

ENEL 671 Adaptive Signal Processing Project Report

Pouyan Keshavarzian, *Student Member, IEEE*,

pkeshava@ucalgary.ca

UCID 10053710

Electrical & Computer Engineering, University of Calgary

Abstract—The purpose of this project is to analyze the performance of three different adaptive algorithms. These algorithms, which can be used for a variety of applications such as echo cancellation and beamforming, are implemented for the purposes of equalizing a dispersive channel. The first algorithm investigated is the Least Mean Squares (LMS). The LMS filter uses a search method to minimize the mean square error. The performance of the LMS is then compared to the Recursive Least Squares (RLS) method, which minimizes a weighted linear least squares cost function. Finally the Recursive Least Squares Lattice algorithm is implemented in the third project for joint process estimation. The algorithms are tested against multiple channels with varying eigenvalue spreads. Furthermore different design parameters such as step size and filter order are evaluated. The performance with regards to convergence speed and complexity is discussed. The RLSL appears to have performance advantages in terms of convergence speed at the cost of significant increase in complexity.

I. INTRODUCTION

ADAPTIVE filtering is used for a variety of applications. When communicating across a dispersive channel, a signal is distorted by effects of multipath, noise etc REFERENCE. To recover the correct message at the receiver, an adaptive filter is implemented to equalize the channel. In general, an algorithm is used to calculate the optimum filter coefficients to minimize the error between transmit and receive. In practice, a signal transmitted along a channel contains a preamble, which is a known sequence of data that is used to estimate the parameters of the channel. The statistics are assumed to be stationary in the window of time that each preamble is sent.

The most common adaptive equalizers involve the LMS, RLS, and RLSL algorithms. The performance of these methods varies and they each have their advantages and drawbacks. The LMS is relatively simplistic in terms of hardware/software resources. However, its drawback is that it uses a stochastic search gradient and which limits its ability to find an optimum solution, is inefficient and not practically capable of removing all distortions. The RLS algorithm is typically an order of magnitude faster because it whitens the input data by using the inverse correlation matrix of the data, assumed to be of zero mean REF. The RLSL algorithm uses a lattice predictor along with a stage that estimates the error of the next filter order at each time step.

The methodology used for designing of the filters in this project will be discussed. Afterwards a detailed analysis of the results for each project and a summary of their implications will be discussed. Finally a conclusions section will highlight the general results drawn from each experiment.

II. EQUALIZATION METHODOLOGY

ADAPTIVE equalization is designed to remove unwanted signals from a channel output which can be caused by multipath or other distortions. In Figure 1 the general block diagram that is simulated in

these experiments is outlined. A data source that generates a random BPSK signal goes through a dispersive channel that has three paths and white noise. The adaptive filter reduces the effects of these signal distortions to generate the output. Before the specifics of this block diagram is discussed a few general principles of Wiener filter theory will be discussed. The autocorrelation matrix is defined as

$$\mathbf{R} = E[\mathbf{u}(n)\mathbf{u}(n)^T]$$

and the cross-correlation:

$$\mathbf{p} = E[\mathbf{u}(n)d(n)]$$

it can be shown from REF that the Wiener-Hopf or Normal Equation can be solved for the optimum tap weights:

$$\mathbf{R}\mathbf{w} = \mathbf{p}$$

although this is computationally wasteful and the matrix \mathbf{R} may not always be invertible therefore adaptive algorithms are used to determine the tap weight vector.

$$\mathbf{w} = [w_0 \quad w_1 \quad \dots \quad w_{M-1}]^T$$

The output of the channel, which is the input to the adaptive filter is defined as:

$$\begin{aligned} \mathbf{u}(n) &= [u(n) \quad u(n-1) \quad \dots \quad u(n-M+1)]^T \\ &= \mathbf{h}^T(n)\mathbf{a}(n) + \mathbf{v}(n) \end{aligned}$$

where \mathbf{h} is known as the channel impulse response. The impulse response for the four channels with varying distortion are defined as

Channel	\mathbf{h}_1	\mathbf{h}_2	\mathbf{h}_3
1	0.2194	1	0.2194
2	0.2798	1	0.2798
3	0.3365	1	0.3365
4	0.3887	1	0.3887

