Adaptive Algorithms for Echo Cancellation

Second formal report for ACSP Laboratory

Ilias Chrysovergis

MSc Communications & Signal Processing, EEE Department

Imperial College London

London, United Kingdom

ilias.chrysovergis17@imperial.ac.uk

***Abstract*—The aim of this lab experiment is to compare the performance of the Least Mean Square (LMS) and Recursive Least Squares (RLS) adaptation algorithms in the context of echo cancellation. Those algorithms are part of the adaptive digital signal processing theory or adaptive filters theory. Firstly, the LMS algorithm and its' modification, the NLMS algorithm, are developed and their performance are discussed for a variety of different parameters. Secondly, the implementation and the performance of the RLS algorithm are presented for the same parameters used in the previous algorithm. Finally, the performance of the LMS, NLMS and RLS algorithms are presented for the case of a varying channel, i.e. when the coefficients of the unknown system are varied with sample number.**

Keywords*— *adaptive digital signal processing, adaptive filters, Least Mean Square (LMS), Normalized Least Mean Square (NLMS), Recursive Least Squares (RLS), Time-Varying Channel

I. INTRODUCTION

Adaptive digital signal processing is the study of algorithms and techniques which have the capacity to vary in sympathy with changing statistical properties, characteristic of many real signals. Such techniques have been successfully applied in many application areas, for example channel equalization in communications beamforming for seismic prospecting, ECG monitoring in medicine, analysis of multiphase flow and the control of dynamic systems. [1]

A recent application of adaptive digital signal processing is the echo cancellation (Fig.1). In "hands-free" telecommunications the mobile unit, which contains the loudspeaker and microphone, is placed for convenience at some distance from the local speaker. Therefore, when the remote speaker is talking, there is acoustic coupling between the loudspeaker and microphone of the mobile unit, which leads to the far-end speaker hearing a disturbing echo of his own voice. Elimination of this echo can be achieved with an adaptive filter, which models the time-varying acoustic coupling.

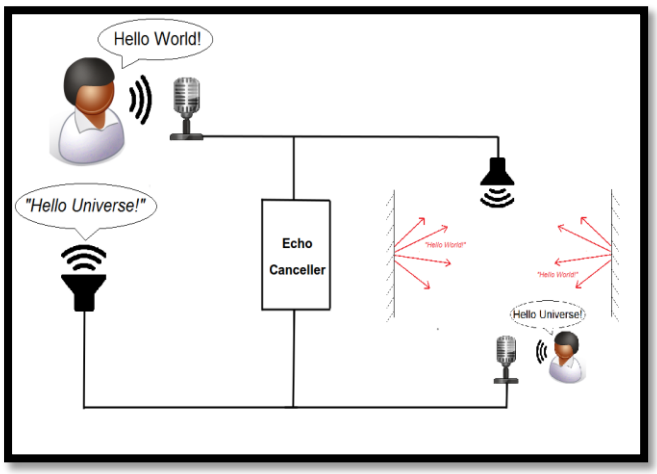

Figure 1: The Echo Cancellation Problem

The rest of the report is organized as follows. Section 2 presents the Least Mean Square and the Normalized Least Mean Square Algorithms with a discussion on their performance. Section 3 presents the Recursive Least Squares Algorithm and a discussion on its' performance is given using the same parameters as in the previous section. Furthermore, the performance of the three algorithms on a time-varying channel is shown. Finally, section 4 concludes the report. [2] [3] [4]

II. Least Mean Square (LMS) Algorithm

A. *LMS Adaptation Algorithm Implementation*

The matlab routine for the LMS Algorithm is given below:

```
function [e, wmat] = LMS(x, d, mu, L)
N = length(x);
wmat = zeros (N, L);
e = zeros(N,1);
for i = L:N
    e(i) = d(i) - wmat(i-1,:)*x(i:-1:(i-L+1));
    wmat(i,:) = wmat(i-1,:) + 2*mu*x(i:-1:(i-L+1))'*e(i);
end
end
```

Figure 2: The Least Mean Square (LMS) Algorithm

B. *Varying the Length of the Adaptive Filter*

The squared error for 3 different lengths of the adaptive filter is depicted in Fig.3. It is easy for someone to realize that as L is increasing, the adaptation gain is decreasing since those two variables are inversely proportional. In this first figure, the plot() function is used and it is obvious that the information for the squared error is not neatly presented.

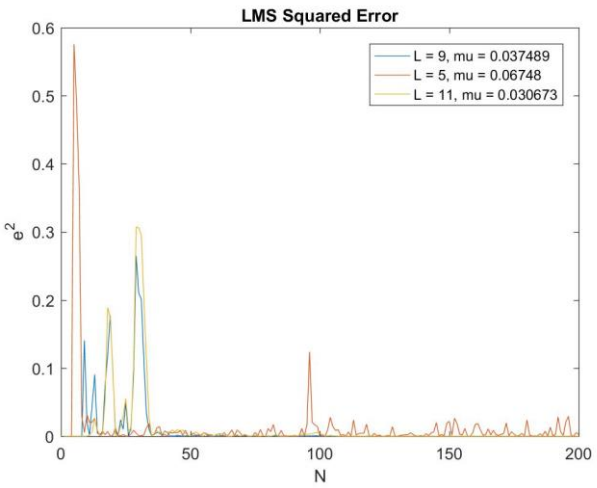

Figure 3: The LMS Squared error in linear scale

For that reason, the semilogy() function, that plots the data with logarithmic scale for y-axis, is used. The diagram below (Fig.4) presents the squared error in logarithmic scale. The difference between the three adaptive filters is obvious as the information is better presented. It can be realized that the best results are achieved when the length of the adaptive filter is equal to the length of the unknown FIR filter. Furthermore, it seems that it is better to overestimate the length of the filter than to underestimate it since the squared error is bigger in the second case. Finally, it is clear that the squared error is descending according to the sample number for L = 9 and L = 11, while the underestimated adaptive filter remains stationary. This descent is the meaning of the adaptive filters and is so important for estimating the desired signals by eliminating the effects of the channel and the noise. In the following figure (Fig.4) the squared error in log-scale vs the sample number is depicted.

