Scilab Textbook Companion for Wireless Communications and Networking by V. Garg[1]

Created by
Vishal Changlani
Electronics and telecommunication engineering
Electronics Engineering
VESIT
College Teacher
None
Cross-Checked by
Spandana

June 7, 2016

[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 and Networking

Author: V. Garg

Publisher: Morgan Kaufmann, San Francisco.

Edition: 2

Year: 2007

ISBN: 978-0-12-373580-5

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

Chapter 2

Multiple Access Techniques

Scilab code Exa 2.1 To get Gos during Busy Hour

```
// Exa 2.1
// TO get Gos during BH(Busy Hour).

clc;
clear all;

LC=10; //Lost calls
CC=380; //Carried calls

// soution
OC=LC+CC;  //Total offered calls
//Gos=Blocking probability=(number of Lost calls/
   Total number of offered calls)
Gos=LC/OC;
printf('The Gos during busy hour is %f \n ', Gos);
```

Scilab code Exa 2.2 To find usage in CCS and Erlangs

```
// Exa 2.2
// To find usage in call-seconds,CCS(centrum call seconds) and Erlangs).

clc;
clear all;

Ht=5; //Average holding time in seconds
PC=450; // Peg count for one hour period
OC=0;    // Overflow count

// solution
// usage(Erlangs)=(peg count- overflow count)*Average holding time(in hrs)
U=(PC-OC)*(5/3600);
printf('Usage = %.3f Erlangs = ',U);

// IN CCS
Uccs=U*36;  // usage in CCS
printf('%.1f CCS = ',Uccs);

Ucs=Uccs *100;  //usage in call-seconds
printf('%d call-seconds\n',Ucs);
```

Scilab code Exa 2.3 To find offered load

```
// Exa 2.3
//TO find Offered Load.

clc;
clear all;

B=0.05;  //Blocking(5%)
CL=3000;  //Carried Load in CCS

```

```
//Solution
//Offered Load=Carried Load/(1-Blocking);
OL=CL/(1-B);   //Offered Load in CCS
printf('Offered load is %d CCS \n',round(OL));
printf(' Overflow is %d CCS \n',round(OL)-CL);
```

Scilab code Exa 2.4 To find traffic intensity

```
// Exa 2.4
 //To find traffic intensity.

clc;
clear all;

Ht=120;   //Holding time in seconds
CR=2;   //call rate per hour

//solution
//Traffic Intensity(Erlangs) = call rate * Holding
    time(in hrs));
I=CR*Ht/3600; //Traffic Intensity
printf('Traffic Intensity is %.4f Erlangs = %.1f CCS
     \n',I,I*36);
```

Scilab code Exa 2.5 To find traffic intensity

```
//Exa 2.5
//To find traffic intensity in Erlangs and CCS.
//Refer-Table 2.1(page No 28): Traffic data used to
    estimate traffic intensity

clc;
clear all;
```

```
time=90;  //in minutes
calls=10;  //no of calls in 90mins

//solution
CR=calls/(time/60);    //call arrival rate in calls/
   hour
tavg=(60+74+80+90+92+70+96+48+64+126)/10;    //
   average call holding time in sec per call
I= CR*(tavg/3600);    //traffic intensity in
   Erlangs
printf('Traffic Intensity is %.3f Erlangs = %.2f CCS
    \n ',I,I*36);
```

Scilab code Exa 2.6 To find traffic intensity in Erlangs and CCS during eight hour period and busy hour

```
// Exa 2.6
//To find traffic intensity in Erlangs and CCS
   during eight hour period and busy hour(4:00 PM to
    5:00 PM).
//Refer-Table 2.2(page no 28): Traffic on customer
   line between 9:00PM and 5:00PM

clc;
clear all;

time=8;  //in hours
calls=11;  //no of calls in 90mins period

//solution
CR=calls/time;    //call arrival rate in calls/hour
tavg=(3+10+7+10+5+5+1+5+15+34+5)/11;    //average
   call holding time in mins per call
I= CR*(tavg/60);    //traffic intensity in Erlangs
```

```
printf('Traffic Intensity during eight hour period
    is %.3f Erlangs = %.1f CCS \n',I,I*36);

//For Busy Hour i.e between 4:00PM and 5:00PM
CRB=2;        //call arrival rate during busy hour in
     calls/hour(from table 2.2)
tavgB=(34+5)/2;  //average holding time during Busy
    Hour in mins/call (from table 2.2)
IB=CRB*(tavgB/60); //Traffic Intensity during Busy
    Hour
printf(' Traffic Intensity during busy hour is %.2f
    Erlangs = %.1f CCS',IB,IB*36);
```

Scilab code Exa 2.7 To find traffic per subscriber per BH in Erlangs

```
//Exa 2.8
// To find traffic per user per BH in Erlangs.

clc;
clear all;

minutes=500; //No of minutes used per month per user
Twork=0.9; //Traffic During Work day
TBH=0.1;  //Traffic during busy hour
Days=20;  //No of workdays in a month

//solution
//Avg BH usage per subscriber (in minutes) = minutes
    *Twork(TBH/Days);
Traffic=minutes*Twork*(TBH/Days);
printf('Traffic per user per BH is %.4f Erlangs \n',
    Traffic/60);
```

Scilab code Exa 2.8 To find average number of BHCAs

```
//Exa:2.8
// To find average busy hour call attempts(BHCA's).

clc;
clear all;

minutes=500; //No of minutes used per month per user
Twork=0.9; //Traffic During Work day
TBH=0.1;  //Traffic during busy hour
Days=20;  //No of workdays in a month
MeanHT=100;  //Mean holding time(in secs)

//solution
//Avg BH usage per subscriber (in minutes) = minutes
    *Twork(TBH/Days);
//BHCAs=traffic (in Erlangs) *3600/(meanHT)

Traffic=minutes*Twork*(TBH/Days);
BHCAs=(Traffic/60)*(3600/MeanHT);

printf('Average Busy Hour call attempts are %.2f ',
    BHCAs);
```

Scilab code Exa 2.9 To calculate total traffic and number of MSCs required

```
//Exa 2.9
// To find total traffic in Erlangs and no of MSCs
    required to handle it.

clc;
clear all;

```

```
Tpopu=200000;//Total population
SP=0.25; //subscriber penetration
HT1=100;  //holding time for Mobile to Land line and vicecersa
c1=3;  //Avg calls/hr for Mobile to Land line and vicecersa
HT2=80;  //For mobile to mobile
c2=4;   //For mobile to mobile
TMSC=1800;  //traffic one msc can hold
TrafDist=0.9 //Traffic distribution for Mobile to Land line and vicecersa

//solution
aM_L=c1*HT1/3600;  //Traffic Generated by Subscriber (M–L and  L–M).
aM_M=c2*HT2/3600;  //Traffic Generated by Subscriber (M–M).
WlessSub=SP*Tpopu;  //total wireless subscribers
TotalTraffic=WlessSub*TrafDist*aM_L+WlessSub*(1-TrafDist)*aM_M;
MSCreqd=TotalTraffic/TMSC;
if(MSCreqd-int(MSCreqd)>0)//for rounding of to next integer ef 2.33 to 3
    MSCreqd=MSCreqd+1;
     end
printf('Total Traffic is %.1f Erlangs \n',TotalTraffic);
printf(' NO of MSCs Required are %d \n',int(MSCreqd));
```

Scilab code Exa 2.10 To find the designed cell capacity for the switch and design Erlangs

```
//Exa_2.10
// TO find ABS/BH(average busy season per busy hour)
```

```
calling rates, design cell capacity for the switch and design Erlangs.

clc;
clear all;

Rlines=15000;//Residential lines
Blines=80000;//Business lines
PWElines=5000;//PBX, WATS, and Foreign Exchange (FX) lines
CR_R=2;//Call rates for Rlines
CR_B=3;// call rates for Blines
CR_PWE=10;//call rates for PWElines
HT_R=140;//average holding time for Rlines(sec)
HT_B=160;//average holding time for Blines(sec)
HT_PWE=200;//average holding time for PWE lines(sec)
Slines=100000;// no of lines carried by switch
HD_ABS=1.5;// HD/ABS for the switch

//solution
percentR_lines=Rlines/Slines;
percentB_lines=Blines/Slines;
percentPWE_lines=PWElines/Slines;
CCSR=CR_R*HT_R/100;
CCSB=CR_B*HT_B/100;
CCSPWE=CR_PWE*HT_PWE/100;

CR=CR_R*percentR_lines+CR_B*percentB_lines+CR_PWE*percentPWE_lines;
printf('The call rate is %.1f calls per line \n ',CR);
CCS=CCSR*percentR_lines+CCSB*percentB_lines+CCSPWE*percentPWE_lines;
AvgHTperline=CCS*100/CR;
ABS_BH_calls=CR*Slines;
ABS_BH_usage=CCS/36*Slines;
printf('Design cell capacity based on HD is %d calls \n',HD_ABS*ABS_BH_calls);
```

```
printf(' DESIGN Erlangs based on HD is %d \n',round(
    HD_ABS*ABS_BH_usage));
```

Scilab code Exa 2.11 To find number of service channels required to handle the load

```
// Exa 2.11
// To find offered load A and number of service
    channels required to handle the load

clc;
clear all;

maxcalls_hour=4000;//maximum call per hour
avgHT=160;//average holding time in sec
Gos=0.02;

//solution
A=maxcalls_hour*avgHT/3600; //offered load
printf('Offered load A = %.2f Erlangs \n ',A);
//Refering Appendix A i.e Erlang B table
disp("For calculated Offered load and referring
    Erlang B table we get Service channels as 182
    giving 168.3 Erlangs at 2% blocking");
```

Scilab code Exa 2.12 To find the number of supported mobile subscribers

```
// Exa 2.12
// To calculate number of mobile subscribers
    supported for the given system.

clc;
clear all;
```

```

channels=50;
blocking=0.02;
HT=120;//average holding time inm sec
BHcall=1.2;// in calls  per hour

//solution
//Refering Erlang B table in appendix A, For 50
    channels at 2% blocking , the offered load=40.26
    Erlangs.
A=40.26;
B=A*(1-0.02); //carried load
Avgtraffic_user=BHcall*HT/3600;
No_users=B/Avgtraffic_user;
printf('NO of mobile subscribers supported are %d \n
    ',round(No_users));
```

Chapter 3

Radio Propagation and Propagation PathLoss Models

Scilab code Exa 3.1 To find free space and reflected surface attenuations

```
// Exa 3.1
// To determine free space and reflected surface
   attenuations.

clc;
clear all;

hb=100; //in feets(height of BS antenna)
hm=5;  // in feets(height of mobile antenna)
f=881.52;//in MHz
lamda=1.116;  //in feet
d=5000;  //in feet
Gb=10^0.8;  //8dB(BS antenna gain)
Gm=10^0;   // 0dB (Mobile antenna gain)

//solution
free_atten=(4*%pi*d/lamda)^2*(Gb*Gm)^-1;
y=round(10*log10(free_atten));
printf('Free space attenuation is %d dB \n',y);
```

```
reflect_atten= (d^4/(hb*hm)^2)*(Gb*Gm)^-1;
x=round(10*log10(reflect_atten));
printf(' Reflecting surface attenuation is %d dB \n
    ',x);
```

Scilab code Exa 3.2 To find received signal power and SNR

```
//Exa 3.2
//To determine received signal power and SNR ratio.

clc;
clear all;

d=8000; //Distance between base station and mobile
    station
f=1.5*10^9;//in Hz
lamda=0.2;  //in metres
Pt=10; //BS transmitted power in watts
Lo=8;   //Total system losses in dB
Nf=5;    //Mobile receiver noise figure in dB
T=290;   //temperature in degree kelvin
BW=1.25*10^6; //in Hz
Gb=8;  //in dB
Gm=0;   //in dB
Hb=30;  //in metres
Hm=3;   //in metres
B=1.38*10^-23; //Boltzmann's constant

//solution
Free_Lp=20*log10(Hm*Hb/d^2);
Pr=Free_Lp-Lo+Gm+Gb+Pt;  //in dBW
Te=T*(3.162-1);
Pn=B*(Te+T)*BW;
printf('Received signal power is %d dBW \n',10*log10
    (Pn));
```

```
SNR=Pr-10*log10(Pn);
printf(' SNR ratio is %d dB \n',round(SNR));
```

Scilab code Exa 3.3 To find the allowable path loss

```
//Exa 3.3
//To determine received power and allowable Path
   loss.

clc;
clear all;

d=3*1000;//in metres
Y=4;// path loss exponent
Pt=4;  //Transmitted power in watts
f=1800*10^6;//in Hz
Shadow=10.5;  //in dB
d0=100;//in metres
P0=-32;   //in dBm

//solution
disp("Using equation 3.11 and including shadow
   effect we get")
Pr=P0+10*Y*log10(d0/d)+Shadow;
printf(' Received power is %.1f dBm \n',Pr);
path_loss=10*log10(Pt*1000)-Pr;
printf(' Allowable path loss is %.1f dB \n ',
   path_loss);
```

Scilab code Exa 3.4 To find the distance between transmitter and receiver

```
//Exa 3.4
```

```
//To determine distance between transmitter and
   receiver.

clc;
clear all;

shadow=10; //in dB
Lp=150;  //in dB

//solution
disp(" Using equation given in Problem i.e Lp
   =133.2+40*log(d) we get,");
d=10^((Lp-10-133.2)/40);
printf("  Separation between transmitter and
   receiver as %.2f km',d);
```

Scilab code Exa 3.5 To determine type of fading and symbol distortion

```
// Exa 3.5
// To calculate coherence time, coherence bandwidth,
    type of Symbol distortion and type of fading.

clc;
clear all;

v=60*0.44704;  //.. mph to mps
fc=860*10^6;//in Hz
td=2*10^-6; //RMS delay spread in sec
c=3*10^8;// speed of light in m/sec
Rs=19200; //Coded symbol rate in bps

//solution
lamda=c/fc;
fm=v/lamda; //Maximum doppler shift
tc=1/(2*%pi*fm);//Channel coherence time
```

```
printf('Channel coherence time is %.4f sec \n',tc);
ts=1/Rs; //symbol interval
printf(' Symbol interval is %d microsec \n',ts*10^6)
    ;
disp(" As the symbol interval is much smaller
    compared to the channel coherence time. So,
    Symbol distortion is minimal and fading is slow."
    );
disp("");
Bc=1/(2*%pi*td);
printf('  Coherence Bandwidth is %.2f kHz \n',Bc
    /1000)
```

Scilab code Exa 3.6 To determine number of fades per second and maximum velocity of mobile

```
// Exa 3.6
// TO determine NO of fades per second, average fade
     duration and maximum velocity of mobile.

clc;
clear all;

p=1;// re ection coef cient of ground
c=3*10^8;// velocity of light in free space(m/sec)
e=2.71828;//Euler's number
fm=20;  //in Hz
fc=900*10^6; //carrier frequency in Hz

//solution
Nr=sqrt(2*%pi)*fm*p*e^-(p^2);
printf('NO of fades per second are %.2f  \n',Nr);
Afd=e^-(p^2)/(p*fm*sqrt(2*%pi));
printf(' Average fade duration is %.4f sec \n ',Afd)
    ;
```

```
v=fm*c/fc;
printf('Maximum velocity of mobile is %.2f m/sec =
    %d Km/hour \n',v,v*18/5);
```

