**NATIONAL INSTITUTE OF TECHNOLOGY CALICUT**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

****

**DIGITAL COMMUNICATION LABORATORY**

**EXP-8: 8** QAM WITH DIFFERENT CONSTELLATIONS

(SOFTWARE)

**Submitted by:**

B.Veda Sree - B150734EC

B.Sravani - B150783EC

Ch.Priyanka - B150837EC

Ch.Vamsi Narayana Reddy - B150530EC

**Aim:**

Simulate a baseband 8QAM communication system with 1000symbols in the presence of AWGN and compare BER performance for three different constellations (rectangular, circular1 and circular2)

* Implementation of 8QAM modulation without using built-in MATLAB functions.
* Constructing three different constellation diagrams.
* Plotting of theoretical and experimental BER curves for different constellations.

**Theory:**

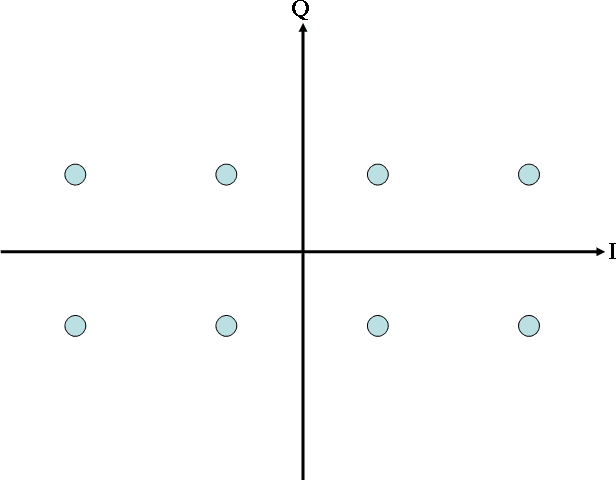
**QAM:**

Quadrature amplitude modulation (QAM) conveys two digital [bit streams](https://en.wikipedia.org/wiki/Bit_stream), by changing (*modulating*) the [amplitudes](https://en.wikipedia.org/wiki/Amplitude) of two [carrier waves](https://en.wikipedia.org/wiki/Carrier_wave), using the [amplitude-shift keying](https://en.wikipedia.org/wiki/Amplitude-shift_keying) (ASK) digital modulation scheme. The two carrier waves of the same frequency, usually [sinusoids](https://en.wikipedia.org/wiki/Sine_wave" \o "Sine wave),are [out of phase](https://en.wikipedia.org/wiki/Out_of_phase) with each other by 90° and are thus called [quadrature](https://en.wikipedia.org/wiki/Quadrature_phase) carriers or quadrature components — hence the name of the scheme. The modulated waves are summed, and the final waveform is a combination of both [phase-shift keying](https://en.wikipedia.org/wiki/Phase-shift_keying) (PSK) and [amplitude-shift keying](https://en.wikipedia.org/wiki/Amplitude-shift_keying) (ASK). In this , a finite number of at least two phases and at least two amplitudes are used.

**RECTANGULAR QAM**

Rectangular QAM constellations are, in general, sub-optimal in the sense that they do not maximally space the constellation points for a given energy. However, they have the considerable advantage that they may be easily transmitted as two [pulse amplitude modulation](https://en.wikipedia.org/wiki/Pulse_amplitude_modulation) (PAM) signals on quadrature carriers, and can be easily demodulated. The non-square constellations, dealt with below, achieve marginally better bit-error rate (BER) but are harder to modulate and demodulate.

