Exploration of Alternative Algorithms for Multi-Channel Acoustic Echo Cancellation

by Julian Chacon-Castaño

S.B. EE. MIT, 2018

Submitted to the
Department of Electrical Engineering and Computer Science
In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1, 2020

Author .....
Department of Electrical Engineering and Computer Science
June 18, 2019

Certified by .....
Joseph Steinmeyer
Lecturer
Thesis Supervisor

Certified by .....
Ricardo Carreras
Manager, Bose Co.
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Exploration of Alternative Algorithms for Multi-Channel Acoustic Echo Cancellation

by

Julian Chacon-Castaño

Submitted to the Department of Electrical Engineering and Computer Science
on June 18, 2019, in partial fulfillment of the
requirements for the Degree of
Master of Engineering in Electrical Science and Engineering

Abstract

Multi-Channel Acoustic Echo cancellation (MCAEC) is a vital component of delivering clean speech to a virtual personal assistant through a smart speaker with multi-channel audio (stereophonic, etc). The use of the Kalman filter as an alternative adaptive filter methodology for this MCAEC application is explored in this work. The Normalized Least Mean Squares filter (NLMS) serves as a benchmark for the Kalman filter. Simulations using room recordings and measured room responses are employed in this exploration. Useful metrics such as the Word Error Rate (WER) and Echo Return Loss Enhancement (ERLE) help to distinguish performance among the two adaptive filter algorithms. For the single channel case, simulations confirm the cancellation and convergence rate advantage of the Kalman filter, in full-band, but the NLMS filter gives similar results in the sub-band domain, as measured by WER and ERLE. In the multi-channel case, both solutions achieve similar steady state cancellation, but the NLMS offers slightly faster convergence rates. In experiments where adaptation was not frozen, the Kalman filter effectively maintains high echo cancellation by tracking input signal statistics. In most cases, the Kalman filter does not present an appropriate alternative for the MCAEC application in this work.

Thesis Supervisor: Joseph Steinmeyer
Title: Lecturer

Thesis Supervisor: Ricardo Carreras
Title: Manager, Bose Co.

Acknowledgments

Thank you to everyone at Bose and MIT that helped make this thesis work possible, and guided me in the exploration of a challenging problem. I thank my company supervisors, Ric Carreras and Mark Sydorenko, for providing constant feedback and introducing me to a variety of resources that helped in facilitating my work.

I offer my gratitude to Bruce Levens and Al Ganeshkumar for providing two opposing, but extremely helpful, views on the multi-channel echo cancellation problem and for pushing me to continue forward whenever I hit a wall.

I want to thank my thesis advisor, Joseph Steinmeyer, for providing an outside perspective on several aspects of this problem and also in guiding logistics and completion of this work.

Finally, thank you to my parents, Rodrigo Chacon and Claudia Castaño, for your lifelong dedication to doing all things possible in helping me reach my goals.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rise of virtual personal assistants (VPA) on smart speakers has required the application of acoustic echo cancellation to deliver clean speech to an automatic speech recognition (ASR) engine. In this case, an acoustic echo occurs when a smart speaker plays music that reverberates throughout a room, and is captured by the microphone on the speaker. The music echo at the microphone interferes with the speech signal (wake-up word) that should be delivered to the VPA. This wake-up word (WUW) is typically a signal to the VPA to await further instruction. Hence, acoustic echo cancellation is needed to help deliver clean speech to the VPA. The canonical case of this phenomenon is known as single channel echo cancellation, where single channel refers to the presence of a single reference channel (i.e. one music signal). There is substantial literature on this particular problem, but an area of active research is in Multi-Channel/Multi-Party acoustic echo cancellation (AEC). Multi-channel refers to the presence of multiple reference signals, playing through separate speakers, echoing separately and summing at the microphone as a combined echo. The topic of this thesis is to explore alternative AEC algorithms for this application that have not been heavily researched before.

1.1 Motivations for Multi-channel Echo Cancellation

1.1.1 Adaptive filtering and AEC's

In order to motivate this problem, a brief description of adaptive filtering is needed. Adaptive filtering is a technique used to estimate an unknown transfer function. In this development, this transfer function is modeled as a set of unknown coefficients to a finite impulse response (FIR) filter. Figure 1-1 illustrates the general block diagram of an adaptive filter.

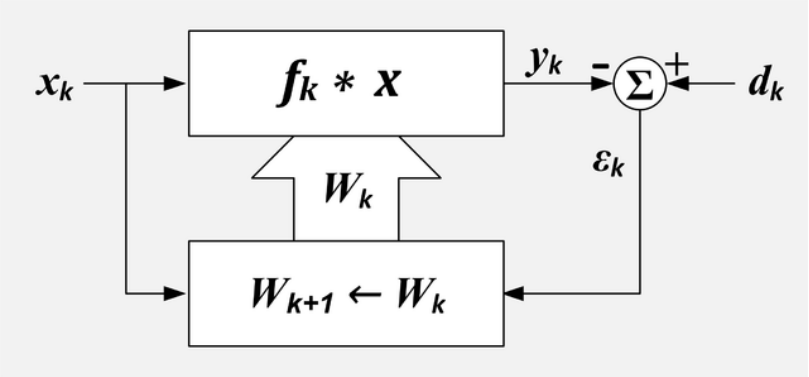


Figure 1-1: Adaptive Filter Block Diagram [3]

The filter works by convolving the input reference signal x_k with an estimated FIR filter response f_k to produce an output estimate y_k. This output estimate is then subtracted from the desired signal d_k, to produce the error signal e_k. The estimated filter coefficients W_k are then updated using the error signal, according to some optimization criteria, to produce new filter coefficients W_{k+1}. These coefficients are then used in the next iteration to convolve the reference signal and so on. Over time, the algorithm converges to the desired filter coefficients and cancels out the portion of the desired signal that is correlated to the reference signal. This algorithm ideally leaves any uncorrelated noise in the desired as a signal to be delivered or transmitted.

The adaptive filtering methodology can be applied to echo cancellation in the VPA case by mapping the signals in Figure 1-1 appropriately. The reference signal x_k is the music being played from the speaker. The desired signal d_k is the microphone

signal captured at the speaker (including music echo and WUW). The error signal is the difference between the estimate of the echo and the echo at the microphone. The filter then uses its optimization criteria to estimate the FIR filter model of the transfer function that represents the speaker to microphone path that music takes to produce the echo. This mapping is shown in Figure 1-2 below.

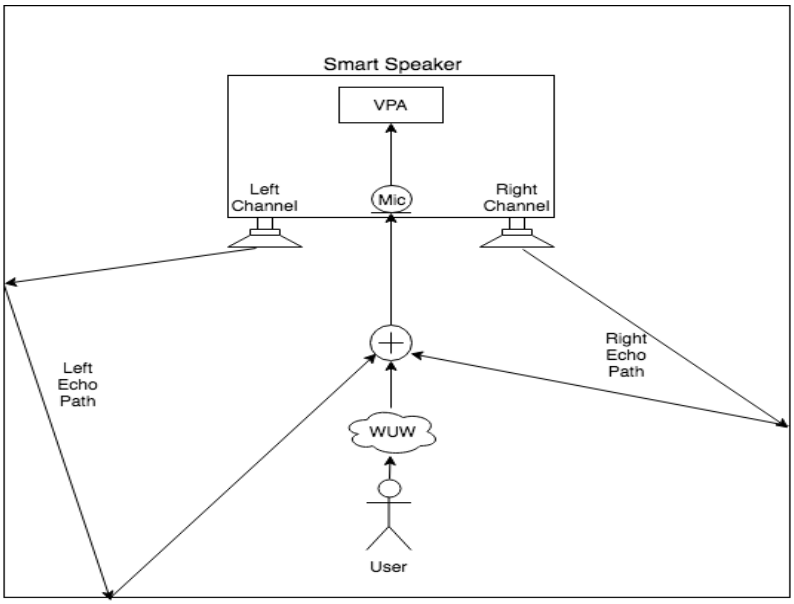


Figure 1-2: Smart Speaker Setup. User utters wake-up word (WUW), which is contaminated by music echo and received by microphone on the speaker. Note that the virtual personal assistant (VPA) above is a symbolic representation and not actually on the speaker.

1.1.2 Non-Uniqueness Issue: Multi-Channel

As mentioned in the introduction, the single channel case is well known and many techniques have been developed for its efficient implementation. The reason why the Multi-Channel case is still an active area of research is because of correlation between multiple input reference signals. Figure 1-3 illustrates this issue with a stereophonic (2-channel) case.

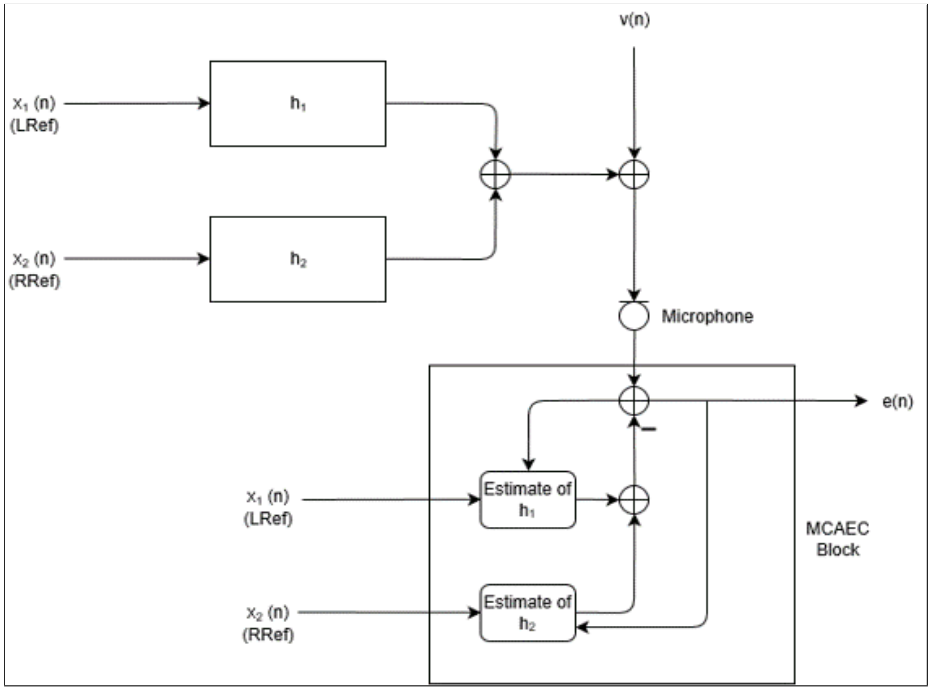


Figure 1-3: Stereophonic AEC

With Figure 1-3 above, we follow a small derivation to arrive at the error signal in the frequency domain. Ignoring the interference signal $v(n)$, the microphone signal is

$$x_1(n) * h_1(n) + x_2(n) * h_2(n)$$

and the output of the filter estimates is

$$x_1(n) * \hat{h}_1(n) + x_2(n) * \hat{h}_2(n)$$

where $\hat{h}_1(n)$ = Estimate of h_1 and $\hat{h}_2(n)$ = Estimate of h_2. The subtraction of the microphone (desired output) and the estimated output is expressed as the error

$$\begin{aligned} e(n) &= x_1(n) * h_1(n) + x_2(n) * h_2(n) - x_1(n) * \hat{h}_1(n) + x_2(n) * \hat{h}_2(n) \\ &= x_1(n) * \overline{h}_1(n) + x_2(n) * \overline{h}_2(n) \end{aligned}$$

where $\overline{h}_1(n) = h_1(n) - \hat{h}_1(n)$ and $\overline{h}_2(n) = h_2(n) - \hat{h}_2(n)$.

This leads to the following equation in the frequency domain.

$$e(n) = x_1(n) * \overline{h}_1(n) + x_2(n) * \overline{h}_2(n) \xrightarrow{\mathscr{F}} E(\omega) = X_1(\omega)\overline{H}_1(\omega) + X_2(\omega)\overline{H}_2(\omega) \quad (1.1)$$

According to Sondhi, the issue is that as the adaptive filter drives the error $e(n)$ and thus $E(\omega)$ in equation 1.1, down to zero, it is not guaranteed that $\overline{H}_1(\omega) = \overline{H}_2(\omega) = 0$, unless the two input signals $x_1(n)$ and $x_2(n)$ are completely uncorrelated [10]. This condition is necessary for the estimated filter taps to actually converge to the unknown filter taps. Therefore, even if the error is driven to zero, there is no guarantee that the estimated filter taps will maintain a low error, if input statistics (i.e. $X_1(\omega), X_2(\omega)$) change. This problem is referred to as the non-uniqueness issue because there are many solutions that the adaptive filter could be driven too, and they are more than likely not all correct. This issue establishes why Multi-Channel AEC is still an active area of research.

1.2 Prior Research

1.2.1 NLMS

The Normalized Least Mean Squares filter is the cornerstone for many adaptive filter applications in development and deployment. The reason for this is because of its simplicity, computational efficiency, and performance. A derivation of the filter equation is presented in Appendix A. The equation[1] and block diagram for the update of the filter coefficients are presented below in Figure 1-4.

[1]Note that bold quantities represent vectors and convolution is represented as an inner product.

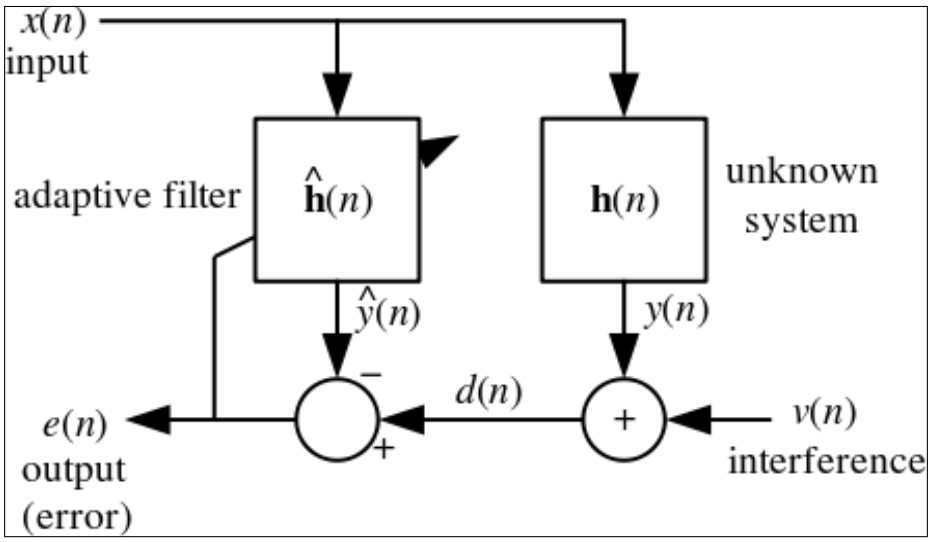


Figure 1-4: Adaptive Filter [5]

$$\hat{\boldsymbol{h}}(n+1) = \hat{\boldsymbol{h}}(n) + \mu \frac{e(n)\boldsymbol{x}(n)}{\boldsymbol{x}^T\boldsymbol{x}}$$

The step size parameter, μ, governs the performance of the filter with a simple trade-off. The trade-off is that of convergence and steady state error. For higher values of μ, a higher rate of convergence is observed as the filter takes larger steps in the direction of the estimated gradient (the error $e(n)$). However, this means a larger steady state error as the steps may bounce around the optimum coefficients. Conversely, lower values of the μ mean a slower rate of convergence, but also a lower steady state error since the filter is taking smaller steps towards the optimum coefficients.

Due to it's simplicity and computational efficiency, this filter serves as a typical benchmark for many new adaptive filter algorithms. Therefore, the NLMS filter is the main source of comparison for the alternative Kalman filter in this thesis. The NLMS filter is still expected to be deficient in the case of the Multi-channel AEC because it does not deal with possible correlations between the multiple input signals and therefore suffers the non-uniqueness issue[2].

[2]Section 1.1.2

1.2.2 Alternative Multi-Channel Techniques

Decorrelation

Since the main issue of the multi-channel case is that of correlation between input signals, one possible technique is to decorrelate the input signals. Decorrelation can be done by adding nonlinear distortion to the input signals. For example, say $x(n)$ is an input signal, and $f(\cdot)$ is a function that non-linearly distorts its input. The new input signal would then be

$$x_{new}(n) = x(n) + f(x(n))$$

This process is done for every input signal and the output signals are then used for the AEC[3] application. This has proven to work well for speech signal applications [2, p. 29] , but is not an option for the application at hand since distortion of music input signals is counter-productive for high-fidelity audio.

Widely Linear General Kalman Filter

The widely linear Kalman filter is an existing solution for stereophonic acoustic echo cancellation, in the teleconferencing case. The development of the filter is detailed in the paper by Paleologu, Benesty, Grant, and Ciochina, [9], but the basics are covered here for context. Figure 1-5 shows the filter block diagram in a teleconferencing case. For ease of analysis, it is best to think of the signals $x_L(n)$ and $x_R(n)$ as the equivalent reference signals for this Multi-channel VPA application.

[3] Acoustic Echo Cancellation

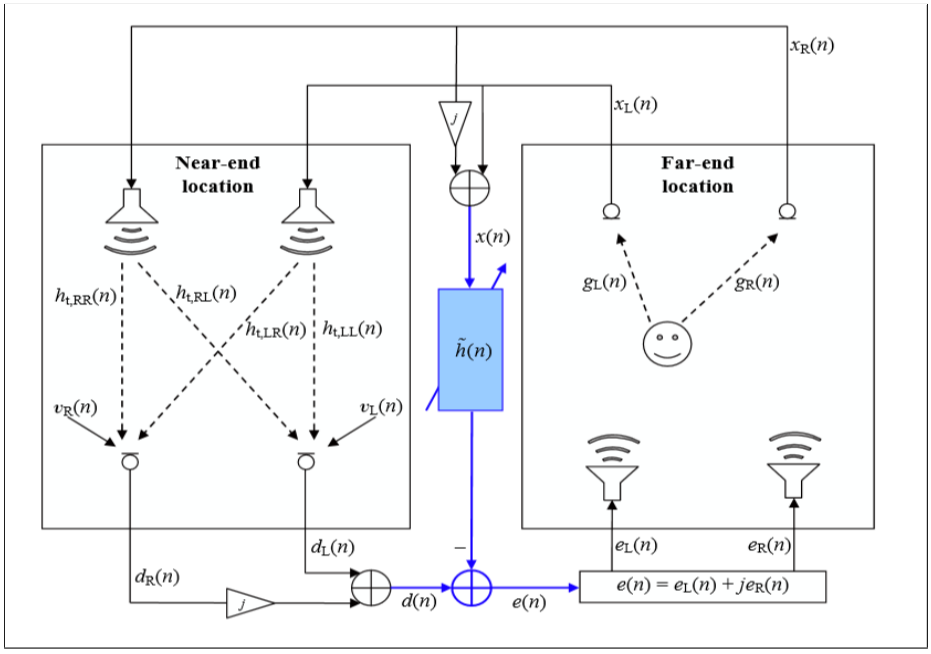


Figure 1-5: Widely Linear Kalman Filter for AEC [9]

The formulation of the filter is to take two input signals and use them as the real and imaginary parts of one complex signal and run the adaptive filter as a complex single channel (shown by the blue highlighted block). This boasts success in some applications, but did not seem promising as this mathematical formulation does not change the underlying physics of the problem (correlation between input signals). Also, in this VPA[4] case, there is only one receiving microphone, while in the widely linear model, there are two microphones for stereophonic audio transmission. This exercise is left for future work to see if the formulation would have any affect at all for the application in this thesis.

Multi-Party Kalman Solution

Another Kalman filter, discussed by Microsoft engineers assures good multi-party performance in audio spatialization. However, as noted using Figure 1-6, the formulation for their model does not map well to the application investigated in this thesis.

This filter is also explained comprehensively in the Microsoft presentation [12], but the main assumption is that the input channels x^i are independent. This does

[4]Virtual Personal Assistant

not apply to the VPA case since the relevant reference signals are indeed correlated.

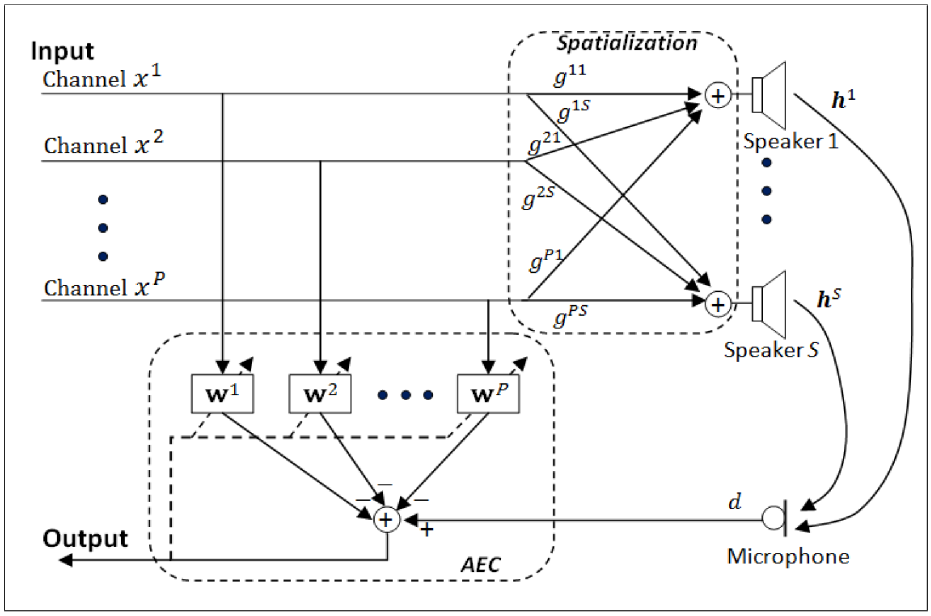


Figure 1-6: Microsoft Multi-Party Kalman Solution [12]

Chapter 2

Main Exploration: Kalman Filter

The main focus of this thesis is to research a novel Kalman filter system as a better solution to the Multi-Channel case of an AEC. For reference, the development of the Kalman filter for echo cancellation presented in [7] is included in Appendix A. This chapter presents a brief description of the application of the Kalman filter equations to acoustic echo cancellation, and why this formulation seems promising for the VPA case.

