**Final Project MATLAB Code**

**Physical Layer Simulation of an OFDM system**

**Group A Archit Garg (2021178730)**

**Nikhil Sangode (2021176404)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**AWGN Simulation for BPSK, QPSK and16-QAM**

%%% AWGN Simulation%%%

clc;

clear all;

close all;

% OFDM parameters for IEEE 802.11a

Nfft=64;

Nused=52;

Nref=4;

Ndata=48;

Nright=5;

Nleft =6;

Ndc=1;

Ncp=16;

Nnull=12;

Ppos=13;

%S for the AWGN Channel simulation

SNR=3:2:11;

%% --------------------BPSK--------------------%%

%%%---Steps involved for Transmitter:

%1)Generate random input sequence and map the signals

%2)Generate 64 samples with Nused=52,Nleft=6,Nright=5,Ndc=1

%3)Converting this 64 point samples from Serial to Parallel and taking IFFT

%4)Generate 16 cyclic prefix and append to 64 point making total 80 samples

%5)Add AWGN to the Sample

% -----This is the OFDM transmitted signal

l=1;

error=zeros(1,length(SNR));

for i=3:2:11

Tx=[];

init=[];

rx\_sym=[];

Count\_error=0;

for blocks=1:100

ip=randint(1,Ndata,2);

init = [init ip];

signal=pskmod(ip,2);

pilot=13:Ppos:Nused;

sym=setxor(1:Nused,pilot);

tx(sym)=signal;

tx(pilot)=1;

sym\_left=zeros(1,6);%Nleft=6

sym\_right=zeros(1,5);%Nright=5

n=length(tx)/2;

b=1:n;

c=(n+1):length(tx);

sym1=zeros(1,n);

sym2=zeros(1,n);

sym1=tx(b);

sym2=tx(c);

tx\_seq = zeros(1,64);

tx\_seq=[sym\_left sym1 Ndc sym2 sym\_right];

S\_to\_P=reshape(tx\_seq,64,1);%Serial to Parallel

tx\_sym\_IFFT=sqrt(64)\*ifft(S\_to\_P,64);

j=Nfft-Ncp;

CP=tx\_sym\_IFFT(j+1:64,1); % generate cyclic prefix %

tx\_IFFT\_CP=[CP

tx\_sym\_IFFT]; %appending CP %

P\_to\_S=reshape(tx\_IFFT\_CP,1,80);%Parallel to serial

Tx=[Tx P\_to\_S];

% adding noise %

N0 = 1/(10.^(i/10));

n = sqrt(N0)\*(randn(1,80)+(1i\*randn(1,80)))/sqrt(2);

P\_to\_S = P\_to\_S + n;

%%%---BPSK Receiver

%----Steps involved in Receiver structure

% 1)Removing 16 cyclic prefix Nleft,Nright and Ndc

% 2)Taking FFT of 64 point sample

% 3)Demodulate the signal and compare with the input transmitted

% signal and calculate the BER

ifft\_sy = P\_to\_S(Ncp+1:end);

sertopar\_sym\_rx = reshape(ifft\_sy,64,1);

rx\_fft = (sqrt(64))\*fft(sertopar\_sym\_rx,64);

PtoS\_rx = reshape(rx\_fft,1,Nfft);

rx\_Nleft = PtoS\_rx(Nleft+1:end);

diff = Nfft-Nright-Nleft;

rx\_Nright = rx\_Nleft(1:diff); %removing Nleft and Nright

y = rx\_Nright(1:(Nused/2)); %removing Ndc

z = rx\_Nright((Nused/2)+2:end);

rx = [y z];

%%%Signal demodulation

rx\_pilot = rx(sym);

demod=[];

op = pskdemod(rx\_pilot,2);

demod = [demod op];

Count\_error = Count\_error + biterr(op,ip);

end

error(l) = Count\_error/(Ndata\*100);

l = l+1;

end

figure(1);

semilogy(SNR,error,'-r^');

hold on;

grid on;

title('BER plots for modulation techniques with AWGN');

xlabel('SNR (dB)');

ylabel('Bit Error Rate');

%% ---------------------------QPSK------------------------------%%

l=1;

error=zeros(1,length(SNR));

M\_QPSK=4;

for i=3:2:11

Tx=[];

init=[];

rx\_sym=[];

Count\_error=0;

for blocks=1:100

ip=randint(1,Ndata,M\_QPSK);

init = [init ip];

signal=pskmod(ip,M\_QPSK);

pilot=13:Ppos:Nused;

sym=setxor(1:Nused,pilot);

tx(sym)=signal;

tx(pilot)=1;

sym\_left=zeros(1,Nleft);

sym\_right=zeros(1,Nright);

n=length(tx)/2;

b=1:n;

c=(n+1):length(tx);

sym1=zeros(1,n);

sym2=zeros(1,n);

sym1=tx(b);

sym2=tx(c);

tx\_seq = zeros(1,64);

tx\_seq=[sym\_left sym1 Ndc sym2 sym\_right];

S\_to\_P=reshape(tx\_seq,64,1);

tx\_sym\_IFFT=sqrt(Nfft)\*ifft(S\_to\_P,Nfft);

j=Nfft-Ncp;

CP=tx\_sym\_IFFT(j+1:Nfft,1); % generate cyclic prefix %

tx\_IFFT\_CP=[CP

tx\_sym\_IFFT]; %appending CP %

P\_to\_S=reshape(tx\_IFFT\_CP,1,80);

Tx=[Tx P\_to\_S];

% adding noise %

N0 = 1/(10.^(i/10));

n = sqrt(N0)\*(randn(1,length(tx\_IFFT\_CP))+(1i\*randn(1,length(tx\_IFFT\_CP))))/sqrt(2);

P\_to\_S = P\_to\_S + n;

%Receiver

ifft\_sy = P\_to\_S(Ncp+1:end);

sertopar\_sym\_rx = reshape(ifft\_sy,Nfft,1);

rx\_fft = (sqrt(Nfft))\*fft(sertopar\_sym\_rx,Nfft);

PtoS\_rx = reshape(rx\_fft,1,Nfft);

rx\_Nleft = PtoS\_rx(Nleft+1:end);

diff = Nfft-Nright-Nleft;

rx\_Nright = rx\_Nleft(1:diff); %removing Nleft and Nright

y = rx\_Nright(1:(Nused/2)); %Removing Ndc

z = rx\_Nright((Nused/2)+2:end);

rx = [y z];

rx\_pilot = rx(sym);

demod=[];

op = pskdemod(rx\_pilot,M\_QPSK);

demod = [demod op];

Count\_error = Count\_error + biterr(op,ip);

end

error(l) = Count\_error/(Ndata\*200);

l = l+1;

end

figure(1);

semilogy(SNR,error,'-b^');

hold on;

grid on;

title('BER plots for modulation techniques with AWGN');

xlabel('SNR (dB)');

ylabel('Bit Error Rate');

%% ------------------------ 16-QAM-------------------------------%%

l=1;

error=zeros(1,length(SNR));

M\_QAM=16;

for i=3:2:11

Tx=[];

init=[];

rx\_sym=[];

