3 Adaptive Signal Processing

3.1 The Least Mean Square (LMS) Algorithm

a. The AR(2) process defined in the coursework has the following general form:

$$x(n) = a_1 x(n-1) + a_2 x(n-2) + \eta(n)(n) \tag{7}$$

where $\eta(n) \sim \mathcal{N}(0, \sigma_\eta^2)$. Since the correlation matrix $\mathbf{R}_x = \mathbb{E}\{\mathbf{x}(n)\mathbf{x}(n)^T\}$, where $\mathbf{x}(n) = [x(n-1), x(n-2)]^T$, the entries of the matrix are:

$$\mathbf{R}_x = \begin{bmatrix} \mathbb{E}\{x(n-1)x(n-1)\} & \mathbb{E}\{x(n-1)x(n-2)\} \\ \mathbb{E}\{x(n-2)x(n-1)\} & \mathbb{E}\{x(n-2)x(n-2)\} \end{bmatrix}$$

In order to determine the entries, we multiply (7) with $x(n-k)$ and obtain:

$$x(n)x(n-l) = a_1 x(n-1)x(n-k) + a_2 x(n-2)x(n-l) + \eta(n)x(n-k)$$

By taking expectations, we get:

$$\gamma(k) = \mathbb{E}\{x(n)x(n-k)\} = a_1 \mathbb{E}\{x(n-1)x(n-k)\} + a_2 \mathbb{E}\{x(n-2)x(n-k)\} + \mathbb{E}\{\eta(n)x(n-k)\}$$

The above equation can be simplified and 3 simultaneous equations can be generated to obtain the entries of the correlation matrix:

$$\begin{aligned} \gamma(0) &= a_1\gamma(1) + a_2\gamma(2) + \sigma_\eta^2 \\ \gamma(1) &= a_1\gamma(0) + a_2\gamma(1) \\ \gamma(2) &= a_1\gamma(1) + a_2\gamma(0) \end{aligned}$$

Solving the simultaneous equations with $a_1 = 0.1$, $a_2 = 0.8$ and $\sigma_\eta^2 = 0.25$, we obtain the ACF matrix:

$$\mathbf{R}_x = \begin{bmatrix} \mathbb{E}\{x(n-1)x(n-1)\} & \mathbb{E}\{x(n-1)x(n-2)\} \\ \mathbb{E}\{x(n-2)x(n-1)\} & \mathbb{E}\{x(n-2)x(n-2)\} \end{bmatrix} = \begin{bmatrix} \gamma(0) & \gamma(1) \\ \gamma(1) & \gamma(0) \end{bmatrix} = \begin{bmatrix} \frac{25}{27} & \frac{25}{54} \\ \frac{25}{54} & \frac{25}{27} \end{bmatrix}$$

To find the range of values for μ for which the LMS algorithm will convergent in mean, the eigenvalues of $\mathbf{R}_x$ has to be found and the following inequality in (8) has to be satisfied.

$$0 \; < \; \mu \; < \; \frac{2}{\lambda_{max}} \tag{8}$$

If the inequality is satisfied, the LMS algorithm will converge to the Winer-Hopf solution. The maximum eigenvalue of $\mathbf{R}_x$ is 1.3889 and thus as long as $0 \; < \; \mu \; < \; 1.44$, the LMS algorithm will converge in mean.

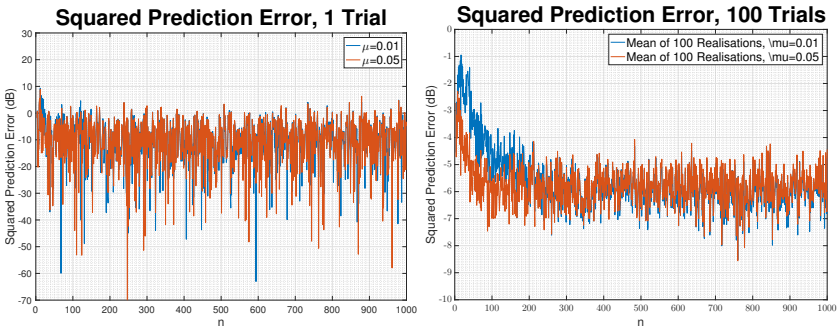

Figure 31: Analysis of Effects of Different Step Size on Convergence Speed of LMS Alogirthm

b. The LMS adaptive predictor using $N = 1000$ samples of $x(n)$ was implemented. Results of 1 realization and the average of 100 realizations are shown in the figure above for $\mu = 0.01$ and $\mu = 0.05$. Looked at the squared

prediction error, it is clear that on average we obtain faster convergence when $\mu = 0.05$. This is exactly as expected as a larger value of μ would allow the algorithm to decent down the error at a faster rate.

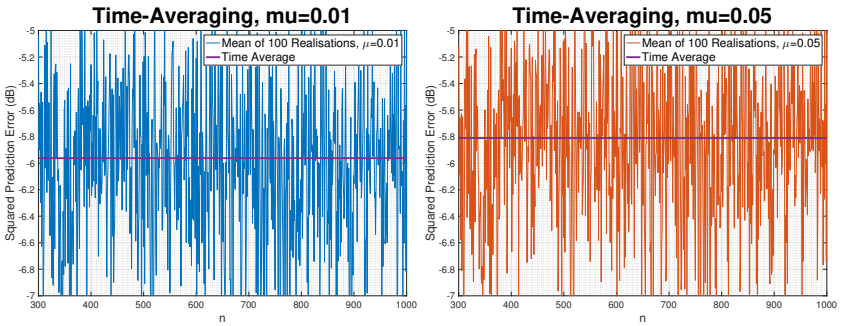

Figure 32: Time-Average of Steady State Error for $\mu = 0.01$ and $\mu = 0.05$

c. The figure above shows the steady-state of the ensemble-average learning curves using 100 independent trials of the experiment. Based on the 100 independent realizations, the excess mean square error, misadjustment and theoretical misadjustment are calculated and the results are shown in the table below. The empirical misadjustment are slightly higher than the theoretical values. Regardless, it is abundantly clear that having a smaller step size introduces a much smaller excess mean square error and results in a smaller misadjustment value. However, as observed above, having a smaller step size causes the rate of convergence to be slow.

