Digital Modulation Simulation Project

Professor: Brian Frost

Fall 2020

Introduction

As you know from class, many digital communication systems can be described by a two-dimensional constellation of symbols with basis functions proportional to $p(t)\cos\omega_c t$ and $-p(t)\sin\omega_c t$, defined from time 0 to T, for some pulse $p(t)$. We usually represent the symbol as $A + Bj$ where A^2 is the energy in the in-phase component and B^2 is the energy in the quadrature component. You should also recall from class that we can analyze the impact of noise entirely through this I/Q representation of the symbol. This simplifies the framing of the least squares decision metric from an L_2 problem (requiring integration) to finding the minimum distance between two points in the plane.

Using this representation, we can simulate *millions* of symbol transmissions in MATLAB quite quickly – this is necessary if we want to explore systems with error rates.

You will explore QAM, PSK and DPSK systems using this framework, with the goal, largely, of producing SNR per bit vs.probability of symbol error plots, and comparing system performance. Many such plots are available in your book for reference. A large emphasis will be placed on the writing of efficient MATLAB code.

MATLAB Problems

1. My constellation consists of M symbols, $c_1, \ldots, c_M \in \mathbb{C}$. I transmit N symbols, $a_1, \ldots, a_N$, from this constellation. They are each corrupted by AWGN, so that at the receiver $a_i + n_i$ is seen. The least squares decision $\hat{a}_i$ is the constellation point nearest $a_i + n_i$ in the usual distance sense over $\mathbb{C}$. Write a function which takes a constellation (a vector of M complex values) and a noisy symbol vector (a vector containing N complex values) as inputs, and outputs the length N estimated symbol vector (all values of which are valid constellation points) based on the least squares decision method. The code for this function should not be long, should contain no loops and should be fast. To ensure it works, run a few tests with, for example, $N = 5$ randomly generated signals and a constellation such as $\{1, -1, j, -j\}$ (don't present this in your report, just do it to make sure your whole project is built on solid ground).

2. Write a function which takes as an input a *base* constellation, a list of N noise-free transmitted symbols (these symbols will be drawn from the base constellation symbols), a value for the noise power N_0 and a desired value for the SNR per bit in dB. The function first should find the desired average symbol energy (from the SNR per bit, the noise power and the number of symbols in the constellation). It will then scale the base constellation to form the true constellation, which has the correct average energy. The transmitted symbols will be scaled by the same amount, as they must come from the true constellation. Produce N realizations

from a noise process with the correct noise power (note that the noise for each symbol is drawn from a complex Gaussian $Z = X + jY$ where X and Y are also zero-mean Gaussian with half of the variance of Z – think about the theorem for linear combinations of Gaussian random variables) and add this to the scaled transmitted symbol vector. Then apply your function from the previous problem to determine an estimate given the true constellation and the noisy signal. The outputs of the function should be the true noise-free transmitted vector (scaled) and the estimated received vector.

I know this sounds long, but it corresponds closely to a MATLAB class homework – I suggest writing and testing it in modular parts to ensure it is working as desired.

3. Write a function that, given the true and estimated symbol sequence determines the number of errors assuming a nondifferential scheme. Divided by N, this estimates probability of error. Do not use any for loops. You might want to make use of the MATLAB function nnz. You also will probably want to avoid near-zero elements being counted as nonzero – use a proper threshold.

4. Repeat the above for a differential scheme (where N transmissions corresponds to $N - 1$ symbols).

   You are done with the hard part! Give yourself a pat on the back! Make sure to drink water. From this point forward, you will simply use the functions you've created to generate nice plots. You should use $N = 10^6$, and generate your symbols from the constellation by using a MATLAB function that samples from a given vector (equiprobably). You are not locked in to using $N = 10^6$ or the given resolution/range in SNR per bit – you can lower or increase either to generate smooth plots in a resaonable amount of time.

5. Note that binary antipodal and binary orthogonal signaling are special cases, with easily defined two-element constellations. In class, we found theroretical formulas for the probability of error as a function of SNR per bit, making this a natural place to start. From -4 dB to 20 dB SNR per bit with 1 dB resolution, use your code to generate a plot of the probability of error for each of these signalling methods, and compare it to a graph of the theoretical SNR per bit.

6. For $M = 4$, 8, 16 and 32, from -4 dB to 20 dB SNR per bit with 1 dB resolution, use your code to generate a plot of the probability of error for M-ary PSK. Provide a legend, and include the $M = 2$ case which you computed above. This corresponds directly to a figure in your text.

7. Perform the above for DPSK (but omit the $M = 2$ case).

8. Repeat for M-ary QAM with $M = 4$, 16, 32 and 64 (you have already computed the $M = 4$ case).

Report Guidelines

I would like you to present your findings in a short report, containing at least four sections:

1. **Introduction**: Give an overview of the project and the theory behind it. What did you set out to explore? Which modulation schemes are you comparing? How are they defined?

2. **Methods**: Describe your MATLAB functions, how they work, and which vector methods you used to avoid loops and speed things up. This can be a short section, but I want to know how your decision function works and how you determine the number of errors.

3. **Results**: The bulk of your report – Each figure generated in the MATLAB code should appear here, with appropriate axis labels, titles and legends. This should be the binary antipodal and orthogonal experimental and theoretical SNR per bit vs probability of error curves, as well as the PSK, DPSK and QAM plots as described above.

4. **Discussion**: How do the schemes compare to each other? How do differential and non-differential methods compare? What is the effect of raising M? Tell me about how long it takes to make these graphs (you can approximate, or use tic and toc).

I would suggest using a LaTeX editor if you can to write this report – it is useful to start learning this skill early – but I will not require it. There is no upper or lower page limit on the report. You must send me the MATLAB files used to generate the results.

Grading Breakdown

1. **Report Content (70%)**: The majority of your grade will be from the *content* of your report, including your responses to the questions and your ability to produce correct graphs (whether or not they are pretty).

2. **Report Style (15%)**: It is important that you present your results with labeled graphs and grammatically correct sentences. Your report should be organized well enough that someone (for example, yourself in the future) might be able to look at it as a reference on the topic it is covering.

3. **MATLAB Style (15%)**: Just as in your homeworks – don't write bad MATLAB code!