Count\_error=0;

for blocks=1:100

ip=randint(1,Ndata,M\_QAM);

init = [init ip];

signal=qammod(ip,M\_QAM)/sqrt(42);

pilot=13:Ppos:Nused;

sym=setxor(1:Nused,pilot);

tx(sym)=signal;

tx(pilot)=1;

sym\_left=zeros(1,Nleft);

sym\_right=zeros(1,Nright);

n=length(tx)/2;

b=1:n;

c=(n+1):length(tx);

sym1=zeros(1,n);

sym2=zeros(1,n);

sym1=tx(b);

sym2=tx(c);

tx\_seq = zeros(1,64);

tx\_seq=[sym\_left sym1 Ndc sym2 sym\_right];

S\_to\_P=reshape(tx\_seq,64,1);

tx\_sym\_IFFT=sqrt(Nfft)\*ifft(S\_to\_P,Nfft);

j=Nfft-Ncp;

CP=tx\_sym\_IFFT(j+1:Nfft,1); % generate cyclic prefix %

tx\_IFFT\_CP=[CP

tx\_sym\_IFFT]; %appending CP %

P\_to\_S=reshape(tx\_IFFT\_CP,1,80);

Tx=[Tx P\_to\_S];

% adding noise %

N0 = 1/(10.^(i/10));

n = sqrt(N0)\*(randn(1,80)+(1i\*randn(1,80)))/sqrt(2);

P\_to\_S = P\_to\_S + n;

%Receiver

ifft\_sy = P\_to\_S(Ncp+1:end);

sertopar\_sym\_rx = reshape(ifft\_sy,Nfft,1);

rx\_fft = (sqrt(Nfft))\*fft(sertopar\_sym\_rx,Nfft);

PtoS\_rx = reshape(rx\_fft,1,Nfft);

rx\_Nleft = PtoS\_rx(Nleft+1:end);

diff = Nfft-Nright-Nleft;

rx\_Nright = rx\_Nleft(1:diff); %removing Nleft and Nright

y = rx\_Nright(1:(Nused/2));

z = rx\_Nright((Nused/2)+2:end);

rx = [y z];

rx\_pilot = rx(sym);

rx\_pilot = rx\_pilot\*sqrt(42);

demod=[];

op = qamdemod(rx\_pilot,M\_QAM);

demod = [demod op];

Count\_error = Count\_error + biterr(op,ip);

end

error(l) = Count\_error/(Ndata\*400);

l = l+1;

end

figure(1);

semilogy(SNR,error,'-m^');

hold on;

grid on;

title('BER plots for modulation techniques with AWGN');

legend('BPSK','QPSK','16-QAM');

xlabel('SNR (dB)');

ylabel('Bit Error Rate');

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Multipath Channel Simulation – Vehicular Channel Model for BPSK,QPSK and 16-QAM**

clear all;

clc;

% Vehicular%

% OFDM parameters

Nfft=64;

Nused=52;

Nref=4;

Ndata=48;

Nright=5;

Nleft =6;

Ndc=1;

Ncp=16; %CP length

Nnull=12;

Ppos=13;

SNR=3:2:11;

%% --------------BPSK-----%%

l=1;

error=zeros(1,length(SNR));

M\_BPSK=2;

% channel parameters for 6-tap model

t1=1;

t2=30;

t3=70;

t4=110;

t5=170;

t6=250;

for i=3:2:11

Tx=[];

init=[];

rx\_sym=[];

Count\_error=0;

for blocks=1:80

ip=randint(1,Ndata,M\_BPSK);

modsignal=pskmod(ip,M\_BPSK);

pilot=13:Ppos:Nused;

sym=setxor(1:Nused,pilot);

tx\_sym(sym)=modsignal;

tx\_sym(pilot)=1;

Nleft\_sym=zeros(1,Nleft);

Nright\_sym=zeros(1,Nright);

n=length(tx\_sym)/2;

b=1:n;

c=(n+1):length(tx\_sym);

sym1=zeros(1,n);

sym2=zeros(1,n);

sym1=tx\_sym(b);

sym2=tx\_sym(c);

tx\_seq=[Nleft\_sym sym1 Ndc sym2 Nright\_sym];

sertopar\_sym=reshape(tx\_seq,length(tx\_seq),1);

tx\_sym\_IFFT=sqrt(Nfft)\*ifft(sertopar\_sym,Nfft);

j=Nfft-Ncp;

CP=tx\_sym\_IFFT(j+1:Nfft,1); % generate cyclic prefix %

tx\_IFFT\_CP=[CP

tx\_sym\_IFFT]; %appending CP %

partoser\_sym=reshape(tx\_IFFT\_CP,1,80);

Tx=[Tx partoser\_sym];

%Generating Channel model using Channel A

p1real=sqrt(1).\*randn(1,1)/sqrt(2);

p1imag=sqrt(1).\*randn(1,1)/sqrt(2);

p1=complex(p1real,p1imag);

p2real=sqrt(0.8).\*randn(1,1)/sqrt(2);

p2imag=sqrt(0.8).\*randn(1,1)/sqrt(2);

p2=complex(p2real,p2imag);

p3real=sqrt(0.125).\*randn(1,1)/sqrt(2);

p3imag=sqrt(0.125).\*randn(1,1)/sqrt(2);

p3=complex(p3real,p3imag);

p4real=sqrt(0.1).\*randn(1,1)/sqrt(2);

p4imag=sqrt(0.1).\*randn(1,1)/sqrt(2);

p4=complex(p4real,p4imag);

p5real=sqrt(0.031).\*randn(1,1)/sqrt(2);

p5imag=sqrt(0.031).\*randn(1,1)/sqrt(2);

p5=complex(p5real,p5imag);

p6real=sqrt(0.01).\*randn(1,1)/sqrt(2);

p6imag=sqrt(0.01).\*randn(1,1)/sqrt(2);

p6=complex(p6real,p6imag);

z=zeros(1,7);

z(t1)=p1;

z(t2)=p2;

z(t3)=p3;

z(t4)=p4;

z(t5)=p5;

z(t6)=p6;

z\_fft=(1/sqrt(Nfft))\*fft(z,Nfft);

tx\_sym\_freqchannel=conv(z,partoser\_sym);

d=tx\_sym\_freqchannel;

% adding noise %

tx\_sym\_freqchannel = tx\_sym\_freqchannel(1:Nfft+Ncp);

N0 = 1/(10.^(i/10));

n = sqrt(N0)\*(randn(1,length(tx\_IFFT\_CP))+(1i\*randn(1,length(tx\_IFFT\_CP))))/sqrt(2);

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannel + n;

%Receiver

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannelAWGN(Ncp + 1:end);

tx\_symfreqchanneleq = ifft(fft( tx\_sym\_freqchannelAWGN)./z\_fft);

sertopar\_sym\_rx = reshape(tx\_symfreqchanneleq,Nfft,1);

rx\_fft = (1/sqrt(Nfft))\*fft(sertopar\_sym\_rx,Nfft);

partoser\_sym\_rx = reshape(rx\_fft,1,Nfft);

rx\_Nleft = partoser\_sym\_rx(Nleft+1:end);

diff = Nfft-Nright-Nleft;

rx\_Nright = rx\_Nleft(1:diff); %removing Nleft and Nright

y = rx\_Nright(1:(Nused/2));

z = rx\_Nright((Nused/2)+2:end);

rx = [y z];

rx\_pilot = rx(sym);

demod=[];

op = pskdemod(rx\_pilot,M\_BPSK);

demod = [demod op];