Scilab code Exa 3.7 To determine L50 pathloss

```
// Exa 3.7
// To calculate L50 path loss for a PCS system using
    Okumura and COST231 models.

clc;
clear all;

d=[1 2 3 4 5];  //in km
hb=30; //Height of BS antenna in metres
hm=2;// height of mobile antenna in matres
fc=900;//carrier frequency in MHz
W=15;  //street width(m)
b=30;  // distance between building along radio path
    (m)
phi=90; // incident angle relative to the street
hr=30; //in m

//solution
dellhm=hr-hm;
//L50=Lf+Lrts+Lms
// By COST 231 model
Lf=32.4+20*log10(d)+20*log10(fc);
L0=4-0.114*(phi-55);
Lrts=-16.9-10*log10(W)+10*log10(fc)+20*log10(dellhm)
    +L0;
Lbsh=-18*log10(11);
ka=54-0.8*hb;
dellhb=hb-hr;
kd=18-15*dellhb/dellhm;
```

```
kf=4+0.7*(fc/925-1);
Lms=Lbsh+ka+kd*log10(d)+kf*log10(fc)-9*log10(b);
L50=[0 0 0 0 0];
L50=Lf+Lrts+Lms;
//Okumura/Hata model
ahm=(1.1*log10(fc)-0.7)*hm-(1.56*log10(fc)-0.8);
L_50=69.55+26.16*log10(fc)+(44.9-6.55*log10(hb))*
    log10(d)-13.82*log10(hb)-ahm;
xlabel("DISTANCE FROM TRANSMITTER IN KM");
ylabel("PATH LOSS in dB");
plot2d(d,[L50',L_50'],[1,2]);
legends(['Cost 231 Model';'Okumura/Hata Model'],[1,2
    ],opt=2)
xgrid();
disp("L50 values by Cost 231 model");
printf('%.2f  %.2f  %.2f  %.2f  %.2f \n ',L50(1),L50
    (2),L50(3),L50(4),L50(5));
disp("L50 values bu Okumura/Hata model");
printf('%.2f  %.2f  %.2f  %.2f  %.2f \n ',L_50(1),
    L_50(2),L_50(3),L_50(4),L_50(5));
disp("The results from the plot of two models shows
    that the calculated path loss with the COST 231
    model is higher than the value obtained by the
    Okumura/Hata model.");
```

Scilab code Exa 3.8 To determine coverage radius of an access point

```
// Exa 3.8
// TO find coverage radius of an access point.

clc;
clear all;

SNRmin=12;//in dB
n=3; //No of floors
```

```
Backgroundnoise=-115; //dBm
pt=100  //in dBm

//solution
pt_db=10*log10(pt);
Sr=Backgroundnoise+SNRmin; //receiver sensitivity
Lpmax=pt_db-Sr;
//Refering table 3.4
Lp_d0=38;  //ref path loss at the first meter(dB)
Lf=15+4*(n-1); //signal attenuation through n floors
y=3; //path loss exponent
X=10; //Shadowing effect(dB)
d=10^((Lpmax-Lp_d0-Lf-X)/30); //max allowable path loss
printf('Coverage radius of an access point = %d m \n',round(d));
```

Scilab code Exa 3.9 To calculate probability of exceeding signal strength beyond receiver sensitivity

```
//Exa 3.9
// To calculate probability of exceeding signal beyond the receiver sensitivity.

clc;
clear all;

SSmean=-100; //signal strength(dBm)
Sr=-110; //receiver sensitivity(dBm)
sd=10;  //standard deviation(dB)

//solution
P_Smin=(0.5-0.5*erf((Sr-SSmean)/(sqrt(2)*sd)));
printf('probability of exceeding signal beyond the receiver sensitivity is %.2f \n',P_Smin);
```

Scilab code Exa 3.10 To find required transmitter power in watts

```
//Exa 3.10
//To find required total transmit power in Watts.

clc;
clear all;

Lp=140; // path losses in dB
k=1.38*10^-23; // Boltzmann s constant (W/Kelvin-Hz)
k_db=10*log10(k);
f=900;//in MHz
Gt=8; //transmitting antenna gain(dB)
Gr=0; //receiver antenna gain(dB)
Ag=24;//gain of receiver ampli er in dB
Fmargin=8;//Fade margin(dB)
Nf=6;//Noise figure(dB)
L0=20; // other losses in dB
Lf=12; // antenna feed line loss in dB
T=24.6;//Temperature expressed in dB
R=39.8; // data rate in dB
M=8;  //overall link margin(dB)
Eb_No=10;//dB

//solution
//From equation  (3.54)
pt_db=M-Gt-Gr-Ag+ Nf + T+ k_db+ Lp+ Lf+ L0 + Fmargin
   + R+ Eb_No;

Pt=10^(pt_db/10);  //dB into normal number
printf('Total transmitted power is %d Watts \n',Pt);
```

Chapter 4

An Overview of Digital Communication and Transmission

Scilab code Exa 4.1 To determine the sampling rates

```
// Exa 4.1
// To calculate the sampling rate.

clc;
clear all;

Fm=20;  // in KHz

//solution
disp(" An Engineering version of the Nyquist
   sampling rate : fs>=2.2*fm.");
printf('Therefore sampling rate of >= %d ksps should
    be used ',(2.2*Fm));
disp("The sampling rate for a compact disc digital
   audio player = 44.1 ksps and for a studio quality
    audio player = 48 ksps are used.")
```

Scilab code Exa 4.2 To determine the SNR

```
// Exa 4.2
// To calculate SNR for L=32, 64, 128, and 256.

clc;
clear all;

Rt=1; //Resistance(ohm)
//L= Number of quantization values
L1=32;
L2=64;
L3=128;
L4=256;

// solution
// L=2^R i.e R=log2(L);
R1=log2(L1);
R2=log2(L2);
R3=log2(L3);
R4=log2(L4);

//P=A^2/2; //average power of signal
//sig^2=0.333*A^2*2^(-2*Rt); //Avg quantization
   noise power
//SNR=P/sig^2;
// SNR(dB)=1.8+ 6R;

SNR1=1.8+6*R1;
SNR2=1.8+6*R2;
SNR3=1.8+6*R3;
SNR4=1.8+6*R4;

printf('For L=32, SNR is %.1f dB\n ',SNR1);
```

```
printf('For L=64, SNR is %.1f dB\n ',SNR2);
printf('For L=128, SNR is %.1f dB\n ',SNR3);
printf('For L=256, SNR is %.1f dB\n ',SNR4);
```

Scilab code Exa 4.3 To calculate spacing between successive pulses of multiplexed signal

```
// Exa 4.3
// To calculate the spacing between successive
   pulses of the multiplexed signal.

clc;
clear all;

Fs=8*10^3;  //in Hz
Fm=3.4*10^3; // in Hz
VCH=24; //voice channels
SCH=1; //sunchronization channel
PDur=1;  //extra pulse duration in microsec

//solution
Ts=1/(Fs);
TimeCH=Ts/(VCH+SCH)*10^6;   // in microsec
printf('Time between the pulses is %d microsec\n',(
   TimeCH-PDur));
//Now by using the engineering version of Nyquist
   rate sampling
NyquistRate=2.2*Fm;
Ts1_microsec=1/NyquistRate*10^6;
Tc=round(Ts1_microsec)/(VCH+SCH);
printf(" Time between the pulses by using
   engineering version of Nyquist rate sampling  is
   %.2f microsec\n",(Tc-PDur));
```

Scilab code Exa 4.4 To calculate bits per PCM word sampling rate resultant and symbol transmission rate

```
// Exa 4.4
// TO calculate:
// A)The minimum number of bits/sample or bits/PCM
   word that should be used.
// B)The minimum sampling rate, and what is the
   resulting transmission rate.
// C)The PCM pulse or symbol transmission rate.

clc;
clear all;

Fm=3000; //highest modulating frequency in signal(Hz
   )
M=32; // number of pulse levels
b=5; //bits per symbol
p=0.01; //Quantization distortion

//solution
//2^R = L >= 1/2P
// where R is the number of bits required to
   represent quantization levels L
R=log10(1/(2*p))/log10(2);
Fs=2*Fm;  // Nyquist sampling criteria (samples per
   second)
fs=round(R)*Fs;
Rs=fs/b;
printf('The minimum number of bits/sample or bits/
   PCM word that should be used are %d',round(R));
printf('\n The minimum sampling rate is %d samples
   per second\n ',Fs);
printf('The resulting transmission rate is %d bps\n
```

```
   ',fs);
printf('The PCM pulse or symbol transmission rate is
    %d symbols/sec\n',Rs);
```

Scilab code Exa 4.5 To determine choice of modulation scheme

```
// Exa 4.5
// To determine choice of modulation scheme if no-
   error correction coding is used.

clc;
clear all;

S_No=53; //dB-Hz
R=9.6*10^3; //bps
BW=4.8*10^3; //Khz
Pb=10^-5; //BER<=10^-5;

//solution
disp("Since the required data rate of 9.6 kbps is
   more than the available bandwidth of 4.8 kHz, the
    channel is bandwidth-limited.");
Eb_No=S_No-10*log10(R); //dB
// Try for 8-PSK modulation scheme
M=8;
Ps=log2(M)*Pb; //Max ps
Es_No=log2(M)*10^(0.1*Eb_No);
//Ps(8)=2*Q(sqrt(2*Es_No)*sin(%pi/8));
//2*Q(sqrt(2*Eb_No))=erfc(sqrt(Eb_No)); //Refer EQn
    C(7) from appendix C

Ps8=erfc(sqrt(Es_No)*sin(%pi/8));
disp("");
printf(' Symbol error rate is given as %.5f \n ',Ps)
   ;
```

```
printf('The ratio of signal energy to noise is %.2f \n ',Es_No);
printf('Symbol error rate for 8-PSK  is %.5f \n ',Ps8);
disp("As symbol error rate for 8-PSK modulation is lower than threshold value. so, We can use 8-PSK modulation.")
```

Scilab code Exa 4.6 To find choice of modulation scheme without error correcting coding

```
// Exa 4.6
// To determine design choice of modulation scheme without an error correction coding.

clc;
clear all;

SNR=48; //dB-Hz
BW=45*10^3; //in Hz
R=9.6*10^3; //bps
Pb=10^-5; //Bit error rate
e=2.71828; //Natural exponent e

//solution
disp(" since the available bandwidth of 45 kHz is more than adequate to support the required data rate of 9.6 kbps.");
disp("So, the channel is not bandwidth limited ");
Eb_No=SNR-10*log10(R);
//We try the 16-FSK modulation scheme
M=16;

Es_No=log2(M)*Eb_No;
Ps=(M-1)/2*e^(-Es_No/2);
```

```
//For orthogonal signalling
Ps16=(2^M-1)/(2^(M-1))*Pb;
disp("");
printf(' The maximum symbol error probability is %0.5f \n ',Ps16);
printf('The symbol error probability achieved by 16-PSK is %.9f \n ',Ps);
disp("As achieved symbol error probability is far less than maximum tolerable value");
disp("So, we can meet the given speci cations for this power-limited channel with a 16-FSK modulation scheme without any error-correction coding")
```

Chapter 5

Fundamentals of Cellular Communications

Scilab code Exa 5.1 To find system capacity

```
// Exa 5.1
// To Calculate
// A) The system capacity if the cluster size, N (
   reuse factor), is 4 and
// B) The system capacity if the cluster size is 7.
// C) How many times would a cluster of size 4 have
   to be replicated to cover the entire cellular
   area?
// D) Does decreasing the reuse factor N increase
   the system capacity?

clc;
clear all;

ToCH=960;// Total available channels
Cellarea=6; //in km^2
Covarea=2000;//in km^2
N1=4;  // Cluster Size
N2=7;  //Cluster Size
```

```

//solution
Area1=N1*Cellarea;//for N=4
Area2=N2*Cellarea;//For N=7
No_of_clusters1=round(Covarea/Area1);
No_of_clusters2=round(Covarea/Area2);
No_of_CH1=ToCH/N1;    // No of channels with cluster size 4
No_of_CH2=ToCH/N2;    // No of channels with cluster size 7
SysCap1=No_of_clusters1*ToCH;
SysCap2=No_of_clusters2*ToCH;
printf(' System Capacity with cluster size 4 is %d channels \n ',SysCap1);
printf(' Number of clusters for covering total area with N equals 4 are %d \n ',No_of_clusters1);
printf(' System Capacity with cluster size 7 is %d channels \n',SysCap2);
disp("      It is evident when we decrease the value of N from 7 to 4, we increase the system capacity from 46080 to 79680 channels. Thus, decreasing the reuse factor (N) increases the system capacity.")
```

Scilab code Exa 5.2 To find reuse factor for AMPS and GSM

```
// Exa 5.2
// To calculate reuse factor for AMP and GSM systems.

clc;
clear all;

S_IAMP=18;// S/I ratio in dB
S_IGSM=12;// S/I ratio in dB
```

```
PPL=4; // propogation path loss coefficient

//solution
// Using Equation 5.16 on page no 132, we get
N_AMP=(1/3)*((6*10^(0.1*S_IAMP))^(2/PPL));//reuse factor for AMPS

N_GSM=(1/3)*((6*10^(0.1*S_IGSM))^(2/PPL));//reuse factor for GSM

printf('Reuse Factor for AMP system is N = %f = approx %d \n',N_AMP,N_AMP+1);
printf(' Reuse Factor for GSM system is N = %f = approx %d \n',N_GSM,N_GSM+1);
```

Scilab code Exa 5.3 To calculate call capacity of cell and Mean S by I for N as 4 and 7 and 12

```
// Exa 5.3
// To calculate
// A) The number of calls per cell site per hour (i.e., call capacity of cell).
// B) Mean S/I ratio for cell reuse factor equal to 4, 7 and 12.

clc;
clear all;

VCH=395;//Total voice channels
CallHT=120;//average call holding time in sec
Blocking=0.02;// 2%
PPL=4;  //propogation path loss coefficient
N1=4       //reuse factor
N2=7;      //reuse factor
N3=12;     //reuse factor
```

```

//solution
No_of_VCH1=VCH/N1;  //for reuse factor N1
No_of_VCH2=VCH/N2;  //for reuse factor N2
No_of_VCH3=VCH/N3;  //for reuse factor N3
printf('\nNO of voice channels for N=4 are %d',round(No_of_VCH1));
printf('\nNO of voice channels for N=7 are %d',round(No_of_VCH2));
printf('\nNO of voice channels for N=12 are %d\n',round(No_of_VCH3));
disp("Using the Erlang-B traf c table (see Appendix A) for 99 channels with 2% blocking, we nd a traf c load of 87 Erlangs.");
TrafLoad1=87.004;
Carryload1=(1-Blocking)*TrafLoad1;
disp("Using the Erlang-B traf c table (see Appendix A) for 56 channels with 2% blocking, we nd a traf c load of 45.88 Erlangs.");
TrafLoad2=45.877;
Carryload2=(1-Blocking)*TrafLoad2;
disp("Using the Erlang-B traf c table (see Appendix A) for 33 channels with 2% blocking, we nd a traf c load of 24.6 Erlangs.");
TrafLoad3=24.629;
Carryload3=(1-Blocking)*TrafLoad3;
// To find cell capacity
Ncall1=Carryload1*3600/CallHT;//Calls per hour per cell
Ncall2=Carryload2*3600/CallHT;
Ncall3=Carryload3*3600/CallHT;
printf('\ncalls per hour per cell for N=4 are %d',round(Ncall1));
printf('\ncalls per hour per cell for N=7 are %d',round(Ncall2));
printf('\ncalls per hour per cell for N=12 are %d \n',Ncall3);
// To find S BY I
```

```
// N=(1/3)[6*(S/I)]^(2/PPL)
S_I1=10*(PPL/2)*(log10(N1)-log10(1/3)-(2/PPL)*log10
    (6));//Mean S/I (dB)

S_I2=10*(PPL/2)*(log10(N2)-log10(1/3)-(2/PPL)*log10
    (6));
S_I3=10*(PPL/2)*(log10(N3)-log10(1/3)-(2/PPL)*log10
    (6));

printf('\nMean S/I(dB) for N=4 is %.1f',S_I1);
printf('\nMean S/I(dB) for N=7 is %.1f',S_I2);
printf('\nMean S/I(dB) for N=12 is %.1f',S_I3);
```