**Rectangular constellation**

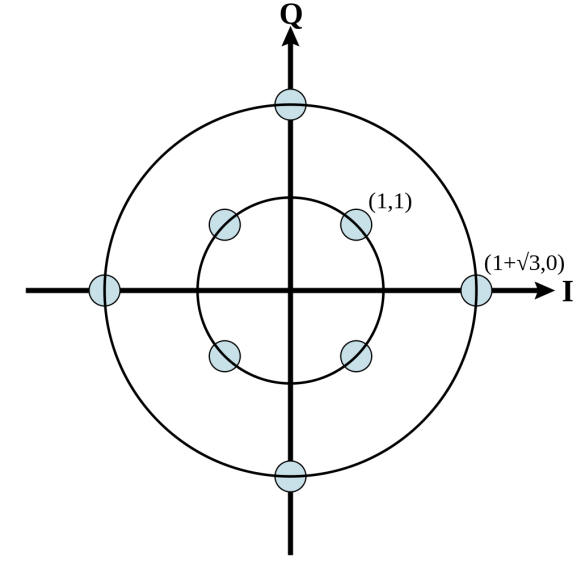
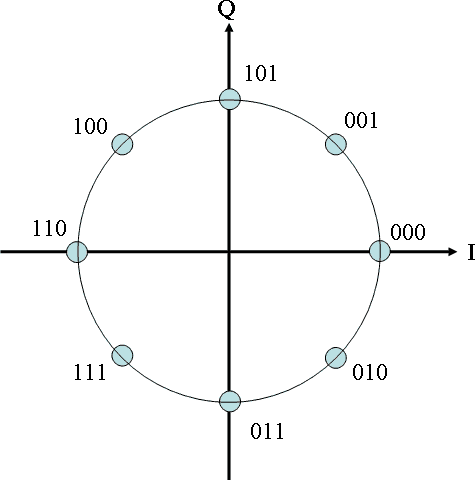


{\displaystyle \,P\_{s}=1-\left(1-P\_{sc}\right)^{2}}

**CIRCULAR QAM**

The circular 8-QAM constellation is known to be the optimal 8-QAM constellation in the sense of requiring the least mean power for a given minimum Euclidean distance (Fig.2).

**Fig1: Circular1 Constellation Fig2:Circular2 Constellation**



The circular 1 Constellation shown above is not exactly QAM as the amplitude is same for all the symbols but we just take it for comparison among different constellations.

**SIMULATION METHODOLOGY**

1. Define constellation points for rectangular, circular1, circular2 constellations and plot the same.
2. Map QAM constellations through index values and add AWGN before transmitting.
3. Find the minimum Euclidean distance between the received symbol and all the 8 symbols in each of the three cases to demodulate the transmitted signal.
4. Find BER theoretically for rectangular, circular1 and circular2 constellations.
5. Compare BER plots of all three constellations.
6. Compare theoretical and experimental BER plots for the three constellations.

**MATLAB CODE**

clear all

close all

clc;

snrdB = [0:2:12];

snr = 10.^(snrdB/10);%snr conversion from db to normal scale

%3 types of constellations, rectangle, circular 1 and 2

ber\_r=zeros(1,length(snrdB));

ber\_c1=zeros(1,length(snrdB));

ber\_c2=zeros(1,length(snrdB));

ber\_rth=zeros(1,length(snrdB));

ber\_c1th=zeros(1,length(snrdB));

ber\_c2th=zeros(1,length(snrdB));

N=30000;%3 bits\*10000 symbols

iter\_max=50;

% constellations

rec=[-3+1i -1+1i 1+1i 3+1i -3-1i -1-1i 1-1i 3-1i];

cir1=[1 -1 1i -1i (1+1i)/sqrt(2) (-1+1i)/sqrt(2) (1-1i)/sqrt(2) (-1-1i)/sqrt(2)];

cir2 = [1+1i 1-1i -1+1i -1-1i 1+sqrt(3) -(1+sqrt(3)) (1+sqrt(3))\*1i -(1+sqrt(3))\*1i];

rec\_bit = [0 0 0; 0 1 0; 1 0 0; 1 1 0; 0 0 1; 0 1 1; 1 0 1; 1 1 1];