2.1 Kalman Filter AEC

The results of the Kalman filter derivation for acoustic echo cancellation are presented in the following equations. It is noted that rather than expressing relevant equations using convolution, they are presented using inner products and matrix notation.

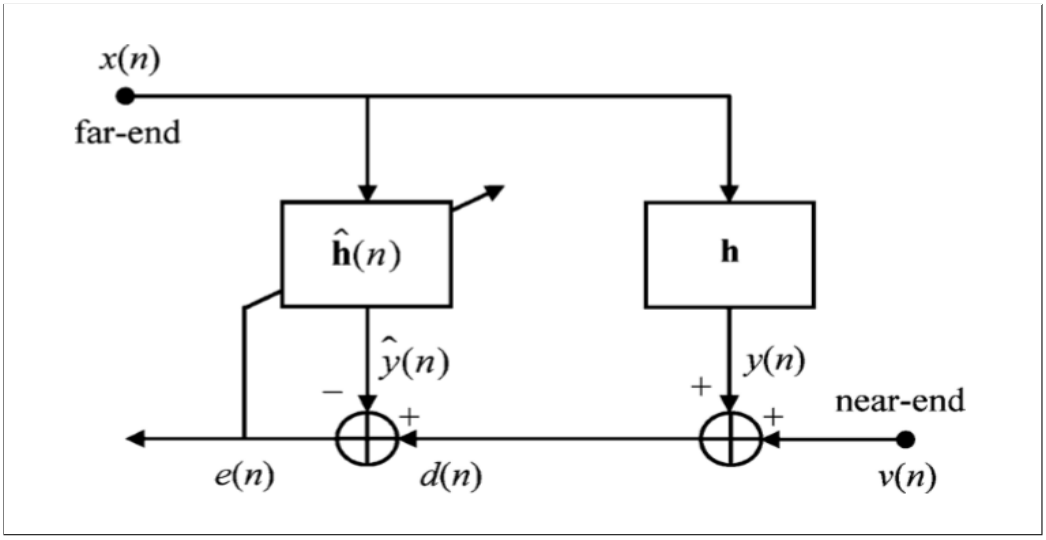


Figure 2-1: General AEC [7]

$$\boldsymbol{R}_m(n) = \boldsymbol{R}_\mu(n-1) + \sigma_w^2(n)\boldsymbol{I}_L \tag{2.1}$$

$$\boldsymbol{R}_e(n) = \boldsymbol{X}^T(n)\boldsymbol{R}_m(n)\boldsymbol{X}(n) + \sigma_v^2(n)\boldsymbol{I}_P \tag{2.2}$$

$$\boldsymbol{K}(n) = \boldsymbol{R}_m(n)\boldsymbol{X}(n)\boldsymbol{R}_e^{-1}(n) \tag{2.3}$$

$$\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{X}^T(n)\hat{\boldsymbol{h}}(n-1) \tag{2.4}$$

$$\hat{\boldsymbol{h}}(n) = \hat{\boldsymbol{h}}(n-1) + \boldsymbol{K}(n)\boldsymbol{e}(n) \tag{2.5}$$

$$\boldsymbol{R}_\mu(n) = [\boldsymbol{I}_L - \boldsymbol{K}(n)\boldsymbol{X}^T(n)]\boldsymbol{R}_m(n) \tag{2.6}$$

The initialization of the filter involves choosing $\hat{\boldsymbol{h}}(0)$ and $\boldsymbol{R}_\mu(0) = \epsilon I_L$, where ϵ is some small positive constant.

As mentioned previously, the derivation of these equations is explained in detail in Appendix A, while here they are used to define concepts for basic development. The first to note is the assumption that the room transfer function is the unknown filter response

$$\boldsymbol{h}(n) = [h_0(n), h_1(n), \cdots h_{M-1}(n)]^T$$

where M is the filter length. The Kalman filter is formulated in state-space modeling,

and so $\boldsymbol{h}(n)$ represents the state we are trying to estimate. With this,

$$\hat{\boldsymbol{h}}(n) = [\hat{h}_0(n), \hat{h}_1(n), \cdots \hat{h}_{L-1}(n)]^T$$

where $\hat{\boldsymbol{h}}(n)$ is the state estimate vector, with length L. In this application, the filter length of the state $\boldsymbol{h}(n)$ is expected to be very long due to typical acoustic reverberation times (100-200 ms [1]), and take "500-1500 taps at normal sampling frequencies"[1], to model [6]. Due to practical computational constraints, it is unlikely that the length of $\hat{\boldsymbol{h}}(n)$ will match that of the real state ($L < M$), so L becomes a parameter that can be varied and thus affect performance of the AEC. The number of reference input samples that are used to generate the output at each iteration depends on this length L. Hence, the input vector that is used every iteration n is

$$\boldsymbol{x}(n) = [x(n), x(n-1), x(n-2), ...x(n-L+1)]^T$$

Another notable parameter of the Kalman filter is its order P. This parameter determines how many blocks of input samples the filter uses every iteration. These blocks form the columns of the input $\boldsymbol{X}(n)$ which is a matrix of dimension LxP.

$$\boldsymbol{X}(n) = [\boldsymbol{x}(n), \boldsymbol{x}(n-1), \boldsymbol{x}(n-2), ...\boldsymbol{x}(n-P+1)]$$

The other quantities follow naturally. For example, the quantities $\boldsymbol{e}(n)$ and $\boldsymbol{d}(n)$ are vectors of length P, when $P > 1$. When P = 1, the Kalman filter follows the same development as the NLMS filter, where one sample of $\boldsymbol{e}(n)$ and $\boldsymbol{d}(n)$ are used every iteration.

2.2 Tracking and Noise Variances

The most potentially beneficial part of the Kalman filter for the AEC application is the assumption that the state is not time invariant, but instead dynamic. This

[1] 8 kHz or 16 kHz

assumption is true for the case when people move around in a room, thus changing the echo paths. This is captured in the Kalman filter through equation 2.7.

$$\boldsymbol{h}(n) = \boldsymbol{h}(n-1) + \boldsymbol{w}(n) \tag{2.7}$$

$\boldsymbol{w}(n)$ is a zero mean white Gaussian noise vector, with variance $\sigma_w^2(n)$. This noise variance is a measure of the uncertainties in the dynamic state [7, p. 13]. This parameter introduces the concept of tracking changes in the echo path or $\boldsymbol{h}(n)$. Remaining uncertain about the accuracy of the estimated filter coefficients allows for a faster response and tracking of any possible changes in the state. Thus, for low values of $\sigma_w^2(n)$, the Kalman filter achieves low steady state performance/cancellation, but maintains poor tracking capabilities. The converse is true of higher values of $\sigma_w^2(n)$. This introduces an important user-designed parameter that greatly affects the Kalman filter's performance, similar to the step size μ for the NLMS filter. There is a distinction, however, as tracking is not to be confused with convergence rate. Tracking refers to a response of the filter to a significant change in the echo path or $\boldsymbol{h}(n)$.

This variance and another ($\sigma_v^2(n)$), appear in the equations 2.1 and 2.2, respectively. These equations show how these noise variances affect the Kalman gain calculation (equation 2.3), although the noise variance $\sigma_v^2(n)$ is considered negligible for the VPA application as it is a measure of the background noise in the room. More often than not, the background noise power is typically much lower than the music echo in the room.

2.3 Explorations

The sections in this thesis consist of the use of the Kalman filter, in both single channel and multi-channel configurations, and also in full-band and sub-band versions. The single channel experiments were for the purpose of comparing the Kalman filter to the single channel NLMS. The main research focus was to use the Kalman filter in the trivial multi-channel configuration. This configuration refers to the update of the

left and right room responses, based on the use of the same error signal, but with the use of separate reference signals.

Chapter 3

Methods and Setup

This section describes the methods and general experimental setup used to benchmark the Kalman and NLMS filter in a variety of situations. These are used in conjunction with different data sets to aid in the exploration of the multi-channel acoustic echo cancellation (MCAEC) solution provided by the Kalman filter. All signals that were captured or used were recorded or synthesized at 16 kHz.

3.1 Room Recordings

Due to the data driven focus of this thesis, the AEC performance of these two algorithms was evaluated in a variety of simulated situations. These situations represented realistic test conditions such as different user/speaker placement, playback volumes, and instances of music.

The different test conditions were generated through the following process. First, transfer functions for several user/speaker placements were measured and then convolved with an audio file of 24 wake up word utterances from several different speakers. The output of this convolution was then added to music played in the room through the smart-speaker, captured by the smart-speaker microphone. The final outputs were referred to as "dirty files" (i.e. speech contaminated by music echo). This methodology was used for repeatability since physically recording these test cases might have presented unexpected changes in the echo path and other unpredictable results. The

input signal used in all of the simulations was a 50-second looping excerpt of a music track that presented a challenging audio environment.

This type of setup modeled realistic situations and recordings in which a user would be interacting with the smart speaker, as shown in Figure 3-1. As mentioned previously, since much of the work presented here was data driven, simulating realistic situations was important in evaluating these AEC algorithms, especially when the existing literature largely focuses on having access to acoustic impulse responses and other ideal[1] conditions.

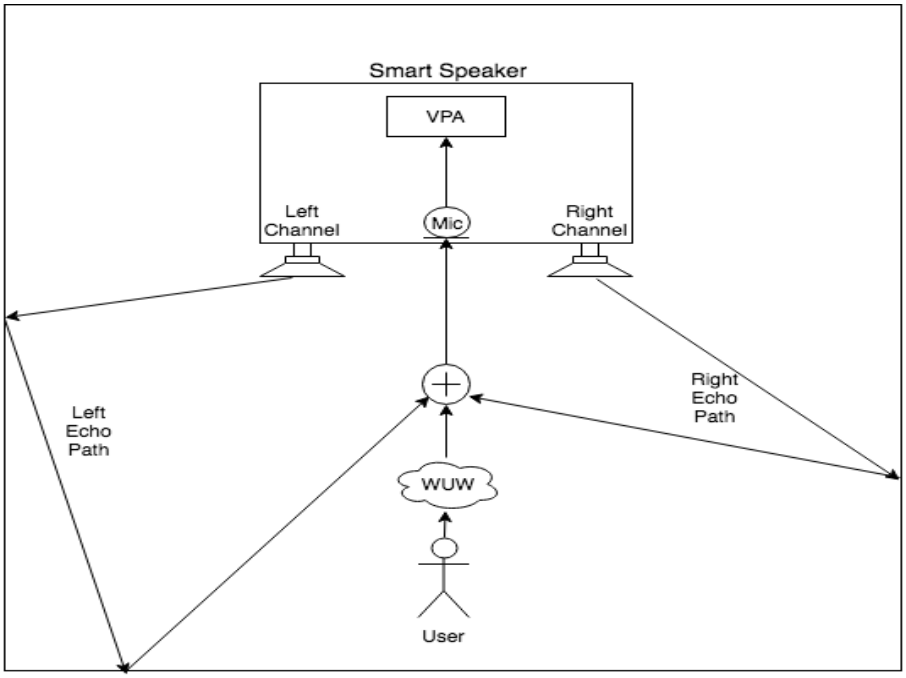


Figure 3-1: Smart Speaker and Room Echo Paths.

3.2 Impulse Responses

Along with the realistic room recordings setup, there were experiments conducted that included access to the actual FIR filter coefficients that produced the echo signal. With this inclusion, there were two main sets of impulse responses that were used. The general framework was to take these responses, convolve the input reference signals to produce the new microphone signal, and then run the AEC algorithm on

[1]White noise inputs, uncorrelated background noise, exact modeling of impulse response

this microphone signal. It should be noted that this set of "dirty files" only included music echo and possibly some background noise, but not wake up word utterances.

3.2.1 Recorded Responses

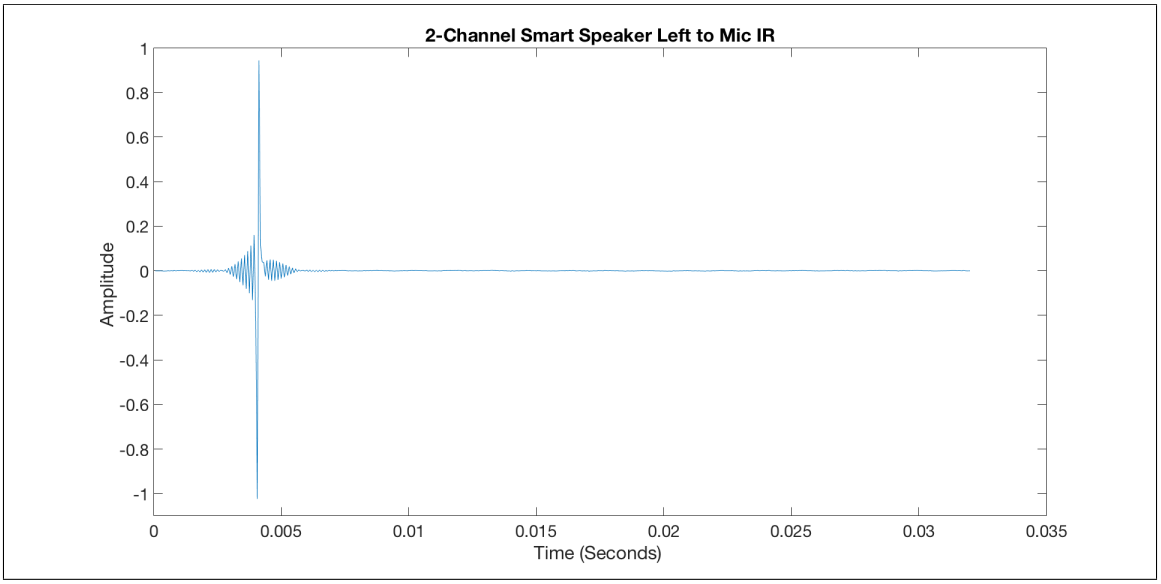


Figure 3-2: Left Speaker to Mic Impulse Response @ 16 Khz

For the most realistic replication of the room responses, it was best to measure the impulse responses (IR) from the left/right channels of a smart speaker to the microphone on the speaker. Figure 3-2 shows the impulse response for the left speaker. These impulse responses were acquired by playing a chirp signal through the speaker and capturing the room echo of the signal at the microphone. The transfer function was then calculated in the frequency domain by using the known input and recorded output signal. The frequency domain transfer function was then converted to a time domain vector and scaled to produce the response shown in the figure above. Since these responses could be measured for the left and right speakers, both were used for the multi-channel case.

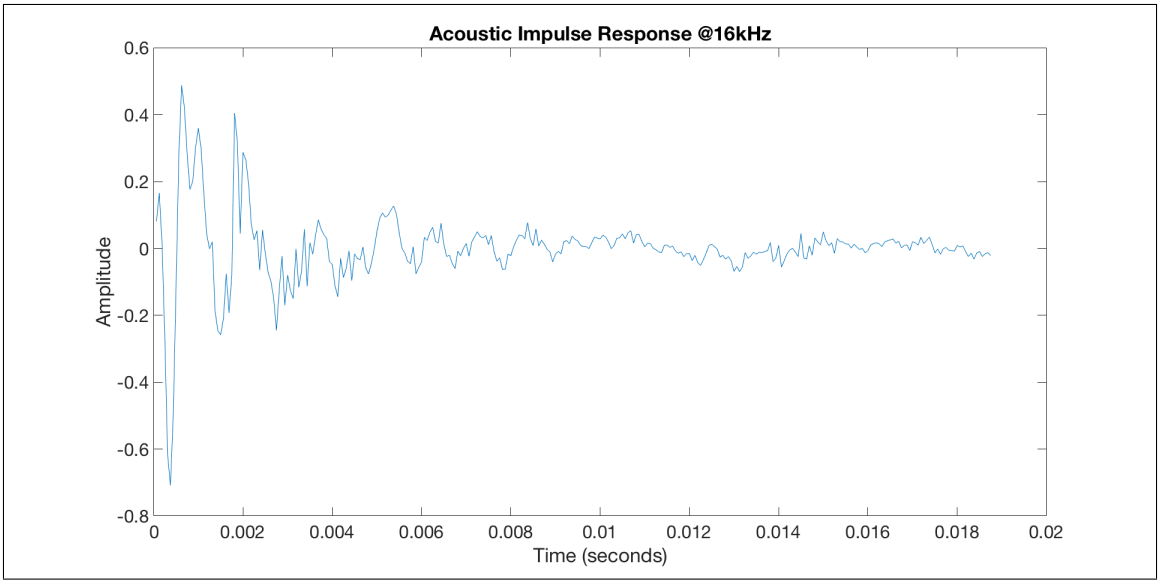


Figure 3-3: Acoustic Impulse Response @ 16kHz

3.2.2 Online Acoustic Responses

In addition to the recorded impulse responses, an acoustic impulse response that was acquired online was also used. This was used for single-channel recordings and was chosen as a generic impulse response, with properties of a typical acoustic echo response. This impulse response included majority of its power in the first reflections and a filter length of length 200ms. Although the original response was around this length, it was truncated to about 20ms (after which most of the energy had decayed) and used in order to allow for quicker simulations.

3.3 Simulation Environment

The entirety of simulations and experiments presented were performed using Matlab. This choice was made for ease of code development and fast prototyping. Most of the simulations used single floating point precision although some others used double floating point where specified. This choice of precision was made in order to appropriately compare with an existing commercial AEC that served as an essential benchmark. The code used for the NLMS and Kalman filters is included in Appendix B. These scripts represent the multi-channel, sub-band (complex) simulations for each

filter, but are also used with slight modifications for single channel and full-band cases.

3.3.1 Freezing Point

To discuss what the freezing point is, a particular case of the echo cancellation formulation must be introduced. This case is known as double talk. The name comes from the origins of echo cancellation in telephony, but is essentially a phenomena in which the signal power of speech affects adaptation. This happens because the interference presents a very uncorrelated signal to the input reference signal and thus causes the filter coefficients to adapt to incorrect coefficients. In the worst case, this interference can cause the adaptive filter to begin diverging [2, p. 104]. Due to this phenomena, a double talk detector is typically implemented to detect this situation and temporarily freeze adaptation by saving the current coefficients until double talk is declared over. Given that implementing an effective double talk detector was out of scope of this thesis, a manual freezing point was implemented instead. This meant that at a certain point in the simulation, adaptation was frozen and the converged coefficients were saved and used for the remainder of the simulation. This point was always chosen slightly before speech comes in, in order to allow the AEC enough time to converge.

Chapter 4

Metrics

Several metrics were used to evaluate the performance of the AEC algorithms and not all were available in every simulation. These metrics were useful in evaluating the AECs, but were not completely holistic. For example, computational considerations are not heavily emphasized in this work. Numerical stability is generally also a concern with the Kalman filter but the analysis of this metric was not included in this work either. The definition of each metric is included in this section.

4.1 Word Error Rate

The main goal of the AEC, in this context, was to provide clean speech to a VPA on a smart speaker. An important metric used to evaluate speech intelligibility is the word error rate (WER) of the VPA. Given an audio file, this metric is defined as the percentage of wake up words that the VPA correctly recognizes from a fixed set of wake up utterances in the file. In the context of this thesis, the audio file provided to the VPA was the "dirty file" described in section 3.1. The WER was determined by an automatic speech recognition (ASR) engine that had access to a transcript of all of the wake-up word utterances in the dirty file. The engine generated an output transcript of words recognized and compared this output to the original transcript to find what percentage of words it missed. Word Error Rate could then be calculated

as the ratio

$$\frac{\text{number of total words - number of words recognized}}{\text{number of total words}}$$

As such, a higher number for the WER indicates worse performance, and vice versa. This metric was only used when speech utterances were embedded in the audio files passed to the ASR engine. [1]

4.2 Echo Return Loss Enchancement

$$ERLE(n) = 10 \log \frac{\overline{\boldsymbol{d}^2}(n)}{\overline{\boldsymbol{e}^2}(n)} \tag{4.1}$$

The echo return loss enhancement (ERLE) is a staple metric for the performance of an AEC. This metric represents the amount of loss of the echo that returns in the error signal. The metric measures the power in the error signal relative to the input microphone signal that contains the echo. Below is the mathematical description, with relevant signals shown in Figure 4-1.