Scilab code Exa 5.4 To calculate the number of calls per hour per cellsite

```
// Exa 5.4
// To find the number of calls per hour per cell
    site.

clc;
clear all;

spectrum=12.5*10^6; //in Hz
CHBW=200*10^3;//in Hz
N=4;//reuse factor
Blocking=0.02; // 2%
callHT=120;//average call holding time in sec
PPL=4;//propogation path loss coefficient
CntrlCH=3; //No of control channels
Ts=8; // No of voice channels per RF channel

//solution
No_ofVCH=((spectrum*Ts)/(CHBW*N))-CntrlCH;
printf('\n No of voice channels for N=4 are %d',
    No_ofVCH);
```

```
disp("");
disp("Using the Erlang-B traf c table for 122 channels with 2% blocking,we nd a traf c load of 110 Erlangs. ");
TrafLoad=110;
CarryLoad=(1-Blocking)*TrafLoad;
Ncall=CarryLoad*3600/callHT;
printf('\n Calls per hour per cell for N=4 are %d calls/hour/cell \n ',round(Ncall));
S_I=10*(PPL/2)*(log10(N)-log10(1/3)-(2/PPL)*log10(6));
printf('\n Mean S/I(dB) for N=4 is %.1f dB \n ',S_I);
```

Scilab code Exa 5.5 To calculate calls per hour per cellsite and mean S by I and spectral efficiency

```
// Exa 5.5
// To Calculate:
// a) The calls per hour per cell site
// b) The mean S/I ratio
// c) The spectral ef ciency in Erlang/km2/MHz
// for Reuse ratio =4,7,12 and for omnidirectional, 120 degree and 60 degree antenna systems.

clc;
clear all;

VCH=395;//Total allocated voice channels
CHBW=30; // in kHz
Spectrum=12.5; // in MHz
CallHT=120; //Average call holding time in sec
Blocking=0.02; // 2%
PL=40; //slope of path loss in dBperdecade

```

```
//solution
disp("We consider only the first tier interferers and neglect the effects of cochannel interference from the second and other higher tiers.");
//FOR 120degree sectorization
//N=4
VCH11=(VCH/(4*3));
OffLoad11=24.629; // Offered traffic load per sector from Erlang-B table(Appendix A)
Load_site11=3*OffLoad11;
CarLoad11=(1-Blocking)*Load_site11;
Calls_hr_site11=CarLoad11*3600/CallHT;
R11=sqrt(CarLoad11/0.52);
Seff11=CarLoad11/(2.6*Spectrum*R11^2);
S_I11=PL*log10(sqrt(3*4))-10*log10(2);
//N=7
VCH12=(VCH/(3*7));
OffLoad12=12.341; // Offered traffic load per sector from Erlang-B table(Appendix A)
Load_site12=3*OffLoad12;
CarLoad12=(1-Blocking)*Load_site12;
Calls_hr_site12=CarLoad12*3600/CallHT;
R12=sqrt(CarLoad12/0.52);
Seff12=CarLoad12/(2.6*Spectrum*R12^2);
S_I12=PL*log10(sqrt(3*7))-10*log10(2);
//N=12
VCH13=VCH/(3*12);
OffLoad13=5.842; // Offered traffic load per sector from Erlang-B table(Appendix A)
Load_site13=3*OffLoad13;
CarLoad13=(1-Blocking)*Load_site13;
Calls_hr_site13=CarLoad13*3600/CallHT;
R13=sqrt(CarLoad13/0.52);
Seff13=CarLoad13/(2.6*Spectrum*R13^2);
S_I13=PL*log10(sqrt(3*12))-10*log10(2);
//For omnidirectional
//N=4
VCH21=VCH/(4);
```

```
OffLoad21=87.004;  // Offered traf c load per sector from Erlang-B table(Appendix A)
Load_site21=OffLoad21;
CarLoad21=(1-Blocking)*Load_site21;
Calls_hr_site21=CarLoad21*3600/CallHT;
R21=sqrt(CarLoad21/0.52);
Seff21=CarLoad21/(2.6*Spectrum*R21^2);
S_I21=PL*log10(sqrt(3*4))-10*log10(6);
//N=7
VCH22=VCH/(7);
OffLoad22=46.817;  // Offered traf c load per sector from Erlang-B table(Appendix A)
Load_site22=OffLoad22;
CarLoad22=(1-Blocking)*Load_site22;
Calls_hr_site22=CarLoad22*3600/CallHT;
R22=sqrt(CarLoad22/0.52);
Seff22=CarLoad22/(2.6*Spectrum*R22^2);
S_I22=PL*log10(sqrt(3*7))-10*log10(6);
//N=12
VCH23=VCH/(12);
OffLoad23=24.629;  // Offered traf c load per sector from Erlang-B table(Appendix A)
Load_site23=OffLoad23;
CarLoad23=(1-Blocking)*Load_site23;
Calls_hr_site23=CarLoad23*3600/CallHT;
R23=sqrt(CarLoad23/0.52);
Seff23=CarLoad23/(2.6*Spectrum*R23^2);
S_I23=PL*log10(sqrt(3*12))-10*log10(6);
// For 60degree Sectorization
//N=3
VCH31=VCH/(6*3);
OffLoad31=14.902;  // Offered traf c load per sector from Erlang-B table(Appendix A)
Load_site31=6*OffLoad31;
CarLoad31=(1-Blocking)*Load_site31;
Calls_hr_site31=CarLoad31*3600/CallHT;
R31=sqrt(CarLoad31/0.52);
Seff31=CarLoad31/(2.6*Spectrum*R31^2);
```

```
S_I31=PL*log10(sqrt(3*3))-10*log10(1);
//N=4
VCH32=VCH/(6*4);
OffLoad32=10.656;  //  Offered  traf  c  load  per
    sector  from  Erlang-B  table(Appendix  A)
Load_site32=6*OffLoad32;
CarLoad32=(1-Blocking)*Load_site32;
Calls_hr_site32=CarLoad32*3600/CallHT;
R32=sqrt(CarLoad32/0.52);
Seff32=CarLoad32/(2.6*Spectrum*R32^2);
S_I32=PL*log10(sqrt(3*4))-10*log10(1);
//N=7
VCH33=VCH/(6*7);
OffLoad33=5.084;  //  Offered  traf  c  load  per
    sector  from  Erlang-B  table(Appendix  A)
Load_site33=6*OffLoad33;
CarLoad33=(1-Blocking)*Load_site33;
Calls_hr_site33=CarLoad33*3600/CallHT;
R33=sqrt(CarLoad33/0.52);
Seff33=CarLoad33/(2.6*Spectrum*R33^2);
S_I33=PL*log10(sqrt(3*7))-10*log10(1);
//N=12
VCH34=VCH/(6*12);
OffLoad34=2.227;  //  Offered  traf  c  load  per
    sector  from  Erlang-B  table(Appendix  A)
Load_site34=6*OffLoad34;
CarLoad34=(1-Blocking)*Load_site34;
Calls_hr_site34=CarLoad34*3600/CallHT;
R34=sqrt(CarLoad34/0.52);
Seff34=CarLoad34/(2.6*Spectrum*R34^2);
S_I34=PL*log10(sqrt(3*12))-10*log10(1);

printf('For Omnidirectional
    Calls_per_hour_per_cellsite        Mean S_I ratio
          SpecrtalEfficiency\n')
printf('For N=4                              %d
                                  %.1f            %.3f\n',
    Calls_hr_site21,S_I21,Seff21);
```

```
printf('For N=7                    %d                    %.1f            %.3f\n',Calls_hr_site22,S_I22,Seff22);
printf('For N=12                   %d                    %.1f            %.3f\n',Calls_hr_site23,S_I23,Seff23);

printf('For 120deg sector    Calls_per_hour_per_cellsite      Mean S_I ratio      SpecrtalEfficiency\n')
printf('For N=4                    %d                    %.1f            %.3f\n',Calls_hr_site11,S_I11,Seff11);
printf('For N=7                    %d                    %.1f            %.3f\n',Calls_hr_site12,S_I12,Seff12);
printf('For N=12                   %d                    %.1f            %.3f\n',Calls_hr_site13,S_I13,Seff13);

printf('For 60 deg Sector    Calls_per_hour_per_cellsite        Mean S_I ratio      SpecrtalEfficiency\n')
printf('For N=3                    %d                    %.1f            %.3f\n',Calls_hr_site31,S_I31,Seff31);
printf('For N=4                    %d                    %.1f            %.3f\n',Calls_hr_site32,S_I32,Seff32);
printf('For N=7                     %d                   %.1f            %.3f\n',Calls_hr_site33,S_I33,Seff33);
printf('For N=12                    %d                   %.1f            %.3f\n',Calls_hr_site34,S_I34,Seff34);
```

Chapter 6

Multiple Access Techniques

Scilab code Exa 6.1 To calculate spectral efficiency of modulation

```
// Exa 6.1
// To calculate spectral ef ciency.

clc;
clear all;

Area=8;  //in km^2
Cover=4000;// in km^2
CallBH=1.2;  //Avg calls during BH
HT=100; // Avg holding time in sec
Block=0.02; //Blocking=2%
N=4;//Frequency reuse factor
Spectrum=12.5;// in MHz
CHBW=200;// in kHz
User_CH=8;//No of  users per RF channel

//solution
RFCH=Spectrum*1000/CHBW;
TCH=int(RFCH)*User_CH;
SigCH=3;//No of signalling channels per cell
TCH_cell=TCH/N-SigCH;
```

```
Cells=Cover/Area;
OffLoad=108.4;  // in Erlangs
printf('Using Erlang-B Tables, Total traffic offered
    by %d channels at 0.02 blocking = %.1f Erlangs/
   cell \n ',TCH_cell,OffLoad*(1-Block));
CarLoad=OffLoad*(1-Block);
Calls_hr_cell=CarLoad*3600/HT;
MaxUser_hr_cell=Calls_hr_cell/CallBH;
Seff=CarLoad*Cells/(Spectrum*Cover);
printf('Spectral Efficiency is %.2f Erlangs/MHz/km
   ^2\n',Seff);
```

Scilab code Exa 6.2 To find multiple access spectral efficiency for FDMA

```
// Exa 6.2
// To calculate spectral efficiency of FDMA.

clc;
clear all;

TCH=395;  // Traffic Channels
SysBW=12.5;  //in MHz
CHspace=30;  // in kHz

//solution
Eff=TCH*CHspace/(SysBW*1000);
printf('Multiple access spectral efficiency of FDMA
   System is %.3f\n ',Eff);
```

Scilab code Exa 6.3 To find multiple access spectral efficiency of the TDMA system

```
// Exa 6.3
```

```
// To calculate spectral efficiency of TDMA.

clc;
clear all;

Tf=40;  //Frame duration in msec
Mt=6;   // Frames per slot
Bu=30; //bandwidth(KHz) of an individual user during his or her time slot
Nu=395;//  number of users sharing the same time slot in the system, but having access to different frequency sub-bands
Bw=12.5; // in MHz
DR=16.2;//Data rate in kbps
FDur=40; // Frame duration in msec
slots=6; //No of slots per time frame
IndiRate=16.2; //Individual data rate in kbps
Srate=13;  //Speech rate in kbps

//solution
TimeSlot=(Srate/IndiRate)*(FDur/slots);
Seff=TimeSlot*slots*Bu*Nu/(FDur*Bw*1000);
printf('Multiple access spectral efficiency of TDMA is %.2f\n ',Seff);
printf('The overhead portion of the frame is %d percent \n ',round((1-Seff)*100));
```

Scilab code Exa 6.4 To calculate the capacity and spectral efficiency of TDMA system

```
// Exa 6.4
// To calculate capacity and spectral ef ciency of a TDMA system.

clc;
```

```
clear all;

nb=0.9; //BW efficiency factor
u=2;  // Bit Efficiency with QPSK
Vf=1;  // Voice activity factor
BW=12.5; //in MHz
IR=16.2; // in kbps
N=19; //frequency reuse factor

//solution
Nu=nb*u*BW*1000/(Vf*IR*N);// number of channels (
    mobile users) per cell
Seff=int(Nu)*IR/(BW*1000);
printf('Capacity of system is %d mobile users per
    cell\n ',Nu);
printf('Spectral efficiency of TDMA system is %.3f
    bit/sec/Hz\n',Seff);
```

Scilab code Exa 6.5 To calculate the frame efficiency and the number of channels per frame

```
// Exa 6.6
// To calculate frame ef ciency and the number of
    channels per frame.

clc;
clear all;

Nr=2;// number of reference bursts per frame
Nt=24;  // number of traf c bursts (slots) per
    frame(120msec)
FL=120;  //Frame length in msec
Br=148;  // number of overhead bits per reference
    burst
Bp=34; // number of overhead bits per preamble per
```

```
slot
Bg=8.25;//number of equivalent bits in each guard time interval
Tf=120; // frame duration in msec
Rrf=270.83333333;  // bit rate of the RF channel in kbps
R=22.8;  //bit rate of each channel in kbps

//solution
BO=Nr*(8*Br)+Nt*(8*Bp)+(Nt+Nr)*(8*Bg);//The number of overhead bits per frame
Bt=FL*10^-3*Rrf*10^3;//The total number of bits per frame
Eff=(1-BO/Bt)*100;
CH_Frame=(Eff/100)*Rrf/R;//No of channels/frame
printf('The frame efficiency is %.2f percent\n ',Eff);
printf('Number of channels/frame are %d\n',round(CH_Frame));
```

Scilab code Exa 6.6 To calculate capacity and spectral efficiency of the DSCDMA system

```
// Exa 6.6
// To calculate capacity and spectral efficiency of the DS-CDMA system.

clc;
clear all;

nb=0.9;//bandwidth efficiency
nf=0.45;//frequency reuse efficiency
Cd=0.8; //capacity degradation factor
Vf=0.4;//voice activity factor
Eb_I0=7; // desired energy-to-interference ratio in
```

```
dB
L=1;// ef ciency of sector-antenna in cell
BW=12.5;//One way system BW in MHz
R=16.2;//Information rate in kbps

//solution
Eb_I=10^(Eb_I0*0.1);//To convert from dB to a normal value
Nu=(nf*nb*Cd*L/Vf)*(BW*1000/(Eb_I*R));//Capacity of system
Seff=round(Nu)*R/(12.5*10^3);
printf('Capacity of system is %d mobile users per cell\n ',round(Nu));
printf('Spectral efficiency of TDMA system is %.3f bits/sec/Hz\n',Seff);

disp("In these calculations, an omnidirectional antenna is assumed. If a three sector antenna (i.e., G=3) is used at a cell site with lamda(efficiency of sector-antenna in a cell)= 2.6, the capacity will be increased to 325 mobile users per cell, and spectral ef ciency will be 0.421 bits/sec/Hz.")
```

