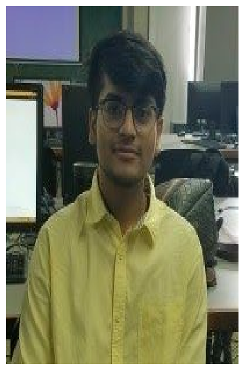
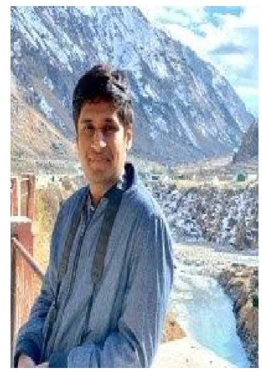
PSK Modulation-Demodulation

Efficiency and reliability

Martand Javia (AU1841064)	Namit Shah (AU1841067)	Yashvi Gandhi (AU1841033)	Harvish Jariwala (AU1841050)

(Group - 11)

ECE210 - Analogue and Digital Communications

Course Coordinator: Prof. Ashok Ranade

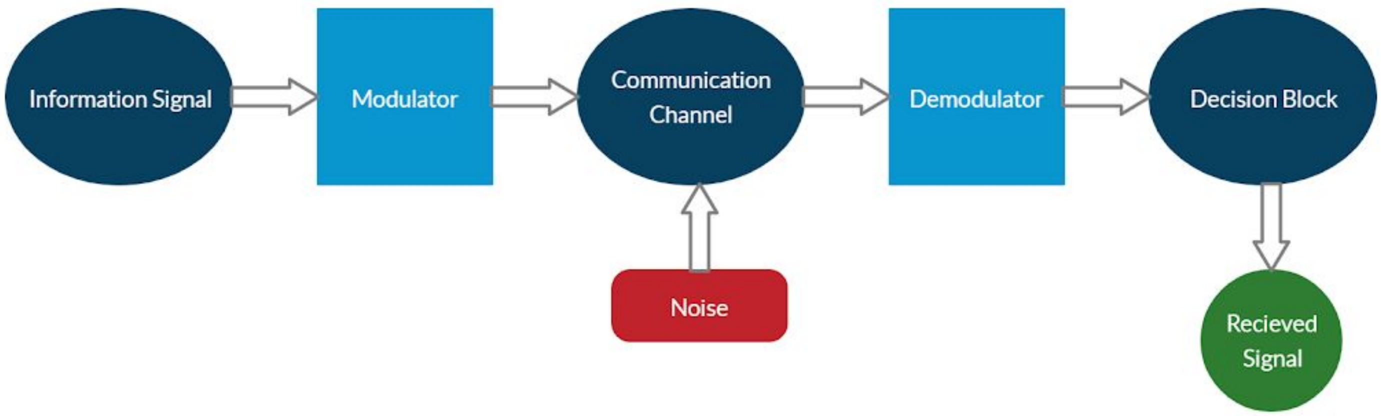
Winter 2020

All simulation files are available here : [Project Files]

Synopsis

- In Wireless Communication, data can be sent via signals. But to do so, the signal needs to be of a sufficiently higher frequency and normally, information signals don't have such higher frequency. So to overcome this problem, this information signal is embedded in the "Carrier signal". Since the signals we send are electromagnetic waves and are sinusoidal, information can be embedded in either frequency, phase or amplitude. In this project, we have focused on embedding it in phase. This process is called Phase Shift Keying.
- Phase Shift Keying is highly efficient and generally compatible with communication of data. Here, the information signal is enveloped with the phase of the carrier signal for transmission of data. Several coding techniques are applied for the efficient working of a communication system.
- PSK is widely used in wireless LANs, RFID and Bluetooth communication.
- BPSK and QPSK are one of them which are explored in this project.
- The project simulates a communication system using both BPSK and QPSK. It has the following flow :
    1. The modulating signal is randomly generated. It is then modulated separately using both BPSK and QPSK methods.
    2. This modulated signal is passed through a communication channel which embeds white noise.
    3. The signal is finally received and decoded back to its original form.

Block Diagram



Note : As we are doing simulations for both BPSK and QPSK the modulator and demodulator blocks changes on the basis of the method of the modulation-demodulation.

Working

- First of all, random information data is generated using a random signal generator. Then the information signal has to be modulated for the sake of transmission. Before modulation, the information signal is converted to BPSK and QPSK forms. In BPSK, 0 value accounts for phase difference of π, while 1 maps with 0 phase change. This converts the signal to -1/1 form instead of 0/1.

bit	Θ	cosΘ
0	π	-1
1	0	1

- For QPSK, 2 bits are grouped at a time and thus $2^2 = 4$ angles are required for each symbol. Following schema is used to map symbols with phase angle :

$$
\begin{aligned}
1\,1 &\rightarrow \pi/4 \\
1\,0 &\rightarrow 7\pi/4 \\
0\,1 &\rightarrow 3\pi/4 \\
0\,0 &\rightarrow 5\pi/4
\end{aligned}
$$

  The phase angles are chosen so as to map 0/1 bits with -1/1. Here odd bits are taken as cosine components while even bits are taken as sine components. So,

x	y	Θ	x’=cosΘ	y’=sinΘ
0	0	5π/4	-1/√2	-1/√2
0	1	3π/4	-1/√2	1/√2
1	0	7π/4	1/√2	-1/√2
1	1	π/4	1/√2	1/√2

- Thus the PSK signals formed can be represented as -

  $x = cos\Theta$ for odd bits

  $y = sin\Theta$ for even bits
- Finally, the PSK signals are modulated and passed through the transmission channel by adding random white noise. Thus the final modulated wave formed are -
-

$$s(t) = Acos(2\pi ft + (\Theta + \Delta))$$

where

Δ = channel noise

s(t) = modulated signal, f = carrier frequency

- As the transmission channel introduces a noise factor, error probability rate and the efficiency of the system is also measured for demonstration.
- For demodulation, the received signal is multiplied with both sine and cosine waves of carrier frequency and further filtered with a low pass filter of appropriate frequency. Here we want to obtain the DC component of the intermediate signal and thus the cut-off freq. Will be somewhat lower than the carrier freq.

$$g(t) = s(t) * cos(2\pi ft)$$
$$g(t) = Acos(\Theta + \Delta) * cos(2\pi ft) * cos(2\pi ft) - Asin(\Theta + \Delta) * sin(2\pi ft) * cos(2\pi ft)$$

Finally, the extracted demodulated wave is converted to the original information signal using a decision block. Here, if the signal value is -ve then 0 is predicted while if it is +ve then 1 is predicted.