As cancellation of the echo signal takes place, the error signal approaches zero (when $v(n)$ is zero), and so the echo return loss enhancement increases. Ideally, the error signal will approach such that the interference left in the background ($\boldsymbol{v}(n)$). It should be noted that this metric is calculated as an average of the power in the error signal. The averaging window is arbitrary, but in this case it was chosen to be around 1 second based on how long the unknown filter length could be (128 milliseconds in the longest case). This decision was made to allow an average that spanned at least 2-3 times the filter length. Allowing the average to span at least 2-3 times the filter length permits the bouncing error signal to stabilize over a couple of filter lengths.

Another important consideration is that the ERLE calculation does not make as much sense when there is a large power in the interference signal $\boldsymbol{v}(n)$. Since this signal consisted of background noise and in some cases, speech utterances, the speech

[1] As mentioned in Chapter 3, not all experimental setups included speech files

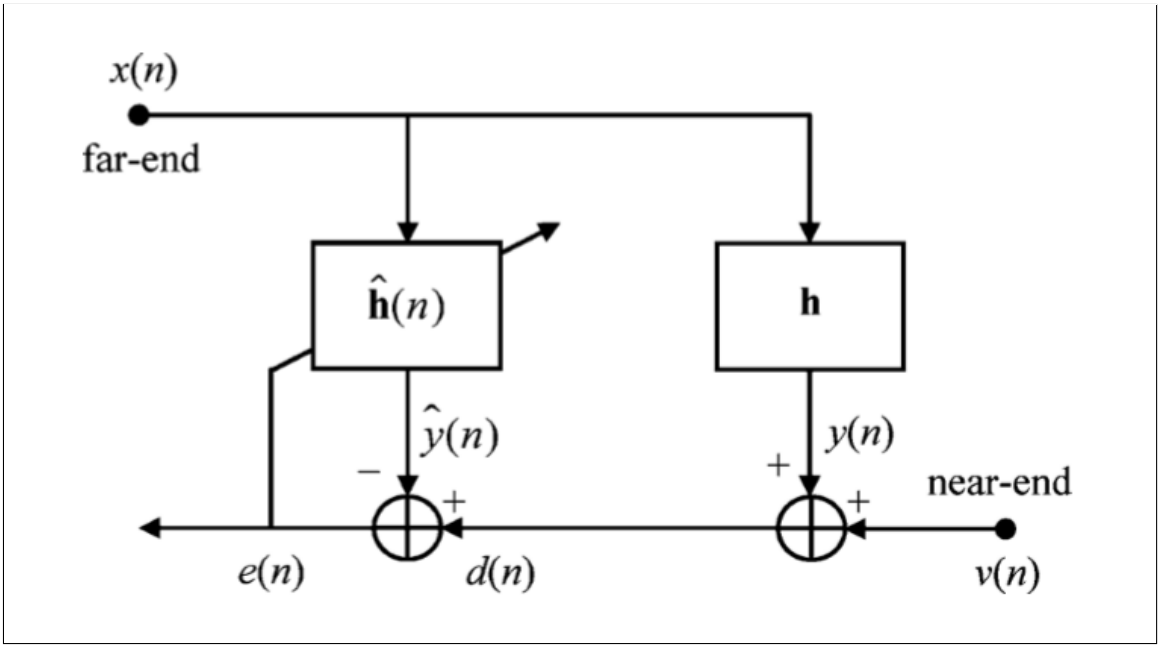


Figure 4-1: AEC configuration [7]

provided this large interference. This interference resulted in a momentary spike in the ERLE calculation, which may be misleading in terms of the performance of the actual cancellation. As such, whenever speech utterances were present in the dirty files, the ERLE calculation is halted before the speech. This approach was appropriate given that in most of the files, speech was not introduced until about 20 seconds in. This amount of time was expected to be more than enough for the AEC to converge.

4.3 Misalignment

$$M(n) = 20 \log \frac{\|\boldsymbol{h}(n) - \hat{\boldsymbol{h}}(n)\|_2}{\|\boldsymbol{h}(n\|_2} \tag{4.2}$$

The misalignment of an AEC is a measure related to the Euclidean distance between the vectors representing the unknown filter coefficients and the estimated filter coefficients. In this case, a normalized misalignment was used, which required the division by the norm of the unknown filter coefficients. To compute this metric, knowledge of the unknown system has to be available. The misalignment is a useful metric when

it is available; for example in cases where the microphone signal was produced by convolving with a known impulse response. As such, the misalignment of an AEC is a highly theoretical metric since these transfer functions are not always available in practice. Even so, misalignment can be a very useful metric in carefully analyzing the dynamics and details of the performance of the algorithms.

4.4 Convergence Rate

Convergence rate of an AEC is typically defined as the time it takes for an AEC system to reach its maximum echo attenuation. This metric was not explicitly calculated, but was apparent in the plots of the ERLE and Misalignment for every simulation. Visually, convergence appears in plots when the ERLE/Misalignment shows no more large changes. Depending on the application, achieving faster convergence rates may or may not be necessary. In the application investigated in this thesis, convergence rate was not as crucial because an AEC on a smart speaker typically has more than enough time to converge before a user asks the speaker anything.

Chapter 5

Single Channel Simulations

This section focuses on the single channel explorations of the Kalman filter, in comparison to the NLMS filter. These explorations were mainly to familiarize with echo cancellers and learn about different trade-offs associated with each filter. However, these explorations led to interested and unexpected results, which are analyzed and presented in this section. This chapter also contains an analysis of the NLMS and Kalman filters under different testing conditions, through various types of simulations. The particular setup and metrics used for each simulation are specified wherever relevant.

5.1 Full-Band

Unless noted, the following sets of simulations and results were generated using the room recordings setup described at the beginning of Chapter 3[1]. Each dirty file, on which the echo cancellation was run, consisted of a speech file convolved with a user-to-speaker transfer function. This output was then added to a recording of an appropriate music test track played through the smart speaker. The metric used for these initial simulations is the word error rate (WER). As explained in Chapter 3, the framework used to calculate WER results assumes access to the transcript of clean speech utterances throughout the dirty file and compares this against the result of the AEC processing.

NLMS

In the case of single channel NLMS, the main parameter varied for these simulations was the step size μ. Several step sizes were tested in the range of 0.01 - 1.2, but only the results of the best performing step sizes are included here. The freezing point chosen for these simulations was 13.1 seconds (well before speech at 20 seconds) and the estimated filter tap length is 900 taps at 16kHz (56 ms). The best performing step size was consistently 0.31. The filter with this step size achieved a WER of 0.375 on a file with 24 speech utterances.

Kalman

With the Kalman filter, a slightly different approach was taken. The main parameter varied in these simulations was the noise variance σ_w^2. However, σ_w^2 was not varied in the same way as the step size μ was varied in the NLMS. Instead of choosing a constant σ_w^2 and varying it per simulation/trial, an estimated version of this parameter was used. The results are shown below, but a detailed description of the estimated version of σ_w^2 can be found in Appendix C. The freezing point of these simulations was 12.5 seconds, and the tap length for the optimal Kalman case was 2000 taps at

[1]Section 3.1

16kHz (filter length of 125ms). This system achieved a WER of 0.20833 on the same file with 24 speech utterances. Figure 5-1 presents a portion of the simulation output error for the optimal NLMS and Kalman solutions.

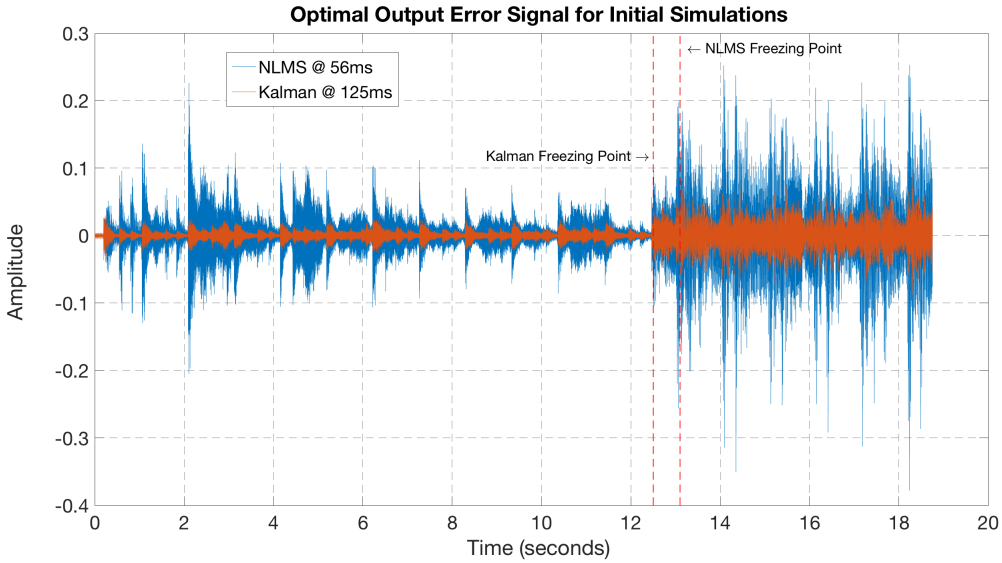


Figure 5-1: Optimal NLMS vs Kalman, with freezing point at 12.5 and 13.1 seconds.

5.1.1 Convergence Analysis

It is noted that these two simulation results were chosen due to their WER performance, with results described in the previous sections. It is also noted that an adaptation freezing point[2] was employed in the following simulations. This freezing point occurred at 12.5 seconds for Kalman and 13.1 seconds for NLMS and can be seen clearly in Figure 5-1 when the error signals increase. A loss in echo attenuation is expected when adaptation is frozen.

From Figure 5-1, the Kalman filter seems to have presented an advantage in reducing the error signal faster and more effectively than the NLMS counterpart. This advantage was confirmed when considering the Root Mean Square (RMS) of the error signals, shown in Figure 5-2. The RMS values represent the average power in the error signal, thus accounting for transient behavior.

[2]Section 3.3.1

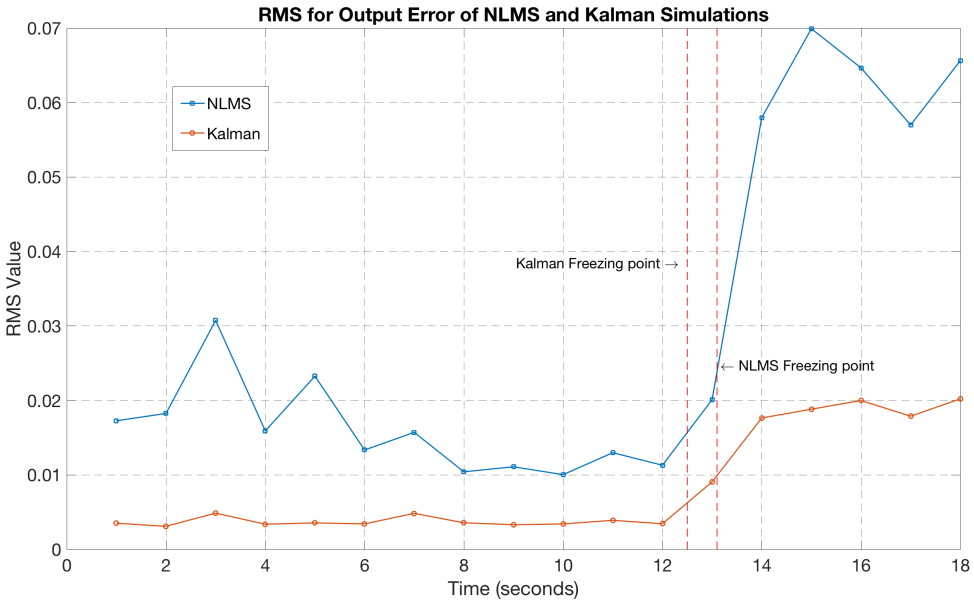


Figure 5-2: RMS Value Optimal NLMS vs Kalman. RMS values were calculated with a window of 1 second of samples. The Kalman filter RMS quickly converges and approaches its minimum value within 1 second. The NLMS RMS converges slowly and does not reach the same minimum as the Kalman filter before adaptation is frozen.

This result was not surprising since the Kalman solution is expected to have much faster convergence rates than the NLMS equivalent [7].

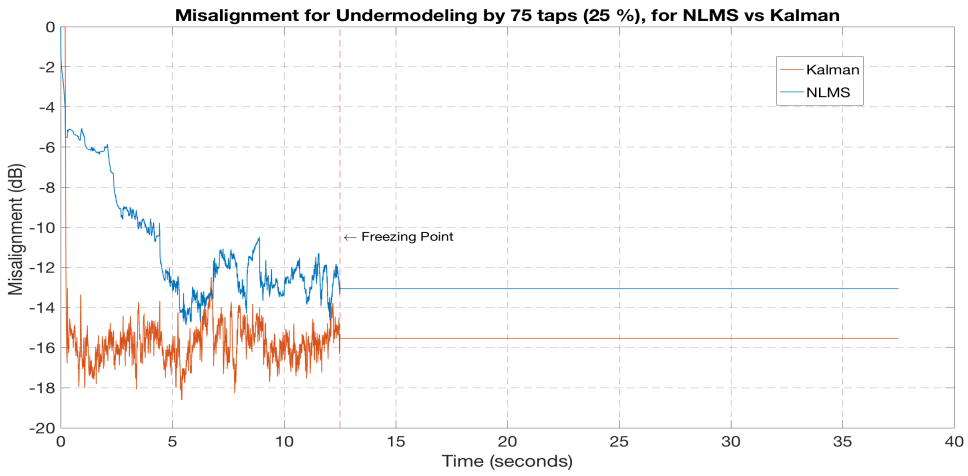
As a result of comparing simulations by WER, the filter lengths for the optimal NLMS and Kalman filters were not necessarily equal. At longer filter lengths, the NLMS filter performed worse than the Kalman filter in terms of WER. This result was unexpected because at shorter lengths, an adaptive filter models the acoustic impulse response poorly, leading to degradation in performance [8]. The poor performance of the NLMS at longer filter lengths was related to the point at which adaptation was frozen. For the NLMS filter, increasing its length also decreases it's convergence rate [6]. Thus, in the attempt to fully model the echo path by increasing the filter length, the NLMS filter showed a slowing convergence rate. Even though the final echo attenuation may have increased as a result of the longer filter, the freezing point at 13.1 seconds, coupled with the slower convergence, halted the improvement in echo reduction. Thus, the Kalman filter showed optimal performance and did not seem to suffer from this phenomena.

5.1.2 Under-modeling

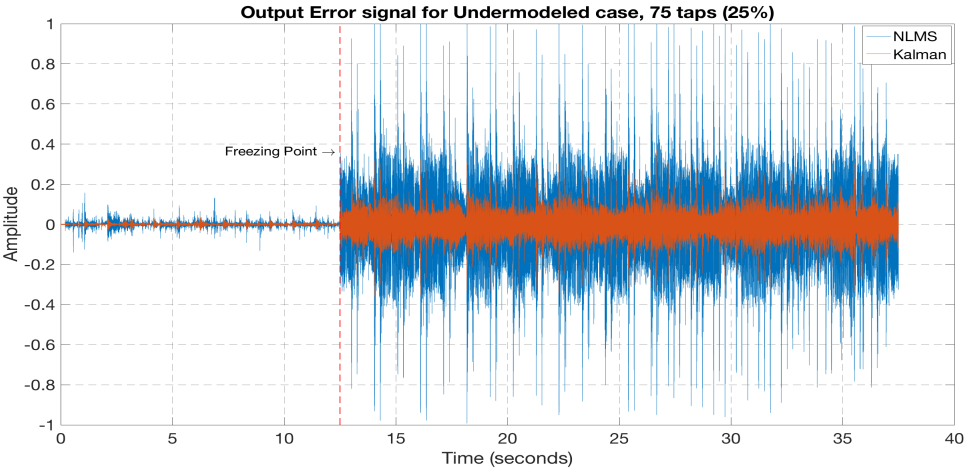
The notion of different degrees of modeling of the system introduces the concept of under-modeling. Under-modeling refers to the case where the estimated filter length is actually lower than the actual filter length. The under-modeling case occurs quite often in practice due to the computational burden of using a long filter length that more accurately matches an acoustic impulse response [8]. To explore this under-modeling case further, an important detail about the experimental setup had to change: full knowledge of the room impulse response. With this consideration, the experimental setup was switched to using known impulse responses[3]. With the actual filter coefficients, $\boldsymbol{h}(n)$, it was possible to use the misalignment metric[4] to evaluate the performance effect of different degrees of under-modeling. Plots of these experiments are shown in Figures 5-3 and 5-4. The total room response is 300 taps, and so under-modeling by x taps means that the estimated filter length is $300 - x$ taps.

.

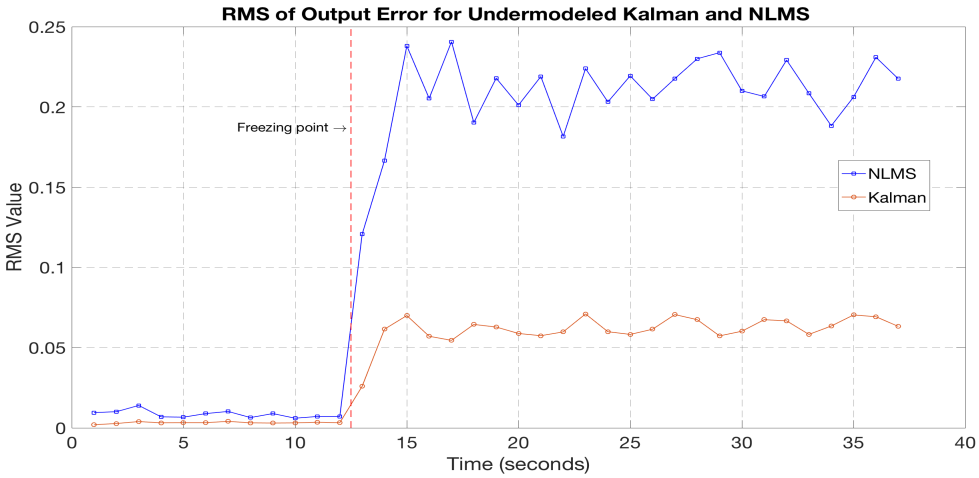
[3]Section 3.2
[4]Section 4.3



(a) Misalignment

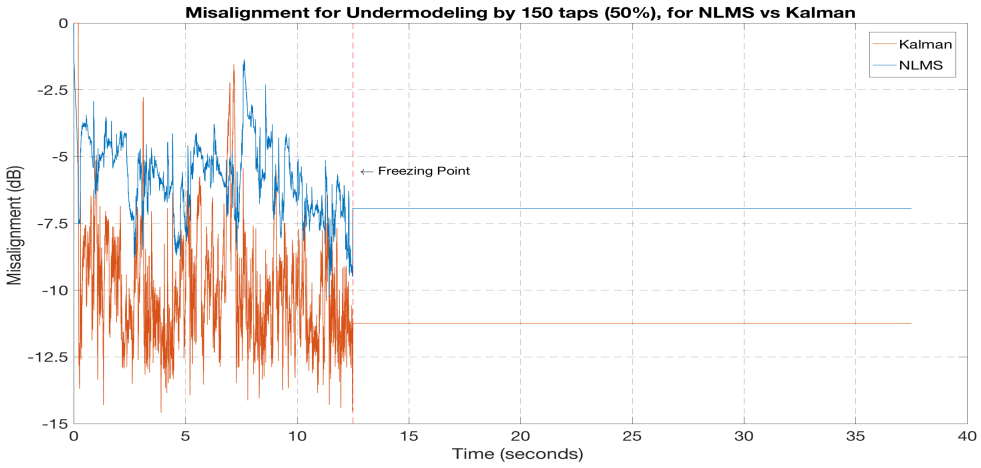


(b) Output Error

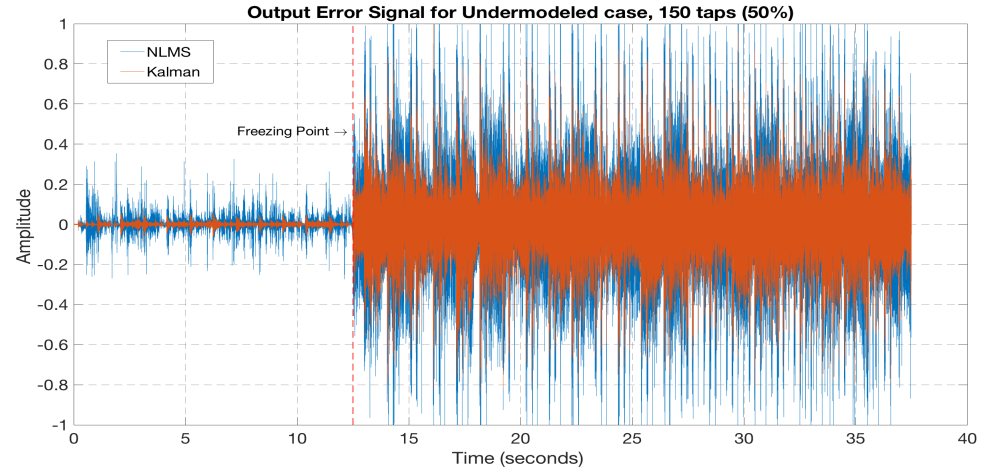


(c) RMS of Output Error

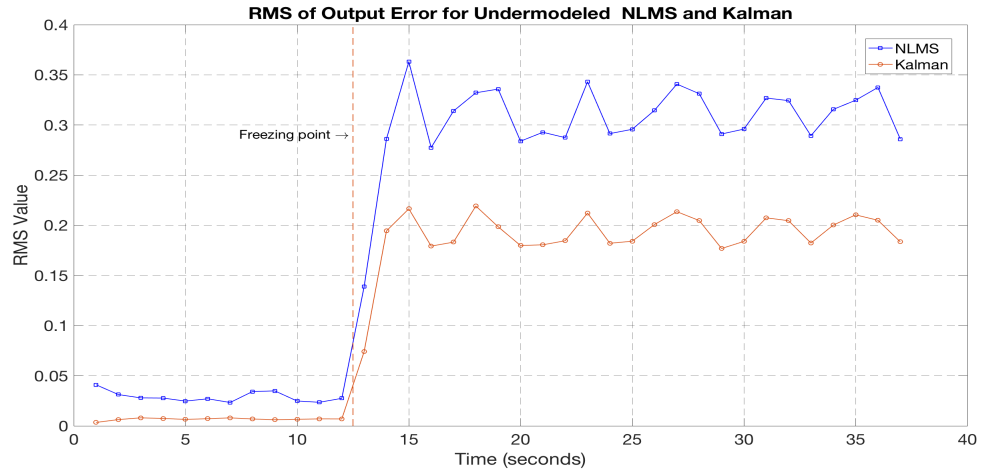
Figure 5-3: Under-modeling by 75 taps, freezing point at 12.5 seconds indicated on plots



(a) Misalignment



(b) Output Error



(c) RMS of Output Error

Figure 5-4: Under-modeling by 150 taps, freezing point at 12.5 seconds indicated on plots

For 25% under-modeling presented in Figure 5-3, the Kalman solution reached -16 dB misalignment at 0.25 seconds, but oscillated around this value until adaptation was frozen. The NLMS reached a misalignment of -14 dB at about 5 seconds before oscillating as well. This result indicates that the Kalman solution was superior in convergence speed and in estimating the acoustic impulse response. However, in Figures 5-3 (b) and (c), it should be noted that both solutions depict high amounts of reduction in the error signal, even with the oscillating misalignments (prior to freezing of adaptation). This reduction then decreased after the freezing point (around 12.5 s) for both the NLMS and Kalman. The Kalman froze at a lower misalignment (-15.5 dB) than the NLMS filter (-13 dB), indicating that the Kalman performed better in this experiment. This result was confirmed in the RMS[5] in Figure 5-3 (c) that shows the Kalman had a lower RMS value after freezing. It is noted that the RMS values for the error of both simulations increased by approximately an order of magnitude after adaptation was frozen. This finding indicates that the presence of oscillations in misalignment, prior to freezing, is effective in reducing the echo.