TABLE I: Channel Parameters

To compare the output of the filter to the truth data source adequate delay must be added to the BPSK data. The optimal delay is given by half the filter order, that is:

$$\Delta = \frac{M+1}{2}$$

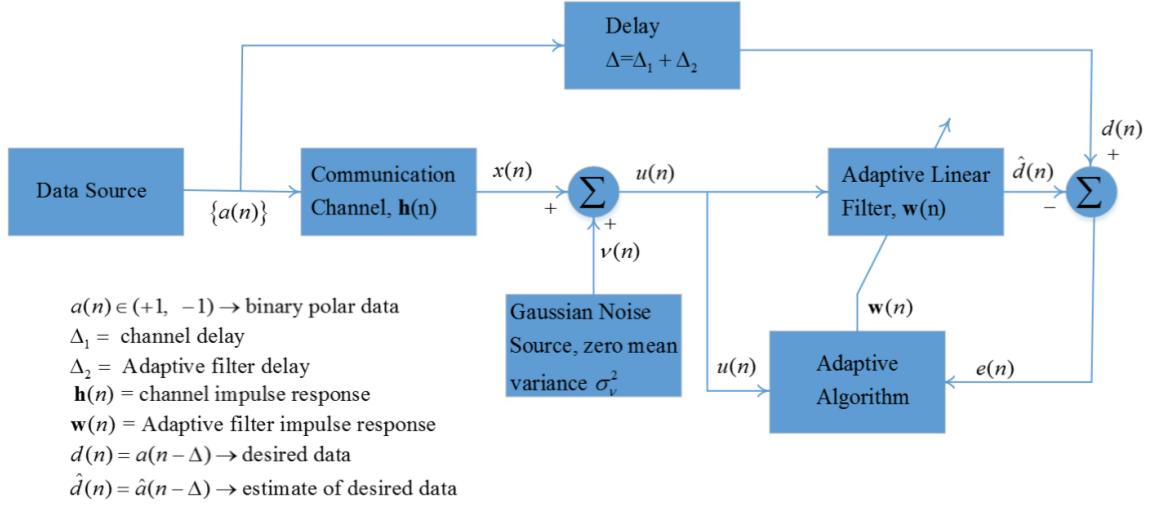


Fig. 1: Block Diagram for Equalization

III. PRE EXPERIMENT CALCULATIONS

From analysis we are given that the input autocorrelation matrix has a quintdiagonal structure of the form given by:

$$R_{input} = \begin{bmatrix} r(0) & r(1) & r(2) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r(1) & r(0) & r(1) & r(2) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r(2) & r(1) & r(0) & r(1) & r(2) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r(2) & r(1) & r(0) & r(1) & r(2) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r(2) & r(1) & r(0) & r(1) & r(2) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r(2) & r(1) & r(0) & r(1) & r(2) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r(2) & r(1) & r(0) & r(1) & r(2) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & r(2) & r(1) & r(0) & r(1) & r(2) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r(2) & r(1) & r(0) & r(1) & r(2) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r(2) & r(1) & r(0) & r(1) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r(2) & r(1) & r(0) \end{bmatrix}$$

$$r(0) = h_1^2 + h_2^2 + h_3^2, \quad r(1) = h_1 h_2 + h_2 h_3$$

$$r(2) = h_1 h_3$$

using the equations given the input autocorrelation matrix and can be evaluated for each channel. The eigenvalues are calculated using the autocorrelation matrix with noise of $\sigma^2 = 0.0001$. The autocorrelation of the noise matrix is simply a diagonal matrix of the variance denoted by, R_v and the overall autocorrelation with noise is defined as:

$$R = R_{input} + R_v$$

The values for comparison are shown in Tables II

Channel	Min Eigen-value, λ_{min}	Max Eigen-value, λ_{max}	Eigenvalue Spread $\chi = \lambda_{max}/\lambda_{min}$
1	0.3330	2.2086	6.0915
2	0.2127	2.3752	11.1663
3	0.1246	2.7256	21.8725
4	0.0647	3.0695	47.4274

TABLE II: Eigenvalues With Noise

As is shown, each channel has progressively worse eigenvalue spread. The effect of this on certain algorithms will be discussed in the following sections

IV. PROJECT 1: LEAST MEAN SQUARES ALGORITHM

The LMS algorithm implements a search method to iteratively update the filter tap weights to find a solution that minimizes the mean square error.

