Scilab Textbook Companion for Satellite Communications by D. Roddy[1]

Created by
Dolar Khachariya
M tech
Electrical Engineering
IIT Bombay
College Teacher
Na
Cross-Checked by
K. V. P. Pradeep

June 1, 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: Satellite Communications

Author: D. Roddy

Publisher: McGraw-Hill

Edition: 3

Year: 2001

ISBN: 0070533709

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

Orbits and Lauching Methods

Scilab code Exa 2.1 Calculate the radius of a circular orbit for which the period is 1 day

```

//Variable Declaration
u=3.986*(10**14)          //Earth's Gravitational constant(m^3/sec^2)

//Calculation
n=(2*3.14)/(24*60*60)     //Mean Motion(rad/sec)
a=((u/n**2)**(0.33333))/1000       //Radius of the orbit by kepler's 3rd law(km)

//Result
printf("The Radius of the circular orbit with 1 day period is : %d km",a)
```

Scilab code Exa 2.2 Calculate the semimajor axis for the satellite parameters given

```

//Variable Declaration
NN=14.22296917        //Mean Motion (1/day)
u=3.986005*(10**14)   //Earth's Gravitational COnstant(m^3/sec^2)

//Calculation
n0=(NN*2*3.142)/(24*60*60)                  //Mean Motion(rad/sec)
a=((u/n0**2)**(0.33333))/1000   //Radius of the orbit by kepler's 3rd law(km)


//Result
printf("The Semimajor axis for given satellite parameters is : %.2f km",a)
```

Scilab code Exa 2.3 Calculate the apogee and perigee for the orbital parameters given

```

//Variable Declaration

R=6371    //Mean Earth's radius(km)
e=0.0011501  //Eccentricity
a=7192.3     //Semimajor axis(km)

//Calculation

ra=a*(1+e)   //Radius Vector at apogee(km)
rp=a*(1-e)   //Radius Vector at perigee(km)
ha=ra-R      //Apogee height(km)
hp=rp-R      //Perigee height(km)


```

```
//Result
printf("The Apogee height for given orbital parameters is: %.2f km",ha)
printf("The Apogee height for given orbital parameters is: %.2f km",hp)
```

Scilab code Exa 2.4 calculate the semimajor axis

```

//Variable Declaration
aE=6378.141      //Earth's equitorial radius(km)
e=0.002          //Eccentricity
p=12             //period from perigee to perigee (hours)
K1=66063.1704   //Constant (km^2)
u=3.986005*(10**14)          //Earth's Gravitational constant(m^3/sec^2)


//Calculation
n=(2*%pi)/(12*60*60)              //Mean Motion(rad/sec)
anp=((u/n**2)**(0.3333))/1000        //Radius of the orbit by kepler's 3rd law(km)
k2=(1-e**2)**1.5

function [y]=f(a)
    y=(n-((u/a**3)**0.5)*(1+K1/a**2*k2))
endfunction
a=fsolve(2,f)
a=a/1000   //Converting a into km

//Result

printf("The nonperturbed value of semimajor axis is
```

```
%.2f km",anp)
printf("\nThe perturbed value of semimajor axis is %.2f km",a)
```

Scilab code Exa 2.5 Determine the rate of regression of the nodes and the rate of rotation of the line of apsides for the satellite parameters specified

```

//Variable Declaration
i=98.6328         //Angle(degrees)
e=0.0011501       //eccentricity
n=14.23304826     //Mean Motion(1/day)
a=7192.3          //Semimajor axis(km)
K1=66063.1704     //Known constant(km^2)

//Calculation

n0=(2*180*n)   //Mean Motion (deg/sec)
K=(n0*K1)/((a**2)*((1-e**2)**2))   //Constant (deg/day)
w=-K*cos(i*3.142/180)     //Rate of regression of nodes(deg/day)
W=K*(2-2.5*(sin(i*3.142/180))**2)    //Rate of rotation of line of apsides(deg/day)

//Results
printf("The rate of regression of nodes is: %.3f deg/day",w)
printf("\nThe rate of rotation of line of apsides is : %.3f deg/day",W)
```

Scilab code Exa 2.6 Calculate the new values for W and w one period after epoch

```

//Variable Declaration
w=0.982  //rate of regression of nodes from Example 2.5(deg/day)
W=-2.903 //rate of rotation of line of apsides from Example 2.5)deg/day)
n=14.23304826  //Mean Motion(1/day)
W0=113.5534   //Argument of perigee(deg)
w0=251.5324   //Right ascension of the ascending node(deg)

//Calculation
PA=1/n   //Period
w=w0+w*PA  //New value of rate of regression of nodes(deg)
W=W0+W*PA  //New Value of rate of rotation of line of apsides(deg)

//Result
printf("New value of rate of regression of nodes is: %.3f deg",w)
printf("\nNew value of rate of rotation of line of apsides is : %.3f deg",W)
```

Scilab code Exa 2.7 Calculate the average length of the civil year in the Gregorian calendar

```

//Calculation
ndays=400*365    //Nominal number of days in 400 years
nleapyrs=400/4   //Nominal number of leap years
gregoriandays=ndays+nleapyrs-3  //number of days in 400 years of Gregorian calendar
gregavg=gregoriandays/400    //number of days in 400
```

```
    years of Gregorian calendar

//Result
disp (gregoriandays)
printf("The average length of the civil year in gregorian calender is : %.4f days",gregavg)
```

Scilab code Exa 2.8 Determine which of the following are leap years

```

//Calculation and Results


if ( modulo(1987,4) == 0. ) then
   disp("1987 is a leap year");
else
   disp("1987 is not a leap year");
end


if ( modulo(1988,4) == 0. ) then
   disp("1988 is a leap year");
else
   disp("1988 is not a leap year");
end

if ( modulo(2000,400) == 0. ) then
   disp("2000 is a leap year");
else
   disp("2000 is not a leap year");
end

if ( modulo(2100,400) == 0. ) then
   disp("2100 is a leap year");
else
```

```
    disp(”2100 is not a leap year”);
end
```

Scilab code Exa 2.9 Calculate the time

```


//Calculation
days=324    //Number of days
hours=floor(24*0.95616765)   // Number of hours
decimalfraction1=24*0.95616765-hours
minutes=floor(60*decimalfraction1)   // Number of minutes
decimalfraction2=60*decimalfraction1-minutes
seconds=60*decimalfraction2   // Number of seconds

//Result

disp(decimalfraction1)
disp(decimalfraction2)
printf(”An Epoch day has %.2f days %.2f hours %.2f minutes %.2f seconds”,days,hours,minutes,seconds)
```

Scilab code Exa 2.10 Find the Julian day for 13h UT on 18December 2000

```


//Variable Declaration

y=2000      //year
mon=12      //month
dy=18       //day
hours=13    //hours of the day
```

```
minutes=0    //Minutes of the day
seconds=0    //Seconds of the day


//Calculation
d=dy+(hours/24)+(minutes/(24*60))+seconds   //Days in December
if mon<=2 then
   y=y-1
   mon=mon+12
else
   y=y
   mon=mon
end

A=floor(y/100)       //Converting years to days
B=2-A+floor(A/4)     //Converting years to days
C=floor(365.25*y)      //rounding the days
D=floor(30.6001*(mon+1)) //Converting months to days
JD=B+C+D+d+1720994.5      //Adding reeference to number of days


//Result

printf(”The Julian day of given day is : %.4f Days”, JD)
```

Scilab code Exa 2.11 Find the time in Julian centuries from the reference time

```

//Variable Declaration

JDref=2415020    //Reference Julian days
```

```
JC=36525
JD=2451897.0417 //Julian days with reference from
   Example 2.10

//Calculation

T=(JD-JDref)/JC  //Time in julian Centuries

//Result

printf("The time for given date is : %.8f Julian
   Centuries",T)
```

Scilab code Exa 2.12 Calculate the time of perigee passage for the NASA elements

```

//Variable Declaration

n=14.23304826  //Mean Motion (rev/day)
M0=246.6853    //Mean Anomaly (degrees)
t0=223.79688452 //Time of anomaly

//Calculation

T = (t0-(M0/(n*360)))  //Time of perigee passage

//Result
printf("The time of perigee passage for NASA
   elements is : %.4f days",T)
```

