load('C:\Bhaskar\Academics\Wireless Comm Lab\Project\I\_OFDM1.txt');

load('C:\Bhaskar\Academics\Wireless Comm Lab\Project\Q\_OFDM1.txt');

% load('C:\Users\bru170000\Downloads\input\_to\_decoder.mat');

I\_OFDM1 = transpose(I\_OFDM1);Q\_OFDM1 = transpose(Q\_OFDM1);

OFDM = I\_OFDM1 + 1i\*Q\_OFDM1;

%figure(2); stem(real(OFDM)); figure(3); stem(imag(OFDM));

%% Frame Synchronization

Fs = 20e6; % sampling frequency

Ts = 1/Fs; % sample duration

tt = 3.2e-6; % one half of long training time

nt = tt\*Fs; % number of half the training samples

Tpre = 16e-6; %time of Preamble field

Tsig = 4e-6; %time of SIGNAL field

Tsym = 4e-6;

Pcoarse = zeros(1,length(OFDM)-160+1);

Rd = zeros(1,length(OFDM)-160+1);

Mcoarse = zeros(1,length(OFDM)-160+1);

Lt = [1 1 -1 -1 1 1 -1 1 -1 1 1 1 1 1 1 -1 -1 1 1 -1 1 -1 1 1 1 1 0 1 -1 -1 1 1 -1 1 -1 1 -1 -1 -1 -1 -1 1 1 -1 -1 1 -1 1 -1 1 1 1 1];

LtFirstHalf = Lt(28:53);

LtSeconHalf = Lt(1:26);

LT = [LtFirstHalf LtSeconHalf];

pilot = [1,1,1,-1];

for d = 0:length(OFDM)-160

a = OFDM(d+1:d+144);

b = OFDM(d+17:d+160);

c = OFDM(d+1:d+160);

Pcoarse(d+1)=sum(conj(a).\*b);

Rd(d+1) = 1/160\*sum(abs(c).^2);

Mcoarse(d+1) = (abs(Pcoarse(d+1)).^2)./(Rd(d+1));

end

[~, ix] = max((Mcoarse));

ix = ix - 1;

[~, ix1] = max(abs(Pcoarse));

%figure(4); stem(abs(Mcoarse));

%title('Correlation Function','FontSize',20);

%% CFO Compensation

Ylong = sum(conj(OFDM(ix+192:ix+192+64-1)).\*(OFDM(ix+192+64:ix+192+64+64-1)));

deltaF = angle(Ylong)/(pi\*128);

deltaF = deltaF/Ts;

% deltaF = 0;

Y\_noCFO = OFDM(ix:end);

Y1=exp(-1\*sqrt(-1)\*2\*pi\*deltaF\*Ts\*(0:(length(OFDM)-1))) .\*OFDM;

YsFcorrect=Y1(ix:end);

%figure(5); plot(real(YsFcorrect),imag(YsFcorrect),'x');

%title('CFO Compensated Frame Synced Constellation','FontSize',14);

%% Signal Field Extraction and Manipulation

ndiscard = round(16.8e-6\*Fs);%Discarding ST LT and GI

signal = YsFcorrect(ndiscard+1:ndiscard+64);

signalf = (fft(signal)); %FFT of the received Signal

%figure(6); plot(real(signalf),imag(signalf),'x');

%title('Signal Field Constellation','FontSize',14);

%figure(7); stem(abs(signalf));

%title('Signal Field Amplitude Spectrum','FontSize',14);

signald = [signalf(2:27) signalf(39:64)]; %signal after discarding null subcarriers

%figure(8); stem(abs(signald));

%title('Signal Field Amplitude Spectrum Null Subcarriers Removed','FontSize',14);

%figure(9); plot(real(signald),imag(signald),'x');

%title('Signal Field Constellation Without Null Subcarriers','FontSize',14);

%% Channel Compensation

LT1Rx = fft(YsFcorrect(193:192+64));