The mean squared error is defined by:

$$J(n) = \sigma_d^2 - 2\mathbf{w}^T \mathbf{p} + \mathbf{w}^T \mathbf{R} \mathbf{w}$$

where σ_d^2 is the constant deterministic variance and the tap-weight vector is

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu \mathbf{u}(n) e(n)$$

The filter output which is the tap-weight vector multiplied by the tap input vector

$$\hat{d}(n) = y(n) = \mathbf{w}^T(n) \mathbf{u}(n) = \mathbf{u}^T(n) \mathbf{w}(n)$$

Intuitively the error of the filter is simply the estimate subtracted from the desired signal.

$$e(n) = d(n) - \hat{d}(n) = d(n) - \mathbf{w}^T \mathbf{u}(n)$$

Which in the context of this project is the randomly generated BPSK signal, $a(n)$. The summary of the implemented LMS algorithm is as follows:

Step 1: $\mathbf{w}(0) = 0$

Step 2:

For $k = 1 : K$

and,

For $n = 1 : N$

(i.e. for each value of the input sequence $\mathbf{u}(n)$, $n = 1, 2, \dots, N$), form the tap-input vector $\mathbf{u}(n)$ and compute the filter output:

$$y(n) = \mathbf{w}^T(n-1) \mathbf{u}(n) \quad (1)$$

Step 3: Compute the error:

$$e(n) = d(n) - \hat{d}(n) \quad (2)$$

Step 4: Update tap-weight vector:

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu \mathbf{u}(n) e(n) \quad (3)$$

End n loop

Step 5: Add squared error for MSE calculation

End k loop

Step 6: Calculate MSE

$$MSE = \frac{1}{K} \sum_{k=1}^K e_k^2(n); \quad n = 1, 2, \dots, N \quad (4)$$

A. Effect of Eigenvalue Spread

The LMS algorithm was run with total $K = 500$ iterations and the error for each value of the tap-input vector $N = 600$ was averaged. This MSE is calculated using Equation 9. This is known as the learning curve and it is used to judge the performance of each experiment. The results for the eigenvalue spread experiment are shown in Figure 2. The experiment was run with $M = 11$, and $SNR = 40dB$.

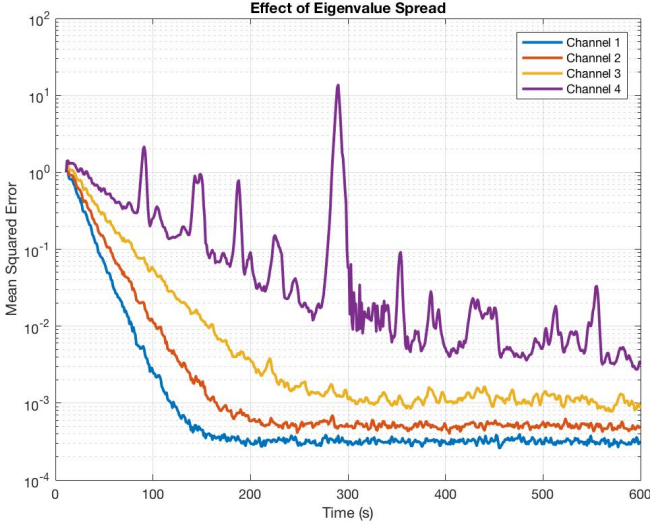


Fig. 2: Learning Curves for Different Channels

These results show that the LMS is very susceptible to eigenvalue spread. Each channel, which has progressively more distortion, has an increased convergence rate and the mean square error settles at a higher value. It is shown that the LMS is not capable of removing the distortion from channel four.

B. Effect of Filter Order

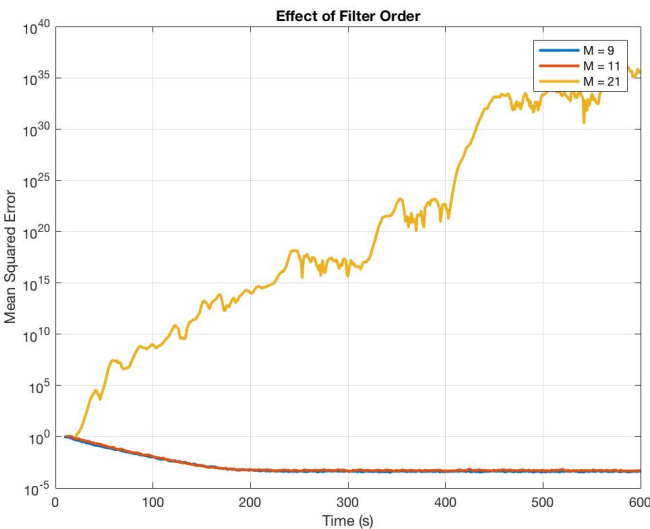


Fig. 3: Channel 2 Learning Curves with Varying Filter Order

Figure 3 reinforces the notion that the misadjustment of the LMS filter increases as the filter order increases [REF]. The parameters for this run were $M = 9, 11, 21$, and $SNR = 40dB$ $\mu =$