Scilab code Exa 2.13 calculate the eccentric anomaly

```

//Variable Declaration
M=205  //Mean anomaly(degrees)
e=0.0025  //Eccentricity
E=%pi //Initial guess value for eccentric anomaly

//Calculation

function [y] = f(E)
    y=M-E+e*sin(E)
endfunction
E=fsolve(3.142,f)

printf("The Eccentric anomaly is: %.4f degrees",E)
```

Scilab code Exa 2.14 Calculate teh true anomaly and the magnitude of the radius vector

```

//Variable Declaration
pi = %pi
n=14.2171401*2*%pi/86400  //Mean motion (rad/sec)
M=204.9779+0.001*180*5/pi //Mean anomaly(rad)
e=9.5981*10**-3   //Eccentricity
a=7194.9   //Semimajor axis(km)

//Calculation

v=(M*pi/180)+2*e*sin(M*pi/180)+(5*e**2*sin(2*M*pi)
   /(4*180)) //True Anomaly (radians)
v=v*180/%pi  //True anomaly(degrees)
r=a*(1-e**2)/(1+e*cos(v))  //Magnitude of radius
   vector after 5s(km)

//Results
```

```
printf("The true anomaly is: %.3f degrees",v)
printf("\nThe magnitude of radius vector 5s after epoch is: %d km",r)
```

Scilab code Exa 2.15 express r in vector form in the perifocal coordinate system

```

//Variable Declaration

v=204.81    //True anomaly(degrees) from Example 2.14
r=7257      //Magnitude of radius vector(km) from Example 2.14

//Calculation

rP=r*cos(v*%pi/180)  //P coordinate of radius vector (km)
rQ=r*sin(v*%pi/180)  //Q coordinate of radius vector (km)

//Result
printf("r in the perifocal coordinate system is %.2f Pkm %.2f Qkm",rP,rQ)
```

Scilab code Exa 2.17 Find the GST for 13h UT on 18December 2000

```

//Variable Declaration
pi = %pi
T=1.009638376  //Time in Julian centuries from Example 2.11
UT=13  //Universal time(hours)
```

```

//Calculation

GST=(99.6910+36000.7689*T+0.004*T**2)*3.142/180   //
   GST(radians)
UT=2*pi*UT/24  //Universal time converted to
   fraction of earth rotation (radians)

GST=GST+UT


GST=(modulo(GST,2*pi))*180/pi

//Result
printf("The GST for given date and time is %.2f
   degrees",GST)
```

Scilab code Exa 2.18 Find the LST for Thunder Bay

```

//Variable Declararion
pi = %pi
WL=-89.26  //Expressing the longitude in degrees
   west
GST=282.449  //GST from Example 2.17 (degrees)

//Calculation

EL=2*pi+WL  //Longitude in degrees East
LST=(GST+EL)*pi/180        //LST(radians)
LST=(modulo(LST,2*pi))*180/pi   //fmod removes
   multiple revolutions(Degrees)

//Results
printf("LST for Thunder Bay on given day is: %.2f
```

```
    Degrees”,LST)
```

Scilab code Exa 2.19 Find the components of the radius vector to the earth station at Thunder Bay

```

//Variable Declaration

LST=167.475 //LST(degrees)
LE=48.42    //Latitude at thunder bay(degrees)
H=200       //Height above sea level(metres)
aE=6378.1414 //Semimajor axis(km)
eE=0.08182   //Eccentricity

//Calculation

l=(aE/sqrt(1-eE**2*sin(LE*3.142/180)**2)+H/1000)*cos
    (LE*3.142/180)
z=((aE*(1-eE**2))/sqrt(1-eE**2*sin(LE*3.142/180)**2)
    +H/1000)*sin(LE*3.142/180)
RI=l*cos(LST*3.142/180) //I component of radius
    vector at thunder bay(km)
RJ=l*sin(LST*3.142/180) //J component of radius
    vector at thunder bay(km)
RK= z                   //Z component of radius vector
     at thunder bay(km)

R=sqrt(RI**2+RJ**2+RK**2)


//Results
printf(”The Radius vector components are %.2f ikm+ %
    .2f jkm+ %.2f kkm”,RI,RJ,RK)
printf(”\nThe Magnitude of radius component is %.2f
    km”,R)
```

Scilab code Exa 2.20 Calculate the corresponding range and the look angles for an earth station the coordinates

```

//Variable Declaration

PI=-1280   //I component of range vector for a
    satellite(km)
PJ=-1278   //J component of range vector for a
    satellite(km)
PK=66      //K component of range vector for a
    satellite(km)
GST=240    //GST(degrees)
LE=48.42   //Latitude(Degrees)
PE=-89.26  //Longitude(Degrees)
H=200      //Height above mean sea level(metres)
aE=6378.1414 //Semimajor axis(km)
eE=0.08182   //Eccentricity


//Calculation

l=(aE/sqrt(1-eE**2*sin(LE*3.142/180)**2)+H/1000)*cos
    (LE*3.142/180)
z=((aE*(1-eE**2))/sqrt(1-eE**2*sin(LE*3.142/180)**2)
    +H/1000)*sin(LE*3.142/180)
SE=(atan(z/l))*180/3.142 //Geocentric latitude angle
     (degrees)
LST=240+PE


a=sin(SE*3.142/180)*cos(LST*3.142/180)
b=sin(SE*3.142/180)*sin(LST*3.142/180)
c=-cos(SE*3.142/180)
```

```
d=-sin(LST*3.142/180)
e=cos(LST*3.142/180)
f=0
g=cos(SE*3.142/180)*cos(LST*3.142/180)
h=cos(SE*3.142/180)*sin(LST*3.142/180)
i=sin(SE*3.142/180)

D = [a,b,c;d,e,f;g,h,i]

P=  [PI;PJ;PK]

R=D*P  //Components of range of earth station
Ro=sqrt(R(1,1)**2+R(2,1)**2+R(3,1)**2) //Magnitude of range of earth station(km)
El=asin(R(3,1)/Ro) //Antenna elevation angle for the earth station(radians)
El= El*180/3.142   //Converting El to degrees
alpha=(atan(R(2,1)/R(3,1)))*180/3.142

if ( R(1,1)<0 & R(2,1)>0 ) then
   Aza=alpha
else
   Aza=0
end
if ( R(1,1)>0 & R(2,1)>0 ) then
   Azb=180-alpha
else:
   Azb=0
end

if ( R(1,1)>0 & R(2,1)<0 ) then
   Azc=180+alpha
else
   Azc=0
end
if ( R(1,1)<0 & R(2,1)<0 ) then
   Azd=360-alpha
else
```

```
    Azd=0
end
Az= Aza+Azb+Azc+Azd   //Azimuth angle (degrees)

printf("The magnitude of range of earth station is %.0f km",Ro)
printf("\nThe antenna elevation angle for the earth station are %.f degrees",El)
printf("\nThe Azimuth angle for the earth station is %.2f degrees",Az)
```

Scilab code Exa 2.21 Determine the subsatellite height latitude and LST

```

//Variable Declaration

rI=-4685.3  //I component of radius vector from Example 2.16(km)
rJ=5047.7   //J component of radius vector from Example 2.16(km)
rK=-3289.1  //K component of radius vector from Example 2.16(km)
aE=6378.1414 //Semimajor axis (km)
eE=0.08182   //Eccentricity

//Calculation

r=sqrt(rI**2+rJ**2+rK**2)
a=%pi  //Guess value for LST(radians)
b=atan(rK/rI)   //Guess Value for latitude(radians)
c=r-aE   //Guess value for height(km)

function [ans] = equations(p)
    L = p(1)
    h = p(2)
```

```
    LST = p(3)
    a = rI-((aE/sqrt(1-eE**2*sin(L)**2))+h)*cos(L)*cos(LST)
    b = rJ-((aE/sqrt(1-eE**2*sin(L)**2))+h)*cos(L)*sin(LST)
    c = rK-((aE*(1-eE**2)/sqrt(1-eE**2*sin(L)**2))+h)*sin(L)
    ans = [a;b;c]
endfunction

ans =  fsolve([b;c;a],equations)
L = ans(1)
h = ans(2)
LST = ans(3)
L= L*180/3.142  //Converting L into degrees
h=round(h)
LST=LST*180/3.142 //Converting LST into degrees

printf("The latitude of subsatellite is %.2f degrees",L)
printf("\nThe height of subsatellite is %.2f km",h)
printf("\nThe LST of subsatellite is %.1f degrees",LST)
```

