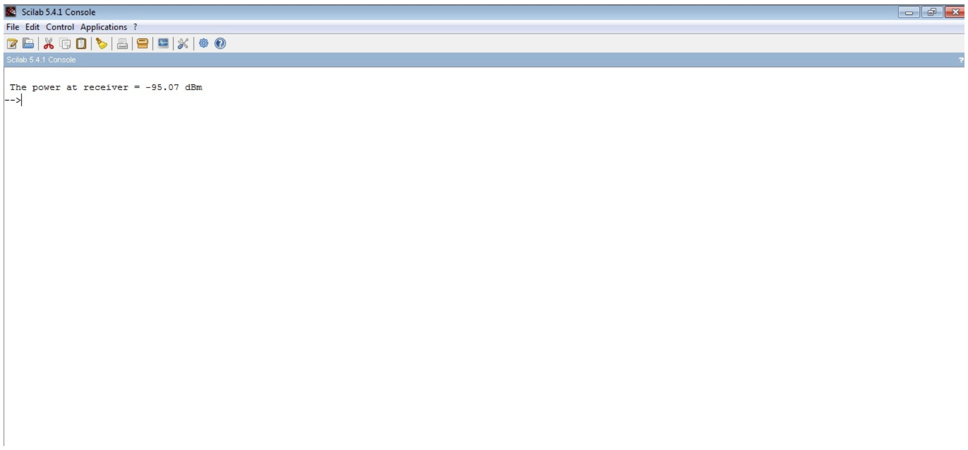

Figure 4.9: To find the power at receiver

```
// Answer is varying due to round off error

// e)To find the percentage of  area within 2 km
A=92;                                                   // From figure 4.18, area receives coverage above −60dBm

// Displaying the result in command window
printf('\n The percentage of  area within 2 km = %0.0f percent',A);
```

Scilab code Exa 4.10 `To find the power at receiver`

```
// Example no 4.10
// To find the power at receiver
// Page no. 152

clc;
clear all;
```

```

// Given data
d=50*10^3;                // Distance between transmitter and receiver in m
hte=100;                  // Effective heigth of transmitter in m
hre=10;                   // Effective heigth of receiver in m
EIRP=1*10^3;              // Radiated power in Watt
f=900*10^6;               // Operating frequency in Hz
c=3*10^8;                 // Speed of ligth in m/s
lambda=c/f;               // operating wavelength in m
EIRP=20*log10(EIRP);      // Radiated power in dB
Gr=0;                     // Receiving gain in dB

Lf=-10*log10(lambda^2/(4*%pi*d)^2);   // Free space path loss in dB
Amu=43;                   // Attenuation relative to free space in dB from Okumuras curve
Garea=9;
```

```
   // Gain due to type of environment in dB from
   Okumuras curve
Ghte=20*log10(hte/200);                                      //
   Base station antenna heigth gain factor for 1000m
    > hte > 30m
Ghre=20*log10(hre/3);
   // Mobile antenna heigth gain factor for 10m >
   hre > 3m
L50=Lf+Amu-Ghte-Ghre-Garea;                                 //
   Total mean path loss

// The median reeived power
Pr=EIRP-L50+Gr;

//Displaying the result in command window
printf('\n The power at receiver = %0.2f dBm',Pr);

//Answer is varrying due to round-off error
```

Scilab code Exa 4.11 To find the mean path loss

```
// Example no 4.11
// To find the mean path loss
// Page no. 166

clc;
clear;

// Given data
```

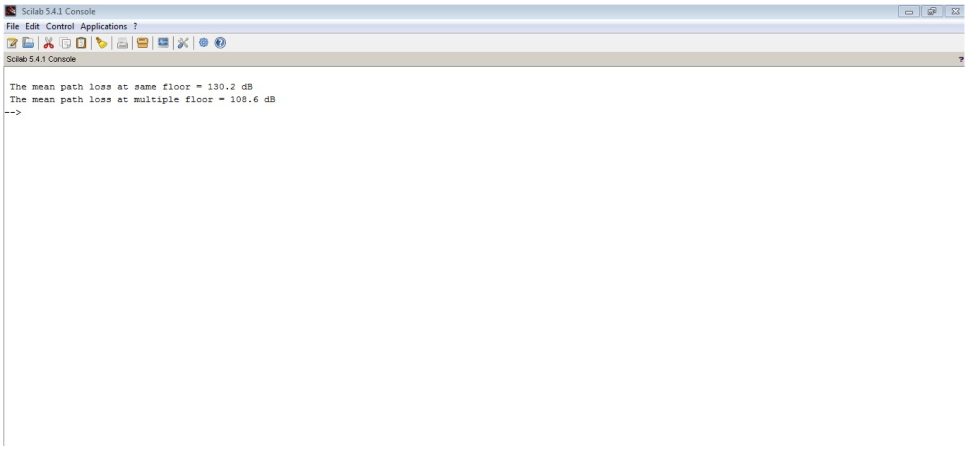

Figure 4.10: To find the mean path loss

```
d0=1;
    // Reference distance in m
d=30;
    // Distance from transmitter in m
nSF=3.27;
    // Exponent value for same floor
nMF=5.22;
    // Path loss exponent value for multiple floors
FAF=24.4;
    // Floor attenuation factor for specified floor
    in dB
n=2;
    // Number of blocks
PAF=13;
    // Particular attenuation factor for paricular
    obstruction in dB
PLSFd0=31.5;
    // Attenuation at reference distance for same
    floor in dB
PLMFd0=5.5;
    // Attenuation at reference distance for multiple
     floor in dB

```

```
//Mean path loss at same floor
PL1=PLSFd0+10*nSF*log10(d/d0)+FAF+n*PAF;

//Mean path loss at multiple floor
PL2=PLMFd0+10*nMF*log10(d/d0)+n*PAF;

//Displaying the result in command window
printf('\n The mean path loss at same floor = %0.1f
   dB',PL1);
printf('\n The mean path loss at multiple floor = %0
   .1f dB',PL2);
```

Chapter 5

Mobile radio propagation small scale propagation

Scilab code Exa 5.1 `To compute received carrier frequency if mobile is moving towa`

```
// Example no 5.1
// To compute received carrier frequency if mobile
   is moving a)towards the transmitter b)away from
   the transmitter c)in the direction perpendicular
   to arrival direction of transmitted signal
// Page no. 180

clc;
clear all;

// Given data
fc=1850*10^6;                                        //
   Carrier frequency in Hz
c=3*10^8;                                               //
    Speed of ligth in m/s
```

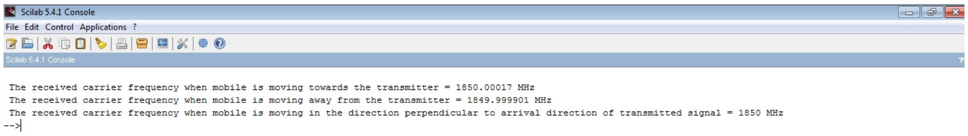

Figure 5.1: To compute received carrier frequency if mobile is moving towards the transmitter and away from the transmitter and in the direction perpendicular to arrival direction of transmitted signal

```
v=60;
    // Speed of receiver (vehicle) in mph
v=v*0.44704;
                                                    //
    Speed of receiver (vehicle) in m/s
lambda=0.162;//c/f;
                                              //
    Wavelength in m

// a)To compute received carrier frequency if mobile
     is moving towards the transmitter
theta=0;
    // Angle between direction of receiver and
    transmitter
fd=(v/lambda)*cos(theta);
                                        // Doppler shift
f=(fc+fd)*10^-6;
                                                //
    Received carrier frequency in MHz
```

```

// Displaying the result in command window
printf('\n The received carrier frequency when mobile is moving towards the transmitter = %0.5f MHz',f);

// b)To compute received carrier frequency if mobile is moving away from the transmitter
theta=180;                                          // Angle between direction of receiver and transmitter
fd=(v/lambda)*cos(theta);                           // Doppler shift
f=(fc+fd)*10^-6;                                    // Received carrier frequency in MHz

// Displaying the result in command window
printf('\n The received carrier frequency when mobile is moving away from the transmitter = %0.6f MHz',f);

// c)To compute received carrier frequency if mobile is moving in the direction perpendicular to arrival direction of transmitted signal
theta=90;                                           // Angle between direction of receiver and transmitter
fd=(v/lambda)*cos(theta);                           // Doppler shift
f=(fc+fd)*10^-6;                                    // Received carrier frequency in MHz

// Displaying the result in command window
printf('\n The received carrier frequency when
```

```
mobile is moving in the direction perpendicular
to arrival direction of transmitted signal = %0.0
f MHz',f);
```

Figure 5.2: To find time delay width and maximum RF bandwidth

Scilab code Exa 5.2 To find time delay width and maximum RF bandwidth

```
// Example no 5.2
// To find a)time delay width (deltat) b)maximum RF
   bandwidth
// Page no. 189

clc;
clear all;

// Given data
tN1=100*10^-6;
   // Excess delays for RF radio channels
```

```
tN2=4*10^-6;
    // Excess delays for microcellular channels
tN3=500*10^-9;
    // Excess delays for indoor channels
N=64;
    // Number of multipath bins

// a)To find time delay width (deltat)
deltat1=(tN1/N)*10^6;                                    //
    Time delay width for RF radio channels
deltat2=(tN2/N)*10^9;                                    //
    Time delay width for microcellular channels
deltat3=(tN3/N)*10^9;                                    //
    Time delay width for indoor channels

// Displaying the result in command window
printf('\n The time delay width for RF radio
    channels = %0.4f microsecond',deltat1);
printf('\n The time delay width for microcellular
    channels = %0.1f nanosecond',deltat2);
printf('\n The time delay width for indoor channels
    = %0.4f nanosecond',deltat3);

//b)To find maximum RF bandwidth
bandwidth1=(2/deltat1);                                 //
    Maximum RF bandwidth for RF radio channels in MHZ
bandwidth2=(2/deltat2)*10^3;                        //Maximum RF
    bandwidth for microcellular channels in MHZ
bandwidth3=(2/deltat3)*10^3;                        //Maximum RF
```

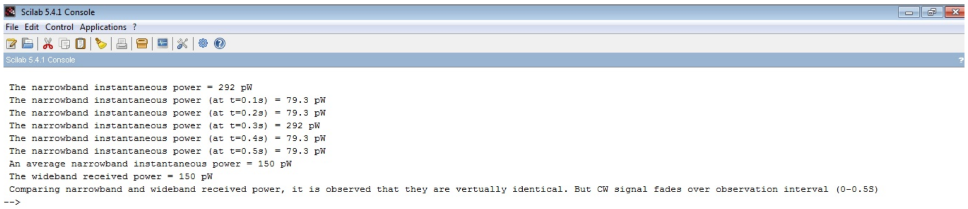

Figure 5.3: To find average narrowband power and to compare average narrow band and wideband power

```
    bandwidth for indoor channels in MHZ

//Displaying the result in command window
printf('\n The maximum RF bandwidth for RF radio
    channels = %0.2f MHz',bandwidth1);
printf('\n The maximum RF bandwidth for
    microcellular channels = %0.0f MHz',bandwidth2);
printf('\n The maximum RF bandwidth for indoor
    channels = %0.0f MHz',bandwidth3);
```

