6-th practical exercise
Noise Immunity of QAM-16 Modulation

<u>The purpose</u> of the work is to design a program for QAM-16 modulated signal detection in AWGN noise. The number of errors must be estimated for different SNR values.

The report should be prepared and sent to email address: elans.grabs@rtu.lv. The report must include the following:

1. The objective of practical exercise;
2. The full source code of simulation program;
3. Received signal constellation plots for different SNR values;
4. BER values for different SNR values.

<u>The tasks to be solved</u>:

1. Complete the missing parts of program code in file "PD6_QAM16.m", which is available in ORTUS. Please, note, that You also need to download and complete missing lines in file "demodData.m" as well, and place it in the same directory.
2. Draw received signal complex plane for each SNR value: 5 dB, 6 dB, 7 dB, 8 dB, 9 dB, 10 dB.
3. Determine the number of errors for each SNR value and estimate error probability.
4. Make a conclusion on noise immunity of QAM-16 modulation based on obtained results.

<u>The guidelines for practical exercise</u>

<u>1. Generating symbols from binary data.</u>
Initially, data must be generated randomly in binary form, as a vector-row consisting of "0" and "1".

```
binDt = [0   0   1   0    1   0   1   1    0   1   …]
```

In order to change data into symbols, first they mut be reformatted, so that each column consists of 4 binary symbols. Use reshape( ) function for this. The resulting matrix will look like:

```
symDt =
     0    1    0    0    1    0    1    1    0    1
     0    0    1    0    1    1    1    0    1    1
     1    1    1    0    1    0    0    0    0    0
     0    1    0    0    0    0    1    1    0    1
```

Next step – binary to decimal conversion. This is performed by matrix multiplication:

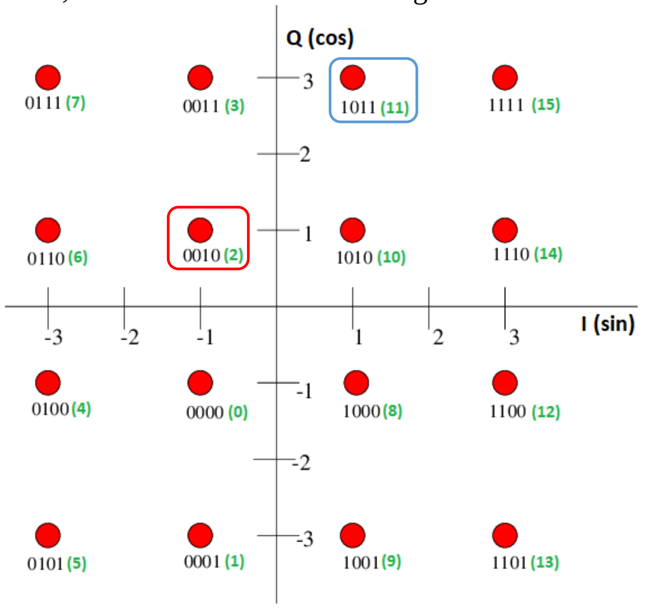
$$\begin{matrix} 8 & 4 & 2 & 1 \end{matrix} \times \begin{matrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{matrix}$$

The resulting vector-row contains symbols encoded in decimal form:

```
symDt = [2   11   6   0   14   4   13   9   4   13   …]
```

2. QAM-16 signal complex envelope generation and adding white noise

Previously calculated QAM-16 symbols will be used as the numbers of signal points in QAM-16 constellation diagram (see picture below). The complex envelope real/imaginary part values are corresponding point coordinates, which must be read from diagram.



For this purpose, the Matlab program already contains the constellation diagram in a form of matrix below:

```
mapQAM16 = [-1-1i; -1-3i; -1+1i; -1+3i;
            -3-1i; -3-3i; -3+1i; -3+3i;
            +1-1i; +1-3i; +1+1i; +1+3i;
            +3-1i; +3-3i; +3+1i; +3+3i];
```

Please, note, that Matlab indexing starts with 1 rather than 0, therefore each symbol must be increased by 1 to match range of [1, 16].

