ECE 408 Wireless Communications Warmup Project

Kevin Jiang

January 29th, 2020

Objective:

The goal of this project is to design and simulate a communication link in MATLAB, to meet a specific BER (bit error rate) requirements and to maximize the total number of bit rate. A skeleton script has been provided. It simulates a basic Binary Modulation through an AWGN channel. There are three channels in the script. A moderate ISI, severe ISI, and a time-varying multipath channel with moderate ISL. Since the former and latter goals are conflicting, it is very important to attempt different linear modulations schemes and different error-correcting methods such as equalizing and bit encoding in order to reach the desired goals.

Project Steps:

There are two parts of the project.

In the first part, QAM (Quadrature amplitude modulation) is used to modulate and demodulate signals. Initially, these signals are not passed through any filter and have no ISI, and they only have added Gaussian White Noise. The BER (bit error rate) after QAM is compared to an ideal bit error rate (generated using berawgn) to confirm that the QAM code is working, i.e. the shapes of the QAM BER curves are different modulation orders should have similar shape to the ideal BER curve, but have overall higher BER. After this is confirmed, we introduce ISI after modulating each signal and attempt to eliminate this ISI by equalizing the signal before demodulating it. We were able to achieve almost complete elimination of ISI, as shown on the BER vs. SNR plot for Part 1. The curves for "with ISI" and "no ISI" for each M are almost identical. Each curve achieved a BER of less than 10e-4. We chose LMS algorithms with a decision feedback equalizer to achieve this; this was a decision made after testing both LMS and RLS for linear equalizers, as well as RLS for decision feedback equalizers. LMS with DFE simply gives us the best BER.

In the second part, convolutional coding is chosen to achieve BER of 10e-6 at 12 dB SNR over moderate ISI channel. We chose convolutional coding because it runs significantly faster than block coding, while still meeting the requirements of the project. We tried rate 1:2 convolutional coding at first, but it could not keep BER below 10e-6, so we tried rate 2:3 convolutional code. This worked and kept the BER at 10e-7. Our convolutional code was taken from Matlab's documentation on poly2trellis. We also kept the modulation number at 16 so as to squeeze more bits into our 1000 symbols, because our code was strong enough to

keep BER low even with 4000 bits. To add even more bits, we ran 5000 iterations of BER calculation. Even with this number of bits, our code achieves BER on the order of 10e-7 at just 4 dB SNR. We weren't able to get any more non-zero BER at higher SNR values, unless we ran the code with maybe four times as more iterations, but due to time constraints this was not possible.

Code and Figures:

Code, comments and figures are on MATLAB published PDFs.