FINAL

"Over the channel and through the noise": A Study in Adaptive Filter Design with DFE

Alexander M. Duda
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, USA
e-mail: amduda@illinois.edu

***Abstract*—A message is sent over a length-6 ISI channel and through additive white Gaussian noise. Given a sequence of input-output pairs, we investigate adaptively designing a filter that is optimal in the MMSE sense. In particular, we study three different algorithms: LMS, Sign-LMS, and RLS. To aid us in our estimation of the original message, a decision feedback equalizer (DFE) is implemented. The results obtained seem to agree with expectation: LMS achieved a 99.59% bit recovery accuracy after 17 iterations, Sign-LMS achieved a 99.49% bit recovery accuracy after 23 iterations, and RLS achieved a 99.59% bit recovery accuracy after 3 iterations. An introduction, background theory, experimental results with analysis, a future work section, and a conclusion are included.**

***Index Terms*—DFE; MMSE; adaptive filter design; computational study; LMS; Sign-LMS; RLS; message recovery**

I. Introduction

The primary goal of the work detailed is to decode a text message that has been sent (after having been translated into a sequence of bits and then mapped sequentially to binary phase shift keying (BPSK) symbols) over a length-6 inter-symbol interference (ISI) channel and through additive white Gaussian noise (AWGN). The received signal-to-noise ratio (SNR) is 13 dB. Preceding the information-bearing signal transmission, a length-350 sequence of training bits is sent. The known input-output pairs enable one to estimate the channel plus noise by designing an adaptive filter (length-6) that, in this case, attempts to achieve optimality in the MMSE sense. Figure 1 below illustrates the setup.

$$\begin{tikzcd} x_t \arrow[r] & \boxed{\text{h}} \arrow[r] & y_t \\ x_t \arrow[r] & \boxed{\text{g}} \arrow[r] & \hat{y}_t \end{tikzcd}$$

Fig. 1. Implementation for optimizing filter coefficients

The labels mean the following: x_t input training bits, h the actual channel plus noise, y_t the actual noisy observed output during training, g the estimated channel plus noise, and $\hat{y}_t$ the estimated output obtained by convolving x_t with g. We use various algorithms to estimate the filter g that minimizes the MMSE, $\mathrm{E}\{|y_t - \hat{y}_t|^2\}$. In particular, we use the standard LMS, Sign-LMS, and RLS adaptive filter update algorithms. We then implement a decision feedback equalizer (DFE) scheme that enables us to estimate the actual information-bearing input bits x_a given only the testing noisy output signal y_a. In particular, we use the method shown in Figure 2.

$$\begin{tikzcd} x_a \arrow[r] & \boxed{\text{h}} \arrow[r] & y_a \\ \hat{x}_a \arrow[r] & \boxed{\text{g}} \arrow[r] & \hat{y}_a \end{tikzcd}$$

Fig. 2. DFE implementation

To be clear, we do not know x_a (this is what we want to decode) or h. All we have is the observed noisy output signal y_a and g (trained previously). We let $\hat{x}_a[n] = \pm 1$, add it to the standard "sliding window" $\vec{x}_{\pm 1}[n] = [\pm 1, x[n-1], ..., x[n-5]]^T$, and compute the error $e_{\pm 1}[n] = (y_a[n] - \vec{x}_{\pm 1}[n]^T \vec{g})$. We then make a decision about the input by comparing the magnitude of the error: If $|e_{+1}[n]| > |e_{-1}[n]|$ then $\hat{x}_a[n] = -1$ else $\hat{x}_a[n] = +1$. We use this simple scheme as our DFE. It is used with all three algorithms. With the general framework and implementation outlined, we can examine some of the theoretical background of our chosen algorithms.

II. Theoretical Background

Here we provide a short and basic theoretical background for each algorithm to motivate their use and enable the reader to implement them.

