ADAPTIVE NOISE SUPPRESSION:

A simulative study of normalized LMS in a non-stationary environment [1]

Arun 'Nayagam

Dept. of Electrical and Computer Engineering

UFID: 6271 – 7860

[1]Submitted in partial fulfillment of the requirements of the course, EEL 6502: Adaptive Signal Processing

Chapter 1

Introduction

In this project we study the effectiveness of the normalized *least-mean-squares* (LMS) algorithm when applied to a noise suppression application. A typical noise suppression application is shown in Fig. 1.1. A speech signal is recorded in the presence of a noise source (a lawn mower in this example). This is the noisy version of the speech, $\mathbf{d} = \mathbf{x} + \mathbf{n}_0$. We also assume that a recording of the noise source alone ($\mathbf{n}_1$) is also available. The purpose of the adaptive filter is to estimate the noise embedded in the speech signal, $\mathbf{n}_0$, using the noise observations, $\mathbf{n}_1$. If the noise observations, $\mathbf{n_1}$ are strongly correlated with $\mathbf{n_0}$, i.e., $\mathbf{n}_0 = f(\mathbf{n_1})$, the adaptive filter can estimate the noise embedded with the desired signal and subtract it from the noisy signal, thereby cleaning the signal. For the purpose of this project we shall assume that an *finite impulse response* (FIR) filter brings out the correlation between $\mathbf{n_0}$ and $\mathbf{n_1}$, i.e., a linear combination of consecutive samples of $\mathbf{n}_1$ gives an estimate of $\mathbf{n_0}$. Thus we have

$$\hat{n}_0(k) = \sum_{i=0}^{M-1} w(k) n_1(k-i), \tag{1.1}$$

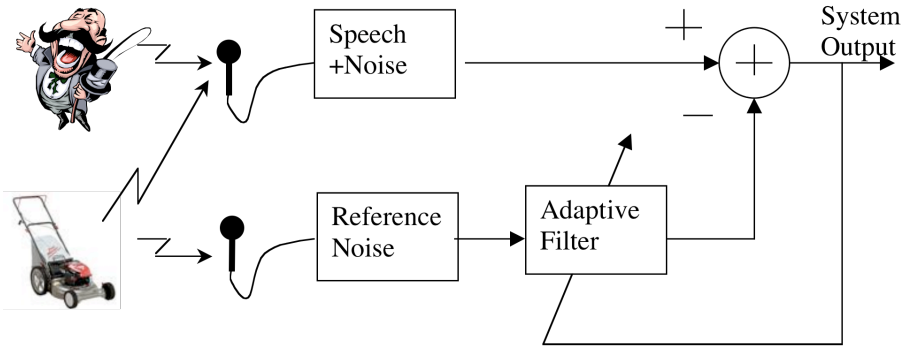

Figure 1.1: An adaptive noise suppression system.

where $\mathbf{w} = \{w_0, w_1, \ldots, w_{M-1}\}$ represents the filter weights and M is the order of the filter. The adaptive filter adjusts the weights of the filter such that $\hat{\mathbf{n}_0}$ is as close to $\mathbf{n}_0$ as possible. The system output is defined as the difference between $d(k)$ and $\hat{n}_0(k)$. In the absence of the speech signal ($x(n) = 0$), this represents the error in the estimation of $n_0(k)$. In the presence of speech, the estimation error is scaled by the speech signal $x(k)$. The adaptive filter changes the weights in every iteration such that the mean square error (MSE), $E[\{d(k) - \hat{n}_0(k)\}^2]$ is minimized. This is an instance of adaptive minimum mean square error (MMSE) estimation. In the absence of the speech signal, the MMSE should converge to zero. In the presence of the speech signal the MMSE should converge to the energy of the speech signal.

The report is organized as follows. In the next chapter, we introduce the optimum solution for MMSE adaptation. We then introduce the steepest descent as a way to adaptively arrive at the optimal set of weights. Then the

LMS algorithm, which is the simplest and most popular of all the steepest descent algorithms is presented. In Chapter 3 we perform a simulative study of the normalized LMS algorithm with a 2-tap FIR filter. In Chapter 4 we study the performance of higher order filters and pick the optimum filter order. The report is concluded in Chapter 5.

Chapter 2

Adaptive Filter Preliminaries

The symbology used in the sequel is first introduced. The weights, input to the adaptive FIR filter and output of the filter are represented by $w(n), u(n)$ and $y(n)$ respectively and they are related as follows

$$y(n) = \sum_{i=0}^{M-1} w(n)u(n-i), \qquad n = 0, 1, \dots \ . \tag{2.1}$$

Since the data in our noise suppression problem is real, all the quantities defined below are real. The purpose of the adaptive filter is produce an estimate of a desired signal $d(n)$. The error, $e(n)$, associated with the estimation of $d(n)$ is defined as

$$e(n) = d(n) - y(n), \qquad n = 0, 1, \dots \ . \tag{2.2}$$

The weights of the filter are optimized by minimizing the mean square error of $e(n)$. Note that $e(n)$ is a random variable since it is a function of the random variable $u(n)$. The cost function (MSE) to be optimized is defined as

$$\xi = E[|e(n)|^2]. \tag{2.3}$$

The optimum set of filter weights produce the minimum value of the cost function, ξ_{min}. We now introduce the optimal solution to the minimization of ξ. Then the LMS algorithm is introduced as an adaptive, computationally viable approach to recursively estimate the optimal solution.

2.1 Optimal Filtering: The Wiener solution

We first represent (2.1) in vector representation as

$$\mathbf{y}(n) = \mathbf{w}^{\mathrm{T}}\mathbf{u}(n). \tag{2.4}$$

Thus, the cost function as defined in (2.3) can be expressed as

$$\xi = E[d^2(k)] - 2\mathbf{P}^{\mathrm{T}}\mathbf{w} + \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w}, \tag{2.5}$$

where $\mathbf{P}$ represents the cross-correlation between the input u and desired response d and $\mathbf{R}$ is the autocorrelation of the input. The statistical expectations are usually computed using time averages as

$$P(i) = E[u(n-i)d(i)] \approx \sum_{m=0}^{N-1} d(m)u(m-i) \tag{2.6}$$

$$R(i-k) = E[u(n-i)u(n-k)] \approx \sum_{m=0}^{N-1} u(m-i)u(m-k). \tag{2.7}$$

We have $\mathbf{P} = [P(0), P(1), \ldots, P(M-1)]^{\mathrm{T}}$ and $\mathbf{R}$ is the cross correlation matrix defined as

$$\mathbf{R} = \begin{pmatrix} R(0) & R(1) & \cdots & R(M-1) \\ R(1) & R(0) & \cdots & R(M-2) \\ \vdots & & \ddots & \\ R(M-1) & R(M-2) & \cdots & R(0) \end{pmatrix}. \tag{2.8}$$

