Scilab Textbook Companion for Principles of Communication Systems by H. Taub and D. L. Schilling[1]

Created by
Abhijeet Pasumarthy
B.E in Electronics & Communication Engineering
Electronics Engineering
Birla Institute of Technology, Mesra
College Teacher
None
Cross-Checked by
Spandana

June 5, 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: Principles of Communication Systems

Author: H. Taub and D. L. Schilling

Publisher: Tmh

Edition: 3

Year: 2011

ISBN: 978-0-07-064811-1

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 1

Introduction of Signals and Spectra

Scilab code Exa 1.1 Calculation of Energy

```
clc;
//page 12
//problem 1.1

//Given signal u = 2*exp(-3*t)

//Since the function integral does not accept %inf
   as limit we need to use approximation by changing
    variables.

//First the signal is to be expressed in terms of 'x
   '.

function y=Signal(x);
y=2*exp(-3*x);
endfunction;

//We then substitute x = tan(z), and then express
   the given signal wrt 'z' and not 'x'.
```

```

function y=Gmodified(z);
x=tan(z);
y=(Signal(x))^2/(cos(z))^2;
endfunction;

E = intg(0,atan(10),Gmodified)

disp(E,'The energy of this signal is ');
```

Scilab code Exa 1.2 Calculation of Power

```
clc;
//page 12
//problem 1.2

//Given signal u = 2*sin(0.5*%pi*t)

//Since u is periodic, averaging over -infinity to +
    infinity will give the same result as averaging
    over -2 to 2, where 4 is the time period.

t0 = -2;
t1 = 2;
E = integrate('(2*sin(0.5*%pi*t))^2','t',t0,t1)/4;

disp(E,'The power of the signal is ');
```

Scilab code Exa 1.3 Time shifting

```
clc;
//page 18
//problem 1.3
```

```
//u1(T) vs T
T = [-5:0.0082:5];
u1(T<=0) = 0;
u1(T>0) = 1;
xlabel('T');
ylabel('u(T)')

subplot(131);
plot2d(T,u1);

//u2(T-t) vs T
//Shifting the given signal by t units to the right, we get
//Let us assume the amount of time to be shited is 3 units
t = 3;

T = [-5:0.0082:5];
u2(T<=t) = 0;
u2(T>t) = 1;
xlabel('T');
ylabel('u(T - t)')

subplot(132);
plot2d(T,u2);

//u(t - T) = u(-(T - t))

T = [-5:0.0082:5];
u3(T>=t) = 0;
u3(T<t) = 1;
xlabel('T');
ylabel('u(t - T)')

subplot(133);
plot2d(T,u3);
```

Scilab code Exa 1.4 Time shifting

```
clc;
//page 18
//problem 1.4

//u1(t)
t = [-5:0.0082:5];
u1(t<=0) = 0;
u1(t>0) = 1;

xlabel('t');
ylabel('u(t)')

subplot(131);
plot2d(t,u1);

//u2(t-T)
//Shifting the given signal by t units to the right, we get
//Let us assume the amount of time to be shited is 3 units
T = 3;

t = [-5:0.0082:5];
u2(t<=T) = 0;
u2(t>T) = 1;

xlabel('t');
ylabel('u(t-T)')

subplot(132);
plot2d(t,u2);

```

```
//u(t) - u(t - T)

t = [-5:0.0082:5];
u3 = u1 - u2;

xlabel('t');
ylabel('u(t) - u(t-T)')

subplot(133);
plot2d(t,u3);
```

Scilab code Exa 1.5 Crosscorrelation and Autocorrelation

```
clc;
//page 18
//problem 1.5

//V1(t) = u(t) - u(t - 5)
t = [-5:0.1:5];
V1(t<=0) = 0;
V1(t>0) = 1;

xlabel('t');
ylabel('V1(t)')
subplot(121);
plot2d(t,V1);


//V2(t) = 2*t*(u(t) - u(t - 3))
t = [0:0.1:3];
V2 = 2*t;

xlabel('t');
ylabel('V2(t)')
```

```
subplot(122);
plot2d(t,V2);

//Autocorrelation R12(0) = R

R = integrate('2*t','t',0,3);

E1 = integrate('1','t',0,5);

//In the textbook, E2 has been computed as 18
   instead of 36
E2 = integrate('4*t^2','t',0,3);

c = R/(E1*E2)^0.5;

disp(R,'The correlation term R12(0) is ');
disp(E1,'The autocorrelation term R1(0) is ');
disp(E2,'The autocorrelation term R2(0) is ');
```

Scilab code Exa 1.6 Autocorrelation

```
clc;
//page 19
//problem 1.6

//V1(t) = u(t) - u(t - 5)
t = [-5:0.1:5];
V1(t<=0) = 0;
V1(t>0) = 1;

xlabel('t');
ylabel('V1(t)')
subplot(121);
plot2d(t,V1);

```

```
//V2(t) = 2*t*(u(t) - u(t - 3))
t = [0:0.1:3];
V2 = 2*t;

xlabel('t');
ylabel('V2(t)')
subplot(122);
plot2d(t,V2);

//Autocorrelation R12(1) = Ra
//The range is t = 0 to 2, as signal V2(t) has been shifted left by one unit, V2(t-1)

Ra = integrate('2*(t+1)','t',0,2);

disp(Ra,'The correlation term R12(1) is ');

//Autocorrelation R12(-1) = Rb
//The range is t = 1 to 4, as signal V2(t) has been shifted right by one unit, V2(t+1)

Rb = integrate('2*(t-1)','t',1,4);

disp(Rb,'The correlation term R12(-1) is ');
```

Chapter 2

Random Variables and Processes

Scilab code Exa 2.1 Probability of two Events

```
clc;
//page 85
//problem 2.1

//A & B are two events occured in sample space S,
   where P(A) & P(B) are their corresponding
   probability
P_S=1

//Given A&B are not mutually exclusive events,
//Probability of A is 0.2 = P_A
//Probability of B is 0.4 = P_B
//Probability of either A or B is 0.5 = P_AUB
P_A = 0.2
P_B =  0.4
P_AUB = 0.5

//Probability of both of A&B jointly occur is
   P_AinterB = P_A+P_B-P_AUB where inter is
```

```
    intersection
P_AinterB = P_A+P_B-P_AUB

disp('Probability of both of A&B jointly occur is '+string(P_AinterB))

//Probability of none of AorB are occur is P_NOAB = Total occurence(P_S) - Probability of either AorB (P_AUB)
P_NOAB = P_S-P_AUB

disp('Probability of none of AorB are occur is '+string(P_NOAB))
```

Scilab code Exa 2.2 Bayes Theorem

```
clc;
//page 86
//problem 2.2

//Probability that A will occur if B has already occurred(P_AB) = ratio of Probability of joint occurence of  A&B (P_A_B) & Probability of B(P_B)
//P_A_B(robability of joint occurence) = Probability that A&B both occur(P_AinterB)

//From given values P_AinterB = 0.1 implies P_A_B = 0.1 & P_B = 0.4
P_AinterB = 0.1
P_A_B = P_AinterB
P_B = 0.4

P_AB = P_A_B/P_B

//Similarly
```

```
//Probability that B will occur if A has already occurred(P_BA) = ratio of Probability of joint occurence of A&B (P_A_B) & Probability of B(P_A)

//From given values P_A = 0.2
P_A = 0.2

P_BA = P_A_B/P_A

//Bayes theorem says that P_AB = (P_A/P_B)*P_BA
//After Calculating LHS & RHS if both are equal then bayes theorem is satisfying

//Calculating LHS
LHS = P_AB

//Calculating RHS
RHS = (P_A/P_B)*P_BA

disp('P(A/B) = '+string(P_AB) );

if LHS == RHS then
    disp('LHS = RHS, Hence Bayes theorem is verified');
end
```

Scilab code Exa 2.5 Optimum Receiver Algorithm

```
clc;
//page 95
//problem 2.5

//Given, The probability that m0 is sent is 0.7 = P_m0
P_m0 = 0.7
```

```

//The probability that m0 is sent is 0.3 = P_m1
P_m1 = 0.3

//The probability that r0 is received given that m0 is sent is 0.9 = P_r0m0 where r is voltage & m is message
P_r0m0 = 0.9

//the probability that r1 is received given that m0 is sent is 0.1 = P_r1m0 where r is voltage & m is message
P_r1m0 = 0.1

//The probability that r1 is received given that m1 is sent is 0.6 = P_r1m1
P_r1m1 = 0.6

//the probability that r0 is received given that m1 is sent is 0.4 = P_r0m1 where r is voltage & m is message
P_r0m1 = 0.4

//With the given values check eqations P_r0m0*P_m0(P00) > P_r0m1*P_m1(P01)
P00 = P_r0m0*P_m0
P01 = P_r0m1*P_m1

if P00>P01 then
    disp('as P(r0|m0)*P(m0) > P(r0|m1)*P(m1) is valid, we whould select m0 whenever r0 is received')
end

//With the given values check eqations P_r1m1*P_m1(P11) > P_r1m0*P_m0(P10)
P11 = P_r1m1*P_m1
P10 = P_r1m0*P_m0
```

```

if P11>P10 then
    disp('as P(r1|m1)*P(m1) > P(r1|m0)*P(m0) is valid , we whould select m1 whenever r1 is received')
end
```

Scilab code Exa 2.6 Optimum Receiver Algorithm

```
clc;
//page 96
//problem 2.6

//Given, the probability that r0 is received given that m0 is sent is 0.9 = P_r0m0 where r is voltage & m is message
P_r0m0 = 0.9

//The probability that m0 is sent is 0.7 = P_m0
P_m0 = 0.7

//The probability that r1 is received given that m1 is sent is 0.6 = P_r1m1
P_r1m1 = 0.6

//The probability that m0 is sent is 0.3 = P_m1
P_m1 = 0.3

//The probability that the transmitted signal is correctly read at receiver is P(c)(P_c) = the probability that m0 was sent when r0 was read(P_r0m0*P_m0) + the probability that m1 was sent when r1 was read(P_r1m1*P_m1)

P_c = P_r0m0*P_m0+P_r1m1*P_m1
```

```

//P(e)(P_e) = 1-P(c)
P_e = 1-P_c

disp('P(e) = '+string(P_e))
disp('P(c) = '+string(P_c))
```

Scilab code Exa 2.7 Optimum Receiver

```
clc;
//page 96
//problem 2.7

//Here P(ra|mb) is denoted as P_ramb where a is 0,1,2 & b is 0,1
//P(X) is denoted as P_X where X is m0, m1, C & E

//From given values P_m0 = 0.6, P_m1 = 0.4, P_r0m0 =0.6, P_r1m1 = 0.7, P_r0m1 = 0, P_r1m0 = 0.2, P_r2m0 = 0.2 & P_r2m1 = 0.3
P_m0 = 0.6
P_m1 = 0.4
P_r0m0 =0.6
P_r1m1 = 0.7
P_r0m1 = 0
P_r1m0 = 0.2
P_r2m0 = 0.2
P_r2m1 = 0.3

//(a)
//Comaparing P(r0|m0)*P(m0) & P(r0|m1)*P(m1) gives result
LHS = P_r0m0*P_m0
RHS = P_r0m1*P_m1