Count\_error = Count\_error + biterr(op,ip);

end

error(l) = Count\_error/(Ndata\*1000);

l = l+1;

end

figure(1);

semilogy(SNR,error,'-r^');

hold on;

grid on;

%% QPSK %%%

l=1;

error=zeros(1,length(SNR));

M\_QPSK=4;

% channel parameters for 6-tap model

t1=1;

t2=30;

t3=70;

t4=110;

t5=170;

t6=250;

for i=3:2:11

Tx=[];

init=[];

rx\_sym=[];

Count\_error=0;

for blocks=1:1000

ip=randint(1,Ndata,M\_QPSK);

modsignal=pskmod(ip,M\_QPSK);

pilot=13:Ppos:Nused;

sym=setxor(1:Nused,pilot);

tx\_sym(sym)=modsignal;

tx\_sym(pilot)=1;

Nleft\_sym=zeros(1,Nleft);

Nright\_sym=zeros(1,Nright);

n=length(tx\_sym)/2;

b=1:n;

c=(n+1):length(tx\_sym);

sym1=zeros(1,n);

sym2=zeros(1,n);

sym1=tx\_sym(b);

sym2=tx\_sym(c);

tx\_seq=[Nleft\_sym sym1 Ndc sym2 Nright\_sym];

sertopar\_sym=reshape(tx\_seq,length(tx\_seq),1);

tx\_sym\_IFFT=sqrt(Nfft)\*ifft(sertopar\_sym,Nfft);

j=Nfft-Ncp;

CP=tx\_sym\_IFFT(j+1:Nfft,1); % generate cyclic prefix %

tx\_IFFT\_CP=[CP

tx\_sym\_IFFT]; %appending CP %

partoser\_sym=reshape(tx\_IFFT\_CP,1,length(tx\_IFFT\_CP));

Tx=[Tx partoser\_sym];

p1real=sqrt(1).\*randn(1,1)/sqrt(2);

p1imag=sqrt(1).\*randn(1,1)/sqrt(2);

p1=complex(p1real,p1imag);

p2real=sqrt(0.8).\*randn(1,1)/sqrt(2);

p2imag=sqrt(0.8).\*randn(1,1)/sqrt(2);

p2=complex(p2real,p2imag);

p3real=sqrt(0.125).\*randn(1,1)/sqrt(2);

p3imag=sqrt(0.125).\*randn(1,1)/sqrt(2);

p3=complex(p3real,p3imag);

p4real=sqrt(0.1).\*randn(1,1)/sqrt(2);

p4imag=sqrt(0.1).\*randn(1,1)/sqrt(2);

p4=complex(p4real,p4imag);

p5real=sqrt(0.031).\*randn(1,1)/sqrt(2);

p5imag=sqrt(0.031).\*randn(1,1)/sqrt(2);

p5=complex(p5real,p5imag);

p6real=sqrt(0.01).\*randn(1,1)/sqrt(2);

p6imag=sqrt(0.01).\*randn(1,1)/sqrt(2);

p6=complex(p6real,p6imag);

z=zeros(1,7);

z(t1)=p1;

z(t2)=p2;

z(t3)=p3;

z(t4)=p4;

z(t5)=p5;

z(t6)=p6;

z\_fft=(1/sqrt(Nfft))\*fft(z,Nfft);

tx\_sym\_freqchannel=conv(z,partoser\_sym);

d=tx\_sym\_freqchannel;

% adding noise %

tx\_sym\_freqchannel = tx\_sym\_freqchannel(1:Nfft+Ncp);

N0 = 1/(10.^(i/10));

n = sqrt(N0)\*(randn(1,length(tx\_IFFT\_CP))+(1i\*randn(1,length(tx\_IFFT\_CP))))/sqrt(2);

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannel + n;

%Receiver

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannelAWGN(Ncp + 1:end);

tx\_symfreqchanneleq = ifft(fft( tx\_sym\_freqchannelAWGN)./z\_fft);

sertopar\_sym\_rx = reshape(tx\_symfreqchanneleq,Nfft,1);

rx\_fft = (1/sqrt(Nfft))\*fft(sertopar\_sym\_rx,Nfft);

partoser\_sym\_rx = reshape(rx\_fft,1,Nfft);

rx\_Nleft = partoser\_sym\_rx(Nleft+1:end);

diff = Nfft-Nright-Nleft;

rx\_Nright = rx\_Nleft(1:diff); %removing Nleft and Nright

y = rx\_Nright(1:(Nused/2));

z = rx\_Nright((Nused/2)+2:end);

rx = [y z];

rx\_pilot = rx(sym);

demod=[];

op = pskdemod(rx\_pilot,M\_QPSK);

demod = [demod op];

Count\_error = Count\_error + biterr(op,ip);

end

error(l) = Count\_error/(Ndata\*2000);

l = l+1;

end

figure(1);

semilogy(SNR,error,'-b^');

grid on;

hold on;

legend('BPSK','QPSK');

%%%%%%%%%%%%%%16-QAM%%%%%%%%%%%%%%%%%%

l=1;

error=zeros(1,length(SNR));

M\_QAM=16;

% channel parameters for 6-tap model

t1=1;

t2=30;

t3=70;

t4=110;

t5=170;

t6=250;

for i=3:2:11

Tx=[];

init=[];

rx\_sym=[];

Count\_error=0;

for blocks=1:1000

ip=randint(1,Ndata,M\_QAM);

modsignal=qammod(ip,M\_QAM)/sqrt(10); %normalise by sqrt(10)

pilot=13:Ppos:Nused;

sym=setxor(1:Nused,pilot);

tx\_sym(sym)=modsignal;

tx\_sym(pilot)=1;

Nleft\_sym=zeros(1,Nleft);

Nright\_sym=zeros(1,Nright);

n=length(tx\_sym)/2;

b=1:n;

c=(n+1):length(tx\_sym);

sym1=zeros(1,n);

sym2=zeros(1,n);

sym1=tx\_sym(b);

sym2=tx\_sym(c);

tx\_seq=[Nleft\_sym sym1 Ndc sym2 Nright\_sym];

sertopar\_sym=reshape(tx\_seq,length(tx\_seq),1);

tx\_sym\_IFFT=sqrt(Nfft)\*ifft(sertopar\_sym,Nfft);

j=Nfft-Ncp;

CP=tx\_sym\_IFFT(j+1:Nfft,1); % generate cyclic prefix %

tx\_IFFT\_CP=[CP

tx\_sym\_IFFT]; %appending CP %

partoser\_sym=reshape(tx\_IFFT\_CP,1,length(tx\_IFFT\_CP));