Chapter 3

The Geostationary orbit

Scilab code Exa 3.1 Calculate the azimuth angle for an earth station antenna

```

//Variable Declaration

Pss=-90   //Location of geostationary satellite(
    degrees)
PE=-100   //Longitude of the earth station antenna(
    degrees)
LE=35     //Latitude of the earth station antenna(
    degrees)

//Calculation

B=PE-Pss    //Angle between planes containing a and c
    (degrees)
b=acos(cos(B)*cos(LE)) //Angle of plane containing b
    (radians)
A=asin(sin(abs(B*3.142/180))/sin(b)) //Angle between
     planes containing b and c (radians)

A=A*180/3.142  //Converting A into degrees
```

```
//LE>0 and B<0 by observation
Az= 180-A     //Azimuth angle(degrees)

//Result

printf("The azimuth angle for the given earth
   station antenna is %.2f degrees",Az)
```

Scilab code Exa 3.2 Find the range and antenna elevation angle

```

//Variable Declaration

R=6371  //Radius of earth (km)
aGSO= 42164  //Circumference of earth(km)
b=0.632 //values of b from Example 3.1 (radians)
//Calculation

d=sqrt(R**2+aGSO**2-2*R*aGSO*cos(b)) //Range of
   earth station antenna (km)
El=acos(aGSO*sin(b)/d)*180/%pi  //Elevation angle(
   degrees)

//Results

printf("The range of earth station antenna is %.0f
   km",d)
printf("Elevation angle is %.0f degrees",El)
```

Scilab code Exa 3.3 Determine the angle

```

//Variable Declaration
```

```

LE=49  //Latitude of earth station(degrees)
aGSO=42164  //Circumference of earth(km)
R=6371  //Radius of earth(km)

//Calculation
d=(R**2+aGSO**2-2*R*aGSO*cos(LE*3.142/180))**0.5  //Range of earth station antenna
El0=acos(aGSO*sin(LE*3.142/180)/d)  //Elevation angle(radians)
El0=El0*180/3.142  //Converting El0 to degrees
delta=round(90-El0-LE)  //Angle of tilt required for polar mount

//Results
printf("The Angle of tilt required for polar mount is %d degrees",delta)
```

Scilab code Exa 3.4 Determine the limits of visibility

```

//Variable Declaration

LE=48.42  //Latitude of earth station(degrees)
PE=-89.26 //Longitute of earth station(degrees)
Elmin=5  //Minimum angle of elevation(degrees)
aGSO=42164  //Circumference of earth(km)
R=6371  //Radius of earth(km)

//Calculation

Smin=90+Elmin
S=asin(R*sin(Smin*3.142/180)/aGSO)*180/%pi  //Angle subtended at the satellite(degrees)

```

```
b=180-Smin-S   //Angle of plane containing b(degrees)
B=acos(cos(b*3.142/180)/cos(LE*3.142/180))*180/%pi//Angle between the planes containing a and c(degrees)

//Results

printf("The satellite limit east of the earth station is at %d Degrees approximately",round(PE+B))

printf("The satellite limit west of the earth station is at %d Degrees approximately",round(PE-B))
```

Scilab code Exa 3.5 calculate the longitude

```

//Variable Declaration
y=2000         //year
d=223.153      //day
n=1.002716     //mean motion(1/day)
w=272.5299     //rate of regression of nodes(degrees)
e=0.000352     //Eccentricity
W=247.9161     //Rate of regression of line of apsides(degrees)
M=158.0516     //Mean Anomaly at given time
JD00=2451543.5   //Julian days for Jan 0.0 2000

//Calculation

JD=JD00+d   //Julian days for given day
JDref=2415020    //Reference Julian days
JC=36525
```

```
T=(JD-JDref)/JC  //Time in julian Centuries
UT=d-223  //Universal Time, fraction of the day
GST=(99.6910+36000.7689*T+0.004*T**2)*3.142/180  // GST(radians)
UT=2*%pi*UT //Universal time converted to fraction of earth rotation (radians)

GST=(GST+UT)*180/3.1421
GST=(modulo(GST,360))//using fmod multiplr revolutions are removed (degrees)

v=M+2*e*M  //True Anomaly(degrees)

Pssmean=W+w+M-GST //longitude for INTELSAT(degrees)
Pssmean=modulo(Pssmean,360) //fmod removes multiple revolutions
Pss=w+W+v-GST//longitude for INTELSAT(degrees)
Pss=modulo(Pss,360)//fmod removes multiple revolutions

//Results
printf("The longitude of INTELSAT 805 is %.3f Degrees",Pss)

printf("The average longitude of INTELSAT 805 is %.3f Degrees",Pssmean)

// Note : Answers may be different because of rounding error. Please check by calculating all variables.
```

Chapter 4

Radio Wave Propagation

Scilab code Exa 4.1 Calculate for the frequency

```

//Variable Declaration

El=50  //Elevation Angle(degrees)
h0=0.6 //Earth station altitude(km)
hr=3   //Rain height(km)
R01=10 //Point Rain Rate(mm/hr)
f=12   //frequency(GHz)
ah=0.0188
bh=1.217
av=0.0168
bv=1.2

//Calculation
Ls=(hr-h0)/sin(El*3.142/180) //Slant path length(km)
LG=Ls*cos(El*3.142/180)      //Horizontal projection
   (km)
r01=90/(90+4*LG)       //Reduction factor
L=Ls*r01               //Effective path length(km)
alphah=ah*R01**bh      //Specific Attenuation
AdBh=alphah*L          //Rain Attenuation for
```

```
   horizontal polarization
alphav=av*R01**bv       //Specific Attenuation
AdBv= alphav*L          //Rain Attenuation for
   vertical polarization

//Results
printf(”Rain Attenuation for given conditions and
   horizontal polarization is %.2f dB”,AdBh)

printf(”Rain Attenuation for given conditions and
   vertical polarization is %.2f dB”,AdBv)
```

Scilab code Exa 4.2 circular polarization

```

//Variable Declaration
ah=0.0188
bh=1.217
av=0.0168
bv=1.2
R01=10          //Point Rain Rate(mm/hr)
L=2.8753812   //Effective path length calculated in
   Example 4.1(km)

//Calculation
//Factors depending on frequency and polarization
ac=(ah+av)/2 //a for circular polarization
bc=(ah*bh+av*bv)/(2*ac) //b for circular
   polarization
alpha=ac*R01**bc   //Specific Attenuation(dB)
AdB= alpha*L      //Rain Attenuation(dB)


//Results
printf(”The Rain Attenuation for circular
```

```
    polarization is %.2f dB”,AdB);
```

Chapter 5

Polarization

Scilab code Exa 5.1 Determine the angle of polarization

```

//Variable Declararion

L=18    //Latitude of earth station(degrees)
PE=-73  //Longitude of earth station(degrees)
Pss=-105 //Satellite location(degrees)
aGSO=42164 //Circumference of earth (km)
R=6371 //Radius of earth(km)


//Calculation

function [ans] = mycross(A,B)
    i = A(2)*B(3) - B(2)*A(3)
    j = A(1)*B(3) - B(1)*A(3)
    k = A(1)*B(2) - B(1)*A(2)
    ans = [i,j,k]
endfunction

B=PE-Pss  //Angle between the planes containing a and c (degrees)
```

```
Rx=R*cos(L*3.142/180)*cos(B*3.142/180) //Geocentric-equitorial coordinate(km)
Ry=R*cos(L*3.142/180)*sin(B*3.142/180) //Geocentric-equitorial coordinate(km)
Rz=R*sin(L*3.142/180) //Geocentric-equitorial coordinate(km)

r= [Rx,Ry,Rz]         //Coordinates for local gravity direction
k=[Rx-aGSO,Ry,Rz]    //geocentric-equitorial coordinates for propagation direction
e=[0,0,1]             //geocentric-equitorial coordinates for polarization vector

f=mycross(k,r) //Direction of normal to reference plane
modf = (f(1)**2+f(2)**2+f(3)**2)**0.5
g = mycross(k,e)// Direction of normal to plane contaning e and k
h=mycross(g,k) //Direction of polarization of the plane
modh=(h(1)**2+h(2)**2+h(3)**2)**0.5
p=(h/modh)

E = asin((p.*f)/modf)*180/3.142

printf("The Angle of polarization at given location is %.2f degrees",E(3))



// Note : cross() function did not work, so i have wrote mycross() function. Answers would be differ because of rounding error.
```