```

```
disp('As P(r0|m0)*P(m0)['+string(LHS)+'] > P(r0|m1)*P(m1)['+string(RHS)+']')
disp('we select m0 whenever r0 is received')

//Similarly compare P(r1|m1)*P(m1) & P(r1|m0)*P(m0)
LHS = P_r1m1*P_m1
RHS = P_r1m0*P_m0

disp('As P(r1|m1)*P(m1)['+string(LHS)+'] > P(r1|m0)*P(m0)['+string(RHS)+']')
disp('we select m1 whenever r1 is received')

//compare P(r2|m0)*P(m0) & P(r2|m1)*P(m1)
LHS = P_r2m0*P_m0
RHS = P_r2m1*P_m1

disp('As P(r2|m0)*P(m0)['+string(LHS)+'] = P(r2|m1)*P(m1)['+string(RHS)+']')
disp('We can accordingly make either assignment and we arbitrarily associate r2 with m0')

//(b)
//The probability of being correct is P(C) = P(r0|m0)*P(m0)+P(r1|m1)*P(m1)+P(r2|m0)*P(m0)
P_C = P_r0m0*P_m0+P_r1m1*P_m1+P_r2m0*P_m0

//The probability of error is P(E) = 1-P(C)
P_E = 1 - P_C;

disp('Probability of being correct is P(C) = '+string(P_C)')
disp('Probability of error is P(E) = '+string(P_E)')
```

Scilab code Exa 2.8 Probability Distribution Function

```
clc;
//page 99
//problem 2.8

//Given, probability density function of X is fX_x
   where fX_x = a*e^(-0.2*x) for x greater than &
   equal to 0 & = 0 eleswhere

//a = fX_x/(a*e^(-0.2*x))
//from definition integration of fX_x with limits -
   infinity to +infinity is 1
//As per given fX_x, integration of a*e^(-0.2*x)
   with limits 0 & +inffinity and obtained value be
   P
//a = 1/p

P = integrate('%e^(-0.2*x)','x',0,100)
a = 1/P

disp('a = '+string(a))
```

Scilab code Exa 2.10 Probability of Error

```
clc;
//page 105
//problem 2.10

//We know that, Probabilty of error(P_error) for the
    signal correpted by Gaussian channel variance
   sigma^2 where signal having voltage levels as 0&V
    is (1/2)*erfc(V/(2*sqrt(2)*sigma))

//P_error for V = 4 & sigma^2 =2
V = 4
sigma = sqrt(2)
```

```
P_error = (1/2)*erfc(V/(2*sqrt(2)*sigma))

disp('Probabilty of error for V = 4 & sigma^2 =2 is
    '+string(P_error))

//P_error for V = 2 & sigma^2 =2
V = 2
sigma = sqrt(2)
P_error = (1/2)*erfc(V/(2*sqrt(2)*sigma))

disp('Probabilty of error for V = 2 & sigma^2 =2 is
    '+string(P_error))

//P_error for V = 4 & sigma^2 =4
V = 4
sigma = sqrt(4)
P_error = (1/2)*erfc(V/(2*sqrt(2)*sigma))

disp('Probabilty of error for V = 4 & sigma^2 =4 is
    '+string(P_error))

//P_error for V = 8 & sigma^2 =2
V = 8
sigma = sqrt(2)
P_error = (1/2)*erfc(V/(2*sqrt(2)*sigma))

disp('Probabilty of error for V = 8 & sigma^2 =2 is
    '+string(P_error))
```

Scilab code Exa 2.11 Probability of Error

```
clc;
//page 106
//problem 2.11

```

```
//(a)
//out of n attempts the probability of message reaching correctly for k times is given by binomial distribution pX(k) = nCk*(q^k)*(1-q)^(n-k) where q is probability of correctly reaching

//Here n = 10, k = 1, q = 0.001
n = 10
k = 1
q = 0.001

//pX(k) is denoted as p_X_1
//10C1 =10
p_X_1 = 10*(q^k)*(1-q)^(n-k)

disp('The probability that out of 10 transmissions 9 are corrent and 1 is incorrect is '+string(p_X_1))

//probability that more than two erroneous out of 100 transmissions(p_100_2) = 1-probability of less than or equal to two error in transmission
//p_100_2 = 1-pX(0)-pX(1)-pX(2)
//p_100_2 =1-100C0*((0.001)^0)*((1-0.001)^100)-100C1*((0.001)^1)*((1-0.001)^99)-100C0*((0.001)^2)*((1-0.001)^98)

//Since, calculation of above is cumbersome we may use Poisson ditribution to approximate above
//Poisson distribution = pX(k) = (alfa^k)*(e^-alfa)/k!, where alfa = n*T

//Here n = 100 & q = 0.001
n = 100
q = 0.001

alfa = n*q

```

```
p_100_2 = 1-(alfa^0)*(%e^-0.1)/factorial(0)-(alfa^1)*(%e^-0.1)/factorial(1)-(alfa^2)*(%e^-0.1)/factorial(2)

disp('probability that more than two erroneous out of 100 transmissions is '+string(p_100_2))

//(c)
//from(b), required probability i.e probability of more than one are erroneous out of 100 transmission(p_100_1) is
p_100_1 = 1-(alfa^0)*(%e^-0.1)/factorial(0)-(alfa^1)*(%e^-0.1)/factorial(1)

disp('probability that more than one erroneous out of 100 transmissions is '+string(p_100_1))
```

Scilab code Exa 2.13 Probability of Error

```
clc;
//page 115
//problem 2.13

//Given, Error probability is 10^-4 = P_e, no of ecperiments conducted = N = 4*10^5 & estimated probability of error p does not differ from P_e by more than 50%
P_e = 10^-4
N = 4*10^5

//Tchebycheff's inequality is P(|p-Pe|>=E)<=P_e/(N*E^2)
//From given values we can find that E = 50*10^-4
E = 50*10^-4

```

```
//Here R.H.S of Tchebycheff's inequality is denoted as Tc_RHS
Tc_RHS = P_e/(N*E^2)

//Tc_RHS in persentage is Tc_RHSper
Tc_RHSper = Tc_RHS/100

//disp(Tc_RHSper,Tc_RHS,'or P(|p-10^-4|>=0.5*10^-2)<=',Tc_RHS,'The probability of estimated probability of error p does not differ from P_e by more than 50% is less than equal to')

//given solution has been computed wrong, obtaines solution is 10^-7
disp('The probability of estimated probability of error p does not differ from P_e by more than 50% is less than equal to '+string(Tc_RHS)+'or P(|p-10^-4|>=0.5*10^-2)<='+string(Tc_RHS)+' = '+string(Tc_RHSper)+'%')
```

Chapter 3

Amplitude Modulation Systems

Scilab code Exa 3.4 Transmission Power Efficiency

```
clc;
//page 163
//problem 3.4

//Transmission power effiency n = {(m^2)/[2+(m^2)
    ]}*100% where m is modulated index

//Given modulated indices are m1 = 0.25, m2 = 0.5 &
    m3 = 0.75

//Transmission power effiencies are n1, n2 & n3
    respectively for m1, m2 & m3
n1 = {(0.25^2)/[2+(0.25^2)]}*100
n2 = {(0.5^2)/[2+(0.5^2)]}*100
n3 = {(0.75^2)/[2+(0.75^2)]}*100

disp('Transmission power effiency for modulated
    index 0.25 is '+string(n1)+ '%')
disp('Transmission power effiency for modulated
    index 0.5 is '+string(n2)+ '%')
disp('Transmission power effiency for modulated
```

```
index 0.75 is '+string(n3)+ '%')
```

Scilab code Exa 3.7 Calculation of L and C of Power Amplifier

```
clc;
//page 185
//problem 3.7

//Given input inmedance of matching networkis R1 = 10 ohm & output impedance of matching networ is R2 = 50 ohm & carrier frequency is fc = 500 KHz
R1 = 10
R2 = 50
fc = 500000

//Wc = 2*pi*fc
Wc = 2*%pi*fc

//AS R1 = R2*(X2^2)/[(R2^2)+(X2^2)], X2 = 25ohm
X2 = 25

//AS X1 = (R2^2)*X2/[(R2^2)+(X2^2)] & R1>R2, X1 = -20ohm
X1 = -20

//|X1| = |jwL| = wL = 20 & |X2| = |1/jwC| = 1/wC = 25, so |X1*X2| = L/C = 500 denotes as LC_div
LC_div = 500

//Wc^2 = 1/(L*C). LC is denoted as LC_prod
LC_prod = 1/(Wc^2)

//In the textbook the calculated LC = 10^-3, in reality the value of LC = 1.013D-13

```

```
L = sqrt(LC_div*LC_prod)

//In the textbook the calculated L^2 = 50*10^-14, in
    reality the value of L^2 = 5.066D-11

C = L/500

//In the textbook the calculated C = 1.4*10^-9, in
    reality the value of C = 1.424D-08

disp('Inductance '+string(L)+' H')
disp('Capacitance '+string(C)+' F')
```

Scilab code Exa 3.8 Calculation of Gain and Power radiated of Antenna

```
clc;
//page 185
//problem 3.8

//Given ohmnic loss resistance is Ro = 12 Ohm,
Ro = 12

//radiation resistance is Rr = 48 Ohm,
Rr = 48

//directivity is D = 2
D = 2

//Input current = 0.1*cos[2*pi*(10^6)*t], Amplitude
    of input current is A = 0.1 Amp
A = 0.1

//Equivalent resistance = Re = Ro+Rr
Re = Ro+Rr

```

```
//Total power used in antenna = Pin = (A^2)*Re/2
Pin = (A^2)*Re/2

//Power used in radiation = Prad = (A^2)*Rr/2
Prad = (A^2)*Rr/2

//Efficiency of the antenna = n = Prad/Pin
n = Prad/Pin

//Gain of antenna = Ga = efficiency*directivity
Ga = n*D

disp('Total power used in antenna '+string(Pin)+'
    Watt')
disp('Power used in radiation '+string(Prad)+' Watt'
    )
disp('Efficiency of the antenna '+string(n))
disp('Gain of antenna '+string(Ga))
```

Chapter 4

Angle Modulation

Scilab code Exa 4.2 Calculation of frequency parameters

```
clc;
//page 199
//problem 4.2

//Given angle modulated signal is x = 3*cos[2*pi*(10^6)*t+2*sin(2*pi*10^3*t)]

//So, phase of the angle modulates signal is Q = 2*pi*(10^6*t)+2*sin(2*pi*(10^3)*t)

//Instantaneous frequency = dQ/dt = 2*pi*(10^6)+ 4*pi*(10^3)*sin(2*pi*(10^3)*t)

//For Instantaneous frequency at 0.25ms, Substituting t = 0.25ms in Instantaneous frequency
//Instantaneous frequency is expressed as f1_rad for frequency in radians per second
f1_rad = 2*%pi*(10^6)+ 4*%pi*(10^3)*sin(2*%pi*(10^3)*0.00025)

