# 1 Introduction

This report presents a detailed performance of the simulation of $M$-ary PSK (Multi-Level Phase Shift Keying) and M-QAM (Multi-Level Quadrature Amplitude Modulation) modulation in an additive white Gaussian noise (AWGN) channel. The performance is analyzed with respect to the Bit Error Rate (BER) and Symbol Error Rate (SER) as a function of the average Signal to Noise ratio per bit ($E_b N_o$).

Throughout this report, we have assumed a complex baseband notation for the transmitted signal $s(t)$

$$s(t) = Re\{u(t)\exp[j2\pi f_c t]\}$$

where, $u(t)$ is the complex baseband signal.

The rest of the report is as follows, in section 2.1, the performance of the $M$-ary PSK modulation for different values of $M$ in an AWGN channel is simulated and corroborated with the result obtained through the theoretical expression for the probability of error. Further, section 2.2 illustrates the effects of the non-idealities (variation in amplitude and phase) on the error performance. Next in section 3, the performance of the $M$-ary QAM modulation for different values of $M$ is discussed. Also, the error vector magnitude (EVM) for the $M$-ary QAM is calculated in section 3.2. Finally, section 4 summarizes the results of the report.

# 2 M-ARY PHASE SHIFT KEYING

This section describes the performance of the $M$-ary phase shift keying (MPSK) modulation in an Additive white Gaussian Noise (AWGN) channel.

## 2.1 ADDITIVE WHITE GAUSSIAN NOISE (AWGN)

### 2.1.1 BINARY PHASE SHIFT KEYING (BPSK)

In Figure 2.1, we illustrate the constellation diagram for a BPSK modulation.

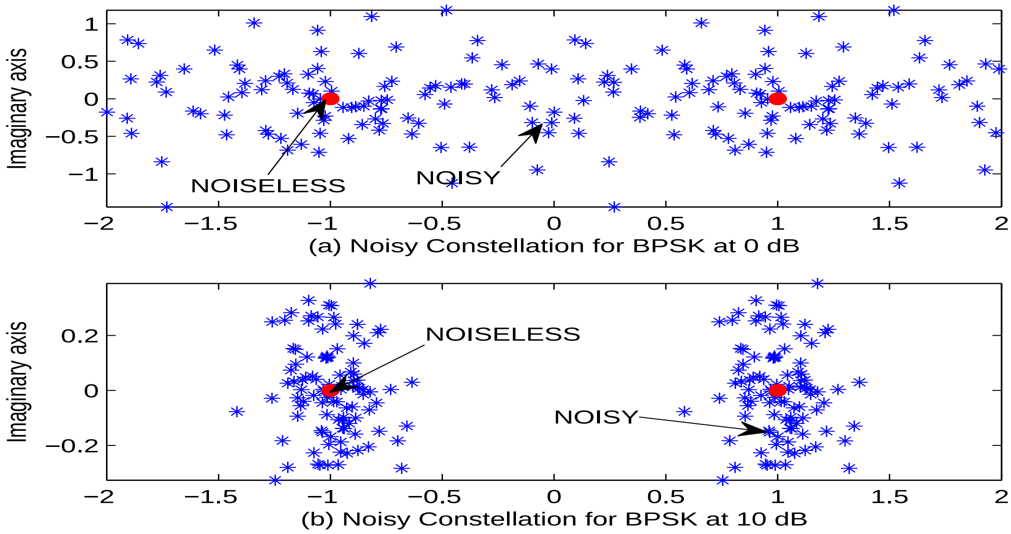



Figure 2.1: Constellation diagram for a BPSK modulation at (a) 0 dB and (b) 10 dB

The BER (which is equal to SER) performance of BPSK modulation is as shown in Figure 2.2.

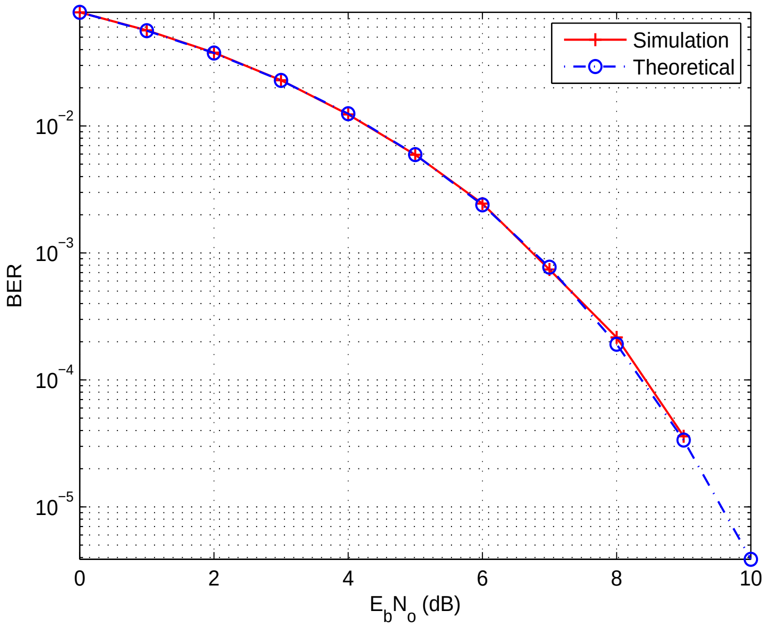
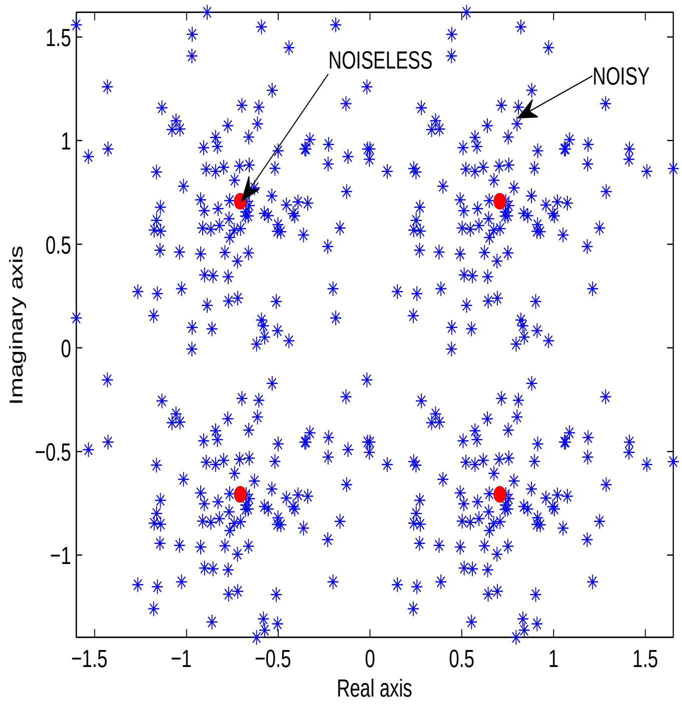


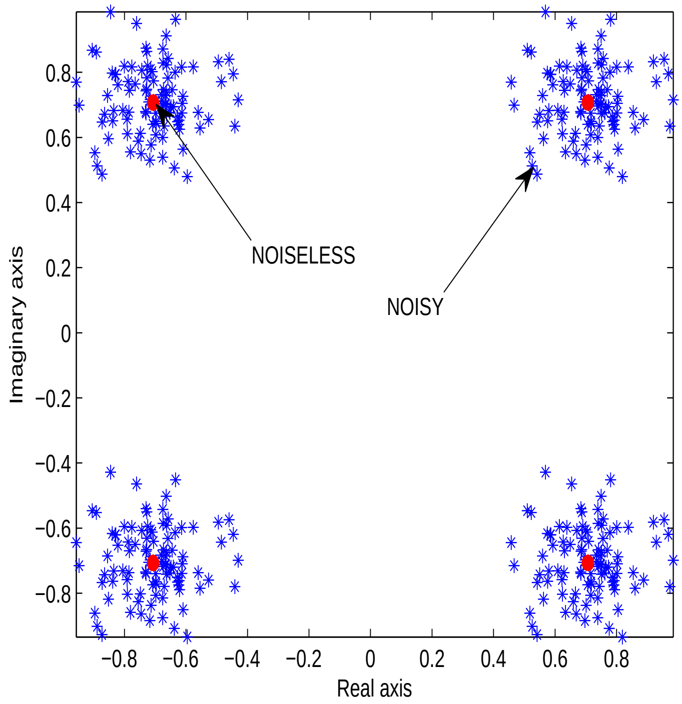

Figure 2.2: BER (and SER) versus $E_bN_o$(dB) for BPSK modulation

#### 2.1.2 QUADRATURE PHASE SHIFT KEYING (QPSK)

The noisy constellation diagram for a QPSK modulation at two different values of $E_b N_o$ is as shown in Figure 2.3.



