Homework 7 - Wireless Systems

-By Siddhesh Deodhar

src :

-mqam_over_rayleigh.m : This code is used to draw the graph of BER vs SNR for 4 QAM

- add_awgn_noise.m : This function is a subroutine for adding noise

-rayleighFading.m :   This is subroutine of Rayleigh test function which simulates Rayleigh fading.

-testRayleighFading.m : This code is used to used to plot Rayleigh envelope fading and calculate the average number of crossing and average fading time

- mpam_modulator.m : This subroutine is for generating 4 QAM modulation

-iqOptDetector.m : this is subroutine for detector

-minEuclideanDistance.m : This is subroutine for finding minimum euclidean distance

-constructQAM.m : This is subroutine for constructing QAM

Doc :

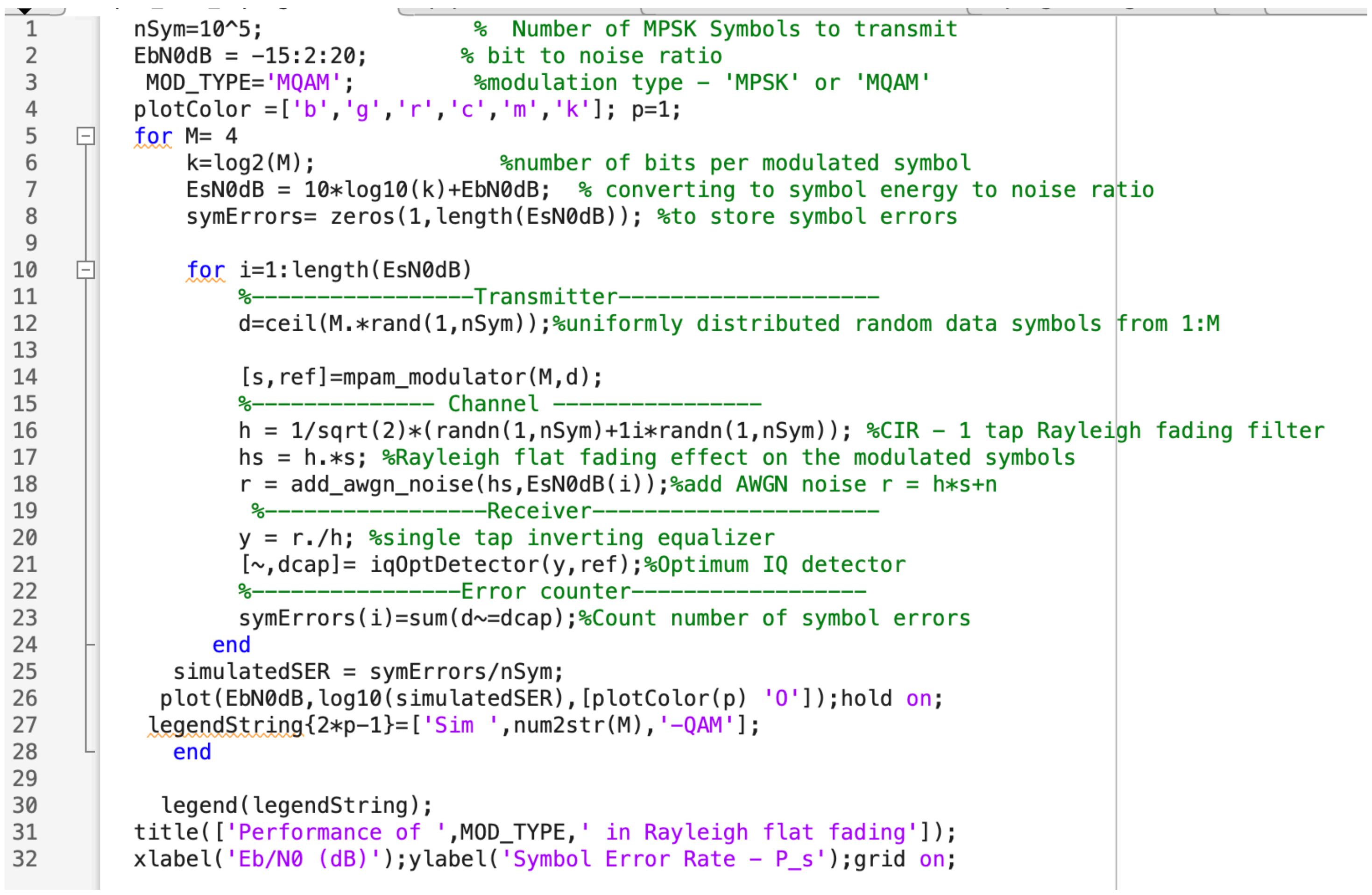
envelope.jpg : This is graph of Rayleigh envelope

