

ECEn 776 Project 2: Adaptive Equalization

Brian Pratt

April 23, 2007

1 Introduction

Beginning with the communication system represented by Figure 1, and given the characteristics of this system in the form of $x(t) = g(t) * c(t) * g(-t)$, we were asked to design an adaptive MMSE equalizer. The equalizer uses the LMS algorithm in order to counter-act the effect that the channel, whose characteristics are unknown to the receiver, has on the incoming data.

The adaptive equalizer is an FIR filter with coefficients that change over time based on the channel response. In this case, the channel is static, but this type of filter can adapt to a channel whose response changes over time. The LMS algorithm, as presented in section 11.1.2 of Proakis, and as studied in class, updates the equalizer coefficients according to the following formula:

$$\hat{\mathbf{C}}_{k+1} = \hat{\mathbf{C}}_k + \Delta \varepsilon_k \mathbf{V}_k^* \quad (1)$$

where the vector $\hat{\mathbf{C}}$ represents the channel coefficients, ε_k is the error in the estimate of the received symbol, the vector \mathbf{V}_k is the section of the input data which contributes to the filter output for the k th received symbol, and Δ (or μ) is the update weighting factor. The adaptive MMSE equalizer is illustrated in Figure 2.

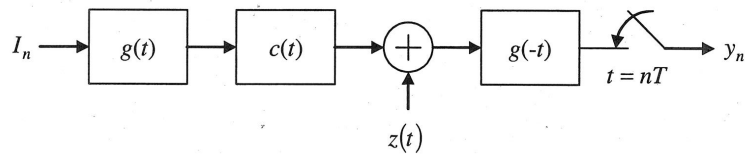


Figure 1: The complex-baseband equivalent communications system

2 Building the Adaptive Filter

I wrote a script in Matlab to simulate the adaptive filter. I made the script flexible enough that I could change the various parameters affecting the adaptive filter to see the results. I will discuss my choice of parameters in Section 3. I used Equation 1 to adapt the filter to the channel response, which was updated every sample, as illustrated in Figure 2.

The channel corrupted the sent data enough that a training period was necessary to get the filter adaptation started in the right direction. With no training period, the filter failed to converge and cancel out the effects of the channel. I therefore preceded my data sequence with a sequence of known data, simulating the transmitter sending a prearranged sequence to the receiver so the filter would know how to correct for the errors introduced by the channel. After the training sequence completed, the filter was able to use the estimates based on the filter outputs themselves (simply a *complex sign* operation for QPSK).

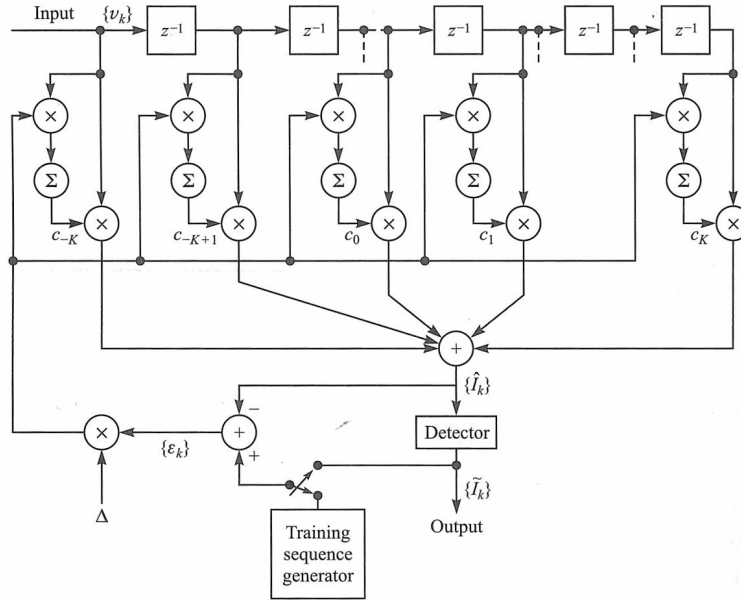


Figure 2: Linear adaptive equalizer based on the MSE criterion (Proakis, pg. 664)