Similar results were obtained for the 50% under-modeling case presented in Figure 5-4. Most notably, the Kalman filter continued to outperform the NLMS in the reduction of error and both filters exhibited better performance prior to freezing. The difference in this second set of results is that the filters oscillated much more in their misalignment, demonstrated in Figure 5-4 (a). Even with these large oscillations, the error signals remained somewhat low, as shown in Figures 5-4 (b) and (c). As in the 25% under-modeling case, this phenomenon is likely a similar result of tracking statistical variations of the input. These results indicate that the degree of under-modeling affects the extent to which the NLMS/Kalman filters track input variations. The RMS comparisons in Figure 5-5 show how the NLMS/Kalman filters react differently to the change of under-modeling.

[5]Figures 5-3 (c) and F-4 (c) were created using a 1 second window of error signal samples

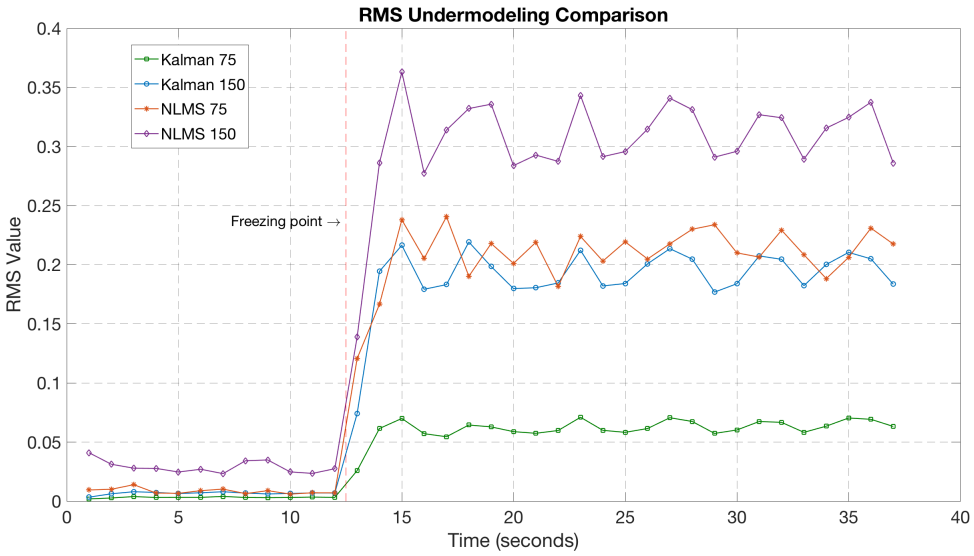


Figure 5-5: RMS Comparison Plot for Under-modeling Cases. Calulcated with 1 second window of samples. 75 and 150 refers to the number of taps by which the filter was under-modeled. 75 is the 25% under-modeling case, and 150 is the 50% under-modeled case.

When comparing the two filters' performance in different degrees of under-modeling, the Kalman filter was less affected by the increase in the degree of under-modeling from 25% to 50% than the NLMS filter. The Kalman filter saw an increase in error after freezing, from the 75 to 150 taps case, but maintained a low error signal, prior to freezing. This behavior is seen in the first 12 seconds of Figure 5-5, where the RMS values for the Kalman 75 and Kalman 150 case are similar. The NLMS 150 case, did see an increase in error, before and after freezing. This result indicates that the Kalman performs well in severe under-modeling cases. This is likely due to the Kalman filter's formulation for estimation of dynamic systems. This formulation may explain the success in the Kalman filter of tracking input music variations, in order to maintain the error signal low. The extent of this result is not explored further in this thesis, and is left for future work.

Results from the mathematical analysis for the under-modeled case are below. The full mathematical analysis can be found in Appendix C. This analysis takes into account how the under-modeling case, coupled with minimization of the mean square error, leads to a dependence on input statistics.

If

$$\boldsymbol{h} = [h_0, h_1, \cdots h_{L-1}]^T$$

is the actual filter state, and

$$\hat{\boldsymbol{h}}(n) = [\hat{h}_0(n), \hat{h}_1(n), \cdots \hat{h}_{M-1}(n)]^T$$

is the estimate, and if $L > M$, then minimization of the mean square error $E\{|e(n)|^2\}$ is not achieved by requiring that $\hat{h}_i = h_i$ for $i \leq M$. Instead, the solution that minimizes the mean square error depends on the correlations of the input signal $x(n)$, and thus relies on it's changing input statistics. For stationary signals like white gaussian noise, this is not the case, but it is clear that these signals are not used as inputs for the VPA application.

5.2 Sub-band

This section presents the results of single channel simulations, but in the sub-band domain. Sub-band refers to the use of perfect reconstruction filter banks that split input signals into different frequency bands. The processing (echo cancellation), then occurs within each sub-band, independent of the other sub-bands. Once the processing occurs in each sub-band, the results are summed together and synthesized into a full-band output. This process is shown generally in Figure 5-6 below.

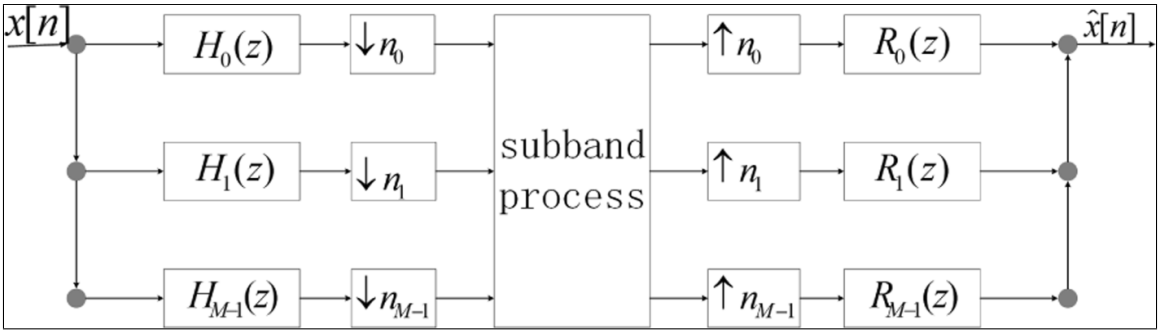


Figure 5-6: Sub-band Processing, with analysis filters shown as $H_i(z)$ and synthesis filters $R_i(z)$. $x[n]$ represents the input full-band signal, and $\hat{x}[n]$ is the processed and reconstructed output signal. [11]

5.2.1 General Principles

Advantages

There are two main characteristics that make sub-band processing attractive for this application.

· **Increased Convergence Rate** This can be attributed to the degree to which the sub-band processing splits the signal into several frequency bands. As the division gets higher (more sub-bands), the signal spectrum begins to resemble white noise within each band, and for algorithms such as the NLMS, this increases the convergence rate.

· **Computational Efficiency** There is an increase in computational efficiency by a decrease in the millions of instructions per second (MIPS) required in the

system. This relationship is due to the downsampling of the broadband input signal into the smaller sub-band signals. This applies to both the Kalman and NLMS filters. Also, with the downsampling of the input signal, it is possible to represent the same broadband length L filter response, with M, length $\frac{L}{M}$ responses, where M is the number of sub-bands. This means that sub-band processing allows one to use shorter filters. Sub-band processing then presents an important advantage for the Kalman filter, which boasts $O(n^3)$ complexity.

The only disadvantage of this approach is an increase in the delay of the processing pipeline due to the analysis and synthesis filters in place. This disadvantage is not overly crucial because the delay is never long enough to affect the response of the VPA-human interaction.

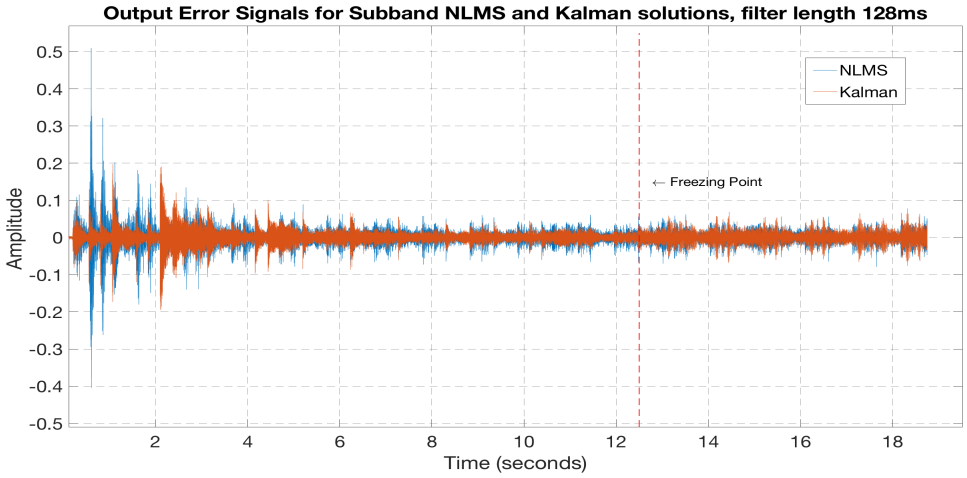
Implementation

A low-pass prototype filter was required to properly construct the perfect reconstruction filter banks. This prototype filter was modulated to different center frequencies for each band. In this case, the low-pass prototype filter had a stop-band attenuation of 80 dB and was used for 64 frequency bands. This design divides the total sampled frequency range into 33 real frequency bands.

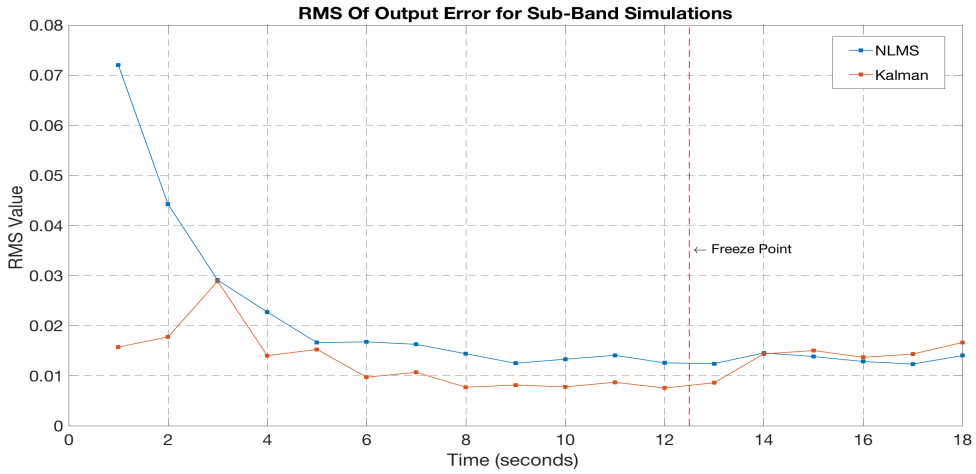
Given that the adaptive filters now work in the frequency domain (after sub-band processing), the complex versions of the equations listed for NLMS and Kalman are needed. The complex version is straightforward for the NLMS case and is presented in Appendix A. However, it is more complicated in the Kalman case. Review of existing literature revealed that there is a general complex Kalman filter as well as an augmented Kalman filter. The general complex version was used in this work and is summarized by simple analogs to the original equations in Appendix A. The augmented Kalman filter is possibly more accurate due to its full consideration of second order statistics [4, p. 1], but its use in the AEC context was not explored in this thesis.

5.2.2 Setup and Results

The experimental setup used for these simulations was that of the same full-band simulations in the previous section, which included the room recordings methodology. This setup included speech file utterances so the main metric used to distinguish performance is the WER. The outputs of the best performing sub-band NLMS and Kalman solutions are presented in Figure 5-7. This time, the filter length is 128ms, which corresponds to 64 filter taps within each of the sub-bands. This length was chosen to match the length of the best performing Kalman solution from the full-band case.



(a) Output Error



(b) RMS

Figure 5-7: Optimal Sub-band NLMS and Kalman Results

As seen in Figure 5-7, it is clear that the Kalman performance advantage in the AEC context was no longer as present when using sub-band processing. This result is due to the speed-up that the NLMS filter receives due to the "white" nature of the inputs to each separate sub-band NLMS filter. It is also noted that the freezing point no longer has much of an effect on the output error. The freezing point is still employed at 12.5 seconds and yet there is no significant increase in the output error or RMS for either filter. This behavior indicates that both filters were able to fully model the room response, and arrive at very accurate coefficients, rather than resort to tracking the input signal. Both of these solutions achieved nearly perfect WER, with the Kalman at 0.0416 and NLMS at 0.0833. The results of these simulations support the conclusion that the Kalman may not be as attractive a choice as the NLMS, in the single channel case, due to the simplicity and computational efficiency of the NLMS.

Chapter 6

Multi-Channel Simulations

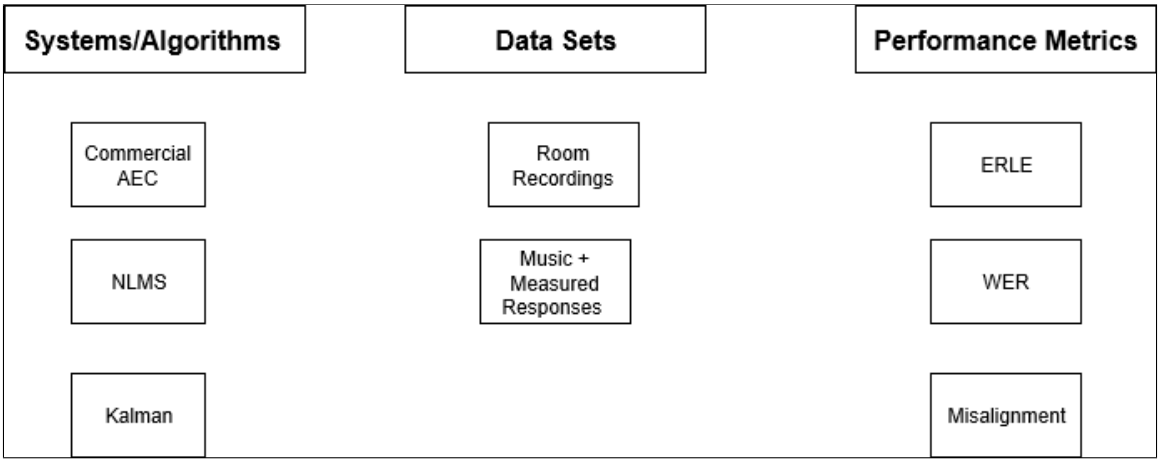


Figure 6-1: Main Testing Overview

This section is structured based on the testing overview presented in Figure 6-1. The inclusion of the room recordings data set was to have a representative and realistic multi-channel test set. The findings from these multi-channel simulations highlighted some very unexpected results. To explore these findings, another data set (Music + Measured responses) was introduced. It should be noted that, in some of the following simulations, a third-party benchmark was available in the form of a proprietary commercial AEC of unknown design and methodology.

6.1 Data Set 1 - Room Recordings

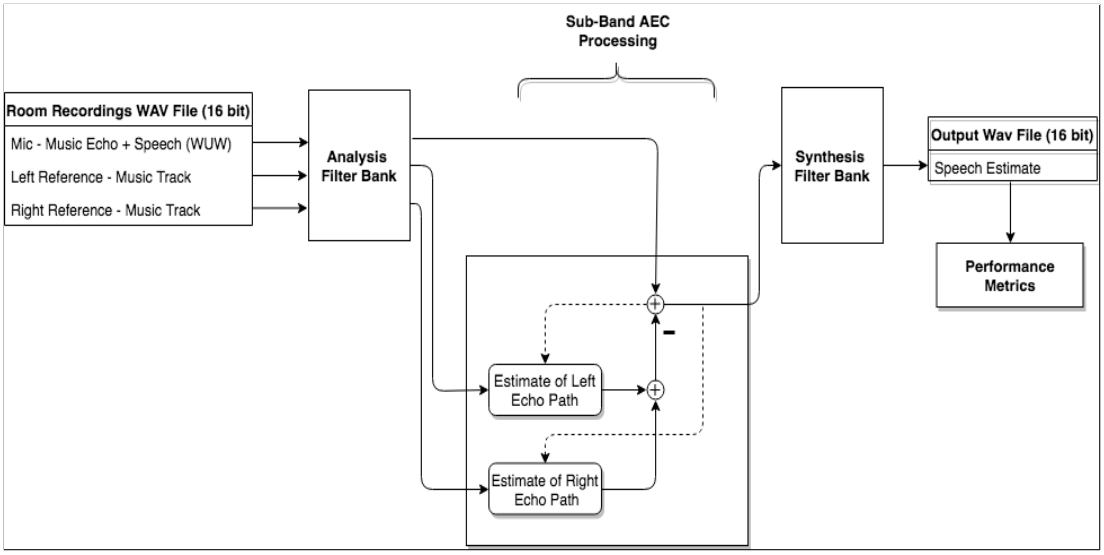
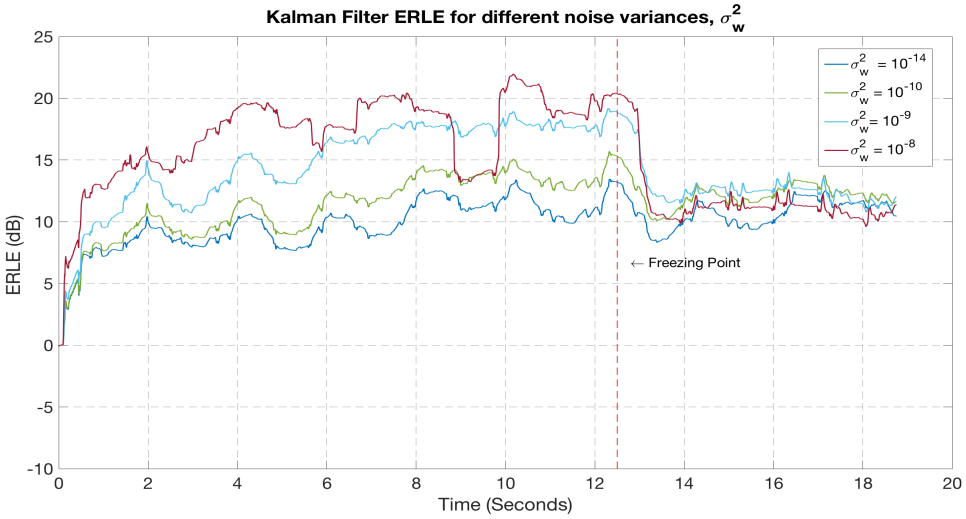


Figure 6-2: Multi-Channel AEC Sub-band Diagram. Inputs are audio WAV files, containing mic signals and reference channels. These inputs are converted to sub-band, as described in Section 5.2, processed by an AEC block per sub-band, and then synthesized to form the output speech estimate (error signal). Dashed lines refer to the output error being fed back to adapt the coefficients for left/right echo paths.