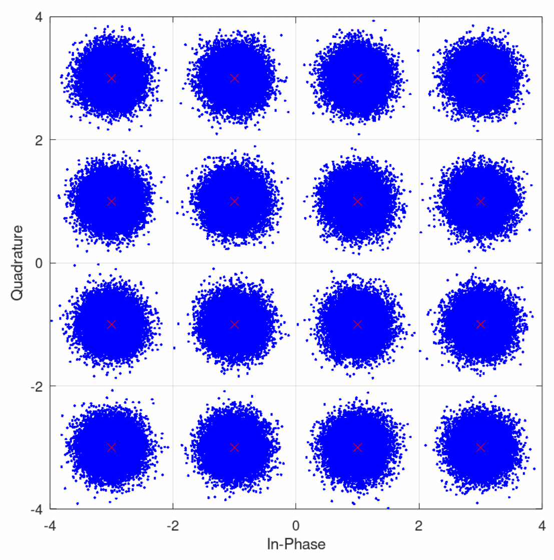
```
symMod = mapQAM16(symDt+1);
```

For this work, the noise must be added by using awgn( ) function, which is included in course for Octave program.

Use ***plot( )*** function to draw received noisy signal and ideal constellation:

- received signal points “symNoi” with blue dots;
- ideal constellation points of “mapQAM16” with red crosses.

The result will look like picture below:



Note: In order to equalize scale of both axes, You can use the following command:
```
axis square;
```

3. QAM-16 signal detecting and error counting

Signal reception and demodulation is performed by “demodData.m” sub-program.

The basic principle of a receiver – calculation of difference between received symbol and each ideal symbol. Then, the closest by distance (Euclidean metric) signal will be selected as the most probable. In order to calculate difference for row vector of 250 000 elements and ideal symbols vector of 16 elements, both vectors must be replicated by using ***repmat( )*** function, as follows:

```
-3.1444 - 3.2862i  -3.1444 - 3.2862i  -3.1444 - 3.2862i   ...
-0.9331 + 2.4174i  -0.9331 + 2.4174i  -0.9331 + 2.4174i   ...
-0.9093 + 1.1833i  -0.9093 + 1.1833i  -0.9093 + 1.1833i   ...
0.7204 + 2.9421i   0.7204 + 2.9421i   0.7204 + 2.9421i    ...
-2.9032 - 3.1174i  -2.9032 - 3.1174i  -2.9032 - 3.1174i   ...
3.0670 - 1.1971i   3.0670 - 1.1971i   3.0670 - 1.1971i    ...
1.2120 - 2.7822i   1.2120 - 2.7822i   1.2120 - 2.7822i    ...
0.9959 - 2.8156i   0.9959 - 2.8156i   0.9959 - 2.8156i    ...
-0.8504 + 3.2101i  -0.8504 + 3.2101i  -0.8504 + 3.2101i   ...
2.6396 - 3.3156i   2.6396 - 3.3156i   2.6396 - 3.3156i    ...
```

```
-1 - 1i   -1 - 3i   -1 + 1i   -1 + 3i   -3 - 1i   -3 - 3i   ...
-1 - 1i   -1 - 3i   -1 + 1i   -1 + 3i   -3 - 1i   -3 - 3i   ...
-1 - 1i   -1 - 3i   -1 + 1i   -1 + 3i   -3 - 1i   -3 - 3i   ...
```

As a result, we obtain two matrixes:

- symNoi [250 000 x 1] → symMtx [250 000 x 16];
- mapQAM16.' [1 x 16] → mapMtx [250 000 x 16].

Note: For complex matrix, transpose must be performed by elementwise .' (instead of simple ')

After these matrix dimensions changes, now these two matrices can be easily subtracted from each other and distance can be calculated for each received symbol.
The absolute value of difference for these complex numbers is Euclidean metric for each of possible 16 ideal constellation symbols.

Furthermore, the symbol with minimum distance matches the column number in this matrix. This minimum index can be found by using min( ) function with two output arguments, and also You must specify, that minimum is searched in the 2-nd matrix dimension (column):
```
[dstMin, symDmd] = min(dstMtx,[],2);
```

The last step is decimal to binary conversion by using ***dec2bin( )*** function:

```
binDmd = dec2bin(symDmd-1,4);           %Conversion to binary (in text)
binDmd = (binDmd == '1')';              %Compare with ‘1’ text character
```

Please note, that here we again consider Matlab indexing specifics and subtract 1, which we have previously added.

The result is saved in vector-row, achieved by forcing column format with (:) operator and further transpose by ' operator:

```
binDmd = binDmd(:)';
```