A. LMS

As discussed in class [2], for designing a length-6 filter $\vec{g} = [g[0], g[1], \ldots, g[5]]^T$, we use the update equation:

$$\vec{g}^{(n+1)} = \vec{g}^{(n)} + 2\mu \vec{x}[n] e[n] \tag{1}$$

where the "sliding window" of $x[n]$'s is $\vec{x}[n] = [x[n], x[n-1], \ldots, x[n-5]]^T$, the error function is $e[n] = y[n] - \vec{x}[n]^T \vec{g}^{(n)}$, and the learning rate is $\mu \approx (\lambda_{\text{min}} + \lambda_{\text{max}})^{-1}$. We obtain μ by estimating R_{xx} of the 350-length input training bits, obtaining the largest (λ_{max}) and smallest (λ_{min}) eigenvalues of R_{xx}, and using the "optimal" estimated learning rate (as shown in class) $(\lambda_{\text{min}} + \lambda_{\text{max}})^{-1}$. In our experiments (as will be discussed in the "Analysis of Experimental Results" section), the μ for LMS was approximately 0.00055.

With a "reasonable" μ selected, we will likely have mean convergence to an MMSE optimal filter $\vec{g}$ (if there was no noise)). The presence of the AWGN just means that our filter will not able to perfectly match the optimal. Furthermore, assuming the channel plus noise do not change between the training and testing periods, our estimated $\vec{g}$ should work well.

B. Sign-LMS

Here we use an adaptive learning rate with an approach that has lower computational complexity. In particular, we use the following update rule:

$$\vec{g}^{(n+1)}[j] = \vec{g}^{(n)}[j] + 2\mu_j \text{sign}(e[n]x[n-j]) \tag{2}$$

where $e[n]$ is defined as in LMS and $x[n-j]$ represents the j^{th} element from the sliding window of training input bits $\vec{x}[n]$ (same as in LMS). For x_z complex, the sign function works as follows:

$$\text{sign}(x_z) = \frac{x_z}{|x_z|}. \tag{3}$$

The familiar scenario (which is a sub-case) is for x_r real, which works as follows:

$$\text{sign}(x_r) = \begin{cases} 1 & \text{if } 0 < x_r \leq 1 \\ 0 & \text{if } x_r = 0 \\ -1 & \text{if } -1 \leq x_r < 0 \end{cases}$$

The adaptive learning rate rule works as follows:

$$\mu_j^{(n+1)} = \begin{cases} \mu_j^{(n)} & \text{if sign does not change} \\ 0.5 * \mu_j^{(n)} & \text{if sign changes} \end{cases}$$

As mentioned in class, though there is low computational complexity, it may take longer to converge to the optimal filter.

C. RLS

The Recursive Least Squares Algorithm (RLS) is known for having high computational complexity but obtaining the optimal filter coefficients extremely quickly (and making the learning rate approach 0). This is a particularly interesting approach and has the following update equation (as shown in class):

$$\vec{g}^{(n+1)} = \vec{g}^{(n)} - \hat{R}_{xx}^{-1}[n]\nabla_{\vec{g}}\epsilon^2 \tag{4}$$

However, in a more detailed fashion we can formulate the recursion in the following manner [3]

$$\vec{g}^{(n+1)} = \vec{g}^{(n)} + e[n]\vec{K}[n] \tag{5}$$

where:

$$\vec{K}[n] = \frac{P[n]\vec{x}[n]}{f + \vec{x}[n]^T P[n]\vec{x}[n]} \tag{6}$$

where $\vec{x}[n]$ is defined as in LMS. The forgetting factor f can take on values $\in (0, 1]$; values less than one mean that the algorithm will place a larger weight on the currently observed value and a smaller weight on values observed in the past [4]. We will discuss how to choose the value of f more in the Experimental Results section. $P[n]$ is the estimated inverse autocorrelation of $\vec{x}[n]$. $P[n]$ is initialized as follows:

$$P[0] = \delta^{-1}\mathbb{I}_{LxL} \tag{7}$$

where $\mathbb{I}_{LxL}$ represents the L-by-L dimension identity matrix (where L is the length of the desired filter, which in our case is 6) and δ is the regularization factor, which ensures the algorithm will converge "smoothly and consistently to the optimal Wiener solution, especially in the presence of additive noise"[1]. The precise selection of δ is discussed in the Experimental Results section. The standard RLS algorithm uses the following recurrence equation to update P:

$$P(n+1) = f^{-1}P(n) - f^{-1}\vec{K}[n]\vec{x}[n]^T P[n] \tag{8}$$

This concludes the theoretical details necessary for context and implementation for the RLS algorithm. Now we may proceed to the Analysis of Experimental Results section where we examine how these algorithms perform (as well as how the parameters for RLS are determined).