Note that (2.5) helps visualize the error signal in the space defined by the filter weights and thus is referred to as the performance surface. ξ can be minimized by differentiating wrt the weights. This results in the following set of equations called the *Wiener-Hopf* equations

$$P(i) = \sum_{k=0}^{M-1} R(i-k) w_{\text{opt}}(k), \quad i = 0, \ldots, M-1, \tag{2.9}$$

In vector notation we can represent (2.9) as

$$\mathbf{P} = \mathbf{R}\mathbf{w}_{\text{opt}}. \tag{2.10}$$

Thus the optimal set of filter weights can be expressed as

$$\mathbf{w}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{P}. \tag{2.11}$$

The optimal set of weights are referred to as the Wiener solution to the optimal filtering problem. The minimum value of the cost function, ξ_{min}, is obtained by using $\mathbf{w} = \mathbf{w}_{\text{opt}}$ in (2.5) as

$$\xi = E[d^2(k)] - \mathbf{P}^{\text{T}}\mathbf{w}_{\text{opt}}. \tag{2.12}$$

Note that the inversion operation in (2.11) is $O(n^3)$ in complexity and thus is computationally not viable for use in real-time on-the-fly implementations. In the next section, we introduce an adaptive approach to iteratively achieve the minimum of the performance surface.

2.2 Adaptive Filtering: Gradient descent and the LMS algorithm

Gradient descent is a class of unconstrained optimization algorithms that search the performance surface and iteratively converge towards the min-

imum value. The basic idea is that local iterative descent. All gradient descent algorithms have the following mode of operation:

- The algorithms operate on an initial guess or seed value of the weights, $\mathbf{w}_0$.
- Then a sequence of weight vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots$ is generated such that the cost function $\xi(\mathbf{w})$ is reduced at each iteration, i.e., $\xi(\mathbf{w}(n)) < \xi(\mathbf{w}(n-1))$.
- The successive refinements applied to $\mathbf{w}$ at each iteration are in the direction opposite to the gradient of the cost function $\xi(\mathbf{w})$, i.e.,

$$\mathbf{w}(n) = \mathbf{w}(n-1) - \frac{1}{2}\mu\nabla\xi(\mathbf{w}), \tag{2.13}$$

  where $\nabla\xi(\mathbf{w}) = \frac{\partial\xi(\mathbf{w}),}{\partial\mathbf{w}}$ and μ is called the it step size that parameterizes the gradient search.

Since the gradient of the error points towards the direction of increasing error, any movement in the opposite direction will tend to decrease the error (provided the step size is chosen properly). This is the philosophy of the gradient search procedures. Note that the step size controls the speed of adaptation. A small step size would require many iterations to converge to the optimum solution and a small MSE. A large step size would cause faster convergence but suffers from the problem of overshooting the minimum and divergence. Also, the steady state MSE is higher with a higher step size.

The above procedure is called local iterative descent because it only used local information (local to the current iteration) to compute the new values of the weights. The procedure assumes that an estimate of the gradient is available. Typically the gradient is estimated using the perturbation method.

For simplicity let $\mathbf{w}_n = \mathbf{w}(n-1)$, then $\nabla\xi(\mathbf{w}_n) = \frac{\xi(\mathbf{w}_n+\triangle\mathbf{w})-\xi(\mathbf{w}_n-\triangle\mathbf{w})}{2\triangle\mathbf{w}}$. Though the gradient descent is simple in operation, the estimation of the gradient requires two samples of the performance surface in the vicinity of the current weights and one division. This limits the speed of the algorithm and hence faster and simpler methods of estimating the gradient are required for real-time applications.

The least-mean-squares (LMS) algorithm is the simplest and most widely used of all the gradient descent algorithms. Its popularity lies in its simplicity. The LMS uses a very noisy estimate of the performance surface. The LMS approximates the performance surface by the power of the current error,

$$\xi(\mathbf{w}(n)) = E[|e(n)|^2] \approx e(n)^2 = (d(n) - y(n))^2. \tag{2.14}$$

This makes the computation of the gradient very simple.

$$\hat{\nabla}\xi(\mathbf{w}(n)) = -2e(n)\mathbf{u}(n), \tag{2.15}$$

where $\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]$. Thus, the weight update equation in (2.13) reduces to

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(d(n) - y(n))\mathbf{u}(n), \tag{2.16}$$

Thus, in LMS adaptation, the weight update only requires a vector multiplication and thus the complexity is just $O(M+1)$. Convergence analysis has shown that the step size is constrained as

$$\mu \leq \frac{1}{E[\mathbf{u}^{\mathrm{T}}\mathbf{u}]}, \tag{2.17}$$

where $E[\mathbf{u}^{\mathrm{T}}\mathbf{u}]$ represents the power of the input signal. Thus, if the signal is not stationary, then the signal power might change causing the initial

step size to exceed the allowed value. This might cause the LMS algorithm to diverge. To overcome this obstacle, a variant of the LMS called the *normalized* LMS was proposed. In the normalized LMS, the step size at each iteration is scaled by the signal power thereby forcing the step size to be within the allowed limit. Thus, the weight update equation becomes

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\alpha}{E[\mathbf{u}^{\mathrm{T}}\mathbf{u}]}(d(n) - y(n))\mathbf{u}(n), \quad 0 < \alpha < 1. \tag{2.18}$$

The normalized LMS can also be derived as an application of the *principle of minimal disturbance*. The basic philosophy is the same as in LMS, but the optimization (minimization) is performed such that the change in weights from one iteration to another is as small as possible. Thus the change in the sequence of weights is made as smooth as possible. The minimization results in the normalized LMS algorithm. The weight update equation in (2.18) keeps the change in weights as small as possible.

Though the estimation of the gradient is very noisy, the algorithm performs extremely well (as demonstrated in homeworks 1 and 2). The performance of the normalized LMS algorithm in the noise cancellation problem is studied in the next chapter.

Chapter 3

Performance evaluation in a non-stationary environment

We first demonstrate the performance of normalized LMS adaptation using a simple two-tap filter. Though this filter does not get perform very well, it serves to demonstrate the principles involved. We define performance measures and study both the time and frequency domain performance of the filter. In the next Chapter, we study the performance of higher order filters.