Scilab code Exa 5.3 To find average narrowband power and to compare average narrow

```
// Example no 5.3
// To find average narrowband power & to compare
    average narrow band and wideband power
// Page no. 190

clc;
```

```
clear all;

// Given data
v=10;                                        // Velocity of moving mobile
f=1000*10^6;                                 // Carrier frequency in Hz
c=3*10^8;                                    // Speed of ligth in air (m/s)
P1=-70;                                      // Received power of first component in dBm
P2=P1-3;                                     // Received power of second component in dBm
theta=0;                                     // Initial phase for both component
P1=(10^(P1/10))*10^-3;                       // Received power of first component in Watt
P2=(10^(P2/10))*10^-3;                       // Received power of second component in Watt
lambda=c/f;                                  // Wavelength

// Narrowband instantaneous power
rt2=(sqrt(P1)*cosd(0)+sqrt(P2)*cosd(0))^2;   // Narrowband instantaneous power in pW

// Displaying the result in command window
printf('\n The narrowband instantaneous power = %0.0
```

```
f pW',rt2*10^12);

// Answer is varrying due to round-off error

// To find average narrowband instantaneous power
t=0.1; // Time interval in seconds
theta=((2*%pi*v*t)/lambda)/10; // Phase interval in rad
theta=theta*(180/%pi); // Phase interval in degree
theta1=theta; // Phase of first component at t=0.1s
theta2=-theta; // Phase of second component at t=0.1s
rt21=(sqrt(P1)*(complex(cosd(theta1),sind(theta1))+sqrt(P2)*(complex(cosd(theta2),sind(theta2))))^2; // Narrowband instantaneous power in pW at t=0.1s
mgrt21=sqrt((real(rt21))^2+(imag(rt21))^2);

// Displaying the result in command window
printf('\n The narrowband instantaneous power (at t=0.1s) = %0.1f pW',mgrt21*10^12);

theta1=theta1+theta; // Phase of first component at t=0.2s
theta2=theta2-theta; // Phase of second component at t=0.2s
rt22=(sqrt(P1)*(complex(cosd(theta1),sind(theta1))+sqrt(P2)*(complex(cosd(theta2),sind(theta2))))^2;
```

```
                // Narrowband instantaneous power in pW at t=0.2s
mgrt22=sqrt((real(rt22))^2+(imag(rt22))^2);

// Displaying the result in command window
printf('\n The narrowband instantaneous power (at t=0.2s) = %0.1f pW',mgrt22*10^12);

theta1=theta1+theta;                                                    // Phase of first component at t=0.3s
theta2=theta2-theta;                                                    // Phase of second component at t=0.3s
rt23=(sqrt(P1)*(complex(cosd(theta1),sind(theta1)))+sqrt(P2)*(complex(cosd(theta2),sind(theta2))))^2;                //Narrowband instantaneous power in pW at t=0.3s
mgrt23=sqrt((real(rt23))^2+(imag(rt23))^2);

// Displaying the result in command window
printf('\n The narrowband instantaneous power (at t=0.3s) = %0.0f pW',mgrt23*10^12);

mgrt24=mgrt21;
    // Narrowband instantaneous power in pW at t=0.4s due to repeating phase

// Displaying the result in command window
printf('\n The narrowband instantaneous power (at t=0.4s) = %0.1f pW',mgrt24*10^12);

mgrt25=mgrt22;
    // Narrowband instantaneous power in pW at t=0.5s due to repeating phase

```

```
// Displaying the result in command window
printf('\n The narrowband instantaneous power (at t
    =0.5s) = %0.1f pW',mgrt25*10^12);

rt=(rt2+mgrt21+mgrt22+mgrt23+mgrt24+mgrt25)/6;
                                        // The average
    narrowband instantaneous power in pW

// Displaying the result in command window
printf('\n An average narrowband instantaneous power
     = %0.0f pW',rt*10^12);

// Wideband power
Pwb=(P1+P2);

    // Widebnd received power in pW

// Displaying the result in command window
printf('\n The wideband received power = %0.0f pW',
    Pwb*10^12);

printf('\n Comparing narrowband and wideband
    received power, it is observed that they are
    vertually identical. But CW signal fades over
    observation interval (0-0.5S)');
```

Scilab code Exa 5.4 To compute RMS delay spread and maximum bit rate

```
//Example no 5.4
//To compute a)RMS delay spread b)maximum bit rate
//Page no. 201

clc;
```

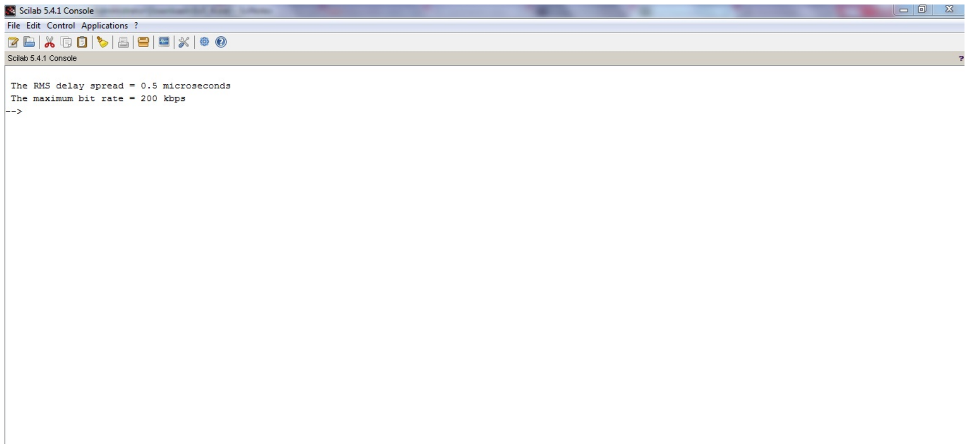

Figure 5.4: To compute RMS delay spread and maximum bit rate

```
clear all;

//Given data
t1=0;    //Excess delay of first signal
a1=0;    //Power level of first signal in dB
t2=1*10^-6;    //Excess delay of second signal
a2=0;    //Power level of second signal in dB
a1=10^(a1);    //Power level of first signal in Watt
a2=10^(a2);    //Power level of second signal in Watt

//a)To compute RMS delay spread
```

```
t=((a1*t1+a2*t2)/(a1+a2))*10^6;                                 //Mean excess delay
t2=((a1*t1^2+a2*t2^2)/(a1+a2))*10^12;                           //Mean square excess delay
sigmat=sqrt(t2-t^2);                                            //RMS delay spread in microseconds

//Displaying the result in command window
printf('\n The RMS delay spread = %0.1f microseconds',sigmat);

//b)To compute maximum bit rate
Ts=(sigmat*10^-6)/0.1;                                          //Sampling time of BPSK modulated signal
Rs=(1/Ts)*10^-3;                                                //Maximum bit rate in kbps

//Displaying the result in command window
printf('\n The maximum bit rate = %0.0f kbps',Rs);
```

Scilab code Exa 5.5 To calculate mean excess delay rms delay spread and maximum ex

```
// Example no 5.5
// To calculate mean excess delay, rms delay spread and maximum excess delay
// Page no. 202

clc;
clear all;
```

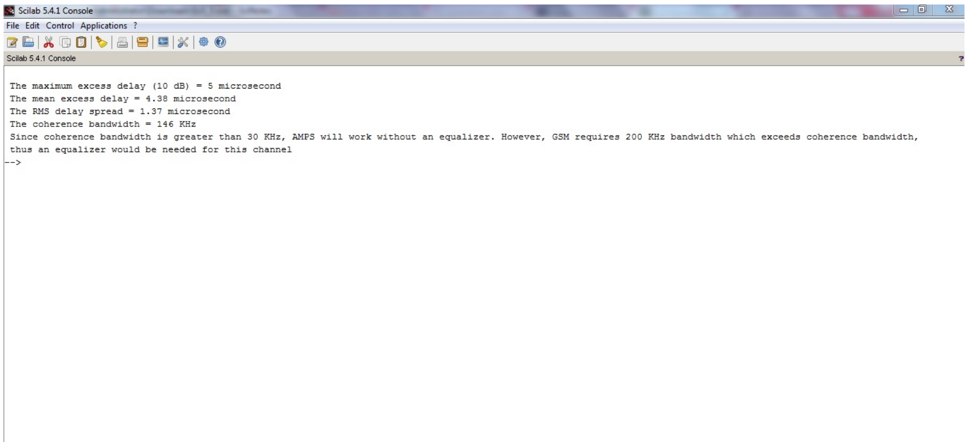

Figure 5.5: To calculate mean excess delay rms delay spread and maximum excess delay

```

// Given data
t10dB=5*10^-6;
    // By definition of maximum excess delay (10dB)
t1=0;
    // Excess delay of first signal in seconds
a1=-20;
    // Power level of first signal in dB
t2=1*10^-6;
    // Excess delay of second signal in seconds
a2=-10;
    // Power level of second signal in dB
t3=2*10^-6;
    // Excess delay of third signal in seconds
a3=-10;
```

```
    // Power level of third signal in dB
t4=5*10^-6;
    // Excess delay of fourth signal in seconds
a4=0;
    // Power level of fourth signal in dB
a1=10^(a1/10);
    // Power level of first signal in Watt
a2=10^(a2/10);
    // Power level of second signal in Watt
a3=10^(a3/10);
    // Power level of third signal in Watt
a4=10^(a4/10);
    // Power level of fourth signal in Watt

// Mean excess delay
t=((a1*t1+a2*t2+a3*t3+a4*t4)/(a1+a2+a3+a4));
                                   // Mean excess delay in
    seconds
tsquare=((a1*t1^2+a2*t2^2+a3*t3^2+a4*t4^2)/(a1+a2+a3
    +a4));            // Mean square excess delay

// RMS delay spread
sigmat=sqrt(tsquare-t^2);
                                                     // RMS
     delay spread

// Coherence bandwidth
Bc=1/(5*sigmat);
    // 50% Coherence bandwidth
// The answer is varrying due to round-off error

```

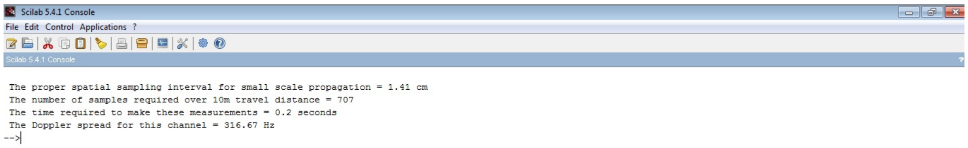

Figure 5.6: To determine proper spatial sampling interval for small scale propagation number of samples required over 10m time required to make these measurements and Doppler spread for this channel

```
// Displaying the result in command window
printf('\n The maximum excess delay (10 dB) = %0.0f
    microsecond',t10dB*10^6);
printf('\n The mean excess delay = %0.2f microsecond
    ',t*10^6);
printf('\n The RMS delay spread = %0.2f microsecond'
    ,sigmat*10^6);
printf('\n The coherence bandwidth = %0.0f KHz',Bc
    *10^-3);
printf('\n Since coherence bandwidth is greater than
     30 KHz, AMPS will work without an equalizer.
    However, GSM requires 200 KHz bandwidth which
    exceeds coherence bandwidth,\n thus an equalizer
    would be needed for this channel');
```