cir1\_bit=[1 1 1; 0 0 1; 0 1 0; 1 0 0; 1 1 0; 0 1 1; 1 0 1; 0 0 0];

cir2\_bit=[0 0 0; 0 1 1; 0 0 1; 0 1 0; 1 0 0; 1 1 0; 1 0 1; 1 1 1];

for i=1:1:length(snrdB)

error = zeros(1, iter\_max);

for k=1:1:iter\_max

signal = round(rand(1,N));

const\_rec = zeros(1, N/3);

const\_cir1 = zeros(1, N/3);

const\_cir2 = zeros(1, N/3);

for m=1:3:length(signal)

symbol=signal(m:m+2);% group 3 bits as a symbol

%Check to what symbol it corresponds and assign to it properly

if(symbol == [0 0 0])

const\_rec((m-1)/3+1)=-3+1i;%INDEXING PROPERLY, MATLAB from 1

const\_cir1((m-1)/3+1)=(-1-1i)/sqrt(2);

const\_cir2((m-1)/3+1)=1+1i;

elseif(symbol==[0 0 1])

const\_rec((m-1)/3+1)=-3-1i;

const\_cir1((m-1)/3+1)=-1;

const\_cir2((m-1)/3+1)=-1+1i;

elseif(symbol==[0 1 0])

const\_rec((m-1)/3+1)=-1+1i;

const\_cir1((m-1)/3+1)=1i;

const\_cir2((m-1)/3+1)=-1-1i;

elseif(symbol==[0 1 1])

const\_rec((m-1)/3+1)=-1-1i;

const\_cir1((m-1)/3+1)=(-1+1i)/sqrt(2);

const\_cir2((m-1)/3+1)=1-1i;

elseif(symbol==[1 0 0])

const\_rec((m-1)/3+1)=1+1i;

const\_cir1((m-1)/3+1)=-1i;

const\_cir2((m-1)/3+1)=1+sqrt(3);

elseif(symbol==[1 0 1])

const\_rec((m-1)/3+1)=1-1i;

const\_cir1((m-1)/3+1)=(1-1i)/(sqrt(2));

const\_cir2((m-1)/3+1)=(1+sqrt(3))\*1i;

elseif(symbol==[1 1 0])

const\_rec((m-1)/3+1)=3+1i;

const\_cir1((m-1)/3+1)=(1+1i)/sqrt(2);

const\_cir2((m-1)/3+1)=-(1+sqrt(3));

elseif(symbol==[1 1 1])

const\_rec((m-1)/3+1)=3-1i;

const\_cir1((m-1)/3+1)=1;

const\_cir2((m-1)/3+1)=-(1+sqrt(3))\*1i;

end

end

%Add Additive white gaussian noise

received\_rec=awgn(const\_rec, snrdB(i)+4.771, 'measured');

received\_cir1=awgn(const\_cir1, snrdB(i)+4.771, 'measured');

received\_cir2=awgn(const\_cir2, snrdB(i)+4.771, 'measured');

for l=1:1:length(received\_rec)

dist\_rec = zeros(1, 8);

dist\_cir1 = zeros(1, 8);

dist\_cir2 = zeros(1, 8);

%calculate distance of every symbol with these 8 fixed symbols

for ll=1:1:8

dist\_rec(ll)=abs(received\_rec(l)-rec(ll));

dist\_cir1(ll)=abs(received\_cir1(l)-cir1(ll));

dist\_cir2(ll)=abs(received\_cir2(l)-cir2(ll));

end

%take min distance out of these values

[v, minrec] = min(dist\_rec);

[v, mincir1] = min(dist\_cir1);

[v, mincir2] = min(dist\_cir2);

%assign that closest symbol codeword to decoder

decoded\_rec(l, :) = rec\_bit(minrec, :);

decoded\_cir1(l, :) = cir1\_bit(mincir1, :);

decoded\_cir2(l, :) = cir2\_bit(mincir2, :);

end

temp = decoded\_rec';

rec\_stream = temp(:);%alligning all the symbols(bits) in a row

temp = decoded\_cir1';

cir1\_stream = temp(:);

temp = decoded\_cir2';

cir2\_stream = temp(:);