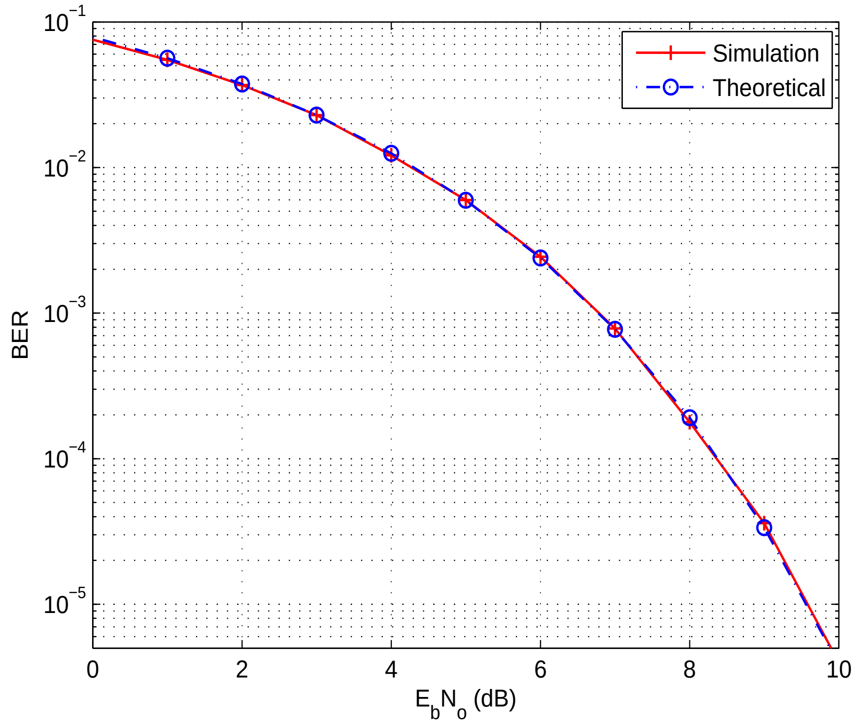

(a) Noisy constellation for QPSK modulation at 0 dB



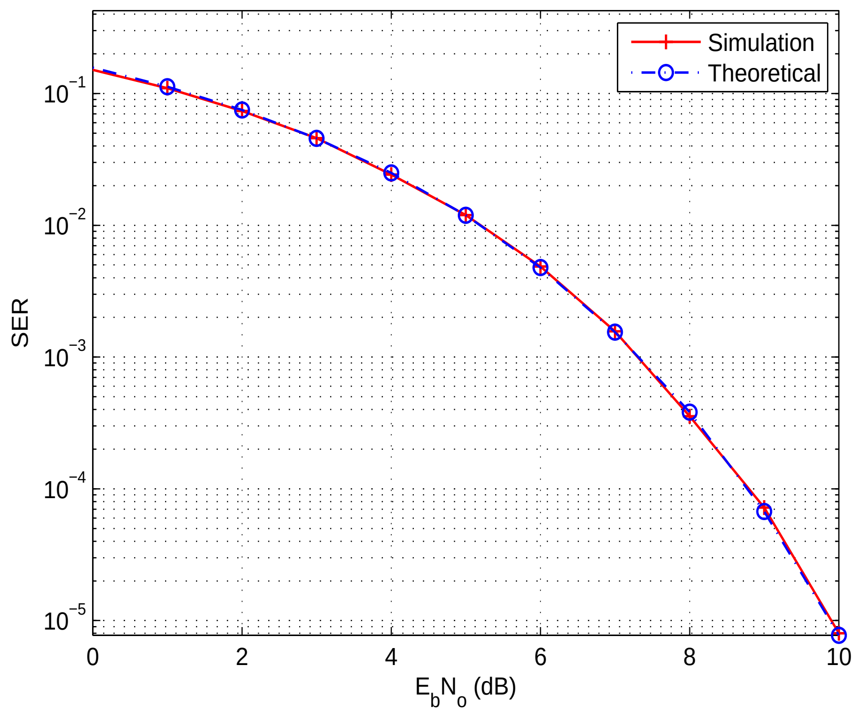

(b) Noisy constellation for QPSK modulation at 10 dB

Figure 2.3: Constellation diagrams for QPSK

The BER and SER performance for a QPSK modulation are as shown in Figure 2.4a and Figure 2.4b respectively.




(a) BER versus $E_b N_o$(dB) for QPSK modulation

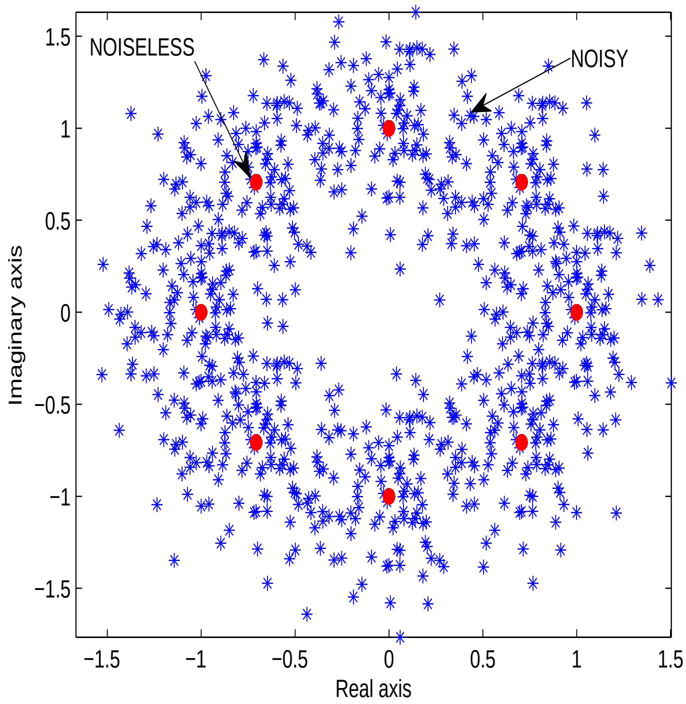



(b) SER versus $E_b N_o$(dB) for QPSK modulation

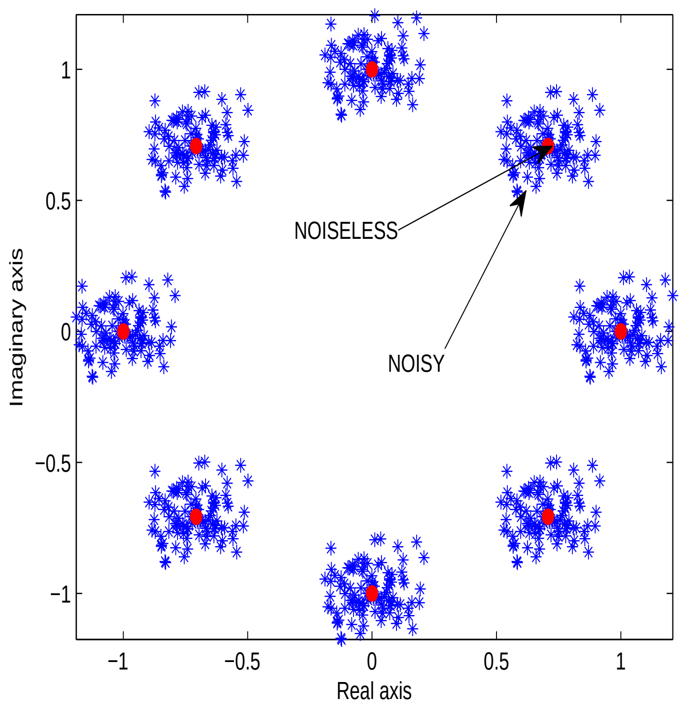
Figure 2.4: Error performance of QPSK modulation

### 2.1.3 8-Phase Shift Keying (8-PSK)

In Figure 2.5, the noisy constellation diagram for an 8-PSK modulation at 0 dB and 10 dB are shown.



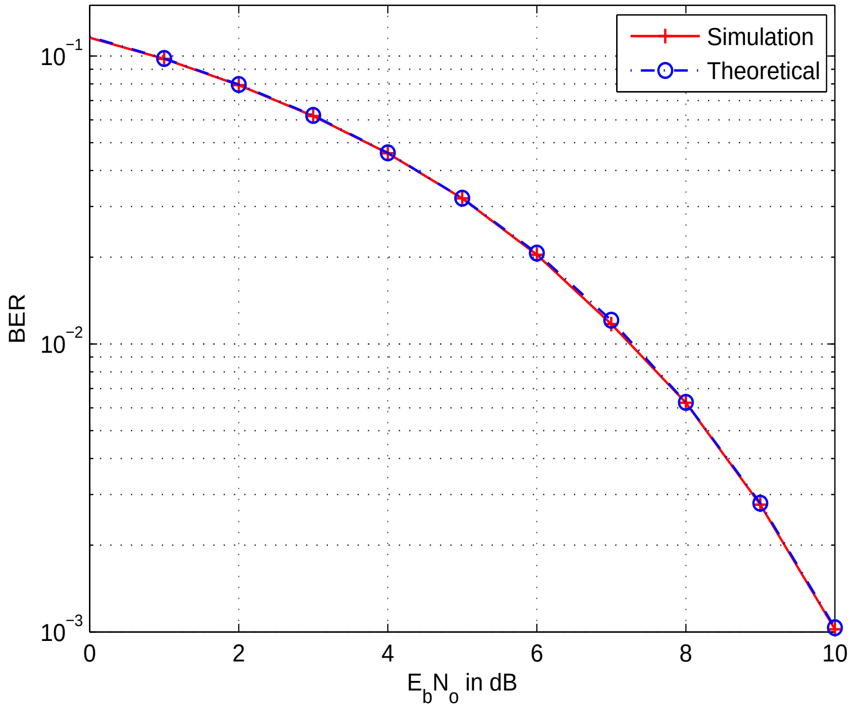

(a) Noisy constellation for 8-PSK modulation at 0 dB



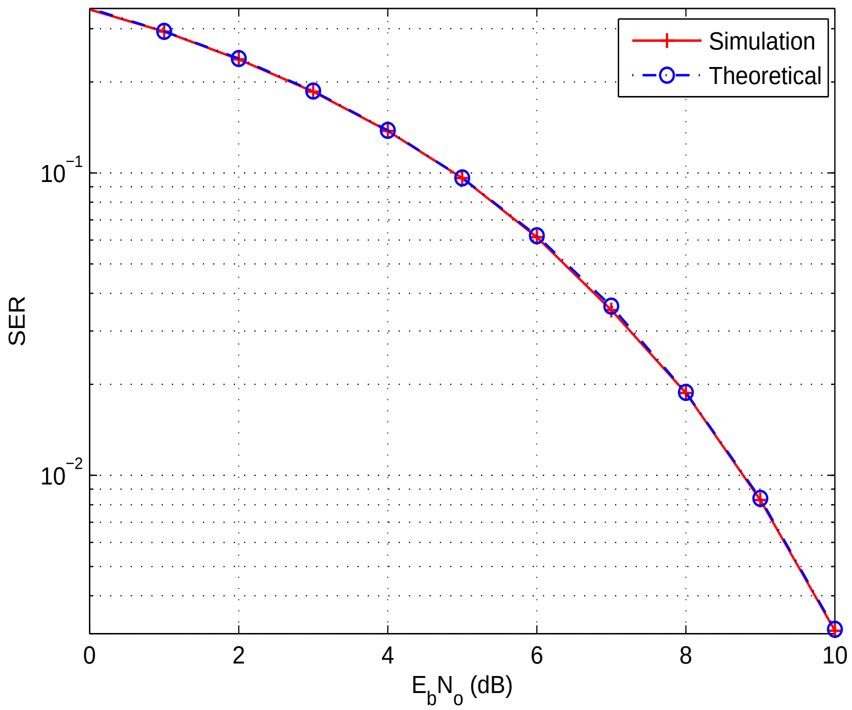

(b) Noisy constellation for 8-PSK modulation at 10 dB

Figure 2.5: Constellation diagrams for 8-PSK modulation

In Figure 2.6a and Figure 2.6b, the BER and SER performance of the 8-PSK modulation is illustrated.