mu	EMSE	Empirical Misadjustment	Theoretical Misadjustment
0.01	0.0032	0.0128	0.0093
0.05	0.124	0.0496	0.0463

d. The graphs below show how the coefficients evolve over time. For the rest of the section, most graphs will show the evolution of coefficients, averaged over 100 realizations, and thus it is good to get a feel of what the graphs look like for just 1 realization. Real world adaptive filters will more often than not only work with 1 realization of a random signal. Looking more specifically at the evolution of coefficients for $\mu = 0.05$ for 1 realization, it is clear that speed has been traded off for steady-state error. The amount of variation around the ideal values is significant.

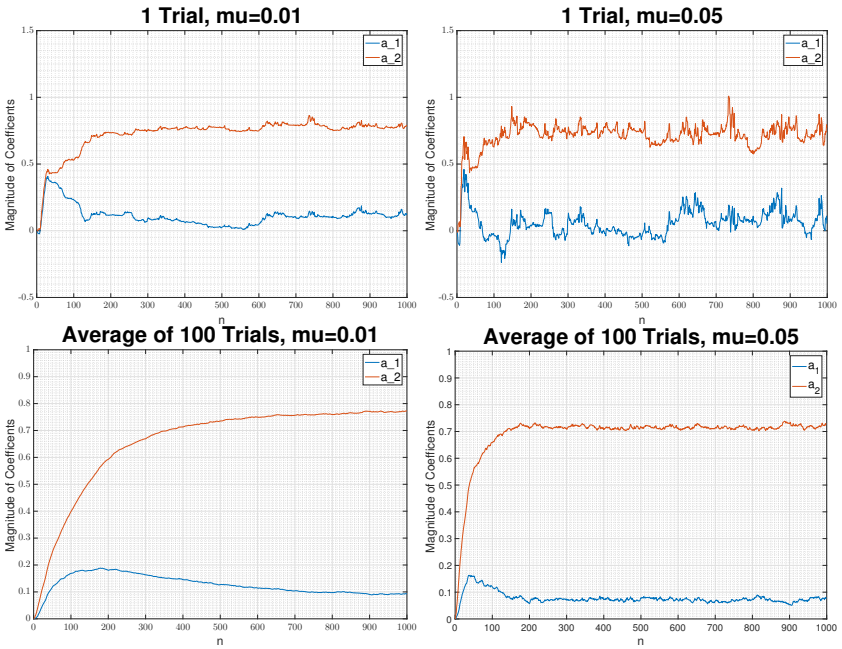

Figure 33: Evolution of Coefficients for 1 Trial and 100 Trial, for $\mu = 0.01$ and $\mu = 0.05$

Next, to answer the question directly, we observe Figure 34. It is clear that for both coefficients, having a small value of μ results in the steady state value that is much closer to the ideal values of $a_1 = 0.1$ and $a_2 = 0.8$. In addition, having a small value of μ results in a much smaller steady-state variance. However, looking at

Figure 33, we find that setting $\mu = 0.01$, we do not approach the steady state value until the 900th iteration. In contrast, using $\mu = 0.05$, the algorithm converges in about 200 iterations.

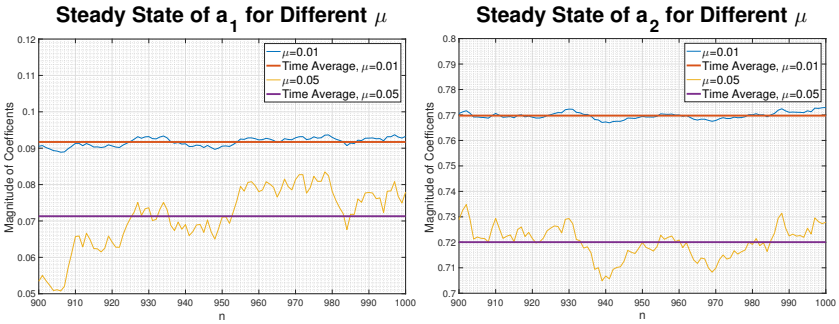

Figure 34: Steady State Values of Coefficients a_1 and a_2, for $\mu = 0.01$ and $\mu = 0.05$

e. In order to minimize the cost function, we differentiate it and set the derivative to 0. Mathematically, we obtain:

$$J(n) = \frac{1}{2}\left(e^2(n) + \gamma||\mathbf{w}(n)||^2\right) \tag{9}$$
$$\frac{\partial J}{\partial \mathbf{w}} = e(n)\frac{\partial e}{\partial \mathbf{w}} + \gamma \mathbf{w}(n) = 0$$

Since,

$$\frac{\partial e}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}\left(\mathbf{x}(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right) = -\mathbf{x}(n)$$