Chapter 6

Antennas

Scilab code Exa 6.1 Plot the E Plane and H Plane radiation

```
//Variable Decalration
a=3
b=2
dB=1

//Calculation
//Initializations
tita= -90:2:91
tita(46) = 1
tita1= -90:2:91
Y=linspace(0,0,91)
E=linspace(0,0,91)
gE=linspace(0,0,91)
GE=linspace(0,0,91)
X=linspace(0,0,91)
E1=linspace(0,0,91)
gH=linspace(0,0,91)
GH=linspace(0,0,91)

for i = 1:length(Y)-1
   Y(i)=%pi*b*sin(tita(i)*3.142/180)
```

```
   X(i)=%pi*a*sin(tita(i)*3.142/180)
   E(i)=(sin(Y(i)))/Y(i)
   E1(i)=cos(tita1(i)*3.142/180)*(sin(X(i)))/X(i)
   gE(i)=(E(i))**2   //Raiation pattern in E-Plane
   gH(i)=E1(i)**2    //Raiation pattern in H-Plane
   GE(i)=10*log10(gE(i))  //Raiation pattern in E-Plane(dB)
   GH(i)=10*log10(gH(i))  //Raiation pattern in H-Plane(dB)
end;

//Results

subplot(211)
plot(tita,GE)   //Plotting E-Plane radiation pattern
xtitle('','tita degrees',"GE(tita)" )
subplot(212)
plot(tita1,GH)  //Plotting H-Plane radiation pattern
xtitle('','tita degrees','GH(tita)')
```

Scilab code Exa 6.2 Plot the magnitue of the array factor as a function of phi

```

//Varable Declaration

N=5   //Number of elements of dipole
s=0.25 //Space between dipole elements(wavelengths)
phi0=0//Angle between array factor and array(degrees)

//Calculation

alpha=-2*3.142*s*cos(phi0)  //Current phase(radians)
phi= -180:2:182
```

```
for k = 1:180
    Si(k)=alpha+2*3.142*s*cos(phi(k)*3.142/180)
end;
AFR=linspace(0,0,181)
AFI=linspace(0,0,181)
for i = 1:180
  for j = 1:N-1
     AFR(i)=AFR(i)+cos(j*Si(i))  //Real part of Array factor
     AFI(i)=AFI(i)+sin(j*Si(i))//Imaginary part of Array factor
end
end

teta= linspace(-3.142,3.142,181)
for k = 1:length(teta)
   AF(k)=(AFR(k)**2+AFI(k)**2)**0.5
end
//Result
polarplot(teta,AF)
```

Scilab code Exa 6.3 for phi 90 degrees

```
//Varable Declaration

N=5   //Number of elements of dipole
s=0.25 //Space between dipole elements(wavelengths)
phi0=90*3.142/180 //Angle between array factor and array(radians)

//Calculation

alpha=-2*3.142*s*cos(phi0)  //Current phase(radians)
phi= -180:2:182
for k = 1:180
```

```
    Si(k)=alpha+2*3.142*s*cos(phi(k)*3.142/180)
end
AFR = linspace(0,0,180)
AFI = linspace(0,0,180)
for i = 1:180
  for j = 1:N-1
     AFR(i)=AFR(i)+cos(j*Si(i))  //Real part of Array factor
     AFI(i)=AFI(i)+sin(j*Si(i))//Imaginary part of Array factor
end
end;

teta=linspace(-3.142,3.142,180)
AF = linspace(0,0,180)
for k  = 1:180
   AF(k)=AF(k)+(AFR(k)**2+AFI(k)**2)**0.5
end
//Result

polarplot(teta,AF)
```

Chapter 9

Analog Siganls

Scilab code Exa 9.1 Calculate the peak deviation and the signal bandwidth

```

//Variable Declaration

Bs=4.2  //Signal Bandwidth(MHz)
delf=2.56  //Deviation Ratio

//Calculation
delF=Bs*delf  //Peak Deviation(MHz)
BIF=2*(delF+Bs)  //Signal Bandwidth(MHz)
BIF=BIF
//Results

printf("The peak deviation is: %.3f MHz", delF)
printf("Signal Bandwidth is : %.1f MHz",BIF)
```

Scilab code Exa 9.2 Calculate the modulation index and the Bandwidth

```
//Variable Declaration

delF=200 //Peak Deviation(kHz)
f=0.8    //Test tone frequency (kHz)

//Calculation
m=delF/f  //Modualtion index
B=2*(delF+f)  //Bandwidth of the signal(kHz)

//Results
printf("The modulation index is %.f" , m)
printf("Bandwidth of the signal is %.1f kHz",B)
```

Scilab code Exa 9.3 Recalculate the bandwidths

```

//Variable Declaration

Bs1=4.2  //Signal Bandwidth(MHz) of Example 9.1
delf=2.56  //Deviation Ratio of Example 9.1

delF2=200 //Peak Deviation(kHz) of Example 9.2
Bs2=0.8    //Test tone frequency (kHz) of Example
   9.2

//Calculation
delF1=Bs1*delf  //Peak Deviation(MHz) of Example 9.1
BIF1=2*(delF1+2*Bs1)  //Signal Bandwidth(MHz) of
   Example 9.1 according to Carson's rule
BIF2=2*(delF2+2*Bs2)  //Signal Bandwidth(kHz) of
   Example 9.2 according to Carson's rule.

//Results

printf("Signal Bandwidth of Example 9.1 by Carsons
```

```
rule is %.1f MHz",BIF1)
printf("\nSignal Bandwidth of Example 9.2 by Carsons rule is %.1f kHz",BIF2)
```

Scilab code Exa 9.4 calculate the receiver processing gain and the post-detector

```

//Variable Declaration

delf=5 //Deviation frequency (kHz)
Bs=1   //Test Tone Frequency (kHz)
CNR=30  //Carrier to noise ration(dB)

//Calculation
m=delf/Bs  //Modulation Index
Gp=3*(m**2)*(m+1)  //Processing gain for sinusoidal modulation
Gp=10*log10(Gp) //Converting Gp into dB
SNR=CNR+Gp

//Results
printf("The receiver processing gain is %.1f dB",Gp)
printf("\nThe Signal to noise ratio is %.1f dB",SNR)
```

Scilab code Exa 9.5 Calculate the signal to noise ratio

```

//Variable Declaration

n=24  //Number of channels
g=13.57  //Peak/rms factor(dB)
b=3.1  //Channel Bandwidth(kHz)
```

```
P=4       //Emphasis improvement (dB)
W=2.5     //Noise weighting improvement(dB)
CNR=25    //Carrier to noise ratio (dB)
delFrms=35  //rms value of Peak Deviation(kHz)
fm=108     //Baseband frequency (kHz)
//Calculation

L=10**((-1+4*log10(n))/20)
g=10**(g/20)  //Converting process gain to ratio
delF=g*delFrms*L  //Peak Deviation(Hz)
BIF=2*(delF+fm)   //Signal Bandwidth(kHz) by Carson'
   s rule
Gp=(BIF/b)*((delFrms/fm)**2)  //Processing Gain
Gp=10*log10(Gp)  //Converting Gp to dB
SNR=CNR+Gp+P+W  //Signal to noise ratio for top
   channel in 24-channel FDM basseband signal

//Results
printf("Signal to noise ratio for top channel in 24-
   channel FDM Baseband signal is %.1f db", SNR)
```

Scilab code Exa 9.6 Calculate the carrier to noise ratio required at the input to the FM detector

```

//Variable Declaration

delF=9   //Peak Deviation (MHz)
fm=4.2   //Baseband frequency(MHz)
SNR=62   //Signal to noise ration(dB)
M=11.8   //Noise weighing(P)+emphasis improvement(W)-
   implementation margin(IMP)

//Calculation