III. Analysis of Experimental Results

We examine the performance of the different algorithms below. It seems that the implementation has definite limits. By estimating the channel plus noise in the fashion we did, as well as estimating the input messages with our DFE scheme, it definitely enabled a cascading of error. A quick glance at Appendix B (where the actual estimated input messages are included) will reveal that there was rarely a single bit error; usually one bit error would lead to two which would lead to three until at some point it would stop. This is certainly due to the way we insert our best estimate into the sliding window vector $\hat{x}[n]$ during the decoding stage (where if an error is made, future predictions will be based on it and therefore could be more likely to produce more errors). This implementation still seemed to do quite well but a change in implementation might improve performance (more on this matter is discussed in the Future Work section). If time allowed, it could be of interest to examine the likelihood of having n consecutive bits incorrect, as well as statistics regarding likelihood of error given recent accuracy. This is beyond the scope of the current study.

A. LMS

We examine the performance of the LMS algorithm in terms of the contour plot (which outlines the convergence behavior of a pair of filter coefficients) and the MSE.

1) Contour Plot: In Figure 3 we compare the real parts of the filter coefficients $g[3]$ and $g[5]$ over the final 175 steps of running LMS. To be clear, for all such plots in the sections below, $g[5]$ is the vertical axis and $g[3]$ is the horizontal axis. We observe that even while the MSE is minimized, the coefficients still jump around within a tight cluster ($\text{Re}\{g[3]\} \in (-0.917, -0.9125)$, a window of size about 0.0045 and $\text{Re}\{g[5]\} \in (0.054, 0.0605)$, a window of size about 0.0065). They also move along in a "zig-zag" pattern that frequently revisits similar points, whose steps are along two different diagonals (which matches the elliptical contour plot we would expect to see in "g-space"). In a similar fashion, in Figure 4 we see the motion of $\text{Im}\{g[3]\}$ and $\text{Im}\{g[5]\}$. Though, in this particular case $\text{Im}\{g[3]\} \in (-0.44, -0.4335)$, a window of size about 0.0065 and $\text{Im}\{g[5]\} \in (-0.0984, -0.093)$, a window of size about 0.0054.

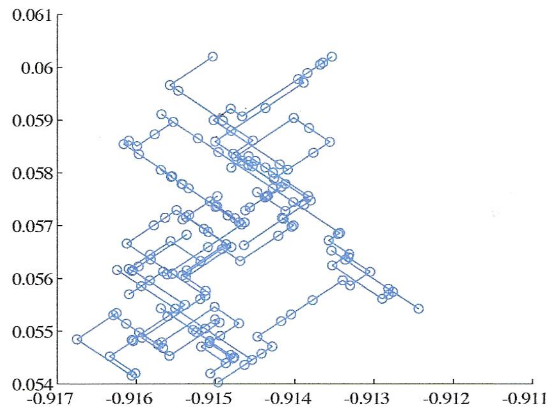

Fig. 3. LMS with DFE Contour for $\text{Re}\{g[3]\}$ vs. $\text{Re}\{g[5]\}$ during the last 175 steps.

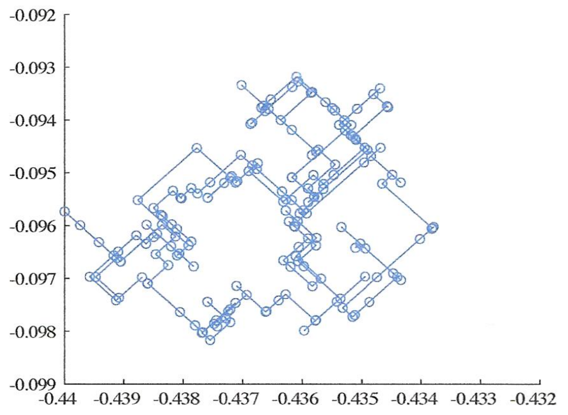

Fig. 4. LMS with DFE Contour for $\text{Im}\{g[3]\}$ vs. $\text{Im}\{g[5]\}$ during the last 175 steps.