3 Choosing Parameters

The adaptive MMSE equalizer can be customized with various parameters, such as those mentioned in Section 2. These parameters affect the performance of the filter and are interrelated in many cases. The parameters I made customizable are:

- The length of the adaptive filter
- The length of the training sequence
- The adaptation step size μ (or Δ)

To improve performance, I also made an option to change μ after a specified amount of time. In this way, the step size could be large in the beginning to facilitate quick movement to the optimal coefficient values and later reduced to a small value so the filter could more accurately converge on the desired coefficient values. This system could, of course, be improved by gradually altering the value of μ over time. Even this simple modification, however, improved performance significantly.

To choose these values, I experimented with the parameters and observed the response of the adaptive filter. I considered the effectiveness of the filter in adapting to the channel, the rate of convergence for the coefficients, and the cost (due to the length) of the filter.

3.1 Filter Length

The length of the filter affects its performance in terms of effectiveness at canceling out the effect of the channel as well as the rate of convergence. I observed that a channel that was too small was not effective. I also saw that a very long filter adapted to the channel well, but took a long time to converge on the final coefficients. With a longer filter, I also needed a longer training sequence. In the end, I chose a filter length of 17 ($17 = 2 * K + 1, K = 8$), which is fairly short and had very good performance.

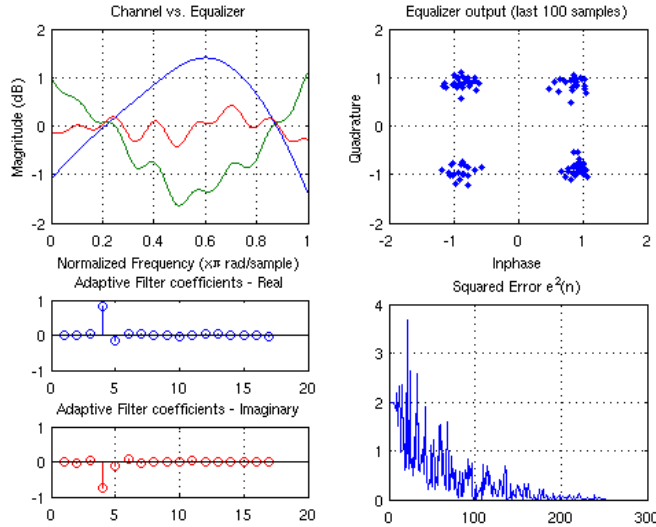


Figure 3: Snapshot of adaptive equalizer in action (in simulation)

3.2 Training Sequence length

The length of the training sequence affected the ability of the adaptive filter to lock in on the correct coefficients. With little or no training data (less than 30 symbols), the filter was unable to converge in most cases. A training sequence of 200 (for $K = 8$) was the maximum that was useful. This can be seen in Figure 5(b), where the error bottoms out at around 200 symbols. At this point, the channel has locked in for the most part and further training is unnecessary. As mentioned before, for longer filters, a longer training sequence is required. For example, with $K = 20$, the error does not bottom out until about 450 symbols.

3.3 Adaptation Step Size μ

The adaptation step size is also interrelated with the other parameters. For longer filter lengths, a smaller value of μ is required than for shorter filters. The value of μ affects the rate of convergence as well as the quality of the match in the end. A larger value of μ converges faster, but is noisy even after converging. A smaller value of μ converges more slowly, but is more accurate in the end. The accuracy after convergence is due to the adaptive filter overshooting the optimal values. A larger μ causes larger overshoots while these are minimized with smaller values of μ .

Because of the trade-offs involved in choosing a value for μ , I implemented my filter such that I could change its value after a specified amount of time. This way, the filter could converge quickly and settle better on the coefficients. For $K = 8$, I used a starting value of 0.01 for μ , and changed it to 0.008 after the training sequence had completed after 200 symbols.