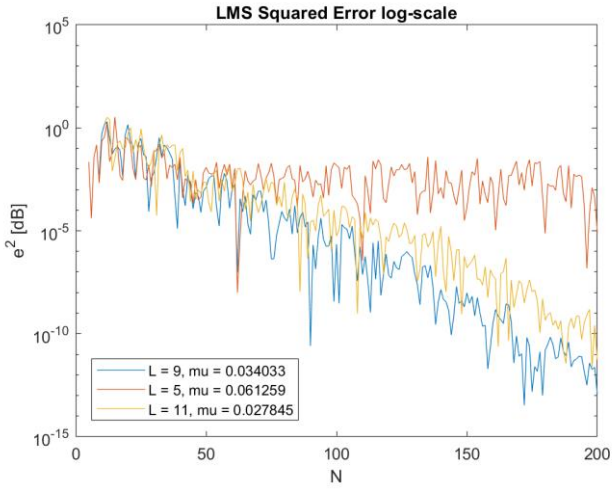

Figure 4: The LMS Squared Error in logarithmic scale

This next figure (Fig.5) presents the normalized weight error vector norm vs the sample number. This metric gives the difference between the coefficient vector of the adaptive filter and the impulse response sequence of the unknown system. Since there are $N = 200$ sample numbers there are also 200 different estimations of the unknown system. This metric is decreasing too, since each estimation of the adaptive filter is better from its' previous one and gives a better approximation of the impulse response of the unknown system. The comments on the performance according to the different lengths is equivalent to the one of the previous figure. The smallest weighted error is achieved when the length of the adaptive filter is equal to the length of the unknown system. Additionally, overestimation of the order of the filter is better than underestimation. Furthermore, the fact that the error for $L = 5$ is not decreasing and remains stationary can be seen here too.

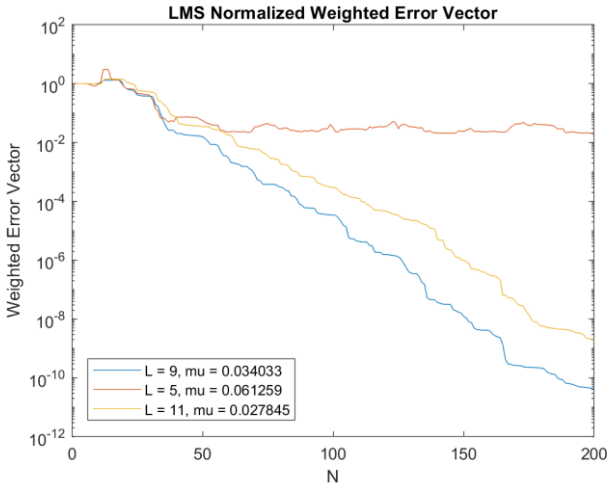

Figure 5: The LMS Normalized Weighted Error Vector

The final weight vector is depicted in Fig.6. It is clear that the LMS algorithm estimates very accurately the unknown system's coefficients.

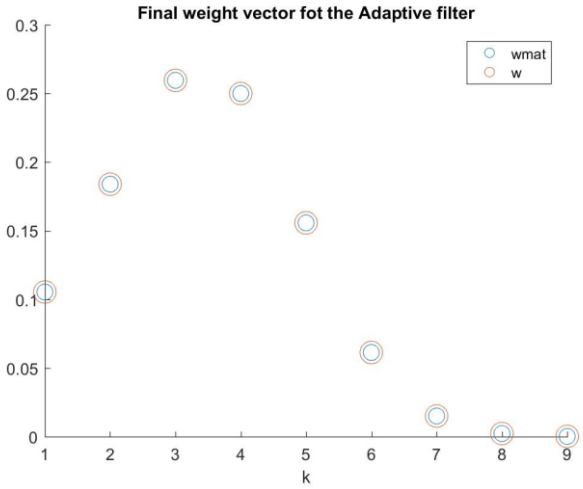

Figure 6: The final weight vector for the adaptive filter

C. Approximation of the Ensemble Average

The average squared error and the average weight error vector norms for 20 independent realizations of the input sequences are presented in the figures below. The realization of 20 independent input sequences corresponds to an approximate ensemble average and gives us a better estimation of how the adaptive filter performs, since it is less depended on the random samples of the input sequence. As happened in the previous diagrams, it is obvious here that the squared error as well as the

weight error vector are decreasing. It can be seen that now -after the averaging- the fluctuations of the lines are smaller, and one may consider that if more independent realizations are created, the two diagrams will become even straighter lines. It is also important to understand that the error is decreasing linearly with the sample number. Therefore, it is clear now that the averaging gives better results than just taking one input sequence and plotting the errors, as happened in the previous question. The two diagrams are depicted below:

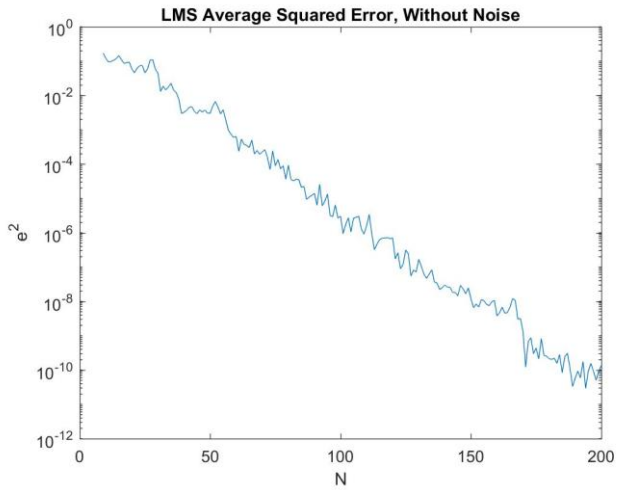

Figure 7: The LMS Average Squared Error, without noise

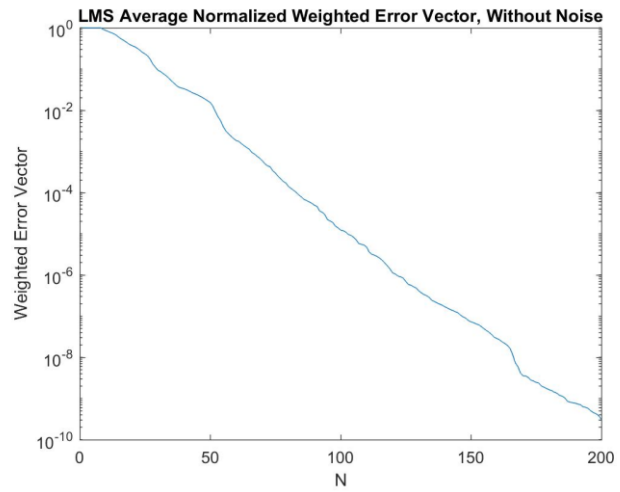

Figure 8: The LMS Average Normalized Weighted Error Vector, without noise

D. Varying the Adaptation Gain

In this question, different values are given for the adaptation gain μ. When $\mu > \frac{1}{L*P_x}$ the adaptive filter does not converge, and the error is always increasing. This fact can be seen in the following two figures, where $\mu = 0.5 > \frac{1}{L*P_x} = 0.11$.

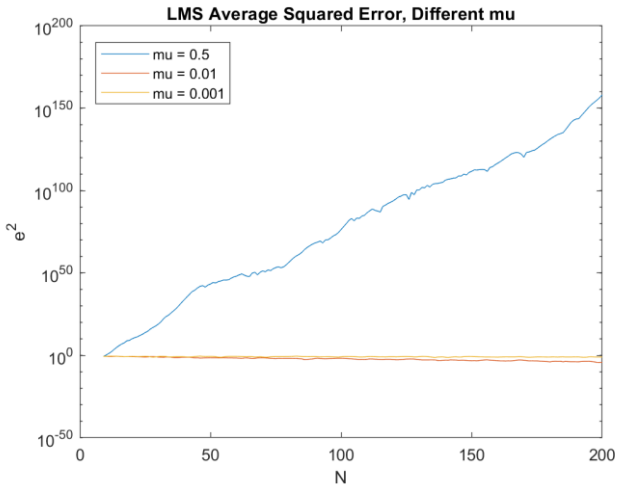

Figure 9: The LMS Average Squared Error for different mu factor, Misadjustment of the LMS algorithm

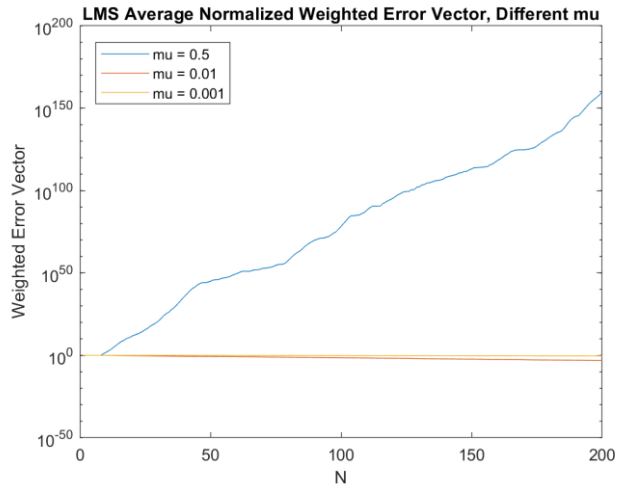

Figure 20: The LMS Average Normalized Weighted Error Vector for different mu factor, Misadjustment of the LMS algorithm

If we choose a different value for the adaptation gain in order to create a stable filter we can have some useful insights with the two diagrams. It can be seen in the two figures below that as we decrease the adaptation gain the error is increasing. Therefore,

the choice of the adaptation gain is a very critical task and the performance of our filter depends on it.

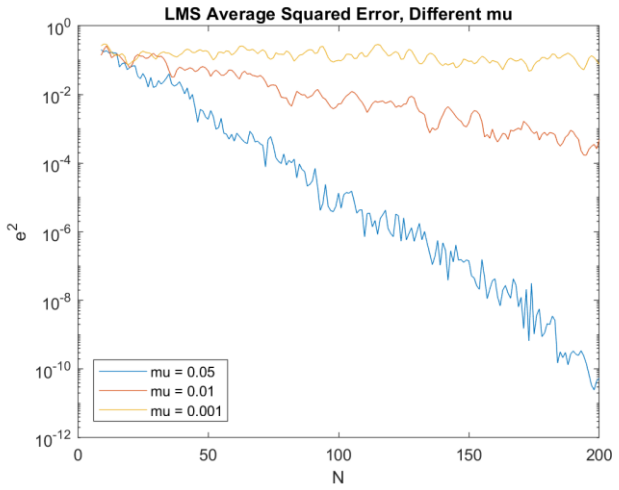

Figure 13: The LMS Average Squared Error for different mu factor, Convergence of the LMS algorithm

