2019 ES 156 Project

Due date: On the day of the Final, 11:59pm.

Instruction

This project has 3 intertwined parts, and is worth up to 15% additional bonus points to your grade. The parts become broader/newer as you progress: Part I follows directly from class, Part II is new material but connected to class, Part III involves completely new material. You should turn in:

1. A well-written, typed write-up with the solution to each part, as well as any references used in your research. **Make sure you make use/refer to concepts seen in class**. The solutions should be written in IEEE format using the `IEEETran` conference (two-column) format. The latex template can be found here: https://ctan.org/pkg/ieeetran?lang=en, and equivalent Word templates can be found on Google. The write-up should include:
   - An abstract, concisely summarizing what you did and your contributions.
   - An introduction section composed of (i) an overview of the project, (ii) a related work section with the references you used, and an (iii) a notation section.
   - Each of the three parts of the project should be a section in your write-up, composing the main body of the document. Any graphs/equations/mathematical derivations that are relevant to your answers should also be included here (make sure you add captions to your figures).
   - A conclusion section.
   - Bibliographical references.
2. Working and well-documented code. You are expected to write your own code and, if possible, minimize the use of existing packages. You can use the programming language of your choice, but Matlab and/or Python are preferred.

This project is open-ended, and no formal solution will be posted online. There is no single solution to each problem, and different approaches may produce different outcomes. Grading will take into account not only your answers, but the overall quality of your write-up, code, insights, references, etc. The grading template below can serve as guidance for what is expected from you. Remember – these are free points that are worth 50% of all the PSets you did or, equivalently, half of your midterm, so the bar for grading will be set very high.

- **OK project (0-3 points):** You attempted 1, maybe 2 of the items below. Your answer is in IEEE format, but not particularly well written. You don't use many references, and some of the sections are missing. The time you put into the project is about the time you spent on 1 or 2 problem sets.
- **Good project (3-6 points):** You attempted all 3 of the components, making significant progress in at least one of them. Your write-up is insightful, with neat figures and significant references, and includes all the items mentioned above. Your code may be a bit sloppy, but can be used to reproduce your results. You spend about 2-3 problem sets worth of time.
- **Very good project (6-10 points):** You successfully solved all of the items below. In two of the components, your write-up includes resources and derivations from a few research papers/advanced text books. The introduction of the write-up can be used as a overview/study-guide for studying communications, coding, or one of the other topics studied in the project. Your code is efficient and well-documented. You spent 3-4 problem sets worth of time.

- **Great project (10-13 points):** Same as the previous item, but for all 3 components. You spent about 4-5 problem sets worth of time.
- **Amazing project (13-15 points):** You produced near research-grade new results in your project, and your write-up is very close to a submission to an IEEE conference like ICASSP, ISIT, Globecom, ICC, etc, or to textbooks in the field. You not only solved the problems below, but extended them significantly by exploring new, state-of-the-art techniques. For an amazing project, your write-up should be at the of a conference submission in its precision, notation, conciseness, clarity, and literature overview.

Manage your time! The goal of the project is to allow you to explore more advanced topics, but that should not come at the cost of preparing for your finals, **so be reasonable and work wisely**. Remember: a good grade on the final is more beneficial than the project.

Take the initiative to seek out more resources/information. Several of the problems below are open-ended and have a research flavor to it, with the expectation that you will be resourceful. **The teaching staff is here to help,** but we count on you to seek out on-line material, other books, or research papers. We are excited to help you through this process, but the initiative has to come from you! So please reach out if you need help.

1 Part I – Digital communications and coding

The first and second part of the project consists of implementing and testing an abstraction of a digital communication system based on what we saw in class. We provide a brief overview of digital communications next – we count on you to seek out other resources or ask the teaching staff if you have questions or are confused.

Roughly speaking, a single-user digital communication channel can be modeled as the following:

1. **Input**: a sequence of d input bits $\mathbf{m} = (m_1, m_2, \ldots, m_d) \in \{0,1\}^d$. This constitutes the d-bit input message to the channel. We assume throughout that the messages are uniformly distributed, i.e. each message can occur with probability $1/2^d$.
2. **Add redundancy**: the message is **encoded** using a channel code. A channel code is a mapping $c : \{0,1\}^d \to \{0,1\}^k$, $k \geq d$, denoted by $\mathbf{m}_c \triangleq c(\mathbf{m})$. This mapping adds *redundancy* to the transmitted message. The simplest channel code is one where every bit is repeated a few times, known as a repetition code. More complicated codes include linear codes, Reed-Solomon codes, LDPC codes, Turbo codes, and others based on algebraic structures. You can learn about several different channel codes via a quick Google search.
3. **Map bits to symbols**: Your binary input sequence $\mathbf{m}_c$ is then mapped to a sequence of n symbols[1] $\mathbf{x}(\mathbf{m}) = (x_1(\mathbf{m}), \ldots, x_n(\mathbf{m})) \in \mathbb{R}^n$. Note that that $\mathbf{x}$ is a produced from $\mathbf{m}_c$, but we denote it as a function of $\mathbf{m}$ for simplicity. The value $\mathbf{x}(\mathbf{m})$ is usually called the *channel input*, and the collection of all channel inputs compose a *codebook*. When no coding is used, this is poetically called a *constellation*.
4. **Average Power**: The codebook has an average power constraint, given by:

$$\frac{1}{2^d} \sum_{\mathbf{m} \in \{0,1\}^d} \|\mathbf{x}(\mathbf{m})\|_2^2 \leq P, \tag{1}$$

[1] Here we consider the symbols to be Real for simplicity. In many resources they might be denoted using complex numbers

where $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$ is the usual Euclidean norm.

5. **Symbols-to-pulses**: We will consider amplitude modulation. The exact modulation scheme won't matter of this component of the project, but we encourage you to add a discussion of the step to your write-up. At a high level, you can consider a sequence of bandlimited, unit-energy waveforms $\phi_k(t)$, $k = 1, \ldots, n$. Each symbol of $\mathbf{x}$ will modulate the amplitude of the corresponding waveform, with the transmitted signal being:

$$x(t) = \sum_{k=1}^{n} x_k \phi_k(t).$$

6. **Received signal:** The received signal (not taking delay into account) can be modeled as

$$y(t) = H(t)x(t) + N(t).$$

Here, $N(t)$ and $H(t)$ stochastic processes that represent additive noise process and *fading*, respectively.

7. **Demodulation**: You can assume that the signal $y(t)$ is processed, and we recover a vector

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{N},$$

where $\mathbf{H} \in \mathbb{R}^{n \times n}$ and $\mathbf{N} \in \mathbb{R}^n$.

In this first part, you will be designing decoders for recovering the message $\mathbf{m}$ from the received signal $\mathbf{y}$.

1.1 Task 1: Testing BPSK

As a first task, consider that no channel coding is used, and we transmit one bit at a time (i.e., $d = 1$, $n = 1$). Each bit of the message is directly mapped to a channel input:

$$\mathbf{x}(0) = -\sqrt{P}, \ \mathbf{x}(1) = \sqrt{P}.$$

In this task you may assume $\mathbf{H} = 1$ and that $\mathbf{N}$ is a Gaussian random variable with 0 mean and variance σ^2, denoted by $\mathbf{N} \sim \mathcal{N}(0, \sigma^2)$.

Describe how you would design a receiver (i.e., a mapping from $\mathbf{y}$ to the message $\mathbf{m}$) that decodes the received signal $\mathbf{y}$ to a transmitted bit. Create a plot where the y-axis is average bit error probability (BER) and the x axis is the energy-per-bit over noise ratio E_b/σ^2, where the energy per bit E_b is

$$E_b \triangleq \frac{P}{d} \tag{2}$$

(in this first task $d = 1$). You can generate the plot by either deriving a closed form expression for the error probability, or via simulation using your language of choice. If you simulate, make sure to report the behavior for different values of P and σ^2. The BER should be in log scale, and E_b/σ^2 should be in dB. You will repeat a plot in the same style in the tasks below.

1.2 Task 2: Testing 4-QAM

Now assume that we send a vector of 2 symbols at a time ($n = 2$), and 2 bits ($d = 2$) without using channel coding. Here, the values that $\mathbf{x}(\mathbf{m})$ can assume are $(\pm\sqrt{P/2}, \pm\sqrt{P/2})$. Once again you may assume $\mathbf{H} = 1$, and each entry of $\mathbf{N} = (N_1, N_2)$ satisfies an i.i.d. Gaussian distribution $N_i \sim (0, \sigma^2)$.

Describe a receiver to recover the received signal (i.e., a mapping from $\mathbf{y}$ to the message $\mathbf{m}$) to recover the message. How would you assign $\mathbf{m}$ to codewords $(\pm\sqrt{P/2}, \pm\sqrt{P/2})$ in order to minimize error probability?

Plot the E_b/σ^2 vs. average bit error probability (BER) for this case (where E_b is given in (2)). You can again either compute an expression or do it via simulation. How does it compare to the results you got in Task 1?

1.3 Task 3: More complex QAM

Now repeat the previous task still considering that you transmit two symbols at a time ($n = 2$), but your constellation becomes more complex by transmitting more bits (e.g., $d = 4, 6, 8$). Suggest a few different maps $\mathbf{x}(\mathbf{m})$ while being careful to maintain the average power constraint. For your choices, create the E_b/σ^2 vs. average bit error probability plot. How does performance scale as d becomes larger?