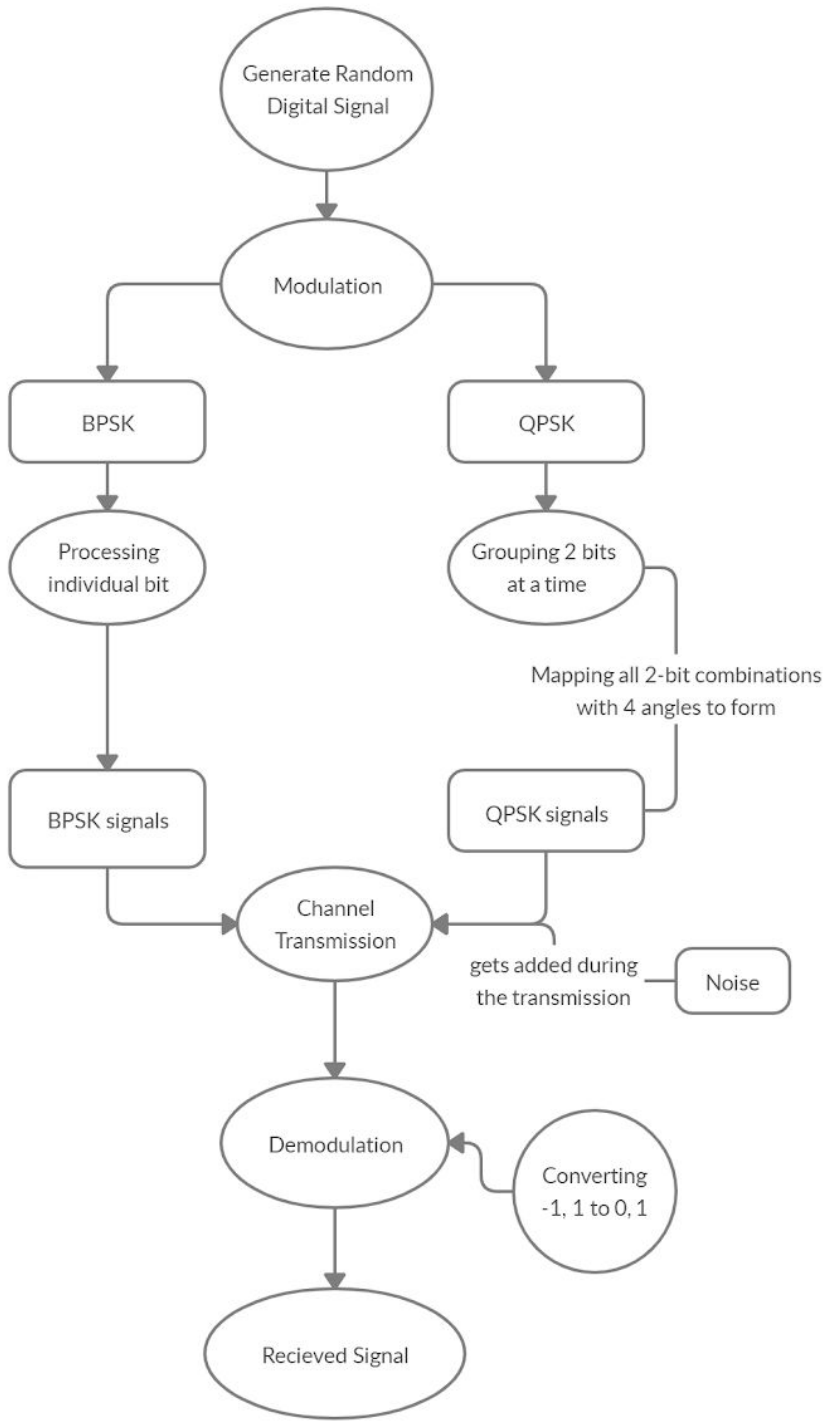
Now, the received signal might not be exactly the same as that of transmitted signal due to channel and transmission noise. Thus, the BER curve is plotted for both BPSK and QPSK signalling using their respective expressions.

$$BER = Q\left(\sqrt{2k \star \frac{Eb}{No}} sin\left(\frac{\pi}{M}\right)\right)$$

Specifications

- Here , signals can be processed both by BPSK and QPSK.
- PSK allows information to be carried with a radio communications signal more efficiently compared with FSK(Frequency shift keying).
- It is less vulnerable to faults when we evaluate with ASK(Amplitude shift keying) modulation.
- For a given transmission bandwidth, higher data rate can be achieved in case of PSK.
- An important parameter is the Bit Error Rate (BER), i.e. the ratio of the number of erroneously received bits to all transmitted bits. It is designed to determine the quality of transmission. BER, of course, depends on the ratio of the received signal power to the noise power.
- QPSK signals are more bandwidth efficient as compared to BPSK signals.
- The system simulates both BPSK and QPSK techniques with added channel noise for real-time communication systems.
- Amplitude and phase deviation parameters are included to control the channel noise margin for data transmission.
- Signal Parameters - T_b = 0.5ms, f_c = 1000 Hz, A = 10, N_0=1e-3
  P_b = 7.691e-13 P_e = 1.538e-12

Flowchart


Generate Random Digital Signal
Modulation
BPSK
QPSK
Processing individual bit
Grouping 2 bits at a time
Mapping all 2-bit combinations with 4 angles to form
BPSK signals
QPSK signals
Channel Transmission
gets added during the transmission
Noise
Demodulation
Converting -1, 1 to 0, 1
Recieved Signal

Codes

All the simulations are done in Matlab.

- **Main Program (To simulate)**

```
clear;

number_of_bits = 128; % Number of bits to generate
carrier_freq = 1000; % Carrier wave frequency
amp_err=0.05;
freq_err=0.025;

original_signal = generate_random_digital_signal(number_of_bits);

[demodulated_signal_bpsk,demodulated_signal_qpsk] = process_signal
(carrier_freq, number_of_bits, original_signal, amp_err, freq_err);

figure(1)
subplot(3,1,1);
stem(original_signal,'markerfacecolor',[0 0 1]);
title('Original Signal');
subplot(3,1,2);
stem(demodulated_signal_bpsk,'markerfacecolor',[0 0 1]);
title('Demodulated BPSK');
subplot(3,1,3);
stem(demodulated_signal_qpsk,'markerfacecolor',[0 0 1]);
title('Demodulated QPSK');

resolution = 20;
max_amp_err=0.5;    %max standard deviation
max_feq_err=0.5;      %max standard deviation

plot_BER( number_of_bits, carrier_freq, resolution, max_amp_err,
max_feq_err);
```

In the above code, we decide the parameters of the modulating bit signal and carrier signal. We also decide the resolution of the BER graph and these values are passed onto other functions