%Calculate error i.e if decoded different from input

% and taking average over all the iterations

error\_rec = abs(rec\_stream'-signal);

errorno\_rec(k) = sum(error\_rec);

ber\_r(i) = ber\_r(i) + (errorno\_rec(k)/iter\_max);

error\_cir1 = abs(cir1\_stream'-signal);

errorno\_cir1(k) = sum(error\_cir1);

ber\_c1(i) = ber\_c1(i) + (errorno\_cir1(k)/iter\_max);

error\_cir2 = abs(cir2\_stream'-signal);

errorno\_cir2(k) = sum(error\_cir2);

ber\_c2(i) = ber\_c2(i) + (errorno\_cir2(k)/iter\_max);

end

end

ber\_r = ber\_r/N;

ber\_c1 = ber\_c1/N;

ber\_c2 = ber\_c2/N;

ber\_rth = 2\*erfc(sqrt((9/14)\*snr));

ber\_c1th = erfc(sqrt(3\*snr/14))\*sin(pi/8);

ber\_c2th= 3.5\*erfc(sqrt(snr));

scatterplot(const\_rec);

xlabel('In-phase');

ylabel('Quadrature');

title('Constellation of rectangular QAM');

scatterplot(const\_cir1);

xlabel('In-phase');

ylabel('Quadrature');

title('Constellation of Circular QAM - 1');

scatterplot(const\_cir2);

xlabel('In-phase');

ylabel('Quadrature');

title('Constellation of Circular QAM - 2');

scatterplot(received\_rec);

xlabel('In-phase');

ylabel('Quadrature');

title('Constellation of received rectangular QAM');

scatterplot(received\_cir1);

xlabel('In-phase');

ylabel('Quadrature');

title('Constellation of received Circular QAM - 1');

scatterplot(received\_cir2);

xlabel('In-phase');

ylabel('Quadrature');

title('Constellation of received Circular QAM - 2');

figure;

semilogy(snrdB, ber\_r, 'r\*-', snrdB, ber\_c1, 'b-', snrdB, ber\_c2, 'k+-');

xlabel('Eb/No in dB');

ylabel('Bit Error Rate');

legend('Rectangular constellation', 'Circular constellation 1', 'Circular constellation 2');

title('Comparison of the three schemes, experimental');

figure;

semilogy(snrdB, ber\_rth, 'r\*-', snrdB, ber\_c1th, 'b-', snrdB, ber\_c2th, 'k+-');

xlabel('Eb/No in dB');

ylabel('Bit Error Rate');

legend('Rectangular constellation', 'Circular constellation 1', 'Circular constellation 2');

title('Comparison of the three schemes, theoritical');

figure;

semilogy(snrdB, ber\_r, 'r', snrdB, ber\_rth, 'r\*');

xlabel('Eb/No in dB');

ylabel('Bit Error Rate');

legend('Simulated', 'Theoretical');

title('Rectangular constellation');

figure;

semilogy(snrdB, ber\_r, 'g', snrdB, ber\_rth, 'g\*');

xlabel('Eb/No in dB');

ylabel('Bit Error Rate');

legend('Simulated', 'Theoretical');

title('Circular constellation 1');

figure;

semilogy(snrdB, ber\_r, 'b', snrdB, ber\_rth, 'b\*');