```

```
D=delF/fm  //Modulation Index
GPV=12*(D**2)*(D+1)  //Processing Gain for TV
GPV=10*log10(GPV)  //Converting GPV into dB
CNR=SNR-GPV-M   //carrier to noise ratio(dB)

//Results
printf("The Carrier to noise ratio required at the
   input of FM detector is %.1f dB",CNR)
```

Chapter 10

Digital Siganls

Scilab code Exa 10.1 determine the bit error rate

```

funcprot(0)

//Variable Declaration
PR=0.01   //The Average power received(watts)
Tb=0.0001   //Bit period(seconds)
N0=10**-7   //Noise power(joule)

//Calculations
Eb=PR*Tb   //Energy per bit received (joule)
x=sqrt(Eb/N0)


erf=integrate("exp(-t^2)","t",0,x)
erf1=erf*(2/%pi**0.5)
BER=(1-erf1)*(10**6)/2

printf("The Bit error rate is %.1f * 10^-6", BER)
```

Scilab code Exa 10.2 Calculate the required C N0

```

//Variable Declaration
Rb=61   //Transmission rate (Mb/s)
ENR=9.5 //Required Energy to noise ratio(dB)

//Calculation

Rb=10*log10(61*10**6)  //Converting Transmission rate to dB
CNR=Rb+ENR  //Carrier to noise ratio

//Results
printf("Required Carrier to noise ratio is %.2f dB",CNR)
```

Scilab code Exa 10.3 Calculate the Eb N0 ratio in decibels

```
funcprot(0)
//Variable Declaration
BER=10**-5  //Maximum allowable bit error rate

//Calculation

x=linspace(8,10,11)  //Eb/N0 ratio represented by x
x1=x**0.5
for i = 1:11
   x(i)=10*log10(x(i))  //Converting x into decibels
end

erf=linspace(0,0,11)  //Initialization for erf function
Pe=linspace(0,0,11)   //Initialization for Probablity of error
```

```


for i = 1:10
    k=integrate("exp(-t**2)",'t',0,x1(i))
    erf(i)=k(1)*(2/%pi**0.5)
    Pe(i)=(1-erf(i))/2         //Probability of error
end
y=linspace(9,9.59,5)
z=linspace(BER,BER,5)
a=linspace(9.59,9.59,5)
b=linspace(0,BER,5)
plot(x,Pe)
plot(y,z)
plot(a,b)
xlabel('','xdB','Pe(x)')

x=9.6    //The Eb/N0 ratio for Maximum BER(dB) from the graph
EbN0=x+2 //Eb/N0 ratio with implementation margin
//Results

printf("The Eb/N0 ratio with allowable BER of 10^-5 and implementation margin of 2dB is %.1f dB",EbN0)
```

Chapter 12

The Space Link

Scilab code Exa 12.1 Calculate the EIRP in dBW

```

//Variable Declaration
P=6   //Transmit power(Watts)
G=48.2 //Antenna Gain(dB)

//Calculation
EIRP=10*log10(P)+G  //Equivalent isotropic radiated
   power(dB)

//Result
printf("Hence the Equivalent isotropic radiated
   power is %.0f dBW",EIRP)
```

Scilab code Exa 12.2 Calculate the gain of a 3 m paraboloidal antenna operating

```

//Variable Declaration
```

```

D=3  //Antenna size(m)
f=12 //Operating Frequency(GHz)
n=0.55 //Aperture efficiency

//Calculation

G=n*(10.472*f*D)**2  //Antenna Gain
G=10*log10(G)  //Converting Antenna gain to dB

//Result
printf("The Antenna gain with given parameters is %.1f dB", G)
```

Scilab code Exa 12.3 Calculate the free space loss at a frequency of 6 GHz

```

//Variable Declaration
r=42000       //Range between ground station and satellite
f=6000           //Frequency(MHz)

//Calculation

FSL=32.4+20*log10(r)+20*log10(f)  //Free space loss(dB)

//Result
printf("The free space loss at given frequency is %.1f dB", FSL)
```

Scilab code Exa 12.4 Calculate the total link for clear sky conditions

```

//Variable Declaration
FSL=207    //Free space loss(dB)
RFL=1.5    //receiver feeder loss(dB)
AA=0.5     //Atmospheric Absorption loss(dB)
AML=0.5    //Antenna Alignment loss(dB)

//Calculation

LOSSES=FSL+RFL+AA+AML    //Total link loss (dB)

//Results

printf("The total link loss is %.1f dB", LOSSES)
```

Scilab code Exa 12.5 Calculate the noise power density and the noise power for a bandwidth of 36 MHz

```

//Variable Declaration

TAn=35  // Antenna Noise Temperature(Kelvin)
TRn=100 // Receiver Noise Temperature(Kelvin)
k=1.38*10**-23   //Boltzman constant(joules)
B=36*10**6   //Bandwidth

//Calculation
N0=(TAn+TRn)*k    //noise power density(10**-21
   joules)
PN=N0*B/10**-12         //Noise power for given
   bandwidth(picoWatts)


//Results
printf("The noise Power density is %.2e Joules",N0)
```

```
printf("The noise power for given bandwidth is %.3f pW",PN)
```

Scilab code Exa 12.6 Calculate the overall noise temperature referred to the LNA input

```
//Variable Declaration

TRn=12   //Receiver Noise figure(dB)
G=40     //Gain of LNA(dB)
T0=120  //Noise temperature(Kelvin)

//Calculation

F=10**(TRn/(10))   //Converting noise power to ratio
Te=(F-1)*290     //Noise Temperature of the amplifier
G=10**(G/10)     //Converting Gain of LNA to ratio
Tn=T0+Te/G       //Overall Noise Temperature(Kelvin)


//Result
printf("The overall noise temperature is %.2f Kelvin", Tn)
```

Scilab code Exa 12.7 Calculate the noise temperature to the input

```

//Variable Declaration

Tant=35      //Antenna noise temperature(kelvin)
Te1=150      //Receiver noise temperature(kelvin)
L=5          //Cable Loss (dB)
```

```
T0=290
G1=10**5     //LNA Gain
F=12         //Receiver Noise figure(dB)

//Calculation

L=10**(L/10)  //Converting L into ratio
F=10**(F/10)  //Converting F into ratio
Ts=Tant+Te1+(L-1)*T0/G1+L*(F-1)*T0/G1  //Noise
   Temperature referred to the input(Kelvin)

//Result
printf("The noise temperature referred to the input
   is %.0f Kelvini",Ts)
```

Scilab code Exa 12.8 Repeat Example 7

```

//Variable Declaration

Tant=35      //Antenna noise temperature(kelvin)
Te1=150      //Receiver noise temperature(kelvin)
L=5          //Cable Loss (dB)
T0=290
G1=10**5     //LNA Gain
F=12         //Receiver Noise figure(dB)

//Calculation

L=10**(L/10)  //Converting L into ratio
F=10**(F/10)  //Converting F into ratio
Ts=Tant+(L-1)*T0+L*Te1+L*(F-1)*T0/G1  //Noise
   Temperature referred to the input(Kelvin)


```

```
//Result
printf(”The noise temperature referred to the input is %.0f Kelvin”,Ts)
```

Scilab code Exa 12.9 Calculate the carrier to noise spectral density ratio

```
//Variable Declaration

FSL=206     //Free space loss(dB)
APL=1       //Antenna Pointing loss(dB)
AAL=2       //Atmospheric Absorption loss(dB)
RFL=1       //Receiver feeder loss(dB)
EIRP=48     //Equivalent isotropically radiated power (dBW)
f=12        //Frequency(GHz)
GTR=19.5    //G/T ratio(dB/K)
k=-228.60   //Value of k(dB)

//Calculation
LOSSES=FSL+APL+AAL+RFL //Total loss(dB)
CNR=EIRP+GTR-LOSSES-k  //Carrier to noise ratio(dBHz)

//Result
printf(”The carrier to noise ratio is %.2f dB”,CNR)
```

Scilab code Exa 12.10 Calculate the earth station EIRP required for saturation

```

//Variable Declaration
f=14           //Frequency(GHz)
```

```
Ps=-120      //Flux density required to saturate the transponder(dBW/m2)
LOSSES=2     //Propogation Losses(dB)
FSL=207      //Free-space loss(dB)

//Calculation

A0=-21.45-20*log10(f)    //Effective antenna aperture (dB)
EIRP=Ps+A0+LOSSES+FSL //Equivalent isotropically radiated power(dB)

//Result
printf("The earth station EIRP required for saturation is %.2f dBW",EIRP)
```