- **Processing the Digital Signal**

```
function [demodulated_signal_bpsk,demodulated_signal_qpsk] =
process_signal (carrier_freq, number_of_bits, original_signal,
amp_err, freq_err)

[qpsk_modulated_wave_e,qpsk_modulated_wave, time_axis_qpsk,
no_samp_qpsk, x_qpsk, y_qpsk] = qpsk_modulation(carrier_freq,
number_of_bits, original_signal, amp_err, freq_err);

[bpsk_modulated_wave_e,bpsk_modulated_wave, time_axis_bpsk,
no_samp_bpsk, x_bpsk, y_bpsk] = bpsk_modulation(carrier_freq,
number_of_bits, original_signal, amp_err, freq_err);

figure(2)
subplot(2,2,1);
plot (time_axis_qpsk,qpsk_modulated_wave_e,'b')
xlabel ('time[s]');
ylabel ('signal');
title('QPSK Modulation with noise');
subplot(2,2,2);
plot (time_axis_qpsk,qpsk_modulated_wave,'b')
xlabel ('time[s]');
ylabel ('signal');
title('QPSK Modulation');
subplot(2,2,3);
plot (time_axis_bpsk,bpsk_modulated_wave_e,'b')
xlabel ('time[s]');
ylabel ('signal');
title('BPSK Modulation with noise');
subplot(2,2,4);
plot (time_axis_bpsk,bpsk_modulated_wave,'b')
xlabel ('time[s]');
ylabel ('signal');
```

```
title('BPSK Modulation');

[bpsk_constellation, qpsk_constellation] = generate_constellation
(number_of_bits, original_signal);
figure(3)
subplot(1,2,1);
scatter(x_qpsk,y_qpsk,10,qpsk_constellation,'filled');
axis([-1 1 -1 1]);
title('Constellation QPSK');
subplot(1,2,2);
scatter(x_bpsk,y_bpsk,10,bpsk_constellation,'filled');
axis([-1 1 -1 1]);
title('Constellation BPSK');

[x_qpsk,y_qpsk] =
filter_signal(modulated_signal,carrier_freq,time,sampling_frequency);
[x_bpsk,y_bpsk] =
filter_signal(modulated_signal,carrier_freq,time,sampling_frequency);

demodulated_signal_bpsk = bpsk_demodulation (x_bpsk, number_of_bits);
demodulated_signal_qpsk = qpsk_demodulation (x_qpsk, y_qpsk,
number_of_bits);

bpsk_BER_val = calculate_BER (number_of_bits, original_signal,
demodulated_signal_bpsk);
qpsk_BER_val = calculate_BER (number_of_bits, original_signal,
demodulated_signal_qpsk);

bpsk_BER_val
qpsk_BER_val
end
```

The modulating signal is then modulated using BPSK and QPSK modulation using functions and both the constellations are generated. These signals and constellations are then plotted along with the BER graphs of each

- **Generate Digital Signal**

```
function original_signal =
generate_random_digital_signal(number_of_bits)

    original_signal = rand(number_of_bits,1);

    for i = 1:number_of_bits
        if(original_signal(i) < 0.5)
            original_signal(i) = 0;
        else
            original_signal(i) = 1;
        end
    end
end
```

The modulating signal is a bit signal which is generated random everytime the code is run using the code above.

- **BPSK Modulation**

```
function [ carrier_wave_e,carrier_wave, time_axis, no_samp_in_symb,
x, y ] = bpsk_modulation( carrier_freq, number_of_bits,
original_signal, amp_err, freq_err )

 % Parameters
    sampling_frequency = 20000; % sampling frequency [Hz]
    amplitude = 0.5;          % amplitude
    signal_length = (1/carrier_freq)*number_of_bits;
    symbol_length = (1/carrier_freq);

    num_samples = signal_length*sampling_frequency;
    no_samp_in_symb = symbol_length*sampling_frequency;
    time_axis =
0:1/(sampling_frequency):(num_samples-1)/(sampling_frequency);

    carrier_wave_e = zeros(1, no_samp_in_symb*number_of_bits);
    carrier_wave = zeros(1, no_samp_in_symb*number_of_bits);
    x= zeros(1, number_of_bits);
    y= zeros(1, number_of_bits);


    for i = 1:number_of_bits
        if original_signal(i) == 0
            offset = 0.5;
        else
            offset = 0;
        end
        amplitude_dev = amp_err*randn();
        freq_dev = freq_err*randn();
        x(i)=(amplitude_dev+amplitude)*cos(2*pi*(offset + freq_dev));
        y(i)=(amplitude_dev+amplitude)*sin(2*pi*(offset + freq_dev));
        for j = 1:no_samp_in_symb
            carrier_wave_e(((i-1)*no_samp_in_symb)+j) =
(amplitude_dev+amplitude)*cos(2*pi*carrier_freq*time_axis(((i-1)*no_s
amp_in_symb)+j) + 2*pi*(offset+freq_dev));
            carrier_wave(((i-1)*no_samp_in_symb)+j) =
amplitude*cos(2*pi*carrier_freq*time_axis(((i-1)*no_samp_in_symb)+j)
```

```
+ 2*pi*offset);
        end
    end
end
```

The preceding code is used to modulate the signal into BPSK. The parameters such as sampling frequency and amplitude are taken here.

- **QPSK Modulation**

```
function [ carrier_wave_e,carrier_wave, time_axis, no_samp_in_symb,
x, y ] = qpsk_modulation( carrier_freq, number_of_bits,
original_signal, amp_err, freq_err )

    sampling_frequency = 20000; % sampling frequency [Hz]
    amplitude = 0.5;          % amplitude
    signal_length = (1/carrier_freq)*(number_of_bits/2);
    symbol_length = (1/carrier_freq);

    num_samples = signal_length*sampling_frequency;
    no_samp_in_symb = symbol_length*sampling_frequency;
    time_axis =
0:1/(sampling_frequency):(num_samples-1)/(sampling_frequency);

    carrier_wave_e = zeros(1, no_samp_in_symb*(number_of_bits/2));
    carrier_wave = zeros(1, no_samp_in_symb*(number_of_bits/2));
    x= zeros(1, floor(number_of_bits/2));
    y= zeros(1, floor(number_of_bits/2));

    for i = 1:(number_of_bits/2)
        if original_signal(2*i) == 0
                if original_signal((2*i)-1) == 0
                    offset = 0.625;
                end
                if original_signal((2*i)-1) == 1
                    offset = 0.875;
                end
        end
        if original_signal(2*i) == 1
                if original_signal((2*i)-1) == 0
                    offset = 0.375;
                end
                if original_signal((2*i)-1) == 1
                    offset = 0.125;
                end
        end
        amplitude_dev = amp_err*randn();
```

```
        freq_dev = freq_err*randn();
        x(i)=(amplitude_dev+amplitude)*cos(2*pi*(offset + freq_dev));
        y(i)=(amplitude_dev+amplitude)*sin(2*pi*(offset + freq_dev));

        for j = 1:no_samp_in_symb
            carrier_wave_e(((i-1)*no_samp_in_symb)+j) =
(amplitude_dev+amplitude)*cos(2*pi*carrier_freq*time_axis(((i-1)*no_s
amp_in_symb)+j) + 2*pi*(offset + freq_dev));
            carrier_wave(((i-1)*no_samp_in_symb)+j) =
amplitude*cos(2*pi*carrier_freq*time_axis(((i-1)*no_samp_in_symb)+j)
+ 2*pi*offset);
        end
    end
end
```