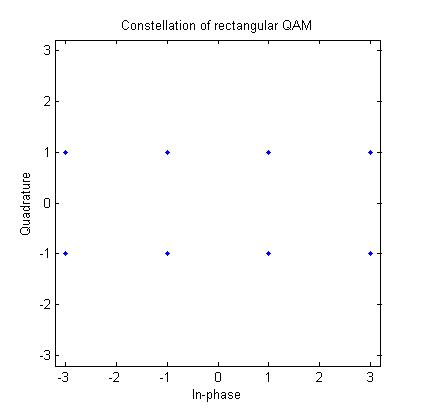
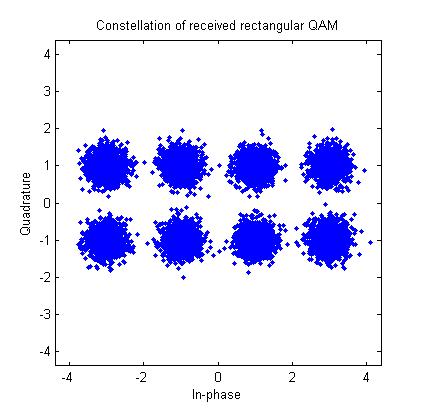
xlabel('Eb/No in dB');

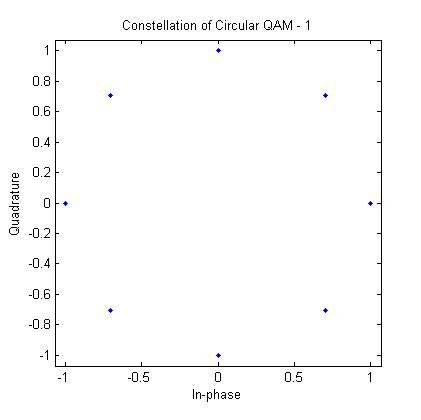
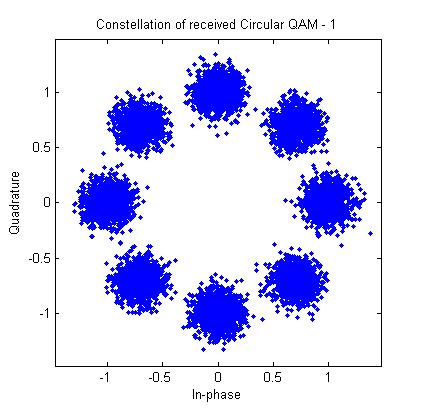
ylabel('Bit Error Rate');

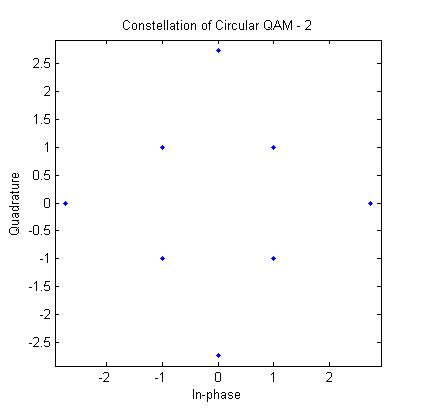
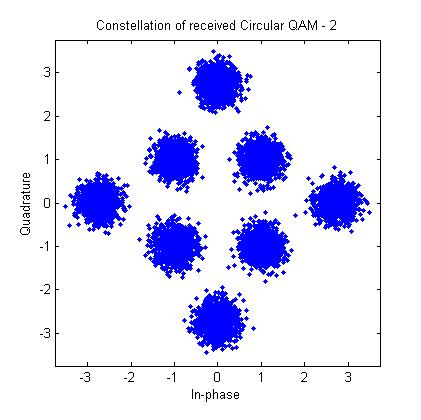
legend('Simulated', 'Theoretical');

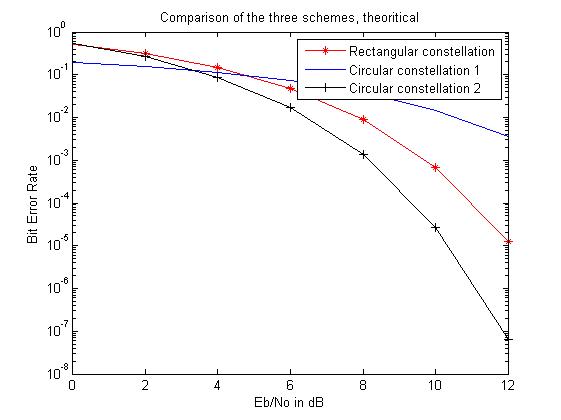
title('Circular constellation 2');

