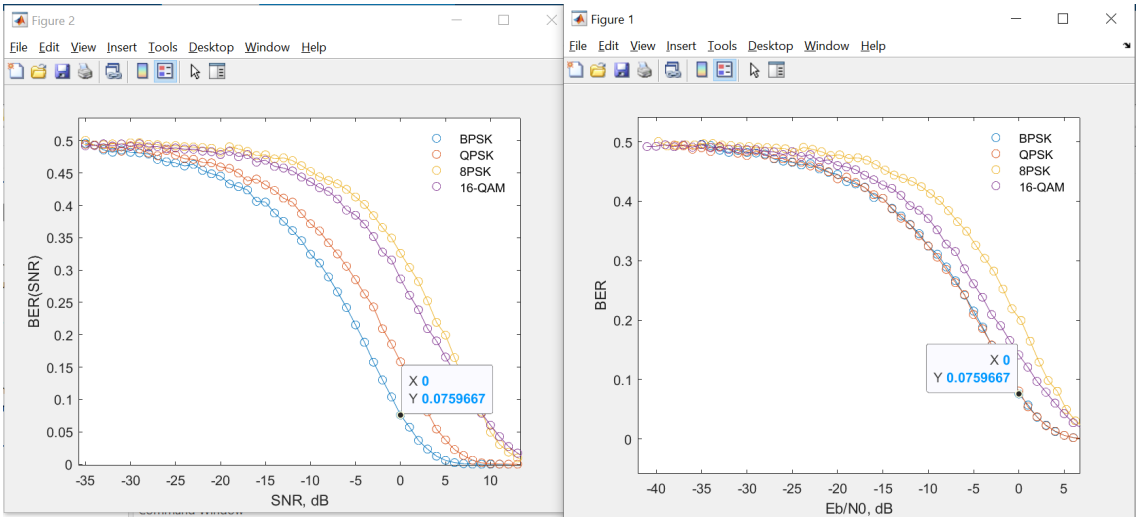
1) Это не критично, но всё же: при конвертировании SNR в Eb/N0, можно было бы формулу оставить одну и ту же, а в зависимости от выбранного созвездия выбирать другие параметры.

```
function [Eb_N0] = Eb_N0_convert(SNR, Constellation)
    switch Constellation
        case "BPSK"
            bitperpoint = 1;
        case "QPSK"
            bitperpoint = 2;
        case "8PSK"
            bitperpoint = 3;
        case "16-QAM"
            bitperpoint = 4;
    end
    Eb_N0 = SNR + 10*log10(1/bitperpoint);
end
```

2) По поводу BER(SNR) и BER(Eb/N0). Это ОЧЕНЬ странно: скриншот 1. Допустим, у нас есть BPSK созвездие: 1 точка IQ передаёт 1 бит информации.
Исправила. Ошибка была в том, что один и тот же массив Eb_N0 откладывала на оси Ox, хотя он меняется от созвездию к созвездию. Теперь в экспериментальные графики для BPSK и QPSK совпадают, **как и должно быть в теории????** :)



3) Mapping. 8PSK созвездие неправильное. Сейчас у тебя созвездие как показано на скриншоте 2. А нужно, как на скриншоте 3.
В чём главная проблема твоего созвездия? Какому требованию оно не отвечает?
Остальные созвездия в порядке, включая нормировку

Исправила порядок словаря в функции constellation_func (закомменчело то, что было до)

```
se  8PSK
  %Dictionary = [+1 000, +(1/sqrt(2))*(1+1i)001, +1i 010, +(1/sqrt(2))*(-1+1i) 011, ...
  %              +-1 100, +(1/sqrt(2))*(-1-1i) 101, +-1i 110, +(1/sqrt(2))*(1-1i) 111];

  Dictionary = [(1/sqrt(2))*(-1-1i), -1, 1i, (1/sqrt(2))*(-1+1i), ...
                -1i, (1/sqrt(2))*(1-1i), (1/sqrt(2))*(1+1i), 1];
```

Главная проблема была в том, что соседние символы отличались на >= 1 бит информации. В идеальном случае, соседние символы должны отличаться на 1 бит информации, чтобы в случае ошибки был потерян только 1 бит, это нужно для помехоустойчивого кодирования.

4) Что за переменная eps находится в demapping()? За что она отвечает?