The preceding code is used to modulate the signal into QPSK. The parameters such as sampling frequency and amplitude are taken here.

- **Constellation Plotting**

```
function [bpsk_constellation, qpsk_constellation] =
generate_constellation (number_of_bits, original_signal)

    bpsk_constellation = zeros(number_of_bits, 3);
    qpsk_constellation = zeros(floor(number_of_bits/2), 1);
    for i=1:number_of_bits
        if original_signal(i) == 0
            bpsk_constellation(i,1)=0;
            bpsk_constellation(i,2)=0;
            bpsk_constellation(i,3)=1;
        end
        if original_signal(i) == 1
            bpsk_constellation(i,1)=1;
            bpsk_constellation(i,2)=0;
            bpsk_constellation(i,3)=0;
        end
    end

    for i = 1:(number_of_bits/2)
        if original_signal(2*i) == 0
                if original_signal((2*i)-1) == 0
                    qpsk_constellation(i) = 1;
                end
                if original_signal((2*i)-1) == 1
                    qpsk_constellation(i) = 2;
                end
        end
        if original_signal(2*i) == 1
                if original_signal((2*i)-1) == 0
                    qpsk_constellation(i) = 3;
                end
                if original_signal((2*i)-1) == 1
                    qpsk_constellation(i) = 4;
                end
        end
    end
end
```

- **Filtering the modulated signal**

```
function [filtered_signal_1,filtered_signal_2] =
filter_signal(modulated_signal,carrier_freq,time,sampling_frequency)
    component_1 = modulated_signal * cos(2*pi*carrier_freq*time);
    component_2 = modulated_signal * sin(2*pi*carrier_freq*time);

    filtered_signal_1 =
lowpass(component_1,carrier_freq-10,sampling_frequency);
    filtered_signal_2 =
lowpass(component_2,carrier_freq-10,sampling_frequency);

end
```

The preceding code is to filter the modulated signal before demodulating it as the carrier wave is present in the modulation waveform.

- **BPSK Demodulation**

```
function [ demodulatedBitArrayBPSK ] = bpsk_demodulation (x,
number_of_bits)

    demodulatedBitArrayBPSK = zeros(number_of_bits, 1);
    for i = 1:number_of_bits
        if x(i) > 0
            demodulatedBitArrayBPSK(i) = 1;
        else
            demodulatedBitArrayBPSK(i) = 0;
        end
    end

end
```

The modulated signal which is passed through the channel (which adds noise to it) is now demodulated to get the original signal back. The BPSK demodulation is done using the preceding code.

- **QPSK Demodulation**

```
function [ demodulatedBitArrayQPSK ] = qpsk_demodulation (x, y,
number_of_bits)

    demodulatedBitArrayQPSK = zeros(number_of_bits, 1);
    for i = 1:number_of_bits/2
        if x(i) > 0
            if y(i) > 0
                demodulatedBitArrayQPSK((2*i) - 1) = 1;
                demodulatedBitArrayQPSK(2*i) = 1;
            else
                demodulatedBitArrayQPSK((2*i) - 1) = 1;
                demodulatedBitArrayQPSK(2*i) = 0;
            end
        else
            if y(i) > 0
                demodulatedBitArrayQPSK((2*i) - 1) = 0;
                demodulatedBitArrayQPSK(2*i) = 1;
            else
                demodulatedBitArrayQPSK((2*i) - 1) = 0;
                demodulatedBitArrayQPSK(2*i) = 0;
            end
        end
    end
end
```

The modulated signal which is passed through the channel (which adds noise to it) is now demodulated to get the original signal back. The QPSK demodulation is done using the preceding code.

- **Calculate Bit Error Probability**

```
function [BER] = calculate_BER (number_of_bits, original_signal,
demodulated_signal)

    iterator = 0;
    for i = 1:number_of_bits
        if original_signal(i) ~= demodulated_signal(i)
            iterator = iterator + 1;
        end
    end
    BER = iterator / number_of_bits;

end
```

Bit error probability is calculated using the preceding code.

- **BER Plots for different signals**

```
function plot_BER ( number_of_bits, carrier_freq, resolution,
max_amp_err, max_freq_err)

bpsk_BER = zeros(resolution, resolution);
qpsk_BER = zeros(resolution, resolution);

%% Generate BER Values

for i=1:resolution
    for j=1:resolution
        amp_err = i/resolution*(max_amp_err);
        freq_err = j/resolution*(max_freq_err);

        original_signal =
generate_random_digital_signal(number_of_bits);
        [qpsk_carrier_wave_e, qpsk_carrier_wave,time_axis_qpsk,
no_samp_in_symb_qpsk, x_qpsk, y_qpsk] = qpsk_modulation(
carrier_freq, number_of_bits, original_signal, amp_err, freq_err);
        [bpsk_carrier_wave_e, bpsk_carrier_wave,time_axis_bpsk,
no_samp_in_symb_bpsk, x_bpsk, y_bpsk] = bpsk_modulation(carrier_freq,
number_of_bits, original_signal, amp_err, freq_err);

        [ demodulated_signal_bpsk ] = bpsk_demodulation (x_bpsk,
number_of_bits);
        [ demodulated_signal_qpsk ] = qpsk_demodulation (x_qpsk,
y_qpsk, number_of_bits);

        bpsk_ber_val = calculate_BER (number_of_bits,
original_signal, demodulated_signal_bpsk);
        qpsk_ber_val = calculate_BER (number_of_bits,
original_signal, demodulated_signal_qpsk);

        bpsk_BER(i, j)=bpsk_ber_val;
        qpsk_BER(i, j)=qpsk_ber_val;
    end
end
```

```
arr_amp_err=
1/resolution*(max_amp_err):1/resolution*(max_amp_err):max_amp_err;
arr_freq_err=
1/resolution*(max_freq_err):1/resolution*(max_freq_err):max_freq_err;

figure(4);
subplot(1,2,1);
surf(arr_freq_err,arr_amp_err,bpsk_BER);
colormap hsv
colorbar
title('BPSK');
xlabel ('$\sigma_{\Omega}$','Interpreter','latex');
ylabel ('$\sigma_{U}$','Interpreter','latex');
zlabel ('BER');

subplot(1,2,2);
surf(arr_freq_err,arr_amp_err,qpsk_BER);
colormap hsv
colorbar
title('QPSK');
xlabel ('$\sigma_{\Omega}$','Interpreter','latex');
ylabel ('$\sigma_{U}$','Interpreter','latex');
zlabel ('BER');

end
```