Tx=[Tx partoser\_sym];

p1real=sqrt(1).\*randn(1,1)/sqrt(2);

p1imag=sqrt(1).\*randn(1,1)/sqrt(2);

p1=complex(p1real,p1imag);

p2real=sqrt(0.8).\*randn(1,1)/sqrt(2);

p2imag=sqrt(0.8).\*randn(1,1)/sqrt(2);

p2=complex(p2real,p2imag);

p3real=sqrt(0.125).\*randn(1,1)/sqrt(2);

p3imag=sqrt(0.125).\*randn(1,1)/sqrt(2);

p3=complex(p3real,p3imag);

p4real=sqrt(0.1).\*randn(1,1)/sqrt(2);

p4imag=sqrt(0.1).\*randn(1,1)/sqrt(2);

p4=complex(p4real,p4imag);

p5real=sqrt(0.031).\*randn(1,1)/sqrt(2);

p5imag=sqrt(0.031).\*randn(1,1)/sqrt(2);

p5=complex(p5real,p5imag);

p6real=sqrt(0.01).\*randn(1,1)/sqrt(2);

p6imag=sqrt(0.01).\*randn(1,1)/sqrt(2);

p6=complex(p6real,p6imag);

z=zeros(1,7);

z(t1)=p1;

z(t2)=p2;

z(t3)=p3;

z(t4)=p4;

z(t5)=p5;

z(t6)=p6;

z\_fft=(1/sqrt(Nfft))\*fft(z,Nfft);

tx\_sym\_freqchannel=conv(z,partoser\_sym);

d=tx\_sym\_freqchannel;

% adding noise %

tx\_sym\_freqchannel = tx\_sym\_freqchannel(1:Nfft+Ncp);

N0 = 1/(10.^(i/10));

n = sqrt(N0)\*(randn(1,length(tx\_IFFT\_CP))+(1i\*randn(1,length(tx\_IFFT\_CP))))/sqrt(2);

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannel + n;

%Receiver

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannelAWGN(Ncp + 1:end);

tx\_symfreqchanneleq = ifft(fft( tx\_sym\_freqchannelAWGN)./z\_fft);

sertopar\_sym\_rx = reshape(tx\_symfreqchanneleq,Nfft,1);

rx\_fft = (1/sqrt(Nfft))\*fft(sertopar\_sym\_rx,Nfft);

partoser\_sym\_rx = reshape(rx\_fft,1,Nfft);

rx\_Nleft = partoser\_sym\_rx(Nleft+1:end);

diff = Nfft-Nright-Nleft;

rx\_Nright = rx\_Nleft(1:diff); %removing Nleft and Nright

y = rx\_Nright(1:(Nused/2));

z = rx\_Nright((Nused/2)+2:end);

rx = [y z];

rx\_pilot = rx(sym);

rx\_pilot = rx\_pilot\*sqrt(10);

demod=[];

op = qamdemod(rx\_pilot,M\_QAM);

demod = [demod op];

Count\_error = Count\_error + biterr(op,ip);

end

error(l) = Count\_error/(Ndata\*4000);

l = l+1;

end

figure(1);

semilogy(SNR,error,'-g^');

grid on;

hold on;

legend('BPSK','QPSK','16 QAM');

title('BER Plot for Vehicular Multipath Channel Profile - Channel A');

xlabel('SNR (dB)');

ylabel('Bit Error Rate');

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Multipath Channel Simulation – Pedestrian Channel Model for BPSK,QPSK and 16-QAM**

clear all;

clc;

% Pedestrian %

% OFDM parameters

Nfft=64;

Nused=52;

Nref=4;

Ndata=48;

Nright=5;

Nleft =6;

Ndc=1;

Ncp=16; %CP length

Nnull=12;

Ppos=13;

SNR=3:2:11;

%% BPSK %%

l=1;

error=zeros(1,length(SNR));

M\_BPSK=2;

% channel parameters for 6-tap model

t1=1;

t2=110;

t3=190;

t4=410;

for i=3:2:11

Tx=[];

init=[];

rx\_sym=[];

Count\_error=0;

for blocks=1:80

ip=randint(1,Ndata,2);

modsignal=pskmod(ip,2);

pilot=13:Ppos:Nused;

sym=setxor(1:Nused,pilot);

tx\_sym(sym)=modsignal;

tx\_sym(pilot)=1;

Nleft\_sym=zeros(1,Nleft);

Nright\_sym=zeros(1,Nright);

n=length(tx\_sym)/2;

b=1:n;

c=(n+1):length(tx\_sym);

sym1=zeros(1,n);

sym2=zeros(1,n);

sym1=tx\_sym(b);

sym2=tx\_sym(c);

tx\_seq=[Nleft\_sym sym1 Ndc sym2 Nright\_sym];

S\_to\_P=reshape(tx\_seq,length(tx\_seq),1);

tx\_sym\_IFFT=sqrt(Nfft)\*ifft(S\_to\_P,Nfft);

j=Nfft-Ncp;

CP=tx\_sym\_IFFT(j+1:Nfft,1); % generate cyclic prefix %

tx\_IFFT\_CP=[CP

tx\_sym\_IFFT]; %appending CP %

P\_to\_S=reshape(tx\_IFFT\_CP,1,length(tx\_IFFT\_CP));

Tx=[Tx P\_to\_S];

p1real=sqrt(1).\*randn(1,1)/sqrt(2);

p1imag=sqrt(1).\*randn(1,1)/sqrt(2);

p1=complex(p1real,p1imag);

p2real=sqrt(0.107).\*randn(1,1)/sqrt(2);

p2imag=sqrt(0.107).\*randn(1,1)/sqrt(2);

p2=complex(p2real,p2imag);

p3real=sqrt(0.012).\*randn(1,1)/sqrt(2);

p3imag=sqrt(0.012).\*randn(1,1)/sqrt(2);

p3=complex(p3real,p3imag);

p4real=sqrt(0.00524).\*randn(1,1)/sqrt(2);

p4imag=sqrt(0.00524).\*randn(1,1)/sqrt(2);

p4=complex(p4real,p4imag);

z=zeros(1,7);

z(t1)=p1;

z(t2)=p2;

z(t3)=p3;

z(t4)=p4;

z\_fft=(1/sqrt(Nfft))\*fft(z,Nfft);

tx\_sym\_freqchannel=conv(z,P\_to\_S);

d=tx\_sym\_freqchannel;

% adding noise %

%%%Transmitted signal

tx\_sym\_freqchannel = tx\_sym\_freqchannel(1:Nfft+Ncp);

N0 = 1/(10.^(i/10));

n = sqrt(N0)\*(randn(1,length(tx\_IFFT\_CP))+(1i\*randn(1,length(tx\_IFFT\_CP))))/sqrt(2);

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannel + n;

%Receiver

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannelAWGN(Ncp + 1:end);

tx\_symfreqchanneleq = ifft(fft( tx\_sym\_freqchannelAWGN)./z\_fft);

sertopar\_sym\_rx = reshape(tx\_symfreqchanneleq,Nfft,1);

rx\_fft = (1/sqrt(Nfft))\*fft(sertopar\_sym\_rx,Nfft);

partoser\_sym\_rx = reshape(rx\_fft,1,Nfft);

rx\_Nleft = partoser\_sym\_rx(Nleft+1:end);

diff = Nfft-Nright-Nleft;

rx\_Nright = rx\_Nleft(1:diff); %removing Nleft and Nright

y = rx\_Nright(1:(Nused/2)); %removing Ndc

z = rx\_Nright((Nused/2)+2:end);

rx = [y z];

rx\_pilot = rx(sym);

op=[];

demod = pskdemod(rx\_pilot,M\_BPSK);

op = [op demod];