Scilab code Exa 5.6 To determine proper spatial sampling interval for small scale

```
// Example no 5.6
// To determine proper spatial sampling interval for small scale propagation, number of samples required over 10m, time required to make these measurements and Doppler spread for this channel
// Page no. 204

clc;
clear all;

// Given data
fc=1900*10^6;                                      // Carrier frequency in Hz
v=50;                                              // Velocity of propagation in m/s
c=3*10^8;                                          // Speed of ligth in air in m/s
Tc=(9*c)/(16*%pi*v*fc);                            // Coherence time

// The spatial sampling interval
deltax=(v*Tc)/2;                                   // Spatial sampling interval in meter

// The number of samples required over 10m travel distance
d=10;                                              // Distance to be travelled
Nx=d/deltax;                                       // Number of samples required over 10m
// Answer is varrying due to round-off error
```

```

// The time required to make these measurements
t=d/v;
    // Time required to make these measurements

// Doppler spread for this channel
BD=(v*fc)/c;
    // Doppler spread for this channel
// Answer is varrying due to round-off error

// Displaying the result in command window
printf('\n The proper spatial sampling interval for
    small scale propagation = %0.2f cm',deltax*10^2);
printf('\n The number of samples required over 10m
    travel distance = %0.0f',Nx);
printf('\n The time required to make these
    measurements = %0.1f seconds',t);
printf('\n The Doppler spread for this channel = %0
    .2f Hz',BD);
```

Scilab code Exa 5.7 To compute the positive going lvel crossing rate and maximum v

```
// Example no 5.7
// To compute the positive-going lvel crossing rate
    and maximum velocity of mobile
// Page no. 224

clc;
clear all;

// Given data
```

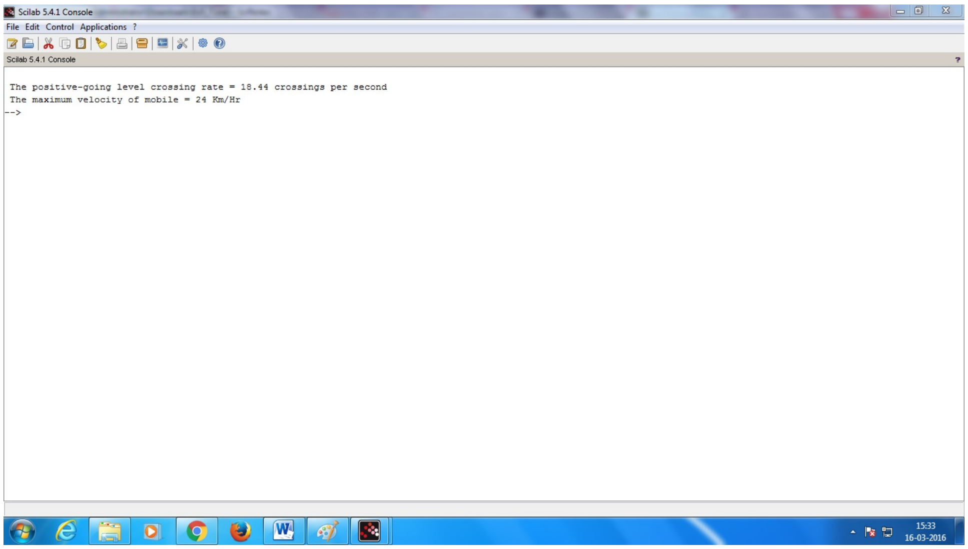

Figure 5.7: To compute the positive going lvel crossing rate and maximum velocity of mobile

```
rho=1;                        // Value of normalized level of fading amplitude to rms amplitude
fm=20;                        // Maximum Doppler frequency in Hz
fc=900*10^6;                  // Carrier frequency in Hz
c=3*10^8;                     // Speed of ligth in air in m/s

// The positive-going level crossing rate
NR=sqrt(2*%pi)*fm*rho*exp(-rho^2);   // Number of zero level crossings per second
lambda=c/fc;                  // Carrier wavelength

// The maximum velocity of mobile
v=fm*lambda;                  // Maximum velocity of mobile in m/s
```

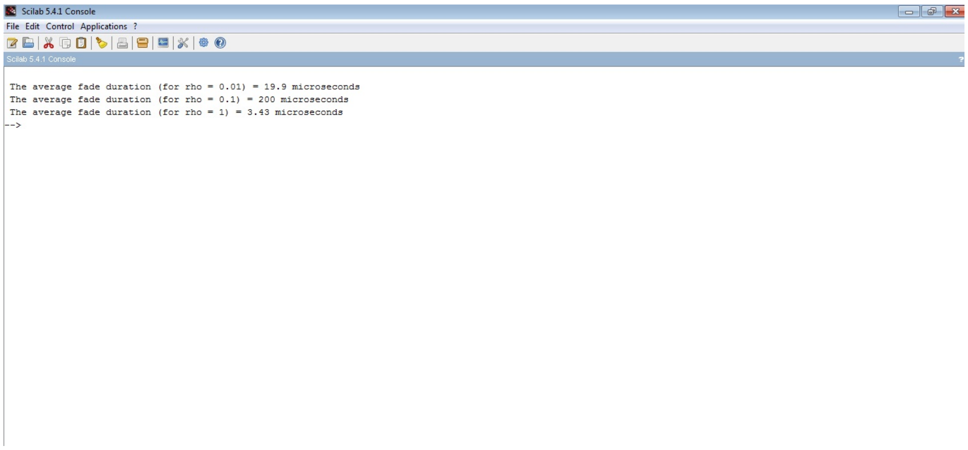

Figure 5.8: To find the average fade duration

```
v=v*(18/5);                                    //
   Maximum velocity of mobile in km/hr

// Displaying the result in command window
printf('\n The positive-going level crossing rate =
   %0.2f crossings per second',NR);
printf('\n The maximum velocity of mobile = %0.0f Km
   /Hr',v);
```

Scilab code Exa 5.8 `To find the average fade duration`

```
// Example no 5.8
// To find the average fade duration
// Page no. 225

clc;
clear all;

// Given data
```

```
rho1=0.01;                                                                    // Threshold level
rho2=0.1;                                                                     // Threshold level
rho3=1;                                                                       // Threshold level
fm=200;                                                                       // Doppler frequency

t1=(exp(rho1^2)-1)/(rho1*fm*sqrt(2*%pi));                  // The average fade duration
t2=(exp(rho2^2)-1)/(rho2*fm*sqrt(2*%pi));                  // The average fade duration
t3=(exp(rho3^2)-1)/(rho3*fm*sqrt(2*%pi));                  // The average fade duration

// Displaying the result in command window
printf('\n The average fade duration (for rho = 0.01) = %0.1f microseconds',t1*10^6);
printf('\n The average fade duration (for rho = 0.1) = %0.0f microseconds',t2*10^6);
printf('\n The average fade duration (for rho = 1) = %0.2f microseconds',t3*10^3);
```

Scilab code Exa 5.9 `To find the average fade duration and average number of bit er`

```
// Example no 5.9
// To find the average fade duration and average number of bit errors per second. & to determine
```

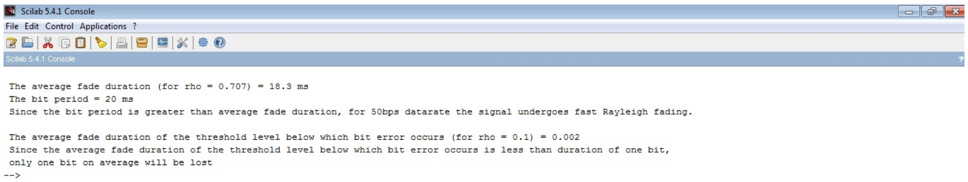

Figure 5.9: To find the average fade duration and average number of bit errors per second and to determine whether the fading is slow or fast

```
    whether the fading is slow or fast.
// Page no. 225

clc;
clear all;

// Given data
rho=0.707;
    // Threshold level
fm=20;
    // Doppler frequency
datarate=50;
                                                    //
    Bit duration of binary digital modulation in bps
errho=0.1;
    // Threshold level below which bit error occurs

t=(exp(rho^2)-1)/(rho*fm*sqrt(2*%pi));
                            // The average fade duration
```

```
tb=1/datarate;                                                          // Bit period
t1=(exp(errho^2)-1)/(errho*fm*sqrt(2*%pi));                    // The average fade duration

// Displaying the result in command window
printf('\n The average fade duration (for rho = 0.707) = %0.1f ms',t*10^3);
printf('\n The bit period = %0.0f ms',tb*10^3);
printf('\n Since the bit period is greater than average fade duration, for 50bps datarate the signal undergoes fast Rayleigh fading.');
printf('\n \n The average fade duration of the threshold level below which bit error occurs (for rho = 0.1) = %0.3f',t1);
printf('\n Since the average fade duration of the threshold level below which bit error occurs is less than duration of one bit,\n only one bit on average will be lost');
```

Chapter 6

Modulation techniques for mobile radio

Scilab code Exa 6.1 To compute the carrier power and percentage of total power in

```
// Example no 6.1
// To compute the carrier power, percentage of total
    power in carrier power and power in each
   sideband.
// Page no. 260

clc;
clear all;

// Given data
PAM=10*10^3;
   // Power of transmitted AM signal
k=0.6;
   // Modulation index

```

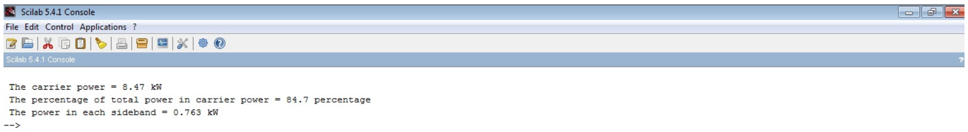

Figure 6.1: To compute the carrier power and percentage of total power in carrier power and power in each side band

```
// To compute the carrier power
Pc=PAM/(1+k^2/2);
    // The carrier power

// To compute percentage of total power in carrier
    power
PercentPc=(Pc/PAM)*100;
    // Percentage of total power in carrier power

// To compute power in each sideband
Psideband=(PAM-Pc)/2;
    // The power in each sideband
// Answer is varrying due to round-off error

// Displaying the result in command window
printf('\n The carrier power = %0.2f kW',Pc*10^-3);
printf('\n The percentage of total power in carrier
    power = %0.1f percentage',PercentPc);
printf('\n The power in each sideband = %0.3f kW',
```

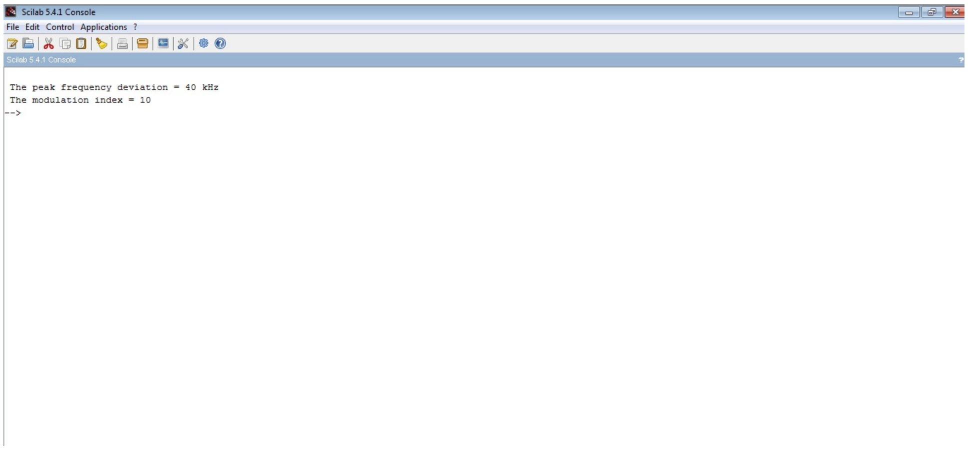

Figure 6.2: To compute the peak frequency deviation and the modulation index

```
    Psideband*10^-3);
```