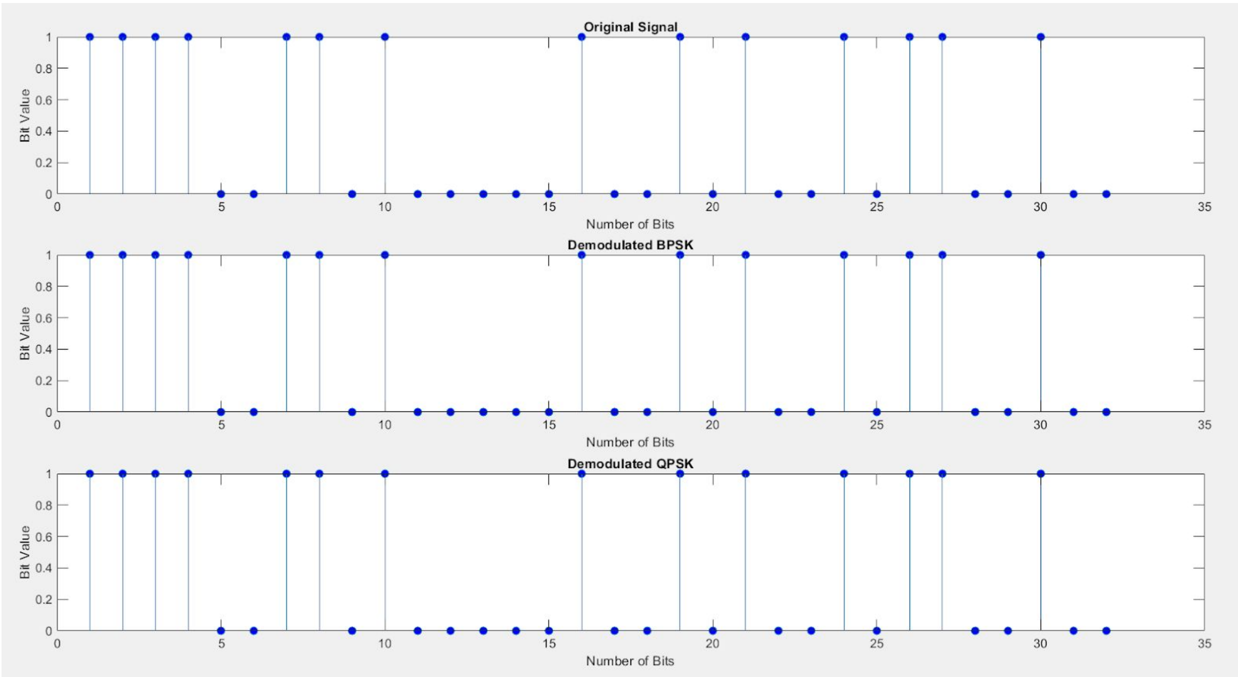
- **Generating BER Plots for BPSK,QPSK with respect to the power change**

```
clc;
eb = 0:1:29;
itr = 30;
k = 2;
berr = zeros(itr);
for j=1:1:itr
    EbNodB = eb(j);
    mu = 0;
    sigma = 1;
    pd = makedist('Normal','mu',mu,'sigma',sigma);
    EbNo = 10.^(EbNodB / 10.0);
    x = sqrt(2 * k * EbNo) * sin(pi / (2.^k));
    bit_error_probability = 1 - cdf(pd,x);
    symbol_correct_probability = (1 - bit_error_probability).^k;
    symbol_error_probability = 1 - symbol_correct_probability;
    berr(j) = symbol_error_probability/k;
    ber_db= mag2db(berr);
end
plot(eb,ber_db,'m');
axis([0 12 -150 1]); %y-axis from 10^-150 to 10
xlabel ('$\frac{Eb}{N0}$','Interpreter','latex');
ylabel ('BER');
legend('BER');
title('BER QPSK = BPSK');
```

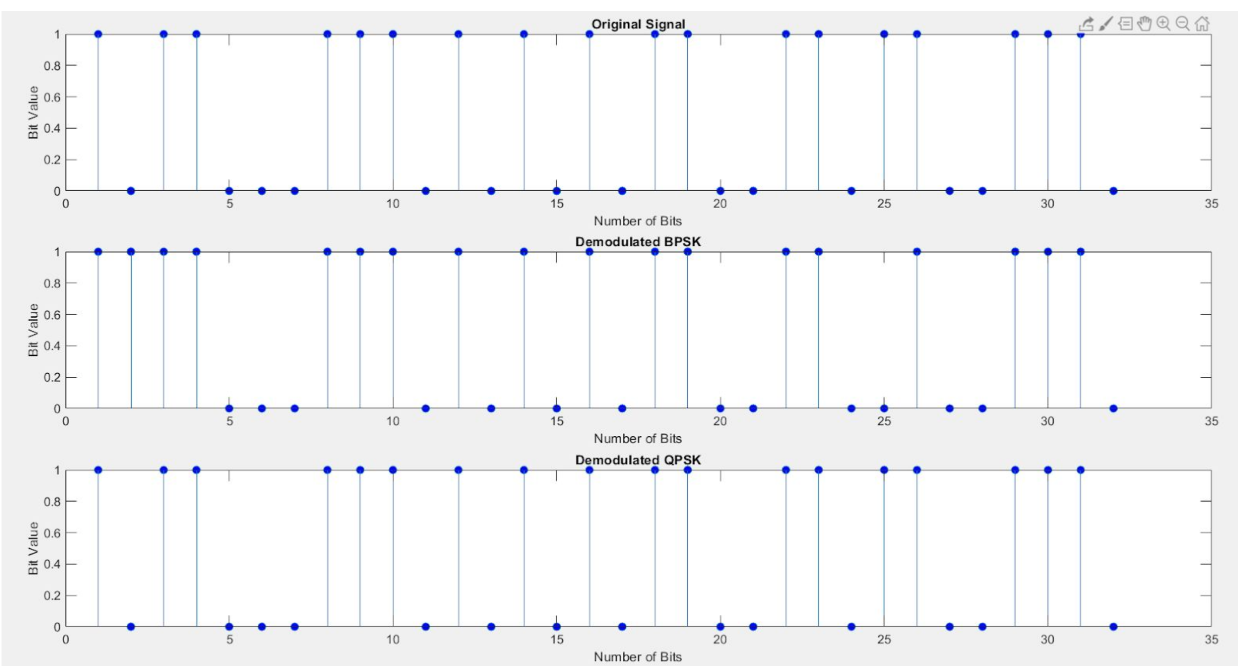
This code is used to plot the BER plots of BPSK and QPSK signal with respect to power change.