Then,

$$\frac{\partial J}{\partial \mathbf{w}} = -\gamma \mathbf{w}(n) + e(n)$$

Now, using the gradient decent method to reach the optimal value of w, we have to implement (10)

$$\mathbf{w}(n) = \mathbf{w}(n) - \mu \nabla_w J(n) \tag{10}$$

Substituting the derivative of the cost function, we obtain:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\Big(- \gamma \mathbf{w}(n) + e(n)\Big)\mathbf{x}(n)$$
$$\mathbf{w}(n+1) = (1 - \mu\gamma)\mathbf{w}(n) - \mu e(n)\mathbf{x}(n) \tag{11}$$

Thereby showing that the leaky LMS equation shown in the notes and derived in (11) is equivalent to minimizing the cost function shown in (9).

f. The graphs shown in Figure 35 show evolution of coefficients for different values of μ and γ using the leaky LMS algorithm. The steady-state values do not converge to the true values of a_1 and a_2. In fact, the larger the value of γ, the greater the steady-state bias. The Weiner-Hopf solution was briefly mentioned above. It gives the optimal weights calculated using the autocorrelation matrix, $\mathbf{R}$, and the cross correlation vector, $\mathbf{p}$:

$$\mathbf{w}_{opt} = \mathbf{R}^{-1}\mathbf{p}$$

The LMS, under certain bounds on the value of μ, iteratively arrives at the Weiner-Hopf solution. The motivation of the algorithm is to prevent the inverting the autocorrelation matrix. The leaky LMS however does not

converge to the Wiener-Hopf solution. The leaky LMS minimizing a cost function that is slightly different than the cost function that the LMS algorithm minimizes and thus it converges to:

$$\lim_{k\to\infty} \mathbb{E}[\mathbf{w}_k] = (\mathbf{R} + \gamma\mathbf{I})^{-1}\mathbf{p}$$

The additional, $\gamma\mathbf{I}$ term causes a bias that means that the leaky LMS does not actually converge to the Wiener-Hopf solution. In fact, since $\mathbf{p} = [\gamma(1), \gamma(2)]^T$, we can mathematically determine the values to which the algorithm converges:

γ	a_1	a_2
0.1	0.1542	0.3919
0.5	0.1626	0.4992
0.9	0.1319	0.7076

Notice that the graphs confirm the theoretical results calculated above. The leaky LMS displays the same properties with regards to μ. A smaller μ means slower convergence but a smaller steady-state error whereas a larger μ means faster convergence but a larger steady-state error.

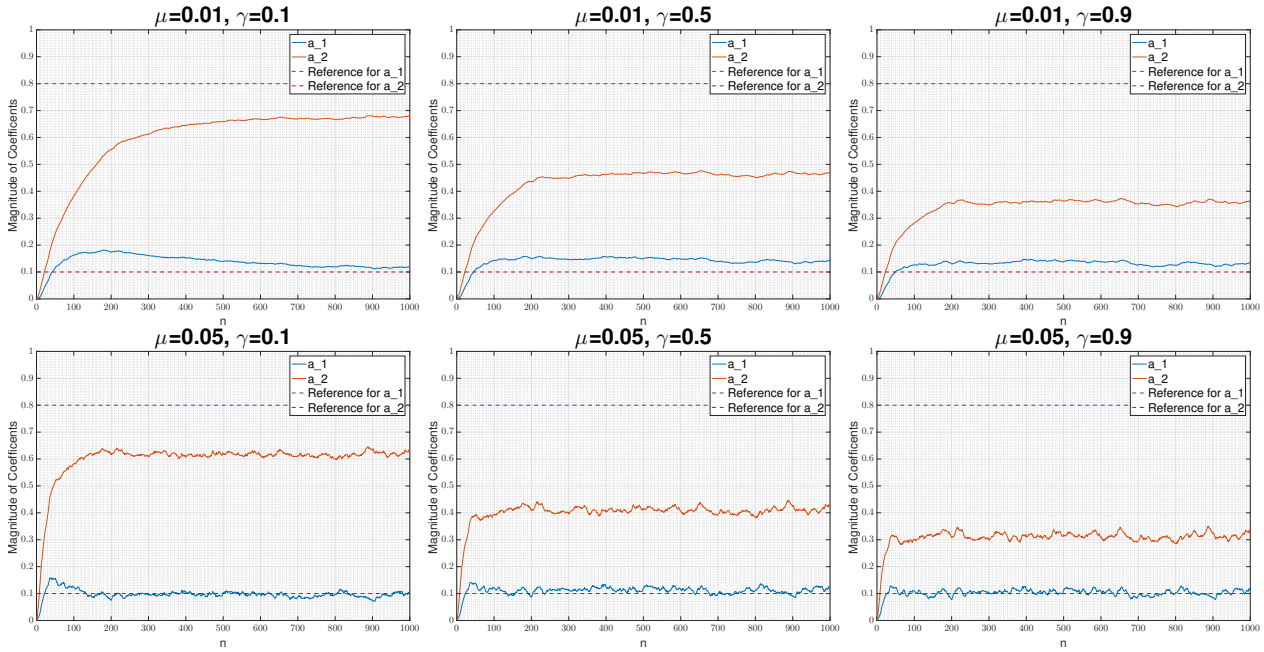

Figure 35: Effect of increasing γ on the Steady State Values of Coefficients a_1 and a_2, for $\mu = 0.01$ and $\mu = 0.05$

3.2 Adaptive Step Sizes

a. Using the LMS algorithm, we have shown the effect that μ has on the convergence time as well as the misadjustment. Depending on the application, we can select a μ to satisfy constraints on convergence time or overshoot or steady-state error. The non-varying nature of μ makes the LMS algorithm computationally cheap. However, keeping a constant μ has certain shortcomings. A major shortcoming is that certain choices μ can cause the algorithm to diverge instead of converging to a solution.