0.075 and $N = 600$. While the speed of convergence generally decreases with increased filter order [REF], it typically increases the eigenvalue spread because the input autocorrelation matrix, \mathbf{R} , has larger dimensions. Therefore it becomes difficult to choose a step-size, μ , that will converge for a large filter order.

$$0 < \mu < \frac{2}{\text{tr}[\mathbf{R}]} = \mu < \frac{2}{Mr(0)} = \frac{2}{M(r(0) = h_1^2 + h_2^2 + h_3^2 + \sigma_v^2)}$$

Which for the second channel and a filter order of 21 is $\mu = \frac{2}{2.3752} \approx 0.082$. The value of 0.075 is very close to this limit and we can see the divergence as expected. The problem can be fixed by changing the step size parameter to a more suitable value such as $\mu = 0.0375$. This divergence is demonstrated in Figure 3

C. Analysis of Step Size Parameter

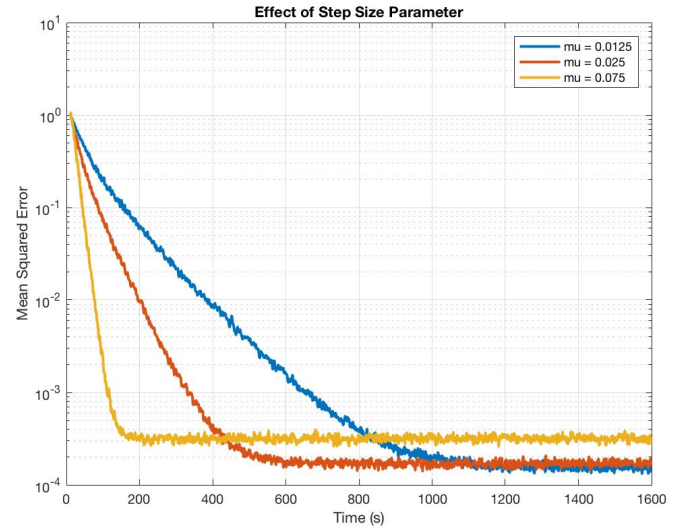


Fig. 4: Learning Curves of Varying Step Size Parameters

As previously discussed, increasing the step-size parameter will decrease the convergence time. This is demonstrated by the convergence speed formula:

$$\tau = \frac{1}{\mu\lambda}$$

However, there are a couple things to keep in mind. First, a step-size which is smaller than the maximum value is required. This must meet the conditions outlined in the previous section.

However we can see that the settling MSE is higher for the faster convergence. Therefore the learning curve clearly demonstrates the important consideration when selecting step-size; an increased step-size provides faster convergence at the cost of increased misadjustment.

D. Comparison of Normalized LMS and Standard LMS

The final experiment studies the performance of the Standard LMS against the Normalized LMS. Two instances of the Standard LMS with values of $\mu = 0.025, 0.075$ are tested alongside the Normalized LMS. The normalized algorithm has the benefit of having a time-varying step size:

$$\mu(n) = \frac{1}{\mathbf{u}^T(n)\mathbf{u}(n)}$$

This solves stability issues with certain applications [REF HAYKIN]. The drawback being that you can no longer optimize for convergence speed. This concept is demonstrated in Figure 5

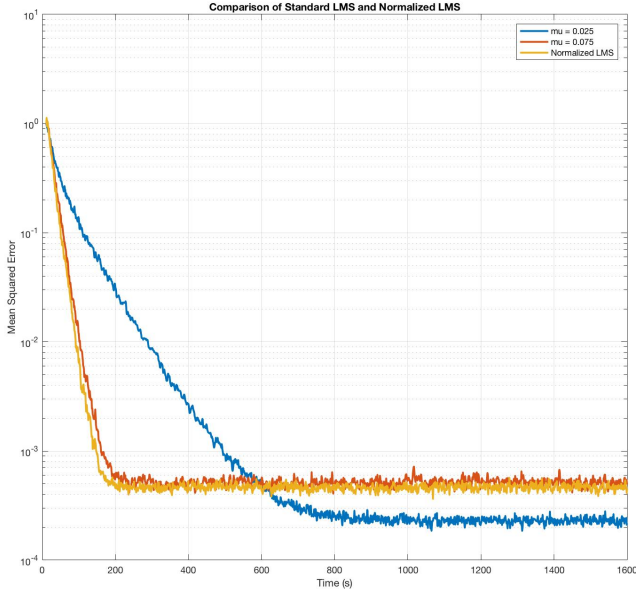


Fig. 5: Learning Curves of Standard and Normalized LMS Algorithms

V. PROJECT 2: RECURSIVE LEAST SQUARES ALGORITHM

The summary of the implemented RLS algorithm is as follows:

Step 1: Initialize the weight vector and the inverse correlation matrix $\mathbf{w}(0) = \mathbf{0}$, $\mathbf{P}(0)\delta^{-1}\mathbf{I}$

Step 2:

For $k = 1 : K$

and,

For $n = 1 : N$

(i.e. for each value of the input sequence $u(n)$, $n = 1, 2, \dots, N$)
compute the Kalman gain vector

$$\mathbf{K}(n) = \frac{\mathbf{P}(n-1)\mathbf{u}(n)}{\lambda + \mathbf{u}^T(n)\mathbf{P}(n-1)\mathbf{u}(n)} \quad (5)$$

Step 3: Compute the a priori error and update the weight vector:

$$\alpha(n) = d(n) - \mathbf{u}(n)\mathbf{w}(n-1) \quad (6)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{K}(n)\alpha(n) \quad (7)$$

End n loop

Step 4: Update the inverse correlation matrix

$$\mathbf{P}(n) = \lambda^{-1}[\mathbf{P}(n-1) - \mathbf{K}(n)\mathbf{u}^T(n)\mathbf{P}(n-1)] \quad (8)$$

Step 5: Add squared error for MSE calculation

End k loop

Step 6: Calculate MSE

$$MSE = \frac{1}{K} \sum_{k=1}^K e_k^2(n); \quad n = 1, 2, \dots, N \quad (9)$$

A. Effect of Eigenvalue Spread

The same experiment that was performed in the LMS project to observe the RLS' ability to converge across increasing eigenvalue spread. As is shown in Figure 6, the RLS algorithm's convergence speed is not effected by eigenvalue spread. All four channels converge within roughly 25 iterations of the N loop. This is in stark contrast to the LMS algorithm which required over 100 iterations to converge even with a large step size. However, the misadjustment is still dependent on the level of dispersion in the channel as the mean square error

settles at a larger value for each consecutive channel. The experiment was run with $M = 11$, $\delta = 0.01$, $SNR = 40dB$, $N = 600$ where, δ is a parameter is for implementation to avoid division by zero errors.

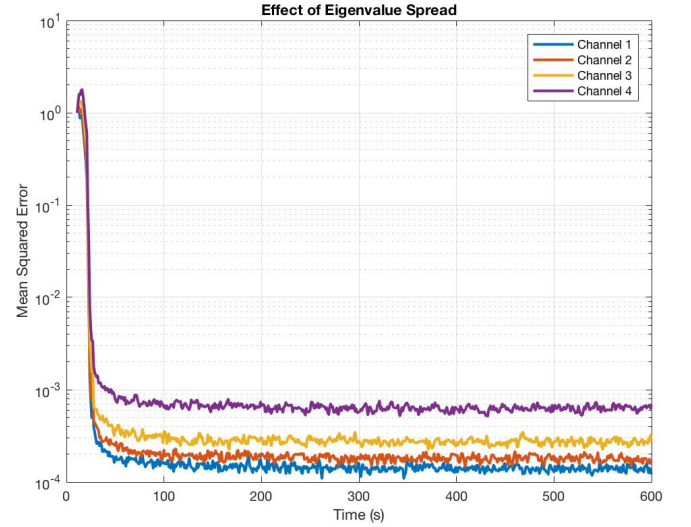


Fig. 6: Effect of Eigenvalue Spread

B. Effect of Filter Order

Now that we've learned that the RLS convergence is robust against eigenvalue spread and that the step size parameter is non-existent, we would expect the algorithm to converge even with a high filter order. For comparison with the LMS we now run the experiment with $M = 9, 11, 21$, $\delta = 0.01$, $SNR = 40dB$, $N = 600$. This is demonstrated in Figure 7. As opposed to the LMS algorithm which diverged for $M = 21$, the RLS is able to track the signal. The plot is offset because a larger delay is required for a larger filter order so it takes time for it to propagate through the system. Figure 7 proves our theory for convergence even with a high filter order.