4 Results

Figures 4(a), 4(b), 5(a), and 5(b) demonstrate the performance of my adaptive filter. Each of the plots were created using the following parameters:

- SNR: 30 dB
- Adaptive Filter Length: 17 ($K = 8$)

- Training Sequence Length: 200 symbols
- Starting μ : 0.01
- Ending μ : 0.008
- When to change mu: after 200 symbols

Figure 4(a) shows the first 100 unequalized outputs that were fed into the adaptive equalizer. These are a good sampling of the raw data coming out of the matched filter which were previously corrupted by the channel.

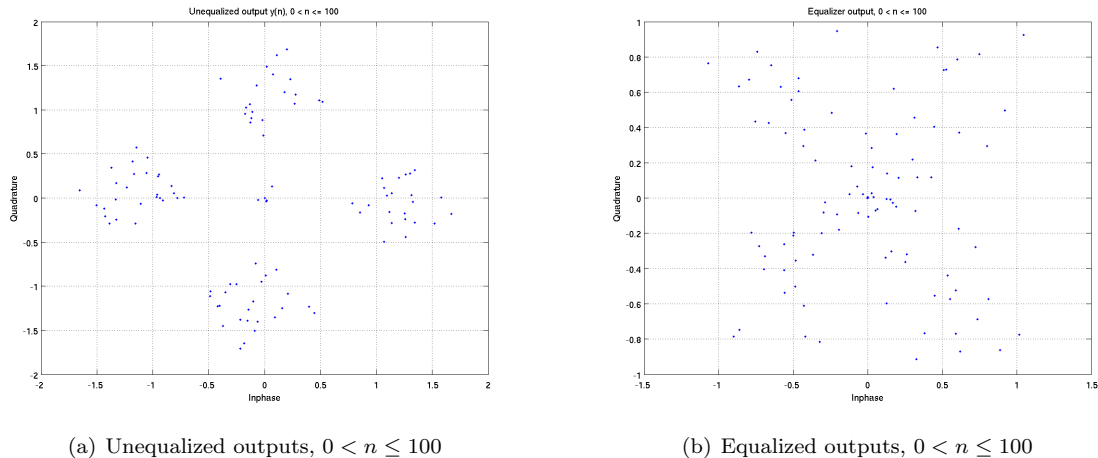


Figure 4: The first 100 unequalized and equalized outputs in the filter system

Figure 4(b) shows the first 100 symbols coming out of the equalizer. This is while the equalizer is still in its training sequence. The symbols have already started to improve over the unequalized samples, but the filter has not had time to adapt completely by the end of this sequence.

Figure 5(a) shows the 5000th to the 5100th samples coming out of the equalizer. At this point, the equalizer has had plenty of time to adapt to the channel. You can see that the symbols coming out of the equalizer at this point have very little error. In fact, most of the variation observed is due to the simulated Gaussian noise (the SNR being 30 dB in this simulation).

Figure 5(b) plots the squared error at the output of the filter. This is the output of the filter \hat{I} minus the estimated data symbol \tilde{I} . The squared error is large to begin with but quickly decays (roughly exponentially). In this example, the error has reached its floor after about 200 samples.

4.1 Signal to Noise Ratio

The signal to noise ratio modeled in the system also affects the system's performance. I performed some simulations to calculate the bit error rate for the adaptive equalizer system under different levels of SNR to compare them with the performance under ideal conditions. Figure 6 shows an ideal QPSK system as well as my adaptive equalizer system. The adaptive equalizer system does not perform as well as the ideal QPSK system. The ideal system's performance is on the order of 2 or 3 dB better than the adaptive system. This is reasonable because in the ideal system, a single symbol pushed out of the correct decision region by noise only affects that symbol. In the adaptive filter system, the same situation would affect that symbol detection as well as alter the coefficients of the adaptive filter. Thus noise has a larger effect on the adaptive filter system and has poorer bit error rate performance.

