EE558 - Project - Due date: 12/17/2020 at 11:59PM

November 20, 2020

1 Important Notes

1. Due date: Thursday 12/17/2020 at 11:59PM. No late submission is allowed. You're strongly suggested to start the project early.
2. Requirements: You're allowed to discuss with classmates how to proceed with this project. However, you're strictly required to write the MATLAB code on your own.
3. Submission: Through Turnitin link on Blackboard. Both MATLAB code and figures should be placed on a same Word or PDF file. Plagiarism will be reported.
4. Weight: 10 percentage points. For each task, you will be given either full points or no points.
5. Weight distribution:
   - Undergraduate students: You need to do Task 1 (3 points), Task 2 (3 points), Task 3 (2 points), and Task 4 (2 points).

2 Provided Materials

1. MATLAB codes have been and will be released through homework exercises. Referring to the available codes will be useful and developing them for this project will help you save time.
2. PAM4_mod.m: to do digital 4-PAM mapping (i.e., 4-ASK mapping). You can input a long sequence of bits or just two bits at time. For each chunk of two bits, the 4-PAM modulator will perform non-Gray mapping to a symbol with the amplitude as follows:

   Above is the two-bit input, below is the output amplitude with the mapping.
3. PAM4_demod.m: to do 4-PAM demapping (i.e., 4-ASK demapping), a reverse process to the above 4-PAM mapping. Depending on the input symbol, it will provide a two-bit output. For example, if the input is a symbol at amplitude 0.3, the output will be bits 10.

3 Tasks

You will be simulating a communication system with 16-QAM modulation/demodulation. The tasks include multiple steps: 16-QAM modulation, 16-QAM demodulation, bit error rate (BER) calculation, up-converting/down-converting, and pulse shaping.

There are 4 tasks. Tasks 1 and 2 involve only baseband discrete-time transmission/reception of Gray-mapping 4-PAM and 16-QAM, respectively. Task 3 involves passband transmission/reception of 16-QAM. Finally, Task 4 involves baseband transmission/reception of 64-QAM.

3.1 Task 1

You will be first rewrite the PAM4_mod.m into PAM4_mod_GRAY.m and PAM4_demod.m into PAM4_demod_GRAY.m. The two new files will be used for GRAY mapping and demapping as follows:

You're required to simulate the performance of Gray-mapping 4-PAM signaling. The following steps should be taken:

1. Generate a random binary information sequence of length $L = 3000$ bits.

2. Map the bit sequence into the symbol sequence using your PAM4_mod_GRAY.m.

3. Note that the symbol sequence has the average energy at 5, as $1/4*(-3)^2+1/4*(-1)^2+1/4*1^2+1/4*3^2 = 5$. Suppose that this symbol sequence is scaled by a factor of a, the average energy per symbol is $E_s = 5a^2$. Since $E_s = 2E_b$ (2 bits per symbols, energy per symbol is twice energy per bit), you should set $a = \sqrt{2/5}$ so that $E_b = 1$. You should have a sequence of $L/2 = 1500$ symbols.

4. Generate a sequence of $L/2$ independent Gaussian random variable with variance $\sigma^2 = N_0/2$. MATLAB code to generate *one* Gaussian random variable with variance σ^2: \sqrt(σ^2)*randn. Make sure that N_0 has to be set to a proper value, depending on the value E_b/N_0 (dB) you're working on.

5. Add the noise sequence n to the transmitted symbol sequence to have the received signal r. If you do correctly to this point, for $E_b/N_0 = 12$ dB, r should look like in Fig. 1.
6. Demodulate the received signal sequence by your PAM4_demod_GRAY.m. Make sure before passing sending the signal sequence through the demodulator, you have to scale back $a = \sqrt{2/5}$ by multiplying $\sqrt{5/2}$.

7. Compare the demodulated sequence with the original bit sequence and find the BER.

8. Save the BER for each E_b/N_0 (0, 1, ..., 12 dB).

9. For low BER (around 10^{-4} with $E_b/N_0 = 12$), you need to generate enough errors (total bits in error of 100 for instance). So you may need to send and detect multiple sequences of length $L = 3000$ bits.