Scilab code Exa 6.7 Compare DS CDMA and TDMA omnidirectional cell

```
// Exa 6.7
// Using the data given in Exa 6.4 and 6.6, compare the capacity of the DS-CDMA and TDMA omnidirectional cell.

clc;
clear all;

//Given Data from Exa 6.4 and Exa 6.6
```

```
Cd=0.8; //capacity degradation factor
R=16.2;//Data rate in kbps
Eb_I0=7; //in dB
Eb_I=10^(Eb_I0*0.1);//To convert from dB to a normal value
Vf=0.4;//voice activity factor
u=2;  // Bit Efficiency
IR=16.2; // in kbps
N=19; //frequency reuse factor
nf=0.45;//frequency reuse ef ciency

//solution
Ncdma_by_Ntdma=Cd*N*nf*IR/(Eb_I*Vf*u*R);
printf('The ratio of capacity of DS-CDMA to TDMA is %.3f\n',Ncdma_by_Ntdma);
```

Scilab code Exa 6.8 To calculate minimum number of PN chips per frequency word

```
// Exa 6.8
// To calculate the minimum number of PN chips that are required for each frequency word.

clc;
clear all;

Bss=600;  //Hopping bandwidth in MHz
stepsize=400; // in Hz

//solution
No_of_Tones=Bss*10^6/stepsize;
Min_chips_required=log2(No_of_Tones);
printf('Minimum number of chips required are %d chips \n ',Min_chips_required);
```

Scilab code Exa 6.9 To find normalized throughput for different CSMA protocols

```
// Exa 6.9
// To calculate the normalized throughput with:
//(a) an unslotted nonpersistent,
//(b) a slotted persistent, and
//(c) a slotted 1-persistent CSMA protocol.

clc;
clear all;

e=2.71828; //Euler's number
Tprop=0.4; //Max propogation delay in sec
R=10; //data rate in Mbps
PackLen=400; //packet length in bits

//solution
Tp=PackLen/R; //packet transmission time in microsec
a=Tprop/Tp;
G=Tp*10^-6*R*10^6/PackLen;//normalized offered traf c load
//Slotted nonpersistent
S0=a*G*e^(-a*G)/(1-e^(-a*G)+a);//normalized throughput
//Unslotted nonpersistent
S1=G*e^(-a*G)/(1+(2*a)+e^(-a*G));//normalized throughput
//Slotted 1-persistent
S2=G*e^(-G*(1+a))*(1+a-e^(-a*G))/((1+a)*(1-e^(-a*G))+a*e^(-G*(1+a)));//normalized throughput
printf('The Normalized throughput with an unslotted non persistent, a slotted persistent and a slotted 1-persistent CSMA protocol are \n %.3f,%
```

```
.3f and %.3f respectively \n',S0,S1,S2);
```

Scilab code Exa 6.10 To find data link protocol efficiency with different protocols

```
// Exa 6.10
// To calculate the data link protocol ef ciency with
//(1) Stop and Wait protocol     full duplex,
//(2) SRP with window size W=8, and
//(3) Go-Back-N protocol with window size W=8.

clc;
clear all;

Tprop=4;  //maximum propogation delay in sec
R=10; // data rate in Mbps
PackLen=400; //data packet length in bits
ACK=20;  //length of ACK packet in bits
Tproc=1; //processing time(sec)
p=0.01;//probability that a data packet or its ACK can be corrupted during transmission

//solution
Tp=PackLen/R; //packet transmission time in microsec
Ta=ACK/R; // transmission time for an ACK in microsec
T=Tp+2*Tprop+2*Tproc+Ta;// total time for transmission time
// Stop and wait ARQ
Eff0=(1-p)*Tp/((1-p)*T+p*Tp);
//SRP with window size W=8
W=8;
Eff1=(2+p*(W-1))/(2+p*(3*W-1));
//Go-Back-N protocol with window size W=8
```

```
Eff2=1/(1+W*(p/(1-p)));
printf('The data link protocol efficiency with Stop
   and Wait protocol, SRP and GBN are \n %.3f, %.3f
   abd %.3f respectively\n',Eff0,Eff1,Eff2);
```

Chapter 8

Speech Coding and Channel Coding

Scilab code Exa 8.1 To calculate the gain in the link budget in dB

```
// Exa 8.1
// To calculate coverage gain in dB.

clc;
clear all;

Pdiff=-3;  //in dB
AMR1=12.2; //in kbps
AMR2=7.95;  //in kbps
AMR3=4.75;  //in kbps

//solution
//CG(dB)=10log{(DPDCH(kbps)+DPCCH)/(DPDCH(AMR bit
   rate (kbps))+ DPCCH)}
CG1=10*log10((AMR1+AMR1*10^(Pdiff/10))/(AMR2+AMR1
   *10^(Pdiff/10)));
CG2=10*log10((AMR1+AMR1*10^(Pdiff/10))/(AMR3+AMR1
   *10^(Pdiff/10)));
printf('By reducing the AMR bit rate from 12.2 to
```

```
7.95 kbps coverage gain becomes %.2f dB \n ',CG1);
printf('By reducing the AMR bit rate from 7.95 to 4.75 kbps coverage gain becomes %.2f dB \n ',CG2);
```

Scilab code Exa 8.2 To calculate output of convolution encoder

```
// Exa 8.2
// To calculate the output of the encoder.

clc;
clear all;

K=4; //constraint length
r=1/2; //code rate(n/k)
x=poly(0,"x");//Defining x as a ploynomial variable
G1=1+x^2+x^3;
G2=1+x+x^2+x^3;
in=[1 0 1 1 1];//input(first bit first)

//solution
//with reference to Fig 8.9 on page no 239
g1=[1 0 1 1]; //converting from G1 polynomial to bit form
g2=[1 1 1 1];////converting from G2 polynomial to bit form
x1=round(convol(g1,in));
x2=round(convol(g2,in));
V1=modulo(x1,2);
V2=modulo(x2,2);
disp("Multiplexing the V1 and V2 to get required output sequence as");
 a=5;
for i= 1:5
```

```
    printf('%d%d',V2(a),V1(a));
    a=a-1;

end
```

Scilab code Exa 8.3 To demonstrate operations of converting burst errors into bit errors

```
// Exa 8.3
// To demostrate 4X4 Bit interleaving/de-interleving
   .

clc;
clear all;

BitStream= [0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1];//Last
   bit to first bit

//solution
disp("Interleaving is performed by storing the data
   in a table containing rows and  columns at the
   transmitter. The data is written in rows and
   transmitted in a vertical direction (according to
    columns). At the receiver, the data is written
   and read in the opposite manner. ")

// Interleaver
              Input1=[1 0 0 0      //Writing data
                row wise
                      1 0 0 0
                      1 1 1 0
                      0 0 0 0];
disp("GIven Bit stream is")
disp(BitStream);
disp("Input to interleaver is")
```

```
disp(Input1);

Output1=[0 0 0 0 0 1 0 0 0 1 0 0 0 1 1 1];     // Reading data column wise
disp("Output of interleaver is");
disp(Output1);
//De-interleaver
              Input2=[1 1 1 0  //Writing o/p data row wise
                      0 0 1 0
                      0 0 1 0
                      0 0 0 0];
 // Let From 6th to 9th bits have Burst Error
 disp("Input to de-interleaver is");
 disp(Input2);
 //Output of deinterleaver

Output2= [0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1];
disp("Output of de-interleaver is")
disp(Output2);
disp( "Bits with Burst error were from 6th to 9th. But in output of de-interleaver , they relocated to positions 3rd, 6th, 10th and 14th.");
```

Chapter 9

Modulation Schemes

Scilab code Exa 9.1 To find Eb by No in dB

```
// Exa 9.1
// To calculate Eb/No in dB for BPSK and Coherent FSK.

clc;
clear all;

Pe=10^-6;//Probability of error
e=2.71828; //Euler's Number

//solution
// For BPSK
//Pe(=10^-6)=e^(-x)/(2*sqrt(%pi*x)); where x=Eb/No

deff('y=f(x)','y=2.71828^(-x)/(2*sqrt(%pi*x))-10^-6');
[x,v,info]=fsolve(0.1,f);

printf('Eb/No For BPSK is %.2f dB\n ',10*log10(x));
printf('FSK requires 3 dB more in terms of Eb/N0 to give the same Pe as BPSK so it comes out to be %
```

```
.2f dB',10*log10(x)+3);
```

Scilab code Exa 9.2 To calculate amplitude A of the carrier signal

```
// Exa 9.2
// To calculate amplitude A of a carrier signal.

clc;
clear all;

Pe=10^-6;//Probability of error
No=10^-10;  // PSD in W/Hz
R=100*10^3;  //data rate in bps

//solution
disp("From Example 9.1, Eb/N0= 10.54dB (11.32) for
   Pe=10^-6 ");
//Therefore
Eb_No=11.32; //From Exa. 9.1
// Eb/No = A^2/(2*No*R);
A=sqrt(2*No*(Eb_No)*R);
printf(' Amplitude of a carrier signal is %.3f mV',A
   *1000);
```

Scilab code Exa 9.3 To calculate final phase for given bitstream for given modulation method

```
// Exa 9.3
// To calculate final phase for the pi/4-DQPSK
   modulation method.

clc;
clear all;
```

```

B=['00','10','01','11','01','00','11','10','10','01','01','00'];//Given Bit stream

//solution
disp("Phase transition table for pi/4-DQPSK Modulation is given as   ")
disp(" By Referring Table 9.1 on page No 266 i.e");
disp("Symbol    Phase transition")
disp("00      =>    45   ");
disp("01      =>   135   ");
disp("10      =>   -45   ");
disp("11      =>  -135   ");
disp("");
disp("sym        Dell phi(k)    Phi(k)")
//BitStream='001001110100111010010100';

phase=0; //Taking initial phase as zero
for i=1:12


    if(B(i)=='00')
      phase=phase+45;
      printf(' %s            45             %d \n',B(i),phase);
    end

    if(B(i)=='01')
        phase=phase+135;
       printf(' %s            135             %d \n',B(i),phase);
    end
    if(B(i)=='10')
        phase=phase-45;
       printf(' %s            -45             %d \n',B(i),phase);
    end
    if(B(i)=='11')
```

```
        phase=phase-135;
        printf(' %s         -135           %d \n',B(i),
           phase);

end
end
disp("");
printf('final phase for the pi/4-DQPSK modulation
   method for given bitstream is %d degree\n',phase)
   ;
```

Scilab code Exa 9.4 To calculate frequency shift along with transmitted frequencies and bandwidth efficiency

```
// Exa 9.4
// To calculate
//(a) the frequency shift between binary 1 and
   binary 0,
//(b) the transmitted frequencies if the carrier
   frequency is 900 MHz, and
//(c) the bandwidth efficiency in bps/Hz.

clc;
clear all;

CHBW=200; //Channel BW in KHz
R=270.83; //Data rate in kbps
Fc=900;  //carrier frequency in MHz

//solution
FreqShift=0.5*R;
//Transmitted Frequencies
Fh=Fc*1000+0.25*R;//Max
Fl=Fc*1000-0.25*R;//Min
BWEff=R/CHBW;
```

```
printf('The frequency shift between binary 1 and binary 0 is %.3f kHz\n ',FreqShift);
printf('Maximum and Minimum value of transmitted frequencies are %.4f mHz and %.4f mHz respectively\n ',Fh/1000,Fl/1000);
printf('Bandwidth efficiency is %.2f bps/Hz',BWEff);
```

Scilab code Exa 9.5 To calculate bit error probability for GMSK

```
// Exa 9.5
// To calculate-
// a) 3-dB bandwidth for a Gaussian low-pass filter,
// b) 99.99% power bandwidth in the RF channel, and
// c) bit error probability for GMSK.

clc;
clear all;

R=270; //data rate in kbps
Eb_No=6; // in dB
GMSK=0.3; //Gaussian minimum shift keying

//solution
Tb=1/R *10^3; //in microsec
B=GMSK/Tb;
printf('3-dB BW for a gaussian low pass filter is %.3f kHz\n',B*1000);
disp("The 3-dB bandwidth is 81.08 kHz. to determine the 99.99% power bandwidth, we use Table 9.3 to find that 1.41*Rb is the required value and degradation factor(beta)= 0.89");
PowerBW=1.41*R;
DegradFac=0.89;
Pe=erfc(sqrt(2*DegradFac*10^(0.1*Eb_No)));
```

```
printf(' Power bandwidth in the RF channel is %.1f kHz\n ',PowerBW);
printf('Bit error probability for GMSK is %f * 10^-5\n',Pe*10^5);
```

Scilab code Exa 9.6 To calculate bit rate of modulator

```
// Exa 9.6
// To calculate bit rate.

clc;
clear all;

Rs=19200; //symbols per second
states=64;

//solution
Bits_symbol=log2(states);
BitRate=Bits_symbol*Rs;
printf('Bit Rate of the modulator is %.1f kbps',BitRate/1000   );
```

Scilab code Exa 9.7 To determine the modulation scheme to be used along with required Eb by No

```
// Exa 9.7
// To determine modulation scheme to be used and Eb/No.

clc;
clear all;

Rb=144; //data rate in kbps
```

```
BW=36; //in MHz
Pb=3*10^-5;//probability of bit error

//solution
Seff=Rb/BW;  //spectral efficiency in bps/Hz

 M=2^(Rb/BW); //since the channel is band limited
 disp("16-QAM (refer Equation 9.66) should be used as it is  more ef cient than 16-PSK (refer Equation 9.50)");
disp("");

//since Q[sqrt(2*Eb_No)]=(1/2)*erfc[sqrt(Eb_No)]             // refer page no 257 equ 9.35
deff('y=f(x)','y=(3/8)*erfc(sqrt((2/5)*x))-Pb');             //from eqn 9.66 and 9.35

[x,v,info]=fsolve(0.1,f);  //x=Eb_No

printf('For a rectangular constellation (refer Figure 9.12), with a Gaussian channel and matched  lter reception, the calculated Eb/No value is %.1f dB\n ',10*log10(x));
```

Scilab code Exa 9.8 Compare the performance of 16PSK with 16QAM

```
// Exa 9.8
// To compare the performance of 16-PSK with 16-QAM.

clc;
clear all;

Pb=10^-8;//BER probability

//solution
```

```
disp("For 16-PSK");
// Pb=0.5*Q(0.552*sqrt(Eb_No));
//since Q[sqrt(2*Eb_No)]=(1/2)*erfc[sqrt(Eb_No)]            // refer page no 257 equ 9.35
deff('y=f(x)','y=0.25*erfc(sqrt(0.5*0.552^2*x))-Pb');
[x,v,info]=fsolve(0.1,f);  //x=Eb_No

printf('Using equation 9.50 we get Eb/No as %d dB (approx) \n ',round(10*log10(x)));
disp("For 16-QAM");
//Pb=0.75*Q(sqrt(0.8*Eb_No));
deff('y=f1(x1)','y=(3/8)*erfc(sqrt(0.4*x1))-Pb'); //x=Eb_No
//since Q[sqrt(2*Eb_No)]=(1/2)*erfc[sqrt(Eb_No)] //refer page no 257 equ 9.35
[x1,v1,info1]=fsolve(0.1,f1);  //x=Eb_No
printf('Using equation 9.66 we get Eb/No as %d dB (approx)\n ',round(10*log10(x1)));
disp("");
printf('Thus 16-QAM has an advantage of about %d dB compared to 16-PSK \n ',10*log10(x)-10*log10(x1));
```