```

```
//Instantaneous frequency is expressed as f1_hz for frequency in hertz
f1_hz = f1_rad/(2*%pi)

disp('the Instantaneous frequency at time t=0.25ms is '+string(f1_rad)+' rad/sec = '+string(f1_hz)+' Hz')

//For Instantaneous frequency at 0.25ms, Substituting t = 0.5ms in Instantaneous frequency
//Instantaneous frequency is expressed as f2rad for frequency in radians per second
f2_rad = 2*%pi*(10^6)+ 4*%pi*(10^3)*sin(2*%pi*(10^3)*0.0005)

//Instantaneous frequency is expressed as f2hz for frequency in hertz
f2_hz = f2_rad/(2*%pi)

disp('the Instantaneous frequency at time t=0.5ms is '+string(f2_rad)+' rad/sec = '+string(f2_hz)+' Hz')

//Maximum phase deviation = max[2*sin(2*pi*(10^3)*t)] = 2*1
maxDp = 2;

disp('Maximum phase deviation is '+string(maxDp)+' rad')

//Maximum frequency deviation = max[4*pi*(10^3)*sin(2*pi*(10^3)*t)] = 4*pi*(10^3)*1
maxDf = 4*%pi*(10^3)*1;

disp('Maximum frequency deviation is '+string(maxDf)+' Hz')
//disp('in rad',maxDf,'Maximum frequency deviation is')
```

```

//In the textbook the calculated value of max
   frequency devaition is = 2000 Hz, in reality the
   value = 12566.371 Hz
```

Scilab code Exa 4.4 Calculation of Bandwidth

```
clc;
//page 208
//problem 4.4

//Given modulating signal m(t) = 2*sin(2*pi*(10^3)*t
   ), B for phase modulation Bp = 10 & for fequency
   modulation Bf = 10
Bp = 10
Bf = 10

//So Amplitude of modulating signal is Am=2 metres
Am = 2

//Frequency of modulating signal is fm = 1000 hertz
fm=1000

//Bandwidth = 2*(1+B)*fm

//Bandwidth for phase modulation with modulating
   signal m(t) is bw_pm = 2*(1+Bp)*fm
bw_pm = 2*(1+10)*1000

//Bandwidth for frequency modulation with modulating
    signal m(t) is bw_fm = 2*(1+Bf)*fm
bw_fm = 2*(1+10)*1000

disp('Bandwidth for phase modulation '+string(bw_pm)
   +' Hz')
```

```
disp('Bandwidth for frequency modulation '+string(bw_fm)+' Hz')

//Bandwidth for phase & frequency modulation if frequency of modulating signal is doubled i.e fm = 2000 hertz

//Bp & Bf after frequency of modulating signal is doubled

//Bp = kp*Am, observing the equation as there is no change in amplitude Bp = 10
Bp = 10

//Bf = kf*Am/fm, observing the equation as there is change in frequency  Bf = 10/2 = 5
Bf = 5

//Bandwidth for phase modulation if frequency of modulating signal is doubled is bw_double_pm = 2*(1+Bp)*fm
bw_double_pm = 2*(1+10)*2000

//Bandwidth for frequency modulation if frequency of modulating signal is doubled is bw_double_fm = 2*(1+Bf)*fm
bw_double_fm = 2*(1+5)*2000

disp('Bandwidth for phase modulation for doubled frequency '+string(bw_double_pm)+' Hz')
disp('bandwidth for frequency modulation for doubled frequency '+string(bw_double_fm)+' Hz')

//Bandwidth for phase & frequency modulation if amplitude of modulating signal is halfed i.e Am = 1 metre

//Bp & Bf after amplitude of modulating signal is
```

```
halfed

//Bp = kp*Am, observing the equation as there is change in amplitude Bp = 10/2 = 5
Bp = 5

//Bf = kf*Am/fm, observing the equation as there is change in amplitude Bf = 5/2 = 2.5
Bf = 2.5

//Bandwidth for phase modulation if frequency of modulating signal is doubled is bw_halfed_pm = 2*(1+Bp)*fm
bw_halfed_pm = 2*(1+5)*2000

//Bandwidth for frequency modulation if frequency of modulating signal is doubled is bw_halfed_fm = 2*(1+Bf)*fm
bw_halfed_fm = 2*(1+2.5)*2000

disp('Bandwidth for phase modulation for halfed amplitude '+string(bw_halfed_pm)+' Hz')
disp('Bandwidth for frequency modulation for halfed amplitude '+string(bw_halfed_fm)+' Hz')
```

Chapter 5

Pulse Modulation and Digital Transmission of Analog Signal

Scilab code Exa 5.2 Minimum sampling rate

```
clc;
//page 247
//problem 5.2

//Highest frequency(fH) = 10000/2 = 5000 Hz
fH = 5000

//Lowest frequency(fL) = 6000/2 = 3000 Hz
fL = 3000

//Minimum sampling frequency from low pass
   consideration(S_LOW) = 2*fH
S_LOW = 2*fH

disp('Minimum sampling frequency from low pass
   consideration is '+string(S_LOW)+' Hz')

//B = fH-fL = 2000 Hz
B = fH-fL
```

```
//k = floor(fH/B) = 2, where floor(x) gives the largest integer that does not exceed x
k = floor(fH/B)

//The required sampling frequency from band pass consideration(S_BAND) = 2*fH/k
S_BAND = 2*fH/k

disp('Minimum sampling frequency from band pass consideration is '+string(S_BAND)+' Hz')
```

Scilab code Exa 5.3 Calculation of Guard time

```
clc;
//page 259
//problem 5.3

//Given width of each pulse W = 150 us
W = 150 * 10^-6

//One cycle is a period,T = 1ms
T = 1000 * 10^-6

//There are 5 messages multiplexed each utilizeallocated time pulse width = s(T_5) = T/5
T_5 = T/5

//Gaurd time(GT_5) = allocated time-pulse width = T_5-W
GT_5 = T_5-W

disp('Gaurd time where 5 messages multiplexed is '+string(GT_5)+' seconds')

```

```
//Here there are 10 messages multiplexed each utilizeallocated time pulse width = s(T_10) = T/10
T_10 = T/10

//Gaurd time(GT_10) = allocated time-pulse width = T_10-norrow pulses width = T_10 -50* 10^-6
GT_10 = T_10 - 50 * 10^-6

disp('Gaurd time where 10 messages multiplexed is '+string(GT_10)+' seconds')
```

Scilab code Exa 5.5 Calculation of minimum number of Binary digits

```
clc;
//page 272
//problem 5.5

//Let Abe the maximum value of the discrete samples.
//Error tolerated is 0.1% i.e. 0.001A
//If D is step size then possible maximum error is D/2
//Thus D/2 = 0.001A or A/D = 500 = no of levels required(Levels)
Levels = 500

//minimum no of binary digits required(B) = rounded value to the next higher integer of log2 (Levels)
B = round(log2 (Levels))

disp('Minimum no of binary digits required '+string(B))
```

Scilab code Exa 5.6 Companding

```
clc;
//page 273
//problem 5.6

//The y axis is uniformly quantized with step size(
    step_size) = 1/[(2^8)/(2-1)] in both +ve & -ve
    direction between 1 & -1 when peak of input
    varies between 1 & -1.
//The smallest step in x direction occurs nearest to
     x=0 i.e between y1 = 0 & y2 = step_size
step_size = 1/[(2^8)/2-1]
y1 = 0
y2 = step_size

//Then, y1 = [ln(1+255*x1)]/[ln(1+255)]

x1 = (%e^(y1*log(256)) - 1)/255;

//y2 = [ln(1+255*x2)]/[ln(1+255)]
x2 = (%e^(y2*log(256)) - 1)/255;

//The smallest step size is 10*(x2-x1)
disp('The smallest step size is '+string(10*(x2-x1))
   +' Volts')

//The largest step size occurs when x is at its
   extreme between y1 = 1-1/127 = 126/127 & y2 = 1
y1 = 1-1/127
y2 = 1

//Then, y1 = [ln(1+255*x1)]/[ln(1+255)]

x1 = (%e^(y1*log(256)) - 1)/255;

//y2 = [ln(1+255*x2)]/[ln(1+255)]
x2 = (%e^(y2*log(256)) - 1)/255;
```

```
//The largest step size is 10*(x2-x1)
disp('The largest step size is '+string(10*(x2-x1))+
    ' Volts')
```

Scilab code Exa 5.9 LMS Algorithm

```
clc;
//page 296
//problem 5.9

//for error calculation e(n) = m(n) - [^hj(n)*m(n-1)
    +^hj(n)*m(n-2)+^hj(n)*m(n-3)+ ........+^hj(n)*m(n
    -N)]

//for coefficient upgradation ^hj(n+1) = ^hj(n)+um(n
    -j)e(n) where u = learning parameter = 0.1.
u = 0.1

//Assign m values taking from m = -3 to 5
//Denoting m(x) as matrix m where each element
    repesents from n = -3 to 5
m = [0 0 0 1 2 3 4 5 6]

//taking e(n) as matrix e, ^hj(n) as matrises h_j
e = zeros(1,5)
h_1 = zeros(1,6)
h_2 = zeros(1,6)

//given ^h1(0)= ^h2(0) = 0
h_1(1) = 0
h_2(1) = 0

for i = 1:5
    e(i) = m(i+3) - h_1(i)*m(i+2) - h_2(i)*m(i+1)
```

```
    h_1(i+1) = h_1(i) + u*m(i+2)*e(i)
    h_2(i+1) = h_2(i) + u*m(i+1)*e(i)
end

//here e(3) is given as 1.32 but it is displaying 0.92
//here ^h2(3) is given as 0.26 but it is displaying 0.46

for i = 1:5
    disp('e('+string(i-1)+') = '+string(e(i)))
    disp('^h1('+string(i)+') = '+string(h_1(i+1)))
    disp('^h2('+string(i)+') = '+string(h_2(i+1)))
end
```