(a) BER versus $E_b N_o$(dB) for 8-PSK modulation

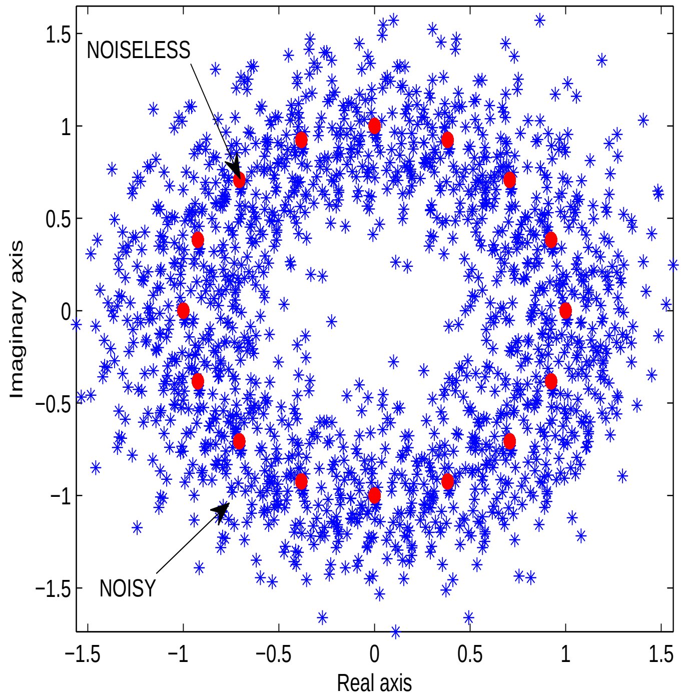



(b) SER versus $E_b N_o$(dB) for 8-PSK modulation

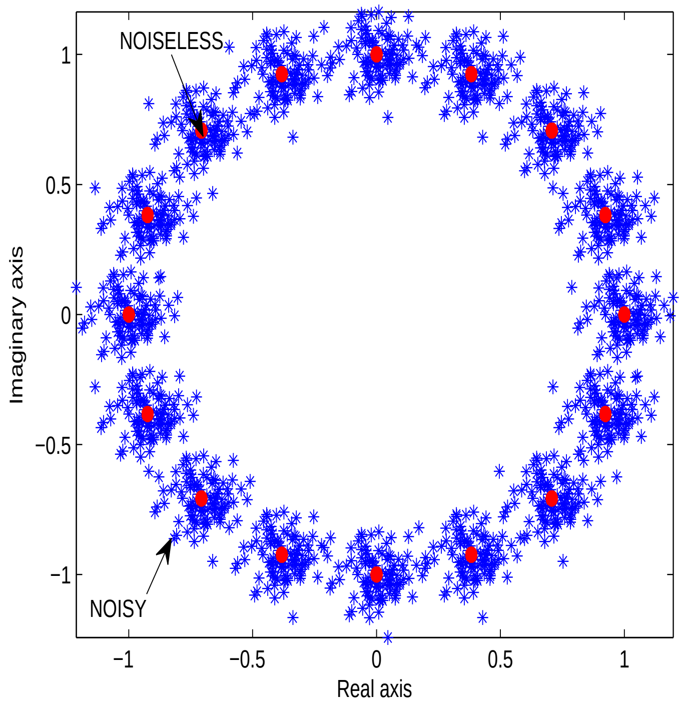
Figure 2.6: Error performance of 8-PSK modulation

### 2.1.4 16-Phase Shift Keying (16-PSK)

Figure 2.7 demonstrates the noisy constellation diagram for an 16-PSK modulation at two different values of $E_b N_o$.



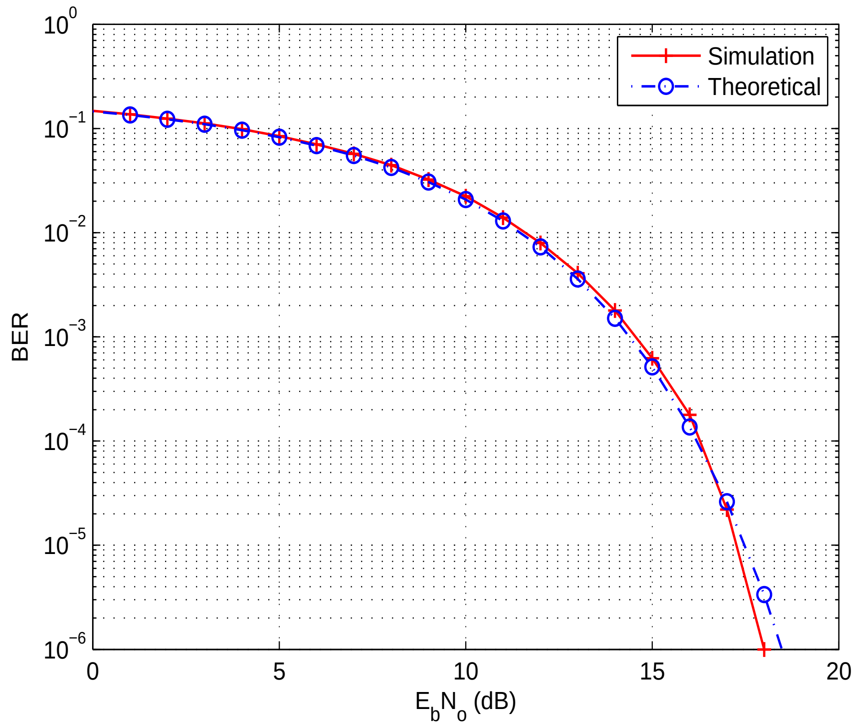

(a) Noisy constellation for 16-PSK modulation at 0 dB



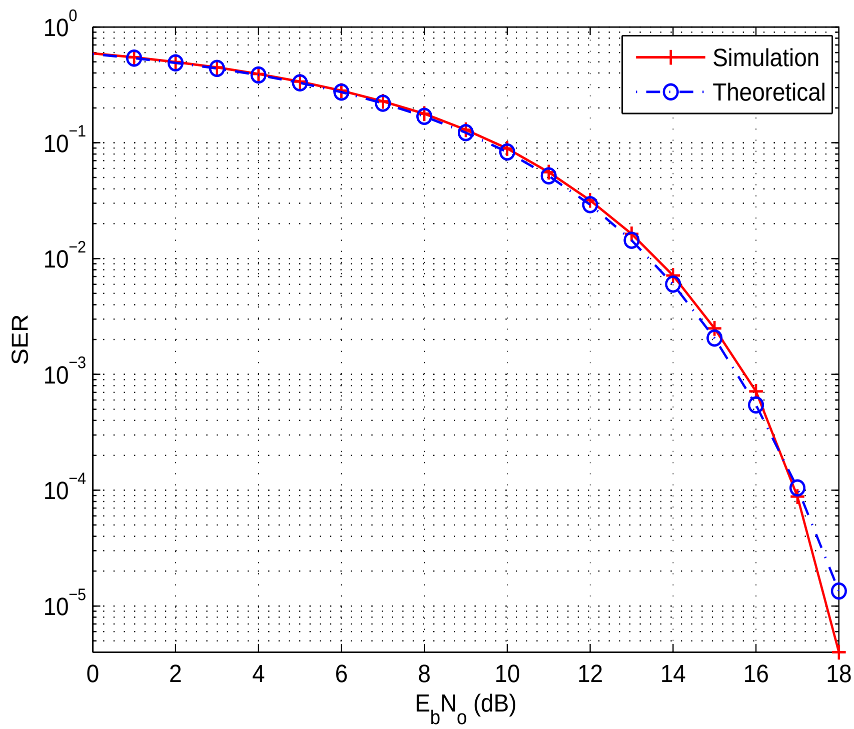

(b) Noisy constellation for 16-PSK modulation at 10 dB

Figure 2.7: Constellation diagrams for 16-PSK

The BER performance is shown as in Figure 2.8a. And illustrated in Figure 2.8b is the SER performance of the 16-PSK modulation.




(a) BER versus $E_bN_o$(dB) for 16-PSK modulation




(b) SER versus $E_bN_o$(dB) for 16-PSK modulation

Figure 2.8: Error performance of 16-PSK modulation

## 2.2 Non-idealities

In this section, the error performance of the MPSK modulation with amplitude and phase non-idealities is elucidated.

### 2.2.1 Binary Phase Shift Keying (BPSK)

In Figure 2.9, the BER (equal to the SER)performance of the BPSK modulation with 1%,5% variations in amplitude and $1^o$,$5^o$ offsets in the phase are shown.