3.1 Problems with Non-Stationarity

If the input to an LMS adaptive filter is stationary, then the only function of the filter is to eventually converge to the optimum set of filter weights. In the noise suppression application, the desired response of the filter is not stationary (has speech embedded in it!). This causes the performance surface to change location in the weight space. To demonstrate this behavior, two

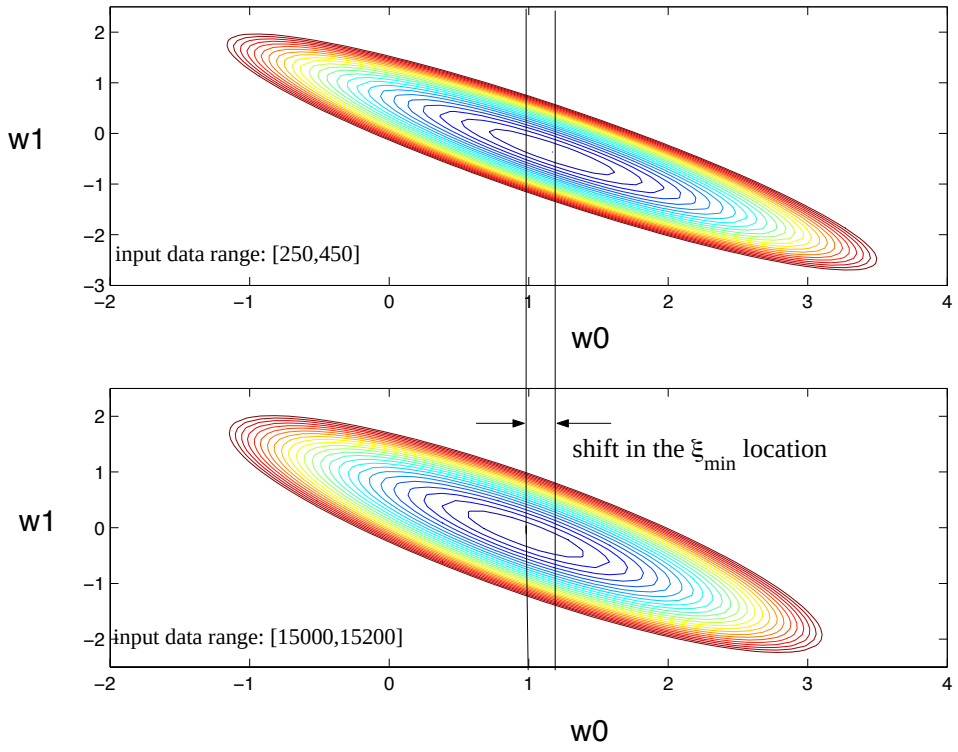

Figure 3.1: Performance surface contours for two non-overlapping segments of data. A window of length 200 is used to compute **R** and **P**.

different segments of data are taken, one from sample 250–450 and the other from sample 15000–15200. The autocorrelation of the filter input, **R**, and the cross-correlation between the filter input and the desired response **P** are computed in the chosen windows. The optimum filters weights are computed using (2.11) and the minimum MSE (minimum value of the performance surface) is then computed using (2.12). The weights are varied in the vicinity of the optimum value and the contours of the performance surface are plotted in Figure 3.1. It can be seen that the performance surface achieves its

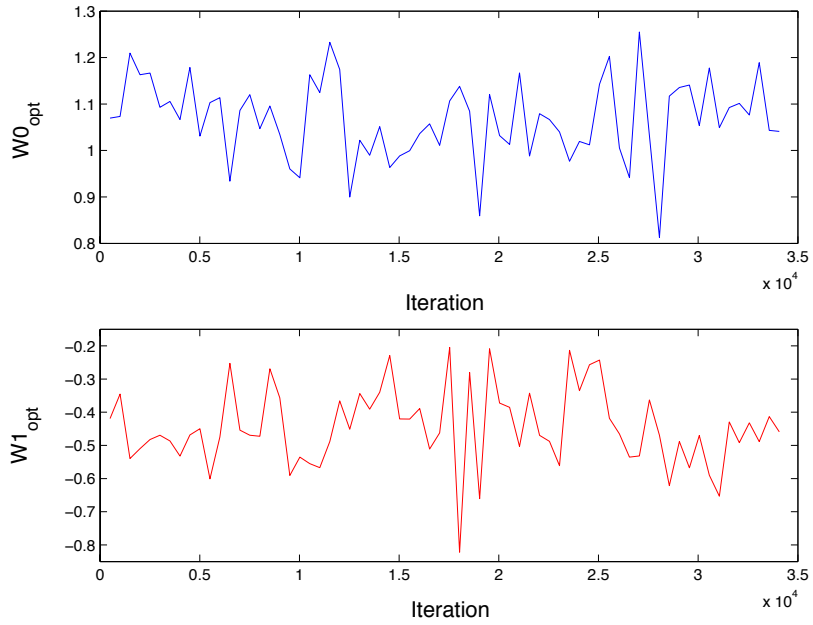

Figure 3.2: Optimum filter weights, $\mathbf{w}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{P}$, at different iterations. A window of length 200 is used to compute $\mathbf{R}$ and $\mathbf{P}$.

minimum, ξ_{min}, at different weights for the two different data segments. This shows that for non-stationary inputs, the performance surface does not have a unique minimum value and that it moves around in the space defined by the weights.

The non-stationary behavior of the input can also be visualized by tracking the optimum filter coefficients through the data sequence. Again a window size of 200 was used to compute $\mathbf{R}$ and $\mathbf{P}$. The optimum filter weights were then obtained for for every 200^{th} sample index. The result is shown in Figure 3.2. It can be see than the optimal filter weights are different depending on which set of samples we are looking at. Each optimal filter weight produces a different value of ξ_{min} and this is shown in Figure 3.3.

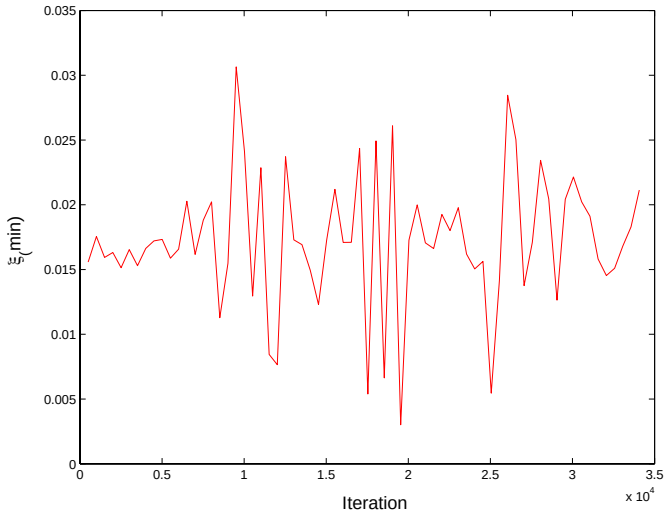

Figure 3.3: Minimum mean squared error (ξ_{min}) at different iterations. A window of length 200 is used to compute $\mathbf{R}$ and $\mathbf{P}$.

Thus, non-stationarity poses a problem for LMS adaptation. In this case, it is not enough for the algorithm to converge to a particular optimal value of the weights. The algorithm should be robust enough track changes in the statistics of the input and then converge to the new set of optimal weights. The convergence and tracking behavior of normalized LMS is studied in the next section.

3.2 Convergence and Tracking Behavior

The convergence behavior of the adaptive filter can be observed in a lot of ways. The learning curve or the path followed on the performance surface while achieving ξ_{min} are usual methods of looking at the convergence behavior. However, these performance metrics do not allow easy visualiza-

tion/interpretation of the tracking behavior.