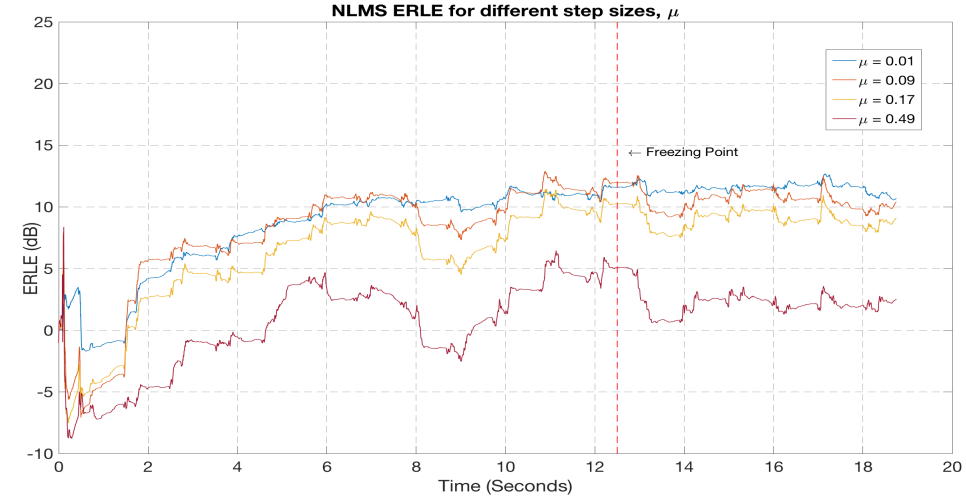
Figure 6-2 explicitly describes the setup for this section. This setup was also described in the single channel case, but a clear difference is the presence of two channels. It is noted that only sub-band simulations were considered in this chapter. The reasons for this are that sub-band echo cancellation presents a state of the art solution, due to advantages outlined in the previous chapter. Another important detail about this particular setup is that the wav-files that represent the dirty files are 16-bit integer precision. This system choice was for the purpose of providing a fair comparison to the commercial AEC in question. Additionaly, the freezing point of 12.5 seconds was present in the following simulations.

6.1.1 User-Design Parameters

The variation of user-designed parameters (μ for NLMS, σ_w^2 for Kalman), were tested for each simulation to see the effects in the multi-channel case. Rather than presenting the results of the best performing parameters, simulations with a variety of parameters are shown here to outline a comprehensive view of the results. First, the ERLE[1] is presented to outline some of these results.



(a) ERLE for Kalman



(b) ERLE for NLMS

Figure 6-3: ERLE for different user-designed parameters. Both Kalman and NLMS filters were 128 ms long.

[1]Section 4.2: Echo Return Loss Enhancement

It's also noted that a variety of signal to noise ratios (SNR's) were introduced in these simulations. This was achieved by choosing different playback volumes, with the speech signals maintained at the same volume. It was expected that changing the SNR would not affect the ERLE measurement because this metric is a ratio. Consequentially, if the input was louder, then the output would also be louder by the same amount. Indeed, changing the SNR did not affect the ERLE curves across different simulations. The trends seen for these initial simulations are as follows.

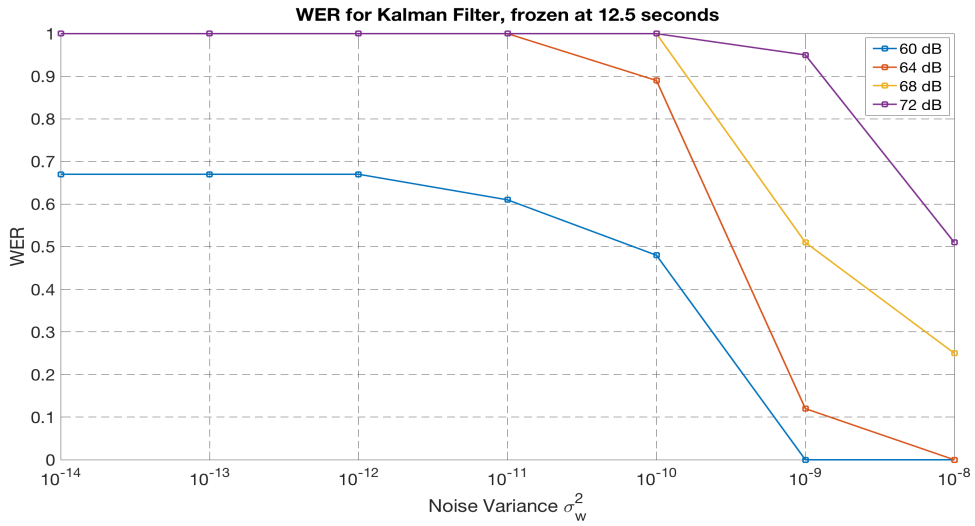
· **General Trends** For the Kalman filter, an increase in the noise variance parameter σ_w^2 generally causes an increase in the performance of the AEC, measured by the ERLE, before the freezing point. On the other hand, the NLMS seemed to exhibit the typical step size trade-off where higher step sizes μ resulted in higher steady state error (lower ERLE), but faster responses to input changes. There is some evidence of this with the best performing step sizes at 0.01 and 0.09 and a significant decrease in performance for any larger step sizes. Also, it is clear to see that the performance of the ERLE for the Kalman solution is initially much greater than that of the NLMS solution, but was not always the case.

· **Freezing Point** In the Kalman solution a freezing point is apparent with a clear drop off of ERLE, after 12.5 seconds. The NLMS however, does not have this clear distinction, even though a freezing point was also employed in the simulation. This different behavior is interesting because it would seem that the Kalman filter solution is tracking input statistics to achieve such good cancellation, as measured by the ERLE. The NLMS did not have this same behavior. After the freezing point, both solutions achieved similar ERLE in their best cases.

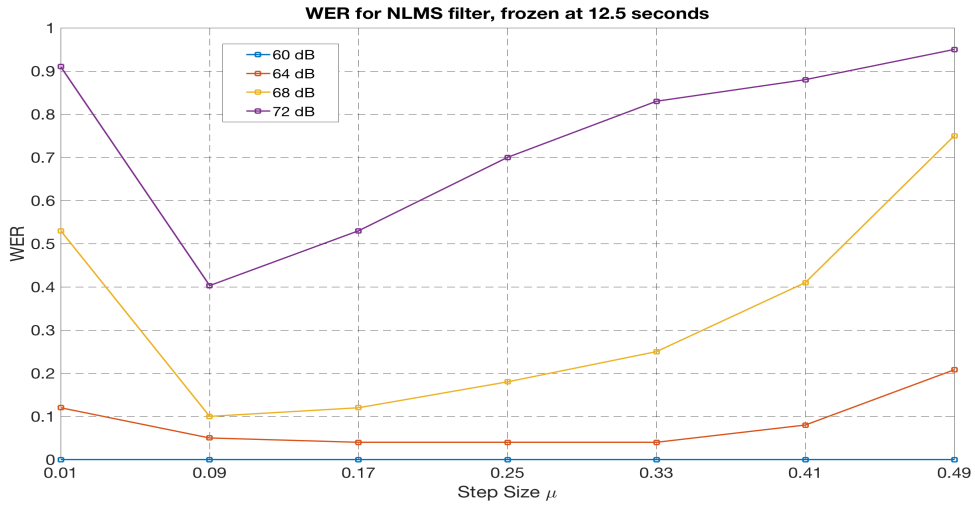
· **Convergence Rate** Convergence rate is difficult to analyze for these simulations because it is unclear when there is a steady state in the ERLE curves. For the Kalman case, it seems that convergence rate is higher due to its immediate increase in ERLE, in comparison with the convergence rate of the NLMS. This

observation is misleading however, due to the significant decrease in ERLE at the freezing point for the Kalman filter. The NLMS exhibited the typical convergence rate trade-off, given that smaller step size simulations slowly rise to a high ERLE and large step size simulations rapidly achieve low ERLE.

· **Divergence** Adaptive filters can easily exhibit instability and diverge during adaptation. In the case of the NLMS, this instability can occur due to a large step size that causes unbounded oscillations in the estimated filter taps. Divergence in the Kalman filter is typically due to numerical instabilities. The reason that Kalman filter simulations were not run at larger noise variances is that at $\sigma_w^2 > 10^{-8}$, the output error signal diverged in those cases. It is noted that even at $\sigma_w^2 = 10^{-8}$, the Kalman filter became more sensitive to input changes. This observation is noted in Figure 6-3(a), where the $\sigma_w^2 = 10^{-8}$ ERLE dips drastically at 9 seconds, in comparison to the other ERLE curves.



(a) Kalman Word Error Rate



(b) NLMS Word Error Rate

Figure 6-4: WER at different user-designed parameters and playback volumes

These simulations were also evaluated in terms of the WER results, which are presented in Figure 6-4. It is noted that these WER results are included from simulations that were conducted at different Signal to Noise Ratios (SNR). The signal in this case is the speech signal, and the noise is music played through the speaker. The reasoning behind including these additional results was to analyze trends from a variety of tests, although it was expected that speech intelligibility will be lower (higher WER) for higher playback volumes. Figure 6-4 above presents this trend through different colored curves, with the color indicating the playback volume.

· **General Trends: Kalman** In Figure 6-4 (a), it is clear that the WER results agree with the initial ERLE results in Figure 6-3 (a). An interesting observation is that the results do not agree with the ERLE results after the freezing point. After the freezing point, the ERLE curves appear to be around the same level, so it is surprising that the two highest σ_w^2 values would have such a dramatic increase in the WER performance. This effect is likely due to the use of sub-band processing in these particular simulations. This refers to the fact that the Kalman filter AEC in these simulations is a collection of individual Kalman filter AECs that worked independently per sub-band of the full-band signal. The ERLE shown above is of the full-band signal, but does not show which frequency bands contribute to this ERLE. As a result, even though the full-band ERLE is similar for all noise variances after the freezing point, there was more effective cancellation in critical[2] speech frequency bands for higher noise variances.

· **General Trends: NLMS** The WER performance for the NLMS filter is somewhat matched to the ERLE performance in Figure 6-3 (b), except for two cases. For $\mu = 0.01$ and $\mu = 0.09$, the ERLE curves appear to indicate that the lower step size is performing better than the higher one. In the WER plots, however, it is noted that $\mu = 0.09$ achieved a much lower WER. This is indicative of the same phenomenon seen in the Kalman case - better cancellation in critical frequency bands.

· **Comparison** The optimal performance of the NLMS and the Kalman solutions was compared at a reasonable playback volume of 68 dB. For comparison, the NLMS actually achieved better cancellation with a WER of 0.1, than the Kalman filter with a WER of about 0.25. The commercial AEC, at the same playback volume, achieved a WER of 0.208333.

[2]Speech frequencies reside predominately in the 100 Hz - 6 kHz range

WER Results for Freezing Point Variation		
Algorithm	WER Freezing @ 4 sec	WER Freezing @ 12.5 sec
NLMS	0.45	0.083
Kalman	0.833	0.25

Table 6.1: WER Results

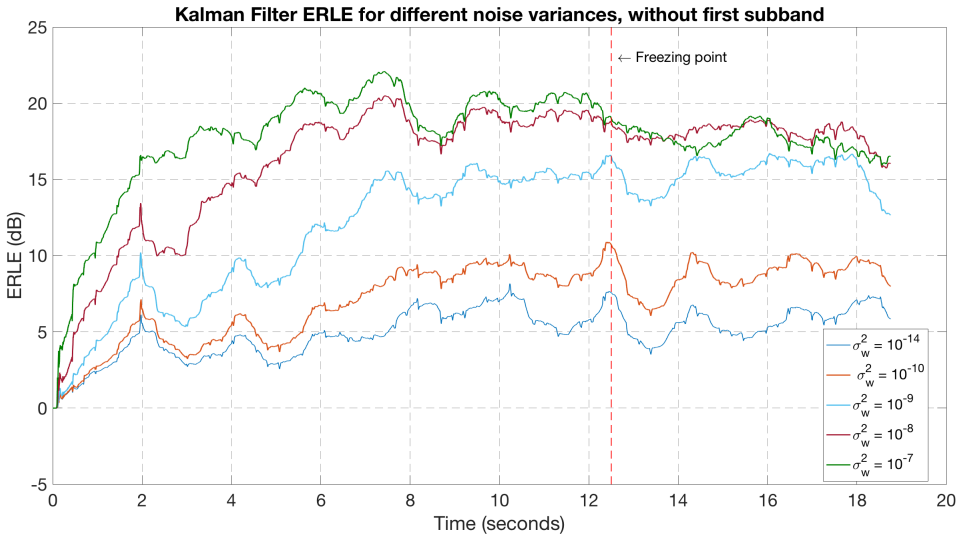
6.1.2 Freezing Point Variation

Due to the misleading nature of the initial ERLE curves for the Kalman filter, some more tests were conducted to determine the convergence rates of the two algorithms. This test was accomplished by moving the freezing point to a time earlier in the simulation. The justification for this was based on the assumption that if algorithms stop adapting at an earlier point, whichever one has converged faster will have better WER. As presented in Table 6.1 it would seem that the NLMS filter is still converging at faster rates than the Kalman filter, in this multi-channel scenario. Given that this test case presented unexpected results, the next sections explore more theoretical tests with metrics such as the Misalignment[3].

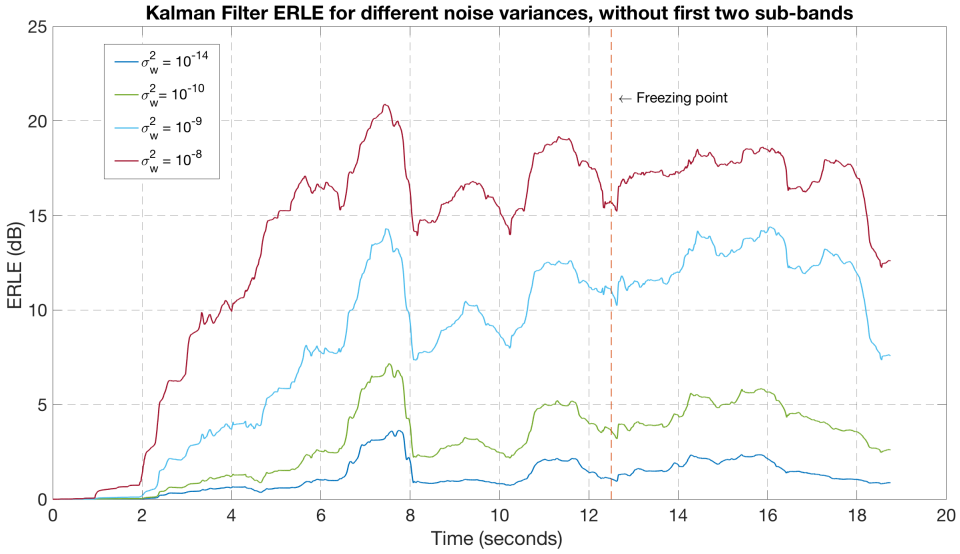
6.1.3 Sub-banding Effects

As noted previously, the discrepancy of the ERLE and WER results may have been due to the misrepresentation of ERLE cancellation in critical frequency bands. Along with this hypothesis and a suspicion of high amounts of uncorrelated low frequency noise, tests were run to demonstrate the effects on the ERLE without the first two sub-bands. With a total frequency band decomposition of 64 bands and 16 kHz sampling rate, each band consisted of a range of 250 frequencies. For example, the first band is centered at 0 Hz (DC), and goes up to 125 Hz. The second band is centered at 250 Hz, and extends until 375 Hz, and so on. This experiment would effectively get rid of frequencies up to 375 Hz, thus focusing on the ERLE in higher frequency bands. The results of these experiments are presented in Figures 6-5 and 6-6.

[3]Section 4.3

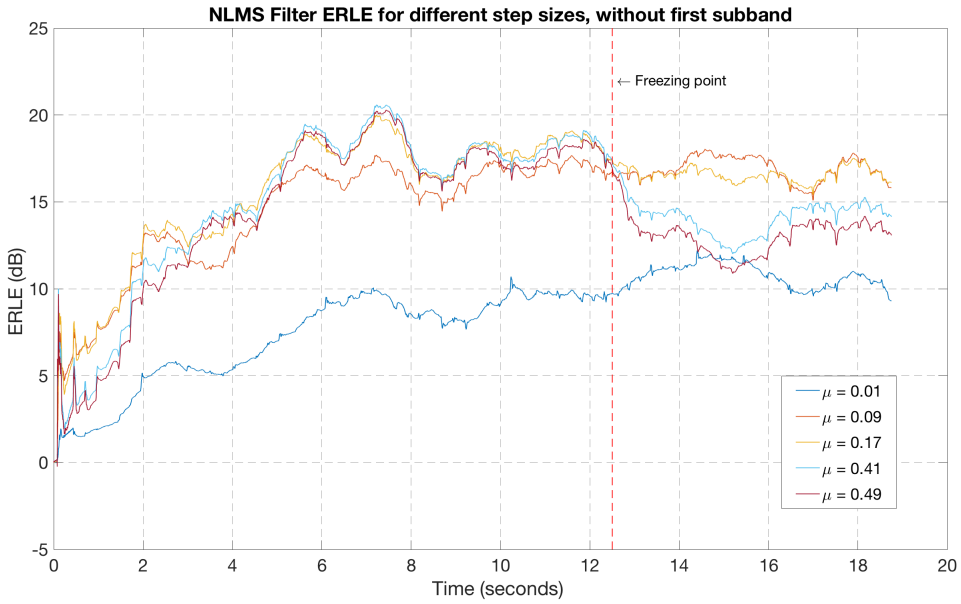


(a) Kalman without 1st band

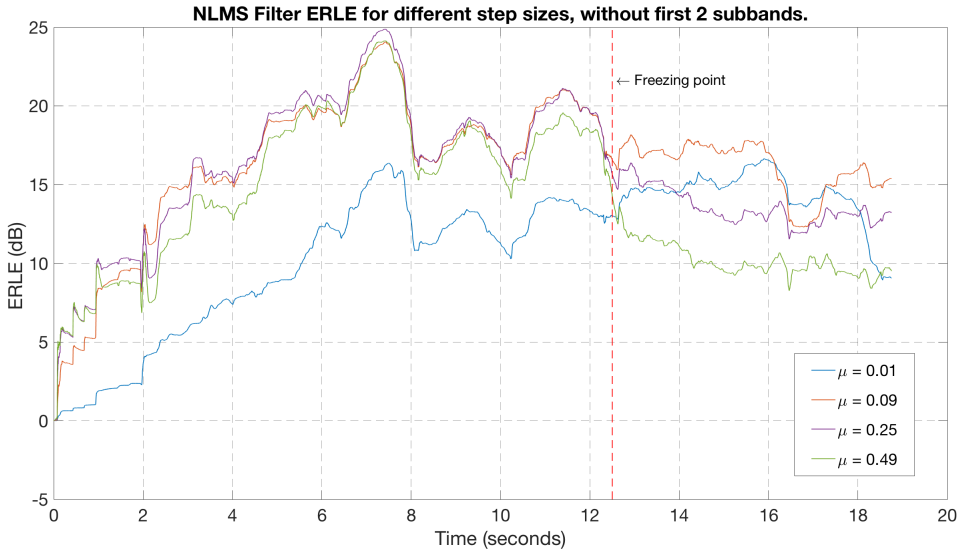


(b) Kalman without first 2 bands

Figure 6-5: Kalman ERLE, without 1 or 2 sub-bands



(a) NLMS without 1st band



(b) NLMS without first 2 bands

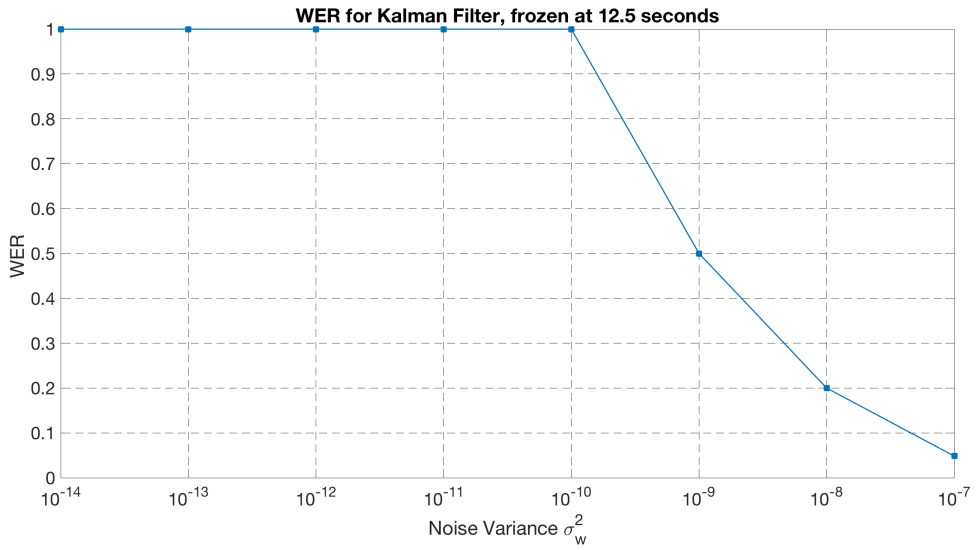
Figure 6-6: NLMS ERLE, without 1 or 2 sub-bands