Scilab code Exa 6.2 To compute the peak frequency deviation and the modulation ind

```
// Example no 6.2
// To compute a)the peak frequency deviation b)the
    modulation index.
// Page no. 265

clc;
clear all;

// Given data
kf=10*10^3;
    // Frequency deviation constant gain in Hz/V
fm=4*10^3;
```

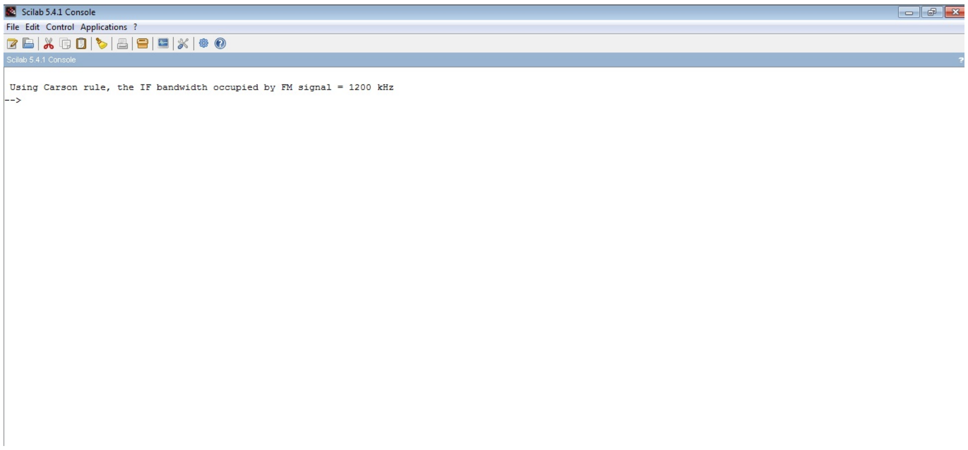

Figure 6.3: To determine the IF bandwidth necessary to pass the given signal

```
    // Modulating frequency in Hz
A=4;
    // Maximum instantaneous value of input signal in
     V

// To compute the peak frequency deviation
deltaf=A*kf;
    // The peak frequency deviation in Hz

// To compute the modulation index
betaf=deltaf/fm;                                    //
    The modulation index

// Displaying the result in command window
printf('\n The peak frequency deviation = %0.0f kHz'
    ,deltaf*10^-3);
printf('\n The modulation index = %0.0f',betaf);
```

Scilab code Exa 6.3 To determine the IF bandwidth necessary to pass the given sign

```
// Example no 6.3
// To determine the IF bandwidth necessary to pass the given signal.
// Page no. 267

clc;
clear all;

// Given data
fm=100*10^3;                                    // Modulating frequency in Hz
deltaf=500*10^3;                                // Peak frequency deviation in Hz
betaf=deltaf/fm;                                // Modulation index

// The IF bandwidth occupied by FM signal using Carson's rule
BT=2*(betaf+1)*fm;                              // The IF bandwidth necessary to pass the given signal

// Displaying the result in command window
printf('\n Using Carson rule, the IF bandwidth occupied by FM signal = %0.0f kHz',BT*10^-3);
```

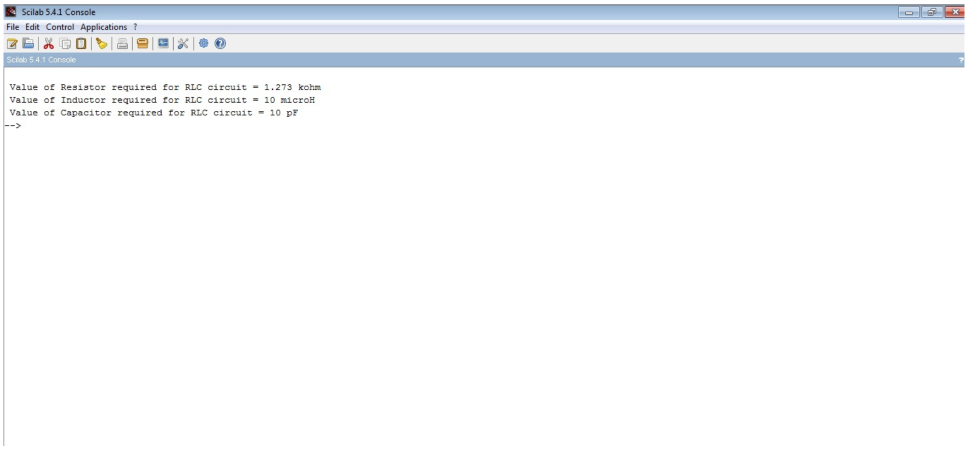

Figure 6.4: To design an RLC network that implements an IF quadrature FM detector

Scilab code Exa 6.4 `To design an RLC network that implements an IF quadrature FM d`

```
// Example no 6.4
// To design an RLC network that implements an IF
    quadrature FM detector
// Page no. 273

clc;
clear all;
close;

// Given data
fc=10.7*10^6;                                                    //
    Cut-off frequency in Hz
B=500*10^3;
```

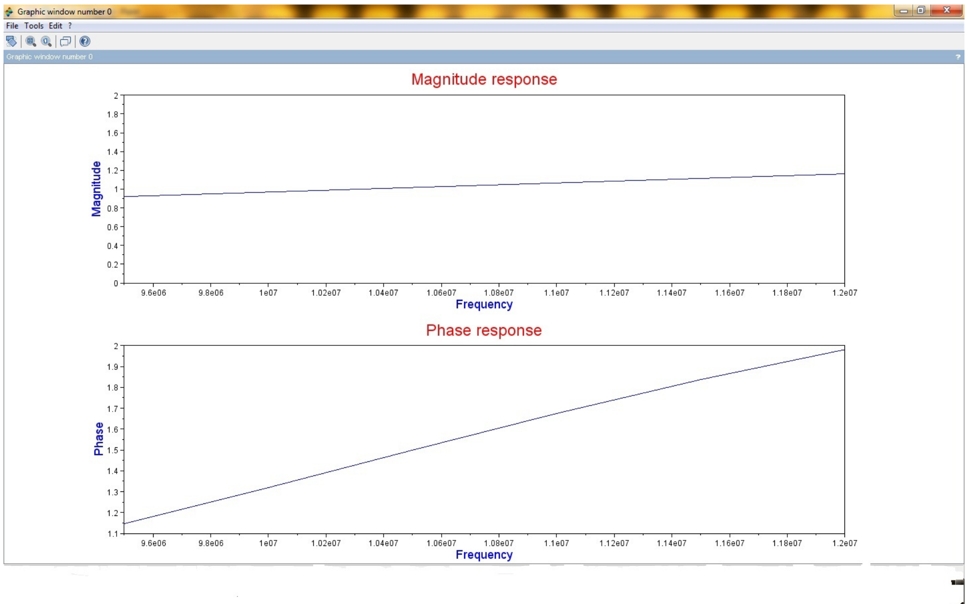

Figure 6.5: To design an RLC network that implements an IF quadrature FM detector

```
    //Bandwidth in Hz
phi=5;                                      //phase shift for good system in degree
Q=tand(phi)/((fc+B/2)/fc-fc/(fc+B/2));      //Q-factor
L=10*10^(-6);                               //Chosen value of inductor
R=Q*2*%pi*fc*L;                             //Value of Resistor
c1=12.13*10^(-12);                          //Chosen value of C1
c=(Q/(R*2*%pi*fc))-c1;                      //Value of capacitor
```

```

// Displaying the result in command window
printf('\n Value of Resistor required for RLC circuit = %0.3f kohm',R*10^(-3));
printf('\n Value of Inductor required for RLC circuit = %0.0f microH',L*10^(6));
printf('\n Value of Capacitor required for RLC circuit = %0.0f pF',c*10^(12));

// Magnitude plot
f=0.95*10^7:0.05*10^7:1.2*10^7;                              // Frequency range for plotting in Hz
mgh=(2*%pi*f*R*c1)/sqrt(1+Q^2*((f^2-fc^2)/(f*fc))^2);    // Magnitude transfer function
subplot(211);
plot(f,mgh);
a=gca();
a.data_bounds=[0.95*10^7 0;1.2*10^7 2];                     // To see the vertical line hiddden by the y axis
xlabel("Frequency","color","blue");
ylabel("Magnitude","color","blue");
title("Magnitude response","fontsize","6","color","red");

// Phase plot
f=0.95*10^7                                                            // Initial frequency for plotting
for i=1:6
    if f<1.25*10^7 then
        phH(i)=(%pi/2)+atan(Q*((f^2-fc^2)/(f*fc)));            // Phase transfer function
        f=f+0.05*10^7;
    end
end

```

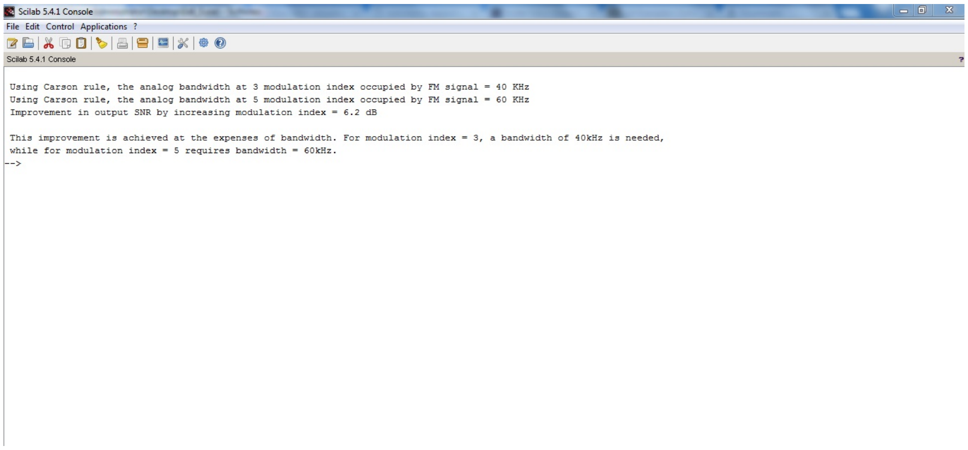

Figure 6.6: To determine the analog bandwidth and output SNR improvement if modulation index is increased from three to five and trade off bandwidth for this improvement

```
f=0.95*10^7:0.05*10^7:1.2*10^7;
subplot(212);
plot(f,phH);
a=gca();
a.data_bounds=[0.95*10^7 1.2;1.2*10^7 2];
                    // To see the vertical line hiddden
    by the y axis
xlabel("Frequency","color","blue");
ylabel("Phase","color","blue");
title("Phase response","fontsize","10","color","red"
    );
```

Scilab code Exa 6.5 `To determine the analog bandwidth and output SNR improvement i`

```
// Example no 6.5
// To determine the analog bandwidth, output SNR
    improvement if modulation index is increased from
```

```
3 to 5 and tradeoff bandwidth for this improvement.
// Page no. 277

clc;
clear all;

// Given data
fm=5*10^3;                    // Audio bandwidth of FM signal
betaf1=3;                     // Initial modulation index
betaf2=5;                     // Final modulation index

// To determine analog bandwidth
BT1=2*(betaf1+1)*fm;          // The analog bandwidth
BT2=2*(betaf2+1)*fm;          // The analog bandwidth

// To determine output SNR improvement factor
SNR1=3*betaf1^3+3*betaf1^2;   // Output SNR factor for modulation index=3
SNR1=10*log10(SNR1);          // Output SNR factor for modulation index=3 in dB
SNR2=3*betaf2^3+3*betaf2^2;   // Output SNR factor for modulation index=3
SNR2=10*log10(SNR2);          // Output SNR factor for modulation index=3 in dB

// To determine improvement in output SNR by increasing modulation index
improvedSNR=SNR2-SNR1;        // Improvement in output SNR by increasing modulation index

// Displaying the result in command window
```