The Gradient Adaptive Step Size (GASS) algorithms allow μ to be time-varying. The algorithms are generally implemented such that the steady-state learning rates approach 0, thereby ensuring stability and convergence. The adaptive nature of the step sizes gives us the flexibilty to vary μ according to the error that is incurred. The three GASS algorithms specified in the coursework all modify the step size according to the following equation:

$$\mu(n+1) = \mu(n) + \rho e(n)\mathbf{x}^T(n)\boldsymbol{\psi}(n)$$

In all algorithms the value of ρ is a constant while ϕ is time-varying. The value of ϕ is altered using the following equations:

$$\text{Benveniste}: \boldsymbol{\psi}(n) = [\mathbf{I} - \mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)]\boldsymbol{\psi}(n-1) + e(n-1)\boldsymbol{x}(n-1)$$
$$\text{Ang \& Farhang}: \boldsymbol{\psi}(n) = \alpha\boldsymbol{\psi}(n-1) + e(n-1)\boldsymbol{x}(n-1), \qquad 0 < \alpha < 1$$
$$\text{Matthew \& Xie}: \boldsymbol{\psi}(n) = e(n-1)\boldsymbol{x}(n-1)$$

Notice that in the Benveniste algorithm, the update equation for $\boldsymbol{\psi}$ is an adaptive low-pass filter with the instantaneous gradient $e(n-1)x(n-1)$ as the input. This algorithm has the ability to deal with noise in the gradient estimate. The Ang & Farhang algorithm modifies the Benveniste algorithm by turning the update equation for $\boldsymbol{\psi}$ from an adaptive to a non-adaptive low-pass filter. The algorithm trades computational complexity for slightly worse convergence properties. The Matthew & Xie algorithm goes a step further and removes the low-pass filtering that the other two algorithms perform on the instantaneous gradient. This makes the algorithm even faster, however once again convergence properties are sacrificed for speed.

The graphs below show the weight error curves for just 1 trial of the algorithms. It is clear that all of the GASS algorithms perform significantly better than the LMS algorithms with a constant step-size.

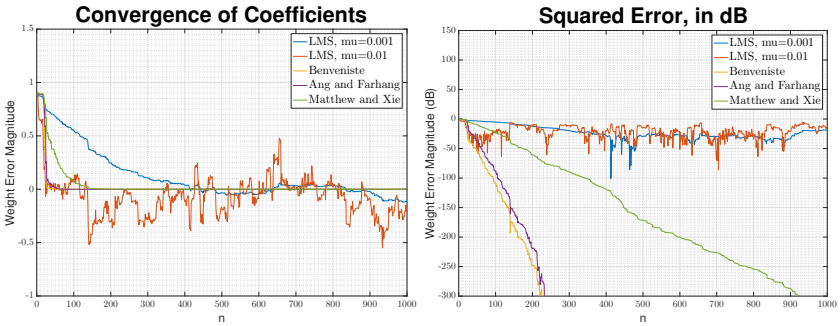

Figure 36: Comparison of Convergence Time using Adaptive Step Sizes and Standard LMS Algorithms

The averaged results of a 100 trials are graphed below. The first thing to notice is that the LMS alogrithms with a fixed step-size do not converge to the correct weights. The results obtained for the GASS algorithms are exactly as expected. The Benveniste algorithm, which is computationally the most expensive, is able to converge the fastest. The next to follow is the Ang & Farhang algorithm. Notice that although the Mattew & Xie algorithm performs the worst amongst the GASS algorithms, it is still significantly better than the LMS algorithms without adaptive step-sizes. Notice also that all GASS algorithms do not have an offset.

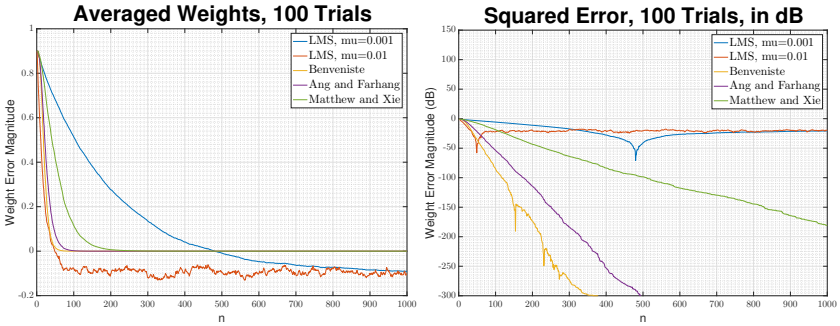

Figure 37: Studying Convergence Time by Averaging Weights over 100 Random Realisations

b. The normalised LMS (NLMS) algorithm has the following update equation:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta}{\epsilon + ||\mathbf{x}||^2} e(n)\mathbf{x}(n) \tag{12}$$

To show that the update equation in (13) based upon the *a posteriori* error $e_p(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n+1)$ is equivalent to the NLMS algorithm, we first multiply both sides by $\mathbf{x}^T(n)$.