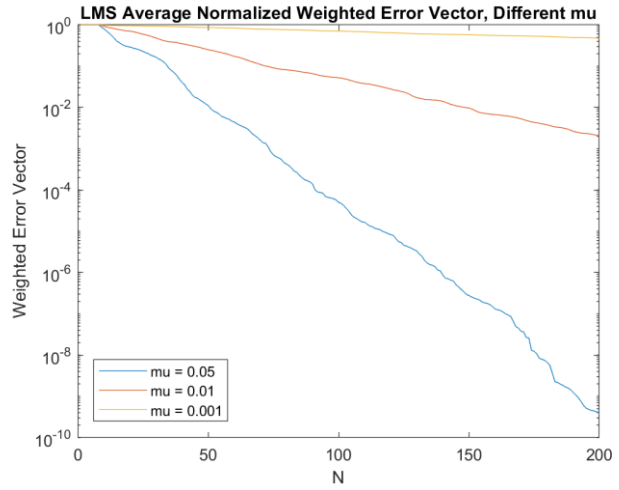

Figure 42: The LMS Average Normalized Weighted Error Vector for different mu factor, Convergence of the LMS algorithm

E. Adding Additive White Gaussian Noise (AWGN)

In this question additive white gaussian noise has been added with an SNR equal to 20. That means that the amplitude of the unknown system is 10 times bigger than that of the independent noise. Furthermore, 10000 independent input sequences have been realized in order to have more smooth graphs. It is clear that the squared error as well as the normalized weighted error vector are converging to their minimums. Due to the presence of noise, after several iterations the algorithm stops decreasing. This happens because the error, after eliminating the distorted input sequence is equal to the noise of the system. Therefore, in that case the noise is the output of the adaptive filter after the echo cancellation and the error cannot be further decreased.

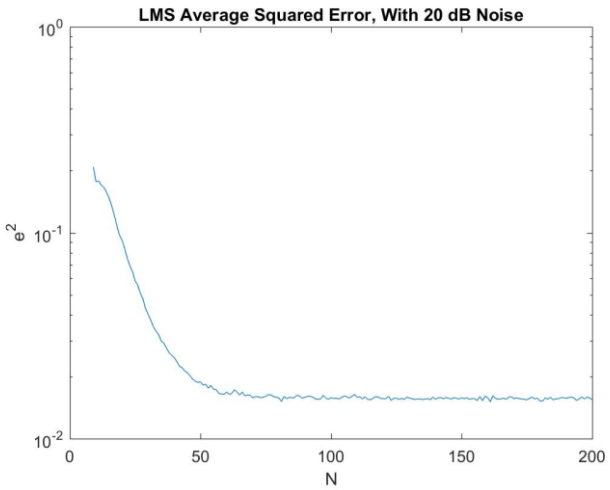

Figure 53: The LMS Average Squared Error with noise

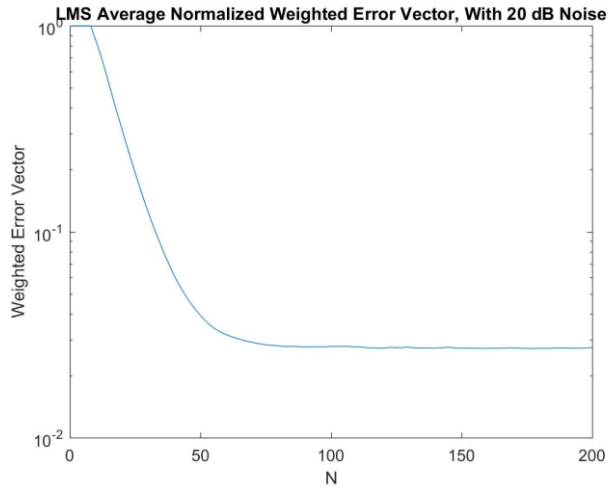

Figure 64: The LMS Average Normalized Weighted Error Vector with noise

F. *NLMS Adaptation Algorithm Implementation*

The matlab routine for the NLMS Algorithm is given below:

```
function [e, wmat] = NLMS(x, d, mu, L )
N = length(x);
wmat = zeros (N, L);
e = zeros(N,1);
for i = L:N
    e(i) = d(i) - wmat(i-1,:)*x(i:-1:(i-L+1));
    dnm = 1 + norm(x(i:-1:(i-L+1)));
    wmat(i,:) = wmat(i-1,:) + 2*mu/dnm*x(i:-1:(i-L+1))'*e(i);
end
end
```

Figure 75: The NLMS Algorithm

By comparing the NLMS algorithm with the LMS algorithm, one can easily realize that the first one requires more operations. Therefore, the computational complexity of NLMS is slightly higher than the one of LMS. However in terms of big O, the complexity is O(L) per iteration.

G. *NLMS Adaptation Algorithm Performance*

In this question a real signal is used as an input to the unknown system. In the following diagrams the squared error and the normalized weight error vector for LMS and NLMS algorithms are presented. For the LMS algorithm three different adaptation gains are used. For the NLMS algorithm the upper bound of the adaptation gain is calculated and then one third of it is chosen as μ to be used in the algorithm. It can be seen that LMS with $\mu = 0.1$ has the same performance with the NLMS algorithm. Furthermore, for LMS the error is increasing while the adaptation gain is decreasing as we have already seen in question iii.

