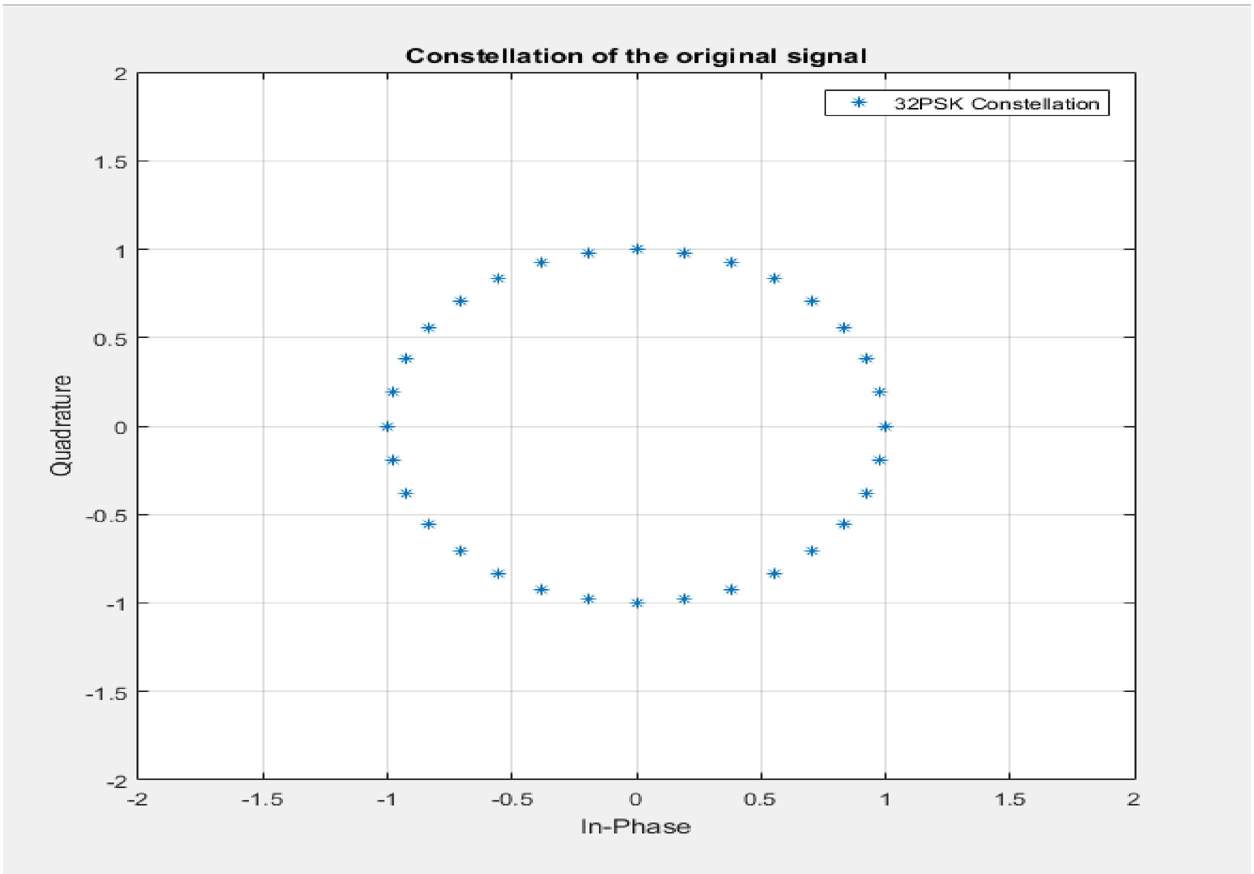
ECE451: Communication systems (2)
Faculty of Engineering
Ain Shams University
4th Year ECE
Fall 2019