Scilab code Exa 5.11 SNR of a DM system

```
clc;
//page 296
//problem 5.11

//case 1(a)
//f = 400Hertz, fs = 8000Hertz
f = 400
fs = 8000

//We know that maximum signal to noise ratio(SNR_max) = 3*(fs^2)/(8*(pi^2)*(f^2))
SNR_max = 3*(fs^2)/(8*(%pi^2)*(f^2))
//SNR_max in decibels is SNR_max_db
SNR_max_db = 10*log10 (SNR_max)

disp('Maximum signal to noise ratio for f = 400 & fs = 8000 is '+string(SNR_max)+' = '+string(SNR_max_db)+' db')
```

```

//case 1(b)
//f = 400Hertz, fs = 16000Hertz
f = 400
fs = 16000

//We know that maximum signal to noise ratio(SNR_max) = 3*(fs^2)/(8*(pi^2)*(f^2))
SNR_max = 3*(fs^2)/(8*(%pi^2)*(f^2))

//SNR_max in decibels is SNR_max_db
SNR_max_db = 10*log10 (SNR_max)

//Given solution is 13.8385 dB obtained solution is 17.838515 dB

disp('Maximum signal to noise ratio for f = 400 & fs = 16000 is '+string(SNR_max)+' = '+string(SNR_max_db)+' db')

//case 2(a)
//f = 400Hertz, fs = 8000Hertz & fc =  1000Hertz
f = 400
fs = 8000
fc = 1000

//If a 1kHz low pass post reconstruction filter is used then maximum signal to noise ratio(SNR_max) = 3*(fs^3)/(8*(pi^2)*(f^2)*fc)
SNR_max = 3*(fs^3)/(8*(%pi^2)*(f^2)*fc)
//SNR_max in decibels is SNR_max_db
SNR_max_db = 10*log10 (SNR_max)

disp('If a 1kHz low pass post reconstruction filter is used then')

disp('Maximum signal to noise ratio for f = 400, fs = 8000 & fc = 1000 is '+string(SNR_max)+' = '+
```

```
string(SNR_max_db)+' db')

//case 2(b)
//f = 400Hertz, fs = 16000Hertz & fc =  1000Hertz
f = 400
fs = 16000
fc = 1000

//If a 1kHz low pass post reconstruction filter is used then maximum signal to noise ratio(SNR_max) = 3*(fs^3)/(8*(pi^2)*(f^2)*fc)
SNR_max = 3*(fs^3)/(8*(%pi^2)*(f^2)*fc)
//SNR_max in decibels is SNR_max_db
SNR_max_db = 10*log10 (SNR_max)

disp('Maximum signal to noise ratio for f = 400, fs = 16000 & fc = 1000 is '+string(SNR_max)+' = '+string(SNR_max_db)+' db')
```

Chapter 6

Digital Modulation and Transmission

Scilab code Exa 6.1 Carrier phase variation

```
clc;
//page 341
//problem 6.1

//Given messages signal m = [1,0,1,1,0,1]
m = [1,0,1,1,0,1];

//Logical 0 corrsponds to pi i.e %pi and Logical 1
    corresponds to 0

//
    /////////////////////////////////////////////////////////////////

//For BPSK, from the above deduction let the carrier
    phase be Carrier_Phase_BPSK
for i = 1:5
    if m(i)==1 then
        Carrier_Phase_BPSK(i) = 0;
```

```
    else
        Carrier_Phase_BPSK(i) = %pi;
    end
end

disp(Carrier_Phase_BPSK,'The Phase of the carrier signal for BPSK varies as ');

//////////////////////////////////////////////////////////////////////

//For DPSK
//Let b represent the input to balance modulator

//If the initial value of b be 0
b = 0;

for i = 2:5
    b(i) = bitxor(m(i),b(i-1))
end

//Now the carrier phase, Carrier_Phase_DPSK
for i = 1:5
    if b(i)==1 then
        Carrier_Phase_DPSK(i) = 0;
    else
        Carrier_Phase_DPSK(i) = %pi;
    end
end

//Now the carrier amplitude, Carrier_Amplitude_DPSK
for i = 1:5
    Carrier_Amplitude_DPSK(i) = cos(Carrier_Phase_DPSK(i));
end

disp(Carrier_Phase_DPSK,'The Phase of the carrier
```

```
signal for DPSK varies as follows, '+'when the initial value of b is 1');
disp(Carrier_Amplitude_DPSK,'The Amplitude of the carrier signal for DPSK varies as follows, '+' when the initial value of b is 1');

//If the initial value of b be 1
b = 1;

for i = 2:5
    b(i) = bitxor(m(i),b(i-1))
end

//Now the carrier phase, Carrier_Phase_DPSK
for i = 1:5
    if b(i)==1 then
        Carrier_Phase_DPSK(i) = 0;
    else
        Carrier_Phase_DPSK(i) = %pi;
    end
end

//Now the carrier amplitude, Carrier_Amplitude_DPSK
for i = 1:5
    Carrier_Amplitude_DPSK(i) = cos(Carrier_Phase_DPSK(i));
end

disp(Carrier_Phase_DPSK,'The Phase of the carrier signal for DPSK varies as follows, '+'when the initial value of b is 0');
disp(Carrier_Amplitude_DPSK,'The Amplitude of the carrier signal for DPSK varies as follows, '+' when the initial value of b is 0');

////////////////////////////////////////////////////////////////////////
```

```

//For DEPSK
//The DEPSK transmitter output is same as that of DPSK

//If the initial value of b be 0
b = 0;

for i = 2:5
    b(i) = bitxor(m(i),b(i-1))
end

//Now the carrier phase, Carrier_Phase_DPSK
for i = 1:5
    if b(i)==1 then
        Carrier_Phase_DEPSK(i) = 0;
    else
        Carrier_Phase_DEPSK(i) = %pi;
    end
end

disp(Carrier_Phase_DEPSK,'The Phase of the carrier signal for DEPSK varies as follows, '+'when the initial value of b is 1');

//If the initial value of b be 1
b = 1;

for i = 2:5
    b(i) = bitxor(m(i),b(i-1))
end

//Now the carrier phase, Carrier_Phase_DPSK
for i = 1:5
    if b(i)==1 then
        Carrier_Phase_DEPSK(i) = 0;
    else
        Carrier_Phase_DEPSK(i) = %pi;
```

```
    end
end

disp(Carrier_Phase_DEPSK,'The Phase of the carrier signal for DEPSK  varies as, '+'when the initial value of b is 0');
```

Scilab code Exa 6.2 DPSK and DEPSK

```
clc;
//page 341
//problem 6.2

//From Ex6_1 the obtained carrier amplitude is c

/////////////////////////////////////////////////////////////////////

//For DPSK
//Considering the initial value of the storage element to be 0 in polar and -1 in biploar
c = [1,1,-1,1,1];
y = -1;
//Let the output be y
for i = 2:5
    y(i) = c(i)*c(i-1);
end

//Converting back to binary data
for i = 1:5
    if y(i)== -1 then
        output_binary(i) = 0;
    else
        output_binary(i) = 1;
```

```
    end
end

//Now inverting the output we get:
for i = 1:5
    output_binary(i) = ~output_binary(i);
end

disp(output_binary,'The DPSK output is');


//
   //////////////////////////////////////////////////////////////////////

//For DEPSK

//From example Ex6_1, we have b when initial storage
    value is assumed to be 1
b = [1,1,0,1,1];

//Output y
y = 1;
for i = 2:5
    y(i) = bitxor(b(i),b(i-1));
end

disp(y,'The DEPSK output is');
```

Scilab code Exa 6.4 Bandwidth and Noise susceptibility

```
clc;
//page 365
//problem 6.4

```

```
//Given energy per bit Eb = 0.01
Eb = 0.01;

//Given fundamental frequency is fb = 8 KHz
fb = 8*10^3;

//No of symbols M = 16
M = 16;

N = log2(M);

BW_BPSK = 2*fb;
disp('Bandwidth for BPSK is '+string(BW_BPSK)+'Hz');

BW_QPSK = fb;
disp('Bandwidth for QPSK is '+string(BW_QPSK)+'Hz');

BW_16MPSK = fb/2;
disp('Bandwidth for 16 MPSK is '+string(BW_16MPSK)+'Hz');

BW_BFSK = 4*fb;
disp('Bandwidth for BFSK is '+string(BW_BFSK)+'Hz');

BW_MSK = 1.5*fb;
disp('Bandwidth for MSK is '+string(BW_MSK)+'Hz');

BW_16MFSK = 2*M*fb;
disp('Bandwidth for 16 MFSK is '+string(BW_16MFSK)+'Hz');


Min_dist_BPSK = 2*(Eb)^0.5;
disp('Minimum distance in signal space in BPSK is '+string(Min_dist_BPSK));

Min_dist_QPSK = 2*(Eb)^0.5;
disp('Minimum distance in signal space in QPSK is '+
```

```
    string(Min_dist_QPSK));

//The given answer in the textbook is 0.0152, which
    appears to be wrong. The correct answer is 0.078
Min_dist_16MPSK = (4*N*Eb*(sin(%pi/16))^2)^0.5;
disp('Minimum distance in signal space in 16 MPSK is
     '+string(Min_dist_16MPSK));

Min_dist_BFSK = (2*Eb)^0.5;
disp('Minimum distance in signal space in ortho BFSK
     is '+string(Min_dist_BFSK));

Min_dist_MSK = 2*(Eb)^0.5;
disp('Minimum distance in signal space in MSK is '+
    string(Min_dist_MSK));

Min_dist_16MFSK = (2*N*Eb)^0.5;
disp('Minimum distance in signal space in ortho 16
    MFSK is '+string(Min_dist_16MFSK));

disp('The best method that provides least noise
    susceptibility is 16 MFSK, then BPSK, then QPSK,
    then comes MSK, then orthogonal BFSK and finally
    16 MPSK')
```

Scilab code Exa 6.5 Duobinary decoding

```
clc;
//page 381
//problem 6.5

//Given input signal is d
d = [0,1,1,1,0,1,0,1,1];

//
```

```
////////////////////////////////////////////////////////////////////////
//The answers obtained here are different from the ones mentioned in the textbook.
//The given answers have been checked rigorously and have been found out to be true.

//When precoded

//Signal b is initially assumed to be 0
b = 0;

for i = 2:9
    b(i) = bitxor(b(i-1),d(i));
end

//Changing bit code to polar signal we get, 0 --> -1, 1 --> +1
for i = 1:9
    if b(i)==1 then
        bp(i) = 1;
    else
        bp(i) = -1;
    end
end

//Let initial value of Vd be 0
//Vd = 0;

for i = 2:9
    Vd(i) = bp(i) + bp(i-1);
end

//Converting polar signal to bit code we get, -2 --> 0, 0 --> 1, 2 --> 0
for i = 1:9
    if Vd(i)== -2 then
        da(i) = 0;
```

```
    elseif Vd(i)== 2 then
        da(i) = 0;
    else
        da(i) = 1;
    end
end

disp(da,'Decoded output when preceded is ')

//
   ////////////////////////////////////////////////////////////

//When not precoded exor gate is not there

//Changing bit code to polar signal we get , 0 -->
   -1, 1 --> +1
for i = 1:9
    if d(i)==1 then
        dp(i) = 1;
    else
        dp(i) = -1;
    end
end

for i = 2:9
    Vd(i) = dp(i) + dp(i-1);
end

//Converting polar signal to bit code we get , -2 -->
    0 , 0 --> 1 , 2 --> 1
for i = 2:9
    if Vd(i)== -2 then
        da(i) = 0;
    elseif Vd(i)== 2 then
        da(i) = 0;
    else
        da(i)= ~da(i-1);
```

```
    end
end

disp(da,'Decoded output when not precoded is ')
```