Scilab code Exa 9.9 To find total bandwitdth and its efficiency along with required Eb by No and carried bits per symbol

```
// Exa 9.9
// To calculate-
// a)  The total bandwidth required,
// b)  The bandwidth efciency,
// c)  The Eb/No required, and
// d)  No of carried bits per symbol.

clc;
```

```
clear all;

M=8; //number of different signal elements
Fc=250; //carrier frequency in kHz
DelF=25; //kHz
Pe=10^-6;//probability of error

//solution
TotalBW=2*M*DelF;
nb=2*log2(M)/(M+3);
//Pe=7*Q(z) and z=approx(5.08)
z=5.08;
Eb_No=(z)^2/log2(M);
bits_sym=log2(M);
printf('Total bandwidth required is %d kHz \n ',
   TotalBW);
printf('The bandwidth efficiency is %.4f \n ',nb);
printf('The required Eb/No is %.3f dB \n ',10*log10(
   Eb_No));
printf('Carried bits per symbol are %d \n ',bits_sym
   );
```

Chapter 10

Antennas Diversity and Link Analysis

Scilab code Exa 10.1 To find received signal power and SNR of antenna

```
// Exa 10.1
// To calculate the received signal power at the
   receiver antenna and the SNR of the received
   signal.

clc;
clear all;

D=10000;  //in metres
TxEIRP=30;  //Effective Isotropic Radiated Power(
   EIRP)dBW
lamda=0.2;  //in metres
Pt=10;  //Transmitted power in dBW
Gt=20; //transmitter gain in dBi
Gr=3;  //receiver gain in dBi
Lo=6;//total system losses in dB
Nf=5;  //noise figure in dB
BW=1.25; //mHz
k=1.38*10^-23;  //Boltzmann constant
```

```
T=290;  //temperature in degree kelvin

//solution
Lp=20*log10(lamda/(4*%pi*D));   //free space loss
Pr=Lp+Pt+Gt+Gr-Lo;// received power in dBW

No=10*log10(k*T);  //Noise density in dBW
NO=No+30;  //factor of '30' to convert from dBW to dBm
Pn=Nf+10*log10(BW*10^6)+NO;// noise signal power in dBm
SNR=(Pr+30)-Pn;
printf('The received signal power is %d dBm\n ', round(Pr+30)); //factor of '30' to convert from dBW to dBm
printf('SNR is %d dB\n',SNR);
```

Scilab code Exa 10.2 To find received signal power and SNR of antenna

```
// Exa 10.2
// To calculate the received signal power at the receiver antenna and the SNR of the received signal.

clc;
clear all;

//As we have to use data from Eg 10.1,
D=10000;  // in metres
TxEIRP=30;  //Effective Isotropic Radiated Power(EIRP)dBW
lamda=0.2;  //in metres
Pt=10;  //trasmitted power in dBW
Gt=20; //transmitter gain in dBi
Gr=3;  //receiver gain in dBi
```

```
Lo=6;//total system lossses in dB
Nf=5;  //noise figure in dB
BW=1.25; //mHz
k=1.38*10^-23;  //Boltzmann constant
T=290;  //temperature in degree kelvin
//additional data given in this eg
hr=40; //height of receiver in metre
ht=2; //trasmittter antenna height in metres

//solution
Lp=20*log10(hr*ht/D^2);
Pr=Lp+Pt+Gt+Gr-Lo;// received power in dBW
No=10*log10(k*T);  //Noise density in dBW
N0=No+30;  //factor of '30' to convert from dBW to
    dBm
Pn=Nf+10*log10(BW*10^6)+N0;// noise signal power in
    dBm
SNR=(Pr+30)-Pn;
printf('The received signal power is %d dBm\n ',
    round(Pr+30)); //factor of '30' to convert from
    dBW to dBm
printf('SNR is %d dB\n',SNR);
```

Scilab code Exa 10.3 To calculate gain of antenna

```
// Exa 10.3
// To calculate gain of antenna.

clc;
clear all;

Pin=12;  //Input power in watts
Ploss=3;  //resistive losses in Watts
D=5;  //Directivity

```

```
//solution
Eff=(Pin-Ploss)/Pin;
G=Eff*D;
printf('Gain of the antenna is %.2f dB = %.2f \n'
    ,10*log10(G),G);
```

Scilab code Exa 10.4 To determine 3dB beam width of an antenna

```
// Exa 10.4
// To calculate the 3-dB beam width of a linear
    element antenna.

clc;
clear all;

G=12; //Gain of antenna in dBi

//solution
Theta=101.5/10^(G/10);
printf('The 3-dB beam width of a linear element
    antenna is %.1f degrees',Theta);
```

Scilab code Exa 10.5 To calculate various parameters of helical antenna

```
// Exa 10.5
// To calculate the optimum diameter (DH), spacing (
    S) for the antenna and total length of the
    antenna,
// To calculate the antenna gain,
// To calculate the beam width of the antenna.

clc;
clear all;
```

```

N=12; //number of turns
fr=1.8; //frequency in GHz

//solution
lamda=3*10^8/(fr*10^9);
DH=lamda/%pi;// diameter of helix in milli-meters
S=lamda/4;//turn spacing in millimetres
L=N*S;
G=15*N*S*(DH*%pi)^2/lamda^3;
Theta=52*lamda/(%pi*DH)*sqrt(lamda/(N*S));
printf('The optimim diameter is %d mm\n ',DH*1000);
printf('Spacing is  %.1f mm\n ',S*1000);
printf('Total Length of antenna is %d mm\n ',L*1000);
printf('The antenna gain is %.1f dBi\n ',10*log10(G));
printf('The BeamWidth of antenna is %d degrees \n ',Theta);
```

Scilab code Exa 10.6 To find probability that SNR will drop below 10dB and compare result with a case of without diversity

```
// Exa 10.6
// To find probability that SNR will fall below 10dB (= 10).

clc;
clear all;

E0=1000; //average SNR
Eg=10; //threshold value for SNR
M=3;  //3-Branch Combiner
e=2.71828; //Euler's number

```

```
//solution
x=Eg/E0;
P3=(1-e^(-x))^M; //Considering 3-branch selection combiner
printf('By considering 3-branch selection combiner technique, probability comes to be %d * 10^-6\n ',round(P3*10^6));

disp("When diversity is not used, M =1");
P1=(1-e^(-x));//M=1;
printf(' BY not considering diversity technique, probability comes to be %d * 10^-2 \n',round(P1*10^2));
```

Scilab code Exa 10.7 To calculate minimum delay difference to successfully resolve multipath components

```
// Exa 10.7
// To determine the minimum delay difference to successfully resolve the multipath components and operate the Rake receiver

clc;
clear all;

SR=3.84; //spreading rate in Mcps

//solution
disp("In order to resolve multipath components, the chip    duration should be equalto or greater than T(tau), where T is de ned as ratio of delay distance to speed to electromagnetic wave");
ChipDur=1/(SR*10^6);
Speed=3*10^8;
```

```
Dd=ChipDur*Speed;
disp("");
printf('Minimum delay distance to successfully
   resolve the multipath components and operate the
   Rake receiver is %d m \n',Dd);
```

Chapter 11

Spread Spectrum and CDMA Systems

Scilab code Exa 11.1 To calculate processing gain and improvemet in information rate achieved

```
// Exa 11.1
// To calculate the processing gain and improvement
   in information rate.

clc;
clear all;

CR1=1.2288; //Mcps(Clock rate 1)
CR2=5;//Mcps(Clock rate 2)
R1=9.6; //Information rate in Kbps for CR1
PG2=256;  //Processing Gain for CR2

//solution
PG1=10*log10(CR1*10^3/9.6);//Processing Gain for CR1
R2=CR2*10^3/PG2;//information rate in Kbps for CR2
printf('The processing gain for clock rate 1.2288
   Mcps is %d dB\n ',PG1);
printf('Improvemrnt in information rate is %.2f Kbps
```

```
\n',R2-R1);
```

Scilab code Exa 11.2 Show that the transmitted signals to mobiles 1 2 and 3 are recovered at the mobile receivers by despreading the resultant signal z

```
// Exa 11.2
// To show that the transmitted signals to mobiles
   1, 2, and 3 are recovered at the mobile receivers
    by despreading the resultant signal z(t).

clc;
clear all;

//solution
disp("From figure 11.4, we note that transmitted
   data for mobile1 as [0 1 1 0 0 ], for mobile2 as
   [0 0 1 0 0 ] and for mobile3 as [0 1 0 0 1] ");
disp("From figure 11.5 we get resultant demodulated
   signal at a mobile ");
Rx={[1 1 1 1 -3 1];[1 -3 1 1 1 1 ];[1 -3 1 1 1
   -3];[1 -3 1 1 1 1];[-1 3 3 -1 3 -1]};//Resultant
   demodulated signal at mobile
disp(Rx);
//from Figure 11.4
c1={[-1 -1 -1 -1 1 1];[1 -1 1 1 -1 -1];[1 -1 1 -1 -1
    -1];[-1 1 1 1 -1 1];[1 -1 -1 1 -1 1]};
c2={[1 1 -1 1 1 -1];[-1 1 -1 1 -1 -1];[-1 -1 1 1 1
   -1];[1 1 -1 -1 1 -1];[1 -1 -1 -1 -1 -1]};
c3={[-1 -1 1 -1 1 -1];[-1 -1 -1 1 1 1];[-1 1 1 -1 -1
    1];[-1 1 -1 -1 -1 -1];[1 1 1 -1 1 1]};
//t={[1  2 3 4 5 6];[7 8 9 10 11 12];[13 14 15 16 17
    18];[19 20 21 22 23 24];[25 26 27 28 29 30]};
//for Mobile 1
for i= 1:5
```

```

Demod1(i)=c1(i,1)*Rx(i,1)+c1(i,2)*Rx(i,2)+c1(i,3)*Rx
   (i,3)+c1(i,4)*Rx(i,4)+c1(i,5)*Rx(i,5)+c1(i,6)*Rx(
   i,6);
if(Demod1(i)<0)
    B1(i)=1;
else
    B1(i)=0;
    end
end
//for mobile 2
for i= 1:5

Demod2(i)=c2(i,1)*Rx(i,1)+c2(i,2)*Rx(i,2)+c2(i,3)*Rx
   (i,3)+c2(i,4)*Rx(i,4)+c2(i,5)*Rx(i,5)+c2(i,6)*Rx(
   i,6);
if(Demod2(i)<0)
    B2(i)=1;
else
    B2(i)=0;
    end
end
//for mobile 3
for i= 1:5

Demod3(i)=c3(i,1)*Rx(i,1)+c3(i,2)*Rx(i,2)+c3(i,3)*Rx
   (i,3)+c3(i,4)*Rx(i,4)+c3(i,5)*Rx(i,5)+c3(i,6)*Rx(
   i,6);
if(Demod3(i)<0)
    B3(i)=1;
else
    B3(i)=0;
    end
end
disp("Value of integration at end of bit period for
   mobile1");
disp(Demod1');
disp("Value of integration at end of bit period for
```

```
   mobile2");
disp(Demod2');
disp("Value of integration at end of bit period for
   mobile3");
disp(Demod3');
disp("The  recovered signal at mobile 1 is ");
disp(B1');
disp("The  recovered signal at mobile 2 is ");
disp(B2');
disp("The  recovered signal at mobile 3 is ");
disp(B3');
disp("In all cases, Recovered signal is negated
   value of transmitted signal")
```

Scilab code Exa 11.3 Minimum number of PN chips required for each frequency symbol

```
// Exa 11.3
// To find the minimum number of PN chips.

clc;
clear all;

BW=100; //in MHz
Fspac=10;  //frequency spacing in kHz

//solution
FreqTones=BW*10^3/Fspac;
Chips=log2(FreqTones);
printf('Minimum number of chips required are %d
   chips \n ',Chips);
```

Scilab code Exa 11.4 To calculate data symbol transmitted per hop and number of non overlapping hop frequencies

```
// Exa 11.4
// To  calculate-
//(a) data symbol transmitted per hop, and
//(b) the number of nonoverlapping hop frequencies.

clc;
clear all;

R=120; //transmission rate in kbps
Hop=2000; //per second
Spectrum=10; //in MHz

//solution
//For 32-FSK
Bits_sym=log2(32);
SR=R/Bits_sym;
printf('Bits per symbol are %d \n ',Bits_sym);
printf('Hops per second are 2000 and Symbol rate is
   %d kbps\n ',SR);
disp("Since the symbol rate is higher than the hop
   rate, the system is a slow FHSS system.");
Sym_hop=SR*10^3/Hop;
Min_BW=Sym_hop*SR;
Nonoverlap_hop=Spectrum*10^3/Min_BW;
disp("");
printf(' Symbols transmitted per hop are %d \n ',
   Sym_hop);
printf('Number of non-Overlapping hop frequencies
   are %d \n ',round(Nonoverlap_hop));
```

Scilab code Exa 11.5 To consider various parameters of FHSS systems

```
// Exa 11.5
// To Calculate:
//(a) minimum separation between frequency tones,
//(b) number of frequency tones produced by a
   frequency synthesizer,
//(c) processing gain, and
//(d) hopping bandwidth.

clc;
clear all;

R=200;//input data rate in bps
Fhop=200;//per second
k=1;//Multipication_Factor

//solution
// We have 32-FSK modulation scheme
Bits_sym=log2(32);
Rs=Fhop/Bits_sym;
printf('There are 200 hops per second and Symbol
   rate is %d symbols per sec \n',Rs);
disp("The hop rate is higher than symbol rate, the
   system is a fast FHSS system.");
SDur=1/Rs;
L=Fhop/Rs;
CDur=SDur/L;
Separation=1/CDur;
M=2^Bits_sym;
Hop_BW=k*M*Fhop*L;
Gp=M*k*L;
disp("");
printf(' Minimum separation between frequency tones
   should be %d Hz\n',Separation);
printf(' Number of different frequency tones
   produced by a frequency synthesizer are %d\n',M);
printf(' Processing Gain is %d\n ',Gp);
printf('Hopping bandwidth is %d kHz\n',Hop_BW/1000);
```

Scilab code Exa 11.6 To show how mobile1 will detect its information using a two path rake receiver

```
// Exa 11.6
// To show how the signal is recovered at mobile by
   using two−rake receivers.

clc;
clear all;

//solution
disp("As we are given that actual bit values for
   mobile are [1 0 0 1 1] ");
M1=[1 0 0 1 1];

Rx1={[1 1 1 1 -3 1];[1 -3 1 1 1 1 ];[1 -3 1 1 1
   -3];[1 -3 1 1 1 1];[-1 3 3 -1 3 -1];[1 -1 -1 0 0
   0]};//Resultant demodulated signal at mobile(Z(t)
   ) from path1

Rx2={[-1 -1 1 1 1 1 ];[-3 1 1 -3 1 1 ];[1 1 1 -3 1 1
    ];[1 -3 1 -3 1 1 ];[1 1 -1 3 3 -1 ];[3 1 -1 0 0
   0]};//Resultant demodulated signal at mobile(Z(t
   −2Tc)) from path2

Rx=Rx1+Rx2; //since,Z(t)=z(t)+Z(t−2Tc)

//from Figure 11.13 (d) & Figure 11.14
c1={[-1 -1 -1 -1 1 1];[1 -1 1 1 -1 -1];[1 -1 1 -1 -1
    -1];[-1 1 1 1 -1 1];[1 -1 -1 1 -1 1]};
c2={[-1 1 -1 -1 -1 -1 ];[1 1 1 -1 1 1 ];[-1 -1 1 -1
   1 -1 ];[-1 -1 -1 1 1 1 ];[-1 1 1 -1 -1 1 ];[-1 1
   0 0 0 0]};

