Acoustic modem project

Session 3: QAM and OFDM modulation

Paschalis Tsiaflakis, Hanne Deprez[1]

October 2017

Goal: Creating m-files for modulating and demodulating a binary signal using QAM and OFDM techniques, and evaluating their transmission performances for an additive white Gaussian noise (AWGN) channel.

Requirements: Matlab in Windows.

Required files from DSP-CIS website: none

Required files from previous sessions: none

Outcome: 7 m-files: *qam_experiment.m*, *qam_mod.m*, *qam_demod.m*, *ber.m*, *ofdm_experiment.m*, *ofdm_mod.m*, *ofdm_demod.m*

Deliverables: See Session 4

Important remarks:

Try to write your matlab functions as general as possible using general input parameters (e.g., M-ary constellation size, FFT size, ...), so that these can be reused in later exercise sessions. Also try to reuse existing Matlab functionality. Hint: Have a look at the Matlab Communications Systems Toolbox: (> help comm/comm). In particular the following functions may be useful: randi, modem.qammod, modem.qamdemod, modem.modulate, modem.demodulate, awgn.

1 Exercise 3-1: Quadrature amplitude modulation (QAM)

In this exercise, you will experiment with different types of QAM modulation to understand their trade-offs in terms of bit rate, signal-to-noise ratio (SNR) and bit error rate. Create an m-file *qam_experiment.m* in which the following steps are implemented:

1. Generate a pseudo random binary sequence of a user defined length.
2. Create a Matlab function *qam_mod.m* that modulates a sequence of bits into M-ary QAM format where M is by definition an integer power of 2, i.e., $M = 2^{N_q}$. The maximum value of N_q for the target OFDM system will be 6, corresponding to 64-QAM. Call the *qam_mod.m* function from within the *qam_experiment.m* script to modulate the bit sequence that you have generated in step 1.
3. Check the constellation diagram for the generated QAM symbol sequence. An example of a 16-QAM constellation diagram is shown in figure 1.

[1]For questions and remarks: hanne.deprez@esat.kuleuven.be

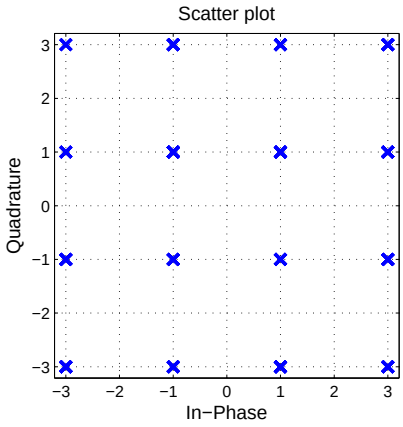

Figure 1: Scatter diagram of 16-QAM constellation

4. Compare the QAM symbol sequences for different constellation sizes. What are their corresponding bit rates? What are their average signal powers? Is normalization to unit power necessary? If yes, can you find the normalization factors for the different constellation sizes?

5. Now put additive white Gaussian noise (AWGN) on the QAM symbol sequence and again check the constellation diagram. What are the trade-offs of using different constellation sizes?

6. Create a Matlab function *qam_demod.m* that demodulates a QAM symbol sequence back to a binary sequence for different values for M. Call this function from within the *qam_experiment.m* script to demodulate the noisy QAM symbols back to bits.

7. Create a Matlab function *ber.m* that takes two binary sequences (a transmitted sequence and a received sequence) and calculates the so-called bit error rate (BER). Call this function from within the *qam_experiment.m* script and check the BER for different M-ary QAM constellations, and different signal-to-noise ratios (SNR).

2 Exercise 3-2: Orthogonal frequency-division multiplexing (OFDM)

In this exercise, you will implement OFDM modulation. Make sure to first understand the concept of OFDM modulation (or similarly DMT modulation) as explained in the lecture slides. Create an m-file *ofdm_experiment.m* in which the following steps are implemented (and try to reuse as much as possible the code you have obtained in Exercise 3-1).

1. Create a Matlab function *ofdm_mod.m* that simulates an OFDM-based transmitter. For this you first generate a packet of P OFDM frames as shown in figure 2. Each frame consists of N complex numbers, of which two numbers are equal to zero (corresponding to the DC frequency and the Nyquist frequency), and the other numbers (one number per so-called frequency bin) are equal to QAM symbols taken from a QAM symbol sequence (which is generated, e.g., in the function *qam_mod.m* that you created in Exercise 3-1), or to the complex conjugate (*) of these QAM symbols. See figure 2 for the exact structure of a frame. Then an IFFT operation is applied to each of the OFDM frames and each resulting frame is parallel-to-serial converted. Explain the necessity of the 'mirror operation' in each frame from a signal processing point of view.

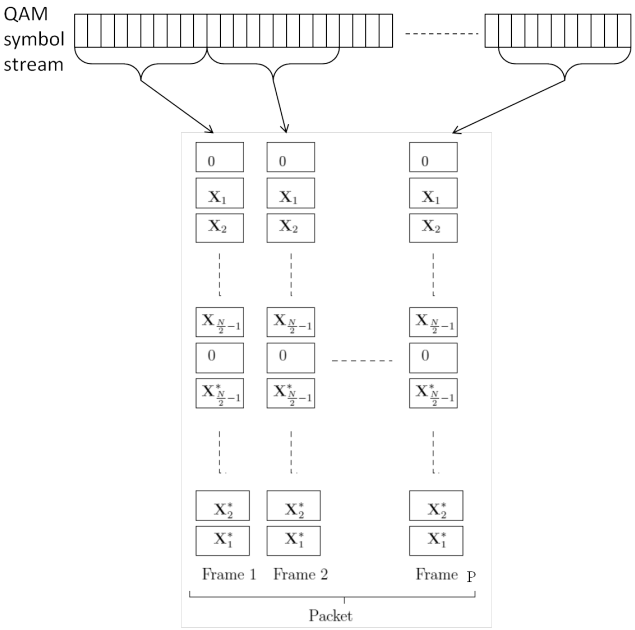

Figure 2: M-ary QAM OFDM frames and packet

2. Create a Matlab function *ofdm_demod.m* that simulates an OFDM-based receiver. In the receiver, a serial-to-parallel conversion is performed, followed by an FFT operation, to regenerate the transmitted OFDM frames and QAM symbols.

3. Test your OFDM-based transmitter and receiver design by simulating a transmission over an ideal (real) channel (i.e., $H(z) = 1$ and SNR=∞ dB). Use the *ber.m* function implemented in Exercise 3-1 to check if your design is correct.

4. Extend the transmitter and receiver with blocks that add and remove a cyclic prefix to every data frame. The cyclic prefix length is user defined, it is usually much shorter than the actual frame length. Again check the *ber.m* function to verify if your design is correct.

5. Evaluate the performance of the obtained OFDM communication chain for AWGN:

   (a) Add AWGN to the receiver input signal. The SNR is user defined.

   (b) Compare the BER obtained for different QAM constellations at a given SNR.

   (c) Run the same experiment for different SNRs.