2) Error Plot: In Figure 5 we summarize the MSE over the entire run-time, which required 17 iterations through the training data for the filter to exhibit a mean convergence about an acceptable result (and for the bit error of the message to be minimized). Considering the training data is 350 bits long, it took nearly 6 iterations for the filter's MSE error to drop into a "reasonable range". The later iterations did not show a large improvement in the MSE but there was an observed improvement in bit error (which ended up settling to 144 bits wrong, which means 99.59% correct). To see the decoded message using LMS, please see Appendix B.

Now that we have examined, in detail, the performance of the LMS algorithm in terms of its contour plots and MSE, we continue to the next algorithm.

B. Sign-LMS

We examine the performance of the Sign-LMS algorithm in terms of the contour plot (which outlines the convergence behavior of a pair of filter coefficients) and the MSE.

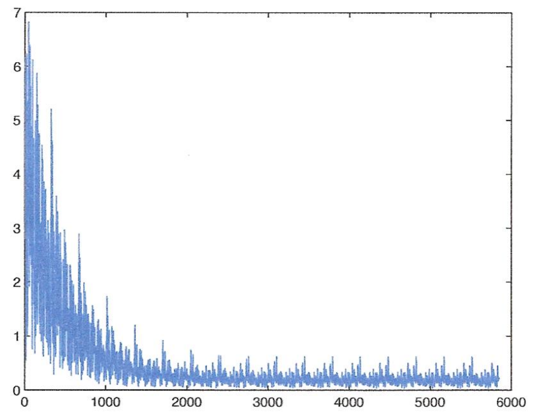

Fig. 5. LMS with DFE MSE over time (showing all 17 iterations through the training bits)

1) Contour Plot: In Figure 6 we compare the real parts of the filter coefficients $g[3]$ and $g[5]$ over the final 175 steps of running Sign-LMS. As with LMS, even while the MSE is minimized, the coefficients still jump around within a tight cluster ($\text{Re}\{g[3]\} \in (-0.9065, -0.899)$, a window of size about 0.0075 (1.67 times larger than LMS) and $\text{Re}\{g[5]\} \in (0.054, 0.063)$, a window of size about 0.009 (1.384 times larger than LMS)). They also move along in a "zig-zag" pattern that frequently revisits similar points, whose steps are along two different diagonals (which matches the elliptical contour plot we would expect to see in "g-space"). In a similar fashion, in Figure 7 we see the motion of $\text{Im}\{g[3]\}$ and $\text{Im}\{g[5]\}$. Though, in this particular case $\text{Im}\{g[3]\} \in (-0.443, -0.436)$, a window of size about 0.007 (1.077 times larger than LMS) and $\text{Im}\{g[5]\} \in (-0.104, -0.097)$, a window of size about 0.007 (1.3 times as large as LMS) . Thus, it is clear from the contour plots that Sign-LMS does not enjoy a convergence that is as tight as LMS.

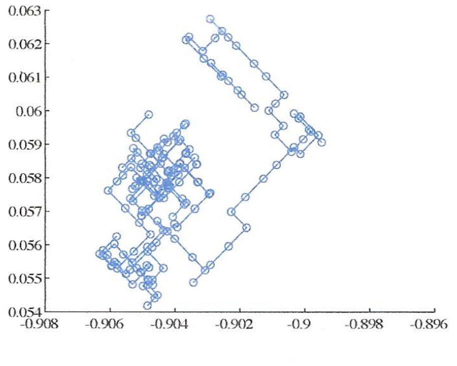

Fig. 6. Sign-LMS with DFE Contour for $\text{Re}\{g[3]\}$ vs. $\text{Re}\{g[5]\}$ during the last 175 steps.