```

```
//case-1:Z(t)*C1(t);
for i= 1:5

Demod_1(i)=c1(i,1)*Rx(i,1)+c1(i,2)*Rx(i,2)+c1(i,3)*
   Rx(i,3)+c1(i,4)*Rx(i,4)+c1(i,5)*Rx(i,5)+c1(i,6)*
   Rx(i,6);
if(Demod_1(i)<0)
    B1(i)=1;
else
    B1(i)=0;
    end
end

//case-2:Z(t)*C1(t-2Tc);
for j=1:5

Demod_2(j)=c2(j,3)*Rx(j,3)+c2(j,4)*Rx(j,4)+c2(j,5)*
   Rx(j,5)+c2(j,6)*Rx(j,6)+c2(j+1,1)*Rx(j+1,1)+c2(j
   +1,2)*Rx(j+1,2);
if(Demod_2(j)<0)
    B2(j)=1;
else
    B2(j)=0;
    end
 end
   disp("case-1:z(t)*c1(t)");
disp("Value of integration at end of bit period for
   mobile(case-1)");
disp(Demod_1');
disp("The recovered signal at mobile(case-1) is ");
disp(B1');
disp("Actual bit values are");
disp(M1);
disp("Recovered and actual values are not matching")
   ;
disp("case-2:z(t)*c1(t-2Tc)");
disp("Value of integration at end of bit period for
   mobile(case-2)");
```

```
disp(Demod_2');
disp("The recovered signal at mobile(case-2) is ");
disp(B2');
disp("Actual bit values are");
disp(M1);
disp("Recovered and actual values are not matching")
    ;
//case3-Sum of path1 and path2
disp("case -3:Sum of path1 & path2 integrator");
disp("Sum of integrator outputs(rake receiver output
    )");
Demod_3=Demod_1+Demod_2;
disp(Demod_3');
for k=1:5

if(Demod_3(k)<0)
    B3(k)=1;
else
    B3(k)=0;
end
end
disp("Detected bit value ");
disp(B3');
disp("Actual bit values are");
disp(M1);
disp("Recovered and actual values are matching");
```

Scilab code Exa 11.7 To calculate the time required by mobile to make the required change

```
// Exa 11.7
// To find power value to be set as a first
    approximation ans time required by mobile station
     to make changes as directed by base station.

```

```
clc;
clear all;

Prm=-97;//the signal strength from the base stations
    in dBm

//The constant ( K ) is the part of the broadcast
   message that is sent to the mobile by the base
   station on the paging channel.
K=-73; //dB
P2=18; //power as directed by BS (dBm)

//solution
Ptm=K-Prm;
printf('The mobile transmitter power be set as a
   first approximation of %d dBm \n ',Ptm);
Pwr_Redu=Ptm-P2;//power reduction
printf('Power reduction = %d dBm \n ',Pwr_Redu);
disp("Therefore, mobile requires 6 decrements each
   at 1.25 ms (1/800 sec) ");
Time=6*1.25;
printf(' Time required by mobile station to make
   changes as directed by base station is %.1f msec\
   n',Time);
```

Scilab code Exa 11.8 To determine the possible pair of soft slope and add intercept values

```
// Exa 11.8
// To determine a possible pair of the SOFT_SLOPE
   and ADD_INTERCEPT values that will trigger the
   mobile station to send a PSMM to the base station
    and if the mobile station is IS-95 compliant,
     nd the value of T_COMP that could trigger the
   mobile station to generate a PSMM.
```

```

clc;
clear all;

P1=-95; //pilot1 in dBm
P2=-100; //pilot2 in dBm
P3=-101; //pilot3 in dBm
P4=-105; //pilot4 in dBm
P5=-102; //pilot in dBm
NoiseP=-107; //Receiver sensitivity(dBm)
Tadd=-13; //dB

//solution
//Pcj = received power of the jth pilot in the candidate set
// Pai= received power of the ith pilot in the active set
Pa1=P1-NoiseP;
Pa2=P2-NoiseP;
Pa3=P3-NoiseP;
Pa4=P4-NoiseP;
Pc5=P5-NoiseP;

X=10*log10(10^(0.1*Pa1)+10^(0.1*Pa2)+10^(0.1*Pa3)+10^(0.1*Pa4)+10^(0.1*Pc5));
disp("Max {14.22 *(SOFT–SLOPE)+ (ADD–INTERCEPT),-13 }<=5");
disp("Thus we have two equations as follows");
disp("14.22*SOFT–SLOPE+ADD–INTERCEPT>=-13 and");
disp("14.22*SOFT–SLOPE+ADD–INTERCEPT<=5");
disp(" Solving these equations, we get SOFT–SLOPE=0.5 and ADD–INTERCEPT=-4");
disp("For an IS-95-compliant mobile station (Pcj-Pai)>=0.5*T–COMP");
disp("Since P1>P2>P3>P4, we replace P4");
T_COMP=(P5-P4)/0.5;
disp("");
printf('The value of T–COMP that could trigger the
```

```
    mobile station to generate a PSMM should be <= %d
    dB (<= %d) .\n ',T_COMP,round(10^(0.1*T_COMP)));
```

Chapter 12

Mobility Management in Wireless Networks

Scilab code Exa 12.1 To evaluate the impact of LUs on the radio resource and calculate MSC or VLR transaction load

```
// Exa 12.1
// To evaluate the impact of LUs on the radio resource and calculate the MSC/VLR transaction load using the  uid  ow model.

clc;
clear all;

P=10000;   //Mobile density(mobiles/km^2)
R=500*10^-3; //km
V=10;  ..//Average moving velocity of a mobile in Kmph
Nc=10; //No of cells per LA
N_LA=5; //Number of LAs per MSC/VLR

//Number of transactions and duration of each transaction to MSC/VLR per LU for different LU types are given in Table 12.1.(page no.374)
```

```

// solution
// L=length (km) of the cell exposed perimeter in an LA
L=6*R*(1/3+1/(2*sqrt(Nc)-3));    //Km
// lamdaLU=number of transactions processed by MSC/VLR in an LA perimeter of the jth cell per hour
LamdaLu=V*P*L/%pi;  //Lus per hour


// case(1)
disp("Case-1:In the  rst case, the jth cell located at the border of two LAs is related to the same MSC/VLR, only intra-VLR LUs are processed in the cell");
R1_LU=LamdaLu/3600*(1*600/1000); //resource occupancy from Table 12.1
disp("");
printf(' The resource occupancy in the jth cell due to MS LUs is  %.1f Erlangs\n',R1_LU);

disp("This requires 18 channels at 1% blocking (refer to the Erlang-B table, Appendix A) or 18/8=2.25 traf c channel (about 1/4 of an RF channel, assuming there are 8 traf c channels per RF channel). ")

//case(2)
disp("case-2:In this case the jth cell is located at the border of two LAs related to two different VLRs.In this case, only inter-VLR LUs will be processed in the cell. We assume 80% of LUs are with TMSI and 20% of LUs are with IMSI");
R2_LU=LamdaLu/3600*(0.8*3500/1000+0.2*4000/1000); //from Table 12.1
disp("");
printf(' The resource occupancy in the jth cell due to MS LUs is  %.2f Erlangs \n',R2_LU);
```

```
disp("This requires 75 channels at 1% blocking (refer to the Erlang-B table, Appendix A) or 75/8=9.38 traf c channels (about 1.25 RF channels).");


disp("MSC/VLR transaction load");

disp("We assume that one LA is in the center of the region and the remaining four LAs are on the border of the region.We also assume that, in the perimeter cells at the border LAs, only intra-VLR LUs are generated. For half of the perimeter cells at the border LAs, only inter-VLR LUs are generated.")

Np=6*sqrt(Nc/3)-3;//Number of cells located on perimeter of an LA
disp("");
printf(' Number of cells where inter-VLR LUs occur will be: %d \n',round(0.5*Np*4));
disp("");
printf(' Number of cells where intra-VLR LUs occur will be: %d \n',4*Nc-16);
disp("");
TNLU=LamdaLu*(2*24+16*(0.8*14+0.2*16));  //from table 12.1
printf(' The MSC/VLR transaction load using the  uid  ow model is %.2f * 10^6 transactions at peak hour \n',TNLU/10^6);
```

Chapter 13

Security in Wireless Systems

Scilab code Exa 13.1 To generate public and private keys for RSA algorithm

```
// Exa 13.1
// To generate public and private keys for RSA
   algorithm.

clc;
clear all;

//Two prime numbers
p=5;
q=7;

//solution
n=p*q;
z=(p-1)*(q-1);
e=input("Choose e such that 1<e<z and e and n are
   coprime= ");
d=input("Choose d such that e*d-1 should be
   exactly divisible by z= ");
printf('Public keys is (%d, %d)\n',n,e);
printf('Private key is (%d, %d)\n',n,d);
```

```

//Results
//Choose _e_such that 1<e<z and e and n are coprime=5
//Choose _d_ such that e*d-1 should be exactly divisible by z= 29
//Public keys is (35, 5)
//Private key is (35, 29)
```

Scilab code Exa 13.2 To determine the secret encrypting key K using DH key exchange algorithm

```
// Exa 13.2
// To determine secret encrypting key K using DH key exchange algorithm.

clc;
clear all;

p=23; //prime number that both parties agreed upon
g=5;// g is primitive mod p
a=6;  //party A choosen number
b=15;  //party B choosen number

//solution
printf('Party A sends to party B as (g^a mod p) = %d \n',modulo(g^a,23));
printf(' Party B sends to party A as (g^b mod p) = %d \n',modulo(g^b,23));
printf(' Party A computes secret key as ((g^b modp)^a mod p) = %d \n',modulo(modulo(g^b,23)^a,p));
printf(' Party B computes secret key as ((g^a modp)^b mod p) = %d \n',modulo(modulo(g^a,23)^b,p));
disp("Thus both parties uses k=2 as secret key for encryption");
```

Chapter 14

Mobile Network and Transport Layer

Scilab code Exa 14.1 To determine minimum possible latency and window size to achieve this latency

```
// Exa 14.1
// To determine the minimum possible latency amd the
    minimum window size that achieves this latency.

clc;
clear all;

O=800*1000; //Object size(Bytes)
S=536*8;  //max Segment Size(in bits)
RTT=0.1; //Round trip-time in sec
R=1*10^6; //Transmission rate of the link from the
   server to the client in bps

//solution
Lmin=2*RTT+(O/R); //latency(msec)
// For minimum latency (S/R) +RTT -(W*S/R) = 0;
   Therefore
W=1+(RTT)/(S/R);
```

```
printf('The minimum possible latency is %d sec \n',Lmin);
printf(' The minimum window size is %.1f segments \n',W);
```

Scilab code Exa 14.2 To calculate upper bound of the throughput for TP connection and throughput with retransmission due to errors

```
// Exa 14.2
// To determine the upper bound of the throughput and the throughput with retransmissions due to errors.

clc;
clear all;

RTT=0.1; //Round trip-time in sec
MSS=536*8; //Maximum segment size in bits
p=0.01;// packet loss probability for the path
RTO=5*RTT; //Retransmission time out(from eqn 14.2 on page 450)

//solution
R=0.93*MSS/(RTT*sqrt(p));
RR=MSS/(RTT*sqrt(1.33*p)+RTO*p*(1+32*p^2)*min(1,3*sqrt(0.75*p)));
printf('The upper bound of the throughput is %.4f Mbps \n',R*10^-6);
printf(' The throughput with retransmission due to errors is %.4f Mbps \n',RR*10^-6);
```

Chapter 17

Planning and Design of a Wireless Network

Scilab code Exa 17.1 To calculate various parameters for GSM 1800 network

```
// Exa 17.1
//To calculate-
//(a) average busy-hour traf c per subscriber,
//(b) traf c capacity per cell,
//(c) required number of base stations per zone, and
//(d) the hexagonal cell radius for the zone.

 clc;
 clear all;

 Susage=150;//subscriber usage per month in mins
 days=24;//days per month
 busyhrs=6;//in a day
 BW=4.8*10^3;  //in kHz
 Freqreuse=4/12;//Frequency reuse plan
 chwidth=200; //in kHz
 subscriber=50000;//Present subscriber count
 Sgrowth=0.05;//Growth rate per year
```

```
Area=500;  //in km
BTScapacity=30; //in Erlangs
N=4;  //Initial installation design years

//solution
Erlangspersub=Susage/(days*busyhrs*60);
printf('Average busy-hour traf c per subscriber is %.4f Erlangs \n ',Erlangspersub);
RFcarriers=BW/chwidth;
RFcarrier_percell=RFcarriers/((Freqreuse^-1)*4); //freq reuse factor of 4

//Assuming 2 control channels per cell
CC=2;//control channels
TC_percell=2*RFcarriers/3-CC;
//Referring Erlang-B table in Appendix A
disp("Referring Erlang-B table in Appendix A, Traf c capacity of a GSM cell at 2% GoS for 14 channels = 8.2 Erlangs ");
Tcapacity=8.2;// in Erlangs
disp("There are 3 cells per BTS");
BTS=3;
Traffic_perBTS=Tcapacity*BTS;
printf(' Traffic capacity per BTS is %.1f Erlangs ',Traffic_perBTS);
disp("Therefore, Traffic per BTS is less than BTS capacity(30 Erlangs)")
maxsubscriber=Traffic_perBTS/Erlangspersub;
initialsub=subscriber*(1+Sgrowth)^N;
BTS_perZone=initialsub/maxsubscriber;
printf(' The required number of base stations per zone are %d \n ',round(BTS_perZone));
cellRadius=(Area/(BTS_perZone*2.6))^0.5;
printf('The hexagonal cell radius is %.1f km \n ',cellRadius);
```

Scilab code Exa 17.2 To estimate the data and voice traffic per subscriber and per cell

```
// Exa 17.2
// To calculate voice and data traffic per cell.

clc;
clear all;

usage=150; //subscriber usage per month in mins
days=24;//Days in a month
BHrs=6;//Busy hours per day
BW=4.8;  //in MHz
RFch=200;  //in kHz
Psubscribers=50000;//present subscriber count
growth=0.05;//subscriber growth per year
rollover=4;//network roll over period
NPCS=5;//Number of packet calls per session
NPP=25;//Number of packets within a packet call
Tr=120;//Reading time between packet calls(sec)
NBP=480*8;//Packet size(in bits)
Tint=0.01;//Time interval between two packets(sec)
Ttot=3000;//Total packet service holding time
BH_PS=0.15;//Busy hour packet sessions per
    subscriber
Penetration=0.25;
datarate=48;  //in kbps
PTT=10;//Packet transmission time(sec)
BTS=40;//NO of BTS sites

//solution
Bitstx_duringPTT=NPCS*NPP*NBP/1000;
PST=PTT+Tr*(NPCS-1)+Tint*(NPP-1);
PT_duringBH=BH_PS*Ttot/PST;
```

```
Bits_persub_persec=Bitstx_duringPTT*PT_duringBH
   /(60*60);
VoiceErlangs=usage/(days*BHrs*60);
Initial_subscribers=round(Psubscribers*(1+growth)^
   rollover);
Data_subscribers=Initial_subscribers*Penetration;
Totalvoice=Initial_subscribers*VoiceErlangs;
Voicetraffic_perBTS=Totalvoice/BTS;
printf('Voice Traffic per Cell(sector) is %.2f
   Erlangs \n ',Voicetraffic_perBTS/3);
Totaldata=Data_subscribers*Bits_persub_persec;
Datathroughput_perBTS=Totaldata/BTS;
printf("Data throughput per Cell(sector) is %.2f
   kbps \n ',Datathroughput_perBTS/3);
```