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n) \\ \mathbf{x}^T(n)\mathbf{w}(n+1) &= \mathbf{x}^T\mathbf{w}(n) + \mathbf{x}^T\mu e_p(n)\mathbf{x}(n) \end{aligned} \tag{13}$$

Next, we add $\mathbf{d}(n)$ to both sides and rearrange:

$$\begin{aligned}\mathbf{d}(n) + \mathbf{x}^T(n)\mathbf{w}(n+1) &= \mathbf{d}(n) + \mathbf{x}^T\mathbf{w}(n) + \mathbf{x}^T\mu e_p(n)\mathbf{x}(n) \\ \mathbf{d}(n) - \mathbf{x}^T\mathbf{w}(n) &= \mathbf{d}(n) - \mathbf{x}^T(n)\mathbf{w}(n+1) + \mathbf{x}^T\mu e_p(n)\mathbf{x}(n)\end{aligned} \tag{14}$$

Next we substitute the *a posteriori* error into (14):

$$\begin{aligned}e(n) &= e_p(n) + \mathbf{x}^T(n)\mu e_p(n)x(n) \\ e(n) &= e_p(n)\Big(1 + \mu||\mathbf{x}||^2\Big) \\ e_p(n) &= \frac{e(n)}{1 + \mu||\mathbf{x}||^2}\end{aligned} \tag{15}$$

We can now substitute (15) into the original update equation (13) and obtain the NLMS algorithm defined in (12):

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\frac{e(n)}{1 + \mu||\mathbf{x}||^2}\mathbf{x}(n) \\ &= \mathbf{w}(n) + \frac{1}{\frac{1}{\mu} + ||\mathbf{x}||^2}e(n)\mathbf{x}(n)\end{aligned} \tag{16}$$

Thereby proving that the update equation in (13) based the *a posteriori* error with $\epsilon = \frac{1}{\mu}$ and $\beta = 1$ is equivalent to the NLMS algorithm defined in (12). Since ϵ is inversely proportional to the step size μ, the algorithm does not become unstable even if $||x||^2$ is extremely small.

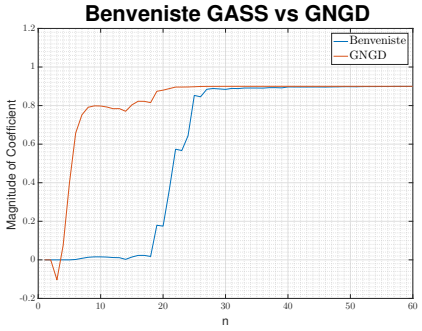

Figure 38: Comparing Convergence Speed of GNGD and Benveniste's GASS Algorithms

c. From the figure above, it is clear that the Generalized Normalized Gradient Descent (GNGD) algorithm converges much faster than the Benveniste algorithm. Both algorithms are similar in the fact that they modify the learning rate in an adaptive manner. The difference lies in the fact that the GNGD algorithm modifies the learning rate in a non-linear manner. The non-linear updates provide compensation for the assumptions made in the derivation of the NLMS. The non-linear updates make the algorithm more robust and the increased stability make it effective at processing non-linear and non-stationary signals (REFERENCE MANDIC PAPER).

In terms of the computational complexity of both algorithms, the GNGD algorithm also beats the Benveniste algorithm. The number of additions and multiplications for the Benveniste algorithm are presented in the Table 2. The variable M represents the model order. It is clear that the algorithm scales quadratically with the model order. The computational complexity of the algorithm can be attributed to the outter product $\mathbf{x}(n-1)\mathbf{x}^T(n-1)$ that has to performed to update the value of $\boldsymbol{\psi}(n)$.

Step	Sub-Step	Multiplications	Additions
$\boldsymbol{\psi}(n)$	$\mathbf{x}(n-1)\mathbf{x}^T(n-1)$	M^2	0
	$\mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)$	M^2	0
	$I-\mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)$	0	M
	$[I-\mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)]\boldsymbol{\psi}(n-1)$	M^2	M^2-M
	$e(n-1)\mathbf{x}(n-1)$	M	0
	$[I-\mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)]\boldsymbol{\psi}(n-1)+e(n-1)\mathbf{x}(n-1)$	0	M
$\mu(n+1)$	$\mathbf{x}^T(n)\boldsymbol{\psi}(n)$	M	$M-1$
	$\rho e(n)$	0	1
	$\rho e(n)\mathbf{x}^T(n)\boldsymbol{\psi}(n)$	0	1
	$\mu(n)+\rho e(n)\mathbf{x}^T(n)\boldsymbol{\psi}(n)$	0	1
$\mathbf{w}(n+1)$	$\mu(n)e(n)$	0	1
	$\mu e(n)\mathbf{x}(n)$	M	0
	$\mathbf{w}(n)+\mu e(n)\mathbf{x}(n)$	0	M
Total		$\mathcal{O}(M^2)$	$\mathcal{O}(M^2)$

Table 2: Computational Complexity of the Benveniste Algorithm

Table 3 shows the computational complexity of the GNGD algorithm. It is clear that the algorithm only scales linearly with the model order.

Step	Sub-Step	Multiplications	Additions
$\epsilon(n+1)$	$\mathbf{x}^T(n)\mathbf{x}(n-1)$	M	$M-1$
	$e(n)e(n-1)$	1	0
	$e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)$	1	0
	$\|\|\mathbf{x}(n-1)\|\|^2$	M	$M-1$
	$(\epsilon(n-1)+\|\|\mathbf{x}(n-1)\|\|^2)^2$	1	1
	$\rho\mu\frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)}{(\epsilon(n-1)+\|\|\mathbf{x}(n-1)\|\|^2)^2}$	3	0
	$\epsilon(n)+\rho\mu\frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)}{(\epsilon(n-1)+\|\|\mathbf{x}(n-1)\|\|^2)^2}$	0	1
$\mathbf{w}(n+1)$	$\|\|\mathbf{x}(n)\|\|^2$	M	$M-1$
	$\frac{\beta}{\epsilon(n)+\|\|\mathbf{x}(n)\|\|^2}$	1	1
	$\frac{\beta}{\epsilon(n)+\|\|\mathbf{x}(n)\|\|^2}e(n)$	1	0
	$\frac{\beta}{\epsilon(n)+\|\|\mathbf{x}(n)\|\|^2}e(n)\mathbf{x}(n)$	M	$M-1$
	$\mathbf{w}(n)+\frac{\beta}{\epsilon(n)+\|\|\mathbf{x}(n)\|\|^2}e(n)\mathbf{x}(n)$	0	M
Total		$\mathcal{O}(M)$	$\mathcal{O}(M)$