It is first noted that the Kalman ERLE curves in Figure 6-5 agree with the WER results previously shown in Figure 6-4 (a). For example, higher noise variances σ_w^2 lead to much higher ERLE results in these critical bands, and thus lead to good speech intelligibility and WER results. The NLMS case agreed with previous WER results initially, except for the case of $\mu = 0.01$ and $\mu = 0.09$. Figure 6-6 (a) and

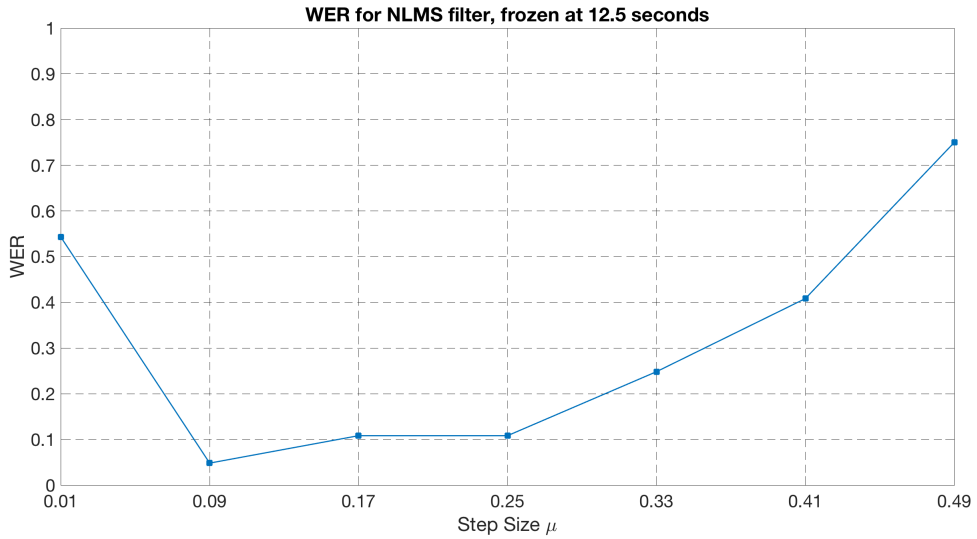
(b) demonstrate that $\mu = 0.09$ should lead to much better WER results than with $\mu = 0.01$, due to higher cancellation in critical frequency bands. The significance of these results is that the lower two sub-bands provide the same echo cancellation across several simulations. In comparison, the energy in higher bands (> 375 Hz) , is much lower, and so cancellation differences in these bands across several simulations are not as well noted, when considered in full-band. However, when the two lower sub-bands are not considered, these differences are more readily seen through ERLE curves, as shown in Figure 6-7 and 6-6.

In the particular case of the Kalman filter, it is also interesting to note that the freezing point is no longer obvious in Figure 6-5 (a) and (b). In contrast, the full-band ERLE curves presented in Figure 6-3 (a) demonstrated a clear drop in ERLE at higher noise variances, after the freezing point. This result confirms that the cancellation in the lower two sub-bands, prior to a freezing point, is due to the Kalman's ability to track the dynamic nature of the input signal. Once the freezing point is reached, this tracking stops, and the cancellation in the two lower bands becomes the same across most noise variances.

For completeness, the WER results for the simulations without the 1st sub-band are included below in Figure 6-7. It is interesting to note that removing the first sub-band allowed the Kalman filter to run at a noise variance of $\sigma_w^2 = 10^{-7}$, without diverging. With this change, the Kalman WER was able to match the optimal NLMS WER performance. The WER results for the second sub-band case are not shown here, but this case proved to be worse than with all sub-bands.



(a) Kalman, without 1st band



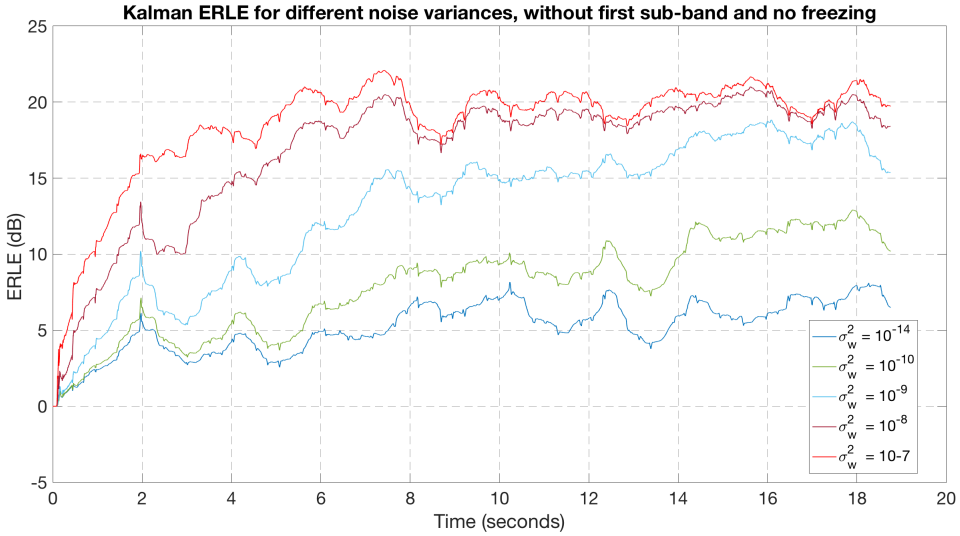
(b) NLMS, without 1st band

Figure 6-7: WER Results, without 1st sub-band

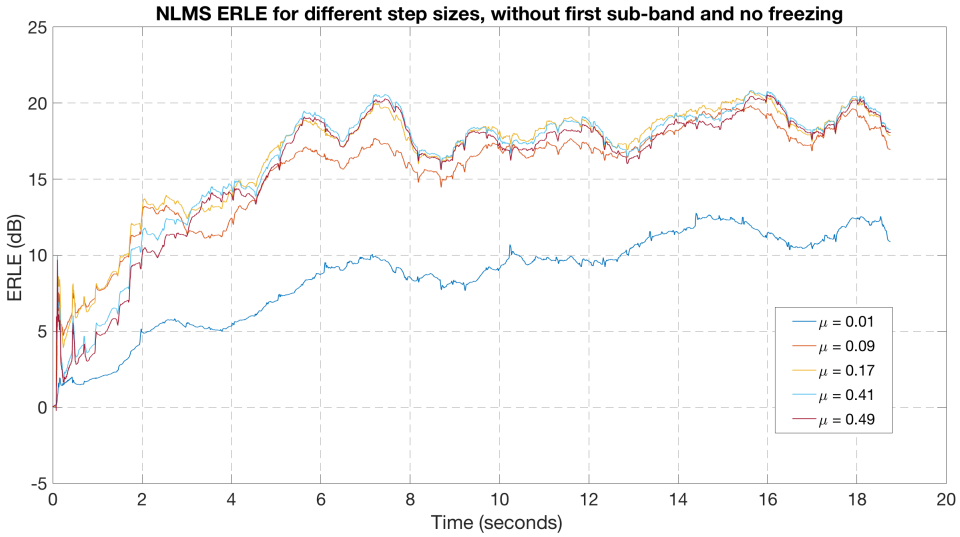
6.1.4 No-Freeze Simulations

Due to the effective echo cancellation that both the NLMS and Kalman filters exhibited prior to freezing of adaptation, simulations were run to analyze the effect of omitting a freezing point entirely. In practice, the lack of a freezing point would mimic an AEC without the use of a double talk detector. It is noted that these results were produced without considering the first sub-band. The reason for this

decision was that the ERLE results could be misleading due to tracking in the lower bands. For the Kalman filter, this also allowed the use of a higher noise variance to be used without divergence. The ERLE and WER results for this experiment are shown below in Figures 6-8 and 6-9.

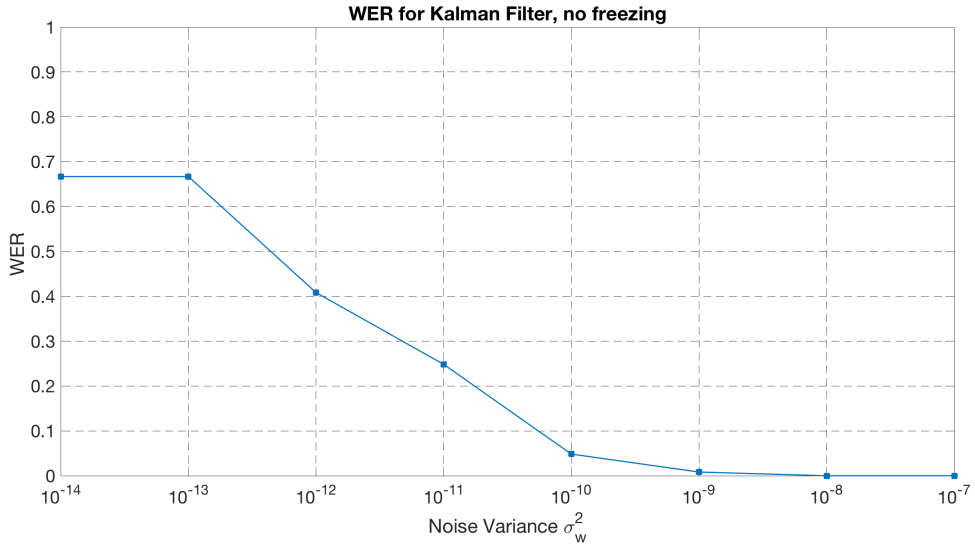


(a) Kalman without freezing

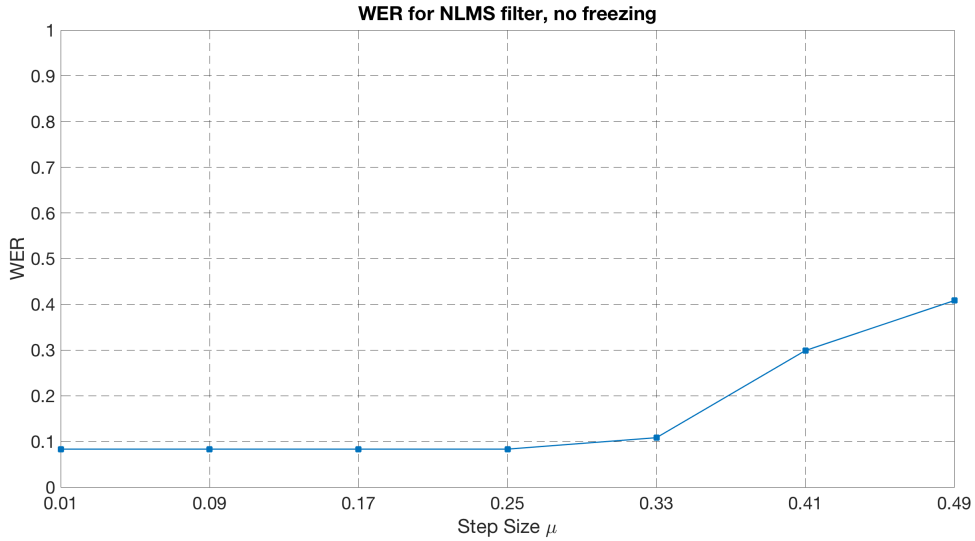


(b) NLMS without freezing

Figure 6-8: No Freeze ERLE Results



(a) Kalman without freezing



(b) NLMS without freezing

Figure 6-9: No Freeze WER Results

The ERLE plots in Figure 6-8 (a) show that the Kalman filter achieves high ERLE rapidly (20 dB at 6 seconds) in the case of $\sigma_w^2 = 10^{-7}$. Adaptation appears to stop improving well before speech is introduced for the higher σ_w^2, but is slower for lower σ_w^2. The WER results in Figure 6-9 (a) show significant improvement from the previous freezing case in Figure 6-7 (a). This observation is expected because the filter has more time to continue adapting for the entire length of the speech file. This advantage relies heavily on the condition that speech does not disrupt adaptation significantly.

The NLMS, however, suffered from the lack of a freezing point as seen through the WER results presented in Figure 6-9 (b). The ERLE results in Figure 6-8 (b) also do not agree with the WER results in Figure 6-9 (b). The ERLE performance seems to be the same for $\mu = 0.25, 0.33, 0.41, 0.49$, which suggests similar WER results across these values. However, in Figure 6-9 (b), it is clear that speech intelligibility suffered as the step size increased to $\mu = 0.41, 0.49$. A possible explanation for this behavior is that speech may have been disrupting adaptation at the reactive, higher step sizes. As a result, this could have lead to distortion in the speech signal and thus less speech intelligibility.

6.2 Data Set 2 - Music + Known IR

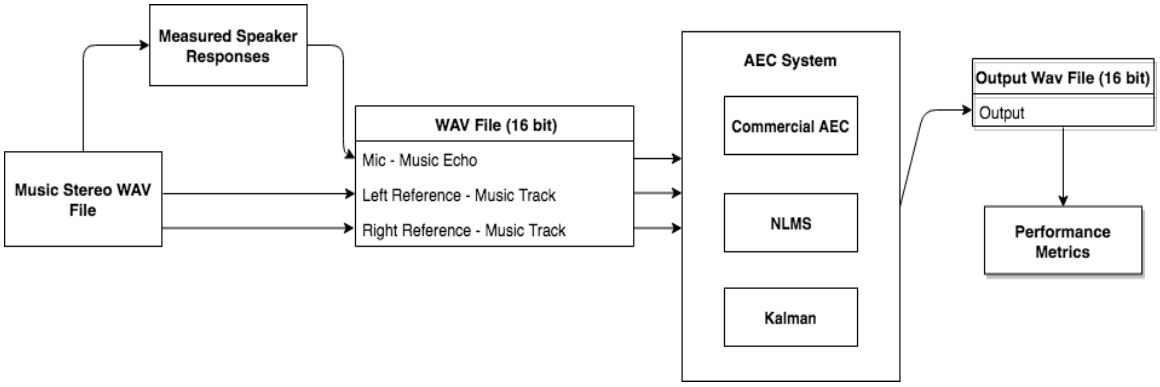
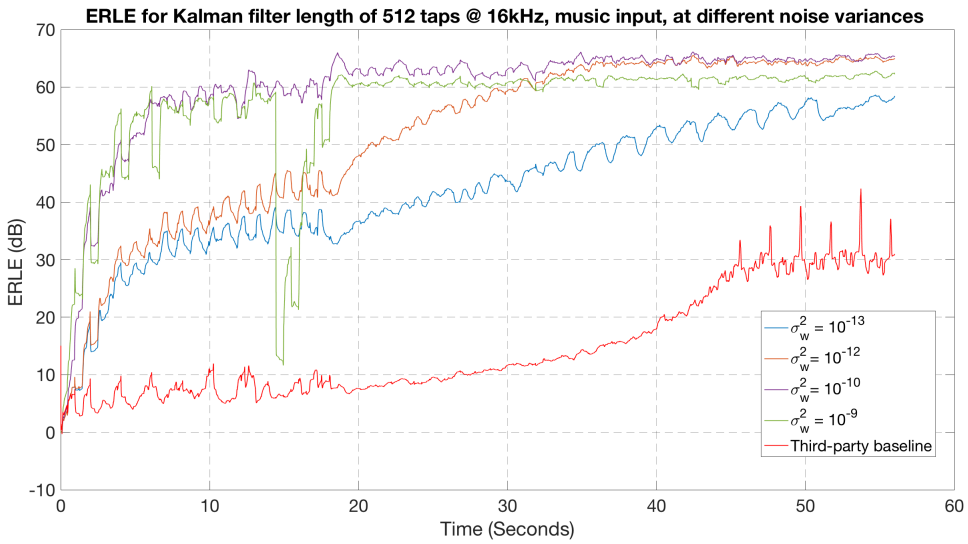


Figure 6-10: Data Set 2 System Diagram

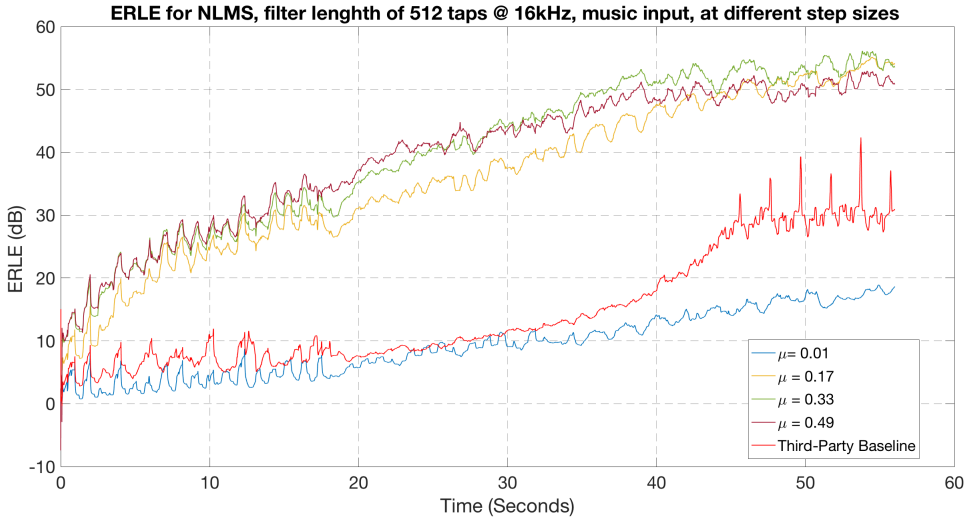
Figure 6-10 above presents the experimental setup of this section, which focuses on simpler experiments with measured speaker responses (both left and right). The reasoning for this focus was to closely analyze the convergence behavior of these algorithms in multi-channel scenarios. These simulations do not include speech utterances in each dirty file, so the WER metric is not available. It is noted that these simulations were performed using full-band processing rather than sub-band processing in the previous section. This choice was based on the assumption that results in the full-band domain would only improve in sub-band domains.

6.2.1 User-Design Parameters

As before, the focus was varying the two main parameters, μ for NLMS and σ_w^2 for Kalman, and observing the effects on performance of the respective AEC solutions. In this case, a measured speaker response that is 512 taps long @ 16 kHz was used in conjunction with the typical stereo music wav-file. A freezing point was not employed in this experiment for two reasons. The first is that there is no speech from which to guard adaptation from. The second is that the misalignment metric is enough to determine in what direction the convergence of the AEC is headed. Figures 6-11 and 6-12 summarize these results for both solutions.

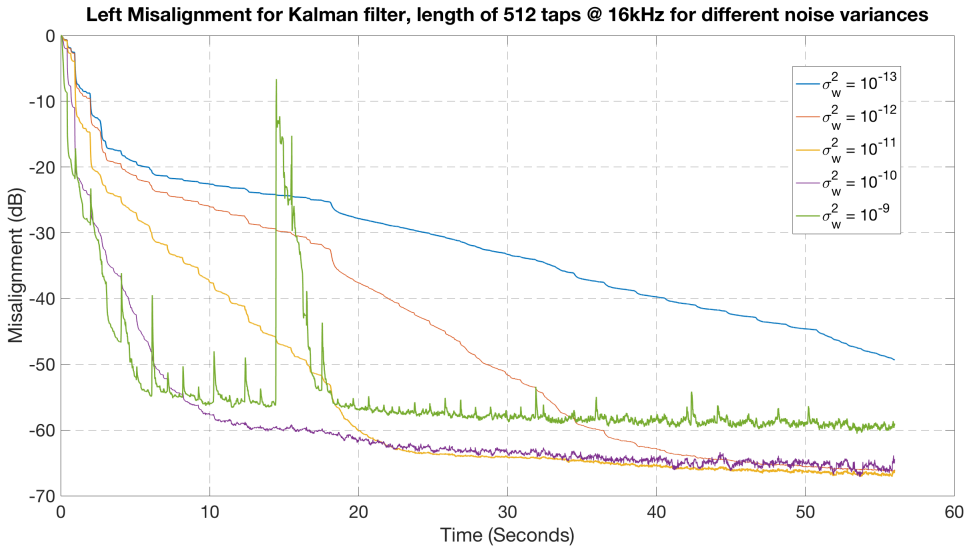


(a) Kalman ERLE

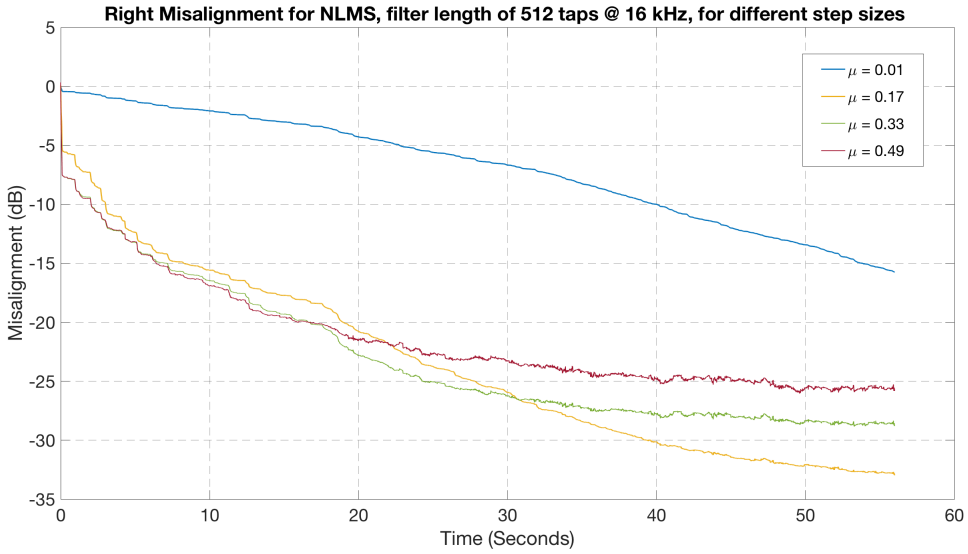


(b) NLMS ERLE

Figure 6-11: ERLE Curves, Kalman filter reaches good cancellation as measured by 60 dB ERLE, in as little as 5 seconds. NLMS takes full simulation time to reach similar ERLE values.