Scilab code Exa 6.6 Roll off factor

```
clc;
//page 381
//problem 6.6

//Given Bandwidth BW = 4 kHz
BW = 4*10^3;

//Given data rate is fb = 6 kbps
fb = 6*10^3;

//The roll off factor alpha is
alpha = ((2*BW)/fb) - 1;

disp('The roll off factor is '+string(alpha));

//////////////////////////////////////////////////////////////////////

//The required data rate supported at alpha = 0.25 is D
alpha = 0.25

//The corresponding expression for D is
D = (2*BW)/(1+alpha);

disp('The supported data rate is '+string(D)+' kbps');
```

```
//For full roll-off alpha = 1.0,
alpha = 1;

fb = 2*BW/(1+alpha);

disp('The data rate is '+string(fb)+' kbps');
```

Chapter 7

Mathematical Representation of Noise

Scilab code Exa 7.3 Noise Power

```
clc;
//page 413
//problem 7.3

//The resistance R = 1000 Ohm
R = 10^3;

//The capacitance C = 0.5*10^-6 F
C = 0.1*10^-6;

//Cutoff frequency for RC filter is f
f = 1/(2*%pi*R*C)

//White noise power spectral density n
n = 10^(-9);

//Noise power at filter output P
P = (%pi/2)*n*f;

```

```
disp('Noise power at output filter is '+string(P)+' Watt');

//Noise power at filter output P_new when cutoff frequency is doubled
P_new = (%pi/2)*n*2*f;

disp('Noise power at output filter when cutoff frequency is doubled is '+string(P_new)+' Watt');

//Ideal Low Pass filter Bandwidth B = 1000 Hz
B = 1000;

disp('Output Noise Power is '+string(n*B)+' Watt');

disp('Output Noise Power when cut-off frequency is doubled is '+string(2*n*B)+' Watt');

//Proportionality constant T = 0.01
T = 0.01;

//Output noise power O
O = n*(B^3)*(T^2)*(4/3)*(%pi)^2;

disp('Output Noise Power when signal is passed through a differentiator passed through ideal low pass filter '+string(O)+' Watt');

O_new = 8*n*(B^3)*(T^2)*(4/3)*(%pi)^2;

disp('Output Noise Power when signal is passed through a differentiator passed through ideal low pass filter and when cut-off frequency is doubled is '+string(O_new)+' Watt');
```

Scilab code Exa 7.4 SNR at output of equalizer

```
clc;
//page 413
//problem 7.4

//Given signal strength S = 0.001 W
S = 0.001;

//Gaussian Noise Magnitude n
n = 10^(-8);

//Frequency of signal f = 4000 Hz
F = 4000;

//Noise at equalizer output N
N = integrate('n*(1+(f^2)/F^2)','f',-F,F);

//Signal to Noise Ratio value is SNR
SNR = S/N;

disp('SNR value is '+string(10*log10(SNR))+' dB');
```

Chapter 8

Noise in Amplitude Modulation System

Scilab code Exa 8.1 SNR of SSB signal

```
clc;
//page 436
//problem 8.1

//Given frequency range fc= 1MHz to fc = 1.0005Mhz
//Single side message bandwidth is fM
fM= (1.0005 - 1)*10^6;
disp('Message bandwidth is '+string(fM)+' Hz');
//The textbook contains a calculation error here.
   The calculated fM in the textbook is 500kHz
   instead of 5kHz, following which all the
   solutions are erroneous

//Given input signal strength Si= 1mW
//Let output signal strength be So
//So=Si/4
Si= 10^(-3);
So= Si/4;
disp('Signal output strength is '+string(So)+' dB');
```

```

//Given Power Spectral Density n = 10^-9 W/Hz
//Let output noise strength be No
n= 10^-9;
No= (n*fM)/4;
disp('Output Noise Strength is '+string(No)+' dB');

//Let SNR at filter output be SNR
SNR= So / No;
disp('Output SNR is '+string(SNR)+' dB');

//By reduction of message signal Bandwidth the
   Output Noise strength changes
//Let the new output noise strength, bandwidth and
   SNR be be No_new, fM_new and SNR_new respectively
fM_new = 75/100*fM;
No_new = n*fM_new/4;
SNR_new = So / No_new;

disp('Changed SNR is '+string(SNR_new)+' dB');
```

Scilab code Exa 8.2 Signal strength and noise power density

```
clc;
//page 436
//problem 8.2

//Given frequency range fc - fm = 0.995MHz to fc +
   fm = 1.005Mhz
//Double side message bandwidth is fM
fM= (1.005 - 0.995)*10^6 / 2;
disp('Message bandwidth is '+string(fM)+' Hz');
//The textbook contains a calculation error here.
//The calculated fM in the textbook is 500kHz
   instead of 5kHz,
```

```
//Following which all the solutions obtained here are erroneous.

//Given input signal strength Si= 1mW
//Let output signal strength be So
//So=Si/2
Si= 10^(-3);
So= Si/2;
disp('Signal output strength is '+string(So)+' dB');

//Given Power Spectral Density n = 10^-9 W/Hz
//Let output noise strength be No
n= 10^-9;
No= (n*fM)/2;
disp('Output Noise Strength is '+string(No)+' dB');

//Let SNR at filter output be SNR
SNR= So / No;
disp('Output SNR of the DSB-SC wave is '+string(SNR)+' dB');

//By reduction of message signal Bandwidth the Output Noise strength changes
//Let the new output noise strength, bandwidth and SNR be be No_new, fM_new and SNR_new respectively
fM_new = 75/100*fM;
No_new = n*fM_new/4;
SNR_new = So / No_new;
disp('Changed SNR is '+string(SNR_new)+' dB');
```

Scilab code Exa 8.3 Minimum transmitter power

```
clc;
//page 446
//problem 8.3
```

```
//Given bandwidth of signal is fM = 4kHZ
fM = 4*10^3;
//Given power spectral density of white noise n =
    2*10^-9 W/Hz
n = 2*10^-9;
//Also given that minimum output SNR is 40dB
//Signal undergoes a loss of 30dB

//For SSB:
// Required minimum output SNR = Si_min_SSB / (n*fM)
     = 40 dB = 10^4
Si_min_SSB = (10^4)*n*fM;
// Required minimum signal strength at transmitter
    output Si_tran = Si_min * 30 dB
Si_tran_SSB = Si_min_SSB * 10^3;
disp('Required minimum SSB signal strength at
    transmitter output is'+string(Si_tran_SSB)+' W');

//For DSB-SC:
// Required minimum output SNR = (Si_min_DSB/3) / (n
    *fM) = 40 dB = 10^4
Si_min_DSB = 3*(10^4)*n*fM;
// Required minimum signal strength at transmitter
    output Si_tran = Si_min * 30 dB
Si_tran_DSB = Si_min_DSB * 10^3;
disp('Required minimum DSB signal strength at
    transmitter output is'+string(Si_tran_DSB)+' W');
```

Scilab code Exa 8.4 SNR of a Square Demodulator

```
clc;
//page 447
//problem 8.4

```

```
//Given bandwidth of signal is fM = 60 kHZ
fM = 60*10^3;

//Given power spectral density of white noise n = 2*10^-6 W/Hz
n = 2*10^-6;

//Given time average of square of mssg signal P = 0.1W
P = 0.1;

//Noise power at input baseband range NM
NM = n * fM;

//Threshold occurs at carrier power Pc = 2.9 * NM
Pc_Threshold = 2.9 * NM;

//For carrier power Pc = 10W, output SNR
Pc = 10;
SNRo = Pc * P / NM ;
disp('Output SNR is '+string(SNRo)+' dB');

//Carrier power is reduced by 100 times making the new power Pc_new
Pc_new = Pc / 100;

//In the given solutions the NM value is 1.2W instead of 0.12W
//The corect answer is 0.0925926 instead of 0.000926
SNR_new = (4/3) * P * (Pc_new/NM)^2;
disp('Output SNR when carrier power is reduced is '+string(SNR_new)+' dB');
```

Chapter 9

Noise in Frequency Modulation Systems

Scilab code Exa 9.1 SNR of an FM Limiter Discriminator Demodulator

```
clc;
//page 463
//problem 9.1

//Input signal strength Si = 0.5 W
Si = 0.5;

//Gaussian Power Spectral Density n = 10^(-10) W/Hz
n = 10^(-10);

//Baseband cutoff signal fM = 15 kHz
fM = 15 * 10^3;

//Maximum frequency deviation Df = 60 kHz
Df = 60 * 10^3;

//Average power of the modulating signal mt = 0.1 W
mt = 0.1;

```

```
SNR = (3/(4*%pi^2))*((Df/fM)^2)*mt^2*(Si/(n*fM));

disp('SNR is '+string(10*log10(SNR))+' dB');

//Part b

//Required SNR at output>40 dB = 10000

//From (a), required Si/0.5 > 10000/4052.8
//Or, required Si > 1.2337 W
//Since, channel loss is 20 dB (=100),
//Required transmitter power > 1.2337*100 = 123.37

disp('Required transmitter power > 1.2337 x 100 = 123.37 ');
```

Scilab code Exa 9.2 FM Limiter Discriminator Demodulator

```
clc;
//page 464
//problem 9.2

//Baseband cutoff signal fM = 15 kHz
fM = 15 * 10^3;

//Maximum frequency deviation Df = 60 kHz
Df = 60 * 10^3;

//Figure of Merit for FM is G_FM
G_FM = (3/2)*(Df/fM)^2;

disp('Figure of Merit for FM system is '+string(G_FM));

//Ratio of Figure of Merits of FM and AM systems is
```

```
R
R = G_FM/(1/3);

disp('Ratio of Figure of Merits for FM and AM systems is '+string(R));

Df_new = 2*Df;

//Figure of Merit for FM when bandwidth is doubled is G_FM_new
G_FM_new = (3/2)*(Df_new/fM)^2;

//Ratio of Figure of Merits of FM and AM systems when bandwidth is doubled is R_new
R_new = G_FM_new/(1/3);

disp('Ratio of Figure of Merits for FM and AM systems when bandwidth is doubled is '+string(R_new));
```

Scilab code Exa 9.3 RC Filter Preemphasis Deemphasis

```
clc;
//page 475
//problem 9.3

//Resistance R = 1000 Ohm
R = 10^3;

//Capacitance C = 0.1 * 10^-6 F
C = 0.1*10^-6;

//Break point for RC filter is f1
f1 = 1/(2*%pi*R*C)

