% ofdm

clear;

clc;

%--------Simulation parameters----------------

nSym=10^4; %Number of OFDM Symbols to transmit

EbN0dB = -20:2:8; % bit to noise ratio

%-----------------------------------

%--------OFDM Parameters - Given in IEEE Spec--

N=64; %FFT size or total number of subcarriers (used + unused) 64

Nsd = 48; %Number of data subcarriers 48

Nsp = 4 ; %Number of pilot subcarriers 4

ofdmBW = 20 \* 10^6 ; % OFDM bandwidth

%----------------------------------------------

%--------Derived Parameters--------------------

deltaF = ofdmBW/N; %=20 MHz/64 = 0.3125 MHz

Tfft = 1/deltaF; % IFFT/FFT period = 3.2us

Tgi = Tfft/4;%Guard interval duration - duration of cyclic prefix

Tsignal = Tgi+Tfft; %duration of BPSK-OFDM symbol

Ncp = N\*Tgi/Tfft; %Number of symbols allocated to cyclic prefix

Nst = Nsd + Nsp; %Number of total used subcarriers

nBitsPerSym=Nst; %For BPSK the number of Bits per Symbol is same as num of subcarriers

%----------------------------------------------

EsN0dB = EbN0dB + 10\*log10(Nst/N) + 10\*log10(N/(Ncp+N)); % converting to symbol to noise ratio

errors= zeros(1,length(EsN0dB));

theoreticalBER = zeros(1,length(EsN0dB));

%Monte Carlo Simulation

for i=1:length(EsN0dB),

for j=1:nSym

%-----------------Transmitter--------------------

s=2\*round(rand(1,Nst))-1; %Generating Random Data with BPSK modulation

%IFFT block

%Assigning subcarriers from 1 to 26 (mapped to 1-26 of IFFT input)

%and -26 to -1 (mapped to 38 to 63 of IFFT input); Nulls from 27 to 37

%and at 0 position

X\_Freq=[zeros(1,1) s(1:Nst/2) zeros(1,11) s(Nst/2+1:end)];

% Pretending the data to be in frequency domain and converting to time domain

x\_Time=N/sqrt(Nst)\*ifft(X\_Freq);

%Adding Cyclic Prefix

ofdm\_signal=[x\_Time(N-Ncp+1:N) x\_Time];

%--------------Channel Modeling ----------------

noise=1/sqrt(2)\*(randn(1,length(ofdm\_signal))+1i\*randn(1,length(ofdm\_signal)));

r= sqrt((N+Ncp)/N)\*ofdm\_signal + 10^(-EsN0dB(i)/20)\*noise;

%-----------------Receiver----------------------

%Removing cyclic prefix

r\_Parallel=r(Ncp+1:(N+Ncp));

%FFT Block

r\_Time=sqrt(Nst)/N\*(fft(r\_Parallel));

%Extracting the data carriers from the FFT output

R\_Freq=r\_Time([(2:Nst/2+1) (Nst/2+13:Nst+12)]);

%BPSK demodulation / Constellation Demapper.Force +ve value --> 1, -ve value --> -1

R\_Freq(R\_Freq>0) = +1;

R\_Freq(R\_Freq<0) = -1;

s\_cap=R\_Freq;

numErrors = sum(abs(s\_cap-s)/2); %Count number of errors

%Accumulate bit errors for all symbols transmitted

errors(i)=errors(i)+numErrors;

end

theoreticalBER(i)=(1/2)\*erfc(sqrt(10.^(EbN0dB(i)/10))); %Same as BER for BPSK over AWGN

end

simulatedBER = errors/(nSym\*Nst);

plot(EbN0dB,log10(simulatedBER),'r-o');

hold on;

plot(EbN0dB,log10(theoreticalBER),'k\*');

grid on;

title('BER Vs EbNodB for OFDM with BPSK modulation over AWGN');

**xlabel('Eb/N0 (dB)');ylabel('BER');legend('simulated','theoretical');**

% BPSK

clear all;

close all;

clc;

fc=1000; % Frequency for "0" bits.

t=linspace(0,1/1000,50);

e0=cos(2\*pi\*fc\*t); % BPSK output for "1"

e1=-cos(2\*pi\*fc\*t); % BPSK output for "0”

b=mod(randperm(16),2);

bnot=1-b;

n=['The binary data is ', num2str(b)];

bpsk1=[]; bpsk2=[]; bin=[];

for i=1:length(b)

bpsk1=[bpsk1,b(i)\*e0];

bpsk2=[bpsk2,bnot(i)\*e1];

bin=[bin,b(i)\*ones(1,50)];

end

bpskout=bpsk1+bpsk2;

tm=[0:length(bpsk1)-1];

plot(tm,bin,'r--');

axis([0 length(bin) 0 1.5]);

hold on;

plot(tm,bpskout,'b');

axis([0 length(tm) -1.5 1.5]);

hold off;

xlabel('Time index');

ylabel('Amplitude');

legend('Random Binary', 'BPSK Output');

title('Simulation of Binary Phase Shift Keying');

gtext(n);

% Hata model

%Matlab code to simulate Hata-Okumura Models

clc;

clear all;

%----------Input Section---------------

Hbts= 50 ;%Height measured from the base of the BTS tower to the radiation centerline

Tbts = 350 ;%Terrain elevation at the location of the BTS

Htav= 300;%Height of the average terrain (from 3 Km to 15 km distance from the BTS)