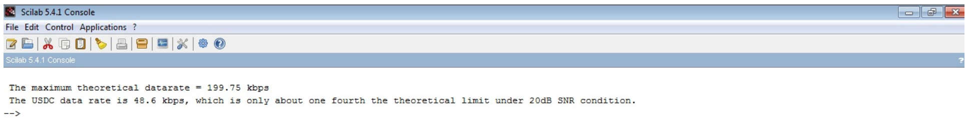

Figure 6.7: To determine the maximum theoretical datarate and to compare this rate to US digital cellular standard

```
printf('\n Using Carson rule, the analog bandwidth at 3 modulation index occupied by FM signal = %0.0f KHz',BT1*10^-3);
printf('\n Using Carson rule, the analog bandwidth at 5 modulation index occupied by FM signal = %0.0f KHz',BT2*10^-3);
printf('\n Improvement in output SNR by increasing modulation index = %0.1f dB',improvedSNR);
printf('\n \n This improvement is achieved at the expenses of bandwidth. For modulation index = 3, a bandwidth of 40kHz is needed,\n while for modulation index = 5 requires bandwidth = 60kHz.');
```

Scilab code Exa 6.6 To determine the maximum theoretical datarate and to compare t

```
// Example no 6.6
```

```
// To determine the maximum theoretical datarate and to compare this rate to US digital cellular standard
// Page no. 280

clc;
clear all;

// Given data
SNR=20;    // Signal to noise ratio of wireless communication link in dB
B=30*10^3;    // RF bandwidth in Hz
SNR=10^(SNR/10);    // Signal to noise ratio of wireless communication link

// To determine the maximum theoretical datarate
C=B*(log10(1+SNR)/log10(2));    // The maximum theoretical datarate in bps

// Displaying the result in command window
printf('\n The maximum theoretical datarate = %0.2f kbps',C*10^-3);
printf('\n The USDC data rate is 48.6 kbps, which is only about one fourth the theoretical limit under 20dB SNR condition.');
```

Scilab code Exa 6.7 To determine the maximum theoretical datarate and to compare t

```
// Example no 6.7
// To determine the maximum theoretical datarate and
```

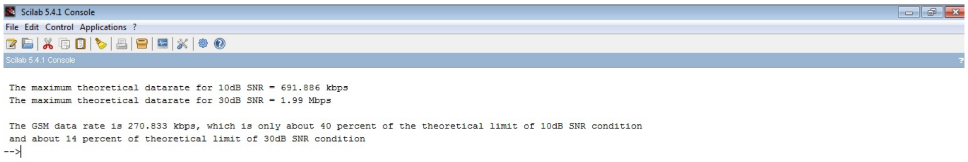

Figure 6.8: To determine the maximum theoretical datarate and to compare this rate to GSM standard

```
    to compare this rate to GSM standard
// Page no. 280

clc;
clear all;

// Given data
SNR1=10;                                                    //
    Signal to noise ratio in dB
SNR2=30;                                                    //
    Signal to noise ratio in dB
B=200*10^3;                                                 //
    RF bandwidth of channel in Hz

SNR1=10^(SNR1/10);                                          //
    Signal to noise ratio
SNR2=10^(SNR2/10);                                          //
    Signal to noise ratio

// To determine the maximum theoretical datarate
C1=B*(log10(1+SNR1)/log10(2));                              //
    The maximum theoretical datarate for SNR=10dB
```

Figure 6.9: To find the first zero crossing RF bandwidth of rectangular pulse and compare to raised cosine filter pulse

```
C2=B*(log10(1+SNR2)/log10(2));                    //
    The maximum theoretical datarate for SNR=30dB

// Displaying the result in command window
printf('\n The maximum theoretical datarate for 10dB
    SNR = %0.3f kbps',C1*10^-3);
printf('\n The maximum theoretical datarate for 30dB
    SNR = %0.2f Mbps',C2*10^-6);
printf('\n \n The GSM data rate is 270.833 kbps,
   which is only about 40 percent of the theoretical
    limit of 10dB SNR condition\n and about 14
   percent of theoretical limit of 30dB SNR
   condition');
```

Scilab code Exa 6.8 To find the first zero crossing RF bandwidth of rectangular pu

```
// Example no 6.8
```

```
// To find the first zero-crossing RF bandwidth of rectangular pulse and compare to raised cosine filter pulse
// Page no. 291

clc;
clear all;

// Given data
RectTs=41.06*10^-6;                                                          // Symbol period of rectangular pulse
cosineTs=41.06*10^-6;                                                        // Symbol period of cosine filter pulse
alpha=0.35;                                                                  // Rolloff factor of cosine filter pulse

// To find the first zero-crossing RF bandwidth of rectangular pulse
B1=2/RectTs;                                                                 // The first zero-crossing RF bandwidth of rectangular pulse

// The first zero-crossing RF bandwidth of cosine filter pulse
B2=(1/cosineTs)*(1+alpha);                                                   // The first zero-crossing RF bandwidth of cosine filter pulse

// Displaying the result in command window
printf('\n The first zero-crossing RF bandwidth of rectangular pulse = %0.2f kHz',B1*10^-3);
printf('\n The first zero-crossing RF bandwidth of cosine filter pulse = %0.2f kHz',B2*10^-3);
```

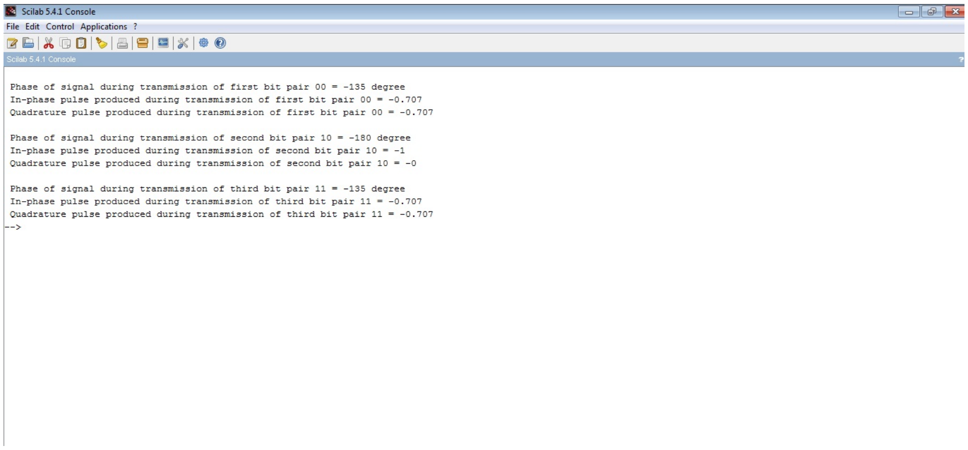

Figure 6.10: To determine phase and values of Ik and Qk during transmission of bit stream 001011 using pi by 4 DQPSK

Scilab code Exa 6.9 `To determine phase and values of Ik and Qk during transmission`

```
// Example no 6.9
// To determine phase and values of Ik and Qk during
    transmission of bit stream 001011 using pi/4
   DQPSK.
// Page no. 307

clc;
clear all;

// Given data
theta0=0;
   // Initial phase in rad
phi1=%pi/4;
```

```
// Carrier phase shift for the input bit pair 11
[Feh91], [Rap91b]
phi2=(3*%pi)/4;
// Carrier phase shift for the input bit pair 01
[Feh91], [Rap91b]
phi3=(-3*%pi)/4;
// Carrier phase shift for the input bit pair 00
[Feh91], [Rap91b]
phi4=-%pi/4;
// Carrier phase shift for the input bit pair 10
[Feh91], [Rap91b]

// For transmission of first pair of bits 00
theta1=theta0+phi3;
// Phase of signal during transmission of first
bit pair 00
I1=cos(theta1);
// In-phase pulse produced at the output of
signal mapping
Q1=sin(theta1);
// Quadrature pulse produced at the output of
signal mapping

// For transmission of second pair of bits 10
theta2=theta1+phi4;
// Phase of signal during transmission of second
bit pair 10
I2=cos(theta2);
// In-phase pulse produced at the output of
```

```
    signal mapping
Q2=sin(theta2);                        // Quadrature pulse produced at the output of signal mapping

// For transmission of third pair of bits 11
theta3=theta2+phi1;                    // Phase of signal during transmission of third bit pair 11
I3=cos(theta3);                        // In-phase pulse produced at the output of signal mapping
Q3=sin(theta3);                        // Quadrature pulse produced at the output of signal mapping

// Displaying the result in command window
printf('\n Phase of signal during transmission of first bit pair 00 = %0.0f degree',theta1*(180/%pi));
printf('\n In-phase pulse produced during transmission of first bit pair 00 = %0.3f',I1);
printf('\n Quadrature pulse produced during transmission of first bit pair 00 = %0.3f',Q1);

printf('\n \n Phase of signal during transmission of second bit pair 10 = %0.0f degree',theta2*(180/%pi));
printf('\n In-phase pulse produced during transmission of second bit pair 10 = %0.0f',I2);
printf('\n Quadrature pulse produced during transmission of second bit pair 10 = %0.0f',Q2);

printf('\n \n Phase of signal during transmission of
```

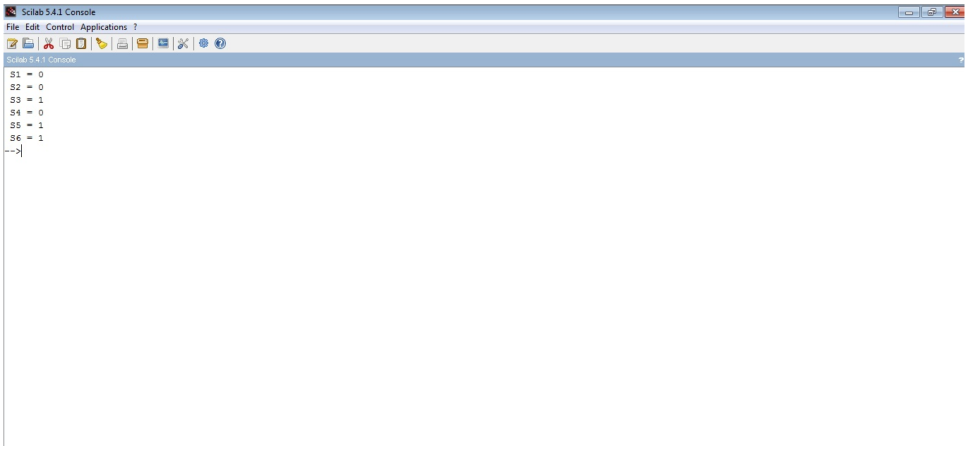

Figure 6.11: To demonstrate how the received signal is detected properly using baseband differential detector

```
    third bit pair 11 = %0.0f degree',theta3*(180/
   %pi));
printf('\n In-phase pulse produced during
   transmission of third bit pair 11 = %0.3f',I3);
printf('\n Quadrature pulse produced during
   transmission of third bit pair 11 = %0.3f',Q3);
```

Scilab code Exa 6.10 To demonstrate how the received signal is detected properly u

```
// Example no 6.10
// To demonstrate how the received signal is
   detected properly using baseband differential
   detector.
// Page no. 310

clc;
clear all;