Results

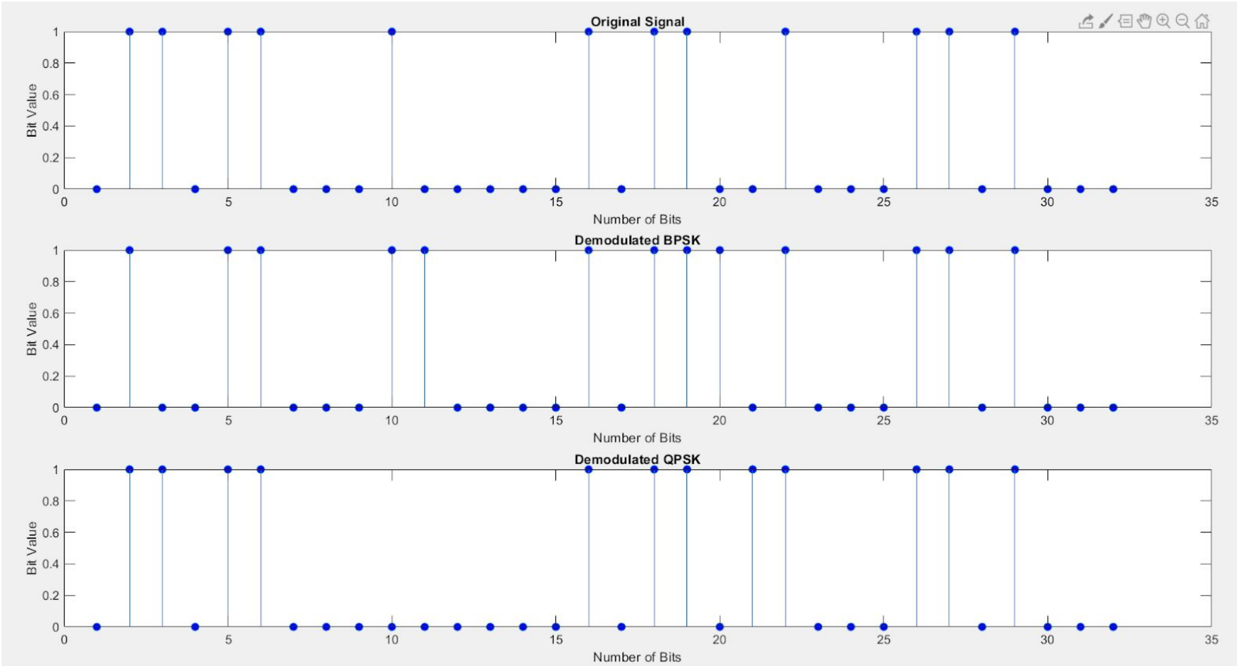
→ Transmitted Signal vs Received Signal. (Number of bits = 32)



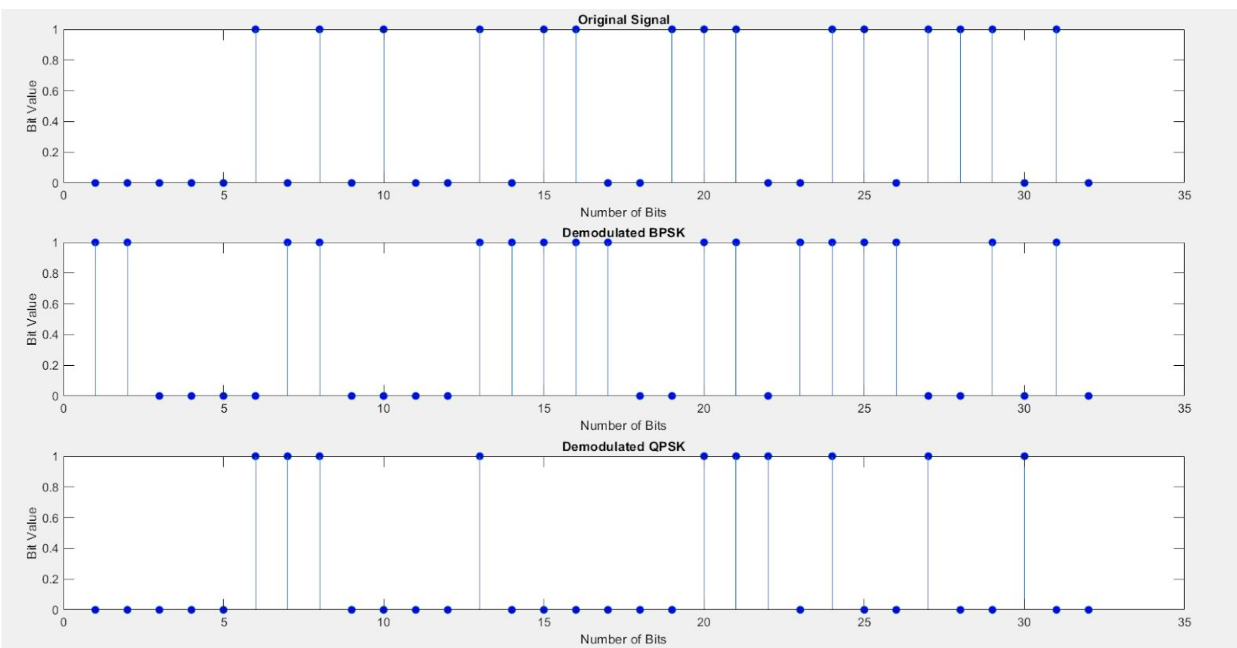
Signal Retrieval Percentage: BPSK : 100% , QPSK : 100% (For Very Less Noise)



Signal Retrieval Percentage : BPSK : 93.75% , QPSK : 100% (Decent Noise Added)



Signal Retrieval Percentage : BPSK :90.62% , QPSK : 93.75% (Some More Noise Added)