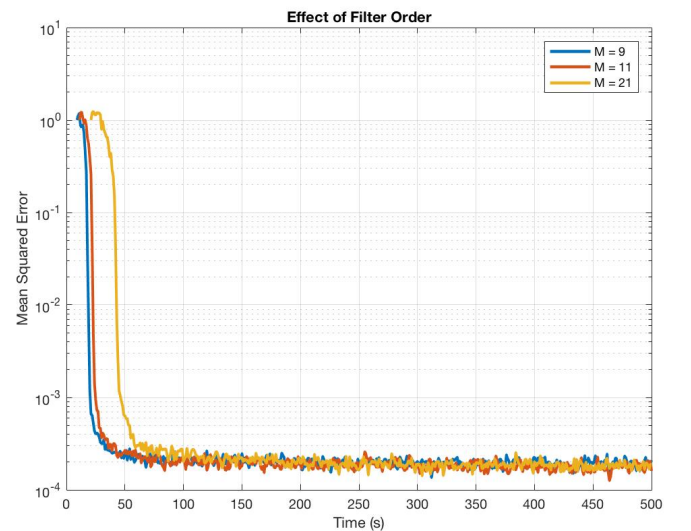


Fig. 7: Effect of Filter Order

C. Tap-Weight Analysis

Next we take a look at the tap-weight vectors for the steady-state averaged RLS algorithm. Figure 8 is a stem plot that shows all eleven

tap values. The symmetry is shown around the center tap, W_6 . This is intuitive when considering that the channel has even symmetry. W_6 is plotted in Figure 9 and it is seen that it reaches a steady-state value within 10 samples. The settled value is approximately 1.15.

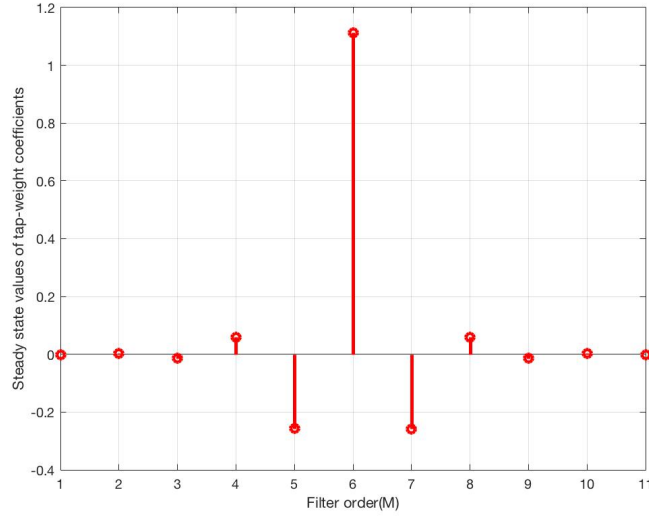


Fig. 8: Stem Plot of Steady-State Tap-Weights

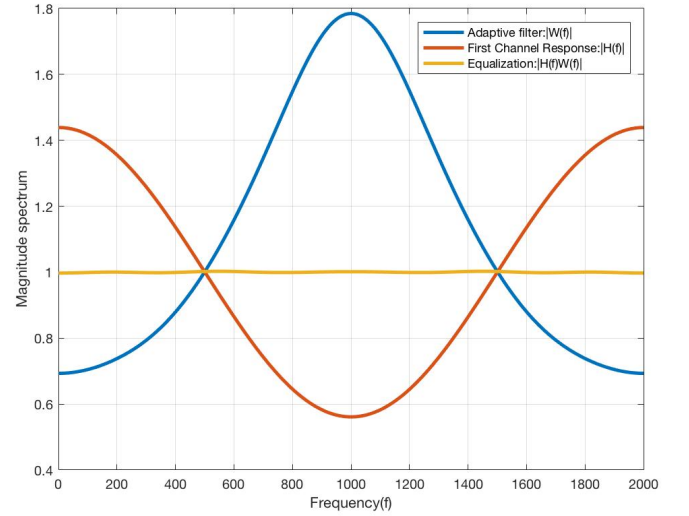


Fig. 10: Magnitdue Spectra of Equalizer

D. Small SNR Analysis

Finally a comparison between the RLS and LMS with small $SNR = 10dB$ is shown in Figure 11. Neither the RLS or LMS performs nearly as well as when the larger signal-to-noise ratio of $40dB$ was used. Their settling values for MSE are roughly 0.1 and 0.8 respectively as opposed to the previous values of roughly 10^{-4} . This experiment gives us a good benchmark for understanding the constraints of the algorithm in a real world implementation.