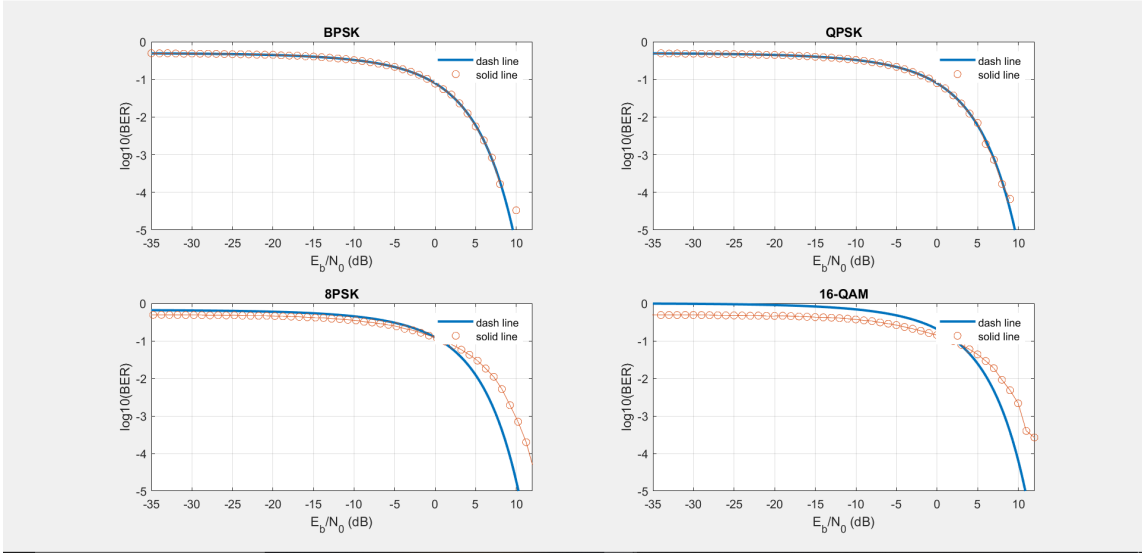
```
eps = 1e-10;

Bit = zeros(1, length(IQ_RX));

for itter1 = 1 : length(IQ_RX)
    t = zeros(1, Num_Constellation);

    %for itter2 = 1 : Num_Constellation
        %t(1, itter2) = IQ_RX(itter1) - Dictionary(itter2);
    %end

    t = IQ_RX(itter1) - Dictionary;


    [~, idx_min] = min(abs(t));
    error_min = t(idx_min);
    c = IQ_RX(itter1) - error_min;

    Bit(itter1) = idx_min-1;
end
```

Забыла удалить переменную, раньше вместо нахождения нужного вектора Bit по индексу просто добавляла вектор ошибки. Из-за того что исходные значения могли не совпадать из-за точности добавила некий аналог машинного эпсилон eps. Я не помню точно почему от этой идеи пришлось отказаться, но вроде после вычитания вектора программа потом его не находила.

5) Попробуй отказаться от второго цикла в пользу поэлементного вычитание 2 векторов (скриншот 4). Это позволит в некоторых случая сократить время исполнения программы и повысит читаемость.

Исправила, видно на предыдущем скрине, в MER тоже

7) Перестроила графики в логарифмическом масштабе по оси Oy + с масштабам по обеим осям. Добавила графики для сравнения теор. и эксперимент. кривых



8) Вынесла в функцию plots_const

Main_BER_SNR.m | constellation_func.m | mapping.m | Eb_N0_convert.m | plots_const.m

```
function [] = plots_const(IQ_TX, Constellation)
figure;
plot(IQ_TX, 'o');
[Dictionary, Bit_depth_Dict] = constellation_func(Constellation);

for itter = 1 : length(IQ_TX)
    switch Constellation
        case "BPSK"
            switch IQ_TX(itter)
                case Dictionary(1)
                    text(-1.5, 0.5, '0')
                case Dictionary(2)
                    text(1.5, 0.5, '1')
            end

        case "QPSK"
            switch IQ_TX(itter)
                case Dictionary(1)
                    text(real(Dictionary(1)), imag(Dictionary(1)),  '00')
                case Dictionary(2)
                    text(real(Dictionary(2)), imag(Dictionary(2)),  '01')
                case Dictionary(3)
                    text(real(Dictionary(3)), imag(Dictionary(3)),  '10')
                case Dictionary(4)
                    text(real(Dictionary(4)), imag(Dictionary(4)),  '11')
```