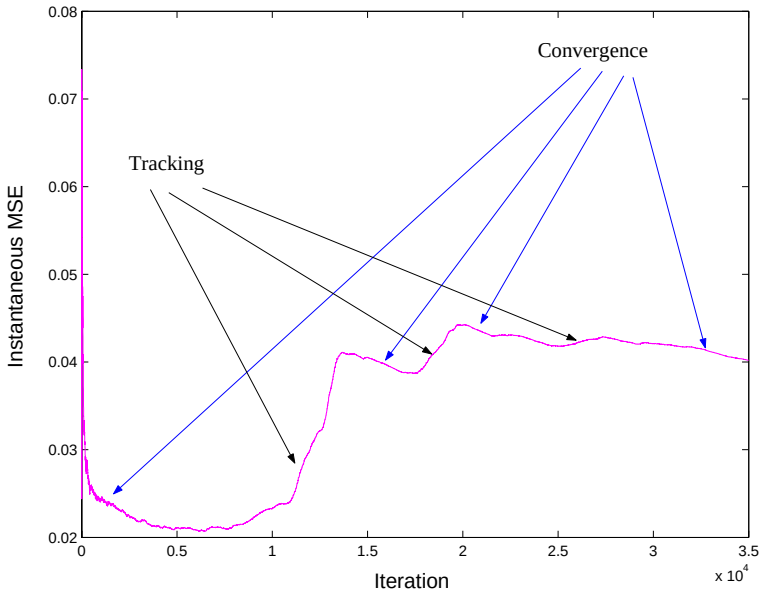

Figure 3.4: Convergence and tracking behaviour of the normalized LMS with $\alpha = 0.01$.

The instantaneous MSE offers a convenient way of observing the tracking behavior of the adaptive filter. The instantaneous MSE is shown in Figure 3.4. It can be seen that the adaptive filter first tries to converge to an optimal set of weights. As soon as the input statistics change (speech begins or a different word begins), the filter starts tracking the change and then tries to converge to the new set of optimal weights. The convergence regions and the tracking regions are indicated in Figure 3.4.

The convergence behavior can be assessed by looking at the progression of weights during the iterations. This is shown in Figure 3.5. With both the step sizes shown, it is seen that the filters try to converge to the vicinity of

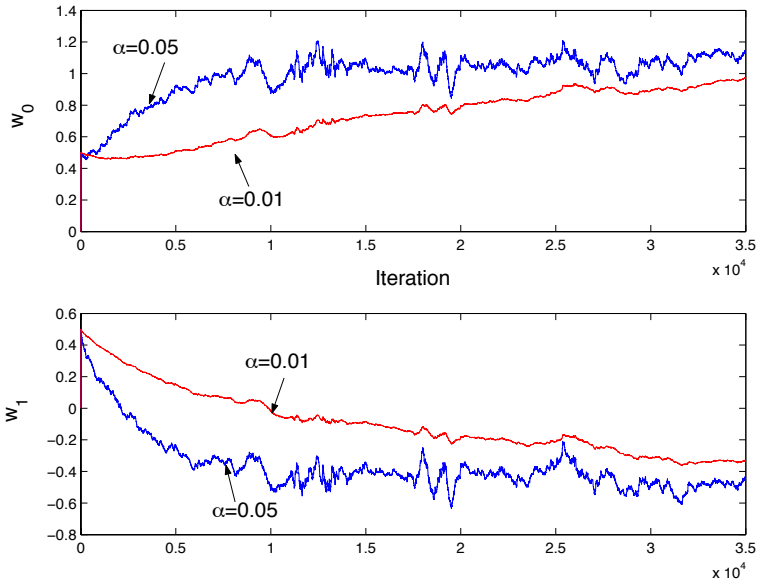

Figure 3.5: Convergence of the weights using normalized LMS with two different step sizes.

some steady-state value. It is observed that a larger step size causes faster convergence but there is a lot of oscillations around the steady state value. A smaller step size shows slower convergence but the oscillations around the final value are very small. This suggests that the filter with a larger step size tries to track variations in the signal statistics faster creating oscillations in the filter tap weights. A filter with a smaller step size cannot track signal variations that fast and thus shows the smooth behavior seen in Figure 3.5.

Another convenient way of visualizing the convergence behavior is to look at the path that the filter follows on the performance surface. Initially, the filter weights are set to random values between 0 and 1. This gives a random point on the performance surface. As the filter adapts, the points on

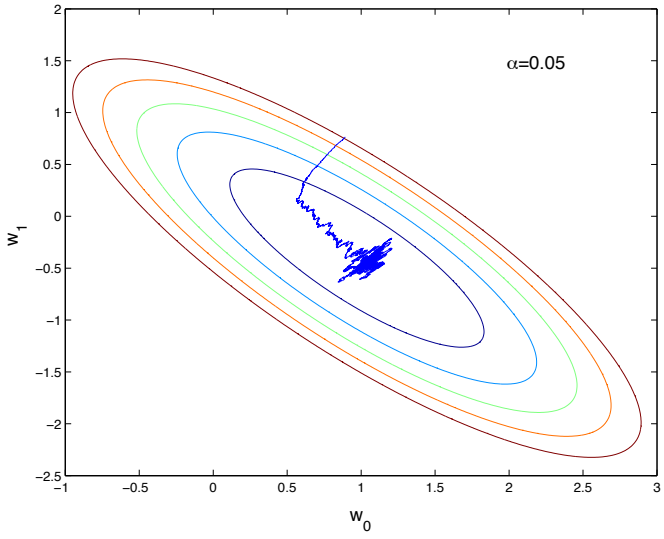

Figure 3.6: Convergence of the MSE (ξ) on the performance surface. Normalized LMS with $\alpha = 0.05$ is used .

the performance surface should move towards the bottom of the quadratic surface, since the MSE should decrease in each iteration. Ideally, the minimum on the performance surface would eventually be reached. If instead of looking directly at the performance surface, we looked at the contours of the performance surface, we would start at point far away from the center (note that the center of the performance contours corresponds to ξ_{min}) and gradually move towards the center as the iterations progressed. Remember that the performance surface shifts for non-stationary signals and therefore, a point in the vicinity of the center and not the actual center is reached. This behavior is shown in Figures 3.6 and 3.7. As in the case of the weight convergence, it is seen that a larger step size creates an oscillatory path towards the minimum whereas a smaller step size follows a smooth path. Note

that this indicates that a larger step size stops at a value farther away from the true center when compared to a filter with a smaller step size. Thus, we can expect that a filter with a smaller step size produces a steady state MSE that is smaller than the steady state MSE for a filter with a larger step size. This fact is demonstrated in the next section.