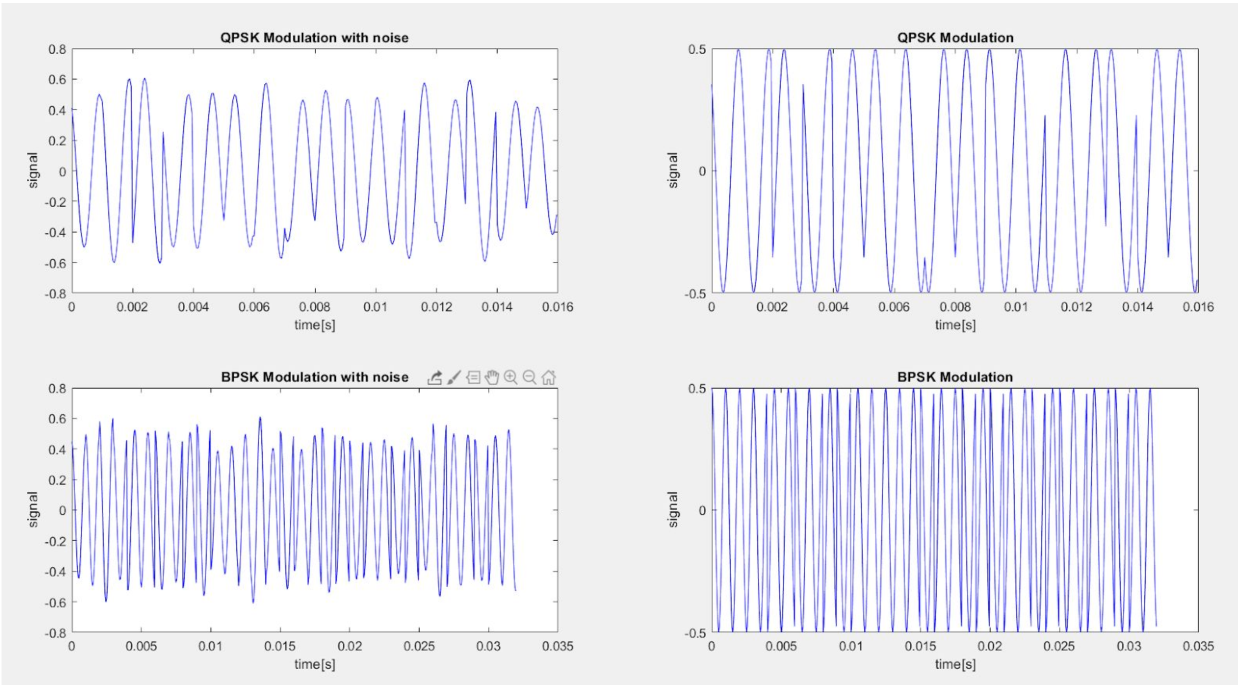


Signal Retrieval Percentage : BPSK :62.5% , QPSK : 65.63% (Huge Noise Added)

We get the plots for the original signal, demodulated BPSK and demodulated QPSK as shown.

→ Modulation

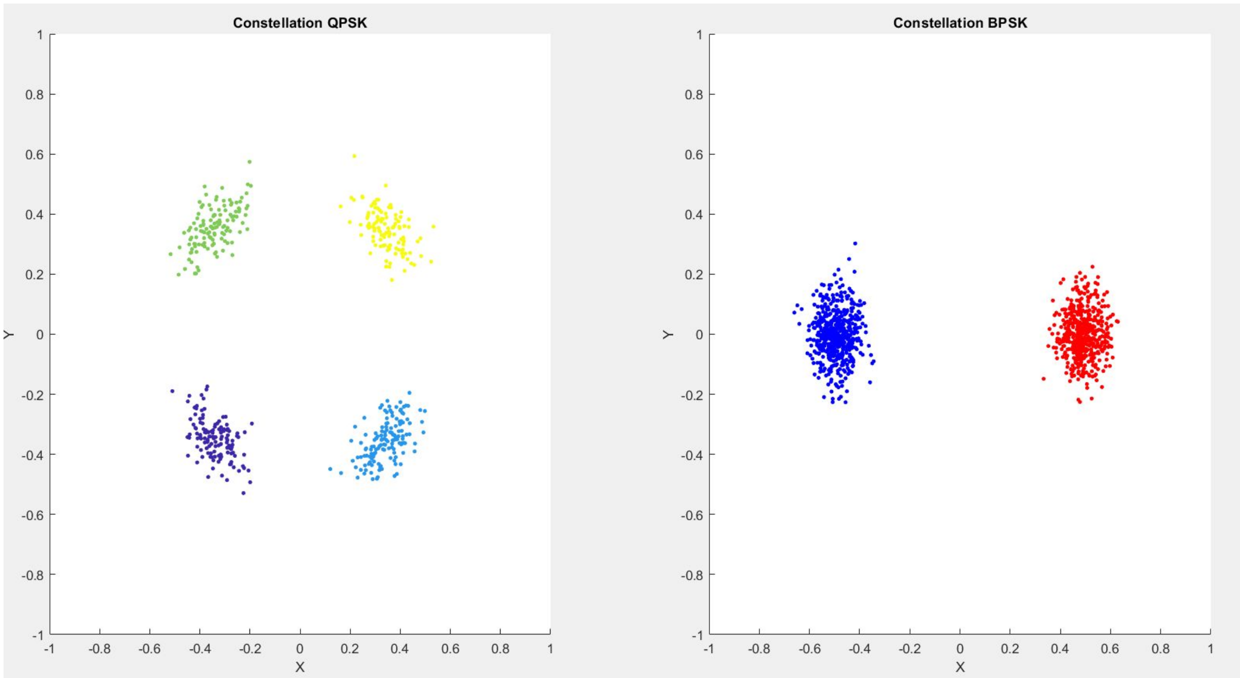
Here, number of bits = 32



The plots for QPSK modulation(with and without noise) and BPSK modulation(with and without noise) are shown.
It can be seen that noise does affect modulation.