LT2Rx = fft(YsFcorrect(193+64:192+64+64));

% LT1Rx = fft(OFDM(15662:15662+63));

% LT2Rx = fft(OFDM(15662+64:15662+64+63));

LT1Rx = [LT1Rx(2:27) LT1Rx(39:64)]; % Removing Null Subcarriers

LT2Rx = [LT2Rx(2:27) LT2Rx(39:64)]; % Removing Null Subcarriers

h1 = LT1Rx./LT;

h2 = LT2Rx./LT;

h = 0.5\*(h1+h2);

%hxx = h./abs(h);

%signaldCh = signald.\*(h./(abs(h)));

signaldCh = signald./(h);

% figure(10); plot(real(signaldCh), imag(signaldCh),'x');

% title('Channel Compensated Signal Constellation','FontSize',14);

%% Residual CFO Compensation

pilotRx = [signaldCh(32) signaldCh(46) signaldCh(7) signaldCh(21)]; %[-21 -7 7 21] received pilot sequence

comp\_symbols = pilotRx./[1, 1, 1, -1]; % Pilot symbols used is 1 1 1 -1 according to my finding. Not sure. Seems correct based on the sign of the received pilots

comp\_symbols = comp\_symbols./abs(comp\_symbols);

comp\_symbols\_avg = mean(comp\_symbols,2);

signaldRCFO = signaldCh\*conj(comp\_symbols\_avg);

%signaldRCFO = signaldRCFO./abs(signaldRCFO);

% figure(11); plot(real(signaldRCFO),imag(signaldRCFO),'x'); %axis([-1.1 1.2 -0.8 0.8]);

% title('Channel Compensated Phase Compensated Signal Constellation','FontSize',14);

%% Decoding Signal and Extracting Information

signaldRCFO(real(signaldRCFO)>0)= 1;

signaldRCFO(real(signaldRCFO)<0)= 0;

signalRxFinal = [signaldRCFO(27:52) signaldRCFO(1:26)];

prof = [-1 1 1 -1 1 1 -1 -1 -1 1 -1 -1 1 1 1 1 1 -1 -1 1 1 -1 -1 -1 1 1 1 1 1 -1 -1 -1 1 -1 1 1 -1 1 1 -1 -1 1 -1 1 -1 -1 -1 -1 1 -1 1 1];

prof(prof<0)=0;

%nErr = 0.5\*abs(sum(signalRxFinal-prof));

pilotThresholded = [signaldRCFO(32) signaldRCFO(46) signaldRCFO(7) signaldRCFO(21)]; %[-21 -7 7 21] received pilot sequence

%% Removal of Pilot and Deinterleaving

sig\_No\_pilots = [signalRxFinal(1:5) signalRxFinal(7:19) signalRxFinal(21:32) signalRxFinal(34:46) signalRxFinal(48:end)];

pilots\_final = [signalRxFinal(6) signalRxFinal(20) signalRxFinal(33) signalRxFinal(47)]; %after reshaping it as -N/2 DC N/2, the locations of the pilots are at the locations given

Ncbps\_signal = 48;

Nbpsc\_signal = 1;

Ndbps\_signal = 24;

j = 0:Ncbps\_signal-1;

s = max(Nbpsc\_signal/2,1);

i = s\*floor(j/s)+mod((j+floor(16\*j/Ncbps\_signal)),s);

k = 16\*i-(Ncbps\_signal-1)\*floor(16\*i/Ncbps\_signal);

deinterleaved(k+1)=sig\_No\_pilots(j+1);

%% Viterbi Decoding

%stem(deinterleaved);

trellis = poly2trellis(7, [133 171]);

decoded1 = vitdec(deinterleaved, trellis, 24, 'term', 'hard');

ratedec = bin2dec(num2str(decoded1(1:4)));

switch ratedec

case 3

rate = 54e6; %rate in mbps

modulation = '64-QAM';

coding\_rate = '3/4';

nbpsc = 6; %number of bits per OFDM subcarrier

ncbps = 288; %number of coded bits per symbol (ofdm block)

ndbps = 216; %number of data bits per symbol (ofdm block)