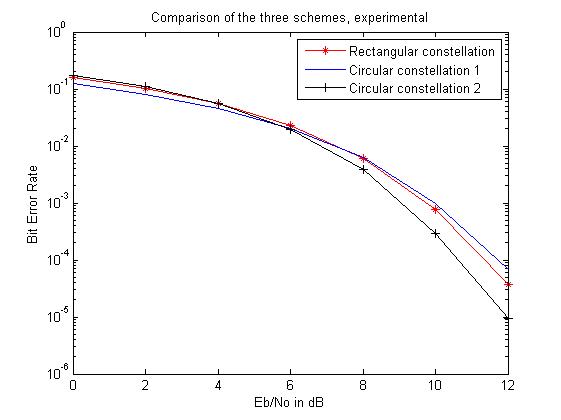
**PLOTS:**

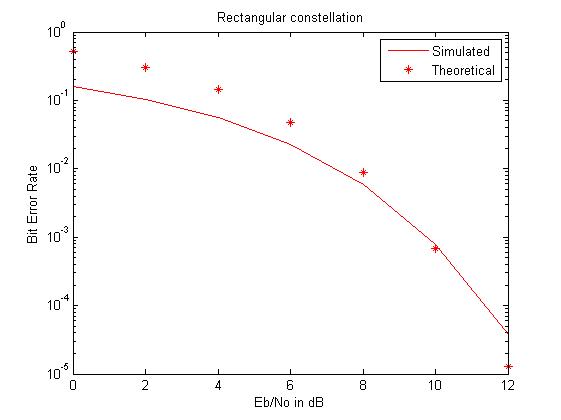
** **

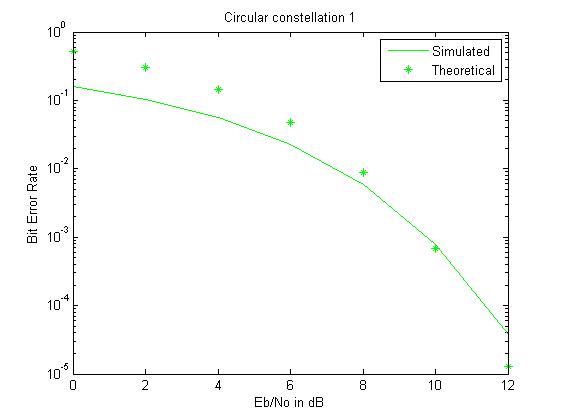
** **

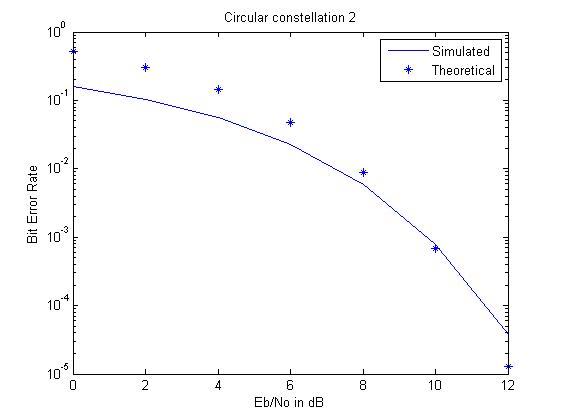
** **

****

****

****

****

****

**INFERENCE:**

1. The constellation diagram of the received signal is spread due the addition of noise.
2. Circular 2 constellation was found to be better when compared to other constellations (because of the low BER for higher SNR values).

**RESULT:**

8QAM modulation technique was simulated in MATLAB and their BER plots and different constellations were studied.