The advantages of the NLMS algorithm is that first of all it is independent of signal power, therefore that algorithm is suitable for real-world changing environments. Furthermore, this fact improves the convergence and the stability of the LMS although the two algorithms converge to the same Wiener solution. Finally, the disadvantage of the NLMS algorithm is that it requires a bigger amount of additions and multiplications than the LMS algorithm, therefore it is more computational expensive. [5]

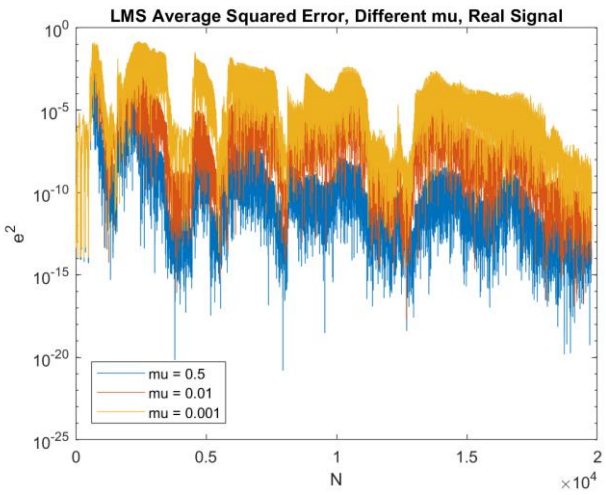

Figure 86: The LMS Average Squared Error for a real signal

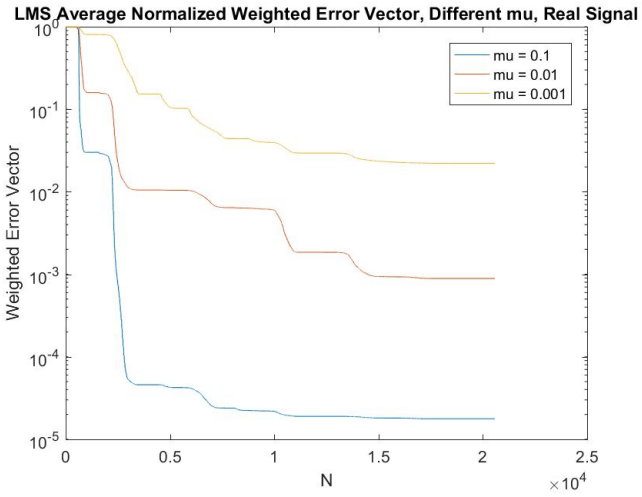

Figure 97: The LMS Average Normalized Error Vector for a real signal

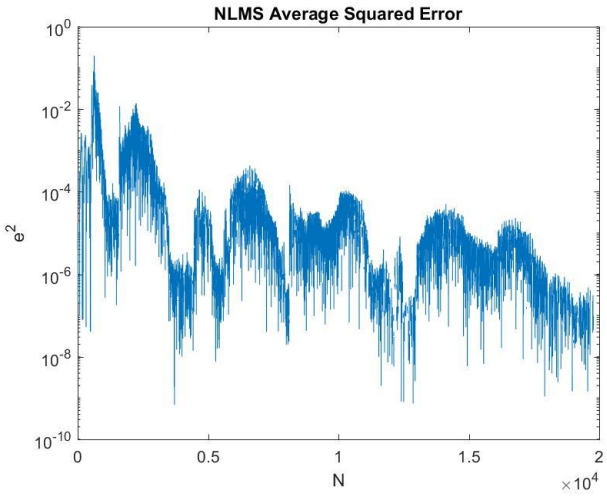

Figure 108: The NLMS Average Squared Error for a real signal

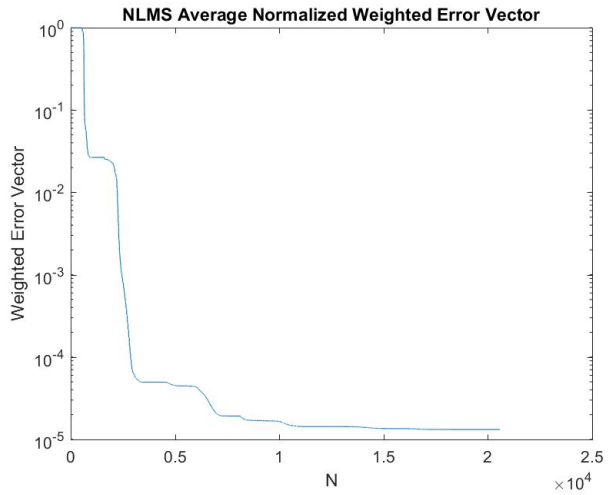

Figure 119: The NLMS Average Normalized Weighted Error Vector for a real signal

III. Recursive Least Squares (RLS) Algorithm

A. RLS Adaptation Algorithm Implementation

The matlab routine for the RLS Algorithm is given below:

```
function [e, wmat] = RLS(x, d, lambda, L)
N = length(x);
wmat = zeros(N,L);
e = zeros(N,1);
z = zeros(L,1);
rinv = eye(L,L);
for count = L:N
    y = x(count:-1:(count-L+1))'*wmat(count-1,:)';
    e(count) = d(count) - y;
    z = rinv*x(count:-1:(count-L+1));
    q = x(count:-1:(count-L+1))'*z;
    v = 1/(lambda + q);
    update = e(count)*v;
    wmat(count,:) = wmat(count-1,:) + update*z';
    rinv = (rinv - z*z'*v)/lambda;

end
end
```

Figure 20: The RLS Algorithm

B. RLS Adaptation Algorithm Explanation

The matlab code implements the RLS algorithm in a tricky and clever way. Firstly, the filter coefficients and the error are initialized into zero, while the inverse of the data input matrix *R* is initialized as the identity matrix. Afterwards, the product of the R inverse matrix and the input sequence are stored into a variable, named z, because it is going to be used in most of the calculations. Then, the transpose of the input vector is multiplied with z in order to produce q, a value that is going to be used for the calculation of v to update the filter coefficient and the R^{-1}. Therefore, z, q, v & e are enough to update the filter coefficients and the inverse of the data input matrix. The complexity of the algorithm, in terms of number of additions and multiplications is $O(L^2)$ per iteration, so totally $O(L^2 * N)$ for all the samples, where L is the length of the adaptive filter and N is the sample number.