BER_SNR.jpg : This is graph of BER vs SNR

crossings1.png: This is screenshot which show average number of crossings and average envelope fading duration.

crossings2.png: This is screenshot which show average number of crossings and average envelope fading duration.

Code explanation :



```
nSym=10^5;                  %  Number of MPSK Symbols to transmit
EbN0dB = -15:2:20;         % bit to noise ratio
 MOD_TYPE='MQAM';           %modulation type - 'MPSK' or 'MQAM'
plotColor =['b','g','r','c','m','k']; p=1;
for M= 4
    k=log2(M);                  %number of bits per modulated symbol
    EsN0dB = 10*log10(k)+EbN0dB;  % converting to symbol energy to noise ratio
    symErrors= zeros(1,length(EsN0dB)); %to store symbol errors

    for i=1:length(EsN0dB)
        %------------------Transmitter--------------------
        d=ceil(M.*rand(1,nSym));%uniformly distributed random data symbols from 1:M

        [s,ref]=mpam_modulator(M,d);
        %--------------- Channel -----------------
        h = 1/sqrt(2)*(randn(1,nSym)+1i*randn(1,nSym)); %CIR - 1 tap Rayleigh fading filter
        hs = h.*s; %Rayleigh flat fading effect on the modulated symbols
        r = add_awgn_noise(hs,EsN0dB(i));%add AWGN noise r = h*s+n
         %------------------Receiver----------------------
        y = r./h; %single tap inverting equalizer
        [~,dcap]= iqOptDetector(y,ref);%Optimum IQ detector
        %-----------------Error counter------------------
        symErrors(i)=sum(d~=dcap);%Count number of symbol errors
      end
     simulatedSER = symErrors/nSym;
    plot(EbN0dB,log10(simulatedSER),[plotColor(p) 'O']);hold on;
   legendString{2*p-1}=['Sim ',num2str(M),'-QAM'];
     end

    legend(legendString);
  title(['Performance of ',MOD_TYPE,' in Rayleigh flat fading']);
  xlabel('Eb/N0 (dB)');ylabel('Symbol Error Rate - P_s');grid on;
```

Comments indicate what each line or sections do.

First we initialize number of symbols to transmit in line 1. Then we initialize bit to noise ratio.

The for loop in lines 5 and 10 are for generating 4 Quadrature Amplitude Modulation (QAM).

In line 5 for loop we set the number of iterations as 4 because we want 4 QAM.