```

```
//Baseband bandwidth of signal fM = 15 kHz
fM = 15 * 10^3;

Gain = atan(fM/f1)/(3*(f1/fM)*[1 - (f1/fM)*atan(fM/
    f1)]);

disp('Initial Gain is '+string(10*log10(Gain))+' dB'
    );

//New Baseband bandwidth of signal fM_new = 15 kHz
fM_new = 2*15 * 10^3;

Gain_new = atan(fM_new/f1)/(3*(f1/fM_new)*[1 - (f1/
    fM_new)*atan(fM_new/f1)]);

disp('Final Gain is '+string(10*log10(Gain_new))+'
    dB');
```

Scilab code Exa 9.6 SNR at Input

```
clc;
//page 495
//problem 9.6

//Baseband cutoff signal fM = 15 kHz
fM = 15 * 10^3;

//Carrier filter bandwidth is B = 60 kHz
B = 60 * 10^3;

//RMS frequency division Df_RMS = 30 kHz
Df_RMS = 30 * 10^3;

//Let a = Df_RMS/fM for substitution
a = Df_RMS/fM;
```

```

//Let b = fM/B for substitution
b = fM/B;

//Let input SNR 1 be I_SNR1 = 10 dB = 10
I_SNR1 = 10;

//Output SNR is O_SNR1
O_SNR1 = (3*(a^2)*I_SNR1)/(1+6*((2/%pi)^0.5)*I_SNR1*exp(-(b)*I_SNR1));

disp('Output SNR is '+string(10*log10(O_SNR1))+' dB');

//Let input SNR 2 be I_SNR2 = 20 dB = 100
I_SNR2 = 100;

//Output SNR is O_SNR2
O_SNR2 = (3*(a^2)*I_SNR2)/(1+6*((2/%pi)^0.5)*I_SNR2*exp(-(b)*I_SNR2));

//Solution given in the book is 13.5431 which is fallacious , the correct answer is 24.32444
disp('Output SNR is '+string(10*log10(O_SNR2))+' dB');

//Let input SNR 3 be I_SNR3 = 30 dB = 1000
I_SNR3 = 1000;

//Output SNR is O_SNR3
O_SNR3 = (3*(a^2)*I_SNR3)/(1+6*((2/%pi)^0.5)*I_SNR3*exp(-(b)*I_SNR3));

disp('Output SNR is '+string(10*log10(O_SNR3))+' dB');
```

Chapter 10

Phase Locked Loops

Scilab code Exa 10.3 SNR of Phase Discriminator

```
clc;
//page 520
//problem 10.3

// Part (a)

//Input SNR SNR_ip
SNR_ip = 1000;

//Beta B
B = 10;

//Output SNR SNR_op
SNR_op = (1.5*(B^2)*SNR_ip)/(1 + (12*B/%pi)*(SNR_ip)
    *exp(-0.5*(1/(B+1))*(SNR_ip)));

disp('Output SNR  is  '+string(10*log10(SNR_op))+' dB
    ');

// Part (b)

```

```
//Input SNR SNR_ip
SNR_ip = 10;

//Output SNR SNR_op
SNR_op = (1.5*(B^2)*SNR_ip)/(1 + (12*B/%pi)*(SNR_ip)*exp(-0.5*(1/(B+1))*(SNR_ip)));

disp('Output SNR  is '+string(10*log10(SNR_op))+' dB');
```

Scilab code Exa 10.5 Channel Spacing

```
clc;
//page 533
//problem 10.5

//Given reference frequency is fref = 10 MHz
fref = 10 * 10^6;

//Given step frequency is fstep = 100 KHz
fstep = 100 * 10^3;

//Division ratio M
M = fref/fstep;

//Required output frequency F = 100.6 MHz
F = 100.6 * 10^6;

N = F/fstep;

//Given P = 64
P = 64;

//Truncating value B = 15
B = 15;
```

```
A = N - P*B;

disp('The value of A is '+string(A));
disp('The value of B is '+string(B));
disp('The value of M is '+string(M));
```

Scilab code Exa 10.7 Phase Locked Loops

```
clc;
//page 534
//problem 10.7

//Given reference frequency for PLL is fref = 0.48
    MHz
fref = 0.48 * 10^6;

//Frequency divider N = 2000
N = 2000;

//Output Frequency fout
fout = fref*N;

//Output Frequency favg
favg = (2000*15 + 2001*1)*(0.48/16) * 10^6;

disp('Output frequency is '+string(fout)+' Hz');

//Reference frequency is not subdivided before going
    to comparator and it is an integer divider in
    the feedback path the frequency resolution fres =
    0.48 * 10^6;
fres = 0.48 * 10^6;

disp('Frequency resolution is '+string(fres)+' Hz');
```

```

disp('Output frequency resolution is '+string(favg -
    fout)+' Hz');
```

Chapter 11

Optimal Reception of Digital Signal

Scilab code Exa 11.3 Optimal threshold and probability of error

```
clc;
//page 558
//problem 11.3

//Prior probability of s1 P_s1 = 0.4
P_s1 = 0.4;

//Prior probability of s2 P_s2 = 1 - P_s1
P_s2 = 1 - P_s1;

//Voltage level V1 = 1
V1 = 1;

//Voltage level V2 = -1
V2 = -1;

//Part a

//Noise Variance sigma1 = 10^-3
```

```
sigma1 = 10^-3;

//Descision Threshold lambda1
lambda1 = (V1+V2)/2 + (sigma1)*log(P_s2/P_s1)/(V1-V2);

//Probability of error Pe
Pe = 0.5*(2*P_s1 - P_s1*erfc(((V2-V1)/(2*sigma1*2^0.5)) + (sigma1)*log(P_s2/P_s1)/((V1-V2)*2^0.5)));

disp('The decision threshold is '+string(lambda1)+' V');
disp('The probability of error is approximately '+string(Pe));

//Part b

//Noise Variance sigma2 = 10^-1
sigma2 = 10^-1;

//Descision Threshold lambda2
lambda2 = (V1+V2)/2 + (sigma2)*log(P_s2/P_s1)/(V1-V2);

//Probability of error Pe
Pe1 = 0.5*(2*P_s1 - P_s1*erfc(((V2-V1)/(2*sigma2*2^0.5)) + (sigma2)*log(P_s2/P_s1)/((V1-V2)*2^0.5)));

//In the textbook Pe has been calulated to be 0.0021 because of the use of a very high precision calculator, unfortunately in scilab the function erfc approximates the output value to a larger extent due to which an exact value cannot be obtained.

disp('The decision threshold is '+string(lambda2)+'
```

```
V');
disp('The probability of error is approximately '+string(Pe1));
```

Scilab code Exa 11.4 Decision threshold

```
clc;
//page 559
//problem 11.4

//Part b

//Voltage level V1 = 1
V1 = 1;

//Voltage level V2 = -1
V2 = -1;

//Prior probability of s1 P_s1 = 0.4
P_s1 = 0.4;

//Prior probability of s2 P_s2 = 1 - P_s1
P_s2 = 1 - P_s1;

//Cost of selecting s1 when s2 is transmitted C12 = 0.7
C12 = 0.7;

//Cost of selecting s2 when s1 is transmitted C21 = 1 - C12
C21 = 1 - C12;

//Noise Variance sigma = 10^-3
sigma = 10^-3;

```

```
//Descision Threshold lambda
lambda = (V1+V2)/2 + (sigma)*log((C12*P_s2)/(C21*
    P_s1))/(V1-V2);

disp('The decision threshold is '+string(lambda)+' V
    ');
```

Scilab code Exa 11.5 Probability of error of optimum filter

```
clc;
//page 567
//problem 11.5

//The voltage level of reciever is V = 5 mV
V = 5*10^-3;

//The time required to transfer one bit is T =
    1/9600 sec
T = 9600^-1;

//the signal energy of a bit be Es
Es = (V^2)*T;

//The power spectral density is n/2 = 10^-9 Watt/
    hertz
n = 2*10^-9;

//Probability error for optimal reciever is Pe
Pe = 0.5*erfc((Es/n)^0.5);

disp('The probability of error is '+string(Pe));

//When the data rate is doubled, the new effective
    energy per bit is Es_new
Es_new = (V^2)*(T/2);
```

```
//The new probability of error is Pe_new
Pe_new = 0.5*erfc((Es_new/n)^0.5);

//Percentage increase in error rate is P
P = 100*(Pe_new - Pe)/Pe;

disp('Percentage increase in error rate is '+string(P));

//Voltage required to restore probability levels, V_new
V_new = V*2^0.5;

disp('Voltage required to restore the probability levels is '+string(V_new)+' Volts');
```

Scilab code Exa 11.6 Error probability of BPSK signal

```
clc;
//page 575
//problem 11.6

//Amplitude of signal is A = 10 mV
A = 10*10^-3;

//Power Spectral Density n = 2 * 10^(-9) W/Hz
n = 2 * 10^(-9);

//Frequency is f = 1 MHz
f = 1*10^6;

//Data rate is D = 10^4 bps;
D = 10^4;

```

```
//Time taken for a bit to traverse
T = 1/D;

//Energy per signal element is Es
Es = A^2/(2*D);

//Probability of error Pe
Pe = 0.5*erfc((Es/n)^0.5);

disp('Probability of error is '+string(Pe));

//Phase shift phi = %pi/6
phi = %pi/6;

//Probability of error Pe_local_oscillator
Pe_local_oscillator = 0.5*erfc(((Es/n)^0.5)*cos(phi));

disp('Probability of error of local oscillator with phase shift is '+string(Pe_local_oscillator));

//Timing error t
t = 0.1*T;

//Probability of error when there is a synchronization fault Pe_timing_error
Pe_timing_error = 0.5*erfc(((Es/n)*(1 - 2*(t/T))^2)^0.5);

disp('Probability of error with synchronization fault is '+string(Pe_timing_error));

//Probability of error when both faults occur Pe_both
Pe_both = 0.5*erfc(((Es/n)*(cos(phi)^2)*(1 - 2*(t/T))^2)^0.5);

disp('Probability of error when both faults occur '+
```

```
    string(Pe_both));
```

Scilab code Exa 11.7 Error probability of coherent FSK signal

```
clc;
//page 575
//problem 11.7

//Amplitude of signal is A = 10 mV
A = 10*10^-3;

//Power Spectral Density n = 2 * 10^(-9) W/Hz
n = 2 * 10^(-9);

//Data rate is D = 10^4 bps;
D = 10^4;

//Time taken for a bit to traverse
T = 1/D;

//Energy per signal element is Es
Es = A^2/(2*D);

//Probability of error Pe_a
Pe_a = 0.5*erfc((0.6*Es/n)^0.5);
disp('Probability of error when offset is small is '
    +string(Pe_a));

//Probability of error Pe_b
Pe_b = 0.5*erfc((Es/(2*n))^0.5);
disp('Probability of error when frequencies used are
     orthogonal is '+string(Pe_b));

//Probability of error Pe_c
Pe_c = 0.5*exp(-(Es/(2*n)));
```

```
disp('Probability of error for non coherent
    detection is '+string(Pe_c));
```