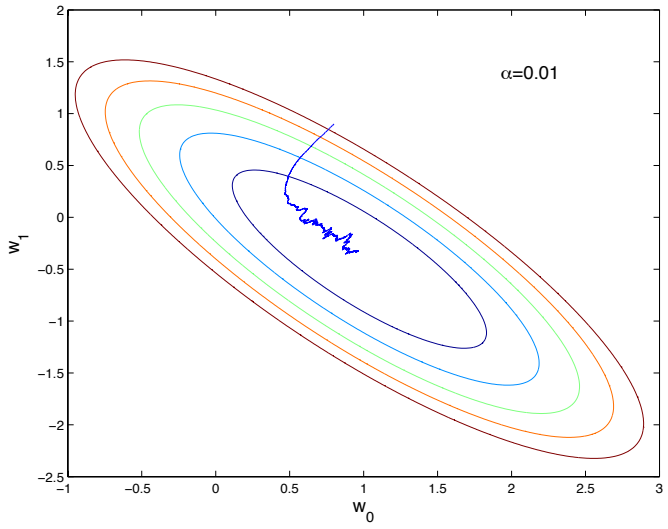

Figure 3.7: Convergence of the MSE (ξ) on the performance surface. Normalized LMS with $\alpha = 0.01$ is used .

3.3 Learning curves

The progression of the MSE as the number of iterations increase is called the learning curve. Usually for stationary signals, the MSE decreases as the filter adapts and then reaches a steady state value, that produces a point on

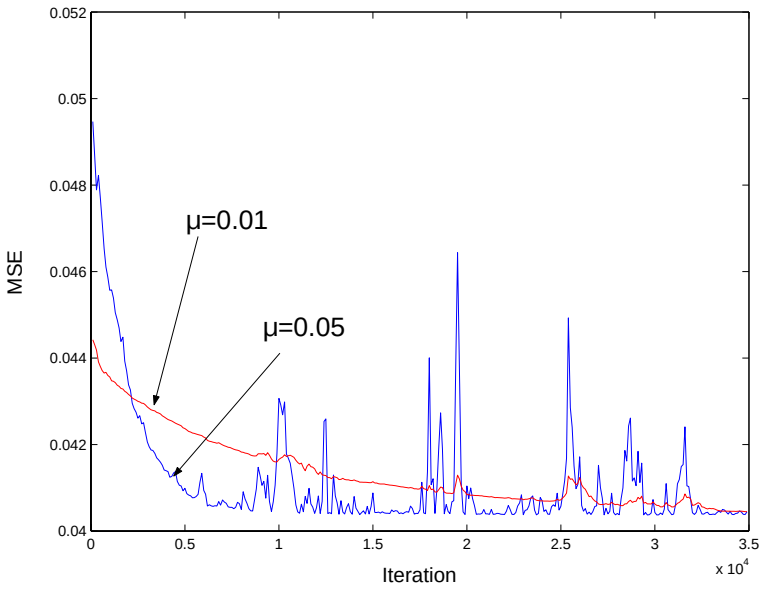

Figure 3.8: Convergence of the MSE to $\xi_{\min}$ on the performance surface. Normalized LMS was used with $\alpha = 0.05$.

the performance surface that coincides with $\xi_{\min}$. The performance surface does not have a unique minimum value for non-stationary signals. The speed of adaptation can be easily interpreted using the learning curve. A learning curve that shows a sharp transition and achieves the steady state value soon implies fast convergence whereas a learning curve that reaches the steady state value slowly implies slow convergence. The steady state value gives an indication of the effectiveness of the adaptation. A high steady state value implies that the residual error after the filter has adapted is large and hence the adaptive filter is not very good. A performance metric involving the steady state MSE value (ξ_∞) is called the misadjustment and will be studied in the next section. The learning curve for the two-tap

filter is shown in Figure 3.8. It is seen that a lower step size makes the filter converge slowly when compared to a higher step size. Though the higher step size shows faster convergence, it produces a lot of ripples after convergence. Note that the lower step size also produces ripples at the exact same sample indices. This could indicate locations where the statistics of the input (desired response) changes. For example there is a ripple near sample 1000, which could indicate the start of speech. A higher step size tries to track these changes faster causing bigger ripples in the MSE. A smaller step size converges to a lower value of the MSE. Looking at the behavior after convergence suggests that a larger step size should have a higher residual error. This is illustrated in the next section.

3.4 Misadjustment

The convergence behavior can be visualized using the learning curves. However, this is not a convenient measure for looking at the long term behavior of the LMS algorithm. To quantify the steady state MSE achieved by the LMS algorithm, let us first introduce some terminology. Let $\xi(n)$ be the MSE produced by the LMS filter at iteration n. Let the final value of the MSE be denoted by ξ_∞. The difference between ξ_∞ and the minimum value $\xi_{\min}$ achieved by the optimal Wiener filter is called the *excess* MSE, $\xi_{\text{ex}}(\infty)$. The ratio of the excess MSE to the minimum MSE is called the *misadjustment*, $\mathcal{M}$. To summarize,

$$\xi_{\text{ex}}(\infty) = \xi_\infty - \xi_{\min}, \tag{3.1}$$

$$\mathcal{M} = \frac{\xi_{\text{ex}}(\infty)}{\xi_{\min}}. \tag{3.2}$$

Thus, the misadjustment is a measure of how far the steady-state solution is from the Wiener solution. We compute the misadjustment as follows. The data is divided into blocks of length 200. In each window, we use time averages to compute the input autocorrelation $\mathbf{R}$ and the cross-correlation $\mathbf{P}$ (see eqns. (2.6) and (2.7)). Then $\xi_{\min}$ is calculated using (2.11) and (2.12). Then we use the value of the weights that the LMS algorithm has converged to in that window in (2.5) to compute ξ_{∞}. A time average is used to compute $E[d^2(k)]$. The misadjustment is then computed for each window using (3.2). Then the time average over all windows is used to compute the misadjustment. The LMS algorithm with random initial weights is used. Therefore, the misadjustment could depend on the initial value of the weights used. To get rid of this dependence on the initial values of the weights we use 10 independent realizations of the LMS algorithm and perform an ensemble average to get a final estimate of the misadjustment.

The misadjustment for a two-tap filter as a function of the step size is shown in Figure 3.9. Very small step sizes will force the filter to converge too slowly to be of any use and thus causes a high MSE which translates to a large misadjustment. Further increases in step size causes convergence which decreases the MSE and thus the misadjustment. The misadjustment decreases till an optimal step size is reached after which it increases. This increases could be attributed to the larger MSE that a filter with a large step size converges to and the ripples in the MSE after convergence (see Figure 3.8).