```

```
// Given data
x1=-0.707;
y1=-0.707;
x2=0.707;
y2=-0.707;
x3=0.707;
y3=0.707;

if x1<0 then // Applying decision rule
printf('S1 = 0');
else
printf('\n S1 = 1');
end
if y1<0 then
printf('\n S2 = 0');
else
printf('\n S2 = 1');
end
if x2<0 then
printf('\n S3 = 0');
else
printf('\n S3 = 1');
end
if y2<0 then
printf('\n S4 = 0');
else
printf('\n S4 = 1');
end
if x3<0 then
printf('\n S5 = 0');
else
printf('\n S5 = 1');
end
if y3<0 then
printf('\n S6 = 0');
else
```

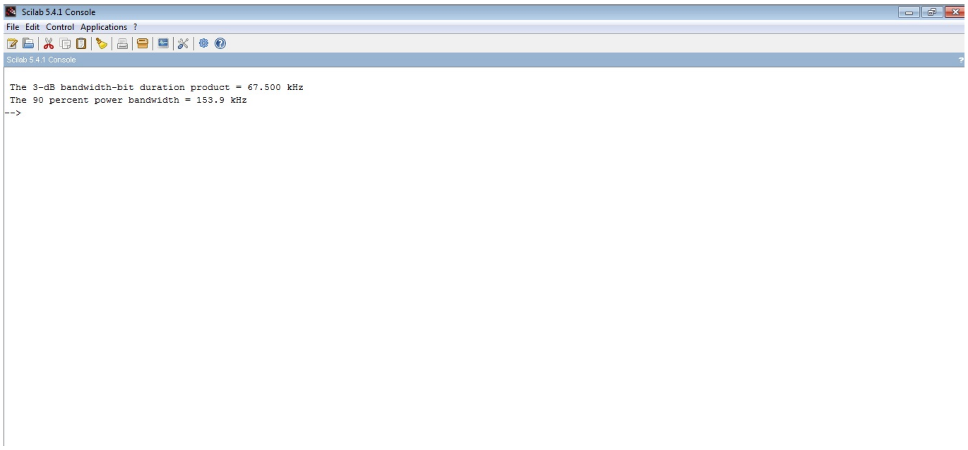

Figure 6.12: To find 3 dB bandwidth for Gaussian low pass filter to produce 90 percent power bandwidth

```
printf('\n S6 = 1');
end
```

Scilab code Exa 6.11 `To find 3 dB bandwidth for Gaussian low pass filter to produc`

```
// Example no 6.11
// To find 3-dB bandwidth for gaussian low pass
   filter to produce 0.25GMSK, 90% power bandwidth.
// Page no. 321

clc;
clear all;

// Given data
Rb=270*10^3;
   // Channel data rate in bps
```

```
BT=0.25;
    // 3-dB bandwidth-bit duration product

T=1/Rb;
    // Time
B=BT/T;
    // 3-dB bandwidth in Hz
// Answer is varrying due to round-off error

// 90% power bandwidth
B1=0.57*Rb;
    // The 90% power bandwidth
// Answer is varrying due to round-off error

// Displaying the result in command window
printf('\n The 3-dB bandwidth-bit duration product =
     %0.3f kHz',B*10^-3);
printf('\n The 90 percent power bandwidth = %0.1f
    kHz',B1*10^-3);
```

Chapter 7

Equalization diversity and channel coding

Scilab code Exa 7.3 To determine the maximum Doppler shift and the coherence time

```
// Example no 7.3
// To determine a)the maximum Doppler shift b)the
   coherence time of the channel c)the maximum
   number of symbolsthat could be transmitted
// Page no. 373

clc;
clear all;

//Given data
f=900*10^6;

   // Carrier frequency in Hz
c=3*10^8;

   // Speed of ligth in air (m/s)
v=80;
```

Figure 7.1: To determine the maximum Doppler shift and the coherence time of the channel and the maximum number of symbols that could be transmitted

```
    // Velocity of mobile in km/hr
v=v*(5/18);
    // Velocity of mobile in m/s
lambda=c/f;
    // Carrier wavelength in meter

// a)To determine the maximum Doppler shift
fd=v/lambda;
    // The maximum Doppler shift in Hz

// b)To determine the coherence time of the channel
Tc=sqrt(9/(16*%pi*fd^2));                    // The
    coherence time of the channel
// Answer is varrying due to round-off error

// c)To determine the maximum number of symbols that
```

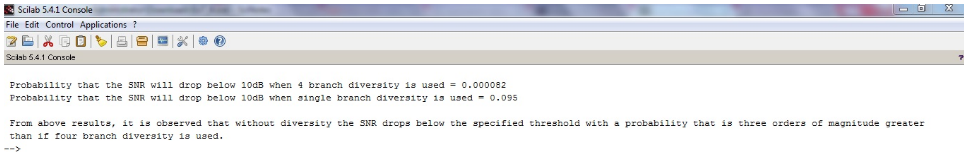

Figure 7.2: To determine probability that the SNR will drop below threshold SNR

```
    could be transmitted with symbol rate 24.3
    ksymbols/sec
Rs=24.3*10^3;                                                  //
    Symbol rate in symbols/sec
Nb=Tc*Rs;
    // The maximum number of transmitted symbols

// Displaying the result in command window
printf('\n The maximum Doppler shift = %0.2f Hz',fd)
    ;
printf('\n The coherence time of the channel = %0.2f
     ms',Tc*10^3);
printf('\n The maximum number of symbols that could
    be transmitted with symbol rate 24.3 ksymbols/sec
     = %0.0f symbols',Nb);
```

Scilab code Exa 7.4 To determine probability that the SNR will drop below threshol

```
// Example no 7.4
// To determine probability that the SNR will drop below threshold SNR
// Page no. 383

clc;
clear all;

// Given data
M1=4;                                        // Number of branch diversity
M2=1;                                        // Number of branch diversity
gamm=10;                                     // Specified SNR threshold in dB
gamm=10^(gamm/10);                           // Specified SNR threshold
Gamma=20;                                    // Average SNR in dB
Gamma=10^(Gamma/10);                         // Average SNR

// Probability that the SNR will drop below 10dB when 4 branch diversity is used
P4=(1-exp(-gamm/Gamma))^M1;                 // Probability that the SNR will drop below 10dB

// Probability that the SNR will drop below 10dB when single branch diversity is used
P1=(1-exp(-gamm/Gamma))^M2;                 // Probability that the SNR will drop below 10dB

// Displaying the result in command window
printf('\n Probability that the SNR will drop below 10dB when 4 branch diversity is used = %0.6f',P4);
```

```
printf('\n Probability that the SNR will drop below 10dB when single branch diversity is used = %0.3f',P1);
printf('\n \n From above results, it is observed that without diversity the SNR drops below the specified threshold with a probability that is three orders of magnitude greater \n than if four branch diversity is used.')
```

Chapter 8

Speech coding

Scilab code Exa 8.1 To compute the mean square error distortion and output signal

```
// Example no 8.1
// To compute the mean square error distortion and output signal-to-distortion ratio.
// Page no. 420


clc;
clear all;

//Given data
l1=1;                                       // 1st Quantization level
l2=3;                                       // 2nd Quantization level
l3=5;                                       // 3rd Quantization level
```

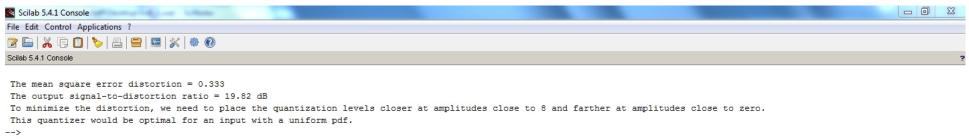

Figure 8.1: To compute the mean square error distortion and output signal to distortion ratio

```
l4=7;
    // 4th Quantization level

U1=(l1+l2)/2;
    // upper boundary of 1st level
U2=(l2+l3)/2;
    // upper boundary of 2nd level
U3=(l3+l4)/2;
    // upper boundary of 3rd level
U4=l4+(U1-l1);
    // upper boundary of 4th level
L1=l1-(U1-l1);
    // Lower boundary of 1st level

```

```
D1=integrate('(x^3-2*x^2+x)/32','x',L1,U1);                    // Mean square error distortion of 1st level
D2=integrate('(x^3-6*x^2+9*x)/32','x',U1,U2);                  // Mean square error distortion of 2nd level
D3=integrate('(x^3-10*x^2+25*x)/32','x',U2,U3);                // Mean square error distortion of 3rd level
D4=integrate('(x^3-14*x^2+49*x)/32','x',U3,U4);                // Mean square error distortion of 4th level
D=D1+D2+D3+D4;                                                  // Total square error distortion

P=integrate('x^3/32','x',L1,U4);                                // Signal power

SDR=10*log10(P/D);                                              // Output signal-to-distortion ratio.

// Displaying the result in command window
printf('\n The mean square error distortion = %0.3f',D);
printf('\n The output signal-to-distortion ratio = %0.2f dB',SDR);
printf('\n To minimize the distortion, we need to place the quantization levels closer at amplitudes close to 8 and farther at amplitudes close to zero.');
printf('\n This quantizer would be optimal for an input with a uniform pdf.');
```

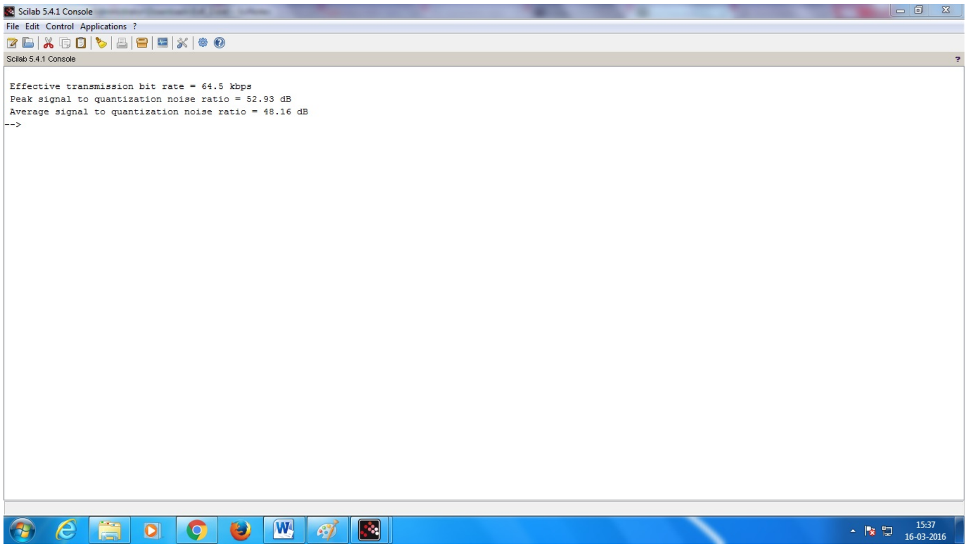

Figure 8.2: To compute transmission bit rate average and peak signal to quantization noise ratio

Scilab code Exa 8.2 `To compute transmission bit rate average and peak signal to qu`

```
// Example no 8.2
// To compute transmission bit rate, average and
   peak signal to quantization noise ratio
// Page no. 424

clc;
clear all;

// Given data
fs=8*10^3;                                        //
   Sampling frequency in Hz
n=8;                                              //
```

```
   Number of bits per sample
stepsize=10*10^-3;                              //
   Time after which step size is recomputed
overhead=5;                                     //
   Number of overhead bits

N=fs*n;                                         //
   Number of information bits pe second
Toverhead=overhead/stepsize;                    // The
    number of overhead bits/second

// Effective transmission bit rate
bitrate=N+Toverhead;                           //
   Transmission bit rate in bps

// Peak signal to quantization noise ratio
PSQNR=6.02*n+4.77;                             // Peak
    signal to quantization noise ratio in dB

// Average signal to quantization noise ratio
ASQNR=6.02*n;                                  //
   Average signal to quantization noise ratio in dB

// Displaying the result in command window
printf('\n Effective transmission bit rate = %0.1f
   kbps',bitrate*10^-3);
printf('\n Peak signal to quantization noise ratio =
    %0.2f dB',PSQNR);
printf('\n Average signal to quantization noise
   ratio = %0.2f dB',ASQNR);
```