%ndbps = ncbps\*coding rate or 216 = 288\*3/4

case 1

rate = 48e6;

modulation = '64-QAM';

coding\_rate = '2/3';

nbpsc = 6;

ncbps = 288;

ndbps = 192;

case 11

rate = 36e6;

modulation = '16-QAM';

coding\_rate = '3/4';

nbpsc = 4;

ncbps = 192;

ndbps = 144;

case 9

rate = 24e6;

modulation = '16-QAM';

coding\_rate = '1/2';

nbpsc = 4;

ncbps = 192;

ndbps = 96;

case 7

rate = 18e6;

modulation = 'QPSK';

coding\_rate = '3/4';

nbpsc = 2;

ncbps = 96;

ndbps = 72;

case 5

rate = 12e6;

modulation = 'QPSK';

coding\_rate = '1/2';

nbpsc = 2;

ncbps = 96;

ndbps = 48;

case 15

rate = 9e6;

modulation = 'BPSK';

coding\_rate = '3/4';

nbpsc = 1;

ncbps = 48;

ndbps = 36;

case 13

rate = 6e6;

modulation = 'BPSK';

coding\_rate = '1/2';

nbpsc = 1;

ncbps = 48;

ndbps = 24;

otherwise

disp('Theres something wrong my friend');

end

reserved\_bit = decoded1(5);

length\_octets = bin2dec(num2str(decoded1(17:-1:6)));

parity\_bit = decoded1(18);

tail = decoded1(19:24);

%% DATA Blocks Extraction

Nsym = ceil((16+8\*length\_octets+6)/ndbps); %number of ofdm data blocks

Ndata = Nsym\*ndbps; %number of data bits in the data field

Npad = Ndata - (16+8\*length\_octets+6); %number of pad bits

Txtime = Tpre + Tsig + Tsym\*Nsym; %Nsym = ceil((16+8\*length\_octets+6)/ndbps)

%Txtime = Tpre + Tsig + Tsym\*ceil((16+8\*length\_octets+6)/ndbps);

TxSamples = floor(Txtime\*Fs);

OFDMdata = YsFcorrect(400+1:TxSamples);

%nOFDMblocks = length(OFDMdata)/80;

dataF = transpose(reshape(OFDMdata,80,Nsym));

dataF = dataF(1:end,17:end); % Removing Guard Intervals From Each Data Block

for jj = 1:size(dataF,1)

dataF(jj,:) = fft(dataF(jj,:));

end

dataF = [dataF(:,2:27) dataF(:,39:64)]; %signal after discarding null subcarriers

%figure; stem(abs(dataF1(1,:)));

hData = repmat(h,Nsym,1); % Just repeating the channel matrix Nsym times

dataFch = dataF./(hData);

% figure(10);

plot(real(dataFch(:,:)), imag(dataFch(:,:)),'x');

% title('Channel Compensated Signal Constellation','FontSize',14);

pilot\_polarity = [1,1,1,1, -1,-1,-1,1, -1,-1,-1,-1, 1,1,-1,1, -1,-1,1,1, -1,1,1,-1, 1,1,1,1, 1,1,-1,1,...

1,1,-1,1, 1,-1,-1,1, 1,1,-1,1, -1,-1,-1,1, -1,1,-1,-1, 1,-1,-1,1, 1,1,1,1, -1,-1,1,1,...

-1,-1,1,-1, 1,-1,1,1, -1,-1,-1,1, 1,-1,-1,-1, -1,1,-1,-1, 1,-1,1,1, 1,1,-1,1, -1,1,-1,1,...