Scilab code Exa 17.3 To calculate data Erlangs along with TS utilization and capacity

```
// Exa 17.3
//Using traf c data per cell for a GSM/GPRS
   network from Example 17.2
// To calculate-
// (a) data Erlangs,
// (b) time slot utilization, and
// (c) TS capacity.

clc;
clear all;

Holdtime=120;//Average holding time during Busy
   Hours(in sec)
Tx=3;// No of transreceivers
TSsig=3;//No of TSs per cell for signaling
RLC=0.80;//Radio link control efficiency
Radioblocks=9000;//Total numbers of transmitted
```

```
radio blocks
TSdata=3;//TSs allocated for data traf c per cell
Datarate=15.5; //From eg 17.2
Voicetraffic=8.82;  //From eg.17.2
Duration=0.02;//Duration of block in sec

//solution
DataEr=Radioblocks*Duration/Holdtime;
printf('Data Erlangs = %.1f \n ',DataEr);
TSuti=DataEr/TSsig;
printf('Time Slot(TS) utilization = %.1f \n ',TSuti);
Throughput=(Datarate/TSdata)*RLC;
TScap=Throughput/TSuti;
printf("TS capacity is %.2f kbps \n ',TScap);
```

Scilab code Exa 17.4 To develop downlink and uplink cell budget and calculate cell radius

```
// Exa 17.4
//To calculate the cell radius.

clc;
clear all;

Pt=36;//Base station transmitted power in dBm
Pms=24;//mobile station transmitted power in dBm
Nms=8;//mobile station noise figure in dB
Nbs=5;//Base station nise figure in dB
Ga=18;//Base station transmit and receive antenna gain in dBi
Gm=0;//Mobile antenna gain in dBi
SNR=12;// in dB
Lc_TX=5;//BS transmit antenna cable, connector, and lter losses in dB
```

```
Lc_RX=2;//BS receiver antenna cable, connector, and lter losses in dB
Bodyloss=3;// Body losses at mobile
fading=10.2;// in dB
ThermalNoise=-174;// in dBm/Hz
Gdiversity=5;//Antenna diversity gain at BS in dB
//Assuming standard value of RF channel as
RFch=200*10^3;  //in Hz

//solution
N=ThermalNoise+10*log10(RFch)+Nms;
Smin=N+SNR;
Smean=Smin+fading+Bodyloss;
Lp=Pt-Lc_TX+Ga-Smean;
N1=ThermalNoise+10*log10(RFch)+Nbs;
Smin=N1+SNR-Gdiversity;
Smean1=Smin+fading+Bodyloss;
Lp1=Pms-Smean1+Ga-Lc_RX;
disp("Using uplink path loss and Hata model to calculate cell radius");
R=10^((Lp1-133.2)/33.8);
printf(' Cell radius is %.1f km \n',R);
```

Scilab code Exa 17.5 To calculate Uplink cell load factor and pole capacity

```
// Exa 17.5
// To calculate uplink cell load factor, number of voice users and poll capacity of the cell.

clc;
clear all;

Ri=12.2*10^3;//Information rate in bps
Rc=3.84*10^6;//Chip rate in cps(chips per second)
```

```
Eb_Nt=4;  //in dB
Imargin=2;  //Interference margin(3 dB)
B=0.5;//Interference factor due to other cells
Vi=0.65;//Channel activity factor

//solution
Eb_Ntreqd=10^(Eb_Nt/10);
LF_peruser=(1+B)*(1/(1+(Rc/Ri)*(1/Eb_Ntreqd)*(1/Vi)));  //M(no of users=1) in Eq 17.13
printf("Cell load factor per voice user is %.5f \n',LF_peruser);
CellLoading=(Imargin-1)/Imargin;
VoiceUsers=CellLoading/LF_peruser;
printf('No of Voice Users are %d per cell \n ',VoiceUsers);
//From EQ 17.6 assuming Power control efficiency=1
Polecap=Rc/(Ri*Vi*(1+B)*Eb_Ntreqd);
printf('Pole Capacity is %d \n',Polecap);
```

Scilab code Exa 17.6 To calculate uplink throughput for WCDMA data service

```
// Exa 17.6
// To calculate Uplink throughput for a WCDMA cell.

clc;
clear all;

Eb_Nt=1;  //in dB
cellLoading=0.5;//Required interference margin(3 dB)
B=0.5;//Interference factor due to other cells
Vi=1;//Channel activity factor

//solution
Eb_Ntreqd=10^(Eb_Nt/10);
```

```
//Assuming standard value of chip rate as 3.84Mcps
Rc=3.84*10^6;//in cps(chips per second)
Throughput=(cellLoading*Rc)/(Eb_Ntreqd*(1+B));
printf('Uplink Throughput is %d kbps \n ',Throughput
   /1000);
```

Scilab code Exa 17.7 To calculate downlink cell load factor and number of voice users per cell

```
// Exa 17.7
// To calculate downlink cell load-factor and number
    of voice users per cell for a WCDMA system.

clc;
clear all;

Ri=12.2*10^3;//Information rate in bps
Rc=3.84*10^6;// Chip rate in chips per second
Eb_Nt=4;    // in dB
Eb_Nt=10^(Eb_Nt/10);
B=0.5;//Average interference factor due to other
   cells
Zeta=0.6; // orthogonality factor
Imargin=2;//Interference margin(3 dB)
Vi=0.65  //assuming Channel activity factor as 0.65

//solution
Loadfactor_peruser=(Zeta+B)*(1/((Rc/Ri)*(1/Eb_Nt)
   *(1/Vi)))
printf('Downlink cell load factor is %.4f \n ',
   Loadfactor_peruser);
cellLoading=(Imargin-1)/Imargin;
Voiceusers=cellLoading/Loadfactor_peruser;
printf('No of voice users per cell are %d \n ',
   Voiceusers);
```

Scilab code Exa 17.8 To determine minimum signal power and maximum allowable path loss

```
// Exa 17.8
// To calculate minimum signal power required and
   maximum allowable path loss.

clc;
clear all;

N0=-174;//Noise density in dBm/Hz
Bc=1.25;//Channel bandwidth in mHz
Rc=1.2288;//Chip rate in Mcps
Nf=6;  //Receiver Noise figure in dB
Pt=27;  //Effective radiated power from mobile in
   dBm
Lct=0.5;  //Transmitter cable and connector loss in
   dBm
Lbody=1.5;//Body loss in dB
Lcr=2;  //Receiver cable and connector loss in dB
Mint=0;  //Interference margin in dB
Mfading=2;//fast fadinf margin in dB
Lpent=8;//Penetration loss in dB
Gm=0;//Transmitter antennna gain in dBi
Gb=12;//Receiver antenna gain in dBi
Fm=8;//Fade margin in dB
Eb_Nt=7;// in dB

//solution
Nth=N0+Nf;
S_Nt=Eb_Nt+10*log10((Rc*10^6)/(Bc*10^6));
Smin=S_Nt+10*log10(Rc*10^6)+Nth;
Lpmax=(Pt-Smin)+(Gb+Gm)-(Lbody+Lct+Lcr+Fm+Lpent)-
   Mint-Mfading;
```

```
printf('Minimum signal power required is %.2f dBm \n
    ',Smin);
printf('Maximum allowable path loss is %.2f dB \n ',
   Lpmax);
```

Scilab code Exa 17.9 To develop a radio link budget for uplink and downlink

```
//Exa 17.9
// To calculate Radio link budget for uplink and
   downlink
// Refering Table 17.1 on page no 613

clc;
clear all;

Rc=3.84;//Chip rate in Mcps
Ri=16;//Data rate in kbps
UL=0.5;//UL loading factor
DL=0.9;//DL loading factor
Eb_NtU=4;//in dB
Eb_NtD=6;// in dB
Gm=0;//Mobile antenna gain in dBi
Gb=18;//Base station gain in dBi

//solution
disp("The Okumara-Hata model for an urban macro-cell
    with a base station antenna height of 25m, a
   mobile station height of 1.5m, and a carrier
   frequency of 1950MHz gives Lp =138.5+35.7*log10(R
   ) where R is radius of hexagonal cell");
disp("From table 17.1, Lp(Allowable path loss) for
   uplink is 139.65 dB");
R=10^((139.65-138.5)/35.7);
printf(' Cell Radius is %.3f km \n ',R);
```

```
Area=round(2.6*R^2);
printf('Area covered by hexagonal cell is %d km^2 \n
    ',Area);
printf('Number of BTSs required to cover an area of
   2400 Km^2 are %d \n ',2400/Area);
```

Scilab code Exa 17.10 To find the number of users supported on the downlink of a WCDMA Network

```
//Exa 17.10
//To calculate No of users that can be supported on
   the downlink of the WCDMA network.

clc;
clear all;

Rc=3.84;//chip rate in Mcps
N=3;//Noise rise in dB
OF=0.8;//orthogonality factor
B=0.55;//Interference from other cells
Eb_N0=4;//in dB
Sec_Eff=0.85;//Sector efficiency
Pwr_Eff=0.80;//Power control efficiency
Y=1.2; //Retransmit rate
X=10; //10MB at 384Kbps
X1=2;  //2MB at 144Kbps
X2=1;  //1MB at 64Kbps

//solution
//Assuming Voice activity=Vf=1
Vf=1;
AvgDR=Y*X*10^6*(1/3600)+Y*X1*10^6*(1/3600)+Y*X2
   *10^6*(1/3600);
CLoad=(N-1)/(N+1);
DLcap=(Rc*10^6*Pwr_Eff*Sec_Eff)/(((10^(Eb_N0/10))*(B
```

```
   +OF)*Vf));
Allowcap=CLoad*DLcap;
users=Allowcap/AvgDR;
printf('NO of users that can be supported on the
   downlink of the WCDMA network are %d \n ',round(
   users));
```

Scilab code Exa 17.11 find the average throughput for various cases

```
// Exa 17.11
// To calculate average throughput and compare it
   with equal latency condition.

clc;
clear all;

P1=1/2;//relative frequency of packets for user
   class1
P2=1/3;//relative frequency of packets for user
   class2
P3=1/6;//relative frequency of packets for user
   class3
R1=16; //data rate in kbps for P1
R2=64; //data rate in kbps for P2
R3=1024; //data rate in kbps for P3
S1=16;//number of slots assigned to the R1 user
S2=8;//number of slots assigned to the R2 user
S3=2;//number of slots assigned to the R3 user

//solution
//Using Equation 17.20 from page no 616
Ravg=(P1*R1*S1+P2*R2*S2+P3*R3*S3)/(P1*S1+P2*S2+P3*S3
   );
// For equal latency, using Eq 17.18
Rav=1/(P1/R1+P2/R2+P3/R3);
```

```
// For Latency ratio=4, using Eq 17.19 from page no
    616
PL=4;
C=(P1+P2+PL*P3)/(P1/R1+P2/R2+P3/R3);
printf('The average throughput for equal access
    condition is %.1f kbps \n ',Ravg);
printf('The average throughput by considering equal
    latency is %.1f kbps \n ',Rav);
printf('The average throughput by considering
    latency ratio as 4 is %.2f kkbps \n ',C)
disp("Ii is observed that equal access provides the
    highest average output")
```

Scilab code Exa 17.12 TO find the allowable throughput of a reverse link of a CDMA 2000

```
// Exa 17.12
// To calculate allowable throughput of reverse link
     in cdma 2000.

clc;
clear all;

Ec_Nt=-23;//in dB
DRC=-1.5;//DRC gain with respect to pilot in dB
Tg=3.75; // Traf c channel gain with respect to
    pilot in dB
B=0.85;//Interference factor due to other cells

//solution
Mmax=(1/(1+10^(DRC/10)+10^(Tg/10)))*(1/((10^(Ec_Nt
    /10)*(1+B))));
//The sector loading can be expressed as a fraction
    of the pole capacity Mmax. This is typically 70%
    of the pole capacity.
```

```
M_allow=int(0.7*Mmax);
//From table 17.2 we get Traffic channel rate as 9.6
   kbps since we are given traffic channel gain with
    respect to pilot as 3.75 dB
Ri=9.6; //in kbps(see table 17.2)
Tput=Ri*M_allow;
printf(' Allowable reverse link throughput is %d
   kbps \n ',round(Tput));
```

Scilab code Exa 17.13 To estimate average SINR of a HSDPA

```
// Exa 17.13
// To calculate average SINR of HSDPA.

clc;
clear all;

Ptmax=5.5;//Maximum transmit power of DSCH in watts
Pbs=18;// Total base station power in watts
alpha=0.2;//downlink orthogonality factor
G=0.363;// geometry factor
SF=16; //Spreading Factor for DSCH;fixed at value of
    16

//solution
// Using equation no 17.27 given on page no 623
SINR=SF*(Ptmax/(Pbs*(1-alpha+(1/G))));
// In dB
SINR_db=10*log10(SINR);

printf('The average SINR of HSDPA is %.1f dB = %.4f
   \n ',SINR_db,SINR);
```

Scilab code Exa 17.14 To find bandwidth of a Iub interface

```
// Exa 17.15
// To calculate bandwidth of Iub interface.

clc;
clear all;

Users=350;//no of users supported
ExpectedTraf=1.8; // From section 17.7 (in Kbps)
BHTraf=1.785;//Busy hour traffic in kbps
BTS=180;

//solution
IubBW=(ExpectedTraf*Users*BHTraf)/1000;  // in Mbps
TotalBW=BTS*IubBW;
printf('Required total bandwidth of Iub Interface is
    %.2f Mbps \n ',TotalBW);
```

Scilab code Exa 17.15 To find the required RNCs

```
// Exa 17.15
// To calculate No of RNC required.

clc;
clear all;

BTS=800;//No of BTS sites
Sec=3;//No of sectors per BTS
freq_sec=2;//No of frequencies used per sector
cellsRNC=1152;//Maximum capacity of cellRNC
btsRNC=384;//One RNC can support btsRNC(BTSs)
VE=25;//Voice service(mErl/subscriber)
BRV=16;// bitrate Voice
CS1=10;//CS data service 1(mErl/subscriber)
```

```
BRC1=32;//bit rate for CS1 in kbps
CS2=5;//CS data service 2(mErl/subscriber)
BRC2=64;////bit rate for CS2 in kbps
PSdata=0.2;//PS data service(kbps per subscriber)
PSoverhead=0.15;
SHO=0.4;//40%
Totalsub=350000;//Total subsribers
Maxcap=196;//Maximum Iub capacity of tpRNC (in Mbps)
FR1=0.9;FR2=0.9;FR3=0.9;// Filler rates

//solution
RNCreqd=(BTS*Sec*freq_sec)/(cellsRNC*FR1);//from eqn
    17.28
printf('using equation 17.28,Number of RNC required
   are %d \n ',round(RNCreqd));
RNC_reqd=BTS/(btsRNC*FR2);//from eqn 17.29
printf('using equation 17.29,Number of RNC required
   are %d \n ',round(RNC_reqd));
RNCrequired=((VE/1000*BRV+CS1/1000*BRC1+CS2/1000*
   BRC2+(PSdata/(1-PSoverhead)))*(1+SHO)*Totalsub)/(
   Maxcap*1000*FR3);//from eqn 17.30
printf('using equation 17.30,Number of RNC required
   are %d \n ',round(RNCrequired));

printf(' We select first value which is %d RNCs \n '
   ,round(RNCreqd));
```