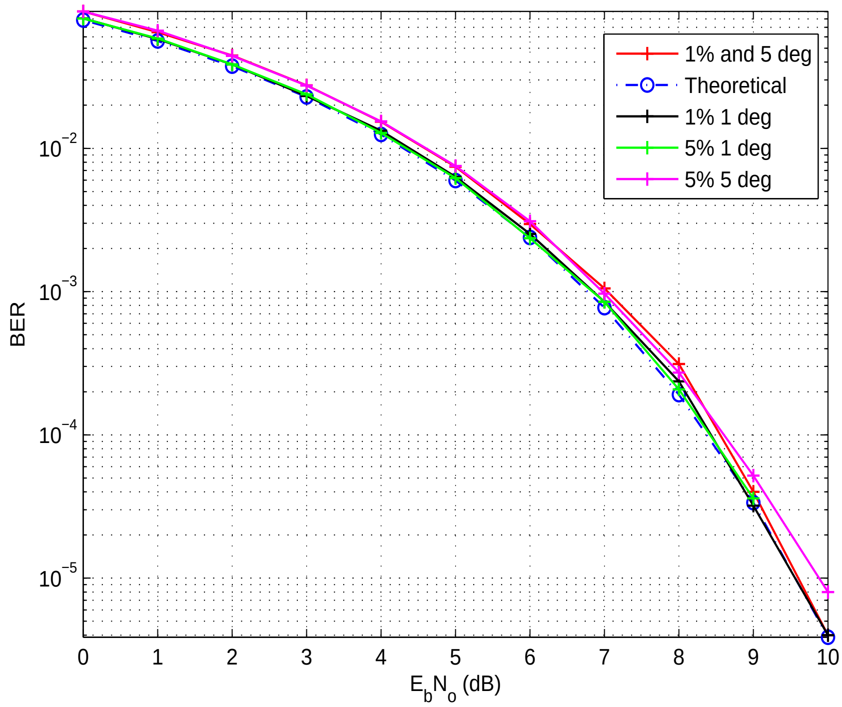



Figure 2.9: BER(and SER) versus $E_bN_o$(dB) for BPSK modulation with non-idealities

### 2.2.2 QUADRATURE PHASE SHIFT KEYING (QPSK)

The bit error rate BER performance of QPSK modulation along with the amplitude and phase variations are illustrated in Figure 2.10.

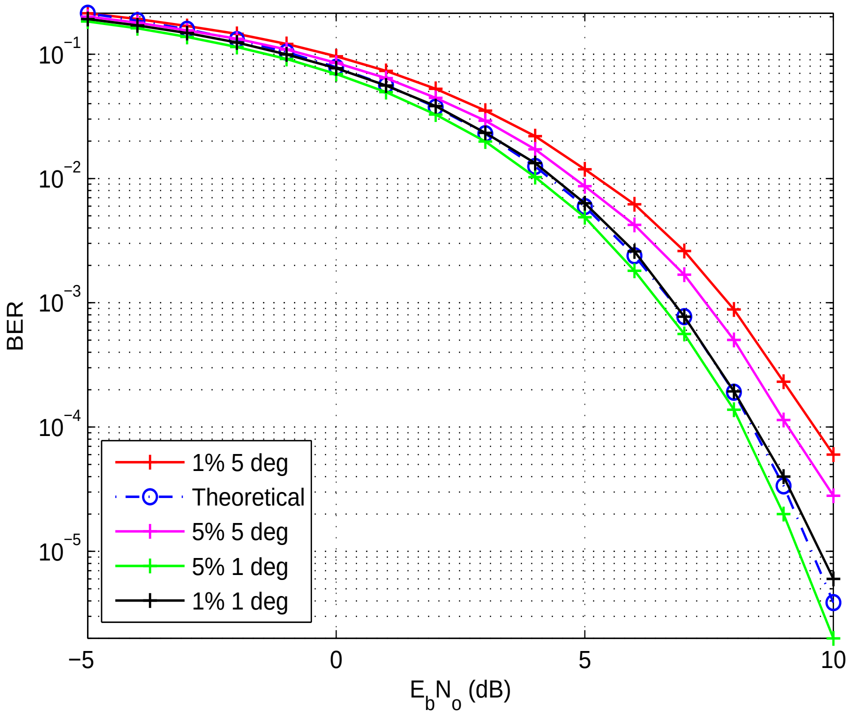



Figure 2.10: BER versus $E_bN_o$(dB) for QPSK modulation with non-idealities

### 2.2.3 8-Phase Shift Keying (8-PSK)

The effect of the amplitude and phase non-idealities on the error performance of the 8-PSK modulation is as demonstrated in Figure 2.11.

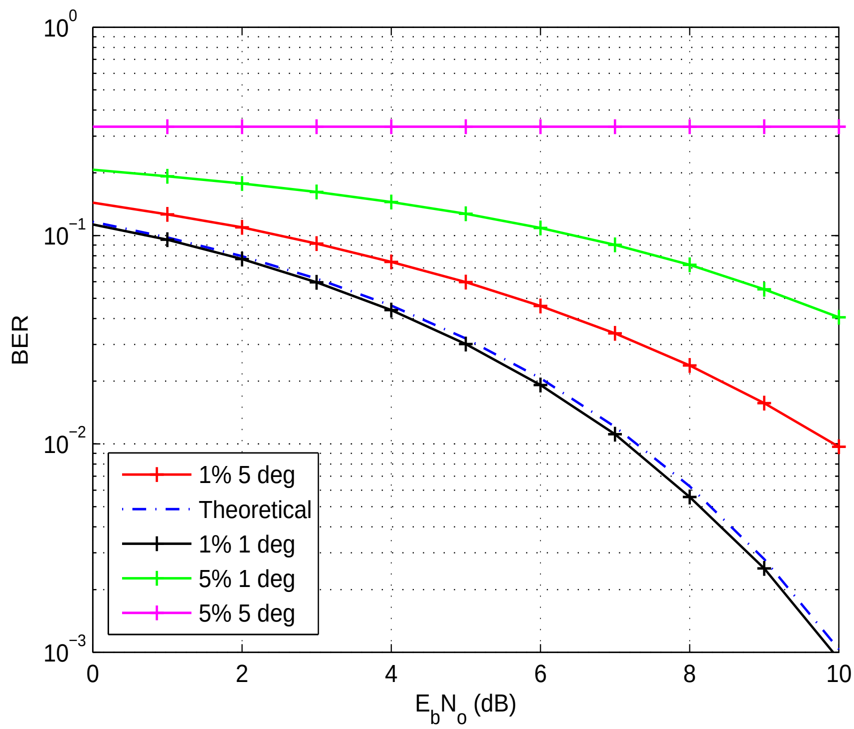



Figure 2.11: BER versus $E_bN_o$(dB) for 8-PSK modulation with non-idealities

### 2.2.4 16-PHASE SHIFT KEYING (16-PSK)

Similar to previous cases, the non-idealities for a 16-PSK are as shown in Figure 2.12.

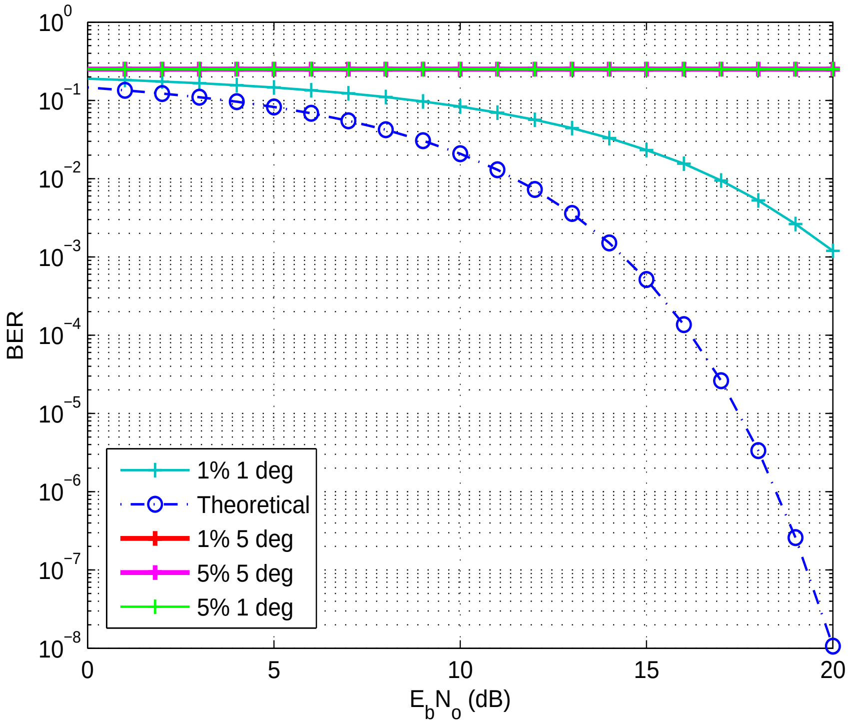
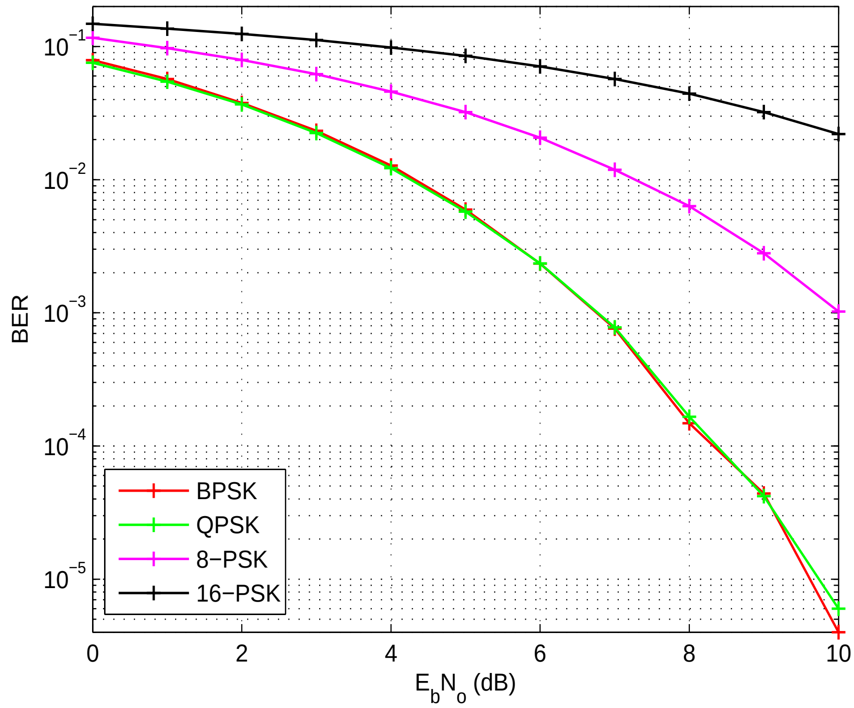