2) Error Plot: In Figure 8 we summarize the MSE over the entire run-time, which required 23 iterations through the training data for the filter to exhibit a mean convergence about an acceptable result (and for the bit error of the message to be minimized). Considering the training data is 350 bits long,

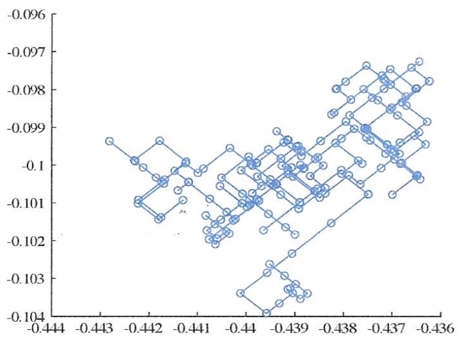

Fig. 7. Sign-LMS with DFE Contour for Im$\{g[3]\}$ vs. Im$\{g[5]\}$ during the last 175 steps.

it took nearly 14 iterations for the filter's MSE error to drop into a "reasonable range"(which is almost 2.3 times longer than LMS; though it did finally exhibit an acceptable mean convergence in only 1.35 times as long as LMS so it is not that terrible). The later iterations did not show a large improvement in the MSE but there was an observed improvement in bit error (which ended up settling to 178 bits wrong, which means 99.49% correct). To see the decoded message using Sign-LMS, please see Appendix B.

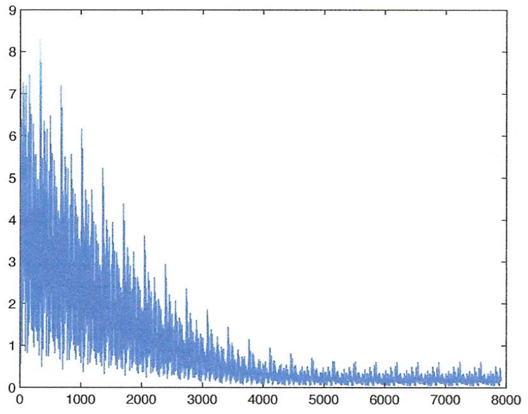

Fig. 8. Sign-LMS with DFE MSE over time (showing all 23 iterations through the training bits)

C. RLS

To begin, we examine how to select the parameters δ and f. Instead of selecting δ in an ad-hoc fashion it has been suggested that one should choose it as follows[1]:

$$\delta = \frac{L(1+\sqrt{1+\text{SNR}})}{SNR}\text{Var}(x) \tag{9}$$

where SNR represents the signal-to-noise ratio, which is given as 13dB. Thus, in our situation we should select:

$$\delta = \frac{6(1+\sqrt{1+13})}{13}\text{Var}(x) \tag{10}$$

where the Var(x) can be estimated from the training data as about 1.002. This is the value used in our experiments. Concerning f, a parameter sweep was carried out and it was clear that 0.999 was an optimal value (smallest that still obtained best results). The high value can be attributed the short length of the desired filter. Now that it is clear how the parameters should be selected, we examine the performance of the RLS algorithm in terms of the contour plot (which outlines the convergence behavior of a pair of filter coefficients) and the MSE.

1) Contour Plot: In Figure 9 we compare the real parts of the filter coefficients $g[3]$ and $g[5]$ over the final 175 steps of running RLS. As with LMS and Sign-LMS, even while the MSE is minimized, the coefficients still jump around within a tight cluster (Re$\{g[3]\} \in (-0.918, -0.9115)$, a window of size about 0.0065 (1.44 times larger than LMS and 0.867 of Sign-LMS) and Re$\{g[5]\} \in (0.05, 0.06)$, a window of size about 0.01 (1.538 times larger than LMS and about 1.11 times larger than Sign-LMS)). They also move along in a "zig-zag" pattern that frequently revisits similar points, whose steps are along two different diagonals. Likewise, in Figure 10 we see the movement of Im$\{g[3]\}$ and Im$\{g[5]\}$. However, in this particular case Im$\{g[3]\} \in (-0.443, -0.433)$, a window of size about 0.01 (1.54 times larger than LMS and 1.43 times larger than Sign-LMS) and Im$\{g[5]\} \in (-0.0986, -0.0913)$, a window of size about 0.0073 (1.352 times as large as LMS and 1.043 times as large as Sign-LMS) . Thus, it is clear from the contour plots that in three cases RLS does not enjoy a convergence that is as tight as LMS or Sign-LMS. However, as we shall see from an inspection of the MSE, this has not ruled out the effectiveness of the algorithm.