Scilab code Exa 12.11 Calculate the carrier to noise density ratio

```

//Variable Declaration

Ps=-91.4       //saturation flux density(dBW/m2)
f=14           //uplink frequency(GHz)
GTR=-6.7       //G/T (dB/k)
BO=11         //Input Back off(dB)
k=-228.6        //Value of k(dB)
RFL=0.6       //receiver feeder loss

//Calculation

A0=-21.5-20*log10(f)    //Effective antenna aperture(dB)
CNR=Ps+A0-BO+GTR-k-RFL     //carrier to noise ratio(dB)

```

```
//Result
disp(A0)
printf("The carrier to noise ratio is %.1f dB",CNR)
```

Scilab code Exa 12.12 calculate the satellite EIRP required

```

//Variable Declaration

B=36      //Transponder Bandwidth(MHz)
CNR=22    //Carrier to noise ratio(dB)
LOSSES=200  //Total transmission losses(dB)
GTR=31      //Earth station G/T (dB/K)
k=-228.6    //Value of k(dB)

//Calculation
B=10*log10(B*10**6)    //Converting Bandwidth to dB
EIRP=CNR-GTR+LOSSES+k+B     //Equivalent isotropically radiated power(dB)

//Result
printf("Satellite EIRP required is %.0f dB",EIRP)
```

Scilab code Exa 12.13 Calculate the bit rate which can be accomodated and the EIRP required

```

//Variable Declaration

B=36*10**6        //Transponder Bandwidth(Hz)
R=0.2      //Roll off factor
GTR=31     //Earth station G/T(dB/K)
LOSSES=200  //Total transmission losses(dB)
```

```
k=-228.6     //Value of k(dB)
BER=10**-5   //Value of Bit error rate
EbN0R=9.6    //Value of Eb/N0 from fig.10.17
//Calculation

Rb=2*B/(1+R)       //Bit rate(sec^-1)
Rb=10*log10(Rb)   //Converting Rb into decibels
CNR=EbN0R+Rb       //Carrier to noise ratio(dB)
EIRP=CNR-GTR+LOSSES+k   //Equivalent Isotropically radiated power(dBW)


//Results
printf("Bit rate that can be accommodated is %.1f dB",Rb)
printf("The EIRP required is %.1f dBW",EIRP)
```

Scilab code Exa 12.14 Calculate the carrier to noise ratio at the earth station

```


//Variable Declaration

EIRP=25     //Satellite saturation value(dBW)
BO=6        //Output Backoff loss(dB)
FSL=196     //Free space loss(dB)
DL=1.5      //Downlink losses(dB)
GTR=41      //Earth station G/T(dB/K)
k=-228.6    //Value of k(dB)

//Calculation
CNR=EIRP-BO+GTR-FSL-DL-k   //Carrier to noise ratio(dB)

```

```
//Result
printf("The Carrier to noise density ratio at the earth station is %.1f dB",CNR)
```

Scilab code Exa 12.15 Calculate the power output of the TWTA required for full saturated EIRP

```

//Variable Declaration

EIRP=56     //Equivalent Isotropically radiated power(dBW)
BO=6        //Output Backoff(dB)
TFL=2       //Transmitter feeder loss(dB)
GT=50       //Antenna gain(dB)

//Calculation
PTWTA=EIRP-GT+TFL    //Power output of TWTA(dBW)
PTWTAS=PTWTA+BO      //Saturated power output of TWTA(dBW)

//Result
printf("Power output of the TWTA required for full saturated EIRP is %.f dBW",PTWTAS)
```

Scilab code Exa 12.16 calculate the value

```


//Variable Declaration

alpha=1.9     //Rain attenuation(dB)
CNR=20        //Downlink carrier to noise ratio(dB)
```

```
Tn=400          //Effective Noise temperature(Kelvin)
Ta=280          //Reference temperature(Kelvin)

//Calculation
alpha1=10**(alpha/10)  //Converting alpha to ratio
Trn=Ta*(1-1/alpha1)    //Equivalent noise temperature of rain(kelvin)
Ts=Tn+Trn          //New system noise temperature
delp=10*log10(Ts/Tn)  //Decibel increase in noise power
CNRN=CNR-delp-alpha    //Value below which CNR falls(dB)


//Result
printf(”The value below which C/N falls for 0.1 percent of time is %.2f dB”,CNRN)
```

Scilab code Exa 12.17 Calculate the percentage of time the system stays above the threshold

```


//Variable Declaration

CNR=17.4    //Clear sky input C/N (dB)
T=10        //Threshold level for FM etector(dB)
Ta=272      //Value of Ta(Kelvin)
Tscs=544    //Value of Tscs(Kelvin)

//Calculation

TM=CNR-T     //Threshold margin at FM detector(dB)
CNR=10**(CNR/10)   //Converting CNR to ratio
NCR=1/CNR
```

```

function [y]=f(A)
    y=0.1-NCR*(A+(A-1)*Ta/Tscs)
endfunction
A=fsolve(2,f)

A=10*log10(A)   //Converting A into decibels
A=round(A)

// Getting the value of probablity of exceeding A
   from the curve

if (A==6) then
   P=2.5*10**-4
else
   printf("error")
end
Av=100*(1-P)   //Availability(percentage)

//Result

printf("The time system stays above threshold is %.3
   f percentage",Av)
```

Scilab code Exa 12.18 Calculate the combined C N0 ratio

```

//Variable Declaration

Nu=100    //Noise spectral density for uplink(dBHz)
Nd=87     //Noise spectral density for downlink(dBHz)

//Calculation

NOCR=10**(-Nu/10)+10**(-Nd/10)   //Noise to carrier
```

```
ratio
CNR=-10*log10(N0CR)    //Combined c/N0 ratio(dBHz)

//Result
printf("The combined carrier to noise ratio is %.2f dBHz",CNR)
```

Scilab code Exa 12.19 Calculate the C N for both links

```

//Variable declaration
//For uplink
Ps=-67.5    //Saturation flux density(dB)
A0=-37      //Antenna aperture at 6GHz(dB)
IBO=-11     //Input Backoff(dB)
GTRs=-11.6 //Satellite saturation G/T (dB)
k=-228.6    //Value of k(dB)

//For Downlink
EIRP=26.6  //Satellite EIRP(dB)
OBO=-6      //output Backoff(dB)
FSL=-196.7 //Free Space loss(dB)
GTRe=40.7  //Earth station G/T(dB)

//Calculation
CNRu=Ps+A0+IBO+GTRs-k      //Carrier to noise ratio for uplink(dB)
CNRd=EIRP+OBO+FSL+GTRe-k//Carrier to noise ratio for downlink(dB)
N0CR=10**(-CNRu/10)+10**(-CNRd/10)   //Noise to carrier ratio
CNR=-10*log10(N0CR)    //Combined c/N0 ratio(dBHz)

//results
printf("The Carrier to noise ratio for uplink is %.2
```

```
f dB”,CNRu)
printf(”The Carrier to noise ratio for downlink is %.2f dB”,CNRd)
printf(”The combined carrier to noise ratio is %.2f dBHz”,CNR)
```

Scilab code Exa 12.20 Calculate the overall carrier to noise ratio in decibels

```

//Variable Declaration

CNRu=23    //carrier to noise ratio for uplink(dB)
CNRd=20    //carrier to noise ratio for downlink(dB)
CNRm=24    //carrier to noise ratio for intermodulation(dB)

//Calculation

NCR=10**(-CNRu/10)+10**(-CNRd/10)+10**(-CNRm/10)   //Combined Noise to carrier ratio
CNR=-10*log10(NCR)   //Combined carrier to noise ratio(dB)

//Result
printf(”The combined carrier to noise ratio is %.2f dB”,CNR)
```