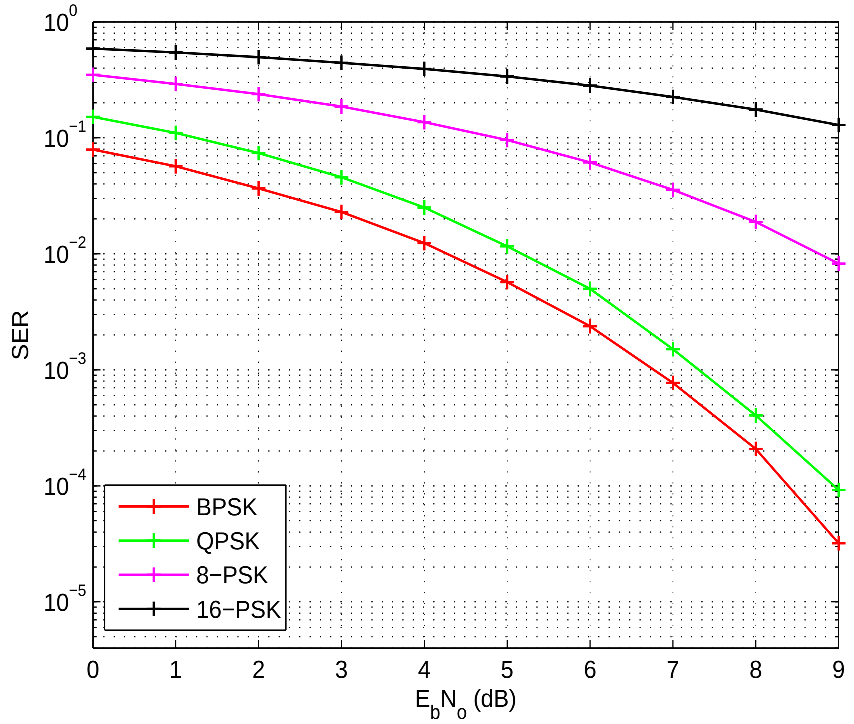

Figure 2.12: BER versus $E_b N_o$(dB) for 16-PSK modulation with non-idealities

### 2.3 Conclusion

The comparison in the BER and SER performance for $M = 2, 4, 8$ and 16 is as shown in Figure 2.13. We see that there is a degradation in the error performance as $M$ increases. This is due to the fact that, as $M$ becomes larger the constellation points come closer to each other rendering the decoding process more difficult. As a result, we require a higher SNR to decode the symbols.




(a) BER versus $E_b N_o$ (dB) for M-PSK modulation




(b) SER versus $E_b N_o$(dB) for M-PSK modulation

Figure 2.13: Comparison of BER and SER performance for M-ary PSK

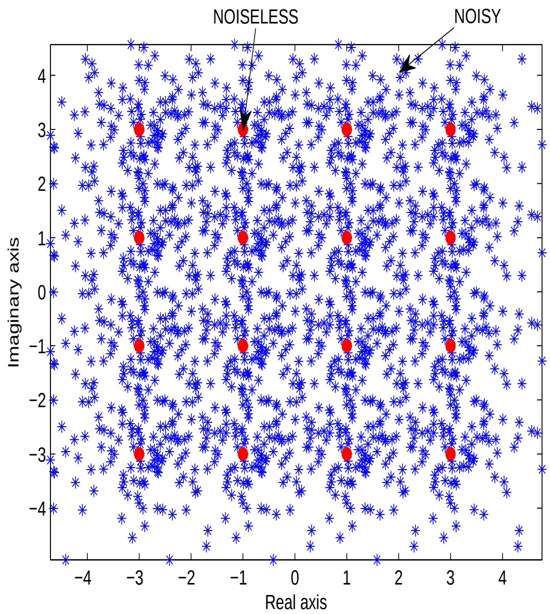
# 3 M-ARY QUADRATURE AMPLITUDE MODULATION (QAM)

## 3.1 ADDITIVE WHITE GAUSSIAN NOISE (AWGN)

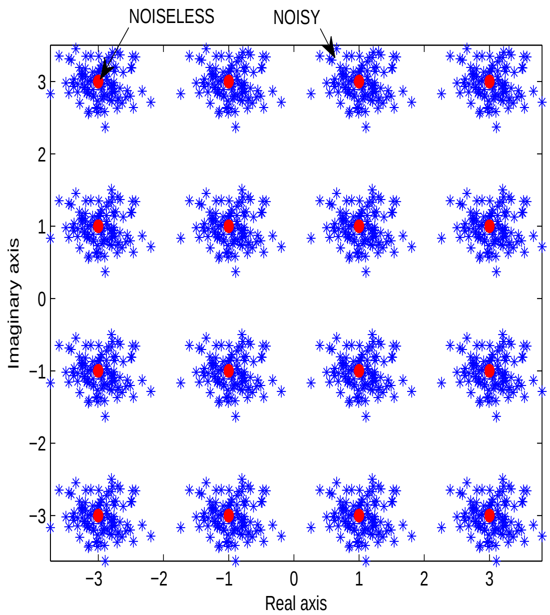
In this section, the performance of the $M$-ary QAM modulation in an Additive white Gaussian Noise (AWGN) channel is simulated and compared with the theoretical expression obtained for error probability.

### 3.1.1 16-QUADRATURE AMPLITUDE MODULATION (16-QAM)

The noisy constellation diagram for a 16-QAM modulation at two different values of $E_b N_o$ is as shown in Figure 3.1.



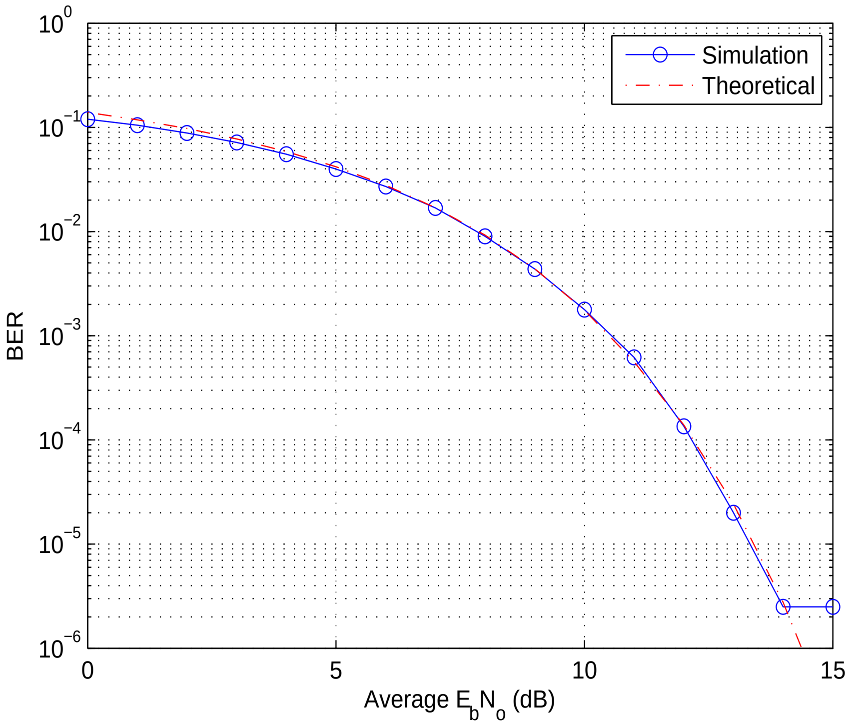

(a) Noisy constellation for 16-QAM at 0 dB



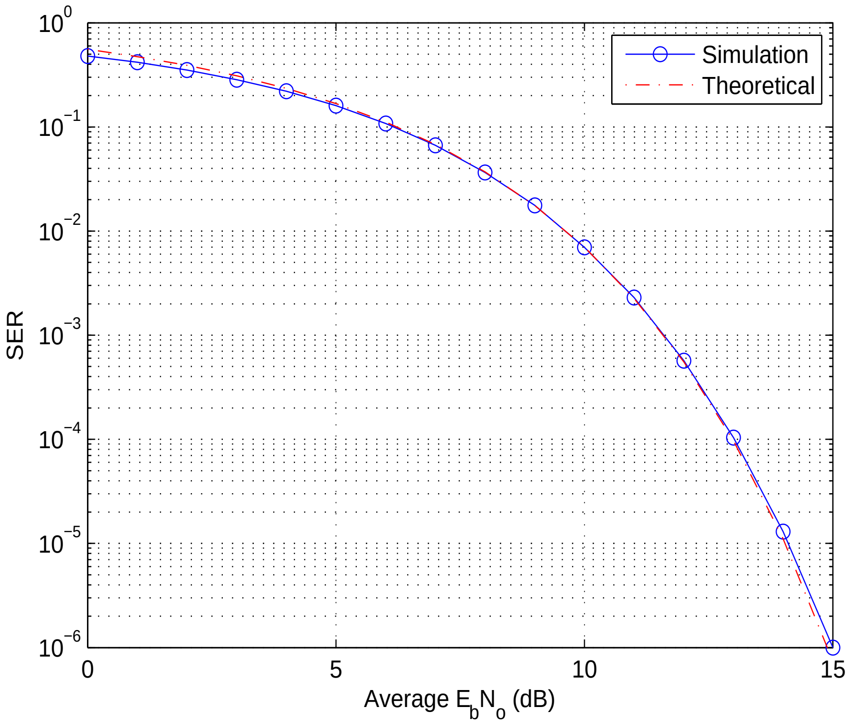

(b) Noisy constellation for 16-QAM at 10 dB

Figure 3.1: Constellation diagrams for 16-QAM

Figure 3.2 elucidates the BER and SER performance of the 16-QAM.