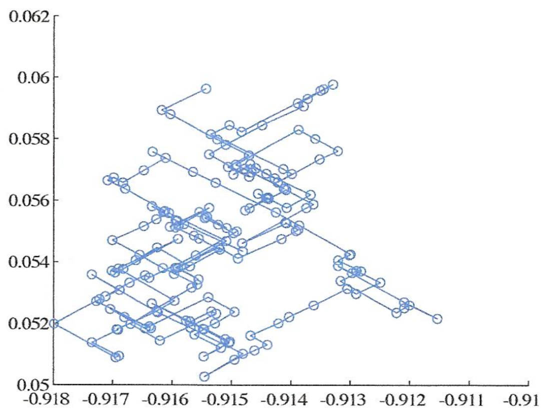

Fig. 9. RLS with DFE Contour for Re$\{g[3]\}$ vs. Re$\{g[5]\}$ during the last 175 steps.

2) Error Plot: In Figure 11 we summarize the MSE over the entire run-time, which required just 3 iterations through the training data for the filter to exhibit a mean convergence about an acceptable result (and for the bit error of the message to be minimized). Considering the training data is 350 bits long, it took less than 1 iteration for the filter's MSE error to drop into a "reasonable range"(which is considerably faster than either of the other algorithms!). The later iterations did not show a large improvement in the MSE but there was an

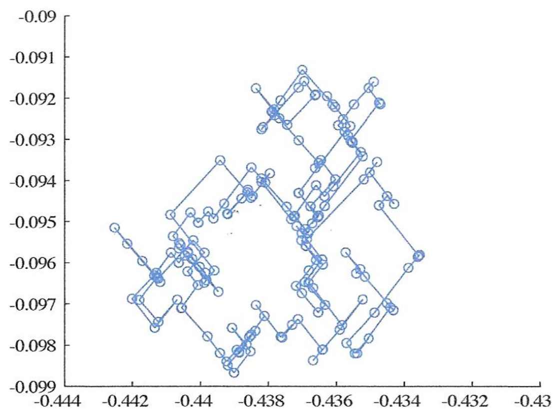

Fig. 10. RLS with DFE Contour for Im$\{g[3]\}$ vs. Im$\{g[5]\}$ during the last 175 steps.

observed improvement in bit error (which ended up settling to 144 bits wrong, which means 99.59% correct– just as good as LMS and 34 bits better than Sign-LMS). To see the decoded message using RLS, please see Appendix B.

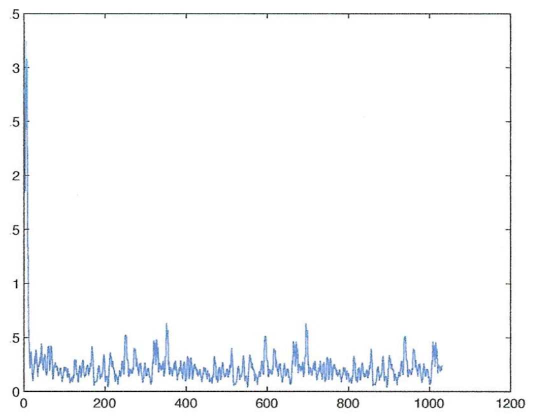

Fig. 11. RLS with DFE MSE over time (showing all 3 iterations through the training bits)

IV. Future Work

A. Different Implementations

It is evident that the implementation used was very simple and most likely did not lead to the best possible results. It remains to be seen how much could be improved by adjusting the implementation. However, if time was not an issue, the following would be interesting avenues of exploration.

1) Keep Main Implementation, Adjust DFE: It is evident that our implementation paired with the DFE certainly lead to compounding errors. It seems that a more sophisticated and subtle DFE could possibly improve results. For instance, taking the hint from Sign-LMS, it could be worthwhile to investigate how sign changes in the error between consecutive steps could be used to better predict whether the original input bit was a -1 or +1. Keeping track of such changes might enable one to better enumerate confidence in a given selection. Even if this criteria was used in addition to the current DFE, it could prove useful. For instance, by keeping track of sign changes, one could set a threshold to say that "though the naive DFE indicates one should choose -1, perhaps +1 is a better choice". The exact details of such an adjustment should be studied!