10. Plot the BER curve by the *semilogy* function. For example, semilogy(EbN0dB, BER).

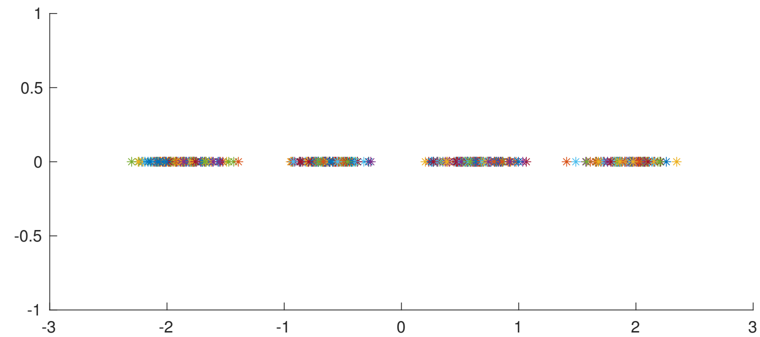


Figure 1: Received signal with 4-PAM signaling at $E_b/N_0 = 12$ dB. The received signal should be grouped around 4 possible transmitted symbols of 4-PAM.

The following figure 2 plots the results with non-Gray and Gray mappings. You should get similar results to the red curve (the lower BER curve).

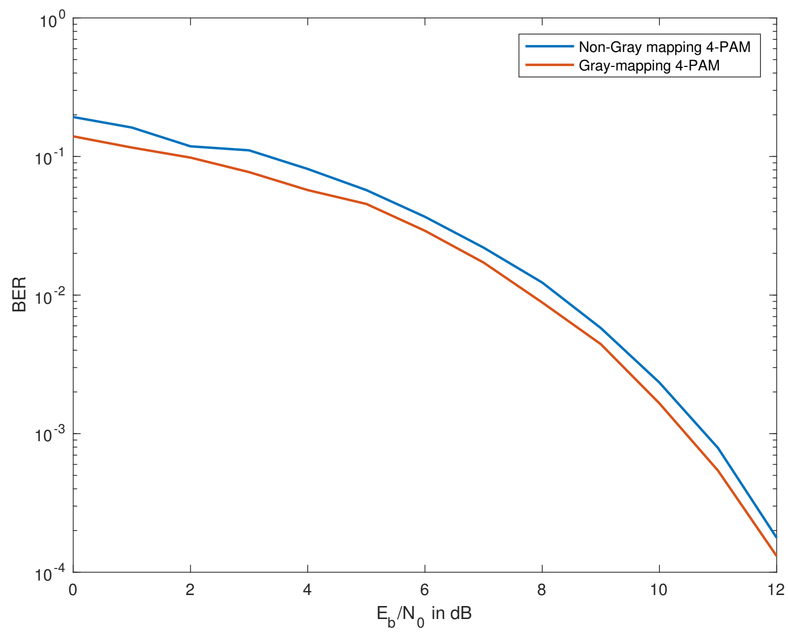


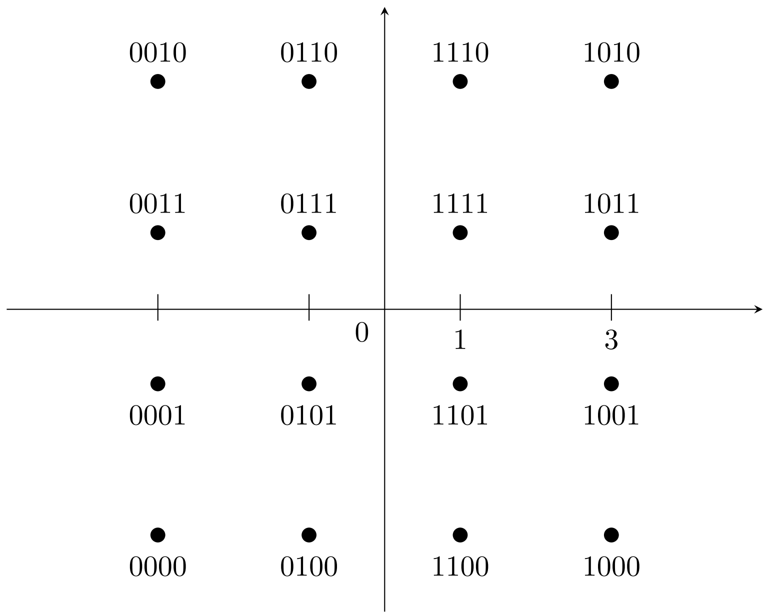
Figure 2: BER performance of 4-PAM with non-Gray and Gray mappings.