(a) BER versus $E_bN_o$(dB) for 16-QAM modulation

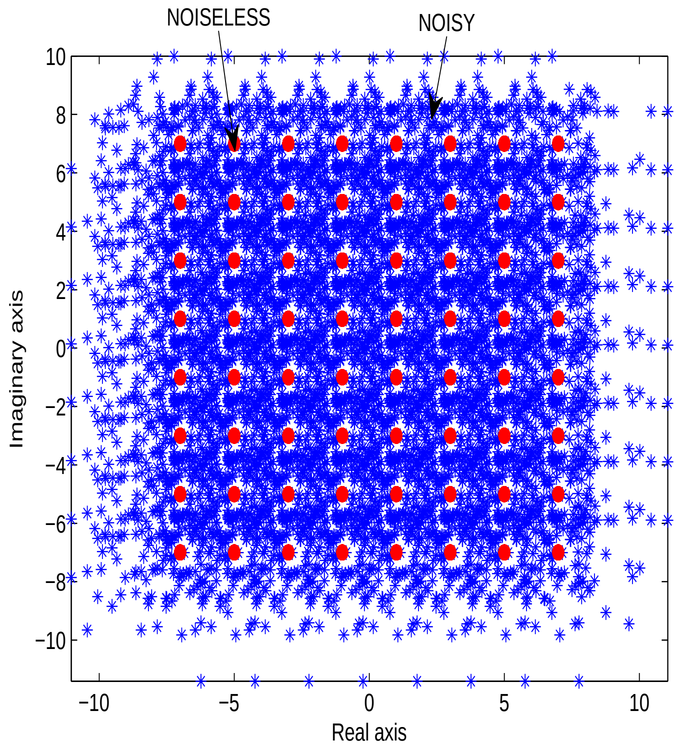



(b) SER versus $E_bN_o$(dB) for 16-QAM modulation

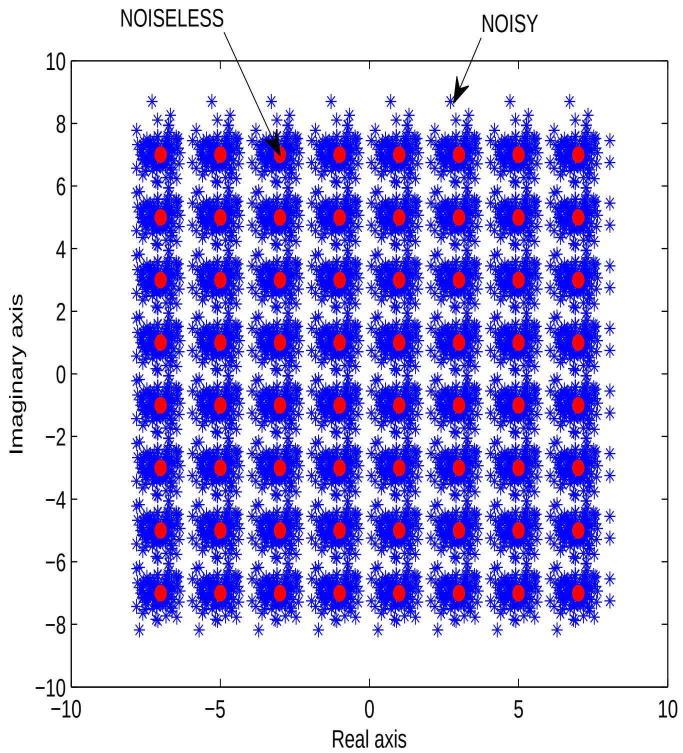
Figure 3.2: Error performance of 16-QAM

### 3.1.2 64-Quadrature Amplitude Modulation (64-QAM)

Figure 3.3 shows the noisy constellation of the 64-QAM at 0 dB and 10 dB.



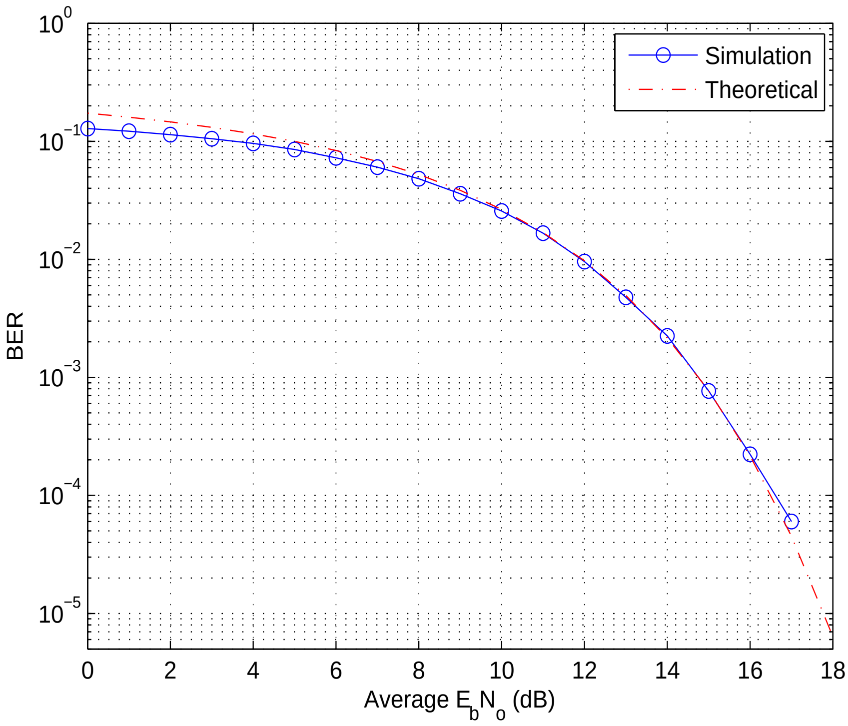

(a) Noisy constellation for 64-QAM at 0 dB



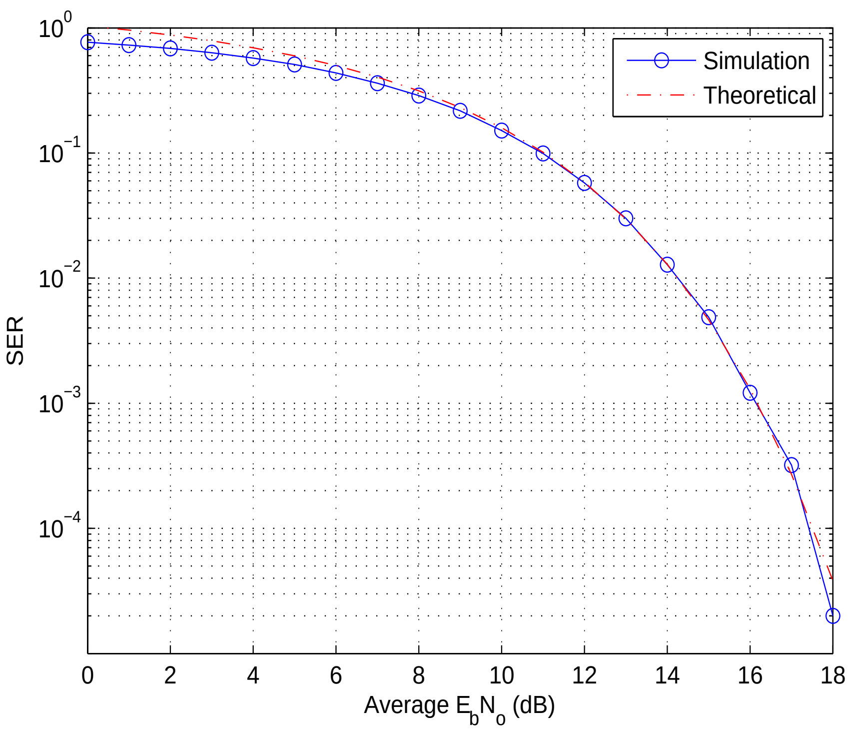

(b) Noisy constellation for 64-QAM at 10 dB

Figure 3.3: Constellation diagrams for 64-QAM

In Figure 3.4, the BER and SER performance of the 64-QAM is illustrated.




(a) BER versus $E_bN_o$(dB) for 64-QAM modulation




(b) SER versus $E_bN_o$(dB) for 64-QAM modulation

Figure 3.4: Error performance of 64-QAM modulation

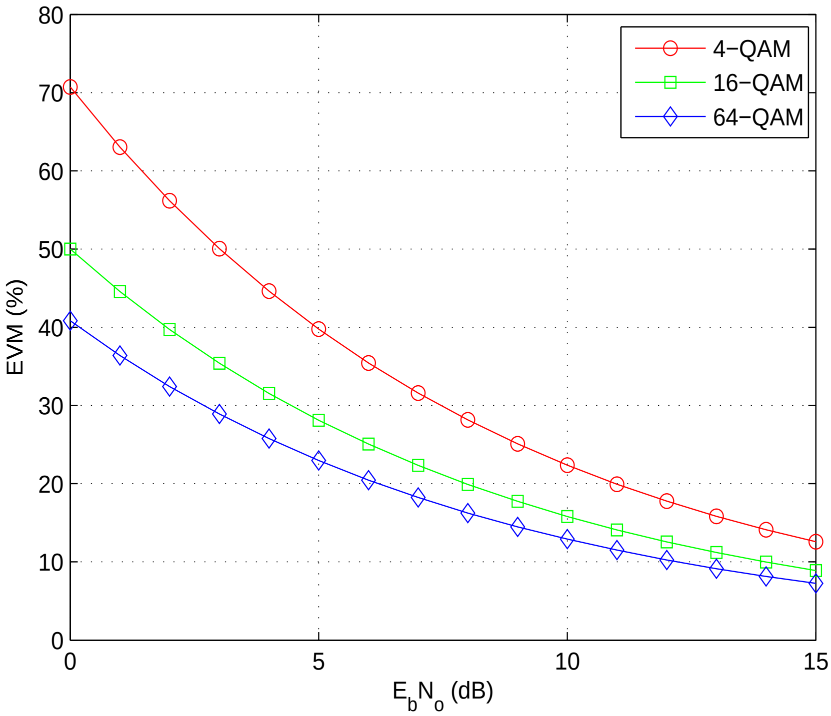



Figure 3.5: EVM (%) as a function of SNR

## 3.2 Error Vector Magnitude (EVM)

In this section, we calculate the error vector magnitude (EVM) as a function of $E_b N_o$. We know that the relationship between the two is given by,

$$EVM = \sqrt{\frac{1}{SNR}}$$

where, $SNR$ is the signal to noise ratio. The result is plotted as shown in Figure 3.5.