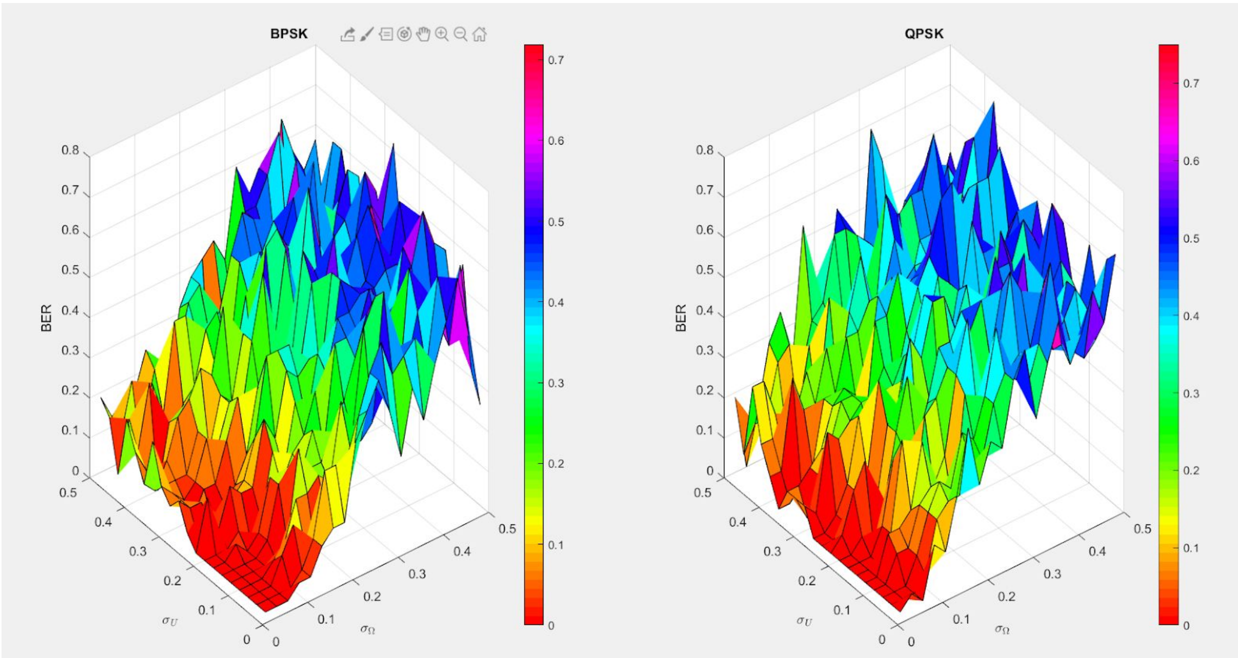
→ Constellation Diagram :

Here, number of bits = 1024



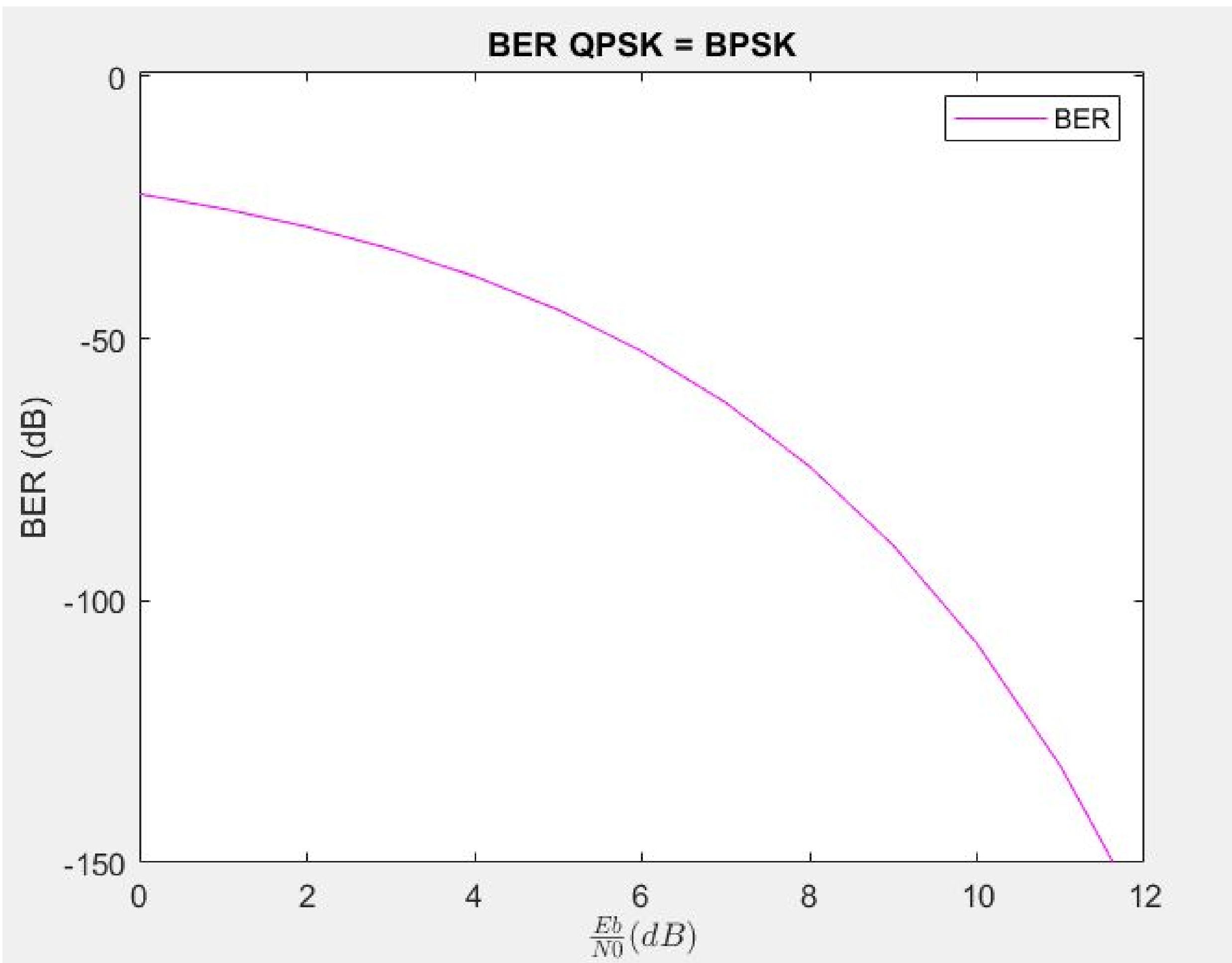
The constellation diagram of QPSK and BPSK is as above. The scattered points show the effect of noise in the transmission channel.

→ 3D BER Plots w.r.t to Noise :



Above is the BER plot of different random digital information signals for different values of amplitude and phase noise constants. We can infer that the BER increases with an increase in phase noise constant which is quite expected. As the noise increases in the channel, it becomes more difficult to predict the transmitted signal values and thus both bit error probability and symbol error probability likely increases.

→ BER Plot w.r.t. energy per bit to noise power spectral density ratio (Eb/N0) :



The BER curve(for BPSK and QPSK) for different values of $\frac{Eb}{No}$ is plotted.