3.2 Task 2

You will be reusing the code in Task 1 for Gray-mapping 16-QAM. Following the same procedure as in Task 1, except that the code performs the modulation and demodulation of 4 bits at the time. The additional steps should be taken:

1. For every chunk of 4 bits, map the first 2 bits using the PAM4_mod_GRAY.m into the in-phase signal s_I and map the last 2 bits using the PAM4_mod_GRAY.m into the quadrature signal s_Q.

2. Transmitted signal will be formed as $s = s_I + 1i * s_Q$. Make sure you have to scale the symbol properly by a factor $a = \sqrt{2/5}$ to have $E_b = 1$. You should have a sequence of $L/4 = 250$ complex symbols. The 16-QAM Gray-mapping with real/imaginary component should be similar to this:



3. Generate complex noise vector of variance $\sigma^2 = N_0/2$ per dimension. MATLAB code to generate *one* Gaussian random variable with variance σ^2 per dimension: \sqrt(σ^2) $*$ (randn $+$ $1i$ $*$ randn).

4. Add the noise vector to the transmitted signal, let's the received signal as $r = s + n$. If you do correctly until this point, plotting $plot(real(r), imag(r),' *')$ in MATLAB should provide a similar noisy constellation as in Fig. 3.

5. When performing detection, you can do separate 4-PAM demodulations on the *real component* of r to get the first 2 bits and the *imaginary component* of r to get the last 2 bits. You should use your PAM4_demod_GRAY.m.

6. Compare the demodulated sequence with the original bit sequence and find the BER.

7. Save the BER for each E_b/N_0 (0, 1, ..., 12 dB) and plot the result.

You should obtain the same performance as 4-PAM signaling. The key thing here is you can double the data-rate by using two-dimensional signaling and 16-QAM.

3.3 Task 3

You will be reusing the code in Task 2 for passband modulation and demodulation of 16-QAM. In real life, there is nothing called imaginary. Instead, you will use the cosine function at carrier frequency to carry the in-phase signal and the sine function at carrier frequency to carry the quadrature signal. You need to refer to the code in Homework 4 and the waveformdetection.m file in Homework 5.

The following tasks should be taken:

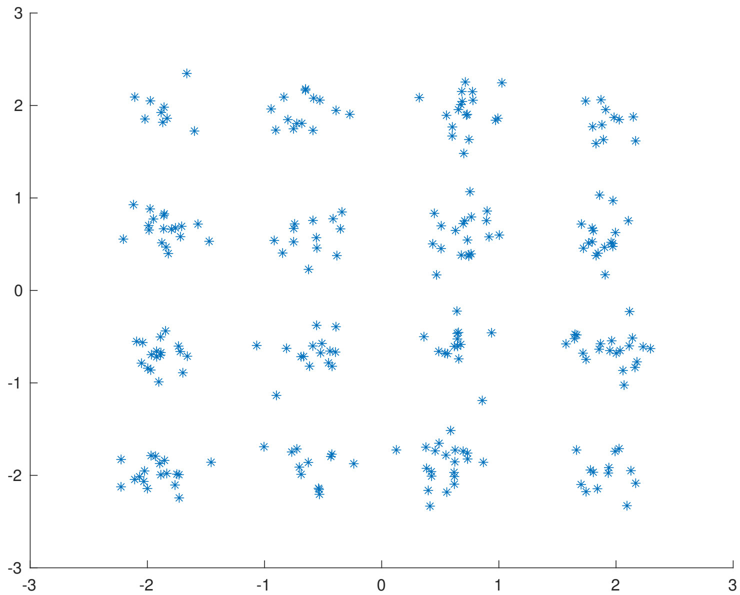


Figure 3: Received signal with 4-PAM signaling at $E_b/N_0 = 12$ dB. The received signal should be grouped around 16 possible transmitted symbols of 16-QAM.