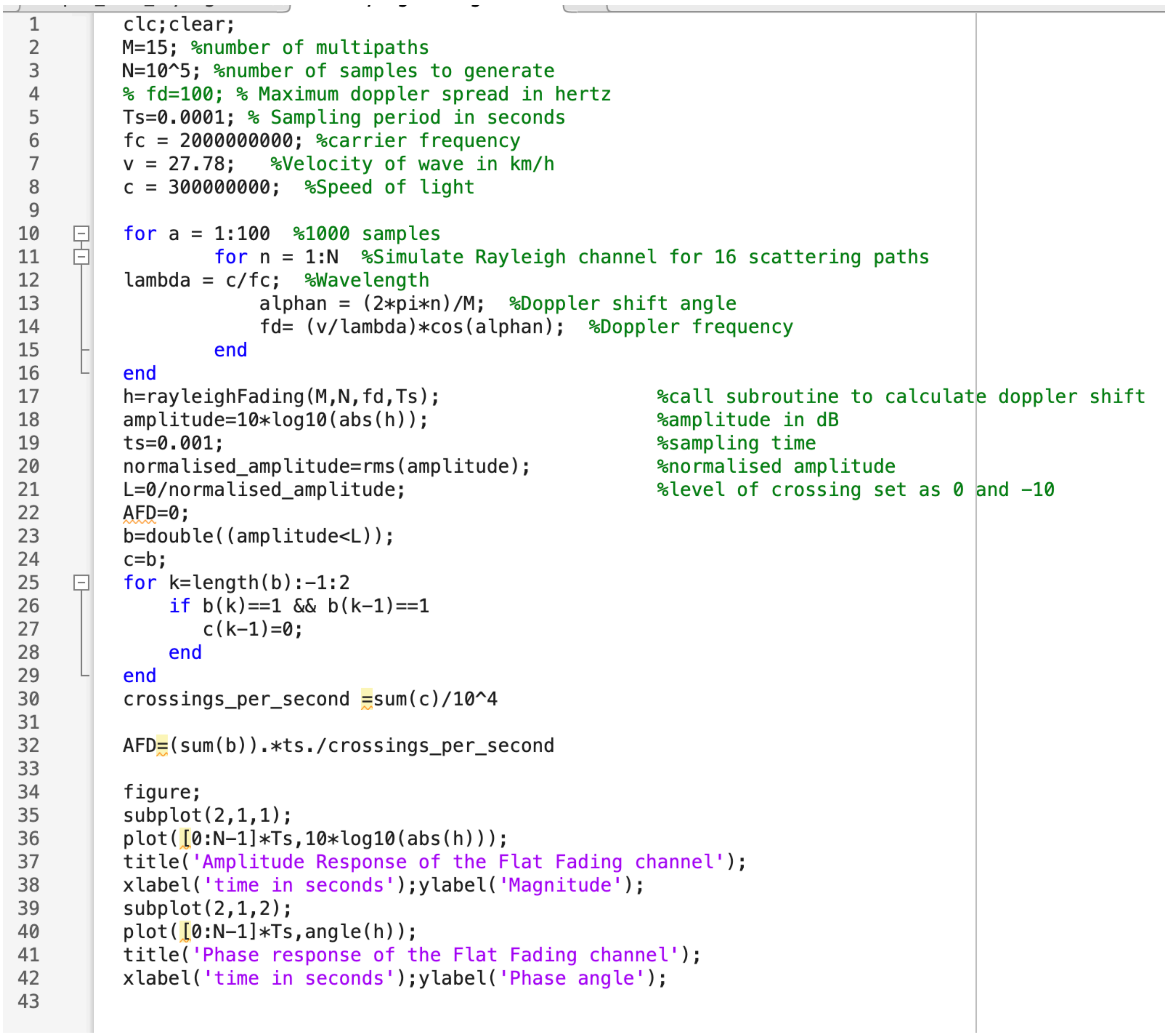
Lines 12 to 14 simulate transmitter which is nothing but uniformly distributed random symbols.

Lines 16 to 18 simulate Rayleigh channel. Rayleigh effect is simulated by 2 independent random variables as seen in line 22. Line 24 is for additive white gaussian noise.

Lines 20 and 21 are for simulating receivers. It is a single tap inverting equalizer and optimum detector.

Line 23 simulates error counter.

Finally line 25 calculates simulated symbol rate and then plot is made for BER vs SNR in lines 26 to 32.



```
clc;clear;
M=15; %number of multipaths
N=10^5; %number of samples to generate
% fd=100; % Maximum doppler spread in hertz
Ts=0.0001; % Sampling period in seconds
fc = 2000000000; %carrier frequency
v = 27.78;   %Velocity of wave in km/h
c = 300000000;  %Speed of light

for a = 1:100  %1000 samples
        for n = 1:N  %Simulate Rayleigh channel for 16 scattering paths
lambda = c/fc;  %Wavelength
            alphan = (2*pi*n)/M;  %Doppler shift angle
            fd= (v/lambda)*cos(alphan);  %Doppler frequency
        end
end
h=rayleighFading(M,N,fd,Ts);                    %call subroutine to calculate doppler shift
amplitude=10*log10(abs(h));                     %amplitude in dB
ts=0.001;                                       %sampling time
normalised_amplitude=rms(amplitude);            %normalised amplitude
L=0/normalised_amplitude;                       %level of crossing set as 0 and -10
AFD=0;
b=double((amplitude<L));
c=b;
for k=length(b):-1:2
    if b(k)==1 && b(k-1)==1
       c(k-1)=0;
    end
end
crossings_per_second =sum(c)/10^4

AFD=(sum(b)).*ts./crossings_per_second

figure;
subplot(2,1,1);
plot([0:N-1]*Ts,10*log10(abs(h)));
title('Amplitude Response of the Flat Fading channel');
xlabel('time in seconds');ylabel('Magnitude');
subplot(2,1,2);
plot([0:N-1]*Ts,angle(h));
title('Phase response of the Flat Fading channel');
xlabel('time in seconds');ylabel('Phase angle');