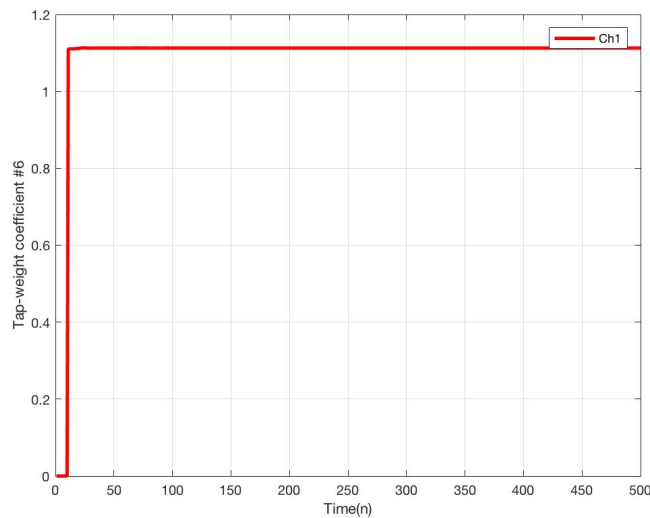


Fig. 9: Center Tap Weight, w_6

The magnitude spectra of Channel 1's response, the adaptive filter, and the resulting equalized output are all plotted in Figure 10. As is expected, the output of the filter is completely equalized and shows a flat response across frequency.