Table 3: Computational Complexity of the GNGD Algorithm

3.3 Adaptive Noise Cancellation

a. By choosing an appropriate value of Δ, the Adaptive Line Enhancement scheme can be set up such that the noise, $\eta(n)$, in the signal, $s(n)$, and the predicted input, $u(n)$ are uncorrelated. If the noise $\eta(n)$ and the predicted input $u(n)$ are uncorrelated, the noise can be suppressed and a clean signal can be obtained. To find an optimal value of Δ, we start with the mean-squared error:

$$\mathbb{E}\Big\{\Big(s(n)-\hat{x}(n)\Big)^2\Big\} = \mathbb{E}\Big\{\Big(x(n)+\eta(n)-\hat{x}(n)\Big)^2\Big\}$$
$$= \mathbb{E}\{\eta^2(n)\} + \mathbb{E}\Big\{\Big(x(n)-\hat{x}(n)\Big)^2\Big\} + 2\mathbb{E}\Big\{\eta(n)\Big(x(n)-\hat{x}(n)\Big)\Big\}$$

Expanding the mean-sqaured error shows that there are three terms that are contributing to the error. Ideally, we would like the mean-squared error to be exactly equal to the noise-power, however by changing the value of δ, it is only possible to manipulate $2\mathbb{E}\Big\{\eta(n)\Big(x(n)-\hat{x}(n)\Big)\Big\}$. As such, δ should be chosen so as to minimise $2\mathbb{E}\Big\{\eta(n)\Big(x(n)-\hat{x}(n)\Big)\Big\}$. It is important to note that since $x(n)$ and $\eta(n)$ are uncorrelated, the minimization problem reduces to minimizing $\mathbb{E}\{\eta(n)\hat{x}(n)\}$. Since $\eta(n) = v(n) + 0.5v(n-2)$ we get:

$$\mathbb{E}\{\eta(n)\hat{x}(n)\} = \mathbb{E}\Big\{\Big(v(n)+0.5v(n-2)\Big)\mathbf{w}^T\mathbf{u}(n)\Big\}$$
$$= \mathbb{E}\Big\{\Big(v(n)+0.5v(n-2)\Big)\Big(\sum_{i=0}^{M} w_i s(n-\Delta-i)\Big)\Big\}$$
$$= \mathbb{E}\Big\{\Big(v(n)+0.5v(n-2)\Big)\Big(\sum_{i=0}^{M} w_i\big(x(n-\Delta-i)+\eta(n-\Delta-i)\big)\Big)\Big\}$$

Again using the fact that $x(n)$ and $v(n)$ are uncorrelated:

$$\mathbb{E}\{\eta(n)\hat{x}(n)\} = \mathbb{E}\Big\{\Big(v(n)+0.5v(n-2)\Big)\Big(\sum_{i=0}^{M} w_i\eta(n-\Delta-i)\Big)\Big\}$$
$$= \mathbb{E}\Big\{\Big(v(n)+0.5v(n-2)\Big)\Big(\sum_{i=0}^{M} w_i\big(v(n-\Delta-i)+0.5v(n-\Delta-i-2)\big)\Big)\Big\}$$

Thus, it is clear that for a value of Δ greater than 2, the expectation would reduce to adding the expectations of white noise samples at different time indexes, which are uncorrelated. This is also clear by observing the results presented in Figure 39. Note that the top row of the figure shows the averaged results from 100 realisations, whereas the bottom row shows the overlay of 100 realisations, along with the ideal sinewave. All the figures conclusively prove that for $\Delta > 2$ the results obtained are significantly better. This is also corroborated by the large decrease in the mean-squared error from when Δ is increased from 2 to 3.

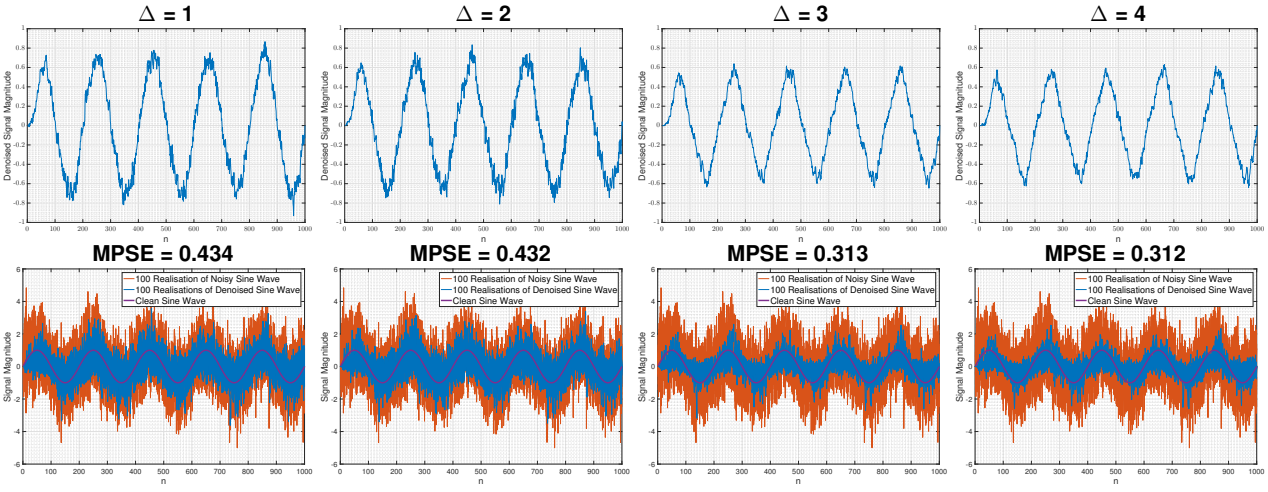

Figure 39: Determining Ideal Value for Δ to be used in the Adaptive Line Enhancement Algorithm