```

Code Comments explain various steps.

Lines 2 to 8 are for initialization as seen in the comments.

Lines 10 to 18 calculate Rayleigh fading amplitude.

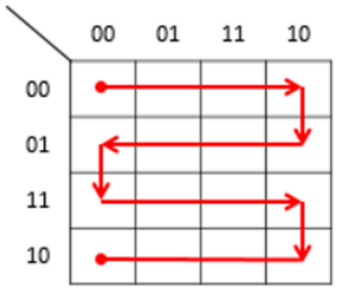
Lines 19 to 32 calculate average number of crossings per second and average fade duration.

Line 30 is crossings per second and line 32 is average fading duration.

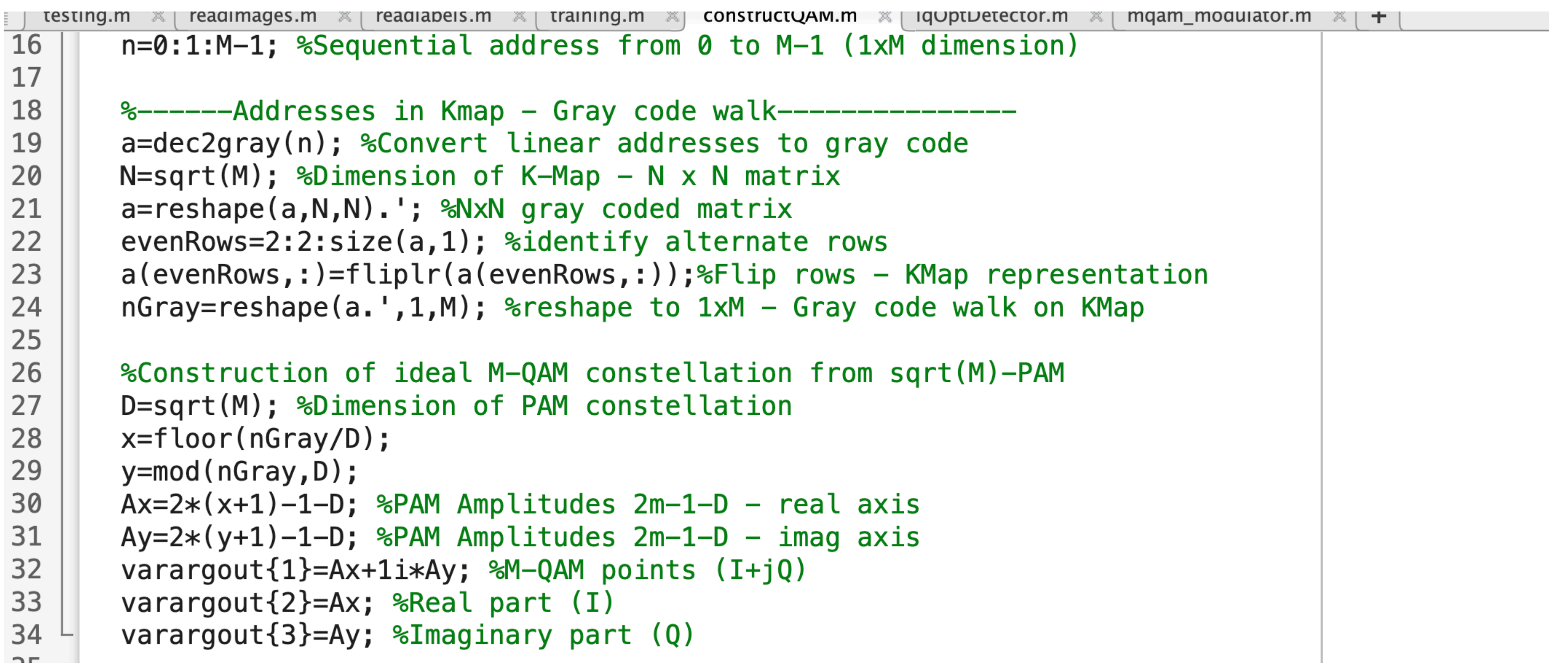
Lines 34 to 42 are for plotting.