C. Varying the Length of the Adaptive Filter for different forgetting factors

In this section the performance of RLS is examined. Firstly, the squared error and the weighted error vector norms are presented, for different lengths of the adaptive filter. It is obvious that the errors for $L = 9$ & $L = 11$ are almost identical. This result shows that the RLS algorithm performs better than the LMS algorithm when we are not aware of the order of the unknown system that we are trying to approximate with the adaptive filter. Furthermore, it is obvious that when the forgetting factor is equal to one the best results are not achieved, since we have found in the previous questions that the LMS algorithm can achieve smaller errors. Secondly, as the forgetting factor is decreasing, the squared error is decreasing too. This is a very positive result when there is no presence of noise in the system. Finally, it can be seen again that when $\lambda = 0.8$ the error is almost zero.

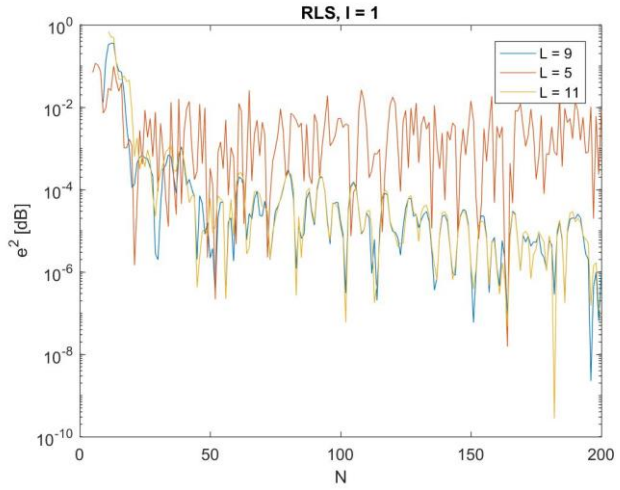

Figure 212: The RLS Squared Error, l = 1

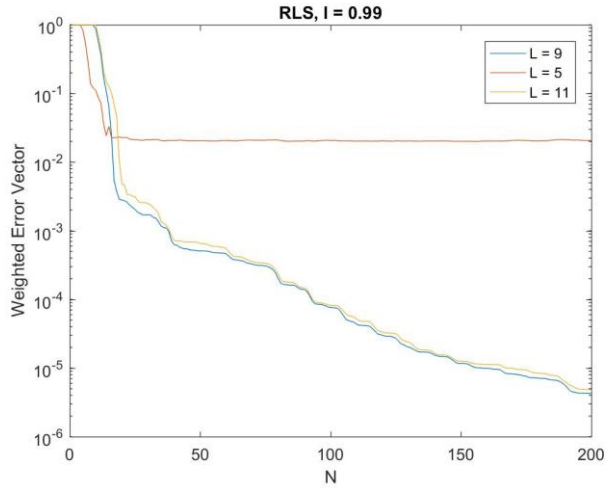

Figure 24: The RLS Weighted Error Vector, l = 0.99

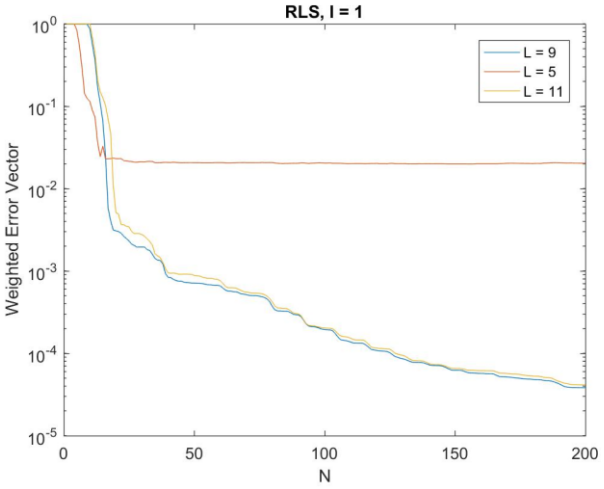

Figure 22: The RLS Weighted Error Vector, l = 1

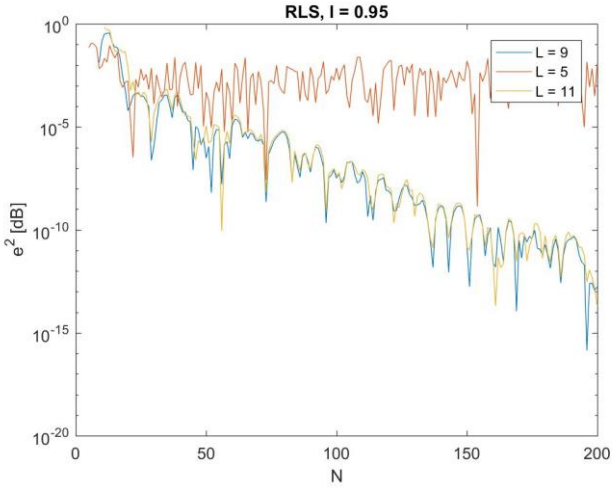