Chapter 13

Interference

Scilab code Exa 13.1 Determine the carrier to interference ratio at the ground receiving antenna

```
//Variable Declaration
EIRP1=34      //desired carrier EIRP from satellite(dB)
G1=44     // ground station receiving antenna gain(dB)
G2=24.47 //Gain in desired direction(dB)
EIRP2=34  //EIRP by interfering satellite(dB)
PD=4      //Polarization discrimination(dB)

//Calculation
CIR=EIRP1-EIRP2+G1-G2+PD  //Carrier to Interference ratio(dB)

//Result
printf(”The Carrier to interfernce ratio at the ground receiving antenna is %.2f dB”,CIR)
```

Scilab code Exa 13.2 Calculate the C I ratio on the uplink

```

//Variable Decalration

PA=24   //Transmit power by station A(dBW)
G1=54   //Antenna Gain(dB)
PC=30   //Transmit power by station C(dBW)
G2=24.47//off-axis gain in the S1 direction(dB)
PD=4      //Polarization discrimination(dB)

//Calculation

CIR=PA-PC+G1-G2+PD   //Carrier to Interference ratio
    (dB)

//Result
printf("The Carrier to interfernce ratio on uplink
    is %.2f dB",CIR)
```

Scilab code Exa 13.3 find the overall ratio

```

//Variable Declaration
CIR1=27.53   //Carrier to interference ratio from
    Example 13.1(dB)
CIR2=23.53   //Carrier to interference ratio from
    Example 13.2(dB)

//Calculation
ICRu=10**(-CIR1/10)  //Interferece to carrier ratio
    for uplink
ICRd=10**(-CIR2/10)  //Interferece to carrier ratio
    for downlink

ICRant=ICRu+ICRd   //Overall Interferece to carrier
    ratio
```

```
CIRant=-10*log10(ICRant)//Overall Carrier to
   interference ratio (dB)

//Result
printf("The overall carrier to interference ratio is
    %.2f dB",CIRant)
```

Scilab code Exa 13.4 Determine the degradation in the downlink

```

//Variable Declaration

SSi=4 //Initial satellite spacing(degrees)
SSl=2 //Later Satellite spacing(degrees)

//Calculation

IIR=(29-25*log10(SSl))-(29-25*log10(SSi))  //
   Increase in Interference(dB)

//Result
printf("The degradation in downlink C/I is %.1f dB",
   IIR)
```

Scilab code Exa 13.5 Calculate the protection ratio required to give a quality impairment factor

```

//Variable Declaration

f=4.2        //modualating frequency(MHz)
m=2.571      //Modulation index
QIF1=4.2     //Quality Impairment factor(a)
```

```
QIF2=4.5     //Quality Impairment factor(b)

//Calculation
Dv=2*m*f  //Peak to peak deviation(MHz)
PR1=12.5-20*log10(Dv/12)-QIF1+1.1*QIF1**2 //Protection ratio for case(a)
PR2=12.5-20*log10(Dv/12)-QIF2+1.1*QIF2**2 //Protection ratio for case(b)

//Results
printf("The protection ratio for quality impairment factor of 4.2 is %.1f dB",PR1)
printf("The protection ratio for quality impairment factor of 4.5 is %.1f dB",PR2)
```

Scilab code Exa 13.6 Calculate the transmission gain y

```

//Variable Decalration
LU=200   //Uplink propogation loss(dB)
LD=196   //Downlink propogation loss(dB)
GE=25    //Receiving gain of earth station(dB)
GE1=25   //Transmit gain of E1 in the direction of S(dB)
GS=9     //receive gain of S in the direction of E1(dB)
GS1=9   //Transmit gain of satellite S1 in the direction of E(dB)
GTE=48  //Transmit gain of E(dB)
GRE=48  //Receive gain of E(dB)
GRS=19  //Receive gain of S(dB)
GTS=19  //Transmit gain of S(dB)
US=-60    //Maximum power spectral density(dBJ)
US1=1   //Maximum power spectral density(uJ)
UE1=10  //Maximum power spectral density transmitted
```

```
by earth station(uJ)
UE=-50  //Maximum power spectral density transmitted by earth station(dBJ)
k=-228.6

//Calculation
URS=UE+GTE+GRS-LU//Received power spectral density at satellite S(dB)
URE=US+GTS+GRE-LD//Received power spectral density at satellite E(dB)
y=URE-URS    //Transmission gain for network R(dB)

I1=US+GS1+GE-LD  //Interference received by earth station(dB)
I2=UE+GE1+GS-LU  //Uplink Interference(dB)

delTE=I1-k   //Earth station receiver input(dBK)
delTE=10**(delTE/10)   //Earth station receiver input(K)
delTS=I2-k   //Noise temperature at satellite receiver input(dBK)

delTSE=y+delTS  //Noise Temperature rise(dBKelvin)
delTSE=10**(delTSE/10)  //Noise Temperature rise(Kelvin)
delT=delTSE+delTE  //Equivalent noise temperature rise


disp(URE)
disp(URS)

//Results
printf(”The transmission gain is %.f dB”,y)
printf(”The interference levels I1 an I2 are %.f %.f dBJ respectively”,I1,I2)
printf(”The equivalent temperature rise overall is %.2f Kelvin”,delT)
```

Chapter 14

Satellite Access

Scilab code Exa 14.1 Compare this with when no backoff needed

```

//Variable Declaration

Btr=36  //Transponder Bandwidth(MHz)
B=3     //Carrier Bandwidth(MHz)
EIRP=27  //saturated EIRP(dBW)
BO=6    //Back off loss(dB)
LOSSES=196  //Combined losses(dB)
GTR=30    //Earth station G/T ratio(dB)
k=228.6  //Value of k(dB)
//Calculation

Btr1=10*log10(Btr*10**6)  //Converting transponder
   Bandwidth into decibels
B1=10*log10(B*10**6)  //Converting carrier Bandwidth
    into decibels

CNR=EIRP+GTR-LOSSES+k-Btr1 //Carrier to noise ratio
   for single carrier operation(dB)
CNR=round(CNR)
alpha=-BO
```

```
K=alpha+Btr1-B1  //Fraction of Bandwidth actually
    occupied(dB)
K=10**(K/10)  //Converting decibels to ratio
K=round(K)

//Results

printf("The downlink carrier to noise ratio is %.0f
    dB",CNR)
printf("Fraction of Bandwidth actually occupied is %
    .0f",K)
printf("No. of carriers that would be accommodated
    without backoff is %.f",Btr/B)
```

Scilab code Exa 14.2 Determine the miss probabilty

```

//Variable decalration

N=40      //No.of bits
E=5       //Maximum number of errors allowed
p=10**-3 //Average probability of error in
    transmission

//Calculation

Pmiss=0
for i = E+1:N
  Pmiss=Pmiss+(factorial(N)/((factorial(i)*factorial
      (N-i))))*(p**i)*((1-p)**(N-i))
end

Pmiss=Pmiss*10**12

//Result
```

```
printf("The probability of miss is %.1f * 10^-12",Pmiss)
```

Scilab code Exa 14.3 Determine the probability of false detection

```

//Variable decalration
N=40      //No.of bits
E=5       //Maximum number of errors allowed

//Calculation
Pfalse=0
for i = 0:E
  Pfalse=Pfalse+(factorial(N)*2**-N)/((factorial(i)*factorial(N-i)))
end

Pfalse=Pfalse*10**7

//Result
printf("The probability of miss is %.1f * 10^-7",Pfalse)
```

Scilab code Exa 14.4 Calculate the frame efficiency

```

//Variable ecalration
Lf=120832       //Total frame length
Tb=14           //Traffic burts per frame
Rb=2            //Reference bursts per frame
T=103           //Guard interval(symbols)
P=280           //Preamble Symbols
```

```
R=P+8          //Reference channel symbols with
   addition of CDC
//Calculation

OH=2*(T+R)+Tb*(T+P)    //Overhead Symbols
nF=1-(OH/(Lf))    //Frame Efficiency

//Result
printf("Hence the frame efficiency of INTELSAT frame
    is %.3f",nF)
```

Scilab code Exa 14.5 Calculate the voice channel capacity for the INTELSAT frame

```

//Variable Declaration

Lf=120832    //Number of symbols per frame
Tf=2         //Frame period(ms)
nF=0.949     //INTELSAT fram efficiency from Example
   14.4
//Calculation

Rs=(Lf/(Tf))*10**-3   //Symbol rate(megasymbol/s)
Rt=Rs*2      //Transmission Rate
n=nF*Rt*10**3/64    //Voice channel capacity
n=round(n)
//Result

printf(" The voice channel capacity for the INTELSAT
    frame is %.0f Channels",n)
```