The technique used to generate QAM is discussed below.

The diagram below shows how a K-Map matrix is traversed in Karnaugh walk.



The binary values which we get by traversing the k-map are successive constellation points.



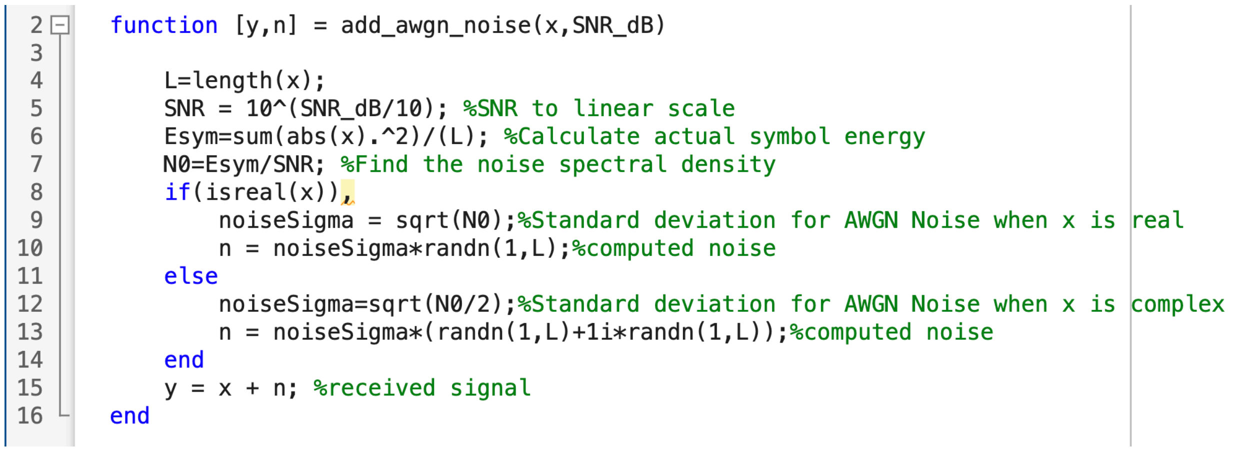
```
n=0:1:M-1; %Sequential address from 0 to M-1 (1xM dimension)

%------Addresses in Kmap - Gray code walk---------------
a=dec2gray(n); %Convert linear addresses to gray code
N=sqrt(M); %Dimension of K-Map - N x N matrix
a=reshape(a,N,N).'; %NxN gray coded matrix
evenRows=2:2:size(a,1); %identify alternate rows
a(evenRows,:)=fliplr(a(evenRows,:));%Flip rows - KMap representation
nGray=reshape(a.',1,M); %reshape to 1xM - Gray code walk on KMap

%Construction of ideal M-QAM constellation from sqrt(M)-PAM
D=sqrt(M); %Dimension of PAM constellation
x=floor(nGray/D);
y=mod(nGray,D);
Ax=2*(x+1)-1-D; %PAM Amplitudes 2m-1-D - real axis
Ay=2*(y+1)-1-D; %PAM Amplitudes 2m-1-D - imag axis
varargout{1}=Ax+1i*Ay; %M-QAM points (I+jQ)
varargout{2}=Ax; %Real part (I)
varargout{3}=Ay; %Imaginary part (Q)
```

Lines 19 to 24 simulate gray code walk.

Lines 27 to 34 are for any rectangular QAM constellation generation. We know that QAM is equivalent to superimposing two *Amplitude Shift Keying (ASK)* signal. Lines 27 to 34 do just that.