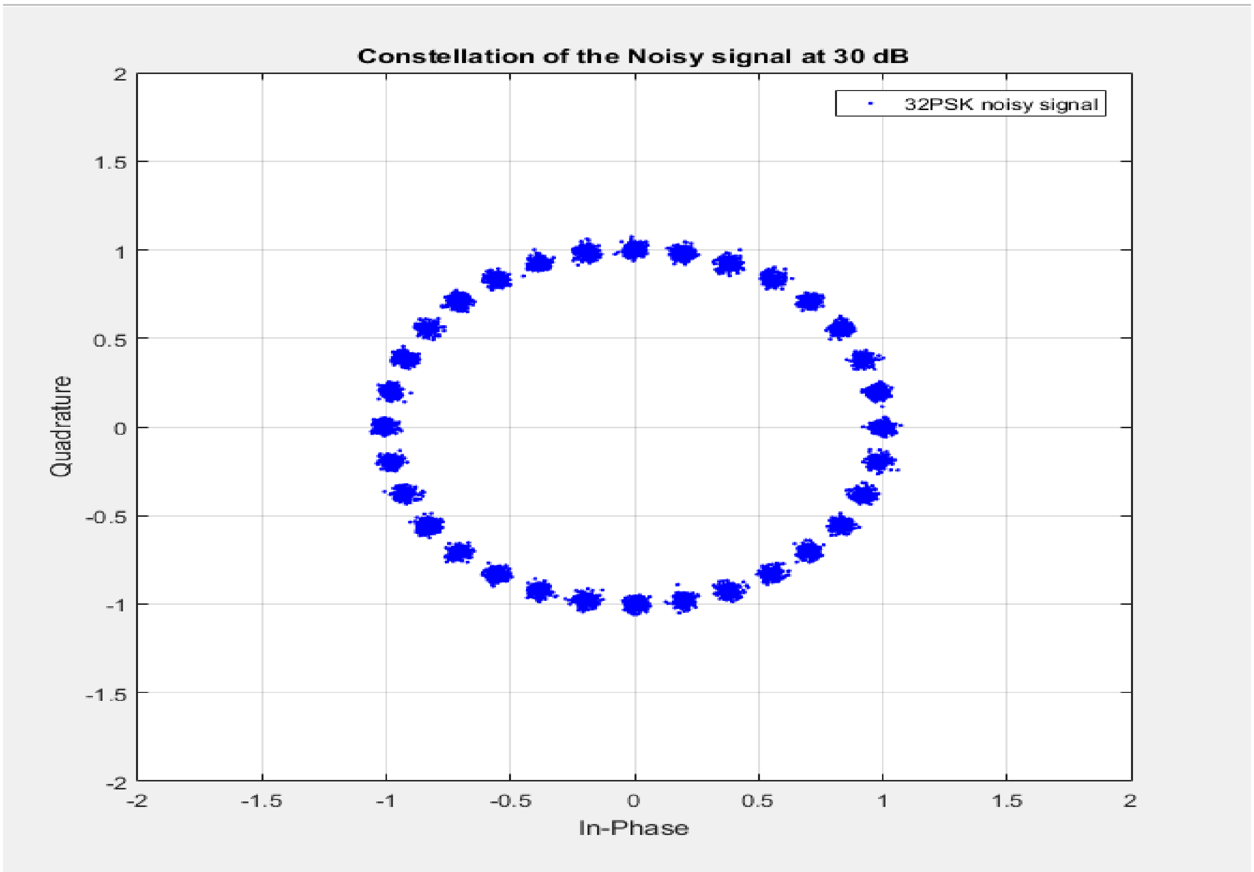
Course Mini-Project: Implementation of a 32MPSK Modulation & Demodulation	
Group No :18	
Student Name:	**Report Mark**
1. محمود رمزي محمد السعيد	
2.مصطفي اخلاق محمد	
3.طه احمد طه	
4.خالد محمد السيد	
5.محمود عبدالمرضي محمود	
Delivery Date : Thursday @(1:00)PM	

Constellation diagram of the signals:

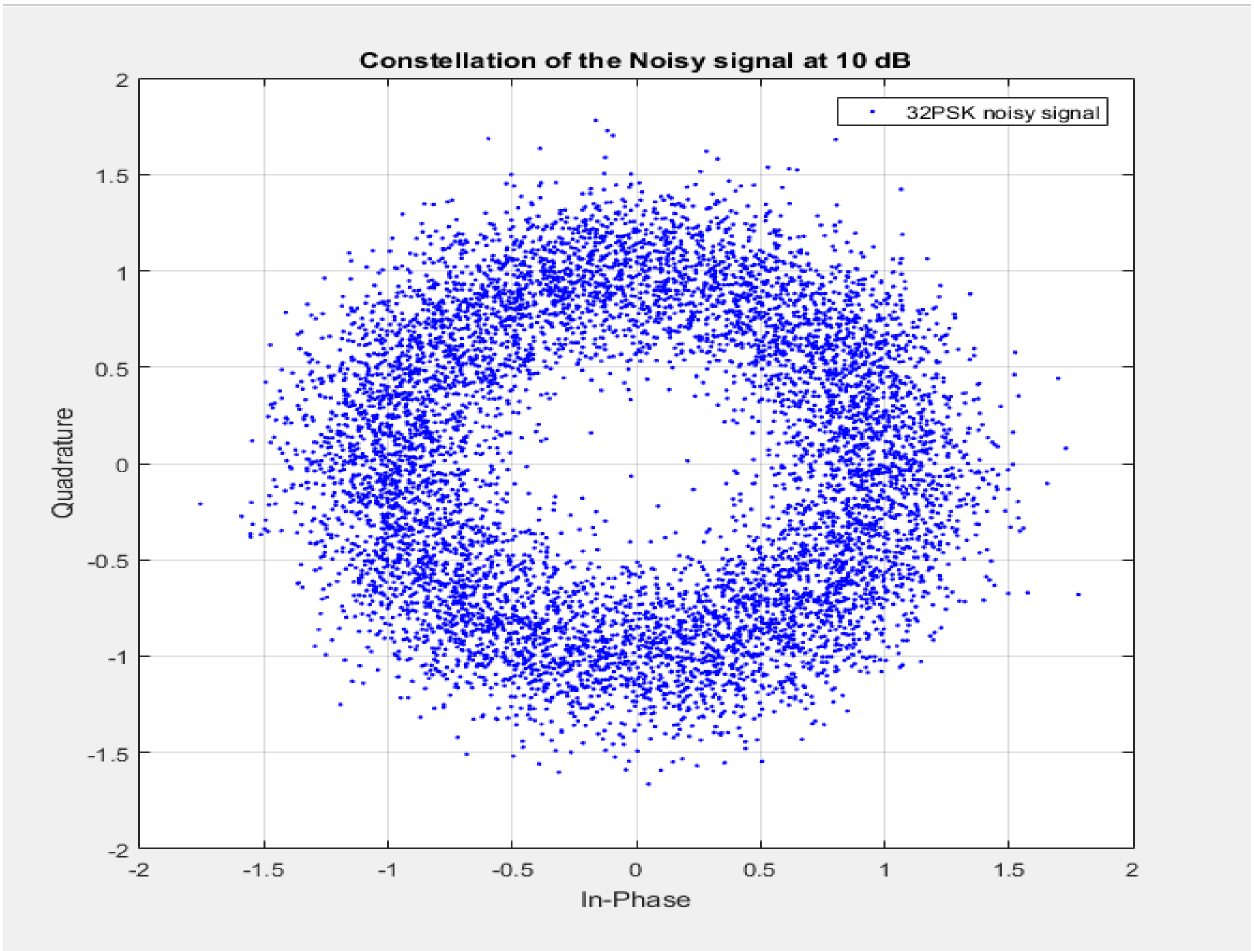
For original signal:



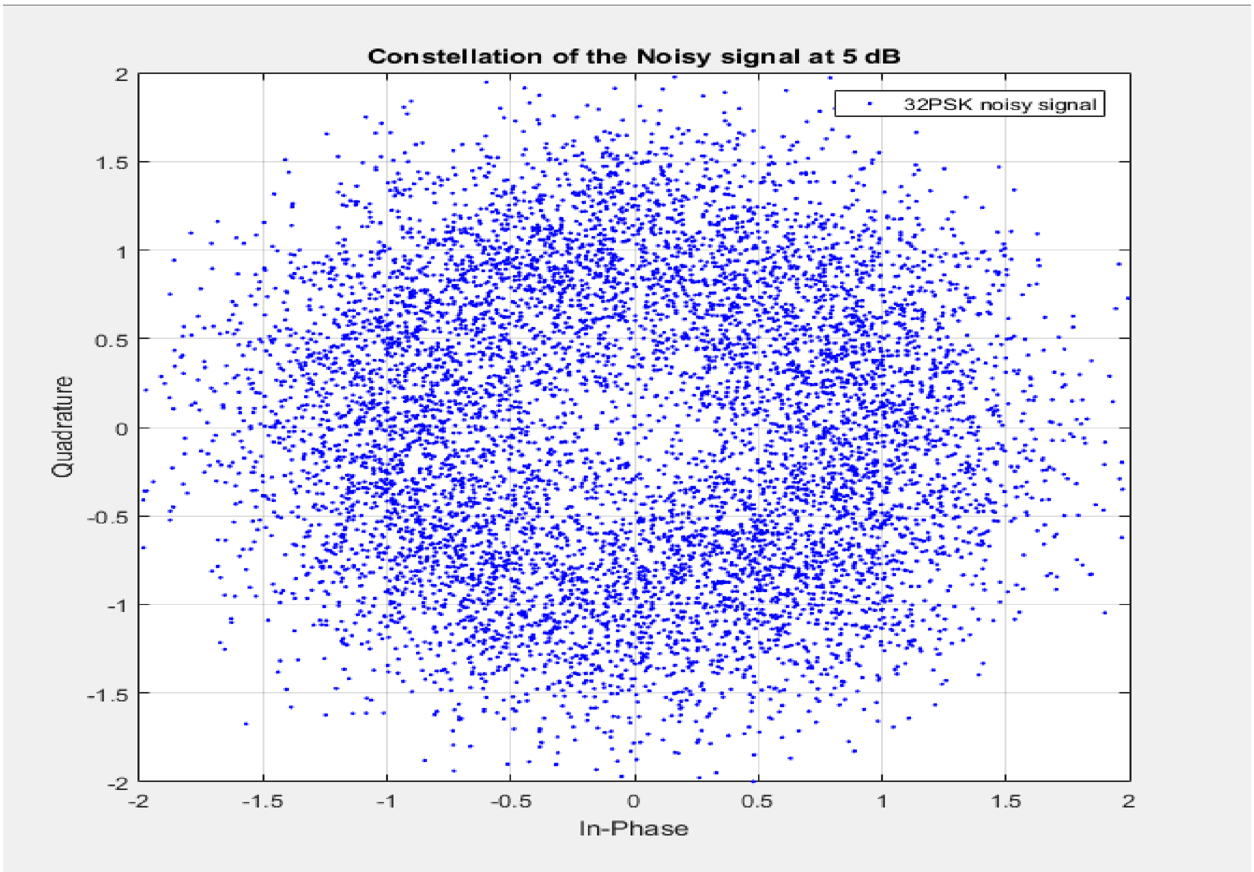
@ 30db:



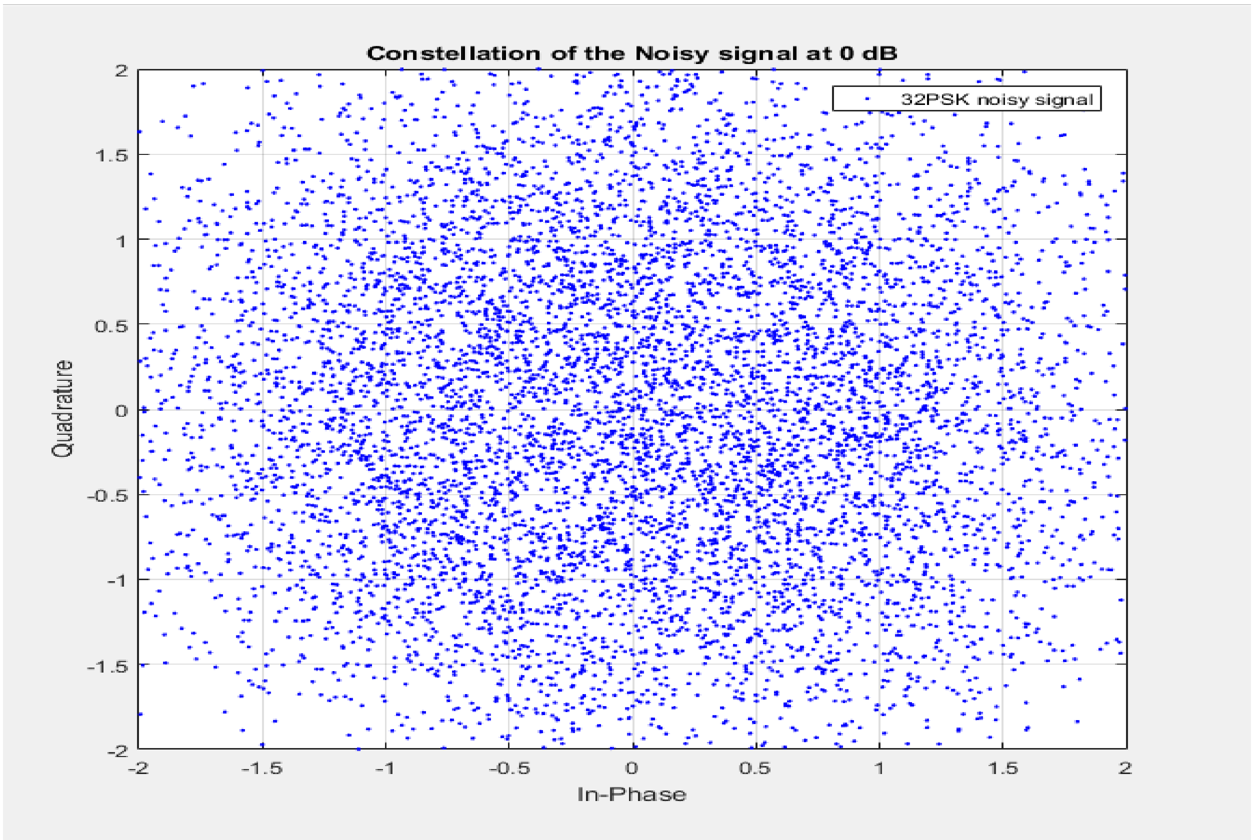
@ 10db:


Constellation of the Noisy signal at 10 dB
32PSK noisy signal
Quadrature
In-Phase
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2

@ 5db:

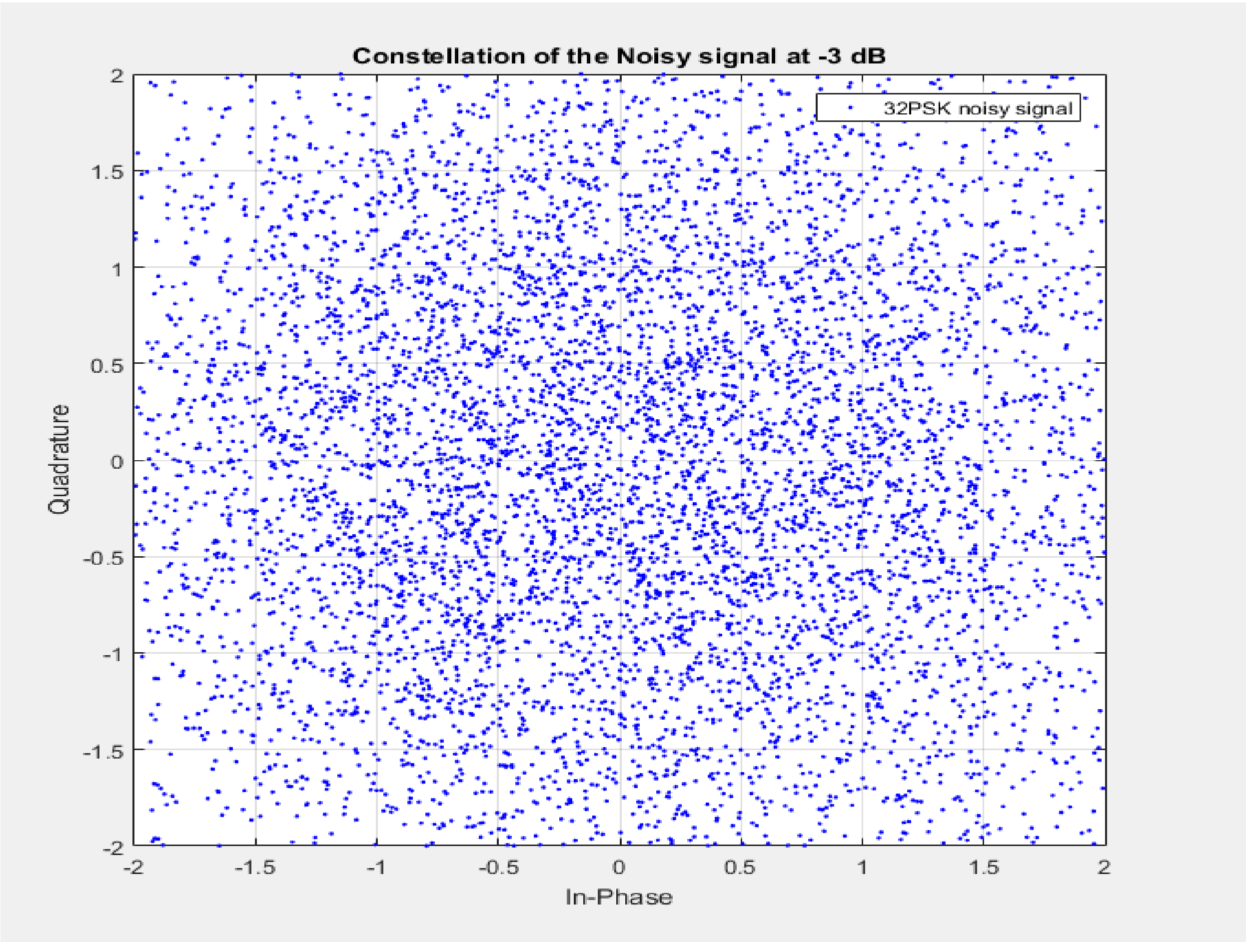

Constellation of the Noisy signal at 5 dB
32PSK noisy signal
Quadrature
In-Phase
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2

@ 0db:



Constellation of the Noisy signal at 0 dB
32PSK noisy signal
Quadrature
In-Phase
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2

@ -3 db:

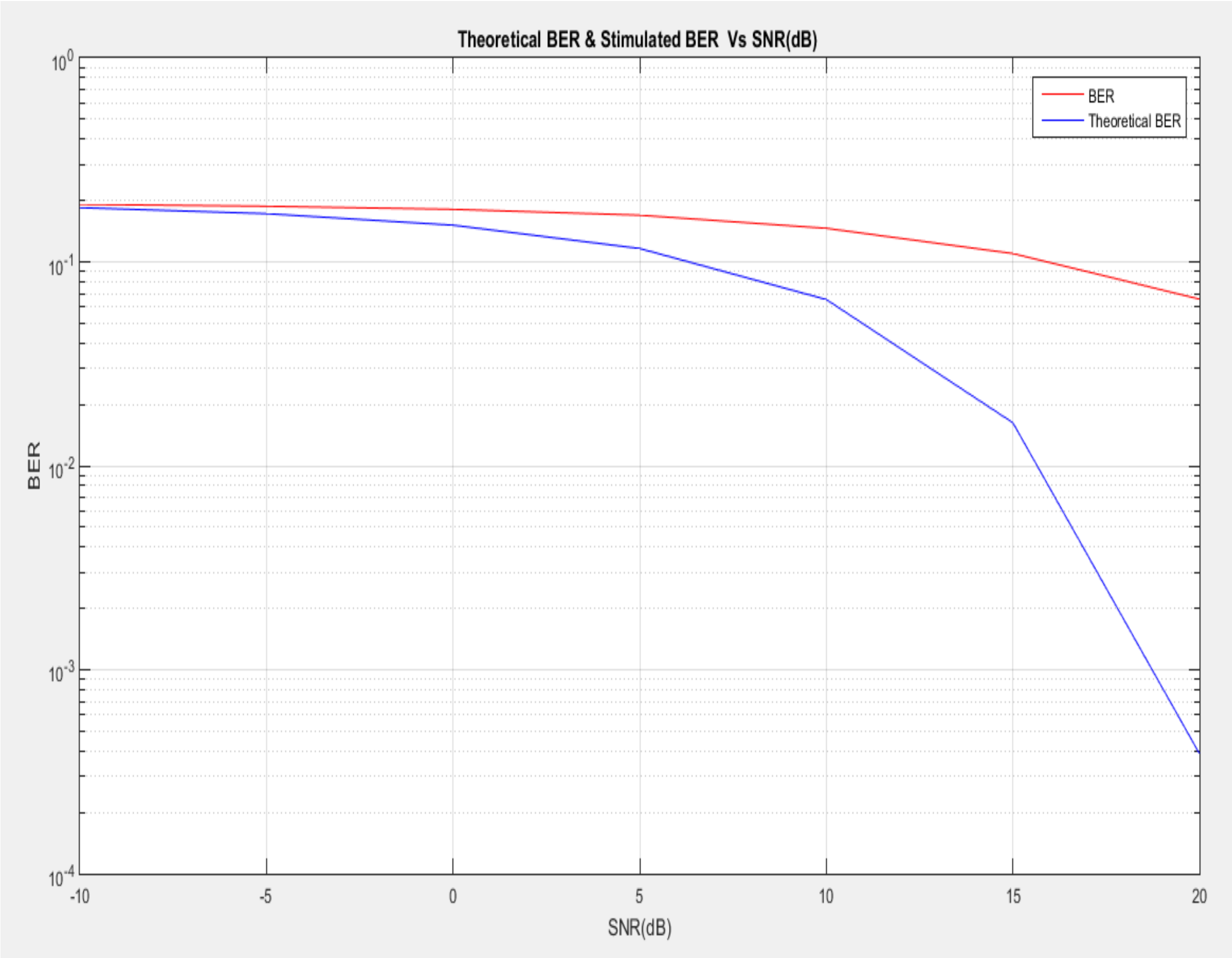


Constellation of the Noisy signal at -3 dB
32PSK noisy signal
Quadrature
In-Phase
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2

Comment:

We can observe that the lower SNR value in db the more the constellation be noisy, as the SNR is the ration between the E of signal &E of noise so by making its value less the noise effect dominates.

Theortical BER & Simulated BER vs SNR:



Main Code:

```
close all;
clc;

%% Generating Input Bits And Shaping it

N_symbol = 32; % Number of Symbols in MPSK
N_bits = log2(N_symbol); % Number of Bits/Symbol
x_stream = randi([0 1],[2000 5]);% generating streams of 1's and 0's by rounding random numbers to nearest 1 or 0 ,
%Mtx size = 2000 x 5
x_decimal = bi2de(x_stream,'left-msb'); % Converting Binary symbols to decimal values so we can map it later

%% Mapping the symbols and drawing constellation

y=MPSK_32_MOD(x_decimal); % mapping symbols to constellation

%% Drawing of constellation of Original signal after mapping without effect of channel noise
scatterplot(y,1,0);   %produces a scatter plot for the signal y every 1 value of the signal,starting from the 0 off
grid on
axis([-2 2 -2 2]);
title('Constellation of the original signal')
legend('32PSK Constellation')
```

```
%% AWGN channel effect

% 1. For SNR= 30 dB
scatterplot(White_noise(y,30)); %produces a scatter plot for the signal y after making its SNR dwn to 30db
grid on
axis([-2 2 -2 2]); legend('32PSK noisy signal');
title('Constellation of the Noisy signal at 30 dB')

% 2.For SNR= 10 dB

scatterplot(White_noise(y,10));  %produces a scatter plot for the signal y after making it SNR dwn to 10db
legend('32PSK noisy signal');
axis([-2 2 -2 2]);  grid on
title('Constellation of the Noisy signal at 10 dB')

% 3.For SNR= 5 dB

scatterplot(White_noise(y,5));  %produces a scatter plot for the signal y after making its SNR dwn to 5db
legend('32PSK noisy signal');
axis([-2 2 -2 2]);  grid on
title('Constellation of the Noisy signal at 5 dB')
```

```
% 4.For SNR= 0 dB