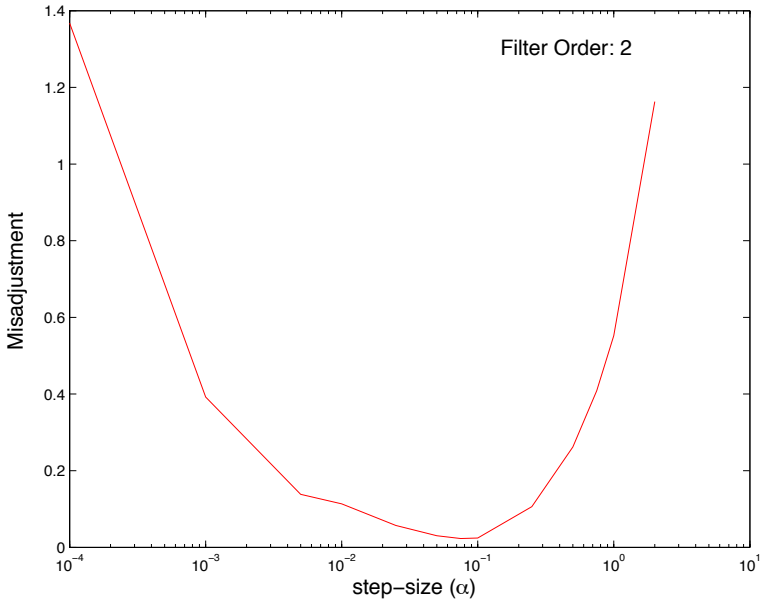

Figure 3.9: Misadjustment for a two-tap filter as a function of step size.

3.5 Signal-to-noise ratio improvement

Another performance metric that can be used to study adaptive filters is the signal-to-noise (SNR) ratio improvement. The SNR improvement is defined as the ratio between the output SNR and the input SNR,

$$SNR_{\text{imp}} = \frac{SNR_{\text{out}}}{SNR_{\text{in}}} \tag{3.3}$$

Since we are not given an estimate of the signal power at the input, we use the power of the desired response as the input power. The reference noise samples are correlated with the noise samples embedded in the speech and hence the power of the reference noise is a very coarse estimate of the input noise power. The output of the adaptive filter is an estimate of the noise in the speech signal. In the case of perfect adaptation, the noise in the

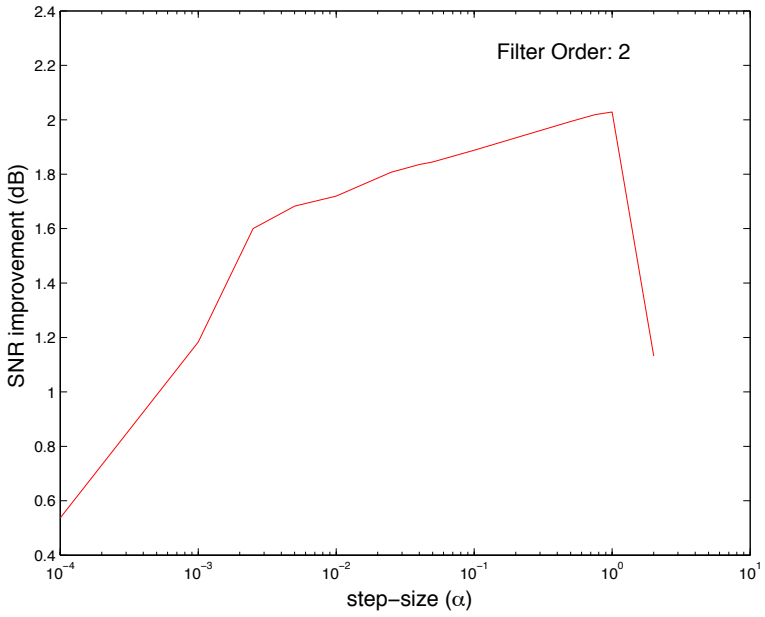

Figure 3.10: Misadjustment for a two-tap filter as a function of step size.

speech is identical to the output of the adaptive filter and the output of the noise canceler will have clean speech. But now we have speech corrupted with a smaller noise power. We use the power of the error(noise canceler output) to quantify the output signal power. Therefore we have,

$$SNR_{\text{imp}} = \frac{E[d^2]}{E[e]^2}, \tag{3.4}$$

where $e(n) = d(n) - y(n)$ represents the output of the noise canceler.

The SNR improvement in decibels is shown as a function of the step size in Figure 3.10. Again the SNR improvement was time-averaged over the sample size and ensemble-averaged over 10 independent runs of the LMS algorithm. It is observed the SNR improvement increases with step size and then decreases. For very small step sizes, the filter does not converge fast

enough. The high misadjustment leads to a small SNR_{imp}. Again, for high step sizes, the misadjustment is high causing a small SNR_{imp}. In this case the optimal step size is around 0.01. In Figure 3.8, a step size of 0.05 had more ripples after convergence than a step size of 0.01. Correspondingly, the SNR improvement is higher for $\alpha = 0.01$ when compared to $\alpha = 0.05$.

3.6 Frequency Response

In this section the frequency response of the two-tap adaptive filter is studied. The non-stationarity of the given data does not facilitate taking the discrete Fourier transform (DFT) to get the frequency domain representation. To illustrate the principles involved, we look at a small window of data between sample 20000 and 25000 and take the DFT of this window. This particular window was chosen because this window includes both speech an background noise. Windows towards the beginning and end of the given data samples contain only noise and thus is not a good choice for observing the frequency domain behavior.

The frequency domain representation of the reference noise, desired signal and the output of the adaptive LMS filter is shown in Figure 3.11. We observe that the reference noise and the noise in the desired signal are highly correlated. i.e, both of them produce spikes with comparable powers at the same frequencies. Also, note the spikes at the lower frequency spectrum ($< 4KHz$). Since speech lies in the lower frequency spectrum, the noise in this part of the speech produces more distortion to the speech. A good noise canceler should suppress these spikes. Note especially, the spikes around $100Hz$, $1.3KHz$ and $2.1KHz$. We shall compare the effects of the filters with these spikes as reference. Note that with a two-tap filter, the spikes

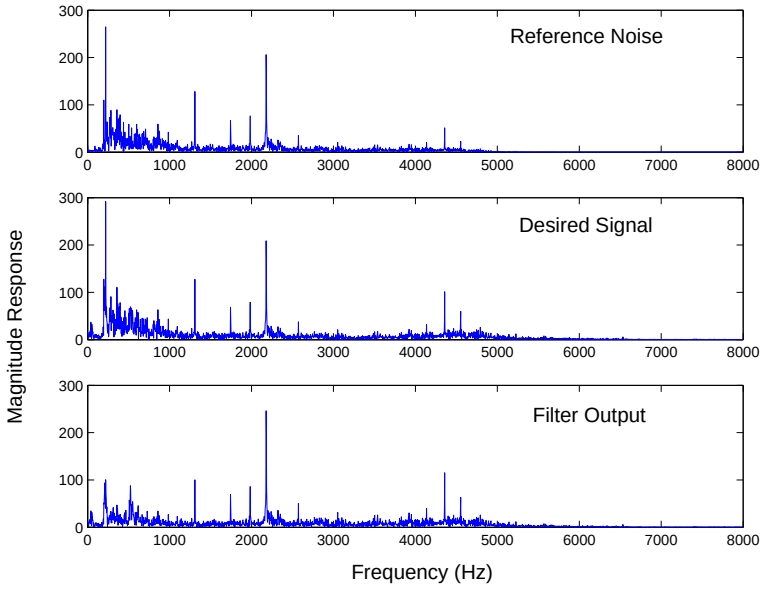

Figure 3.11: Frequency domain representations of signals at various locations in the noise canceler.

at $100Hz$ is almost completely removed whereas the spikes at $1.3KHz$ and $2.1KHz$ are well suppressed. This can be explained by looking at the frequency response of the filter which is shown in Figure 3.12. We see a deep notch around $100Hz$ and very small amplitude gains at the other frequencies mentioned above. Thus, the two-tap filter is reasonably successful in getting rid of the noise.