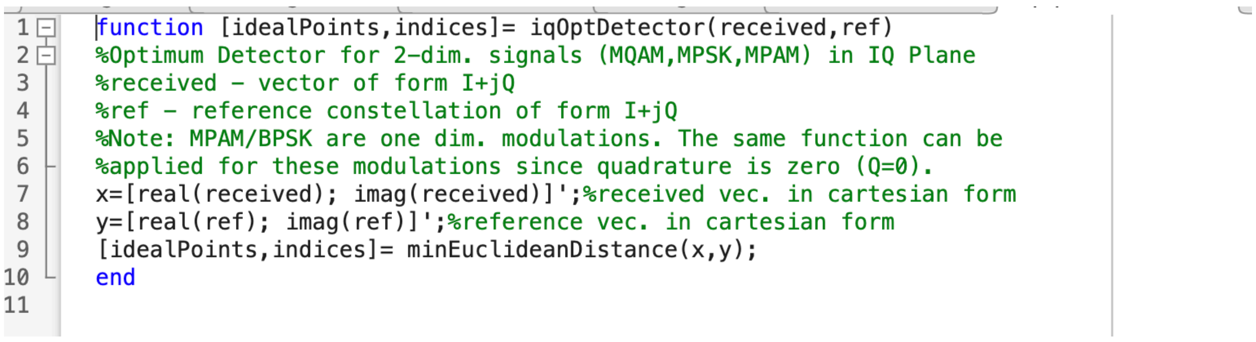


```
 2  function [y,n] = add_awgn_noise(x,SNR_dB)
 3  
 4      L=length(x);
 5      SNR = 10^(SNR_dB/10); %SNR to linear scale
 6      Esym=sum(abs(x).^2)/(L); %Calculate actual symbol energy
 7      N0=Esym/SNR; %Find the noise spectral density
 8      if(isreal(x)),
 9          noiseSigma = sqrt(N0);%Standard deviation for AWGN Noise when x is real
10          n = noiseSigma*randn(1,L);%computed noise
11      else
12          noiseSigma=sqrt(N0/2);%Standard deviation for AWGN Noise when x is complex
13          n = noiseSigma*(randn(1,L)+1i*randn(1,L));%computed noise
14      end
15      y = x + n; %received signal
16  end
```

The screenshot above is of code which simulates white noise.

In line 4, length of vector of modulated symbols with Rayleigh effect applied.

Lines from 5 to 16 add white noise to modulated symbols using the uncorrelated random variables.



```matlab
1  function [idealPoints,indices]= iqOptDetector(received,ref)
2  %Optimum Detector for 2-dim. signals (MQAM,MPSK,MPAM) in IQ Plane
3  %received - vector of form I+jQ
4  %ref - reference constellation of form I+jQ
5  %Note: MPAM/BPSK are one dim. modulations. The same function can be
6  %applied for these modulations since quadrature is zero (Q=0).
7  x=[real(received); imag(received)]';%received vec. in cartesian form
8  y=[real(ref); imag(ref)]';%reference vec. in cartesian form
9  [idealPoints,indices]= minEuclideanDistance(x,y);
10 end
11
```