Count\_error = Count\_error + biterr(demod,ip);

end

error(l) = Count\_error/(Ndata\*1000);

l = l+1;

end

figure(1);

semilogy(SNR,error,'-r^');

hold on;

grid on;

%% QPSK %%

l=1;

error=zeros(1,length(SNR));

M\_QPSK=4;

% channel parameters for 6-tap model

t1=1;

t2=110;

t3=190;

t4=410;

for i=3:2:11

Tx=[];

init=[];

rx\_sym=[];

Count\_error=0;

for blocks=1:1000

ip=randint(1,Ndata,M\_QPSK);

modsignal=pskmod(ip,M\_QPSK);

pilot=13:Ppos:Nused;

sym=setxor(1:Nused,pilot);

tx\_sym(sym)=modsignal;

tx\_sym(pilot)=1;

Nleft\_sym=zeros(1,Nleft);

Nright\_sym=zeros(1,Nright);

n=length(tx\_sym)/2;

b=1:n;

c=(n+1):length(tx\_sym);

sym1=zeros(1,n);

sym2=zeros(1,n);

sym1=tx\_sym(b);

sym2=tx\_sym(c);

tx\_seq=[Nleft\_sym sym1 Ndc sym2 Nright\_sym];

S\_to\_P=reshape(tx\_seq,length(tx\_seq),1);

tx\_sym\_IFFT=sqrt(Nfft)\*ifft(S\_to\_P,Nfft);

j=Nfft-Ncp;

CP=tx\_sym\_IFFT(j+1:Nfft,1); % generate cyclic prefix %

tx\_IFFT\_CP=[CP

tx\_sym\_IFFT]; %appending CP %

P\_to\_S=reshape(tx\_IFFT\_CP,1,length(tx\_IFFT\_CP));

Tx=[Tx P\_to\_S];

p1real=sqrt(1).\*randn(1,1)/sqrt(2);

p1imag=sqrt(1).\*randn(1,1)/sqrt(2);

p1=complex(p1real,p1imag);

p2real=sqrt(0.107).\*randn(1,1)/sqrt(2);

p2imag=sqrt(0.107).\*randn(1,1)/sqrt(2);

p2=complex(p2real,p2imag);

p3real=sqrt(0.012).\*randn(1,1)/sqrt(2);

p3imag=sqrt(0.012).\*randn(1,1)/sqrt(2);

p3=complex(p3real,p3imag);

p4real=sqrt(0.00524).\*randn(1,1)/sqrt(2);

p4imag=sqrt(0.00524).\*randn(1,1)/sqrt(2);

p4=complex(p4real,p4imag);

z=zeros(1,7);

z(t1)=p1;

z(t2)=p2;

z(t3)=p3;

z(t4)=p4;

z\_fft=(1/sqrt(Nfft))\*fft(z,Nfft);

tx\_sym\_freqchannel=conv(z,P\_to\_S);

d=tx\_sym\_freqchannel;

% adding noise %

tx\_sym\_freqchannel = tx\_sym\_freqchannel(1:Nfft+Ncp);

N0 = 1/(10.^(i/10));

n = sqrt(N0)\*(randn(1,length(tx\_IFFT\_CP))+(1i\*randn(1,length(tx\_IFFT\_CP))))/sqrt(2);

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannel + n;

%Receiver

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannelAWGN(Ncp + 1:end);

tx\_symfreqchanneleq = ifft(fft( tx\_sym\_freqchannelAWGN)./z\_fft);

sertopar\_sym\_rx = reshape(tx\_symfreqchanneleq,Nfft,1);

rx\_fft = (1/sqrt(Nfft))\*fft(sertopar\_sym\_rx,Nfft);

partoser\_sym\_rx = reshape(rx\_fft,1,Nfft);

rx\_Nleft = partoser\_sym\_rx(Nleft+1:end);

diff = Nfft-Nright-Nleft;

rx\_Nright = rx\_Nleft(1:diff); %removing Nleft and Nright

y = rx\_Nright(1:(Nused/2));

z = rx\_Nright((Nused/2)+2:end);

rx = [y z];

rx\_pilot = rx(sym);

op=[];

demod = pskdemod(rx\_pilot,M\_QPSK);

op = [op demod];

Count\_error = Count\_error + biterr(demod,ip);

end

error(l) = Count\_error/(Ndata\*2000);

l = l+1;

end

figure(1);

semilogy(SNR,error,'-b^');

grid on;

hold on;

%%%%%%%%%%%%%%16-QAM%%%%%%%%%%%%%%%%%%

l=1;

error=zeros(1,length(SNR));

M\_QAM=16;

% channel parameters for 6-tap model

t1=1;

t2=110;

t3=190;

t4=410;

for i=3:2:11

Tx=[];

init=[];

rx\_sym=[];

Count\_error=0;

for blocks=1:1000

ip=randint(1,Ndata,M\_QAM);

modsignal=qammod(ip,M\_QAM)/sqrt(10); %normalise by sqrt(10)

pilot=13:Ppos:Nused;

sym=setxor(1:Nused,pilot);

tx\_sym(sym)=modsignal;

tx\_sym(pilot)=1;

Nleft\_sym=zeros(1,Nleft);

Nright\_sym=zeros(1,Nright);

n=length(tx\_sym)/2;

b=1:n;

c=(n+1):length(tx\_sym);

sym1=zeros(1,n);

sym2=zeros(1,n);

sym1=tx\_sym(b);

sym2=tx\_sym(c);

tx\_seq=[Nleft\_sym sym1 Ndc sym2 Nright\_sym];

S\_to\_P=reshape(tx\_seq,length(tx\_seq),1);

tx\_sym\_IFFT=sqrt(Nfft)\*ifft(S\_to\_P,Nfft);

j=Nfft-Ncp;

CP=tx\_sym\_IFFT(j+1:Nfft,1); % generate cyclic prefix %

tx\_IFFT\_CP=[CP

tx\_sym\_IFFT]; %appending CP %

P\_to\_S=reshape(tx\_IFFT\_CP,1,length(tx\_IFFT\_CP));

Tx=[Tx P\_to\_S];

p1real=sqrt(1).\*randn(1,1)/sqrt(2);

p1imag=sqrt(1).\*randn(1,1)/sqrt(2);

p1=complex(p1real,p1imag);

p2real=sqrt(0.107).\*randn(1,1)/sqrt(2);

p2imag=sqrt(0.107).\*randn(1,1)/sqrt(2);

p2=complex(p2real,p2imag);

p3real=sqrt(0.012).\*randn(1,1)/sqrt(2);

p3imag=sqrt(0.012).\*randn(1,1)/sqrt(2);

p3=complex(p3real,p3imag);

p4real=sqrt(0.00524).\*randn(1,1)/sqrt(2);

p4imag=sqrt(0.00524).\*randn(1,1)/sqrt(2);

p4=complex(p4real,p4imag);

z=zeros(1,7);

z(t1)=p1;

z(t2)=p2;

z(t3)=p3;

z(t4)=p4;

z\_fft=(1/sqrt(Nfft))\*fft(z,Nfft);

tx\_sym\_freqchannel=conv(z,P\_to\_S);

d=tx\_sym\_freqchannel;