2) Change Main Implementation Filter: We took for granted that we should try to model the channel plus noise. However, there is another very obvious method that could prove fruitful as well. Namely, it would be wise, to try estimating g as shown in Figure 12 with $\hat{g}$.

$$x_t \longrightarrow \boxed{\text{h}} \longrightarrow y_t \longrightarrow \boxed{\text{g}} \longrightarrow x_t$$

$$x_t \longrightarrow \boxed{\text{h}} \longrightarrow y_t \longrightarrow \boxed{\hat{g}} \longrightarrow \hat{x}_t$$

Fig. 12. Alternative Implementation

In this scenario, it seems that the training stage could wisely take place in the frequency domain. Once the training is complete, we could use a DFE that would act like sign$(\hat{x}_t)$. In this sense, it would be elegant to have the output of the filter be an estimate of the original message immediately. It is not clear if performance would be improved by this method but it seems likely that the cascading of errors might be less likely to occur.

3) Frequency Domain: For either of the implementations discussed, it would be interesting to compute the desired filter during the training phase in the frequency domain. For instance, in the original implementation (as shown in Figure 1), g in the frequency domain: $g = \text{IDTFT}(\frac{Y(e^{j\omega})}{X(e^{j\omega})})$.

B. Different Algorithms

There are a number of different adaptive filter algorithms that could follow very naturally and nicely augment the current study. For instance, Normalized leas mean squares (NLMS) seems to provide a method to choose a learning rate that better guarantees stability (and is less sensitive to scaling of the input). Another relevant option, assuming the unknown filter is LTI and the noise stationary, would be the Wiener filter. It would be fascinating to see in practice how these two algorithms perform in the various implementations mentioned, in comparison to the original three algorithms.

C. Performance Metric

One additional metric that should be considered is exact computation required– in terms of actual computations run for a given algorithm on a given computer. This might provide a clearer perspective on the precise tradeoffs one has to make between computational complexity and speed of mean convergence.

V. Conclusion

In this study we examined the performance of the LMS, Sign-LMS, and RLS algorithms in decoding a message sent over an ISI channel and through AWG noise. We provided a context for the problem, a basic theoretical background on the different algorithms, a detailed analysis of the experimental results (with a specific focus on the contour plots and MSE), and we discussed some future avenues of exploration that could be taken immediately as a means of furthering this study. From a decoding perspective, the LMS and RLS algorithms were the clear champions. In this sense (as well as the other metrics reviewed) the algorithms performed as expected. To summarize, the LMS algorithm, with 144 bits incorrect, achieved a 99.59% recovery rate in 17 iterations. The Sign-LMS algorithm, with 178 bits incorrect, achieved a 99.49% recovery rate in 23 iterations. The RLS algorithm, with 144 bits incorrect, achieved a 99.59% recovery rate in only 3 iterations.

Acknowledgment

The author would like to thank Professor Hasegawa-Johnson and Yen-Wei Huang for their great efforts throughout the semester to help us all to learn DSP!

References

[1] J. Benesty and S. Ciochina, *Regularization of the RLS Algorithm*, IEICE Trans. Fundamentals, Vol. E94-A, no.8, 99. 1629-1629, August 2011. [Online]. Available: http://externe.emt.inrs.ca/users/benesty/papers/ieice_aug2011.pdf

[2] M. Hasegawa-Johnson, *ECE 551 Lecture Notes*, 25 October 2011.

[3] National Instruments Support, *Recursive Least Squares (RLS) Algorithms (Adaptive Filter Toolkit)*, June 2008. [Online], Available: http://zone.ni.com/reference/en-XX/help/372357A-01/lvaftconcepts/aft_rls_algorithms/.

[4] National Instruments Support, *Adaptive Filter Algorithms (Adaptive Filter Toolkit)*, June 2008. [Online], Available: http://zone.ni.com/reference/en-XX/help/372357A-01/lvaftconcepts/aft_algorithms/#calculate_yn.