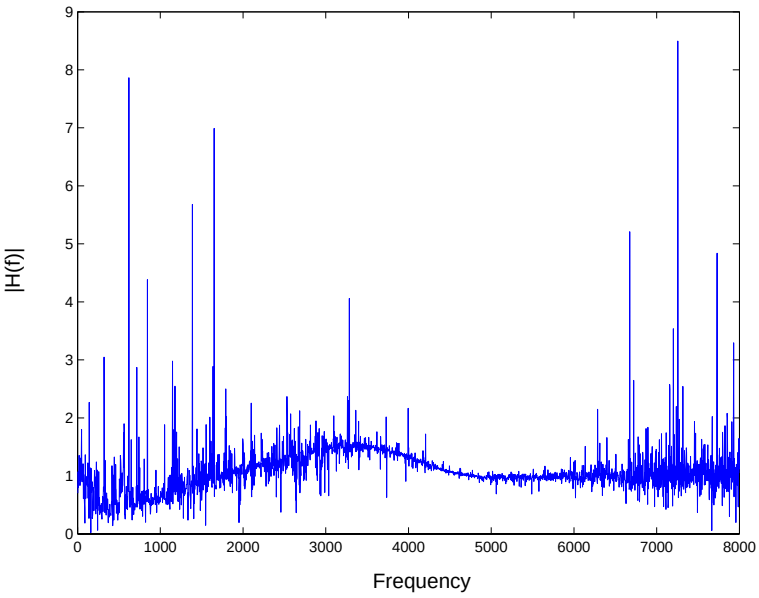

Figure 3.12: Frequency response of the two-tap adaptive noise canceler.

Chapter 4

Performance with higher filter orders

The performance of filters with higher filter orders is demonstrated in this chapter. The misadjustment ($\mathcal{M}$) is shown in Figure 4.1 for three different filter orders. It is seen that the misadjustement is lower for higher filter order. This is because a higher filter order is able to better approximate the relation (correlation) between the reference noise and the noise in the desired signal. It is also seen that a higher filter order reaches its optimum performance (minimum $\mathcal{M}$) at a smaller step size. This is because the maximum allowed step size (in order to keep the filter from diverging) decreases with the filter order. The maximum allowed step size is inversely proportional to the filter order. Thus, a higher-order filter reaches its minimum $\mathcal{M}$ at a smaller step size when compared to a filter with a smaller order. This fact is also corroborated by plotting the misadjustment as a function of the filter order for a fixed step size. Thus, we see that keeping the step size fixed and increasing the filter order causes and increase in $\mathcal{M}$. This fact has also been

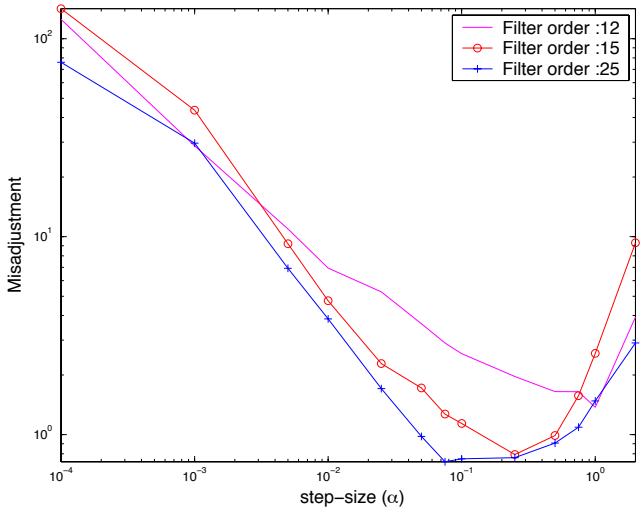

Figure 4.1: Misadjustment as a function of step size for different filter orders.

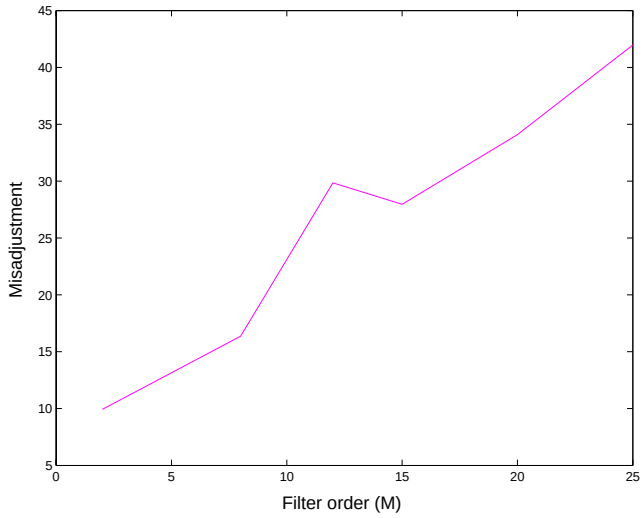

Figure 4.2: Misadjustment as a function of filter order for a fixed step size of $\alpha = 0.001$.

demonstrated by plotting the learning curves for filters of different order in Homework 3. Note that minimum achievable misadjustment for an order-15 filter is considerably lower than an order-12 filter. At the same time it is not much worse than the lowest misadjustment achieved by an order-25 filter. This suggests that going from a filter of order 15 to a filter of order 25 provides only negligible gains. Thus, to keep the implementation complexity and latency of the filter down, we suggest the use of an order-15 filter for this application.