b. The results in Figure 40 futher verify the results above. Choosing a value of Δ greater than 2 causes the mean-squared error to decrease for all model-orders. Note that arbitarily increase the value of Δ is however not

wise. Note that one of the terms which contributing to $\mathbb{E}\left\{\left(s(n)-\hat{x}(n)\right)^2\right\}$ is $\mathbb{E}\left\{\left(x(n)-\hat{x}(n)\right)^2\right\}$. Increasing the value of Δ will increase the delay between $x(n)$ and $\hat{x}(n)$. This effect is also visualized in Figure 40. Notice that the output sine-wave is shifted relatively to the ideal input sinewave when $\Delta = 25$. This shift causes $\mathbb{E}\left\{\left(x(n)-\hat{x}(n)\right)^2\right\}$ to be large, thereby increasing the mean-squared error.

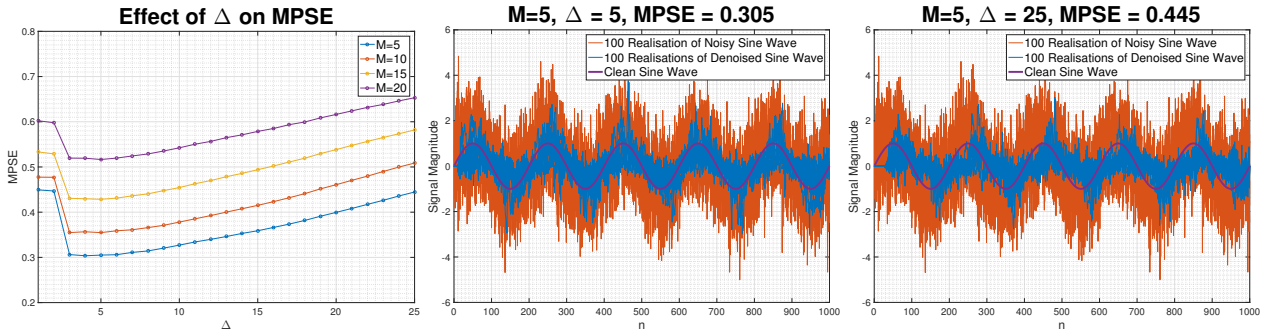

Figure 40: Studying the Effects of Increasing Delay on the MPSE

Theoretically, a higher model-order should be able to describe the highly periodic process with greater accuracy. However, since we are deriving the solution numerically, factors such as convergence and stability should also be taken into account. Note that for high-model orders, if the mu is not correctly chosen, the mean-squared error starts to increase. It is possible to keep μ small and increase the model order however it would cause the algorithm to converge slowly. Looking at Figure 41, a pragmatic choice would be to set $M = 6$ and set $\mu = 0.005$.

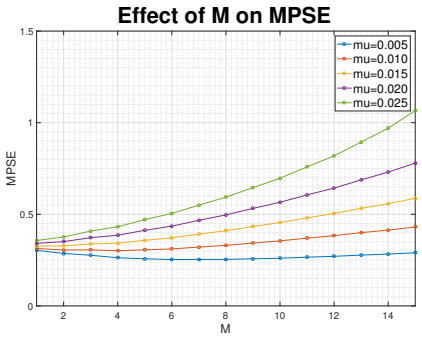

Figure 41: Studying the Effects of Increasing Model Order on the MPSE

c. The performance of the Adaptive Line Enhancement (ALE) algorithm and the Adaptive Noise Cancellation (ANC) algorithm are compared in the figure below. It is clear that the ANC algorithm performs significantly better, once it has converged. However, notice that for small values of n, the ANC algorithm incurs errors larger than the ALE algorithm

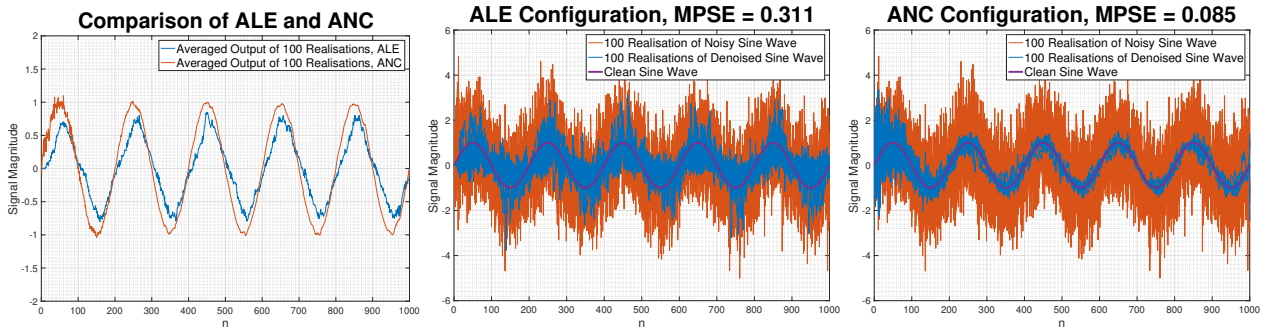

Figure 42: Comparison of ALE and ANC Configurations for Denoising Sinewave

d. The spectrogram below is graphed for reference. It shows the original EEG data with a very strong component at 50 Hz. The length of each segment is 4096, with an 80% overlap between segments. Also, each segment is

windowed with a Hamming window.