1. Set the carrier frequency $fc = 100$.
2. Set symbol duration $Tsymb = 0.1$.
3. Set sampling time $Ts = 0.001$. In this case, you will have $sampPerSymb = 100$ samples per symbol duration.
4. Create two orthogonal basis functions: cosine and sine functions that last for one symbol duration. Make sure to normalize the two functions to make it orthonormal basis function set.
5. For the baseband signal s_I, you have to perform line coding by upsampling by the factor of $sampPerSymb$ into UPs_I. Create the in-phase passband signaling by filtering UPs_I with the cosine basis function into PBs_i.
6. For the baseband signal s_Q, you have to perform line coding by upsampling by the factor of $sampPerSymb$ into UPs_Q. Create the quadrature passband signaling by filtering UPs_Q with the sine basis function into PBs_Q.
7. The transmitted signal is formed by $s = PBs_I + PBs_Q$. If you do correctly, running $plotspec(s)$ should give similar result as in the Fig. 4.
8. Generate the real noise vector with variance $N_0/2$. Make sure correct vector length.
9. Add noise the s to get $r = s + n$.
10. Retrieve the in-phase baseband signaling by filter r with the matched filter corresponding the cosine basis function. Perform sampling at every $Tsymb$ time, or every $sampPerSymb = 100$ samples.

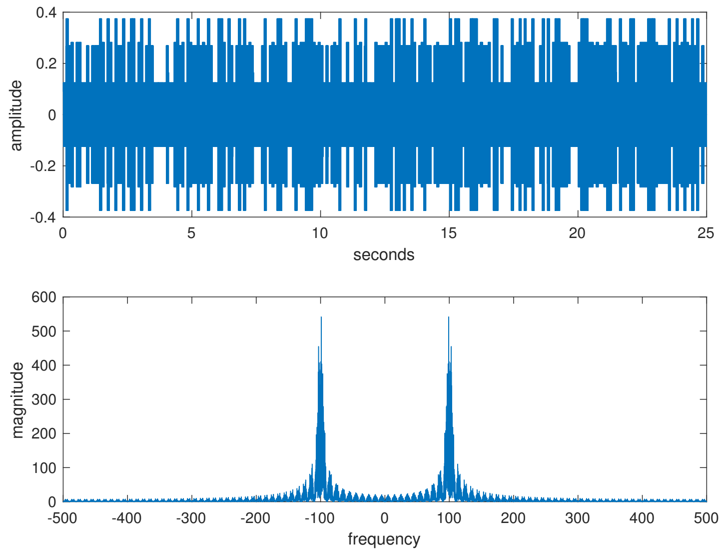


Figure 4: Power spectral density of unfiltered 16-QAM at transmission. Note that the carrier frequency is 100 Hz, symbol time is $Tsymb = 0.1$ s. Thus, you would see zero-crossing at every $1/Tsymb = 10$ Hz. You would observe there are three amplitude levels in the waveform. This can be verified by the three levels of energy of the symbols in a 16-QAM constellation.

11. Retrieve the quadrature baseband signaling by filter r with the matched filter corresponding the sine basis function. Perform sampling at every $Tsymb$ time, or every $sampPerSymb = 100$ samples.

12. Perform 4-PAM demodulation for in-phase and quadrature signals separately.

13. Obtain and plot the BER for E_b/N_0 (0, 1, ..., 12 dB). You should get the same BER performance at Task 1 and 2.

3.4 Task 4

64-QAM can be implemented by send an 8-PAM signal on the in-phase dimension and another 8-PAM signal on the quadrature dimension. You are required to implement the baseband discrete model for the modulation and demodulation of 64-QAM signaling in additive white Gaussian noise channel. Here are the hints to do this task:

1. Create a new file PAM8_mod_GRAY.m to implement Gray 8-PAM signaling. This will be similar to the PAM4_mod_GRAY.m you did in Task 1.

2. Create a new file PAM8_demod_GRAY.m to implement Gray 8-PAM signaling. This will be similar to the PAM4_demod_GRAY.m you did in Task 1. You need to make sure how to derive and assign the factor a correctly to get the average energy per bit as 1.

3. Follow the steps in Task 2 for modulation and demodulation of 64-QAM signaling by placing one 8-PAM signaling on the in-phase (real) dimension and another 8-PAM signaling on the quadrature (imaginary) dimension.

4. Plot the BER performance of 64-QAM and include to your report.