(a) Kalman Misalignment



(b) NLMS Misalignment

Figure 6-12: Misalignment Curves, Kalman filter reaches good cancellation as measured by -60 dB misalignment, in as little as 10 seconds. NLMS does not reach this low of misalignment and takes 40 seconds to reach the lowest misalignment at around -30 dB.

Initially, these simulations seem to suggest different conclusions from those of the first data set[4], but these discrepancies can be explained in the context of full-band processing. From Figures 6-12 and 6-11 it is clear that the Kalman filter had faster convergence rates than the NLMS in these experiments. This finding contradicts

[4]Room Recordings

earlier results from data set 1 where the NLMS had faster convergence rates as measured by better WER performance, in simulations with an earlier freezing point. As noted previously, the NLMS experiences a significant speed-up in convergence rates with the use of sub-band processing. Therefore, the slower convergence rate for the NLMS in this case is due to the use of full-band processing. It is also interesting to note that for both solutions, the typical parameter trade-off is observed: a higher parameter (μ or σ_w^2) increased the convergence rate, but also increased steady state error. This trade-off is readily seen from both the Misalignment and ERLE plots of both algorithms in Figures 6-11 and 6-12.

Another interesting point to note is that in this case, the performance of both the NLMS and Kalman filter far surpassed that of the commercial third-party plugin. It is difficult to determine exactly what the reasons for this difference in performance are because the commercial AEC is presented as a black box with few parameters to vary. The ease with which both solutions converge in this multi-channel scenario is also extremely apparent. In the Sondhi paper [10], this scenario is outlined as a significant issue for convergence of multi-channel echo cancelers. However, the Kalman and NLMS solutions converge easily and reach satisfactory levels of ERLE and Misalignment. This observation highlights further investigation into the reasons for this, and if the multi-channel issue is as relevant for this particular application as initially assumed.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The work presented in this thesis suggests that, in comparison to the NLMS filter, the Kalman filter is not a viable AEC option for the VPA application in the majority of cases. There were some experimental results which suggest that the Kalman filter is an attractive choice for a specialized version of this application, but these results were generally outweighed by the instances that prove the NLMS filter to be a better solution.

7.1.1 Single Channel

The single channel simulations, in the full-band case, proved largely in favour of the Kalman filter solution, due to its rapid convergence speed. In the sub-band case, however, the NLMS experienced a significant improvement in it's convergence rates, and reached similar WER results as the Kalman filter. This would seem to put the NLMS at an advantage, due to the fact that it is more computationally efficient and simple to implement than the Kalman filter. However, the phenomenon of under-modeling presents a case for the Kalman filter. It was observed that the Kalman filter was more capable of tracking input statistics than the NLMS, in order to maintain the error low. This result would appear to be a disadvantage, except in the case

of foregoing an adaptation freezing point. Given that the presence of such a large, uncorrelated signal could cause filter divergence, the absence of a freezing point would seem counter-intuitive. However, the wake-up word speech burst is short, meaning that the speech may not necessarily disrupt adaptation long enough for the filter to begin adapting incorrectly or even begin diverging. Thus, the Kalman filter would be a clear choice for this application, due to the fact that it converges to a lower error very quickly and maintains this lower error by effectively tracking the input signal statistics.

7.1.2 Multi-Channel

The results of Data Set 1 showed that both filters provide similar ERLE and WER performance, but the NLMS had the added benefit of being computationally efficient and simple to implement. This advantage is significant, given that this data set was the most realistic representation of an actual user-case (speech utterances, playback volumes, etc). Additionally, in Data Set 1 the Kalman filter presented itself as an attractive tracking alternative, due to its high ERLE performance, when the lower two sub-bands (< 375 Hz) were considered. In these cases, the Kalman filter was able to track the uncorrelated, low frequency rumble, and maintain this performance until the freezing point. Not only was it well suited for this tracking, but it did so in a very fast manner, bringing the echo down within seconds. The VPA application does not require such fast convergence speeds, but it is something to note for a future adaptive filter application that can withstand not freezing or employing a double-talk detector. Data Set 2 again demonstrated the Kalman filter speed advantage in the full-band case. This advantage is likely not as present in the sub-band, due to the NLMS filter's improvement under sub-band conditions.

Along with this advantage presented by the Kalman filter, a significant disadvantage was the filters propensity to diverge easily. The Kalman filter became extremely reactive to changes in the input statistic at higher noise variances, as seen in Figures 6-11 (a) and 6-12 (a) with sudden dips in ERLE and increases in Misalignment. This was also seen in the Data Set 1 simulations where the noise variance could not be

increased past 10^{-8} before divergence occurred.

7.1.3 Future Work

This research led to many unexpected results, although not all of these were explored in their entirety and are included in this section as future work.

· **Third-Party Comparison** A significantly unexpected conclusion from this work was that, in some cases, both algorithms outperformed the third-party commercial solution. The reasons for this result are still unknown, but a possibility is that this commercial solution traded faster convergence for a much lower steady-state error and also for stability. It is difficult to predict what kind of effect this trade-off would have in practice. More data collection on realistic situations using the third-party commercial solution would help in order to form a more definitive conclusion.

· **Kalman Filter Stability** The divergence issue that the Kalman filter experienced is also somewhat worrisome due to the fact that it is not clear as to why the filter became so reactive in the multi-channel case. Simpler system cases might help in clearing this issue up. For example, one could test the same Kalman system with uncorrelated white inputs that were colored[1] in part of the experiment. This test would then determine if it was an unequal power spectrum that caused the divergence to occur. Also, a filter sweep on this white input could highlight if it was the rate at which input statistics change that affected the divergence. It is also likely that this divergence issue was an artifact of the numerical instability of the Kalman filter, as a result of ill-conditioned matrices. If this reasoning were to be the case, further numerical simulation would be needed in order to test the stability of the Kalman filter under these conditions.

· **Estimated User-Designed Parameters** In this work, most of the user-designed parameters were held constant for each simulation and varied across different

[1]Processed to resemble other forms of noise with unequal power spectra

simulations. Several papers in the AEC realm suggest using some form of adapted system parameter that depends on the system inputs and outputs instead. This was not implemented in this work due to the scope of the thesis and also since there is extensive literature on these estimated parameter AECs.

· **Sub-Band Optimizations** The sub-banding effects in Data Set 1 provide interesting insights into the way that convergence in individual frequency bands can affect the full-band output signal. Some possible explorations in this area would be to employ an estimated parameter much like the VSS (variable step size) NLMS. If such a system was used and adapted per individual frequency band, then results could be better optimized per frequency band. This would likely have the effect of increasing the overall performance in the full-band output error. Another possible optimization, more specific to the Kalman filter, is to update the covariance matrices (through the noise variance), according to the power in each frequency band.

· **Multi-Channel Simplification** With most of the multi-channel simulations, it is surprising to see that such good cancellation occurred in a reasonable amount of time. This finding was unexpected because the configuration of these systems should give rise to the non-uniqueness issue, in which the filters continually face convergence issues due to varying input statistics. The lack of this issue was highlighted with Data Set 2, which was a multi-channel system that was generated by measured smart speaker responses. Even so, the misalignment curves demonstrated that the filter coefficients converged to good estimates without having to re-converge many times. This begs the question of whether the non-uniqueness issue is as much of a problem in this application, and why that might be. This is possibly due to a fundamental difference between the VPA application and the teleconferencing case in which the multi-channel issue is described [10]. Particularly, the proximity of the microhphone on the speaker to both the left and right output channels may have a simplifying effect on the analysis of the multi-channel case. This question is left as an exercise to be

verified in a scope beyond the purpose of this thesis.

Appendix A

Adaptive Filter Derivations

A.1 Normalized Least Mean Squares

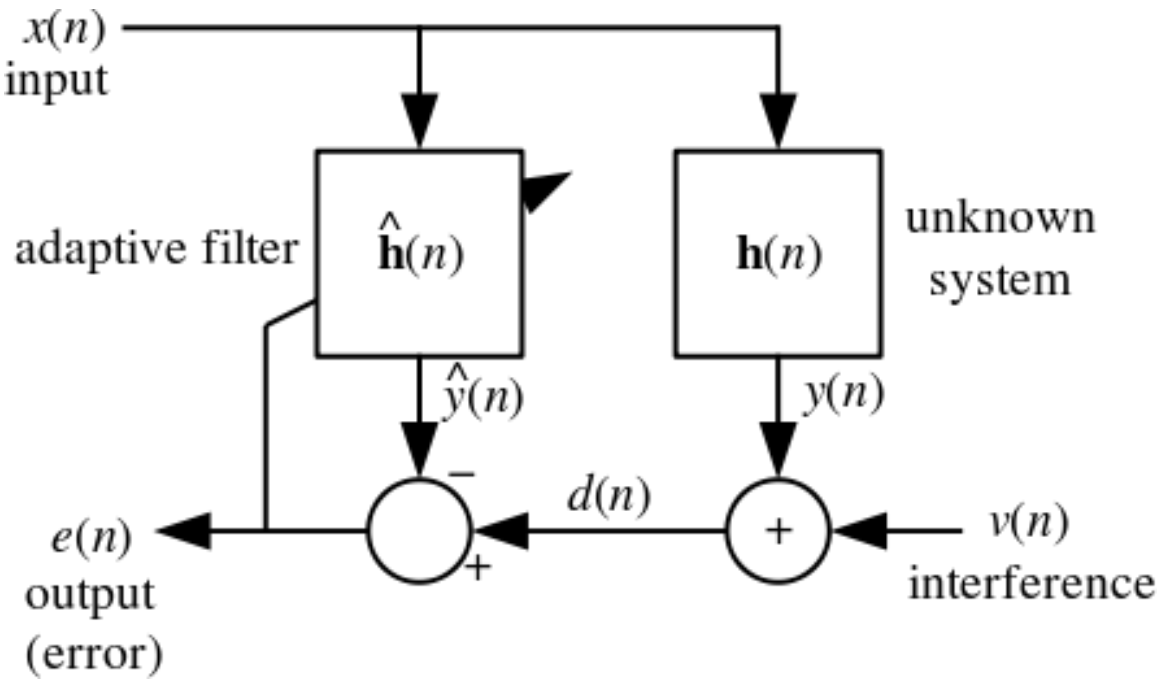


Figure A-1: NLMS Filter [5]

To derive the NLMS filter, we first begin with the LMS filter, and then normalize accordingly. To begin with the LMS filter, we need to define the cost function. As the name suggests, this is chosen as

$$C(n) = E\{|e(n)|^2\} \tag{A.1}$$

where this cost function is referred to as the mean square error. We then apply steepest descent and take partial derivatives to get the gradient of this cost function.

$$\nabla_{\hat{\boldsymbol{h}}^H} C(n) = \nabla_{\hat{\boldsymbol{h}}^H} E\{e(n)e^*(n)\} = 2E\{\nabla_{\hat{\boldsymbol{h}}^H}(e(n))e^*(n)\} \tag{A.2}$$

$$\nabla_{\hat{\boldsymbol{h}}^H}(e(n)) = \nabla_{\hat{\boldsymbol{h}}^H}(d(n) - \hat{\boldsymbol{h}}^H \cdot \boldsymbol{x}(n)) = -\boldsymbol{x}(n) \tag{A.3}$$

Thus,

$$\nabla_{\hat{\boldsymbol{h}}^H} C(n) = 2E\{\boldsymbol{x}(n)e^*(n)\} \tag{A.4}$$

The method of steepest descent then requires us to correct the filter coefficient estimates at each iteration, by the negative of this gradient.

$$\hat{\boldsymbol{h}}(n+1) = \hat{\boldsymbol{h}}(n) - \frac{\mu}{2}\nabla_{\hat{\boldsymbol{h}}^H} C(n) \tag{A.5}$$

where μ is chosen as the step size and determines how big of a step in the direction of the gradient we take.

The main simplification, and the reason that the LMS filter (and thus NLMS), is so attractive, is in it's estimation of the gradient function, where

$$\nabla_{\hat{\boldsymbol{h}}^H} C(n) = 2E\{\boldsymbol{x}(n)e^*(n)\} \approx \boldsymbol{x}(n)e^*(n) \tag{A.6}$$

and so the final update equation for the LMS filter is

$$\hat{\boldsymbol{h}}(n+1) = \hat{\boldsymbol{h}}(n) - \mu \boldsymbol{x}(n) e^*(n) \tag{A.7}$$

The NLMS filter follows naturally in normalizing by the input power signal.

$$\hat{\boldsymbol{h}}(n+1) = \hat{\boldsymbol{h}}(n) + \mu \frac{e(n)\boldsymbol{x}(n)}{\boldsymbol{x}^T \boldsymbol{x}} \tag{A.8}$$

A.2 Kalman Filter Derivation for Echo Cancellation

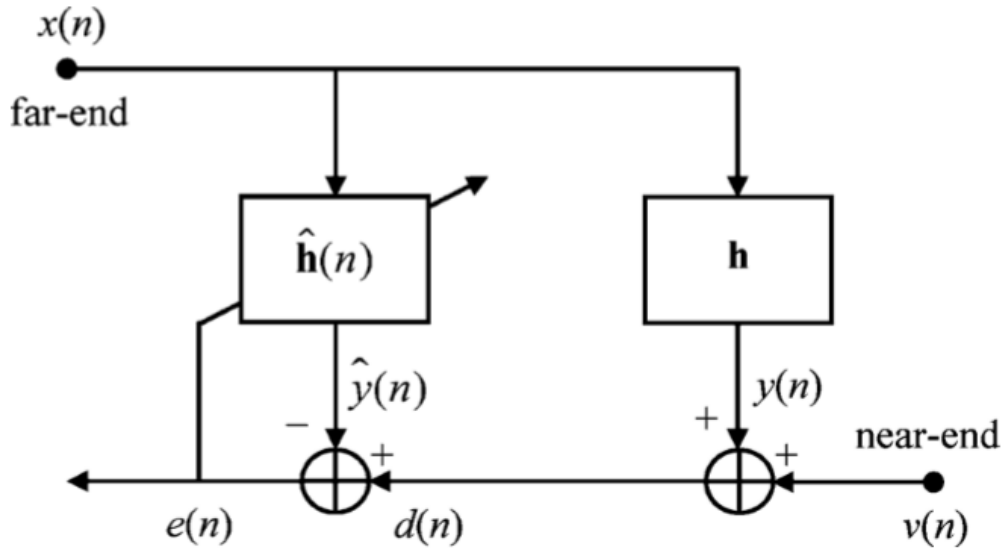


Figure A-2: General AEC [7]

To derive the filter, we must introduce relevant variables and definitions, using the diagram above.

$$d(n) = \boldsymbol{x}^T(n)\boldsymbol{h} + v(n) = y(n) + v(n) \tag{A.9}$$

$$\boldsymbol{x}(n) = [x(n), x(n-1), \cdots x(n-L+1)]^T \tag{A.10}$$

where L is the length of input samples taken in each iteration. The objective is to design a filter that uses a state variable model to update estimated filter coefficients $\hat{\boldsymbol{h}}(n) = [\hat{h}_0(n), \hat{h}_1(n), \cdots \hat{h}_{L-1}(n)]^T$, that estimates $\boldsymbol{h}$.

The state variable model is as follows.

$$\boldsymbol{d}(n) = [d(n), d(n-1), \cdots, d(n-P+1)] \tag{A.11}$$

$$= \boldsymbol{y}(n) + \boldsymbol{v}(n) \tag{A.12}$$

$$\boldsymbol{y}(n) = \boldsymbol{X}^T(n)\boldsymbol{h}(n) \tag{A.13}$$

$$\boldsymbol{X}(n) = [\boldsymbol{x}(n), \boldsymbol{x}(n-1), \cdots \boldsymbol{x}(n-P+1)] \tag{A.14}$$

where P is the model order. This essentially means how many sets of input vectors are considered at a single iteration. The key part of the Kalman filter comes from the next assumption.

$$\boldsymbol{h}(n) = \boldsymbol{h}(n-1) + \boldsymbol{w}(n) \tag{A.15}$$

This assumes that the unknown filter coefficients are not time-invariant, but instead vary from iteration to iteration by some random white Gaussian noise vector with variance $\sigma_w^2(n)$.
The update equation for the Kalman filter is given as follows,

$$\hat{\boldsymbol{h}}(n) = \hat{\boldsymbol{h}}(n-1) + \boldsymbol{K}(n)\boldsymbol{e}(n) \tag{A.16}$$

where $\boldsymbol{K}(n)$ is the Kalman gain, and $\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{X}^T(n)\hat{\boldsymbol{h}}(n-1)$

To derive the Kalman gain, we consider first the a posteriori misalignment or,

$$\boldsymbol{\mu}(n) = \boldsymbol{h}(n) - \hat{\boldsymbol{h}}(n) \tag{A.17}$$

It's correlation matrix is $\boldsymbol{R}_{\mu}(n) = E\big[\boldsymbol{\mu}(n)\boldsymbol{\mu}^T(n)\big]$, and to continue we also define the a priori misalignment, or

$$\boldsymbol{m}(n) = \boldsymbol{h}(n) - \hat{\boldsymbol{h}}(n-1) = \boldsymbol{\mu}(n-1) + \boldsymbol{w}(n) \tag{A.18}$$

where we have made use of equation A.17, to respresent this in terms of the a posteriori misalignment. With this, the correlation matrix of this is

$$\boldsymbol{R}_m(n) = E\big[\boldsymbol{m}(n)\boldsymbol{m}^T(n)\big] \tag{A.19}$$

$$= \boldsymbol{R}_{\mu}(n-1) + \sigma_w^2(n)\boldsymbol{I}_L \tag{A.20}$$

The minimization criteria that leads to the Kalman gain expression is

$$J(n) = \frac{1}{L} tr\big[\boldsymbol{R}_{\mu}(n)\big] \tag{A.21}$$

Thus, to minimize this, we get

$$\boldsymbol{K}(n) = \boldsymbol{R}_m(n)\boldsymbol{X}(n)\big[\boldsymbol{X}^T(n)\boldsymbol{R}_m(n)\boldsymbol{X}(n) + \sigma_v^2\boldsymbol{I}_P\big]^{-1}$$

With this, we are able to formulate the full set of Kalman filter equations, presented in the order that they are updated during each iteration.

$$\boldsymbol{R}_m(n) = \boldsymbol{R}_\mu(n-1) + \sigma_w^2(n)\boldsymbol{I}_L \tag{A.22}$$

$$\boldsymbol{R}_e(n) = \boldsymbol{X}^T(n)\boldsymbol{R}_m(n)\boldsymbol{X}(n) + \sigma_v^2(n)\boldsymbol{I}_P \tag{A.23}$$

$$\boldsymbol{K}(n) = \boldsymbol{R}_m(n)\boldsymbol{X}(n)\boldsymbol{R}_e^{-1}(n) \tag{A.24}$$

$$\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{X}^T(n)\hat{\boldsymbol{h}}(n-1) \tag{A.25}$$

$$\hat{\boldsymbol{h}}(n) = \hat{\boldsymbol{h}}(n-1) + \boldsymbol{K}(n)\boldsymbol{e}(n) \tag{A.26}$$

$$\boldsymbol{R}_\mu(n) = [\boldsymbol{I}_L - \boldsymbol{K}(n)\boldsymbol{X}^T(n)]\boldsymbol{R}_m(n) \tag{A.27}$$

The initialization of the filter involves choosing $\hat{\boldsymbol{h}}(0)$ and $\boldsymbol{R}_\mu(0) = \epsilon I_L$, where ϵ is some small positive constant.

