CS1031 – Lab 5
BER curves and constellations

Exercise 1: Building BER-SNR curves

Description

In this exercise you will build a number of BER-SNR curves as those we have examined during the course.

In order to build the curve you have to simulate a transmission: first generate a number of bits, then modulate them using an appropriate digital modulation (4-QAM; 16-QAM; 64-QAM; 256-QAM), then add Gaussian noise and finally receive the bits and compare them to the original ones to evaluate how many bit were received incorrectly.

This needs to be repeated for different levels of noise (thus simulating different SNR values): each of these simulation runs will generate one point for the curve.

In order to build one curve you will calculate a number of points, for SNR values starting from 0 dB and at increments of 2 dB.

The range of BER required is between a BER of 10^-1 and 10^-5.

One you have all point for one modulation you can build the plot using the *semilogy(..)* function.

You need to build 4 curves, one for each of the following modulations: 4-QAM; 16-QAM; 64-QAM; 256-QAM.

Implementation

In order to calculate each point of the plot you will need to carry out the following operations:

1. generate a random stream of data bits, whose *size* should be at least 10 times higher than the inverse of the smallest BER you want to calculate. For example if you are trying to calculate a point for a BER of 10^{-3} you need to use a *size* of 10 x 10^3= 10^4 random bits. You can use the following function:

   stream=randi([0 QAM-1], size, 1), where QAM is the number of levels of the modulation you are considering
2. Modulate the data with the appropriate modulation:

   mod = qammod(stream, QAM);

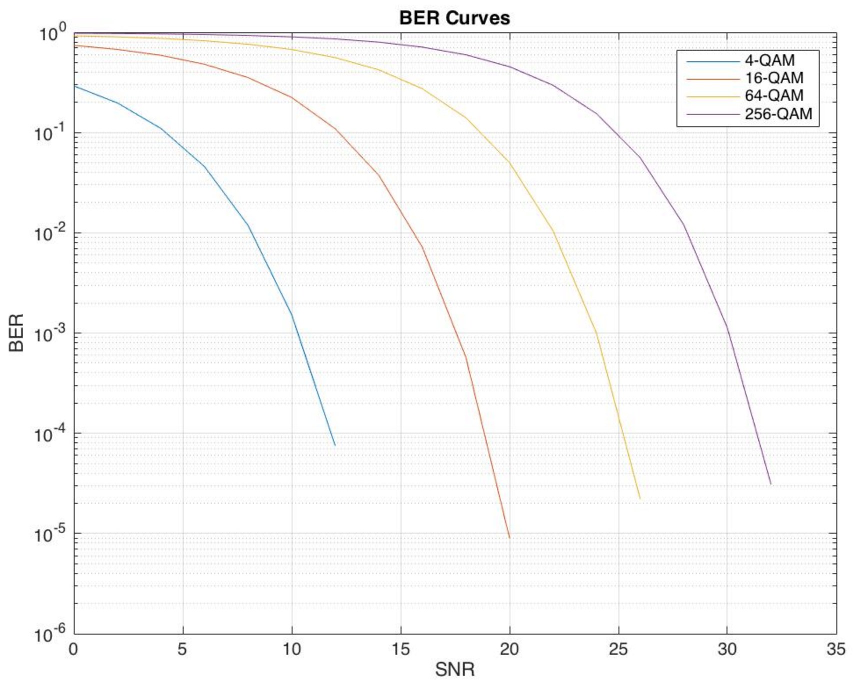
3. Add noise with a given SNR. Each plot should be generated for a series of SNR values, from 0 dB at incremental steps of 2 dB. You can use the function:

   signal_noise=awgn(mod, SNR, 'measured') to add noise to the modulated stream at a given SNR level;
4. Demodulate the data, using:

   Dem=qamdemod(signal_noise, QAM);
5. Compare the demodulated data with the original sequence to calculate the number of bit errors (it is up to you to find the best way of doing this)

In order to plot each QAM curve you need to calculate the bit error rate for each different SNR point. You should use for loops to recursively calculate the BER for all the different SNR levels and for the different QAM modulations.

The final plot should look like this:



Exercise 2: Constellation plots

In this exercise you will examine a constellation plot during the signal transmission, reusing some of the code you wrote for exercise 1.
The constellation indicates how noise affects each modulation symbol.

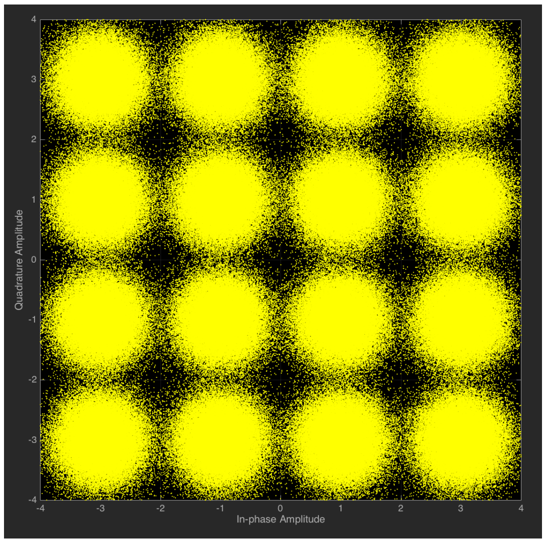
Draw a constellation graph, for the 16 QAM modulation, for two different SNR levels: SNR=16 and SNR=20;

Looking back at the code for exercise 1, you need to apply the following function to the modulated signal after adding the noise (but before it is demodulated):

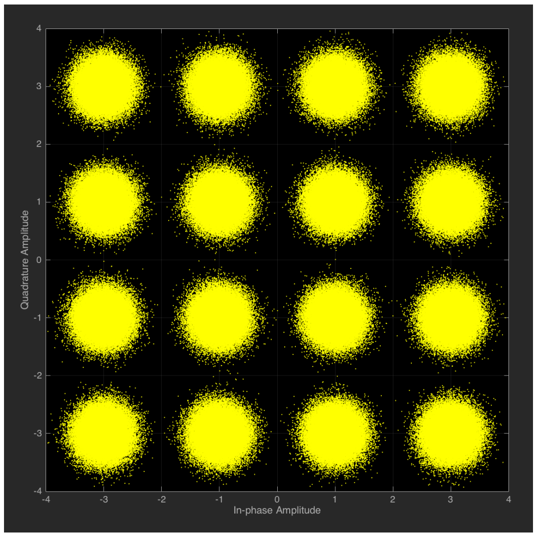
```
        const=comm.ConstellationDiagram('ShowReferenceConstellation',
false, 'XLimits', [-4 4], 'YLimits', [-4 4]);
        step(const, signal_noise);
```

You can see that in the plot with SNR=16 there is higher chance that the symbols might be confused with their adjacent ones, thus generating higher BER.

Plot for SNR=16



Plot for SNR=20


Quadrature Amplitude
In-phase Amplitude