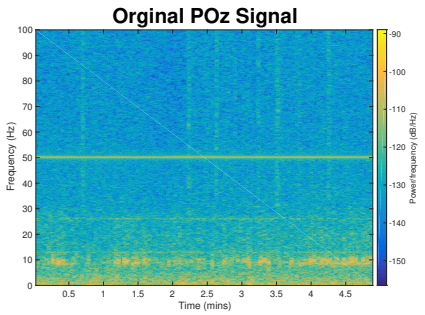

Figure 43: Original EEG Data collected from POz Location on the Scalp with Strong Component at 50 Hz

The first step is to determine the value of M that will remove the strong 50 Hz component. The figure below shows the effect of increasing the model order. Increasing the model order results in a filter that is more efficient at removing the noise. However, notice that certain artifacts have started to occur. With $M = 25$ and $\mu = 0.01$, the filter is not only removing the 50 Hz component but it is also suppressing components between 40 Hz and 60 Hz.

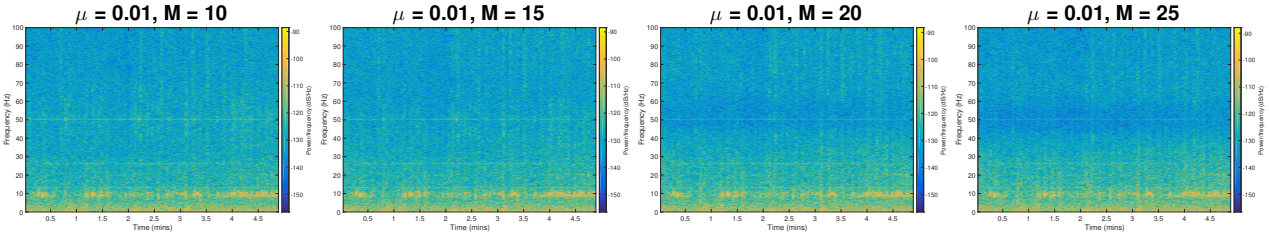

Figure 44: Effect of Increasing Model Order on the Spectrogram of EEG Data

To combat the large spread of frequencies over which the filter is acting, we vary μ. The figure below shows the effects of decreasing μ. Decreasing μ reduces the bias in the ANC algorithm and leads to better performance. It is clear that the filter is now acting over a much smaller range of frequencies.

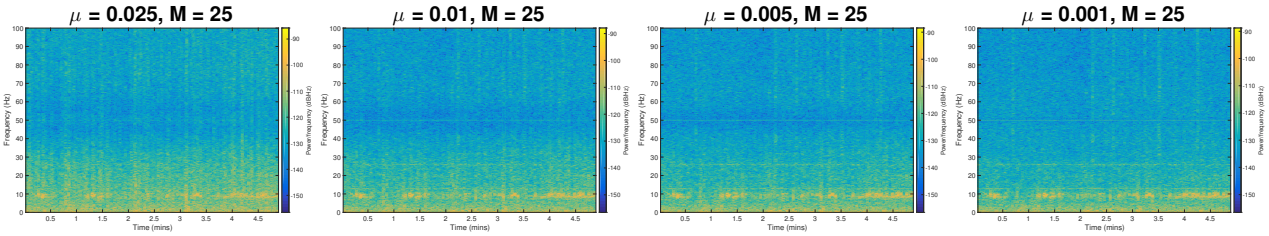

Figure 45: Effect of Varying μ on the artifacts observed in the Spectrogram of Denoised EEG Data

The periodogram and squared error plots corroborate the findings above. For large values of μ such as 0.025, large errors are incurred even at low frequencies. In fact, with $\mu = 0.025$, the filter is suppressing the DC component to a greater extent than the component at 50 Hz. This does not occur with $\mu = 0.001$.

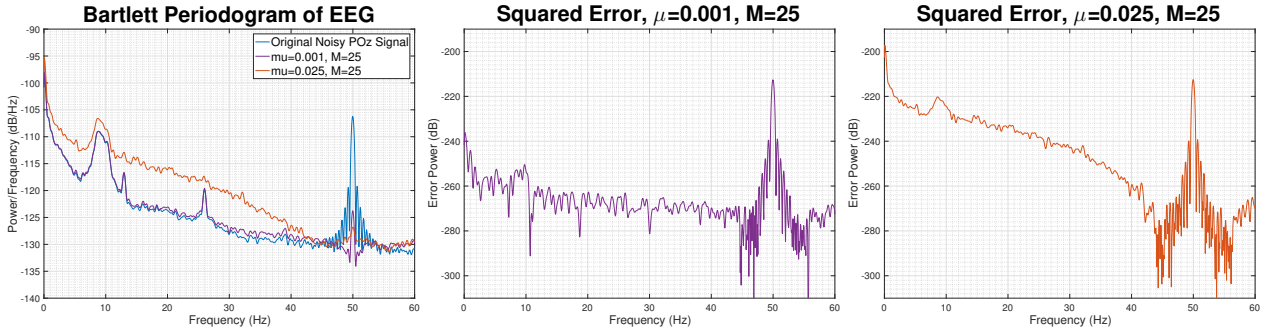

Figure 46: Welch Periodogram, averaged over 2 second intervals, and Squared Errors