% adding noise %

tx\_sym\_freqchannel = tx\_sym\_freqchannel(1:Nfft+Ncp);

N0 = 1/(10.^(i/10));

n = sqrt(N0)\*(randn(1,length(tx\_IFFT\_CP))+(1i\*randn(1,length(tx\_IFFT\_CP))))/sqrt(2);

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannel + n;

%Receiver

tx\_sym\_freqchannelAWGN = tx\_sym\_freqchannelAWGN(Ncp + 1:end);

tx\_symfreqchanneleq = ifft(fft( tx\_sym\_freqchannelAWGN)./z\_fft);

sertopar\_sym\_rx = reshape(tx\_symfreqchanneleq,Nfft,1);

rx\_fft = (1/sqrt(Nfft))\*fft(sertopar\_sym\_rx,Nfft);

partoser\_sym\_rx = reshape(rx\_fft,1,Nfft);

rx\_Nleft = partoser\_sym\_rx(Nleft+1:end);

diff = Nfft-Nright-Nleft;

rx\_Nright = rx\_Nleft(1:diff); %removing Nleft and Nright

y = rx\_Nright(1:(Nused/2));

z = rx\_Nright((Nused/2)+2:end);

rx = [y z];

rx\_pilot = rx(sym);

rx\_pilot = rx\_pilot\*sqrt(10);

op=[];

demod = qamdemod(rx\_pilot,M\_QAM);

op = [op demod];

Count\_error = Count\_error + biterr(demod,ip);

end

error(l) = Count\_error/(Ndata\*4000);

l = l+1;

end

figure(1);

semilogy(SNR,error,'-g^');

grid on;

hold on;

legend('BPSK','QPSK','16 QAM');

title('BER Plot for Pedestrian Multipath Channel Profile - Channel A');

xlabel('SNR in dB');

ylabel('BER');

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Adaptive Modulation Simulation - (Vehicular & Pedestrian)**

%%%%%%% Adaptive Modulation Simulation %%%%%%%%%%

clear all;

close all;

clc;

%Given Average SNR

SNR=10:1:30;

N=length(SNR);

N\_bits=zeros(1,length(SNR));

eff=zeros(1,length(SNR));

%No. of Symbols

n=100;

%% VEHICULAR %%

%Channel A (Vehicular) Parameters for a 6-tap model

r1\_r = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r1\_i = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r1 = complex(r1\_r,r1\_i);

r2\_r = sqrt(0.79).\*randn(n\*N,1)/sqrt(2);

r2\_i = sqrt(0.79).\*randn(n\*N,1)/sqrt(2);

r2 = complex(r2\_r,r2\_i);

r3\_r = sqrt(0.126).\*randn(n\*N,1)/sqrt(2);

r3\_i = sqrt(0.126).\*randn(n\*N,1)/sqrt(2);

r3 = complex(r3\_r,r3\_i);

r4\_r = sqrt(0.1).\*randn(n\*N,1)/sqrt(2);

r4\_i = sqrt(0.1).\*randn(n\*N,1)/sqrt(2);

r4 = complex(r4\_r,r4\_i);

r5\_r = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r5\_i = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r5 = complex(r5\_r,r5\_i);

r6\_r = sqrt(0.01).\*randn(n\*N,1)/sqrt(2);

r6\_i = sqrt(0.01).\*randn(n\*N,1)/sqrt(2);

r6 = complex(r6\_r,r6\_i);

%Delay

t1=1;

t2=30;

t3=70;

t4=110;

t5=170;

t6=250;

%%% Magnitude Response of the frequency selective channel

gamma=zeros(64,1);

for i=1:length(SNR)

for t=1:n

h=zeros(64,1);

h(t1)=r1(i\*t);

h(t2)=r2(i\*t);

h(t3)=r3(i\*t);

h(t4)=r4(i\*t);

h(t5)=r5(i\*t);

h(t6)=r6(i\*t);

%Frequency Response of the channel

h\_fft=abs(fft(h,64));

count=0;

No=1/(10^(SNR(i)/10));

%%%%% Calculating Spectral Efficiency

for k=9:1:56

%gamma=(|H(f)|^2)\*SNR

gamma(k)=10\*log10((h\_fft(k)^2)\*(1/No));

if ((gamma(k)>=7) && (gamma(k)<10)) %BPSK

count=count+1;

elseif ((gamma(k)>=10) && (gamma(k)<15.25)) %QPSK

count=count+2;

elseif ((gamma(k)>=15.25) && (gamma(k)<18)) %8-PSK

count=count+3;

elseif ((gamma(k)>=18) && (gamma(k)<23.5)) %16-QAM

count=count+4;

elseif (gamma(k)>=23.5) %64-QAM

count=count+6;

end

end

N\_bits(i)=count;

eff(i)=eff(i)+N\_bits(i);

end

eff(i)=eff(i)/n;

end

eff\_vehicular=eff/48;

plot(SNR,eff\_vehicular,'-r');

grid on; hold on;

%% PEDESTRIAN %%

%Channel A Parameters for a 6-tap model

r1\_r = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r1\_i = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r1 = complex(r1\_r,r1\_i);

r1\_r = sqrt(0.107).\*randn(n\*N,1)/sqrt(2);

r2\_i = sqrt(0.107).\*randn(n\*N,1)/sqrt(2);

r2 = complex(r1\_r,r2\_i);

r3\_r = sqrt(0.012).\*randn(n\*N,1)/sqrt(2);

r3\_i = sqrt(0.012).\*randn(n\*N,1)/sqrt(2);

r3 = complex(r3\_r,r3\_i);

r4\_r = sqrt(0.00525).\*randn(n\*N,1)/sqrt(2);

r4\_i = sqrt(0.00525).\*randn(n\*N,1)/sqrt(2);

r4 = complex(r4\_r,r4\_i);

t1=1;

t2=110;

t3=190;

t4=410;

%%% Magnitude Response of the frequency selective channel

gamma=zeros(64,1);

for i=1:1:length(SNR)

for t=1:1:n

h=zeros(64,1);

h(t1)=r1(i\*t);

h(t2)=r2(i\*t);

h(t3)=r3(i\*t);

h(t4)=r4(i\*t);

%Frequency Response of the channel

h\_fft=abs(fft(h,64));

count=0;

No=1/(10^(SNR(i)/10));

for k=9:1:56

gamma(k)=10\*log10((h\_fft(k)^2)\*(1/No));

if ((gamma(k)>=7) && (gamma(k)<10)) %BPSK

count=count+1;

elseif ((gamma(k)>=10) && (gamma(k)<15.25)) %QPSK

count=count+2;

elseif ((gamma(k)>=15.25) && (gamma(k)<18))

count=count+3;

elseif ((gamma(k)>=18) && (gamma(k)<23.5)) %16-QAM

count=count+4;

elseif (gamma(k)>=23.5) %64-QAM

count=count+6;

end

end

N\_bits(i)=count;

eff(i)=eff(i)+N\_bits(i);

end

eff(i)=eff(i)/n;

end

eff\_pedestrian=eff/48;