1.4 Task 4: The impact of fading

Now repeat Task 3 considering that $\mathbf{H}$ is a 2-by-2 diagonal matrix, with diagonal entries having h_1, h_2 each with distributions $\mathcal{N}(0, 1)$ (i.e., Gaussian distribution, unit mean and zero variance). What happens to the error probability now? This kind of "multiplicative" noise is called fading, and is one of the main challenges in wireless communications.

Next, assume that $\mathbf{H}$ is still drawn from the above distribution, yet is *known* when you are decoding $\mathbf{y}$. In practice, $\mathbf{H}$ is estimated using a pilot transmission, for example. How would you use this knowledge to decrease the BER? Plot the corresponding curves for your method.

2 Part II – The Impact of Coding in Reliable Communications

In the previous task we did not consider channel coding. Here, we will study how coding can potentially help reduce transmission errors. In this task you can assume that that $\mathbf{N} = (N_1, \ldots, N_n)$, where $N_i \sim \mathcal{N}(0, \sigma^2)$ are i.i.d. and n is the number of channel uses.

2.1 Task 5: Repetition Code

In Part I you computed/simulated BPSK and QAM. Now consider that you use each of the schemes repeated times to transmit the same message. This is known as a *repetition code*, where $\mathbf{c}(\mathbf{m}) = m_1 m_2 \ldots m_d m_1 \ldots m_d m_1 \ldots m_d$. For example, using a 3-repetition code and BPSK, when transmitting $\mathbf{m} = 0$, we have $\mathbf{c}(0) = 000$ and the channel input would be $\mathbf{x}(0) = (\sqrt{P}, \sqrt{P}, \sqrt{P})/\sqrt{3}$ (recall that a repetition code must still satisfy the power constraint). Assuming $\mathbf{H} = 1$, compute the performance of BPSK and QAM for different repetition codes by plotting the corresponding BER vs Eb/σ^2 plots. Does repetition help?

2.2 Task 6: Other Codes

There are many different channel codes that perform significantly better that repetition (e.g., Hamming Codes, Reed-Solomon Codes, LDPC Codes). You can find may implementations of them online, or you can implement your own. Investigate the performance of these codes when used in conjunction with BPSK and QAM. Do they improve performance? Report what you find. How does fading affect this coding scheme?

If you use a given pre-existing code, explain a little bit about how the code works and why you chose it in your write-up.

2.3 Task 7: Taking it to the limit

In his remarkable 1948 paper "A Mathematical Theory of Communications," Claude Shannon showed that for the for the above model we can achieve asymptotically vanishing error probability (i.e. BER $\to 0$) as long as

$$R = \frac{d}{n} < \frac{1}{2} \log \left(1 + \frac{P}{\sigma^2} \right). \tag{3}$$

However, his construction requires $n \to \infty$ and $d \to \infty$. How close can you approach this limit?

3 Part III – Disaster Strikes the Galileo Probe

Attention: this part is very open-ended. Reach out to the teaching staff for additional pointers if you want to undertake this part in earnest.

The Galileo was a probe launched by NASA to investigate Jupiter and it's moons. Galileo was put it space by the Space Shuttle Atlantis, and was launched towards Jupiter via gravitational assisted flybys of Venus and Earth.

During its first flyby around Earth after launch, Galileo received a (planned) remote command to open its high-gain antenna. The antenna was shaped like an umbrella, and was supposed to transmit crisp images of Jupiter back to Earth. To the dismay of many NASA engineers, the antenna failed to fully open: when the driver was remotely activated to open up the antenna's "ribs", some of the ribs got stuck, leaving the antenna looking like a lop-sided, half-open umbrella.

The high gain antenna was rendered useless, and the spacecraft was left with only a low-rate, omnidirectional antenna that was supposed to be used for control purposes. The expected transmission rate that was ≈ 130 kbps had now fallen to $8 - 16$ bps. NASA could increase the signal-to-noise ratio of the received transmission slightly by using its Deep Space Network — an array of antennas distributed across Earth. This increased the transmission rate to 160 bps — a significant gain, but still far beyond mission requirements.

In order to maintain the mission, NASA engineers had to implement a custom-made image compression algorithm remotely on the probe (based on the DCT), together with channel coding to overcome errors due to low SNR. The combination of image compression (reducing image size without significant loss of quality) and channel coding (increasing reliable transmission rate) was key for Galileo's success. Before this even, scientists at NASA were often cautious in using image compression algorithms, worrying that the loss of resolution could lead to an important discovery to be missed due to decreased resolution.

What would you have done? Explain and give evidence for your solution.