Figure 25: The RLS Squared Error, l = 0.95

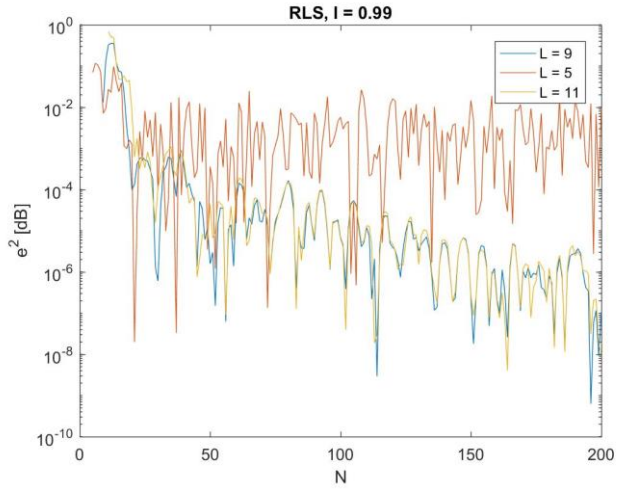

Figure 23: The RLS Squared Error, l = 0.99

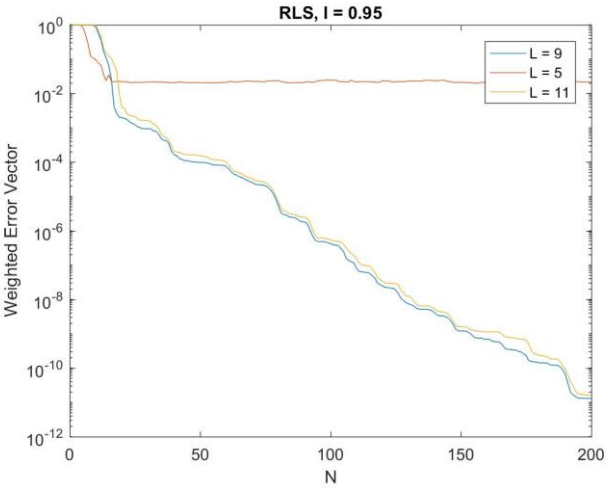

Figure 26: The RLS Weighted Error Vector, l = 0.95

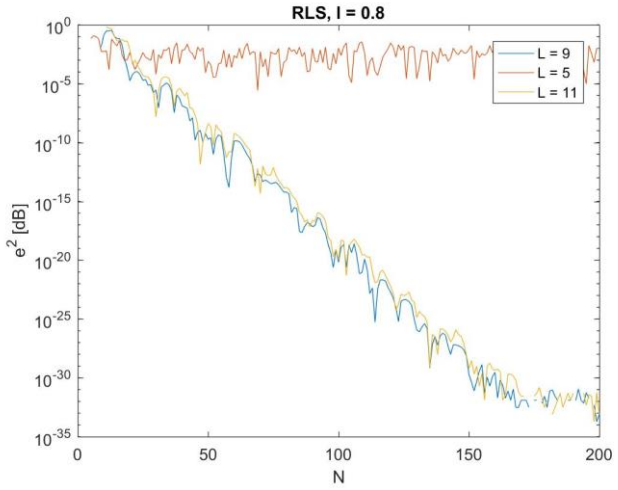

Figure 27: The RLS Squared Error, l = 0.8

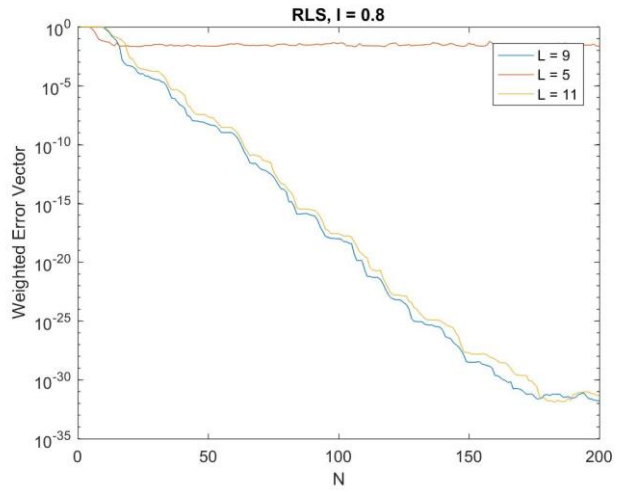

Figure 28: The RLS Weighted Error Vector, l = 0.8

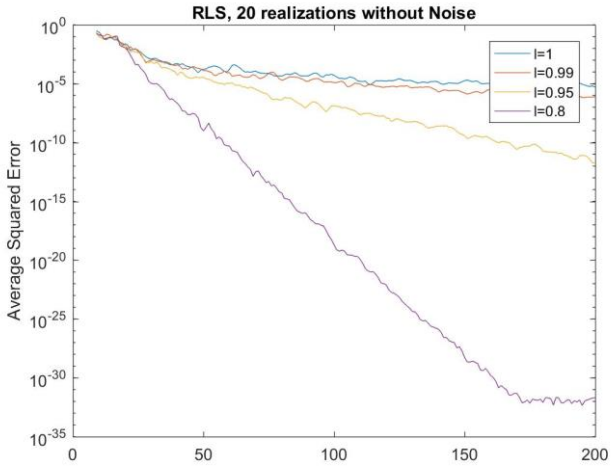

Figure 29: The RLS Average Squared Error, without noise

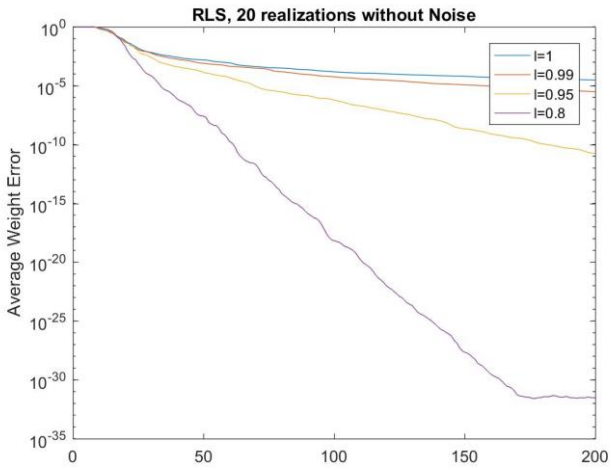

Figure 30: The RLS Average Weight Error, without noise

D. Approximation of the Ensemble Average

In the following diagrams 20 independent realizations of the input sequence have been implemented and the performance of the RLS algorithm is depicted. Two different sets of figures are shown, that correspond to a noiseless and a noisy system. In the second system, when white gaussian noise is added the SNR is equal to twenty. From the first set of figures, where there is no presence of noise, it is clear that while the forgetting factor is decreasing the error is decreasing too. It is obvious that when $\lambda = 0.8$ both the average squared error and the average weight error vector are almost zero. Thus, the adaptive filter is identical to the unknown system.