scatterplot(White_noise(y,0));  %produces a scatter plot for the signal y after making its SNR dwn to 0db
legend('32PSK noisy signal');
axis([-2 2 -2 2]);  grid on
title('Constellation of the Noisy signal at 0 dB')

% 5.For SNR= -3 dB

c1=scatterplot(White_noise(y,-3));  %produces a scatter plot for the signal y making its SNR dwn to -3db
legend('32PSK noisy signal');
axis([-2 2 -2 2]);  grid on
title('Constellation of the Noisy signal at -3 dB')
%% Demapped signal
Y_Rx        = MPSK_32_DMOD(y);     %Demodulation of the original signal
Y_Rx_SNRn10 = MPSK_32_DMOD(White_noise(y,-10));  %Demodulation of the sig after adding  noise from channel to it
Y_Rx_SNRn5  = MPSK_32_DMOD(White_noise(y,-5));   % with different values that makes its SNR dwn
Y_Rx_SNR0   = MPSK_32_DMOD(White_noise(y,0));    % to different values too=-10,-5,0,5,...20 db
Y_Rx_SNR5   = MPSK_32_DMOD(White_noise(y,5));
Y_Rx_SNR10  = MPSK_32_DMOD(White_noise(y,10));
Y_Rx_SNR15  = MPSK_32_DMOD(White_noise(y,15));
Y_Rx_SNR20  = MPSK_32_DMOD(White_noise(y,20));
```

```
%% BER Calculations
SNR.dB = -10:5:20;  %preparing the SNR axis to be used in plots

BER = [SER(x_decimal,Y_Rx_SNRn10) SER(x_decimal,Y_Rx_SNRn5) ...  %form a vector for the the symbol error rate
    SER(x_decimal,Y_Rx_SNR0) SER(x_decimal,Y_Rx_SNR5) ...         %of the RX demodulation signals in different
    SER(x_decimal,Y_Rx_SNR10) SER(x_decimal,Y_Rx_SNR15) ...       %SNR levels then divide it by #bits/symbol
    SER(x_decimal,Y_Rx_SNR20)]/N_bits ;                          %to get the bit error rate for easch signal
SNR.lin= 10.^(SNR.dB/10);     %convert the SNR  from db into decimal
BER_T = (2/log2(N_symbol))*qfunc(sqrt(2*SNR.lin*log2(N_symbol))*sin(pi/32));
%this is the theoretical relation for the Bit Error Rate of the MPSK-->M=32
figure
semilogy(SNR.dB , BER,'red')
title('BER Vs SNR(dB)')
legend('BER')
xlabel('SNR(dB)')
ylabel('BER')
grid on
hold on;
semilogy(SNR.dB , BER_T,'blue' )
title('Theoretical BER & Stimulated BER  Vs SNR(dB) ')
legend('BER','Theoretical BER')
xlabel('SNR(dB)')
ylabel('BER')
grid on
```