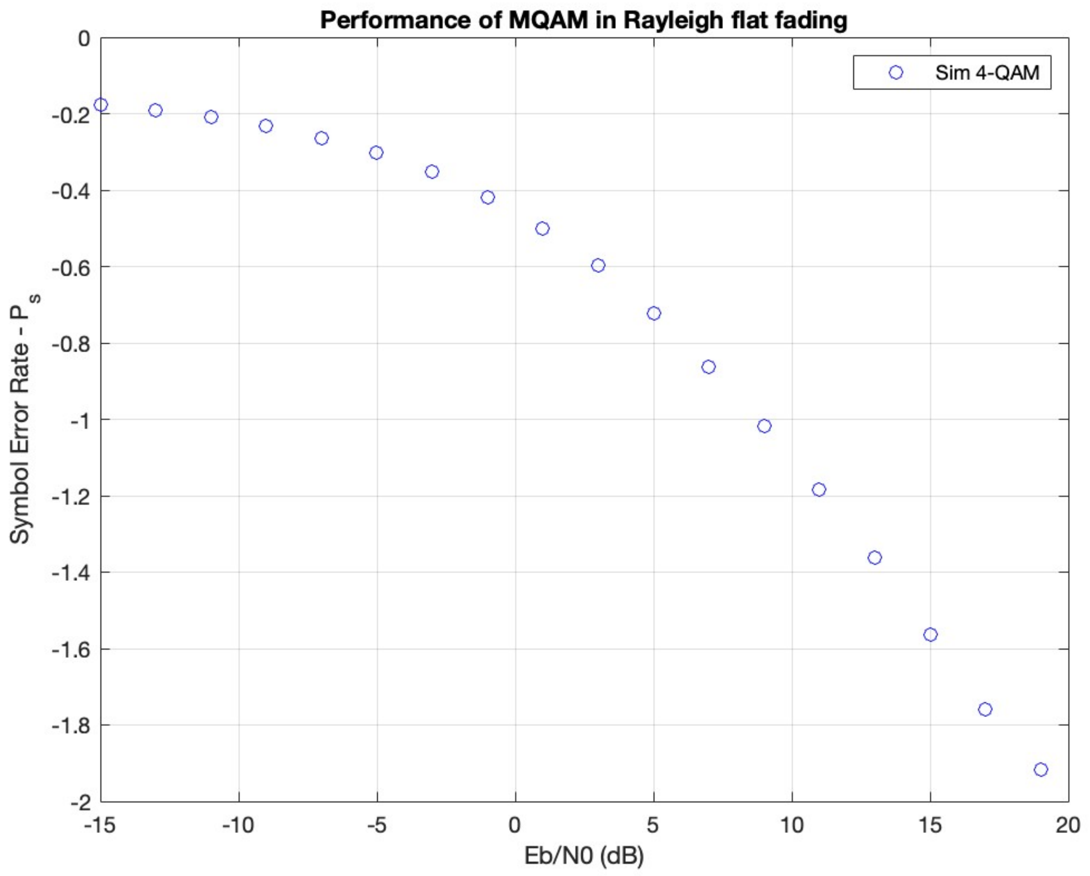
The screenshot above is of code of optimum detector.

Line 7 is the received symbol vector and line 8 is reference symbol vector.

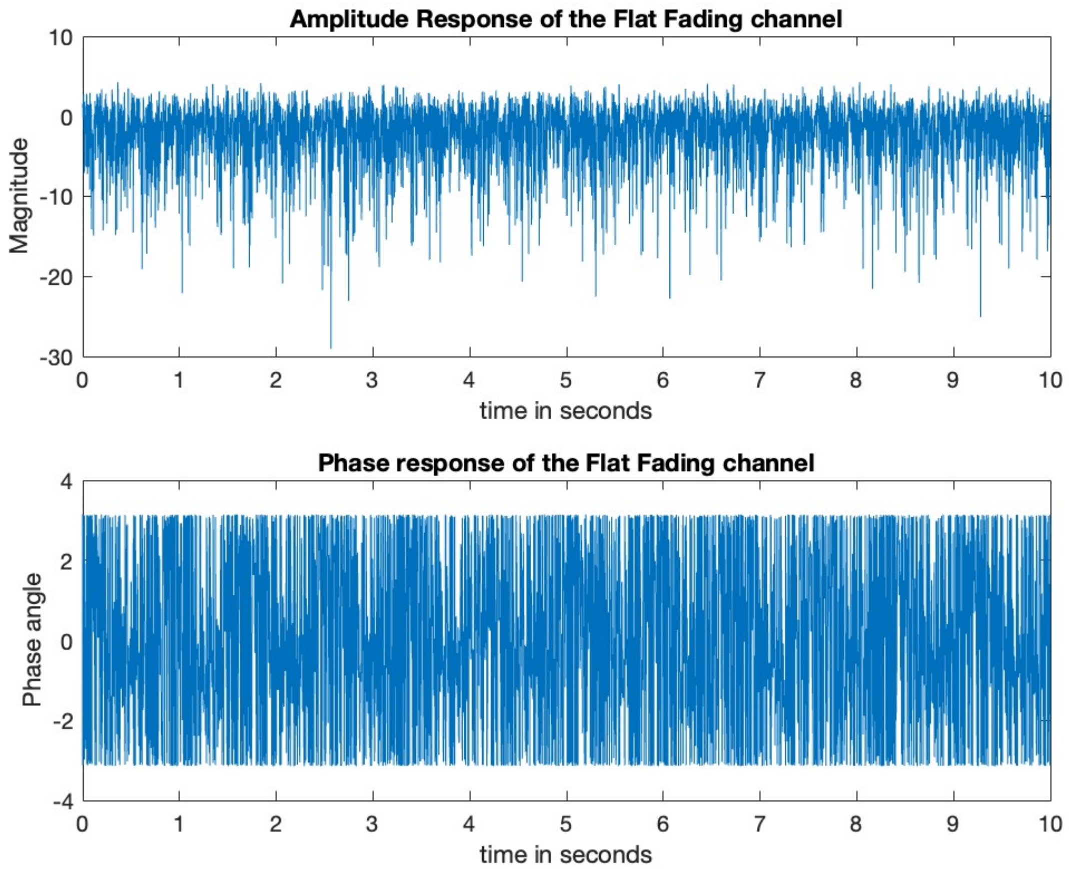
Both the vectors are passed to another function in line 9 to find minimum euclidean distance between the received and reference symbol vectors for detection of received symbols.