## 3.3 Conclusion

Figure 3.6 shows the variation in the error performance of $M$-ary QAM for different values of $M$. We notice that as the constellation size increases (for a fixed transmit power), the error performance deteriorates and is limited by the noise.

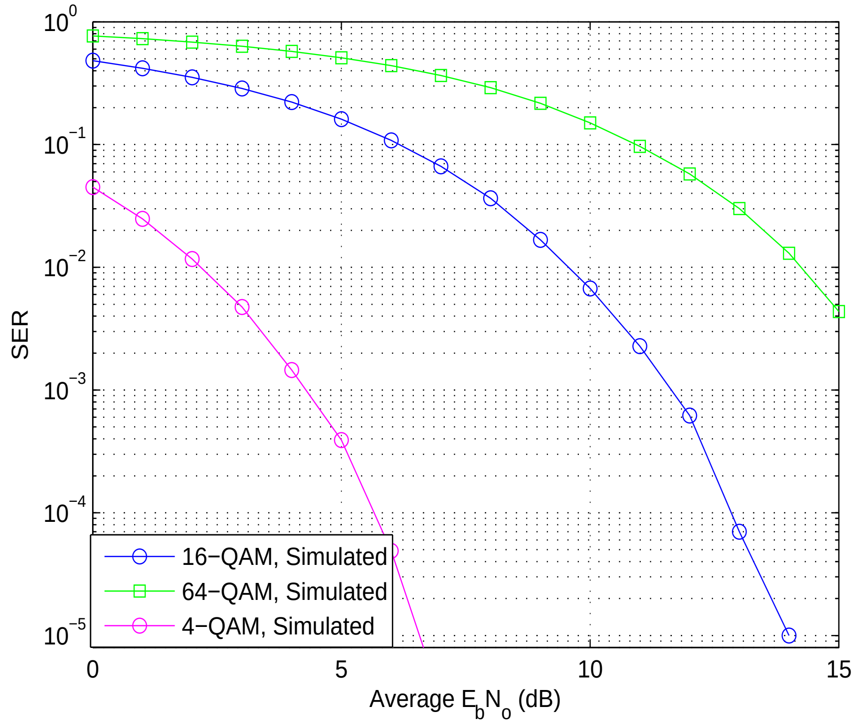



Figure 3.6: Comparison of BER performance for M-QAM with M=16, 64

# 4 SUMMARY

In this report, we have simulated the BER and SER performance of the MPSK and MQAM modulation in an AWGN channel. The MATLAB scripts for generating the plots are as shown in appendix A and B for MPSK and MQAM respectively. It is seen from the plots that, the curves obtained from Monte Carlo simulation are substantiated by the curves obtained through simulation of the theoretical expression for the error probabilities. Therefore, we can deduce the following inferences from the curves,

- For an MPSK modulation signal we see the the error performance is worsened considerably when increasing the constellation size. This is because as the points are restricted to lie on a circle (constant amplitude modulation), to pack more bits the phasor points have to be placed in close proximity to each other resulting in the requirement of a higher SNR to decode the symbol.
- Since the MPSK is a phase modulation technique it is highly susceptible to changes in the phase of the signal. This is evident in the plot with the non-idealities. Even a small change in the phase of the signal could lead to erroneous detection.
- In the next section, we simulated the performance of the MQAM signal. As it seen from the plots, that even in this case the BER increases for higher constellation sizes. However, this is due to noise. Thereby, requiring a higher SNR to overcome the effect of the noise.
- Next, we plotted the EVM as a function of SNR. It can be realized from the plot that EVM is inversely proportional to square root of the SNR. This is evident from the shape of the curve. Thus, this tells us that as we increase the size of the constellation for a given value of SNR the effect of EVM is reduced.

Appendix A: MPSK modulation Matlab Codes
BPSK modulation:

```
clc
clear all
it=250000;
EbNo=0:10;
sigma=sqrt(1./(2*10.^(0.1*EbNo)));

error = zeros(1,length(sigma));
BER_analytical=zeros(1,length(sigma));
SER_analytical=zeros(1,length(sigma));
const_x=zeros(1,it);
const_y=zeros(1,it);

for k = 1:length(sigma)
 for i=1:it
     %%%%%%%%% mapping of symbol %%%%%%%%%%%
    x=rand();

    if x <= 0.5
       symbol=1;
       A=[1 0];
    else
       symbol=2;
       A=[-1 0];
    end;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%% addition of noise %%%%%%%%%%%
    Inoise = sigma(k) * randn;
    Qnoise = sigma(k) * randn;
    Ircv = A(1) + Inoise;
    Qrcv = A(2) + Qnoise;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    B=[Ircv Qrcv];
    const_x(i)=B(1);
    const_y(i)=B(2);

    %%%%%%%% Symbol Detection %%%%%%%%%%%%%
    B_Mag=sqrt((Ircv*Ircv)+(Qrcv*Qrcv));
    C=dot(A,B);
    C=C/B_Mag;
    theta=acosd(C);
    if theta < 90
       error(k)=error(k);
    else error(k)=error(k)+1;
    end;
end;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%% BER and SER calculations %%%%%
    BER_analytical(k)=error(k)/it;
    SER_analytical(k)=error(k)/it;


    BER_theoretical(k)=0.5*erfc(1/(sqrt(2)*sigma(k)));
    SER_theoretical(k)=0.5*erfc(1/(sqrt(2)*sigma(k)));

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
end;
semilogy(EbNo,BER_analytical,'-red+','LineWidth',2);
hold on;
semilogy(EbNo,BER_theoretical,'-.blueo','LineWidth',2);
xlabel('EbNo in dB');
ylabel('BER in log scale');
title('BER v/s EbNo for BPSK');
legend('Simulation','Theoretical');
grid on;
figure;
semilogy(EbNo,SER_analytical,'-red+','LineWidth',2);
hold on;
semilogy(EbNo,SER_theoretical,'-.blueo','LineWidth',2);
xlabel('EbNo in dB');
ylabel('SER in log scale');
title('SER v/s EbNo for BPSK');
legend('Simulation','Theoretical');
grid on;
figure;
scatter(const_x,const_y,'*');
title('Constellation for BPSK at 20dB');
```

QPSK modulation:

```
clc
clear all
it=250000;
EbNo=-5:10;
sigma=sqrt(1./(4*10.^(0.1*EbNo)));

error = zeros(1,length(sigma));
BER_analytical=zeros(1,length(sigma));
SER_analytical=zeros(1,length(sigma));
const_x=zeros(1,it);
const_y=zeros(1,it);

for k = 1:length(sigma)
for i=1:it
     x=rand();

     %%%%%%%%% mapping of symbol %%%%%%%%%%%
     if x <= 0.25
        symbol=1;
        A=[0.707 0.707];
        else if x <= 0.5 && x > 0.25
            symbol=2;
            A=[-0.707 0.707];
            else if  x <= 0.75 && x > 0.5
                symbol = 3;
                A=[-0.707 -0.707];
                else
                   symbol = 4;
                   A=[0.707 -0.707];
                end
            end
     end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%% addition of noise %%%%%%%%%%%
    Inoise = sigma(k) * randn;
    Qnoise = sigma(k) * randn;
    Ircv = A(1) + Inoise;
```

```
    Qrcv = A(2) + Qnoise;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    B=[Ircv Qrcv];
    const_x(i)=B(1);
    const_y(i)=B(2);

    %%%%%%%% Symbol Detection %%%%%%%%%%%%%
    B_Mag=sqrt((Ircv*Ircv)+(Qrcv*Qrcv));
    C=dot(A,B);
    C=C/B_Mag;
    theta=acosd(C);
    if theta < 45
       error(k)=error(k);
    else error(k)=error(k)+1;
    end
end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%% BER and SER calculations %%%%%
    BER_analytical(k)=error(k)/(2*it);
    SER_analytical(k)=error(k)/(it);

    BER_theoretical(k)=0.5*erfc(1/(2*sigma(k)));
    SER_theoretical(k)=erfc(1/(2*sigma(k)));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end;
semilogy(EbNo,BER_analytical,'-red+','LineWidth',2);
hold on;
semilogy(EbNo,BER_theoretical,'-.blueo','LineWidth',2);
xlabel('EbNo in dB');
ylabel('BER in log scale');
title('BER v/s EbNo for QPSK');
legend('Simulation','Theoretical');
grid on;
figure;
semilogy(EbNo,SER_analytical,'-red+','LineWidth',2);
hold on;
semilogy(EbNo,SER_theoretical,'-.blueo','LineWidth',2);
xlabel('EbNo in dB');
ylabel('SER in log scale');
title('SER v/s EbNo for QPSK');
legend('Simulation','Theoretical');
grid on;
figure;
scatter(const_x,const_y,'*');
title('Constellation for QPSK at 20dB');
```