Scilab code Exa 11.8 Error probability in Optimal Reception

```
clc;
//page 588
//problem 11.8

//Energy aasosciated with each bit Eb = 5 * 10^(-8)
    J
Eb = 5 * 10^(-8);

//Power Spectral Density n = 2 * 10^(-9) W/Hz
n = 2 * 10^(-9);

//No of symbols M
M = 16

//No of bits N
N = log2(M);

//Error limit for 16-PSK is P_16_PSK
P_16_PSK = erfc(((N*Eb*(%pi)^2)/(((M)^2)*n))^0.5);

disp('Upper limit of error probability of 16 PSK
    system is '+string(P_16_PSK));

//Error limit for 16-QASK is P_16_QASK
P_16_QASK = 2*erfc(((0.4*Eb)/(n))^0.5);

disp('Upper limit of error probability of 16 QASK
    system is '+string(P_16_QASK));

//Error limit for 16-FSK is P_16_FSK
```

```
P_16_FSK = ((2^4 - 1)/2)*erfc(((N*Eb)/(2*n))^0.5);

disp('Upper limit of error probability of 16 FSK
    system is '+string(P_16_FSK)+', negligibly small'
    );
```

Scilab code Exa 11.9 Probability of error of QPR System

```
clc;
//page 595
//problem 11.9

//Energy aasosciated with each bit Eb = 5 * 10^(-8)
    J
Eb = 5 * 10^(-8);

//Power Spectral Density n = 2 * 10^(-9) W/Hz
n = 2 * 10^(-9);

//Probability of error Pe
Pe = 0.5*erfc(((Eb*(%pi)^2)/(16*n))^0.5);

disp('Probability of error of QPR system is '+string
    (Pe));

//Given Bandwidth of channel is BW
BW = 10*10^3;

D = 2*BW;

disp('Data rate is '+string(D)+' bps');
```

Chapter 12

Noise in Pulse Code Modulation and Delta Modulation Systems

Scilab code Exa 12.2 SNR Optimal receiver

```
clc;
//page 608
//problem 12.2

//Baseband cutoff signal fM = 4 kHz
fM = 4 * 10^3;

//White noise power spectral density n
n = 2*10^(-9);

// Part (a)

//Input Signal energy Si = 0.001
Si_a = 0.001;

//No of levels used for PCM Coding M = 8
M_a = 8;
```

```

N_a = log2(M_a);

//Input SNR is SNR_ip
SNR_ip = Si_a/(n*fM);

//Output SNR is SNR_op
SNR_op = (2^(2*N_a))/(1 + (2^(2*N_a + 1))*erfc((SNR_ip*(1/(2*N_a))))^0.5);

disp('Input SNR for (a) is '+string(10*log10(SNR_ip))+' dB');
disp('Output SNR (a) is '+string(10*log10(SNR_op))+' dB');

// Part (b)

//Input Signal energy Si = 0.001
Si_b = 0.001;

//No of levels used for PCM Coding M = 256
M_b = 256;

N_b = log2(M_b);

//Input SNR is SNR_ip_b
SNR_ip_b = Si_b/(n*fM);

//Output SNR is SNR_op_b
SNR_op_b = (2^(2*N_b))/(1 + (2^(2*N_b + 1))*erfc((SNR_ip_b*(1/(2*N_b))))^0.5);

//Unfortunately in scilab the function erfc approximates the output value to a larger extent due to which an exact value cannot be obtained.
//The difference in the textbook answer and obatined answer is significant because of converting the answer into dB.
```

```
disp('Input SNR for (b) is '+string(10*log10(
    SNR_ip_b))+' dB');
disp('Output SNR for (b) is '+string(10*log10(
    SNR_op_b))+' dB');

// Part (c)

//Input Signal energy Si = 0.01
Si_c = 0.01;

//No of levels used for PCM Coding M = 256
M_c = 256;

N_c = log2(M_c);

//Input SNR is SNR_ip_c
SNR_ip_c = Si_c/(n*fM);

//Output SNR is SNR_op_c
SNR_op_c = (2^(2*N_c))/(1 + (2^(2*N_c + 1))*erfc((
    SNR_ip_c*(1/(2*N_c))))^0.5);

disp('Input SNR for (c) is '+string(10*log10(
    SNR_ip_c))+' dB');
disp('Output SNR for (c) is '+string(10*log10(
    SNR_op_c))+' dB');
```

Scilab code Exa 12.3 SNR of Optimal reciver

```
clc;
//page 609
//problem 12.3

//Baseband cutoff signal fM = 4 kHz
```

```
fM = 4 * 10^3;

//White noise power spectral density n
n = 2*10^(-9);

// Part (a)

//Input Signal energy Si = 0.001
Si = 0.001;

//No of levels used for PCM Coding M = 8
M = 8;

N = log2(M);

//Input SNR is SNR_ip
SNR_ip = Si/(n*fM);

//Output SNR is SNR_op
SNR_op = (2^(2*N))/(1 + (2^(2*N + 1))*erfc((SNR_ip
   *(3/(10*N))))^0.5);

disp('Input SNR for (a) is '+string(10*log10(SNR_ip)
   )+' dB');
disp('Output SNR (a) is '+string(10*log10(SNR_op))+'
    dB');

// Part (b)

//Input Signal energy Si = 0.001
Si = 0.001;

//No of levels used for PCM Coding M = 256
M_b = 256;

N_b = log2(M_b);

//Input SNR is SNR_ip_b
```

```
SNR_ip_b = Si/(n*fM);

//Output SNR is SNR_op_b
SNR_op_b = (2^(2*N_b))/(1 + (2^(2*N_b + 1))*erfc((SNR_ip_b*(3/(10*N_b))))^0.5);

//Unfortunately in scilab the function erfc approximates the output value to a larger extent due to which an exact value cannot be obtained.
//The difference in the textbook answer and obatined answer is significant because of converting the answer into dB.

disp('Input SNR for (b) is '+string(10*log10(SNR_ip_b))+' dB');
disp('Output SNR for (b) is '+string(10*log10(SNR_op_b))+' dB');

// Part (c)

//Input Signal energy Si = 0.01
Si = 0.01;

//No of levels used for PCM Coding M = 256
M = 256;

N = log2(M);

//Input SNR is SNR_ip_c
SNR_ip_c = Si/(n*fM);

//Output SNR is SNR_op_c
SNR_op_c = (2^(2*N))/(1 + (2^(2*N + 1))*erfc((SNR_ip_c*(3/(10*N))))^0.5);

disp('Input SNR for (c) is '+string(10*log10(SNR_ip_c))+' dB');
disp('Output SNR for (c) is '+string(10*log10(
```

```
    SNR_op_c))+' dB');
```

Scilab code Exa 12.4 Output SNR in DM including Thermal Noise

```
clc;
//page 618
//problem 12.4

//Upper cut off frequecny fb = 3200 Hz
fM = 3200;

//Lower cut off frequecny fl = 300 Hz
fl = 300;

//Data rate fb = 32000 bps
fb = 32000;

//White noise power spectral density n
n = 2*10^(-9);

//Input Signal energy Si = 0.001
Si = 0.001;

//Output SNR is SNR_op
SNR_op = (0.6*(fb/fM)^3)/(1 + (0.3*(fb^2/(fl*fM)))*
    erfc(Si/(n*fb)));

disp('Output SNR  is  '+string(10*log10(SNR_op))+'dB'
    );

//Data rate fb_n = 32000 bps
fb_n = 2*32000;

//Output SNR is SNR_op_n
SNR_op_n = (0.6*(fb_n/fM)^3)/(1 + (0.3*(fb_n^2/(fl*
```

```
    fM)))*erfc(Si/(n*fb_n)));

disp('Output SNR when data rate is doubled is '+
    string(10*log10(SNR_op_n))+'dB');
```

Chapter 13

Information Theory and Coding

Scilab code Exa 13.1 Information rate of source

```
clc;
//page 631
//problem 13.1

//Given probabilities p1 = p4 = 1/8 & p2 = p3 = 3/8
p1 = 1/8
p4 = 1/8
p2 = 3/8
p3 = 3/8

//The average information H is p1*log2 (1/p1)+p2*
    log2 (1/p2)+p3*log2 (1/p3)+p4*log2 (1/p4) bits/
    message
H = p1*log2 (1/p1)+p2*log2 (1/p2)+p3*log2 (1/p3)+p4*
    log2 (1/p4)

//information rate R is r*H bits/sec where r is 2*B
//R1 = R/B
R1 = 2*H
```

```

disp('The average information is '+string(H)+' bits/
    message')
disp('Information rate '+string(R1)+'*B bits/sec')
```

Scilab code Exa 13.4 Channel Capacity

```
clc;
//page 649
//problem 13.4

//Given bandwidth(B) = 4000Hertz & Noise PSD(n/2) =
    10^-9 Watt/Hertz
B = 4000
n = 2*10^-9

//Chanel capacity(C) = B*log2 (1+S/(n*B))

//case 1
//Signal energy(S) =  0.1Joule
S = 0.1

C = B*log2 (1+S/(n*B))

disp('Channel capacity for bandwidth = 4000Hertz,
    Noise PSD = 10^-9 Watt/Hertz & Signal energy(S) =
      0.1Joule is '+string(C)+' bits/sec')

//case 2
//Signal energy(S) =  0.001Joule
S = 0.001

C = B*log2 (1+S/(n*B))

disp('Channel capacity for bandwidth = 4000Hertz,
```

```
    Noise PSD = 10^-9 Watt/Hertz & Signal energy(S) =
      0.001 Joule is '+string(C)+' bits/sec')

//case 3
//Signal energy(S) =  0.001 Joule & incresed
    bandwidth(B) = 10000 Hertz
B = 10000
S = 0.001

C = B*log2 (1+S/(n*B))

disp('Channel capacity for bandwidth = 10000 Hertz,
    Noise PSD = 10^-9 Watt/Hertz & Signal energy(S) =
      0.001 Joule is '+string(C)+' bits/sec')
```

Scilab code Exa 13.8 Probability of error

```
clc;
//page 675
//problem 13.8

//With single parity bit added, the code size = 4.
    An error evades parity check if any 2 or all
    symbols ofthe code arrives are erroneous.
//Probability of any symbol from n are erroneous =
    nCm*(p^m)*(1-p)^(n-m)

//Thus, the probability of error undetected,
    P_undeterr = (4C2*(p^2)*(1-p)^2)+4C4*(p^4) = 6*(p
    ^2)*(1-p)^2)+(p^4)

//Probability of error in detection(P_undeterr1) for
     p = 0.1
p = 0.1
P_undeterr1 = 6*(p^2)*((1-p)^2)+(p^4)
```

```

disp('Probability of error in detection for p = 0.1 is '+string(P_undeterr1))

//Probability of error in detection(P_undeterr2) for p = 0.01
p = 0.01
P_undeterr2 = 6*(p^2)*((1-p)^2)+(p^4)

disp('Probability of error in detection for p = 0.01 is '+string(P_undeterr2))
```