Figures :

BER (Bit Error Rate) vs SNR graph is below-



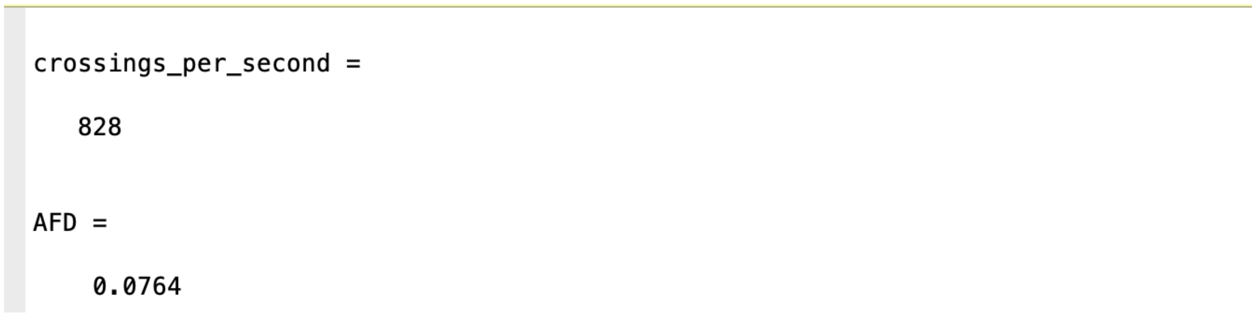
Graph of Envelope fading is below-



The average number of crossings for 0 dB and -10 dB below RMS is respectively as follows-

AFD is average fading time.

0 dB :



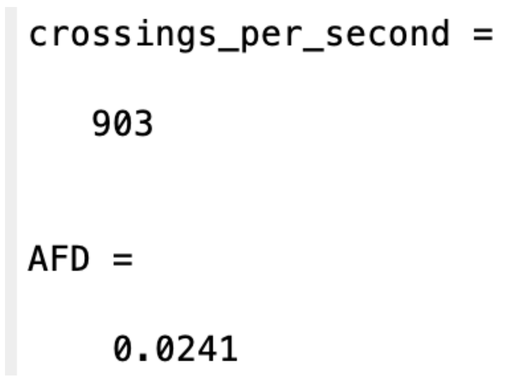
```
crossings_per_second =

   828


AFD =

    0.0764
```

-10 dB :



```
crossings_per_second =

   903


AFD =

    0.0241
```

Discussion :

As said in problem 1, doppler fading envelope was generated. Average number of crossings per second was calculated and average fading time was calculated.

As said in problem 2, BER(bit error rate) was calculated using the mentioned steps and SNR was calculated. Graph was plotted between SNR and BER.

Code execution :

Put all the files in workspace.

Run the code mqam_over_rayleigh.m to generate BER vs SNR graph.

Run testRayleighFading.m to calculate envelope crossings.