Scilab code Exa 8.3 `To compute the minimum encoding rateof given 4 sub band coder`

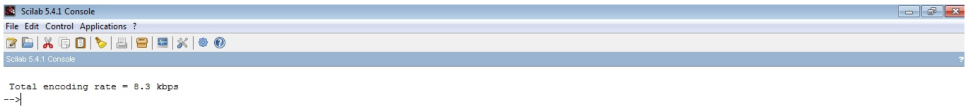

Figure 8.3: To compute the minimum encoding rateof given 4 sub band coder

```
// Example no 8.3
// To compute the minimum encoding rateof given 4
   sub-band coder
// Page no. 427

clc;
clear all;

// Given data
N=4;
   // Total number of sub-bands
L1=225;                                                          //
    Lower limit of first sub-band
U1=450;                                                          //
    Lower limit of first sub-band
L2=450;                                                          //
    Lower limit of second sub-band
U2=900;                                                          //
```

```
Lower limit of second sub-band
L3=1000;                                   //
    Lower limit of third sub-band
U3=1500;                                   //
    Lower limit of third sub-band
L4=1800;                                   //
    Lower limit of fourth sub-band
U4=2700;                                   //
    Lower limit of fourth sub-band
E1=4;
    // Encoding bit of first sub-band
E2=3;
    // Encoding bit of second sub-band
E3=2;
    // Encoding bit of third sub-band
E4=1;
    // Encoding bit of fourth sub-band

// Sampling rate of the sub-bands according to
    Nyquist theorem
sr1=2*(U1-L1);                             //
    Sampling rate of first sub-band in samples/second
sr2=2*(U2-L2);                             //
    Sampling rate of second sub-band in samples/
    second
sr3=2*(U3-L3);                             //
    Sampling rate of third sub-band in samples/second
```

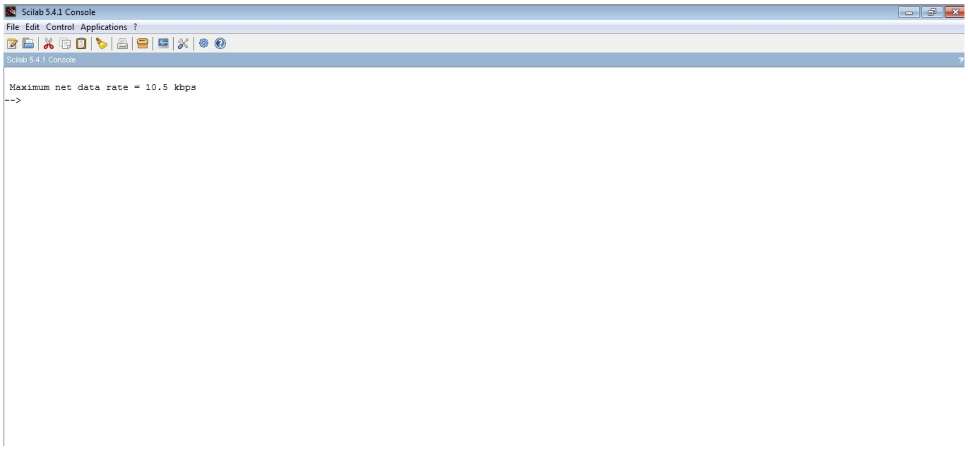

Figure 8.4: To find the upper bound of the transmission bit rate

```
sr4=2*(U4-L4);                                          // Sampling rate of fourth sub-band in samples/second

// Total encoding rate
SR=sr1*E1+sr2*E2+sr3*E3+sr4*E4;                    // Total encoding rate in bps

// Displaying the result in command window
printf('\n Total encoding rate = %0.1f kbps',SR*10^-3);
```

Scilab code Exa 8.4 To find the upper bound of the transmission bit rate

```
// Example no 8.4
// To find the upper bound of the transmission bit rate
```

```
// Page no. 439

clc;
clear all;

// Given data
FL=810*10^6;    // Lower limit of forward channel frequency band
FU=826*10^6;    // Upper limit of forward channel frequency band
N=1150;    // Number of simultaneous users;
SE=1.68;    // Spectral efficiency in bps/Hz
CR=0.5;    // Coder rate
bandused=90/100;    // 90 % bandwidth is used

bandwidth=bandused*(FU-FL);    // Total bandwidth available for traffic channels in Hz
Cbandwidth=bandwidth/N;    // Maximum channel bandwidth in Hz
ChannelDR=SE*Cbandwidth;    // Maximum channel data rate in bps
DR=ChannelDR*CR;    // Maximum net data rate in bps

```

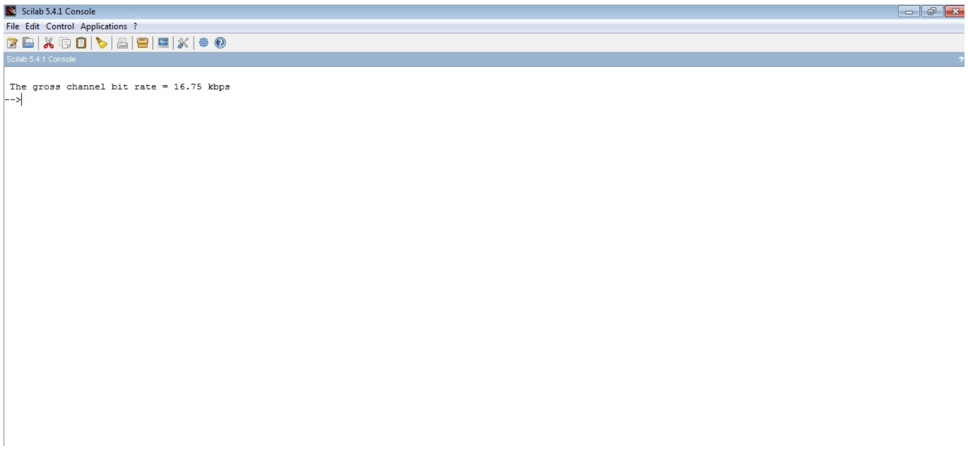

Figure 8.5: To compute the gross channel data rate

```
// Displaying the result in command window
printf('\n Maximum net data rate = %0.1f kbps',DR
    *10^-3);
```

Scilab code Exa 8.5 To compute the gross channel data rate

```
// Example no 8.5
// To compute the gross channel data rate
// Page no. 439

clc;
clear all;

// Given data
t=20*10^-3;                                          //
    Duration of encoding of one block in second
B1=50;                                                    //
```

```
The first bits in Type-1 channel
CRC1=10;                                    // Number of CRC bits in Type-1 channel
FEC=0.5;                                    // FEC rate for Type-1 channel
B2=132;                                     // Next bits in Type-2 channel
CRC2=5;                                     // Number of CRC bits in Type-2 channel
B3=78;                                      // The last bits in Type-3 channel

N1=(B1+CRC1)/FEC;                           // Total number of bits transmitted in Type-1 channel
N2=(B2+CRC2);                               // Total number of bits transmitted in Type-2 channel
N3=B3;                                      // Total number of bits transmitted in Type-3 channel
N=N1+N2+N3;                                 // Total number of channel bits transmitted enery t seconda

// The gross channel data rate
BR=N/t;                                     // The gross channel data rate in bps

// Displaying the result in command window
```

```
printf('\n The gross channel bit rate = %0.2f kbps',
   BR*10^-3);
```

Chapter 9

Multiple access techniques for wireless communications

Scilab code Exa 9.1 To find the intermodulation frequencies generated

```
// Example no 9.1
// To find the intermodulation frequencies generated
// Page no. 451

clc;
clear all;

// Given data
f1=1930;                                          //
   First carrier frequency
f2=1932;                                          //
   second carrier frequency
F1=1920;                                          //
   Lower frequency of the band
F2=1940;                                          //
   Upper frequency of the band

```

Figure 9.1: To find the intermodulation frequencies generated

```
for n=0:3
    x1=(2*n+1)*f1-2*n*f2
    if x1 < = F2 then
        printf('\n IF frequency %0.0f MHz lies
           inside the band',x1);
    else
        printf('\n IF frequency %0.0f MHz lies
           outside the band',x1);
    end
end

for n=0:3
    x2=(2*n+2)*f1-(2*n+1)*f2
    if x2 < = F2 then
        printf('\n IF frequency %0.0f MHz lies
           inside the band',x2);
    else
        printf('\n IF frequency %0.0f MHz lies
           outside the band',x2);
    end
end

for n=0:3
```

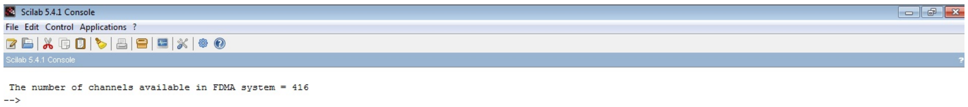

Figure 9.2: To find number of channels available

```
    x3=(2*n+1)*f2-2*n*f1
    if x3 < = F2 then
        printf('\n IF frequency %0.0f MHz lies inside the band',x3);
    else
        printf('\n IF frequency %0.0f MHz lies outside the band',x3);
    end
end

for n=0:3
    x4=(2*n+2)*f2-(2*n+1)*f1
    if x4 < = F2 then
        printf('\n IF frequency %0.0f MHz lies inside the band',x4);
    else
        printf('\n IF frequency %0.0f MHz lies outside the band',x4);
    end
end
```

Scilab code Exa 9.2 To find number of channels available

```
// Example no 9.2
// To find number of channels available
// Page no. 452

clc;
clear all;

// Given data
Bt=12.5*10^6;                                    //
    Total spectrum allocation in Hz
Bguard=10*10^3;                                  // Guard
    band allocated in Hz
Bc=30*10^3;                                      //
    Channel bandwidth in Hz

// The number of channels available
N=(Bt-2*Bguard)/Bc;                              // The
    number of channels available

// Displaying the result in command window
printf('\n The number of channels available in FDMA
    system = %0.0f',N);
```

Scilab code Exa 9.3 To find number of simultaneous users accommodated in GSM

Figure 9.3: To find number of simultaneous users accommodated in GSM

```
// Example no 9.3
// To find number of simultaneous users accommodated in GSm
// Page no. 455

clc;
clear all;

// Given data
m=8;                 // Maximum speech channels supported by single radio channel
Bc=200*10^3;         // Radio channel bandwidth in Hz
Bt=25*10^6;          // Total spectrum allocated for forward link
Bguard=0;            // Guard band allocated in Hz

// The number of simultaneous users accommodated in GSm
N=(m*(Bt-2*Bguard))/Bc;        // The number of simultaneous users
```