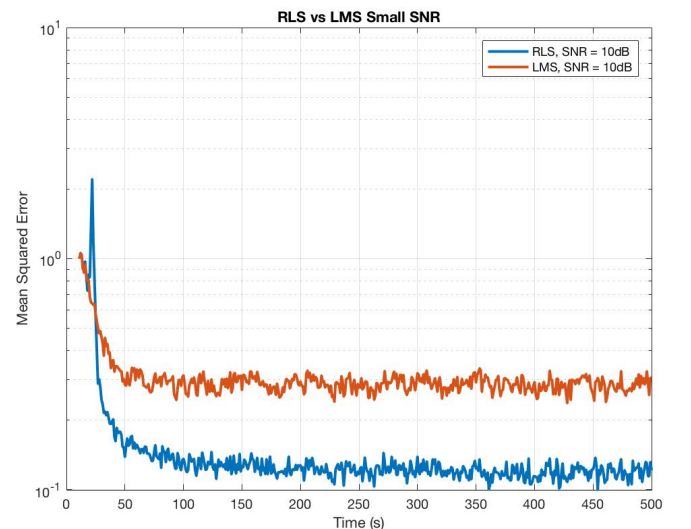


Fig. 11: RLS vd LMS Comparison using $SNR = 10dB$

VI. PROJECT 3: RECURSIVE LEAST SQUARES LATTICE

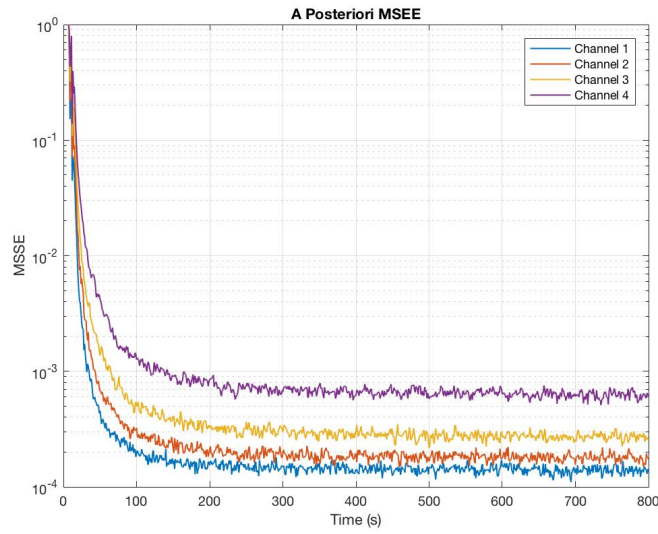


Fig. 12: A Posteriori Mean Square Error

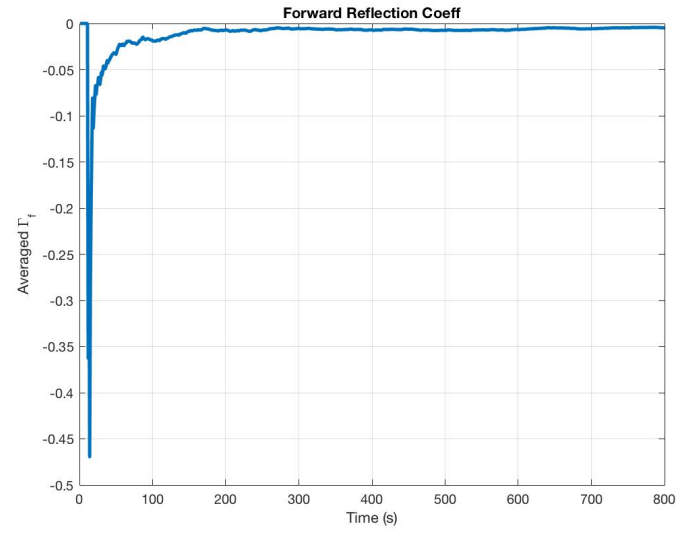


Fig. 14: Forward Reflection Coefficient

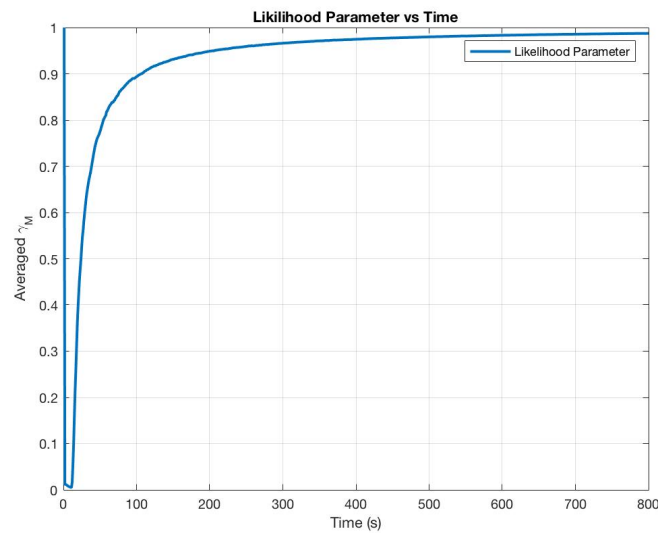
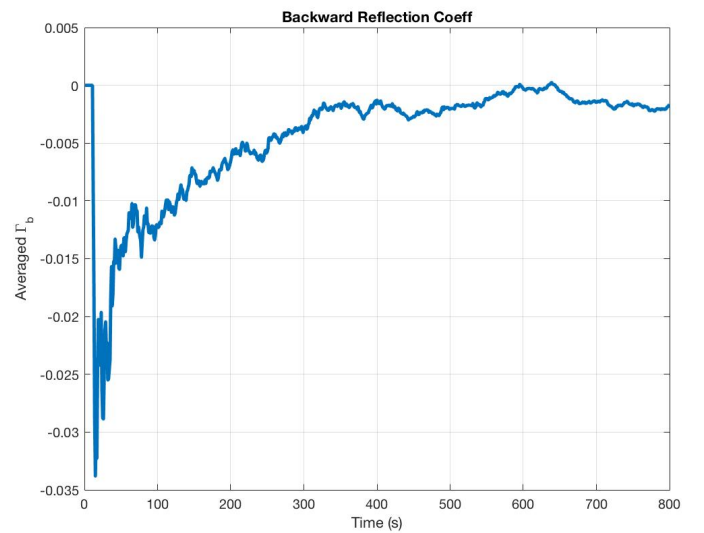
Fig. 13: Likelihood Parameter, γ_m 

Fig. 15: Backward Reflection Coefficient

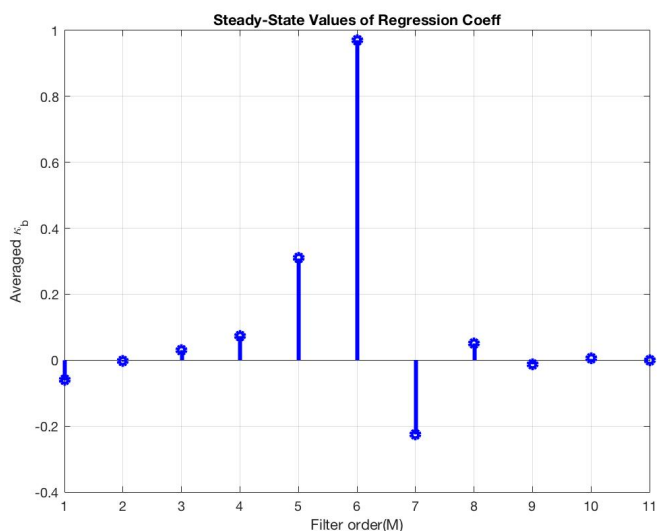


Fig. 16: Regression Coefficients

VII. CONCLUSION

The conclusion goes here.

APPENDIX A

PROOF OF THE FIRST ZONKLAR EQUATION

Appendix one text goes here.

APPENDIX B

Appendix two text goes here.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

- [1] H. Kopka and P. W. Daly, *A Guide to \LaTeX* , 3rd ed. Harlow, England: Addison-Wesley, 1999.



Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.