-1,-1,-1,-1, -1,1,-1,1, 1,-1,1,-1, 1,1,1,-1, -1,1,-1,-1, -1,1,1,1, -1,-1,-1,-1, -1,-1,-1];

pilotRx = [dataFch(:,32) dataFch(:,46) dataFch(:,7) dataFch(:,21)]; %[-21 -7 7 21] received pilot sequence

pilotTx = repmat(pilot,Nsym,1).\*repmat(transpose(pilot\_polarity(2:Nsym+1)),1,4); %The first 1 is already used for the signal so we start the index from 2

comp\_symbols = pilotRx./pilotTx; % Repetitive pilots with changed polarity

comp\_symbols = comp\_symbols./abs(comp\_symbols);

comp\_symbols\_avg = mean(comp\_symbols,2);

dataFchRc = dataFch.\*repmat(conj(comp\_symbols\_avg),1,52); % Data symbols channel compensated and phase compensated

% signaldRCFO = signaldRCFO./abs(signaldRCFO);

figure;plot(real(dataFchRc),imag(dataFchRc),'x'); %axis([-1.1 1.2 -0.8 0.8]);

title('Channel Compensated Phase Compensated Signal Constellation With BPSK Pilots','FontSize',14);

OFDMsymMP = [dataFchRc(:,27:52) dataFchRc(:,1:26)]; %The Modulated Symbols arranged as -Nc/2 to Nc/2

OFDMsymNoPilots = [OFDMsymMP(:,1:5) OFDMsymMP(:,7:19) OFDMsymMP(:,21:32) OFDMsymMP(:,34:46) OFDMsymMP(:,48:end)];

OFDMsymNoPilots = OFDMsymNoPilots\*sqrt(42);

pilots\_finalOFDM = [OFDMsymMP(:,6) OFDMsymMP(:,20) OFDMsymMP(:,33) OFDMsymMP(:,47)]; %after reshaping it as -N/2 DC N/2, the locations of the pilots are at the locations given

figure;plot(real(OFDMsymNoPilots),imag(OFDMsymNoPilots),'x'); %axis([-1.1 1.2 -0.8 0.8]);

grid on;

title('Channel Compensated Phase Compensated Signal Constellation Without BPSK Pilots','FontSize',14);

%% 64 QAM Demodulation

%z = qamdemod(OFDMsymNoPilots,64,wlanSymMap,'UnitAveragePower',true);

OFDMsymNoPilotsR = real(OFDMsymNoPilots);

OFDMsymNoPilotsI = imag(OFDMsymNoPilots);

OFDMsymNoPilotsR1 = zeros(size(OFDMsymNoPilotsR));

OFDMsymNoPilotsR1(OFDMsymNoPilotsR<-6) = 0;%000;

OFDMsymNoPilotsR1(OFDMsymNoPilotsR>-6 & OFDMsymNoPilotsR<-4) = 1;%001;

OFDMsymNoPilotsR1(OFDMsymNoPilotsR>-4 & OFDMsymNoPilotsR<-2) = 3;%011;

OFDMsymNoPilotsR1(OFDMsymNoPilotsR>-2 & OFDMsymNoPilotsR<0) = 2;%010;

OFDMsymNoPilotsR1(OFDMsymNoPilotsR>0 & OFDMsymNoPilotsR<2) = 6;%110;

OFDMsymNoPilotsR1(OFDMsymNoPilotsR>2 & OFDMsymNoPilotsR<4) = 7;%111;

OFDMsymNoPilotsR1(OFDMsymNoPilotsR>4 & OFDMsymNoPilotsR<6) = 5;%101;

OFDMsymNoPilotsR1(OFDMsymNoPilotsR>6 & OFDMsymNoPilotsR<10) = 4;%100;

OFDMsymNoPilotsR1 = (dec2bin(OFDMsymNoPilotsR1'));

OFDMsymNoPilotsR2 = zeros(size(OFDMsymNoPilotsI));