Scilab code Exa 13.11 Turbo code

```
clc;
//page 696
//problem 13.11

//The output equations are as follows v1 = s1 xor s2 xor s3 & v2 = s1 xor s3
//the no of bits in output mode(bits_out) is v*(L+K), v = no of outputs for commutatot = 2, L = length of input = 3 & K = no of memeory elements = 3
v = 2
L = 3
K = 3
bits_out = v*(L+K)

//Taking in, s1, s2 , s3, v1 & v2 as row matrix where each column represents its corresponding input or output, in means input
in = [0 1 0 1 0 0 0]

s1 = zeros(1,7)
```

```
s2 = zeros(1,7)
s3 = zeros(1,7)
v1 = zeros(1,7)
v2 = zeros(1,7)


for i = 2:7
    s3(i) = s2(i-1)
    s2(i) = s1(i-1)
    s1(i) = in(i-1)
    v1(i-1) = bitxor(s1(i),bitxor(s2(i),s3(i)))
    v2(i-1) = bitxor(s1(i),s3(i))
end

//Output matrix is out
out = zeros(1,12)
for i = [1 3 5 7 9 11]
    out(i) = v1((i+3)/2)
    out(i+1) = v2((i+3)/2)
end

disp('output is')
disp(out)
```

Scilab code Exa 13.12 Turbo code

```
clc;
//page 697
//problem 13.12

//The qeneratr matrix requires impulse response of
    the coder.
//This is the ourput generated when the initially
    reset coder is fed with a single 1.
//The no of bits in the output code = 2(1+3) = 8
```

```

//Taking in , s1 , s2 , s3 , v1 & v2 as row matrix
   where each column represents its corresponding
   input or output , in means input
in = [0 1 0 0 0]

s1 = zeros(1,5)
s2 = zeros(1,5)
s3 = zeros(1,5)
v1 = zeros(1,5)
v2 = zeros(1,5)


for i = 2:5
    s3(i) = s2(i-1)
    s2(i) = s1(i-1)
    s1(i) = in(i-1)
    v1(i-1) = bitxor(s1(i),bitxor(s2(i),s3(i)))
    v2(i-1) = bitxor(s1(i),s3(i))
end

//Output matrix is out
out = zeros(1,8)
for i = [1 3 5 7]
    out(i) = v1((i+3)/2)
    out(i+1) = v2((i+3)/2)
end

disp('impulse response is')
disp(out)

//Then generator matrix is G
G = [1 1 1 0 1 1 0 0 0 0 0 0;0 0 1 1 1 0 1 1 0 0 0
   0;0 0 0 0 1 1 1 0 1 1 0 0]

//Note that , in G, impulse responses appear in
   staggered apper in a staggered manner in each row
    while the rest of the elements are 0.
```

```
//Now, output is b_o = b_i*G where input b_i =[1 0
    1]
b_i = [1 0 1]

b_o = b_i*G

//Here multiplication means Exor operation so
    whereever two occurs it should be changed to 1

for i = 1:12
        if  b_o(i) > 1 then
            b_o(i) = 0;
        end
end

disp('output is ')
disp(b_o)
disp('The output obtained is exactly same as example
    13.1')
```

Scilab code Exa 13.14 GO back N algorithm

```
clc;
//page 701
//problem 13.14

//Given, Tw = 10microsec, BCH(1023973) code is used
    implies k as 973 & n as 1023, P_A = 0.99, T1 = 40
    microsec & N = 4
Tw = 10*10^-6
k = 973
n = 1023
P_A = 0.99
T1 = 40*10^-6
```

```
N = 4

//efficiency of Stop-and-Wait ARQ(n_SandW) = (k/n)*(
    P_A/(1+(T1/Tw)))
n_SandW = (k/n)*(P_A/(1+(T1/Tw)))

//efficiency of Go-Back-N ARQ(n_GBN) = (k/n)*(1/(1+(
    N*(1-P_A)/P_A)))
n_GBN = (k/n)*(1/(1+(N*(1-P_A)/P_A)))

//efficiency of Selective Repeat ARQ(n_SR) = (k/n)*
    P_A
n_SR = (k/n)*P_A

disp('efficiency of Stop-and-Wait ARQ is '+string(
    n_SandW))
disp('efficiency of Go-Back-N ARQ is '+string(n_GBN)
    )
disp('efficiency of Selective Repeat ARQ is '+string
    (n_SR))
```

Scilab code Exa 13.15 Power of a Transmitter

```
clc;
//page 718
//problem 13.15

//Bit interval T = 1/10^6 = 10^-6 sec
T = 10^-6

//White Noise Power Spectral Density n/2 = 10^-9 W/
    Hz
n = 2*10^-9

//Power required Ps = Eb/T, where Eb = energy per
```

```
    bit

//For information system feedback system Eb = n
Ps = n/T

disp('power required for information system feedback
    system is '+string(Ps)+' Watt')

//For optimal system Ps = (0.69 * n)/T
Ps = (0.69 * n)/T;

disp('power required for optimal system is '+string(
    Ps)+' Watt')
```

Scilab code Exa 13.16 Probability of error for Trellis decoded modulation

```
clc;
//page 719
//problem 13.16

//Given, Eb = 10^-9Joule, n/2 = 10^-9 Watt/Hertz
Eb = 10^-8
n = 2*10^-9

//Probability of error for trellis-decoded
    modulation(Pe) = (1/2)*erfc(sqrt(1.5*Eb/n))
Pe = (1/2)*erfc(sqrt(1.5*Eb/n))

disp('Probability of error for trellis-decoded
    modulation is '+string(Pe))

//Probability of error for Qpsk modulation(Pe) =
    (1/2)*erfc(sqrt(Eb/n))
Pe = (1/2)*erfc(sqrt(Eb/n))

```

```
disp('Probability of error for Qpsk modulation is '+
   string(Pe))
```

Chapter 14

Communication Systems and Component Noises

Scilab code Exa 14.1 Thermal noise voltage

```
clc;
//page 738
//problem 14.1

//Boltzman constant  k = 1.3806488    10-23 m2 kg s -2 K-1
k = 1.3806488 * 10^-23;

//Let room temperature be 27 C
T = 27 + 273;

//Bandwidth BW = 10 MHz
BW = 10 * 10 ^6;

//For (a)
//Let the equivalent resistance be Ra
Ra = 10 + 10;

//RMS Noise Voltage be Va
```

```
Va = (4*k*T*Ra*BW)^0.5;

disp('The rms voltage at output a is '+string(Va)+' Volt');

//For (b)
//Let the equivalent resistance be Rb
Rb = (10 * 10)/(10+10);

//RMS Noise Voltage be Vb
Vb = (4*k*T*Rb*BW)^0.5;

disp('The rms voltage at output b is '+string(Vb)+' Volt');

//For (c)

Rc = 10;
C = 1*10^-9;

//In the textbook, the author has forgotten to multiply the result with T, hence has obtained an erroneous result.
//The given answer is 28.01uV but the correct answer is found out to be 1.2uV

Vc_square = 2*k*integrate('Rc/(1 + (2*%pi*Rc*C*f)^2)','f',-10^7,10^7);
Vc = Vc_square^0.5;

disp('The rms voltage at output c is '+string(Vc)+' Volt');
```

Scilab code Exa 14.2 Output Noise power

```
clc;
//page 741
//problem 14.2

//The Antenna noise temperature is T_ant = 10 K
T_ant = 10;

//The reciever noise temperature is Te = 140 K
Te = 140;

//Midband available gain of reciever gao = 10^10
gao = 10^10;

//Noise bandwidth is BN = 1.5 * 10^5 Hz
BN = 1.5 * 10^5;

//Boltzman constant  k = 1.3806488    10-23 m2 kg s
    -2 K-1
k = 1.3806488 * 10^-23;

//Available noise power at output is pao

pao = gao*k*(T_ant + Te)*BN;

disp('The available output noise power is '+string(
    pao)+' Watts');
```

Scilab code Exa 14.3 Transmitted power of antenna

```
clc;
//page 748
//problem 14.3

//The distance d = 30 * 1.6 * 10^3 m;
d = 30 * 1.6 * 10^3;
```

```
//Frequency f = 4 * 10^9 Hz
f = 4 * 10^9;

//Wavelength w = c/f m
w = 3*10^8 / f;

//Transmitter gain KT = 40 dB
KT = 10^4;

//Reciever gain KT = 40 dB
KR = 10^4;

//Reciever power PR = 10^-6 Watt
PR = 10^-6;

//Transmitter power PT
PT = PR*(4*%pi*d/w)^2/ (KT*KR);

disp('The transmitter output is '+string(PT)+' Watt'
  )
```

Chapter 15

Spread Spectrum Modulation

Scilab code Exa 15.1 Jamming

```
clc;
//page 764
//problem 15.1

//Signal Power data rate fb = 100 Kbps
fb = 10^5;

//Signal Strength Ps = 1 mW
Ps = 1*10^(-3);

//Chip frequency fs = 100 MHz
fs = 10^8;

//Noise Spectral Density n = 2*10^(-9) W/Hz
n = 2*10^(-9);

//Jamming Signal power is Pj = 1 W
Pj = 1;

//Processing Gain P
P = fs/fb;
```

```
disp('Processing Gain is '+string(P));

//Bit Interval T
T = 1/fb;
disp('Bit Interval is '+string(T)+'s');

//Energy per bit Eb
Eb = Ps*T;
disp('Energy per bit is '+string(Eb));

//Error Probability without jamming
   E_without_jamming
E_without_jamming = 0.5*erfc((Eb/(n))^0.5);
disp('Error probability without jamming is '+string(
   E_without_jamming));

//Error Probability with jamming E_jamming
E_jamming = 0.5*erfc(((2*Ps*P)/(Pj))^0.5);
disp('Error probability jamming is '+string(
   E_jamming));
```

Scilab code Exa 15.2 Ranging using DS spread spectrum

```
clc;
//page 764
//problem 15.2

//Chip Rate fc = 110 MHz
fc = 10*10^6;
Tc = 1/fc;

//Delay D = 0.1 ms
D = 0.1*10^-3;

//Speed of light c = 3*10^8 Kmps
```

```
c = 3*10^8;

//Estimated Distance d
d = 0.5*c*D;

//Tolerance Tol
Tol = 0.5*c*Tc;

disp('The target is between '+string(d-Tol)+' metres
    and '+string(d+Tol)+' metres of the source.');
```

Scilab code Exa 15.3 Sequence length

```
clc;
//page 769
//problem 15.3

//Number of Flip Flops N
N = 13;

//Maximal length of sequence L
L = 2^N - 1;

//Upper Bound S
S = (L - 1)/N;

//No of basic sequences and mirror images
disp('No of basic sequences and mirror images is '+
    string(S/2)');
```