%%% Plot of Spectral Efficiency for Vehicular and Pedestrian Channel Profile

plot(SNR,eff\_pedestrian,'-b');

title('Adaptive modulation - Empirical Spectral eff for the channel models');

xlabel('SNR (dB)');

ylabel('Avg. Spectral eff');

legend('Vehicular','Pedestrian');

grid on; hold off;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**MIMO OFDM Simulation**

clear all;

close all;

clc;

SNR=10:1:30;

N=length(SNR);

N\_bits=zeros(1,length(SNR));

eff=zeros(1,length(SNR));

n=100;

%% VEHICULAR %%

% Generating 4 channels for 2\*2 MIMO system ---Channel Parameters for a 6-tap model

r11\_r = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r11\_i = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r11 = complex(r11\_r,r11\_i);

r12\_r = sqrt(0.79).\*randn(n\*N,1)/sqrt(2);%Channel 1

r12\_i = sqrt(0.79).\*randn(n\*N,1)/sqrt(2);

r12 = complex(r12\_r,r12\_i);

r13\_r = sqrt(0.126).\*randn(n\*N,1)/sqrt(2);

r13\_i = sqrt(0.126).\*randn(n\*N,1)/sqrt(2);

r13 = complex(r13\_r,r13\_i);

r14\_r = sqrt(0.1).\*randn(n\*N,1)/sqrt(2);

r14\_i = sqrt(0.1).\*randn(n\*N,1)/sqrt(2);

r14 = complex(r14\_r,r14\_i);

r15\_r = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r15\_i = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r15 = complex(r15\_r,r15\_i);

r16\_r = sqrt(0.01).\*randn(n\*N,1)/sqrt(2);

r16\_i = sqrt(0.01).\*randn(n\*N,1)/sqrt(2);

r16 = complex(r16\_r,r16\_i);

tau11=1;

tau12=30;

tau13=70;

tau14=110;

tau15=170;

tau16=250;

r21\_r = sqrt(1).\*randn(n\*N,1)/sqrt(2);%Channel 2

r21\_i = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r21 = complex(r21\_r,r21\_i);

r22\_r = sqrt(0.79).\*randn(n\*N,1)/sqrt(2);

r22\_i = sqrt(0.79).\*randn(n\*N,1)/sqrt(2);

r22 = complex(r22\_r,r22\_i);

r23\_r = sqrt(0.126).\*randn(n\*N,1)/sqrt(2);

r23\_i = sqrt(0.126).\*randn(n\*N,1)/sqrt(2);

r23 = complex(r23\_r,r23\_i);

r24\_r = sqrt(0.1).\*randn(n\*N,1)/sqrt(2);

r24\_i = sqrt(0.1).\*randn(n\*N,1)/sqrt(2);

r24 = complex(r24\_r,r24\_i);

r25\_r = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r25\_i = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r25 = complex(r25\_r,r25\_i);

r26\_r = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r26\_i = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r26 = complex(r26\_r,r26\_i);

tau21=1;

tau22=30;

tau23=70;

tau24=110;

tau25=170;

tau26=250;

r31\_r = sqrt(1).\*randn(n\*N,1)/sqrt(2); %Channel 3

r31\_i = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r31 = complex(r31\_r,r31\_i);

r32\_r = sqrt(0.79).\*randn(n\*N,1)/sqrt(2);

r32\_i = sqrt(0.79).\*randn(n\*N,1)/sqrt(2);

r32 = complex(r32\_r,r32\_i);

r33\_r = sqrt(0.126).\*randn(n\*N,1)/sqrt(2);

r33\_i = sqrt(0.126).\*randn(n\*N,1)/sqrt(2);

r33 = complex(r33\_r,r33\_i);

r34\_r = sqrt(0.1).\*randn(n\*N,1)/sqrt(2);

r34\_i = sqrt(0.1).\*randn(n\*N,1)/sqrt(2);

r34 = complex(r34\_r,r34\_i);

r35\_r = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r35\_i = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r35 = complex(r35\_r,r35\_i);

r36\_r = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r36\_i = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r36 = complex(r36\_r,r36\_i);

tau31=1;

tau32=30;

tau33=70;

tau34=110;

tau35=170;

tau36=250;

r41\_r = sqrt(1).\*randn(n\*N,1)/sqrt(2); % Channel 4

r41\_i = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r41 = complex(r41\_r,r41\_i);

r42\_r = sqrt(0.79).\*randn(n\*N,1)/sqrt(2);

r42\_i = sqrt(0.79).\*randn(n\*N,1)/sqrt(2);

r42 = complex(r42\_r,r42\_i);

r43\_r = sqrt(0.126).\*randn(n\*N,1)/sqrt(2);

r43\_i = sqrt(0.126).\*randn(n\*N,1)/sqrt(2);

r43 = complex(r43\_r,r43\_i);

r44\_r = sqrt(0.1).\*randn(n\*N,1)/sqrt(2);

r44\_i = sqrt(0.1).\*randn(n\*N,1)/sqrt(2);

r44 = complex(r44\_r,r44\_i);

r45\_r = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r45\_i = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r45 = complex(r45\_r,r45\_i);

r46\_r = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r46\_i = sqrt(0.032).\*randn(n\*N,1)/sqrt(2);

r46 = complex(r46\_r,r46\_i);

tau41=1;

tau42=30;

tau43=70;

tau44=110;

tau45=170;

tau46=250;

%%% Magnitude Response of the frequency selective channel

mag\_res=zeros(64,1);

for i=1:1:length(SNR)

for t=1:1:n

h11=zeros(64,1);

h12=zeros(64,1);

h21=zeros(64,1);

h22=zeros(64,1);

h11(tau11)=r11(i\*t);

h11(tau12)=r12(i\*t);

h11(tau13)=r13(i\*t);

h11(tau14)=r14(i\*t);

h11(tau15)=r15(i\*t);

h11(tau16)=r16(i\*t);

h12(tau21)=r21(i\*t);

h12(tau22)=r22(i\*t);

h12(tau23)=r23(i\*t);

h12(tau24)=r24(i\*t);

h12(tau25)=r25(i\*t);

h12(tau26)=r26(i\*t);

h21(tau31)=r31(i\*t);

h21(tau32)=r32(i\*t);

h21(tau33)=r33(i\*t);

h21(tau34)=r34(i\*t);

h21(tau35)=r35(i\*t);

h21(tau36)=r36(i\*t);

h22(tau41)=r41(i\*t);

h22(tau42)=r42(i\*t);

h22(tau43)=r43(i\*t);

h22(tau44)=r44(i\*t);

h22(tau45)=r45(i\*t);

h22(tau46)=r46(i\*t);

h11\_fft=abs(fft(h11,64)); %Frequency Response of the channel

h12\_fft=abs(fft(h12,64));

h21\_fft=abs(fft(h21,64));

h22\_fft=abs(fft(h22,64));

%%%Calculating svd

H=zeros(1,128);

j=0;

for k=1:64

H((1+(2\*j)):(2+(2\*j)))=svd([h11\_fft(k) h12\_fft(k);h21\_fft(k) h22\_fft(k)]);

j=j+1;

end

m=1;

Hmax=zeros(1,64);

for k=1:2:128

Hmax(m)= max(H(k:k+1));

m=m+1;

end

count=0;