OFDMsymNoPilotsR2(OFDMsymNoPilotsI<-6) = 0;%000;

OFDMsymNoPilotsR2(OFDMsymNoPilotsI>-6 & OFDMsymNoPilotsI<-4) = 1;%001;

OFDMsymNoPilotsR2(OFDMsymNoPilotsI>-4 & OFDMsymNoPilotsI<-2) = 3;%011;

OFDMsymNoPilotsR2(OFDMsymNoPilotsI>-2 & OFDMsymNoPilotsI<0) = 2;%010;

OFDMsymNoPilotsR2(OFDMsymNoPilotsI>0 & OFDMsymNoPilotsI<2) = 6;%110;

OFDMsymNoPilotsR2(OFDMsymNoPilotsI>2 & OFDMsymNoPilotsI<4) = 7;%111;

OFDMsymNoPilotsR2(OFDMsymNoPilotsI>4 & OFDMsymNoPilotsI<6) = 5;%101;

OFDMsymNoPilotsR2(OFDMsymNoPilotsI>6 & OFDMsymNoPilotsI<10) = 4;%100;

OFDMsymNoPilotsR2 = (dec2bin(OFDMsymNoPilotsR2'));

bitsdem = strcat(OFDMsymNoPilotsR1,OFDMsymNoPilotsR2);

bitsdem1 = zeros(size(bitsdem));

for jj = 1:size(bitsdem1,1)

for kk = 1:size(bitsdem,2)

bitsdem1(jj,kk) = str2double(bitsdem(jj,kk));

end

end

interleavedbits = transpose(reshape(transpose(bitsdem1),ncbps,Nsym));

%% Deinterleaving the bits

j = 0:ncbps-1;

s = max(nbpsc/2,1);

i = s\*floor(j/s)+mod((j+floor(16\*j/ncbps)),s);

k = 16\*i-(ncbps-1)\*floor(16\*i/ncbps);

deinterleavedbits(:,k+1)=interleavedbits(:,j+1);

%% Depuncturing and Decoding

scramb1 = transpose(deinterleavedbits);

scramb2 = transpose(scramb1(:));

punctpat = [1 1 1 0 0 1];

trellis = poly2trellis(7, [133 171]);

scrambledbits = vitdec(scramb2, trellis, 35, 'term', 'hard', punctpat);

% scrambledbits = scrambledbits(1:length(scrambledbits)-Npad); %Removing the zero padded bits in the end

% a = reshape(transpose(scrambledbits),216,58);

%% Descrambling

init = [0 0 0 0 1 0 1]; %Check descrambleCheck.m this seed will give the first 7 bits as the data

initPerm = init;

scramSeq = zeros(1,127);

for ii = 1:127

scramSeq(ii) = xor(init(1),init(4));

init(1:6) = init(2:7);

init(7) = scramSeq(ii);

end

% scramSeq = [0 1 1 0 1 1 0 0 0 0 0 1 1 0 0 1 1 0 1 0 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 0 1 0 0 1 0 1 0 0 0 1 1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 1 0 1 1 1 0 1];

cyclicSeq = repmat(scramSeq,1,ceil(length(scrambledbits)/length(scramSeq)));

unscrambledbits = xor(cyclicSeq(1:length(scrambledbits)),scrambledbits);

%% Information Extraction

serviceBits = unscrambledbits(1:16);

padBits = unscrambledbits(length(unscrambledbits)-Npad+1:end);

tailBitsUnscrambled = unscrambledbits(length(unscrambledbits)-Npad-5:length(unscrambledbits)-Npad);

unscrNoSerNoPadTail = unscrambledbits(17:length(unscrambledbits)-Npad-6);

protocol = unscrNoSerNoPadTail(2:-1:1);

type = unscrNoSerNoPadTail(4:-1:3);

subtype = unscrNoSerNoPadTail(8:-1:5);