8-PSK modulation:

```
clc;clear;close all;

it=250000;
EbNo=-5:10;
sigma=1./sqrt(6*(10.^(EbNo./10)));

error = zeros(1,length(sigma));
BER=zeros(1,length(sigma));
const_x=zeros(1,it);
const_y=zeros(1,it);

for k = 1:length(sigma)
```

```
for i=1:it
    x=rand();
    %%%%%%%%% mapping of symbol %%%%%%%%%%%
    if x <= 0.125
        symbol=1;                                   % symbol 1 = 000
        A=[1 0];
    else if x <= 0.25 && x > 0.125
            symbol=2;                               % symbol 2 = 010
            A=[0.707 0.707];
        else if x <= 0.375 && x > 0.25
                symbol = 3;                         % symbol 3 = 110
                A=[0 1];
            else if x <= 0.5 && x > 0.375
                    symbol = 4;                     % symbol 4 = 100
                    A=[-0.707 0.707];
                else if  x <= 0.625 && x > 0.5
                         symbol = 5;                % symbol 5 = 101
                         A=[-1 0];
                    else if x <= 0.75 && x > 0.625
                            symbol = 6;             % symbol 6 = 111
                            A=[-0.707 -0.707];
                        else if x <= 0.875 && x > 0.75
                                symbol = 7;         % symbol 7 = 011
                                A=[0 -1];
                            else symbol=8;          % symbol 8 = 001
                                 A=[0.707 -0.707];
                            end
                        end
                    end
                end
            end
        end
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%% addition of noise %%%%%%%%%%%
    Inoise = sigma(k) * randn;
    Qnoise = sigma(k) * randn;
    Ircv = A(1) + Inoise;
    Qrcv = A(2) + Qnoise;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

     B=[Ircv Qrcv];
     const_x(i)=B(1);
     const_y(i)=B(2);

     %%%%%%%% Symbol Detection %%%%%%%%%%%%%
     B_Mag=sqrt((Ircv*Ircv)+(Qrcv*Qrcv));
     C=dot(A,B);
     C=C/B_Mag;
     theta=acosd(C);
     if theta < 22.5
         error(k)=error(k);
     else error(k)=error(k)+1;
     end

     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end;
    %%%%%%%% BER and SER calculations %%%%%
    BER_analytical(k)=error(k)/(3*it);
    BER_theoretical(k)=(1/3)*erfc(0.27/(sigma(k)));
```

```
    SER_analytical(k)=error(k)/(it);
    SER_theoretical(k)=erfc(0.27/(sigma(k)));

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
semilogy(EbNo,BER_analytical,'-red+','LineWidth',2);
hold on;
semilogy(EbNo,BER_theoretical,'-.blueo','LineWidth',2);
xlabel('EbNo in dB');
ylabel('BER in log scale');
title('BER v/s EbNo for 8PSK');
legend('Simulation','Theoretical');
grid on;
figure;
semilogy(EbNo,SER_analytical,'-red+','LineWidth',2);
hold on;
semilogy(EbNo,SER_theoretical,'-.blueo','LineWidth',2);
xlabel('EbNo in dB');
ylabel('SER in log scale');
title('SER v/s EbNo for 8-PSK');
legend('Simulation','Theoretical');
grid on;
figure;
scatter(const_x,const_y,'*');
title('Constellation for 8-PSK');
```

16-PSK modulation:

```
clc;clear;close all;

it=250000;
EbNo=-5:20;
sigma=1./sqrt(8*(10.^(EbNo./10)));

error = zeros(1,length(sigma));
BER=zeros(1,length(sigma));
BER_analytical=zeros(1,length(sigma));
BER_theoretical=zeros(1,length(sigma));
const_x=zeros(1,it);
const_y=zeros(1,it);

for k = 1:length(sigma)
for i=1:it
    x=rand();

    %%%%%%%%% mapping of symbol %%%%%%%%%%%
    if x <= 0.0625
        symbol=1;
        A=[1 0];
    else if x <= 0.125 && x > 0.0625
            symbol=2;
            A=[0.923 0.38];
        else if x <= 0.1875 && x > 0.125
                symbol = 3;
                A=[0.707 0.707];
            else if x <= 0.25 && x > 0.1875
                    symbol = 4;
                    A=[0.38 0.923];
                else if  x <= 0.3125 && x > 0.25
                         symbol = 5;
                         A=[0 1];
                    else if x <= 0.375 && x > 0.3125
                            symbol = 6;
                            A=[-0.38 0.923];
                        else if x <= 0.4375 && x > 0.375
                                symbol = 7;
                                A=[-0.707 0.707];
                            else if x <= 0.5 && x > 0.4375
                                    symbol=8;
                                    A=[-0.923 0.38];
                                else if x <= 0.5625 && x > 0.5
                                        symbol=9;
                                        A=[-1 0];
                                    else if x <= 0.625 && x > 0.5625
                                            symbol=10;
                                            A=[-0.923 -0.38];
                                        else if x <= 0.6875 && x > 0.625
                                                symbol=11;
                                                A=[-0.707 -0.707];
                                            else if x <= 0.75 && x > 0.6875
                                                    symbol=12;
                                                    A=[-0.382 -0.923];
                                                else if x <= 0.8125 && x > 0.75
                                                        symbol=13;
                                                        A=[0 -1];
                                                    else if x <= 0.875 && x > 0.8125
                                                            symbol=14;
                                                            A=[0.382 -0.923];
```

```
                                                    else if x <= 0.9375 && x > 0.875
                                                                symbol=15;
                                                                A=[0.707 -0.707];
                                                            else symbol=16;
                                                                A=[0.923 -0.38];
                                                            end
                                                        end
                                                    end
                                                end
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%% addition of noise %%%%%%%%%%%
    Inoise = sigma(k) * randn;
    Qnoise = sigma(k) * randn;
    Ircv = A(1) + Inoise;
    Qrcv = A(2) + Qnoise;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        B=[Ircv Qrcv];
        const_x(i)=B(1);
        const_y(i)=B(2);

        %%%%%%%% Symbol Detection %%%%%%%%%%%%%
        B_Mag=sqrt((Ircv*Ircv)+(Qrcv*Qrcv));
        C=dot(A,B);
        C=C/B_Mag;
        theta=acosd(C);
        if theta < 11.25
            error(k)=error(k);
        else error(k)=error(k)+1;
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end;
    %%%%%%%% BER and SER calculations %%%%%
    BER_analytical(k)=error(k)/(4*it);
    BER_theoretical(k)=.25*erfc(.137/(sigma(k)));

    SER_analytical(k)=error(k)/(it);
    SER_theoretical(k)=erfc(.137/(sigma(k)));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
semilogy(EbNo,BER_analytical,'-red+','LineWidth',2);
hold on;
semilogy(EbNo,BER_theoretical,'-.blueo','LineWidth',2);
xlabel('EbNo in dB');
ylabel('BER in log scale');
title('BER v/s EbNo for 16PSK');
legend('Simulation','Theoretical');
grid on;
figure;
semilogy(EbNo,SER_analytical,'-red+','LineWidth',2);
hold on;
```

```
semilogy(EbNo,SER_theoretical,'-.blueo','LineWidth',2);
xlabel('EbNo in dB');
ylabel('SER in log scale');
title('SER v/s EbNo for 16-PSK');
legend('Simulation','Theoretical');
grid on;
figure;
scatter(const_x,const_y,'*');
title('Constellation for 16-PSK');
```

Appendix B: MQAM modulation Matlab Codes

```
function modu1=qam_mod_ns(M)

%The output of this function gives the constellation points

if(M==4)
    amp=[1 -1];
elseif(M==8)
    amp=[-3 -1 1 3];
    amp_1=[1 -1];
elseif(M==16)
    amp=[1 -1 3 -3];
elseif(M==64)
    amp=[1 -1 3 -3 5 -5 7 -7];
end

count=1;
modu1=[];

if(M~=2)
    amp1=amp;
    amp_mat(1,:)=amp1;
end

if(M==8)

    for p=1:length(amp)
        for q=1:length(amp_1)
            modu(p,q)=amp(p)+1j*amp_1(q);
        end
        modu1=[modu1 modu(p,:)];
    end

elseif(M~=2)
    for j=2:length(amp)
        count=count+1;
           if(count<=length(amp))
               temp=amp1(1);
               amp1(1)=amp1(count);
               amp1(count)=temp;
           end
           amp_mat(j,:)=amp1;
           amp1=amp;

           if(count>length(amp))
               count=1;
           end
    end

    for m=1:size(amp_mat,1)
        for n=1:size(amp_mat,2)
```

```
                modu(m,n)=amp_mat(m,1)+1j*amp_mat(m,n);
            end
            modu1=[modu1 modu(m,:)];
        end
end
    if(M==2)
        modu1=[-1 1];
    end
end
```

```
function rx_est=ml_detector(modu,rx)

   %ML detection

   min=500;
    for i=1:length(modu)
        dist=(rx-modu(i))*(conj(rx)-conj(modu(i)));
        if(dist<min)
            rx_est=modu(i);
            min=dist;
        end
    end
end
```

```
clc
clear all

M=64;
m=0:M-1;

k=log2(M);

%Function which generates the constellation points for different M
modu=qam_mod_ns(M);

%Average symbol energy
Es=trace(modu*modu')/M;

SNR=0:15;
snr=(10.^(SNR./10));

iter=100000;

err=[];

for i=1:length(snr)

    SNR(i)

    error=0;
    sigma=sqrt(Es/(2*k*snr(i)));

    for it=1:iter

        r=randi([1 M],1,1);
        tx=modu(r);

        n=(randn(1,1)+1j*randn(1,1));

        rx=tx+(sigma*n);
```

```
        %Function which performs the ML detection
        rx_est=ml_detector(modu,rx);
        r1=find(rx_est==modu);

        error=error+sum(m(r1)~=m(r));
    end

    %Average error calculation
    err=[err error/(iter)];

    if(M==16)
        %16-QAM theoretical BER
        err_th(i)=(3/(2*k))*erfc(sqrt((k*snr(i))/10));
    elseif(M==64)
        %64-QAM theoretical BER
        x=sqrt((3*k*snr(i))/(M-1));
        err_th(i)=(4/k)*(1-1/sqrt(M))*(1/2)*erfc(x/sqrt(2));
    end
end

semilogy(SNR,err,'g',SNR,err_th,'r')
```