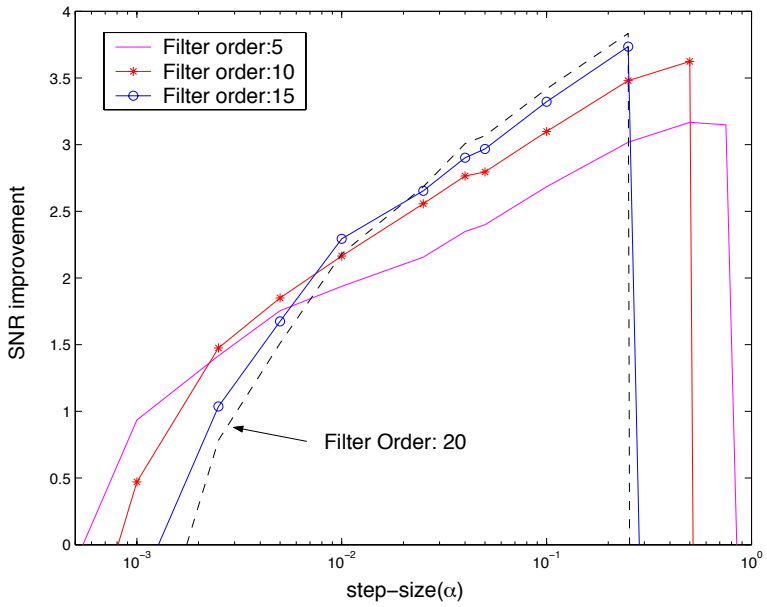

Figure 4.3: SNR improvement (SNR_{imp}) as a function of step size for different filter orders.

Next, we demonstrate the behavior of the SNR improvement as a function of the filter order. It is natural to expect a higher-order filter to estimate the noise in the desired signal with a higher accuracy and hence lead to a

better improvement in the SNR. The SNR improvement as defined in the previous chapter (see eqn. (3.4)) is shown in Figure 4.3 for different filter orders. It can be seen that a higher filter order produces higher SNR improvements. As in the case of the misadjustment behavior, it is seen that a higher order filter achieves the maximum SNR_{imp} at a smaller step size than a lower-order filter. For very small step sizes ($< 10^{-2}$), a filter with a higher order produces a smaller SNR than a filter with a smaller order. This again suggests that as the filter order goes up, the step size should be reduced in order to harvest the benefits of a greater number of taps in the filter. Also note that the SNR improvement between the filters of orders 15 and 20 is almost the same. Hence, to keep the delay and processing requirements small, we recommend the use of a filter of order 15. Any subsequent increase in the size of the filter will only produce marginal improvements. Thus, looking at both the misadjustment and SNR improvement, we see that a filter of order 15 gives very good performance at an affordable complexity.

We know look at the effectives of the order 15 filter by studying its frequency response. As mentioned in the previous chapter, we look at the noise spikes at $100Hz$, $1.3KHz$ and $2.1KHz$. The spectra of signals at various locations in the filter are shown in Figure 4.4 Note that the spike at $100Hz$ is completely suppressed. The two-tap filter does not suppress the spike at $2.1KHz$ very well (see Figure 3.11). Note that the 15-tap filter has suppressed the spike at $2.1KHz$ by more than a factor of 2 when compared to the two-tap filter. The frequency response of the filter is shown in Figure 4.5. Note that the response shows clear notches at the frequencies mentioned. Thus, the filter with order 15 forms a much better noise canceler than the two-tap filter.

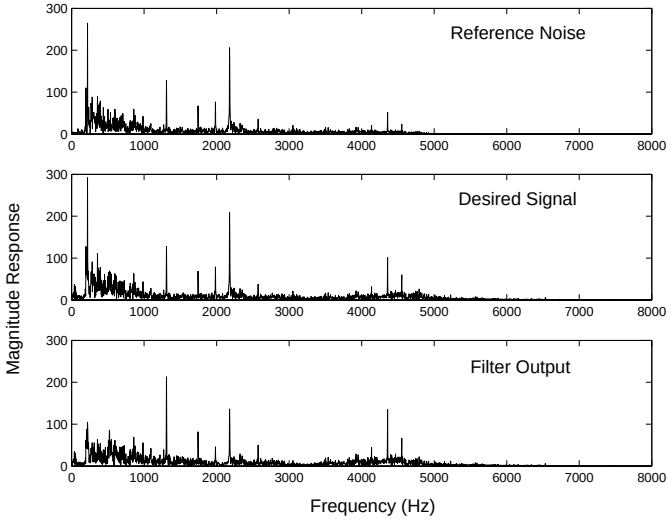

Figure 4.4: Frequency domain representations of signals at various locations in the order 15 noise canceler.

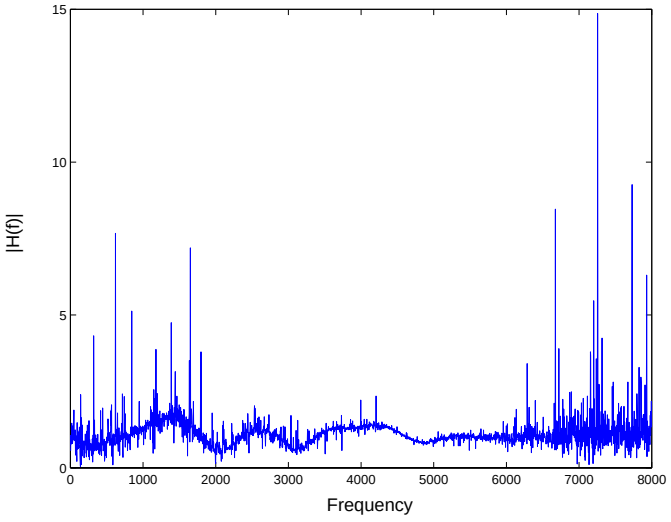

Figure 4.5: Frequency response of the adaptive noise canceler.

Chapter 5

Concluding Remarks

Based on our simulation study the following conclusions can be made about the LMS-based adaptive filter.

- For non-stationary signals, the optimal filter weights change with time. Simulations results show that the LMS algorithm can track these weight changes effectively. The tracking time is inversely proportional to the step size while the minimum MSE (and therefore the misadjustment) is directly proportional to the step size. Thus, there is a tradeoff between speed of adaptation/tracking and the MSE.

- A two-tap filter is not effective in bringing out the correlation between the reference noise and the noise in the desired signal.

- The misadjustment, $\mathcal{M}$, for non-stationary inputs is also due to two factors. The first one is weight-lag (due to sudden change in optimal weights and the lag for the filter to track it) and the second is gradient noise (remember that LMS uses a very noisy estimate of the gradient). A smaller step size leads to a smaller misadjustment. However

extremely small filter sizes does not allow the filter to converge causing an increase in convergence.

- The SNR improvement increases with filter order M (provided step size is reduced considerably). For the same (small enough) step size, higher order filter produces a smaller SNR improvement.

- The step size that achieves optimum performance, smallest misadjustment or largest SNR improvement, decreases with an increase in step size.

- By observing the decrease in misadjustment and increase in SNR improvement, we recommend a filter with order 15 to be used in this particular noise canceling application.

Bibliography

[1] J. C. Principe, *Adaptive Signal Processing.* EEL 6502: Class Notes, Spring 2004.

[2] S. Haykin, *Adaptive filter theory.* 4th ed., Prentice Hall, 2001.