Functions:

1-Mapper

```
%% This is a function that perform mapping of symbols on constellation for 32PSK Modulation techinque

function mappedSymbol=MPSK_32_MOD(symbol)

    mappedSymbol=cos((2*pi/32)*symbol)+1i*sin((2*pi/32)*symbol);
  %Symbole eqn for the MPSK
end
```

2-AWGN channel:

```
%% This is a function that simulating the channel where awgn noise is added

function N = White_noise(Signal,SNR_dB)
rng('default'); % to geneate the same random values of noise in every time
L = length(Signal); %length of Signal
SNR = 10^(SNR_dB/10); %SNR to linear scale
Esym = sum(abs(Signal).^2)/(L); %Calculate actual symbol energy
N0 = Esym/SNR; %Find the noise spectral density
noiseSigma=sqrt(N0/2); %Standard deviation for AWGN Noise
n = noiseSigma*(randn(L,1)+1i*randn(L,1));%computed noise
N = Signal + n; %received signal
end
```

3-De-Mapper:

```
%% This is a function that perform demapping of symbols on constellation for 32PSK Modulation techinque

function [D]= MPSK_32_DMOD(y)

D=mod(angle(y),2*pi);   %returns the remainder after division the phase of Rx signal by 2pi
g = 2*pi/32;  %the distance or phase shift btn 2 successive symboles

for i=1:length(y)  %looping till reach the certain symbole that has a phase btn nxt and previous symboles
    for x = 0:31
        D(and(D >=  x*g - g/2, D < x*g + g/2)) = x;
    end
end
end
```

4-SER:

```
%% This function gets the symbol error rate
function [H]=SER(x,y)
n=0;  %just a counter
for i=1:length(x)
    if x(i) == y(i)  %checks if the original sig matched the RX deomodulated sig
        n=n+1;
    end
end
H=(length(x)-n)/length(x);
end
```

Formulas:

1) #bits/symbole=Log2(#symboles)

$$
\begin{aligned}
s_i(t) &= \sqrt{2P}\cos\left[2\pi f_c t + \frac{2\pi}{M} i\right] p_T(t) \quad 0 \leq t \leq T \\
&= A_{c,i}\sqrt{\frac{2}{T}}\cos(2\pi f_c t)p_T(t) - A_{s,i}\sqrt{\frac{2}{T}}\sin(2\pi f_c t)p_T(t) \\
&= A_{c,i}\phi_0(t) + A_{s,i}\phi_1(t)
\end{aligned}
$$

2) BER=SER/(#bits per symbole)
3) BER_T = (2/log2(#bits/sym))*qfunc(sqrt(2*root(E/N)*log2(#bits/sym))*sin(pi/32))