Chapter 19

Wireless Personal Area Network Bluetooth

Scilab code Exa 19.1 To find hopping rate and various other parameters of Bluetooth

```
// Exa 19.1
// To calculate the hopping rate of bluetooth , No of
    bits transmitted in one slot , efficiency of
   packet transmission , and No of times a packet is
   sent .

clc;
clear all;

SS=80; //Frame length of HV3 voice packet
R=64*10^3; //Data rate in bps

//solution
TS=240; //No of bits carried in a slot
//From table 19.3:Bluetooth air interface details ,
   we get Frequency hopping rate as 1600hopes/second

//From table 19.5 , we can note that for HV3, No of
```

```
    slots are 1 and carry in all(80(user voice sample
    )+160(parity bits))=240 bits in one slot packet
HR=1600; //hopes/second
Eff=SS/TS;
x=R/SS;  //x is number of times a packet is sent
printf('Hopping rate is %d hopes/sec \n ',HR);
printf('No of bits transmitted in one slot are %d \n
    ',TS);
printf('Efficiency of packet transmission is %.4f \n
    ',Eff);
printf('Number of times a packet is sent is equal to
    %d \n ',x);
```

Scilab code Exa 19.2 To find the associated data rate of a symmetric 1soot DM1

```
// Exa 19.2
// To find the associated data rate.

clc;
clear all;

R=136; // bits per slot
SR=800; // no of slots per second

//solution
// A symmetric 1-slot DM1 link is setup  between a
    master and a slave
AR=R*SR; //Data rate in bps

printf('Associated data rate is %0.1f kbps \n ',AR
    /1000);
```

Chapter 21

Wireless Local Area Network

Scilab code Exa 21.1 To find number of users supported by WLAN and its bandwidth efficiency

```
// Exa 21.1
// To find number of users that can be supported by
   the WLAN and the bandwidth ef ciency.

clc;
clear all;

Fl=902; //lower limit frequency MHz
Fh=928; //higher limit frequency in MHz
Rt=0.5; //symbol transmission rate in Mega symbols
   per sec
S=16; //No of symbols
BER=10^-5;//Bir error rate
SG=2.6;//sector gain
B=0.5; //Interference factor
a=0.9; //power control efficiency

//solution
BW=Fh-Fl;
Rb=Rt*log2(S);
```

```
Gp=BW/Rb;
// BER = 10^-5= 0.5*erfc(sqrt(Eb_No))
deff('y=f(x)','y=0.5*erfc(sqrt(x))-10^-5')
[x,v,info]=fsolve(0.1,f);//x=Eb_No
M=Gp/x * 1/(1+B) * SG * a;
printf('Number of users that can be supported by the WLAN are %d \n',M);
eff=Rb*int(M)/BW;
printf(' The bandwidth efficiency is %.2f bps/Hz \n',eff);
```

Scilab code Exa 21.2 To find hopping bandwidth and various other parameters for FH MFSK WLAN system

```
//Exa 21.2
// To find-
//a) the hopping bandwidth,
//b) What is the chip-rate,
//c) How many chips are there in each data symbol,
//d) What is the processing gain.

clc;
clear all;

Stepsize=200; //in Hz
Chipsmin=20;//length of linear feedback shift register
Datarate=1.2*10^3;  //bps

//solution
No_of_tones=2^Chipsmin;
Bss=No_of_tones*Stepsize;
Chiprate=Datarate*Chipsmin;
Gp=Bss/Datarate;//processing gain
Symbolrate=Datarate/3;  //8-ary FSK is used
```

```
Chips_symbol=Chiprate/Symbolrate;
printf('The Hopping Bandwidth is %.3f MHz\n',Bss
   /10^6);
printf(' The chiprate is %d kchip/sec\n',Chiprate
   /10^3);
printf(' Chips per symbol are %d \n',Chips_symbol);
printf(' The processing gain is %.1f\n',Gp);
```

Scilab code Exa 21.3 To find bandwidth of a subchannel and various other parameters of a OFDM WLAN system

```
//Exa 21.3
//To find-
// a) The bandwidth of a subchannel,
// b) modulation ef ciency ,
// c) user symbol rate,
// d) user data rate if the information bits are
   encoded with a rate of 3/4,
// e) time utilization ef ciency of the system.

clc;
clear all;

InfoSc=48;//Information subcarriers
SyncSc=4;//synchronization subcarriers
ReservedSc=12;//Reserved subcarriers
Symrate=250;  //ksps(kilosymbols per second)
BW=20;  ///in MHz
Grdt=800; //Guard time in nsec

//solution
TotalSc=InfoSc+SyncSc+ReservedSc;//Total subcarriers
BW_Sch=BW*10^6/TotalSc;//BW of subchannel
Mod_eff=Symrate*10^3/(BW_Sch);//Modulation
   efficiency
```

```
User_txrate=InfoSc*Symrate*10^3;
User_bitsymbol=4;  //16−QPSK is used
disp("From table 21.7 For modulation scheme as 16−QAM and coding rate =3/4 then User data rate will be 36Mbps");
User_DR=36; //Mbps
Sym_Dur=1/(Symrate*10^3);
TimeUti=Sym_Dur/(Sym_Dur+(Grdt/10^9));

printf(' The bandwidth of subchannel is %.1f kHz\n',BW_Sch/10^3);
printf(' Modulation efficiency is %.1f symbols/sec/Hz \n',Mod_eff);
printf(' User symbol rate is %d Msps \n',User_txrate/10^6);
printf(' Time Utilization efficiency is %.2f \n',TimeUti);
```

Scilab code Exa 21.4 To find coverage of the AP

```
//Exa 21.4
// To determine the coverage of AP.

clc;
clear all;

Eb_No=10;  //in dB
Noise=-120; //in dBm
Pt=20; //in mwatt
R=1; //Data rate in Mbps
CHBW=0.5; //BW in MHz
A=37.7;  //path loss at the  rst meter in dB
Y=3.3; //path loss exponent
Lf=19; //function relating power loss with number of oors n      (in dB)
```

```
Ls=10;  // lognormally distributed random variable
    representing the shadow effect in dB

//solution
S2Nreqd=Eb_No*R/CHBW;
Rx_sensi=Noise+S2Nreqd;
Lp=10*log10(20)-Rx_sensi;
//Lp=A+10Ylod(d)+Lf+Ls;therefore
d=10^((Lp-A-Lf-Ls)/(10*Y));
printf('The coverage of AP is %.1f metres \n',d);
```

Scilab code Exa 21.5 To calculate coded symbol and bit transmission rate per subscriber of the two modes

```
// Exa 21.5
// To determine the coded symbol transmission rate
    per subcarrier and bit transmission rate per
    subcarrier for each of the two modes.

clc;
clear all;

R=3/4;//code rate of convolution encoder
M1=9; //payload transmission rate in Mbps for mode 1
M2=36; //payload transmission rate in Mbps for mode
    2

//solution
D1=M1*10^6/48;//user data  rate in kbps for mode 1
D2=M2*10^6/48;//user data rate in kbps for mode 2
//Refering to Table 21.11
printf('Data transmission rate per carrier with 3/4
    convolution encoder are %.1f Kbps and %d Kbps \n'
    ,D1/10^3,D2/10^3);
C1=D1/R;
```

```
C2=D2/R;
printf(' Carrier transmission rate with R=3/4
    convolutional encoder are %d Kbps and %d Kbps\n',
    C1/10^3,C2/10^3);
printf(' Carrier symbol rate with R=3/4
    convolutional encoder are %d ksps and %d Ksps \n'
    ,C1/10^3,C2/4/10^3);  //Mode1 as BPSK and MOde2
    as 16-QAM
```

Scilab code Exa 21.6 To find user data rate for HIPERLAN 2

```
// Exa 21.6
// To determine the user data rate for HIPERLAN/2.

clc;
clear all;

R=3/4; //code rate for convolution encoder

//solution
//64-QAM modulation is used
Sc=250; //Carrier symbol rate(ksps) from Exa 21.5
Bits_sym=log2(64); //64-QAM is used
User_R=Bits_sym*Sc*10^3*R*48;
printf('The user data rate is %d Mbps \n',User_R
    /10^6);
```

Scilab code Exa 21.7 To determine the collision probability of a FH packet

```
// Exa 21.7
// To determine the PER(Packet error rate) for FH(
    Frequency Hopping packet) and DS(Direct spread
    packet).
```

```

clc;
clear all;

D=1000*8;  //packet size in bits
R=2*10^6;  //transmission rate in bps
L=3; //msec(Dwell time)
H=0.625; //msec(Duration of BT packet)

//solution
Tw=10^3*D/R;   //the packet duration of IEEE 802.11 in msec
H_L=1;
G=(H_L)*L-Tw-H;
Gm=abs(G);
PER_FH=1-((1-Gm/L)*(78/79)^(H_L)+Gm/L*(78/79)^((H_L)-G/Gm));
PER_DS=1-((1-Gm/L)*(57/79)^(H_L)+Gm/L*(57/79)^((H_L)-G/Gm));
printf('The PER for FH packet and PER for DS packet are %d percent & %.2f percent respectively',round(PER_FH*100),PER_DS*100);
disp("The collision probability with 802.11 DS is much higher than with 802.11 FH.")
```

Scilab code Exa 21.8 To calculate min SIR and using this SIR and other data calculate Rmax

```
// Exa 21.8
// To determine SIRmin and r_max.

clc
clear all;

d=10; // distance between AP and IEEE 802.11 device
```

```
    in metres
Y=4; //path loss exponent
PBt=20;  //the transmitted power by the BT in dBm
PAp=40;  //the transmitted power by the AP in dBm
Pe=10^-5;//acceptable error probability

//solution
//Pe=0.5*e^(-0.5*Eb/No)
 SIR=log(Pe/0.5)/(-0.5);// signal-to-interference ratio
rmax=d*(SIR*PBt/PAp)^(1/Y);// range of interference between Bluetooth and 802.11 device
printf('Minimum SIR is %.2f dB = %.1f \n',10*log10(SIR),SIR);
printf(' Maximum coverage range is  %.2f metres \n',rmax);
```

Scilab code Exa 21.9 To calculate Rmax for given interference scenarios

```
// Exa 21.9
// To calculate rmax for the interference scenarios (see Figure 21.21) using Smin from Example 21.8.

clc;
clear all;

SIRmin=21.6; //From eg 21.8 i.e(13.36 dB)
d=10; //distance between AP and IEEE 802.11 device in m
PMs=40; // transmitted power of the IEEE 802.11 device in dBm
PBt=20; //the transmitted power by the BT in dBm
Y=4 ; //path loss exponent

//solution
```

```
rmax=d*(SIRmin*PMs/PBt)^(1/Y);
printf('Maximum coverage range is %.1f metres \n',rmax);
```

Scilab code Exa 21.10 To calculate Rmax for IEEE 802 11 FH and DS devices

```
// Exa 21.10
// Repeat Problems 21.8 and 21.9, if the IEEE 802.11 FH device is replaced by the IEEE 802.11 DS device (Gp=11).

clc
clear all;

Gp=11;//processing gain(given)
//Defining variables from Exa 21.8 & 21.9
PBt=20; //  transmitted power by the BT in dBm
PMs=40;  // transmitted power of the IEEE 802.11 device in dBm
PAp=40;  // transmitted power by the AP in dBm
d=10;  // distance between AP and IEEE 802.11 device in m
Y=4;  //path loss exponent
Pe=10^-5;//Error probability

//solution
//Pe=0.5*e^(-0.5*Eb/No)
SIR=log(Pe/0.5)/(-0.5);
r1max=d*(SIR*PBt/(PAp*Gp))^(1/Y);// range of interference between Bluetooth and 802.11 device
printf(' Maximum coverage range for IEEE 802.11 DS is %.2f metres \n',r1max);
r2max=d*(SIR*PMs/(PBt*Gp))^(1/Y);
printf('  Maximum coverage range for IEEE 802.11 FH
```

```
    is %.2f metres \n',r2max);
disp(" Thus, the interference ranges are smaller for
    the IEEE 802.11 DS device compared to the IEEE
    802.11 FH device.")
```

Chapter 24

Spreading Codes used in CDMA

Scilab code Exa 24.1 To generate m sequence and demonstrate its properties

```
// Exa D.1
// Using the shift register shown in Figure D.3, generate an m-sequence and demonstrate its properties.

clc;
clear all;

//solution
//Referring Fig D.3
x=[0 0 1]; //Initial stage
output=x(3);
disp(" First m-sequence using 3-stage shift register.");
disp("            x1 x2 x3          output");
printf(' Initial    %d  %d  %d          %d \n ',x(1),x(2),x(3),output);
for i= 1:7
```

```
    printf('Shift %d',i);
    x(3)=x(2);
    if(x(3)==1) //TO get values in range of [-1 1] for plot
    dummy(i)=-1
else
    dummy(i)=1;
    end
    x(2)=x(1);
   if(output== 1& x(3)==1) //As new x(1)=prev stage x(3) ored prev stage x(2)
       x(1)=0;
   else
       if(output== 0& x(3)==0)
           x(1)=0;
       else
           x(1)=1;
       end
   end

    printf('    %d  %d  %d ',x(1),x(2),x(3));
     output=x(3);
     printf('          %d',output);
      printf('\n ');
end
bar(dummy,0.2,'green');
xlabel("Time","FontSize",5);
title("7-chip first m-sequence for one T period","FontSize",5);
disp("The properties of m-sequence in Figure(0) are -");
disp("Number of -1s = 4 , Number of 1s = 3 ");
disp("Run length 1 = 2 , Run length 2 = 1");
disp("Run length  = 1");
```

Scilab code Exa 24.2 To generate second m sequence

```
// Exa D.2
// what is the location of the modulo-2 adder for the second m-sequence? Generate the second m-sequence.

clc;
clear all;

//solution
disp("The location of modulo-2 adder for the second m-sequence is shown in Figure D.5(in the book)i.e Modulo-2 adder should be between first(x1) and second(x2) shift register.");
x=[0 0 1]; //Initial stage
output=x(3);
disp("Second m-sequence usinf 3-stage register");
disp("              x1  x2  x3          output");
printf(' Initial    %d   %d   %d            %d \n ',x(1),x(2),x(3),output);
for i= 1:7
    printf('Shift %d',i);
    x(3)=x(2);
    if(x(3)==1) //TO get values in range of [-1 1] for plot
    dummy(i)=-1
else
    dummy(i)=1;
    end
    x(2)=x(1);
   if(output== 1& x(2)==1) //As new x(1)=prev stage x(3) ored prev stage x(2)
```

```
        x(1)=0;
    else
        if(output== 0& x(2)==0)
            x(1)=0;
        else
            x(1)=1;
        end
    end

     printf('    %d  %d  %d ',x(1),x(2),x(3));
      output=x(3);
      printf('          %d',output);
       printf('\n ');
end
figure(1);
bar(dummy,0.2,'green');
xlabel("Time","FontSize",5);
title("7-chip second m-sequence for one T period","
   FontSize",5);
disp("The properties of m-sequence in Figure(1)are -
   ");
disp("Number of -1s = 4 , Number of 1s = 3 ");
disp("Run length 1 = 2 , Run length 2 = 1");
disp("Run length  = 1");
```