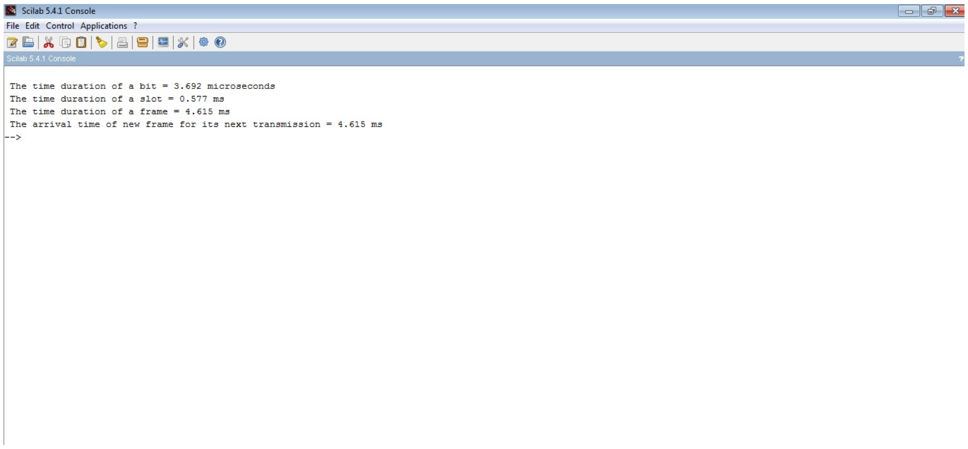

Figure 9.4: To find the time duration of a bit and the time duration of a slot and the time duration of a frame and how long must a user occupying single time slot wait between two successive transmission

```

// Displaying the result in command window
printf('\n The number of simultaneous users
    accommodated in GSM system = %0.0f',N);
```

Scilab code Exa 9.4 To find the time duration of a bit and the time duration of a

```
// Example no 9.4
// To find a)the time duration of a bit b)the time
    duration of a slot c)the time duration of a frame
     d)how long must a user occupying single time
    slot wait between two successive transmission
// Page no. 456

clc;
clear all;

```

```
// Given data
N=8;                          // Number of time slots in each frame
Nb=156.25;                    // Number of in each time slot
DR=270.833*10^3;              // Data rate of transmission in channel

// a)To find the time duration of a bit
Tb=1/DR;                      // The time duration of a bit in sec

// b)To find the time duration of a slot
Tslot=Nb*Tb;                  // The time duration of a slot

// c)To find the time duration of a frame
Tf=N*Tslot;                   // The time duration of a frame

//d) The waiting time between two successive transmission
Tw=Tf;                        // The arrival time of new frame for its next transmission

// Displaying the result in command window
printf('\n The time duration of a bit = %0.3f microseconds',Tb*10^6);
printf('\n The time duration of a slot = %0.3f ms',Tslot*10^3);
```

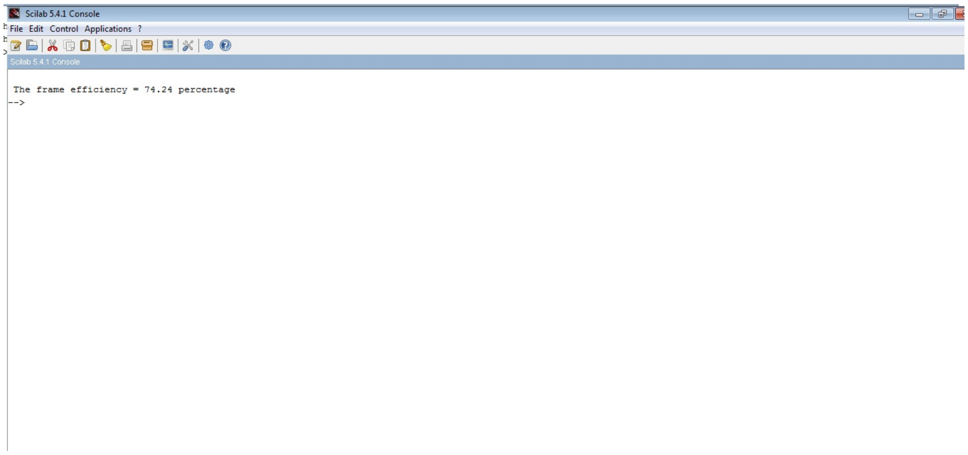

Figure 9.5: To find the frame efficiency

```
printf('\n The time duration of a frame = %0.3f ms',
    Tf*10^3);
printf('\n The arrival time of new frame for its
    next transmission = %0.3f ms',Tw*10^3);
```

Scilab code Exa 9.5 `To find the frame efficiency`

```
// Example no 9.5
// To find the frame efficiency
// Page no. 456

clc;
clear all;

// Given data
Btrail=6;                                        //
    Number of trailing bits per slot
```

```
Bg=8.25;                                    // Number of guard bits per slot
Btrain=26;                                  // Number of training bits per slot
Nb=2;                                       // Number of burst
Bburst=58;                                  // Number of bits in each burst
Nslot=8;                                    // Number of slots in each frame

N=Btrail+Bg+Btrain+2*Bburst;                // Total number of bits in each slot
Nf=Nslot*N;                                 // Total number of bits in a frame
bOH=Nslot*Btrail+Nslot*Bg+Nslot*Btrain;     // Number of overhead bits per frame

// To find the frame efficiency
nf=(1-(bOH/Nf))*100;                        // Frame efficiency

// Displaying the result in command window
printf('\n The frame efficiency = %0.2f percentage',nf);
```

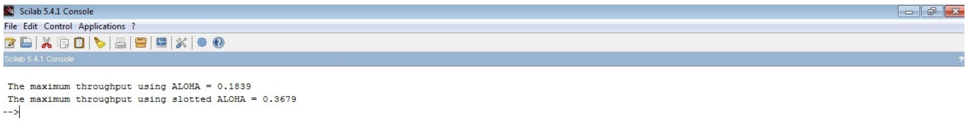

Figure 9.6: To determine the maximum throughput using ALOHA and slotted ALOHA

Scilab code Exa 9.6 To determine the maximum throughput using ALOHA and slotted AL

```
// Example no 9.6
// To determine the maximum throughput using ALOHA
   and slotted ALOHA
// Page no. 466

clc;
clear all;

//The maximum throughput using ALOHA
Rmax=1/2;                                               //
   Maximum rate of arrival calculated by equating
   ALOHA throughput formula derivative to zero
T=Rmax*exp(-1);                                    //The
   maximum throughput using ALOHA

```

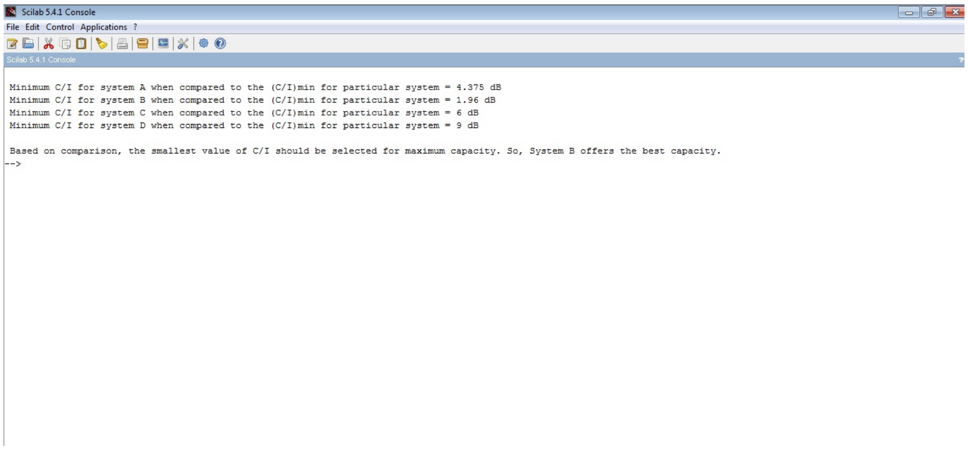

Figure 9.7: To evaluate 4 different radio standards and to choose the one with maximum capacity

```
// Displaying the result in command window
printf('\n The maximum throughput using ALOHA = %0.4
    f',T);

//The maximum throughput using slotted ALOHA
Rmax=1;                                                    //
    Maximum rate of arrival calculated by equating
    slotted ALOHA throughput formula derivative to
    zero
T=Rmax*exp(-1);                                        //The
    maximum throughput using slotted ALOHA

// Displaying the result in command window
printf('\n The maximum throughput using slotted
    ALOHA = %0.4f',T);
```

Scilab code Exa 9.7 To evaluate 4 different radio standards and to choose the one

```
// Example no 9.7
// To evaluate 4 different radio standards and to
   choose the one with maximum capacity
// Page no. 472

clc;
clear all;

// Given data
ABc=30*10^3;                                              //
   Channel bandwidth of system A
ACImin=18;                                                  //
    The tolerable value of carrier to interference
   ratio for system A
BBc=25*10^3;                                              //
   Channel bandwidth of system B
BCImin=14;                                                  //
    The tolerable value of carrier to interference
   ratio for system B
CBc=12.5*10^3;                                          //
   Channel bandwidth of system C
CCImin=12;                                                  //
    The tolerable value of carrier to interference
   ratio for system C // Value of CCImin is given
   wrong in book
DBc=6.25*10^3;                                          //
   Channel bandwidth of system D
DCImin=9;                                                   //
```

```
The tolerable value of carrier to interference ratio for system D
Bc=6.25*10^3;                                                   // Bandwidth of particular system

ACIeq=ACImin+20*log10(Bc/ABc);                        // Minimum C/I for system A when compared to the (C/I)min for particular system
BCIeq=BCImin+20*log10(Bc/BBc);                        // Minimum C/I for system B when compared to the (C/I)min for particular system
CCIeq=CCImin+20*log10(Bc/CBc);                        // Minimum C/I for system C when compared to the (C/I)min for particular system
DCIeq=DCImin+20*log10(Bc/DBc);                        // Minimum C/I for system D when compared to the (C/I)min for particular system

// Displaying the result in command window
printf('\n Minimum C/I for system A when compared to the (C/I)min for particular system = %0.3f dB',ACIeq);
printf('\n Minimum C/I for system B when compared to the (C/I)min for particular system = %0.2f dB',BCIeq);
printf('\n Minimum C/I for system C when compared to the (C/I)min for particular system = %0.0f dB',CCIeq);
printf('\n Minimum C/I for system D when compared to the (C/I)min for particular system = %0.0f dB',DCIeq);
printf('\n \n Based on comparison, the smallest value of C/I should be selected for maximum
```

Figure 9.8: To determine the maximum number of users using omnidirectional base station antenna and no voice activity and three sectors at the base station and voice activity detection

```
    capacity. So, System B offers the best capacity.')
```

Scilab code Exa 9.9 To determine the maximum number of users using omnidirectional

```
// Example no 9.9
// To determine the maximum number of users using a) omnidirectional base station antenna and no voice activity b)three-sectors at the base station and voice activity detection
// Page no. 472

clc;
clear all;

// Given data
```

```
W=1.25*10^6;
    // Total RF bandwidth in Hz
R=9600;
    // Baseband information bit rate in bps
EbNo=10;
    // Minimum acceptable SNR in dB

// a)Maximum number of users using omnidirectional base station antenna and no voice activity
N1=1+(W/R)/EbNo;
    // Maximum number of users using omnidirectional

// b)Maximum number of users using three-sectors at the base station antenna and voice activity with alpha=3/8
alpha=3/8;
    // Voice activity factor
Ns=1+(1/alpha)*((W/R)/EbNo);
    // Maximum number of users
N2=3*Ns;
    // Maximum number of users using three-sectors

// Displaying the result in command window
printf('\n Maximum number of users using omnidirectional base station antenna and no voice activity = %0.0f',N1);
printf('\n Maximum number of users using three-sectors at the base station antenna and voice activity (with alpha=3/8) = %0.0f',N2);
```