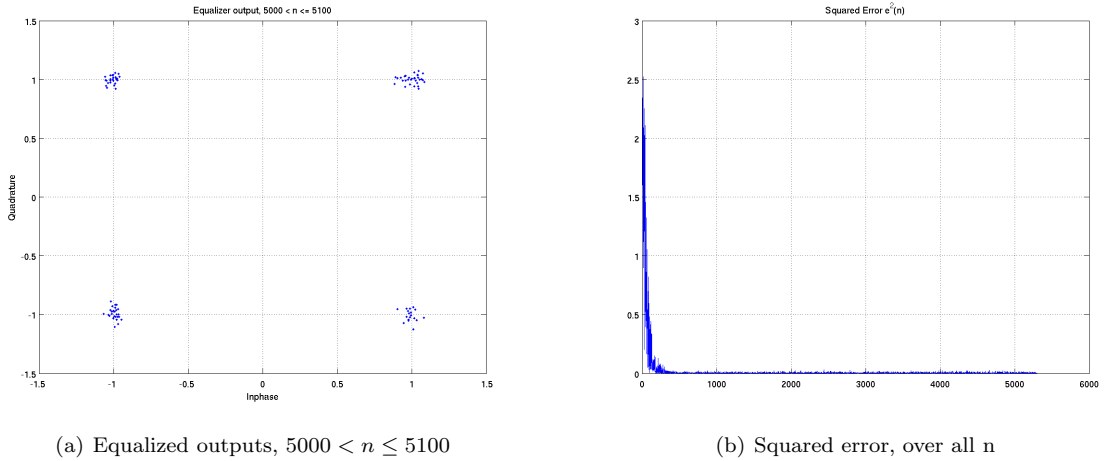


Figure 5: The resulting equalized outputs and the squared error of the system

5 Conclusions

Adaptive equalizers are widely used in non-ideal and changing channels. The adaptive nature of the filter allows it to actively cancel out the adverse effects of the channel even without knowing the characteristics of the channel beforehand. In this report, I illustrated how each of the parameters characterizing the filter affected its performance. The parameters I examined were: filter length, training sequence length, and adaptation step size. These parameters are all interrelated so that changing one parameter means changing another in order to compensate. I also examined the effect of changing the signal to noise ratio of the system affected its bit error rate performance and found that its performance is 2-3 dB worse than a standard QPSK system in an ideal channel.

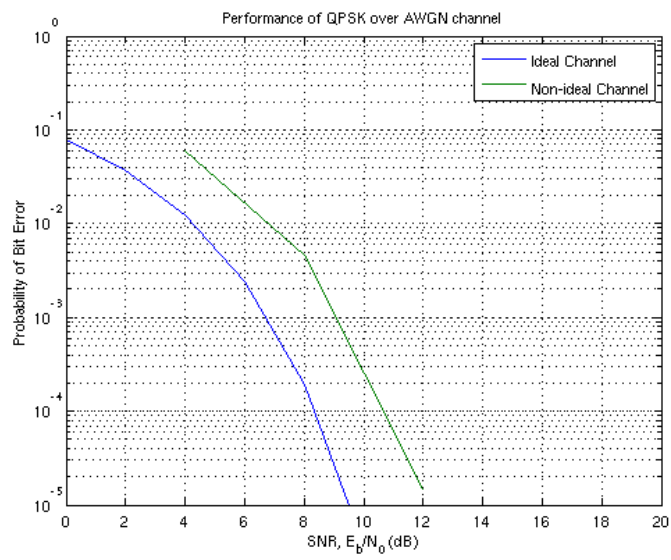


Figure 6: Bit error rate performance of adaptive equalizer system over given channel (blue) vs. optimal performance of QPSK system in ideal channel (green)