Hm=3 ;%Height of the mobile antenna in meters

f=900 ;%100:100:3000; %Range of frequencies in MHz

d=0.5:0.5:15; %Range of Tx-Rx separation distances in Kilometers

Pt = 0.020; %Power transmitted by the BTS antenna in Watts

Gt= 10; %BTS antenna gain in dBi

%--------------------------------------

Hb=Hbts+Tbts-Htav ;%Effective Height of the BTS antenna in meters

%Cell array to store various model names

models = {'Big City (Urban model)';'Small & Medium City (Urban model)';'Sub-urban environment';'Open Rural environment'};

display('Hata-Okumura Model');

display(['1 ' models{1,1}]);

display(['2 ' models{2,1}]);

display(['3 ' models{3,1}]);

display(['4 ' models{4,1}]);

reply = input('Select Your choice of environment : ','s');

if 0<str2num(reply)<4

modelName = models{str2num(reply),1};

display(['Chosen Model : ' modelName])

else

error('Invalid Selection');

end

switch reply

case '1',

C=0;

if f<=200

aHm=8.29\*(log10(1.54\*Hm))^2-1.1;

else

aHm=3.2\*(log10(11.75\*Hm))^2-4.97;

end

case '2',

C=0;

aHm = (1.1\*log10(f)-0.7)\*Hm-(1.56\*log10(f)-0.8);

case '3',

aHm = (1.1\*log10(f)-0.7)\*Hm-(1.56\*log10(f)-0.8);

C=-2\*(log10(f/28))^2-5.4;

case '4',

aHm = (1.1\*log10(f)-0.7)\*Hm-(1.56\*log10(f)-0.8);

C=-4.78\*(log10(f))^2+18.33\*log10(f)-40.98;

otherwise ,

error('Invalid model selection');

end

A = 69.55 + 26.16\*log10(f) - 13.82\*log10(Hb)-aHm;

B = 44.9 - 6.55\*log10(Hb);

PL=A+B\*log10(d)+C;

subplot(2,1,1)

plot(d,PL,'r','LineWidth',2);

title(['Hata-Okumura Path Loss Model for : ' modelName]);

xlabel('Distance - Kilometers');

ylabel('Path Loss (dB)');

%Compute Received Signal Level

Pr = 10\*log10(Pt\*1000)+Gt-PL

subplot(2,1,2)

plot(d,Pr,'r','LineWidth',2);

title(['Hata-Okumura Model for : ' modelName]);

xlabel('Distance - Kilometers');

ylabel('Received Signal Level (dBm)');

% QPSK

% Demonstration of Eb/N0 Vs BER for QPSK modulation scheme

clear;

clc;

%---------Input Fields------------------------

N=1000000;%Number of input bits

EbN0dB = -4:2:10; % Eb/N0 range in dB for simulation

%---------------------------------------------

data=randn(1,N)>=0; %Generating a uniformly distributed random 1s and 0s

oddData = data(1:2:end);

evenData = data(2:2:end);

qpskModulated = sqrt(1/2)\*(1i\*(2\*oddData-1)+(2\*evenData-1)); %QPSK Mapping

M=4; %Number of Constellation points M=2^k for QPSK k=2

Rm=log2(M); %Rm=log2(M) for QPSK M=4

Rc=1; %Rc = code rate for a coded system. Since no coding is used Rc=1

BER = zeros(1,length(EbN0dB)); %Place holder for BER values for each Eb/N0

index=1;

for i=EbN0dB,

%-------------------------------------------

%Channel Noise for various Eb/N0

%-------------------------------------------

%Adding noise with variance according to the required Eb/N0

EbN0 = 10.^(i/10); %Converting Eb/N0 dB value to linear scale

noiseSigma = sqrt(1./(2\*Rm\*Rc\*EbN0)); %Standard deviation for AWGN Noise

%Creating a complex noise for adding with QPSK modulated signal

%Noise is complex since QPSK is in complex representation

noise = noiseSigma\*(randn(1,length(qpskModulated))+1i\*randn(1,length(qpskModulated)));

received = qpskModulated + noise;

%-------------------------------------------

%Threshold Detector

detected\_real = real(received)>=0;

detected\_img = imag(received)>=0;

estimatedBits=reshape([detected\_img;detected\_real],1,[]);

%------------------------------------------

%Bit Error rate Calculation

BER(index) = sum(xor(data,estimatedBits))/length(data);

index=index+1;

end

%Plot commands follows

plotHandle=plot(EbN0dB,log10(BER),'r--');

set(plotHandle,'LineWidth',1.5);

title('SNR per bit (Eb/N0) Vs BER Curve for QPSK Modulation Scheme');

xlabel('SNR per bit (Eb/N0) in dB');

ylabel('Bit Error Rate (BER) in dB');

grid on;

hold on;

theoreticalBER =0.5\*erfc(sqrt(10.^(EbN0dB/10)));

plotHandle=plot(EbN0dB,log10(theoreticalBER),'b\*');

set(plotHandle,'LineWidth',1.5);

legend('Simulated','Theoretical-QPSK','Theoretical-QPSK');

grid on;

% Loss Call System

clc

clear all

close all

n=input('Enter the number of trunks: ');

c=input('Enter the number of calls per hour: ');

th=input('Enter the holding duration of each call: ');

A=c\*th/60;

A=round(A);

fprintf('\n The value of offered traffic A=%d', A);

% Grade of service calculation for performance analysis

for N=1:1:n

num=power(A,N)/factorial(N);

fprintf('\n The value of Numerator of GOS is %d', num);

sum=0;

for k=1:1:n

den=power(A,k)/factorial(k);

sum=sum+den;

end

den=sum+1;

fprintf('\n The value of denominator pf GOS is %d', den);

B=num/den;

fprintf('\n GOS = %d', B);

grid on;

hold on;

plot(N,B, 'r+:')

end