E. Adding Additive White Gaussian Noise (AWGN)

In the presence of noise, the results are worse than before. From the average squared error diagram, it is difficult to extract information about the different forgetting factors, but it seems that the performance is better than the LMS algorithm. From the average weight error vector diagram, it is clear that increase in the forgetting factor causes decrease in the error. Finally, comparing to the corresponding figures from LMS, the RLS algorithm is better.

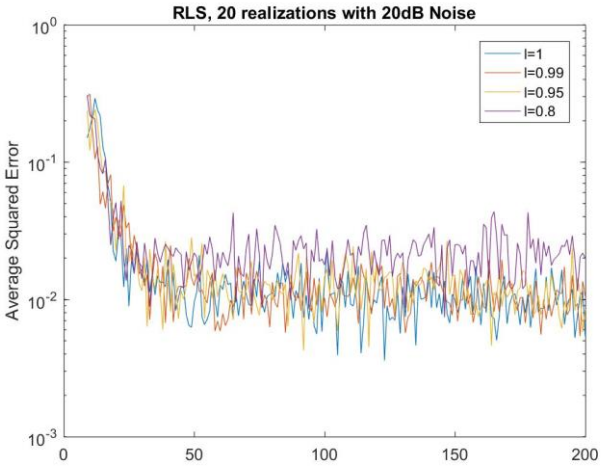

Figure 313: The RLS Average Squared Error, with noise

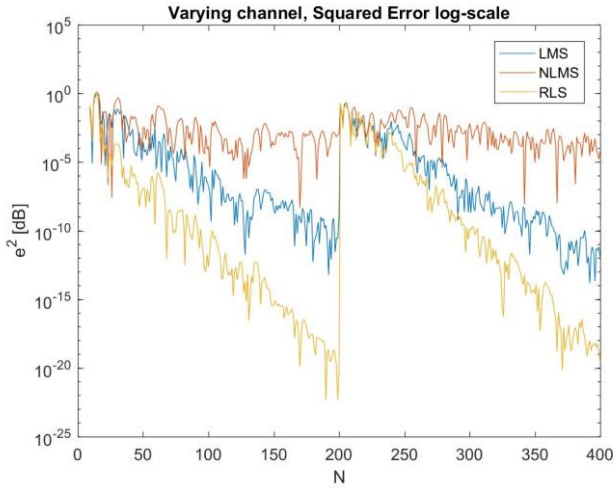

Figure 33: The Squared Error of the three algorithms (LMS, NLMS & RLS) for a signal distorted by a time-varying channel

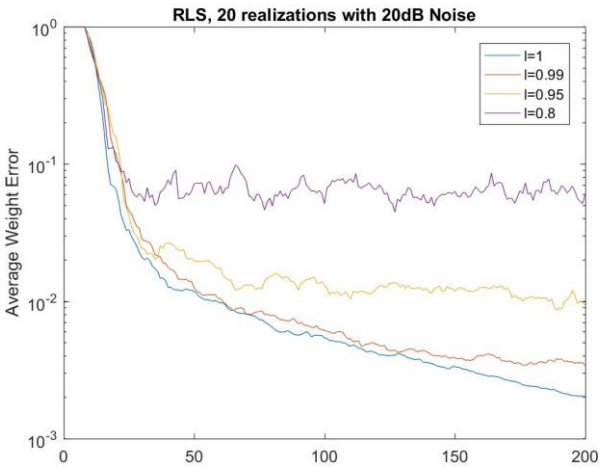

Figure 32: The RLS Average Weight Error, with noise

F. Varying the Coefficients of the Unknown System

If the coefficients of the unknown system are varied with sample number, the adaptive algorithms have a difficulty in estimating the new parameters of it. Thus, the algorithm needs some time to recalculate the new parameters and minimize the error. Below, the performance of the three algorithms for a time-varying channel is depicted.

IV. Conclusion

Concludingly, this report investigates the significance of adaptive digital signal processing algorithms for echo cancellation. Firstly, the LMS algorithm is presented and diagrams as well as comments on its' performance are discussed and made. Then, the NLMS algorithm is shown, with comments on its' difference with the LMS algorithm. Afterwards, the RLS algorithm is introduced and its' performance is compared to the LMS and NLMS algorithms. Finally, the three algorithms are used to recover a signal which is distorted by a time-varying channel. The results show that the RLS algorithm can achieve the lowest error while the LMS algorithm requires the least computation time, since the number of additions and multiplications are less than those of the NLMS and RLS algorithms.

Acknowledgment

The author wishes to thank Prof. Danilo Mandic, who gave him the opportunity to investigate through the adaptive signal processing assignment the two most popular adaptive algorithms for the problem of echo cancellation, as well as Mr. Takashi Nakamura, demonstrator of the experiment, for his valuable help and comments on his work.

V. REFERENCES

[1] D. P. Mandic, Adaptive Algorithms for Echo Cancellation Assignment, London, 2017.

[2] S. Haykin, Adaptive Filter Theory, Prentice-Hall, 1996.

[3] B. Widrow and S. D. Steams, Adaptive Signal Processing, 1985: Prentice Hall.

[4] M. Bellanger, Adaptive Digital Filters and Signal Analysis, Marcel Dekker, 1987.

[5] D. P. Mandic, Adaptive Signal Processing & Machine Intelligence Lecture Notes, London, 2017.