Scilab code Exa 14.6 Calculate the maximum transmission rate

```

//Variable Declaration

CNR=87.3    //Downlink Carrier to noise ratio(dBHz)
BER=10**-5  //Bit Error Rate Required
R=0.2   //Roll off factor
EbN0R=9.5    //Eb/N0 ratio(dB)

//Calculation
Rb=CNR-EbN0R  //Maximum Transmission Rate(dBb/s)
Rb1=10**(Rb/10) //Maximum Transmission Rate(b/s)
BIF=Rb1*1.2*10**-6/2  //IF Bandwith required

//Result
printf("The Maximum Transmission rate is %.2f dBb/s"
   ,Rb)
printf("The IF bandwidth required is %.2f MHz",BIF)
```

Scilab code Exa 14.7 calculate the earth station transmitter power needed for transmission

```

//Variable Declaration

T1=1.544  //Bit rate from sec.10.4(Mb/s)
R=62     //Bit rate from sec.10.4(dBMb/s)
EbN0R=12 //Required Eb/N0 ratio for uplink(dB)
LOSSES=212 //Transmission losses of uplink(dB)
GTR=10    //G/T ratio for earth station(dB/K)
G1=46   //Uplink antenna gain(dB)
Rd=74   //Downlink Transmission Rate(dBb/s)
//Calculation
CNR=EbN0R+R  //Carrier to noise ratio for uplink(dB)
EIRP=CNR-GTR+LOSSES-228.6 //EIRP of earth station
   antenna
```

```
P=EIRP-G1   //Transmitted Power Required(dBW)
P=10**(P/(10))   //Transmitted Power Required(Watts)

Ri=Rd-R   //Rate increase with TDMA operation(dB)
P1=1.4+Ri  //Uplink power increase required for TDMA operation(Watts)
P2=10**(P1/(10))

//Results
printf("Earth station transmission power required for transmission of T1 baseband signal is %.2f Watts",P)
printf("Uplink power increase required for TDMA operation is %f dBWatts or %.1f Watts",P1,P2)
```

Scilab code Exa 14.8 Calculate the processing gain in decibels

```

//Variable Declaration

BIF=36   //Bandwidth of channel over which carriers are spread(MHz)
R=0.4  //Rolloff factor for filtering
Rb=64   //Information bit rate(kb/s)
BER=10**-5  //Bit error rate required
EbN0R=9.6 //Eb/N0 ratio for BER given from Fig.10.18

//Calculation

Rch=BIF*10**6/(1+R) //Rate of unspreaded signal(chips/s)
Gp=Rch/(Rb*10**3)     //Processing gain
Gp1=round(10*log10(Gp))  //Processing gain(dB)
EbN0R1=10**(EbN0R/(10))  //Converting Eb/N0 into ratio
```

```
K=1+(1.4*Gp/EbN0R1)  //Number of channels
K=floor(K)

//Result
printf("The Processing Gain is %.f dB",Gp1)
printf("An estimate of maximum number of channels that can access the system is %.f",K)
```

Chapter 16

Direct Broadcast Satellite Services

Scilab code Exa 16.1 Calculate the look angles for the antenna the range and the Eb N0 at the IRD

```

//Varaible Declaration

EIRP=55  //EIRP for satellite(dBW)
fD=12.5  //Downlink frequency(GHz)
Pss=-101 //Receiving at ground station direction(degrees west)
Rb=40*10**6  //Transmission Rate(Hz)
D=18   //Diameter of antenna(inches)
n=0.55 //Efficiency of antenna
Tant=70 //Antenna noise(Kelvin)
Teq=100 //Equivalent noise temperature at LNA(Kelvin)
R=6371  //Radius of earth(Km)
L=2   //Transmission losses(dB)
aGSO=42164  //Circumference of earth(km)
k=-228.6   //Boltzmann's constant (dB)
PE=-90  //Longitude of Earth station(degrees west)
```

```
LE=45    //Latitude of Earth station(degrees north)
f=14     //Frequency(GHz)
//Calculation
B=PE-Pss
b=acos(cos(B*3.142/180)*cos(LE*3.142/180))
b=b*180/3.142
A=asin(sin(abs(B)*3.142/180)/sin(b*3.142/180))
A=A*180/3.142
Az=180+A  //Azimuth angle of antenna(degrees)
d=(R**2+aGSO**2-2*R*aGSO*cos(b*3.142/180))**0.5 //Range of antenna(km)
El=acos(aGSO*sin(b*3.142/180)/d)  //Elevation angle of antenna(radians)
El=El*180/3.142  //Elevation angle of antenna(degrees)
El=round(El)
d=round(d)
FSL=32.4+20*log10(d)+20*log10(f*10**3)  //Free space loss(dB)
LOSSES=FSL+L  //Total Transmission Losses
Ts=Teq+Tant   //Total system noise temperature(Kelvin)
T=10*log10(Ts)  //Total system noise temperature(dBK)
G=n*(3.192*f*(D/(12)))**2
G=10*log10(G)  //Antenna Gain(dB)
GTR=G-T   //G/T ratio(dB)
CNR=EIRP+GTR-LOSSES-k  //Carrier to noise ratio(dB)
Rb=10*log10(Rb) //Transmission Rate(dBHz)
EbN0R=CNR-Rb   //Eb/N0 ratio at IRD(dB)

//Results
printf("The Azimuth angle of antenna is %.1f degrees" ,Az)
printf("The Elevaation Angle of Antenna is %.f degrees",El)
printf("The Range of Antenna is %.f km",d)
printf("The Eb/N0 ratio at IRD is %.1f dB",EbN0R)
```

Scilab code Exa 16.2 Calculate the upper limit

```

//Varaible Declaration

R01=42  //Rainfall at earth station(mm/hr)
p=0.01  //Percentage of time for which rain exceeds
LE=45   //Latitue of earth station(degrees)
hR=3.5   //Rain Height(km)
h0=0    //Mean Sea level(km)
Ta=272  //
El=37   //Elevation angle of the antenna(degrees)
Ts=170   //Total system noise temperature(Kelvin)
NCR=2.3*10**-9 //Carrier to noise ratio
fD=12.5 //Frequency of operation(GHz)
f12=12   //Frequency 12GHz(GHz)
f15=15   //Frequency 15GHz(GHz)
//Coefficients for horizontal and vertical polarizations at 12GHz and 15GHz as given in Table 4.2

ah12=0.0188
av12=0.0168
bh12=1.217
bv12=1.2

ah15=0.0367
av15=0.0335
bh15=1.154
bv15=1.128

//Calculation

//Using Interpolation to find coefficients at 12.5
```

```
GHz
ah= ah12+(ah15-ah12)*(fD-f12)/(f15-f12)
bh= bh12+(bh15-bh12)*(fD-f12)/(f15-f12)
av=av12+(av15-av12)*(fD-f12)/(f15-f12)
bv= bv12+(bv15-bv12)*(fD-f12)/(f15-f12)

//Coefficients for circular polarization
ac=(ah+av)/2
bc=(ah*bh+av*bv)/(2*ac)
Ls1=(hR-h0)/sin(El*3.142/180)      //Slant Path Length(km)
Ls= Ls1                              //Slant Path Length(km)
LG= Ls*cos(El*3.142/180)        //Horizontal projection of slant path length(km)
r011=90/(90+4*LG)                         //Reduction Factor
r01= r011                              //Reduction Factor
L= Ls1*r01                            //Effective path length(km)
alpha= ac*R01**bc                     //Specific attenuation(dB/km)
A= 10**(alpha*L/(10))     //Total Attenuation(dB)
Trn=Ta*(1-1/A) //noise temperature with effect of rain
Tscs=Ts
NCrain=NCR*(A+(A-1)*Ta/Tscs) //Noise to carrier ratio due to rain
CNrain=-10*log10(NCrain)//Noise to carrier ratio due to rain(dB)
Rb=10*log10(40*10**6) //Transmission rate(dB)
EbN0rain= CNrain-Rb    //Upper limit of Eb/N0 ratio in prescence of rain(dB)

//Result
printf("Hence the upper limit for Eb/N0 for given conditions is %.1f dB",EbN0rain)
```