A.3 General Complex Kalman Filter

The general complex Kalman filter has simple modifications to the General Kalman filter, as shown below.

$$\boldsymbol{R}_m(n) = \boldsymbol{R}_\mu(n-1) + \sigma_w^2(n)\boldsymbol{I}_L \tag{A.28}$$

$$\boldsymbol{R}_e(n) = \boldsymbol{X}^H(n)\boldsymbol{R}_m(n)\boldsymbol{X}(n) + \sigma_v^2(n)\boldsymbol{I}_P \tag{A.29}$$

$$\boldsymbol{K}(n) = \boldsymbol{R}_m(n)\boldsymbol{X}(n)\boldsymbol{R}_e^{-1}(n) \tag{A.30}$$

$$\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{X}^T(n)\hat{\boldsymbol{h}}^*(n-1) \tag{A.31}$$

$$\hat{\boldsymbol{h}}(n) = \hat{\boldsymbol{h}}(n-1) + \boldsymbol{K}(n)\boldsymbol{e}^*(n) \tag{A.32}$$

$$\boldsymbol{R}_\mu(n) = [\boldsymbol{I}_L - \boldsymbol{K}(n)\boldsymbol{X}^H(n)]\boldsymbol{R}_m(n) \tag{A.33}$$

Appendix B

Source Code

B.1 NLMS

```
function [mean_e, Lmisalignment, Rmisalignment] =
    NLMS_music_complex_MC(LREF,RREF,input,filt_length,order,
    step,reg,K,freeze,mis,hleft,hright)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%NLMS Filter
% This code takes in two reference signals (LREF/RREF) and a
    mic input
% signal and runs the affine project adaptive algorithm to
    estimate the
% room response. The algorithm is a generalization of the
    NLMS algorithm,
% which results if the order of the algorithm is 1. In this
    case, it is, so
% the code runs the NLMS algorithm. Other input parameters
    are the lenght
```

```
% of the estimated filter, the step size, (reg)ularization parameter, and
% the freezing point of the simulation.
%
%
%initialization and variables
M = filt_length;                    %filter length
N = order;                          %filter order - how far back we look at data - 1 for NLMS
time = length(LREF);                %length of simulation
delt = reg;                         %regularization parameter
L_coeffs = zeros(M,1);              %filter coefficients
R_coeffs = zeros(M,1);
L_prev_coeffs = zeros(M,1);
R_prev_coeffs = zeros(M,1);
error = zeros(N,1);                 %error vector
mean_e = zeros(1,time);
Lmisalignment = zeros(1,time);      %misalignment vector for left
Rmisalignment = zeros(1,time);      %misaglienment vector for right
L_A = zeros(M,N);                   %matrix with history of past inputs
R_A = zeros(M,N);
L_sigma_e = 0;                      %observation noise variance - ignore most of these as it is not really used here
R_sigma_e = 0;
sigma_d = 0;
R_sigma_d = 0;
L_sigma_y = 0;
```

```
35 R_sigma_y = 0;
36 B = 1 - 1/(K*M);
37
38
39 %algorithm
40 for i = 1:time
41
42     %input stuff - take in mic signal (d_cur)
43     if i < N
44         d_cur = [fliplr(input(1:i)) zeros(1,(N-i))];
45     else
46         d_cur = fliplr(input(i- N + 1:i));
47     end
48
49     %sigma_d = B*sigma_d + (1 - B)*(conj(d_cur(1))*d_cur(1));
50
51     %Left reference
52     %shift in amount of samples corresponding to the filter
         length
53
54     if i < M
55         L_in = [fliplr(LREF(1:i)) zeros(1,(M-i))];
56     else
57         L_in = fliplr(LREF(i- M + 1:i));
58     end
59    L_A = [L_in.' L_A(1:M,1:N-1)];                    %update
        matrix with newest left input vector - just a vector
        for NLMS
60
61    L_y = (L_coeffs')*L_in.';
```

```
62      L_prev_coeffs = L_coeffs;
63      L_sigma_y = B*L_sigma_y + (1 - B)*(conj(L_y)*L_y);
64
65
66      %Right reference input
67      %shift in amount of samples corresponding to the filter
            length
68       if i < M
69           R_in = [fliplr(RREF(1:i)) zeros(1,(M-i))];
70       else
71           R_in = fliplr(RREF(i-M+1:i));
72       end
73      R_A = [R_in.' R_A(1:M,1:N-1)];                    %update
            matrix with newest input vector - just the vector for
            NLMS
74
75      R_y = (R_coeffs')*R_in.';
76      R_prev_coeffs = R_coeffs;                         %save
            previous coefficients
77      %R_sigma_y = B*R_sigma_y + (1 - B)*(conj(R_y)*R_y);
78
79
80      %calculate the error using left and right reference
81      error = d_cur.' - ((L_coeffs')*L_A + (R_coeffs')*R_A);
                  %undisturbed error signal
82  %     L_sigma_e = B*L_sigma_e + (1 - B)*(conj(error(1))*error
        (1));
83  %     R_sigma_e = B*R_sigma_e + (1 - B)*(conj(error(1))*error
        (1));
84
```

```
85
86      %calculate current mu - left (mu encompasses the step size
            and the input multiplication)
87      if i < M
88          L_mu = step*L_A/((L_A')*L_A + delt*eye(N));
89      else
90          L_mu = step*L_A/((L_A')*L_A + delt*eye(N));
91      end
92
93      %calculate current mu - right
94          if i < M
95          R_mu = step*R_A/((R_A')*R_A + delt*eye(N));
96      else
97          R_mu = step*R_A/((R_A')*R_A + delt*eye(N));
98          end
99
100
101     %update left and right coefficients
102     if i < freeze
103           L_coeffs = L_coeffs + L_mu*(error');              %L_mu
                and R_mu encompass the step size and the input of
                the update equations
104           R_coeffs = R_coeffs + R_mu*(error');
105     else
106           L_coeffs = L_prev_coeffs;                         %
                freeze the coefficients
107           R_coeffs = R_prev_coeffs;
108     end
109
```

```
110   %calculate current misalignment and current mean error -
          just the value
111   %for NLMS so error = scalar
112    mean_e(i) = error(1);
113
114
115    if mis == 1
116        Lmis = norm((hleft - L_coeffs(1:length(hleft))));
117        normL = norm(hleft);
118        Rmis = norm((hright - R_coeffs(1:length(hright))));
119        Lmisalignment(i) = 20*log10(Lmis/normL);
120        Rmisalignment(i) = 20*log10((Rmis)/(norm(hright)));
121    end
122
123
124
125
126
127
128 end
```

B.2 GKF

```
1  function [mean_e,Lmisalignment,Rmisalignment] =
       GKF_music_est_nov_complex_MC(LREF,RREF,input,f_length,
       order,reg,K,flag,s_w,freeze,mis,hleft,hright)
2
3
4  %Setup of parameters
5  f_length = f_length;                    %adaptive filter
```

```
    length
order = order;                          %"lookback" order -
    how many block of samples we look in the past
e = 0.0000001;                          %small positive
    constant for some initializations -used in older
    simuliations
Lsigma_w = s_w;                         %process noise
    variance
Rsigma_w = s_w;
len = length(LREF);                     %simulation length
L_coeff_hist = complex(zeros(f_length,800000));
Rcoeff_hist = complex(zeros(f_length,800000));
L_prev_coeffs = complex(zeros(f_length,1));
R_prev_coeffs = complex(zeros(f_length,1));


%Setup of variables
error = complex(zeros(order,1));                    %error
    signal
mean_e = complex(zeros(1,len));
Lmisalignment = zeros(1,len);
Rmisalignment = zeros(1,len);

%Initializaion of vectors/matrices
L_coeffs = complex(zeros(f_length,1));              %
    adaptive filter coeffs - **in column vector form**
Rcoeffs = complex(zeros(f_length,1));
R_uL = complex(e*eye(f_length));                    %
    posteriori misalignment autocorrelation matrix
R_uR = complex(e*eye(f_length));
```

```
28 R_mL = complex(zeros(f_length,f_length));              %priori
       misalignment autocorrelation matrix
29 R_mR = complex(zeros(f_length,f_length));
30 R_eL = complex(zeros(order,order));                    %priori
       error autocorrelation matrix
31 R_eR = complex(zeros(order,order));
32 K_nL = complex(zeros(order,f_length));                 %Kalman
       gain matrix
33 K_nR = complex(zeros(order,f_length));
34 X_mL = complex(zeros(f_length,order));                 %input
       matrix
35 X_mR = complex(zeros(f_length,order));
36
37
38 %output of microphone, for noise calculations
39 d = input;
40
41 %simulation
42 for i = 1:len
43
44     %input stuff first
45     if i < order
46         d_cur = [fliplr(d(1:i)) zeros(1,(order-i))];
47     else
48         d_cur = fliplr(d(i- order + 1:i));
49     end
50
51     %Left reference -populate input matrix with newest
          reference samples
52     if i < f_length
```

```
53          L_in = [fliplr(LREF(1:i)) zeros(1,(f_length-i))];
54      else
55          L_in = fliplr(LREF(i- f_length + 1:i));
56      end
57       X_mL = [L_in.' X_mL(1:f_length,1:order-1)];
58
59
60      %right reference -populate input matrix with newest
            reference samples
61      if i < f_length
62          R_in = [fliplr(RREF(1:i)) zeros(1,(f_length-i))];
63      else
64          R_in = fliplr(RREF(i- f_length + 1:i));
65      end
66
67      X_mR = [R_in.' X_mR(1:f_length,1:order-1)];
68
69     %left - prediction stage
70     L_prev_coeffs = L_coeffs;
71     if i < freeze
72          R_mL = R_uL + Lsigma_w*eye(f_length);
                                        %calculate a priori
              misalignment autocorrealtion matrix
73     end
74
75     %right - prediction stage
76     R_prev_coeffs = Rcoeffs;
77     if i < freeze
78          R_mR = R_uR + Rsigma_w*eye(f_length);
                                        %calculate a priori
```

```
            misalignment autocorrealtion matrix
79      end
80
81      %observation stage - left
82      if i < freeze
83          R_eL = (X_mL')*R_mL*X_mL + (reg)*eye(order);
                                %calculate a priori error signal
            autocorrelation matrix, include regularization
            parameter
84          K_nL = (R_mL*X_mL)/(R_eL);
                                                %calculate Kalman
            gain
85      end
86
87      %observation stage - right
88      if i < freeze
89          R_eR = (X_mR')*R_mR*X_mR + (reg)*eye(order);
                                %calculate a priori error signal
            autocorrelation matrix, include regularization
            parameter
90          K_nR = (R_mR*X_mR)/(R_eR);
                                                %calculate Kalman
            gain
91      end
92
93
94
95
96      %Update Stage - left and right
97      error = d_cur.' - ((X_mL.')*conj(L_coeffs) + (X_mR.')*conj
```

```
    (Rcoeffs));              %error - no noise, make sure the
     transpose is not complex conjugate!
98
99      if i < freeze
100         L_coeffs = L_coeffs + K_nL*(conj(error));
101         Rcoeffs = Rcoeffs + K_nR*(conj(error));
102     else
103          L_coeffs = L_prev_coeffs;
                                          %freeze filter coeffs
104          Rcoeffs = R_prev_coeffs;
                                           %freeze filter
             coeffs
105     end
106
107     if i < freeze
108         R_uL = (eye(f_length) - K_nL*(X_mL'))*R_mL;
                        %update the posteriori misalignment
             autocorrelation matrix
109         R_uR = (eye(f_length) - K_nR*(X_mR'))*R_mR;
                        %update the posteriori misalignment
             autocorrelation matrix
110     end
111
112     Lnorm_coeffs = norm(L_coeffs - L_prev_coeffs);
113     Rnorm_coeffs = norm(Rcoeffs - R_prev_coeffs);
114     if flag == 1
115         Lsigma_w = s_w;
116         Rsigma_w = s_w;
117     else
118         Lsigma_w = (Lnorm_coeffs*Lnorm_coeffs)/(order*f_length
```

```
            );
119         Rsigma_w = (Rnorm_coeffs*Rnorm_coeffs)/(order*f_length
                );
120     end
121
122     %get current error for this timestep
123      mean_e(i) = error(1);
124
125      %calculate misalignment if we can
126      if mis == 1
127         Lmis = norm((hleft - L_coeffs(1:length(hleft))));
128         normL = norm(hleft);
129         Rmis = norm((hright - Rcoeffs(1:length(hright))));
130         Lmisalignment(i) = 20*log10(Lmis/normL);
131         Rmisalignment(i) = 20*log10((Rmis)/(norm(hright)));
132     end
133
134
135 end
```

Appendix C

Miscellaneous

C.1 Under-modeling Analysis

It is clear that the estimated filter taps of the adaptive filter systems should not converge to the real filter taps in under-modeling cases. This finding is seen by considering the criteria that many adaptive filters: the mean square error

$$E\{|e(n)|^2\}$$

A simple example below is sufficient to show this result. Consider an unknown echo path modeled by

$$\boldsymbol{h} = [h_0, h_1, h_2]^T$$

and say that the estimate filter is such that its length is less than the unknown path, i.e.

$$\hat{\boldsymbol{h}}(n) = [\hat{h}_0(n), \hat{h}_1(n)]^T$$

The input is $x(n)$ and thus the error is defined as

$$e(n) = y(n) - \hat{y}(n)$$

where

$$y(n) = [x(n), x(n-1), x(n-2)]\boldsymbol{h} = x(n)h_0 + x(n-1)h_1 + x(n-2)h_2$$
$$\hat{y}(n) = [x(n), x(n-1)]\hat{\boldsymbol{h}}\boldsymbol{(n)} = x(n)\hat{h}_0(n) + x(n-1)\hat{h}_1(n)$$

To find the corresponding filter coefficients of $\hat{\boldsymbol{h}}$ that minimize the mean square error, the following condition must be satisfied:

$$\frac{\partial}{\partial \hat{h}_i} E\{|e(n)|^2\} = 0 \quad \text{for } i = 1, 2$$

Expanding the expression for the mean square error above results in

$$E\{|e(n)|^2\} = E\{|y(n) - \hat{y}(n)|^2\} = E\{y^2(n) - 2y(n)\hat{y}(n) + \hat{y}^2(n)\}$$

Given that the partial derivatives are taken with respect to the estimated filter coefficients, any term that does not contain these will not show up in the minimization. Thus the first term $y^2(n)$ need not be considered. Thus

$$\begin{aligned} E\{-2y(n)\hat{y}(n) + \hat{y}^2(n)\} &= E\{-2[x(n)h_0 + x(n-1)h_1 + x(n-2)h_2][x(n)\hat{h}_0(n) \\ &\quad + x(n-1)h_1] + [x(n)\hat{h}_0(n) + x(n-1)\hat{h}_1(n)]^2\} \\ &= E\{-2[x(n)h_0 + x(n-1)h_1 + x(n-2)h_2][x(n)\hat{h}_0(n) \\ &\quad + x(n-1)h_1] + [x^2(n)\hat{h}_0^2(n) + 2x(n)x(n-1)\hat{h}_0\hat{h}_1 + x^2(n-1)\hat{h}_1^2(n)]\} \end{aligned}$$

Now, taking partial derivatives, and setting them to 0, the optimal filter coefficients can be found.

$$\begin{aligned}\frac{\partial}{\partial \hat{h}_0} E\{|e(n)|^2\} &= \frac{\partial}{\partial \hat{h}_0} E\{ -2[x(n)h_0 + x(n-1)h_1 + x(n-2)h_2][x(n)\hat{h}_0(n)]\} + \\ &\quad \frac{\partial}{\partial \hat{h}_0} E\{[x^2(n)\hat{h}_0^2(n) + 2x(n)x(n-1)\hat{h}_0\hat{h}_1]\} \\ &= E\{-2[x^2(n)h_0 + x(n)x(n-1)h_1 + x(n)x(n-2)h_2]\} + \\ &\quad E\{2[x^2(n)\hat{h}_0(n) + x(n)x(n-1)\hat{h}_1(n)]\}\end{aligned}$$

and

$$\begin{aligned}\frac{\partial}{\partial \hat{h}_1} E\{|e(n)|^2\} &= \frac{\partial}{\partial \hat{h}_1} E\{ -2[x(n)h_0 + x(n-1)h_1 + x(n-2)h_2][x(n)\hat{h}_1(n)]\} + \\ &\quad \frac{\partial}{\partial \hat{h}_1} E\{[x^2(n-1)\hat{h}_1^2(n) + 2x(n)x(n-1)\hat{h}_0\hat{h}_1]\} \\ &= E\{-2[x(n)x(n-1)h_0 + x^2(n-1)h_1 + x(n-1)x(n-2)h_2]\} + \\ &\quad E\{2[x^2(n-1)\hat{h}_1(n) + x(n)x(n-1)\hat{h}_0(n)]\}\end{aligned}$$

where any terms not containing an $\hat{h}_0(n)$ or $\hat{h}_1(n)$ were not considered, since those terms would simply be 0 after the respective derivative was taken. With this, the following two equations should be satisfied in order to minimize the mean square error

$$\begin{aligned}&E\{x^2(n)h_0 + x(n)x(n-1)h_1 + x(n)x(n-2)h_2\} \\ &\qquad = E\{x^2(n)\hat{h}_0(n) + x(n)x(n-1)\hat{h}_1(n)\} \quad \text{(C.1)}\end{aligned}$$

$$\begin{aligned}&E\{x(n)x(n-1)h_0 + x^2(n-1)h_1 + x(n-1)x(n-2)h_2\} \\ &\qquad = E\{x(n)x(n-1)\hat{h}_0(n) + x^2(n-1)\hat{h}_1(n)\} \quad \text{(C.2)}\end{aligned}$$

By inspection of this result, it is easy to see that in this under-modeling case, the taps that minimize the error will not be such that $\hat{h}_0 = h_0$ and $\hat{h}_1 = h_1$, unless the input is white. If the input is white, then the above result may be simplified as follows:

Equation C.1 can be simplified by further splitting the expectation operation:

$$\begin{aligned} \text{LHS of C.1} &= h_0 E\{x^2(n)\} + h_1 E\{x(n)x(n-1)\} + h_2 E\{x(n)x(n-2)h_2\} \\ \text{RHS of C.1} &= \hat{h}_0 E\{x^2(n)\} + \hat{h}_1 E\{x(n)x(n-1)\} \end{aligned}$$

where if $x(n)$ is a white signal, with $E\{x(n-i)x(n-j)\} = 0$ for $i \neq j$, then Equation C.1 reduces to

$$h_0 E\{x^2(n)\} = \hat{h}_0 E\{x^2(n)\}$$

Following the same process but for Equation C.2, this also reduces to

$$h_1 E\{x^2(n-1)\} = \hat{h}_1 E\{x^2(n-1)\}$$

Thus, it is clear to see that to satisfy this and minimize the mean square error, then $\hat{h}_0 = h_0$ and $\hat{h}_1 = h_1$

C.2 Kalman Noise Variance: Practical Estimation

$$\sigma_w^2(n) = \frac{\|\hat{\boldsymbol{h}}(n) - \hat{\boldsymbol{h}}(n-1)\|_2}{PL} \tag{C.3}$$

This shows how practical estimation of the noise variance parameter can be achieved using the estimated filter coefficients. The idea is that at the beginning of the AEC processing, the filter coefficients will be changing rapidly and thus the difference in the norm above will be large. As a result, the noise variance will be large, which will contribute to good tracking. As convergence begins to occur, this difference will get smaller and cause the noise variance to also get smaller, resulting in better steady state error performance. This estimation aims to achieve a better balance in the tracking/steady state error tradeoff that the Kalman filter suffers from.

Bibliography

[1] Reverberation time. www.sciencedirect.com/topics/engineering/reverberation-time.

[2] Jacob Benesty, Tomas Gansler, Dennis R Morgan, M Mohan Sondhi, and Steven L Gray. *Advances in network and acoustic echo cancellation*. Springer, 2001.

[3] Constant314. *Adaptive Filter General*. Mar 2014.

[4] D.h. Dini and D.p. Mandic. Analysis of the widely linear complex kalman filter. *Sensor Signal Processing for Defence (SSPD 2010)*, 2010.

[5] Jmvalin. Lms filter. https://en.wikipedia.org/wiki/File:LMSfilter.svg, Feb 2007.

[6] Sven Nordholm and Jorgen Nordberg. Delayless subband echo cancellation. *Fifth International Congress on Sound and Vibration*, Aug 1997.

[7] C. Paleologu, J. Benesty, and S. Ciochina. Study of the general kalman filter for echo cancellation. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(8):1539–1549, 2013.

[8] C. Paleologu, S. Ciochina, and J. Benesty. Variable step-size nlms algorithm for under-modeling acoustic echo cancellation. *IEEE Signal Processing Letters*, 15:5–8, 2008.

[9] Constantin Paleologu, Jacob Benesty, Steven L. Grant, and Silviu Ciochina. A kalman filter for stereophonic acoustic echo cancellation. *2014 48th Asilomar Conference on Signals, Systems and Computers*, 2014.

[10] M.m. Sondhi, D.r. Morgan, and J.l. Hall. Stereophonic acoustic echo cancellation-an overview of the fundamental problem. *IEEE Signal Processing Letters*, 2(8):148–151, 1995.

[11] Teng Zhang and Ji Wu. Discriminative frequency filter banks learning with neural networks, Jan 2019.

[12] Zhengyou Zhang, Qin Cai, and Jay Stokes. Multichannel acoustic echo cancelation in multiparty spatial audio conferencing with constrained kalman filtering.