No=1/(10^(SNR(i)/10));

for r=9:1:56

mag\_res(r)=10\*log10((Hmax(r)^2)\*(1/No));

if ((mag\_res(r)>=7) && (mag\_res(r)<10)) %BPSK

count=count+1;

elseif ((mag\_res(r)>=10) && (mag\_res(r)<15.25)) %QPSK

count=count+2;

elseif ((mag\_res(r)>=15.25) && (mag\_res(r)<18)) %8-PSK

count=count+3;

elseif ((mag\_res(r)>=18) && (mag\_res(r)<23.5)) %16-QAM

count=count+4;

elseif (mag\_res(r)>=23.5) %64-QAM

count=count+6;

end

end

N\_bits(i)=count;

eff(i)=eff(i)+N\_bits(i);

end

eff(i)=eff(i)/n;

end

b\_vehicular=eff/48;

plot(SNR,b\_vehicular,'-b\*');

grid on; hold on;

%% Pedestrian %%

%Channel Parameters for a 6-tap model

r11\_r = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r11\_i = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r11 = complex(r11\_r,r11\_i);

r21\_r = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r21\_i = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r21 = complex(r21\_r,r21\_i);

r31\_r = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r31\_i = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r31 = complex(r31\_r,r31\_i);

r41\_r = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r41\_i = sqrt(1).\*randn(n\*N,1)/sqrt(2);

r41 = complex(r41\_r,r41\_i);

r12\_r = sqrt(0.107).\*randn(n\*N,1)/sqrt(2);

r12\_i = sqrt(0.107).\*randn(n\*N,1)/sqrt(2);

r12 = complex(r12\_r,r12\_i);

r22\_r = sqrt(0.107).\*randn(n\*N,1)/sqrt(2);

r22\_i = sqrt(0.107).\*randn(n\*N,1)/sqrt(2);

r22 = complex(r22\_r,r22\_i);

r32\_r = sqrt(0.107).\*randn(n\*N,1)/sqrt(2);

r32\_i = sqrt(0.107).\*randn(n\*N,1)/sqrt(2);

r32 = complex(r32\_r,r32\_i);

r42\_r = sqrt(0.107).\*randn(n\*N,1)/sqrt(2);

r42\_i = sqrt(0.107).\*randn(n\*N,1)/sqrt(2);

r42 = complex(r42\_r,r42\_i);

r13\_r = sqrt(0.012).\*randn(n\*N,1)/sqrt(2);

r13\_i = sqrt(0.012).\*randn(n\*N,1)/sqrt(2);

r13 = complex(r13\_r,r13\_i);

r23\_r = sqrt(0.012).\*randn(n\*N,1)/sqrt(2);

r23\_i = sqrt(0.012).\*randn(n\*N,1)/sqrt(2);

r23 = complex(r23\_r,r23\_i);

r33\_r = sqrt(0.012).\*randn(n\*N,1)/sqrt(2);

r33\_i = sqrt(0.012).\*randn(n\*N,1)/sqrt(2);

r33 = complex(r33\_r,r33\_i);

r43\_r = sqrt(0.012).\*randn(n\*N,1)/sqrt(2);

r43\_i = sqrt(0.012).\*randn(n\*N,1)/sqrt(2);

r43 = complex(r43\_r,r43\_i);

r14\_r = sqrt(0.00525).\*randn(n\*N,1)/sqrt(2);

r14\_i = sqrt(0.00525).\*randn(n\*N,1)/sqrt(2);

r14 = complex(r14\_r,r14\_i);

r24\_r = sqrt(0.00525).\*randn(n\*N,1)/sqrt(2);

r24\_i = sqrt(0.00525).\*randn(n\*N,1)/sqrt(2);

r24 = complex(r24\_r,r24\_i);

r34\_r = sqrt(0.00525).\*randn(n\*N,1)/sqrt(2);

r34\_i = sqrt(0.00525).\*randn(n\*N,1)/sqrt(2);

r34 = complex(r34\_r,r34\_i);

r44\_r = sqrt(0.00525).\*randn(n\*N,1)/sqrt(2);

r44\_i = sqrt(0.00525).\*randn(n\*N,1)/sqrt(2);

r44 = complex(r44\_r,r44\_i);

tau11=1;

tau12=110;

tau13=190;

tau14=410;

tau21=1;

tau22=110;

tau23=190;

tau24=410;

tau31=1;

tau32=110;

tau33=190;

tau34=410;

tau41=1;

tau42=110;

tau43=190;

tau44=410;

%%% Magnitude Response of the frequency selective channel

mag\_res=zeros(64,1);

for i=1:1:length(SNR)

for t=1:1:n

h11=zeros(64,1);

h12=zeros(64,1);

h21=zeros(64,1);

h22=zeros(64,1);

h11(tau11)=r11(i\*t);

h11(tau12)=r12(i\*t);

h11(tau13)=r13(i\*t);

h11(tau14)=r14(i\*t);

h12(tau21)=r21(i\*t);

h12(tau22)=r22(i\*t);

h12(tau23)=r23(i\*t);

h12(tau24)=r24(i\*t);

h21(tau31)=r31(i\*t);

h21(tau32)=r32(i\*t);

h21(tau33)=r33(i\*t);

h21(tau34)=r34(i\*t);

h22(tau41)=r41(i\*t);

h22(tau42)=r42(i\*t);

h22(tau43)=r43(i\*t);

h22(tau44)=r44(i\*t);

h11\_fft=abs(fft(h11,64)); %Frequency Response of the channel

h12\_fft=abs(fft(h12,64));

h21\_fft=abs(fft(h21,64));

h22\_fft=abs(fft(h22,64));

%%%Calculating svd

H=zeros(1,128);

j=0;

for k=1:64

H((1+(2\*j)):(2+(2\*j)))=svd([h11\_fft(k) h12\_fft(k);h21\_fft(k) h22\_fft(k)]);

j=j+1;

end

m=1;

Hmax=zeros(1,64);

for k=1:2:128

Hmax(m)= max(H(k:k+1));

m=m+1;

end

count=0;

No=1/(10^(SNR(i)/10));

for r=9:1:56

mag\_res(r)=10\*log10((Hmax(r)^2)\*(1/No));

if ((mag\_res(r)>=7) && (mag\_res(r)<10)) %BPSK

count=count+1;

elseif ((mag\_res(r)>=10) && (mag\_res(r)<15.25)) %QPSK

count=count+2;

elseif ((mag\_res(r)>=15.25) && (mag\_res(r)<18))

count=count+3;

elseif ((mag\_res(r)>=18) && (mag\_res(r)<23.5)) %16-QAM

count=count+4;

elseif (mag\_res(r)>=23.5) %64-QAM

count=count+6;

end

end

N\_bits(i)=count;

eff(i)=eff(i)+N\_bits(i);

end

eff(i)=eff(i)/n;

end

b\_pedestrian=eff/48;

plot(SNR,b\_pedestrian,'-r^');

axis([10 30 2 6.4]);

title('Spectral Efficiency for 2\*2 MIMO OFDM system for different channel models');

xlabel('SNR (dB)');

ylabel('Spectral Eficiency');

legend('Vehicular','Pedestrian');

grid on; hold off;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*