UNDERWATER ACTIVE NOISE CANCELLATION COMBINING KALMAN FILTER WITH ADAPTIVE FxLMS

A thesis submitted to the Department of Electronics and Communication Engineering in partial fulfilment of the requirements for the degree of Master of Science in Electronics and Communication Engineering

Submitted by

Md. Mostafizur Rahman
Student ID: 2005111
Session: 2020

Supervised by

Professor Md. Mehedi Islam
Department of ECE, HSTU

Department of Electronics and Communication Engineering

Hajee Mohammad Danesh Science and Technology University (HSTU)

Dinajpur-5200, Bangladesh.

2023

UNDERWATER ACTIVE NOISE CANCELLATION COMBINING KALMAN FILTER WITH ADAPTIVE FxLMS

A thesis submitted to the Department of Electronics and Communication Engineering in partial fulfilment of the requirements for the degree of Master of Science in Electronics and Communication Engineering

Submitted by

Md. Mostafizur Rahman
Student ID: 2005111
Session: 2020

Supervised by

Professor Md. Mehedi Islam
Department of ECE, HSTU

Department of Electronics and Communication Engineering

Hajee Mohammad Danesh Science and Technology University (HSTU)

Dinajpur-5200, Bangladesh.

2023

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

FACULTY OF POST GRADUATE STUDIES

HAJEE MOHAMMAD DANESH SCIENCE AND TECHNOLOGY UNIVERSITY, DINAJPUR-5200, BANGLADESH

CERTIFICATE

This is to certify that the work entitled as **"Underwater Active Noise Cancellation Combining Kalman Filter with Adaptive FxLMS"** by Md. Mostafizur Rahman has been carried out under our supervision. To the best of our knowledge, this work is an original one and was not submitted anywhere for a diploma or a degree.

Supervisor

……

Md. Mehedi Islam
Professor
Department of Electronics and Communication Engineering.
Hajee Mohammad Danesh Science and Technology University, Dinajpur-5200, Bangladesh.

Co-Supervisor

……..

Mahfujur Rahman
Lecturer
Department of Electronics and Communication Engineering.
Hajee Mohammad Danesh Science and Technology University, Dinajpur-5200, Bangladesh.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
FACULTY OF POST GRADUATE STUDIES

HAJEE MOHAMMAD DANESH SCIENCE AND TECHNOLOGY UNIVERSITY, DINAJPUR-5200, BANGLADESH

DECLARATION

The work entitled **"Underwater Active Noise Cancellation Combining Kalman Filter with Adaptive FxLMS"** has been carried out in the Department of Electronics and Communication Engineering, at Hajee Mohammad Danesh Science and Technology University is original and conforms to the regulations of this University. We understand the University's policy on plagiarism and declare that no part of this thesis has been copied from other sources or been previously submitted elsewhere for the award of any degree or diploma.

……...
Md. Mostafizur Rahman
Student ID: 2005111
Session: 2020-2021

The thesis titled **"Underwater Active Noise Cancellation Combining Kalman Filter with Adaptive FxLMS"** submitted by Md. Mostafizur Rahman, Student ID: 2005111 and Session 'January-June' 2023, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Science in Electronics and Communication Engineering.

BOARD OF THESIS EXAMINERS

Professor Dr. Md. Mahabub Hossain Department of ECE, HSTU, Dinajpur.	Chairman
Professor Md. Mehedi Islam Department of ECE, HSTU, Dinajpur.	Examiner (Internal)
Md. Abubakar Siddik Department of ECE, HSTU, Dinajpur.	Examiner (Internal)
Professor Dr. Sajjad Waheed Department of ICT, MBSTU, Tangail.	Examiner (External)
Professor Dr. Mostofa Kamal Nasir Department of CSE, MBSTU, Tangail.	Examiner (External)

ACKNOWLEDGMENT

First, I want to thank the Almighty for allowing me to continue our research and work hard to finish it.

Second, I want to thank my supervisor, Professor Md. Mehedi Islam, and co-supervisor, Lecturer, Mahfujur Rahman, who have guided me during my research. I could not have completed this assignment without their instructions. I am grateful for their outstanding supervision, direction, and encouragement that helped me complete my research.

I also thank the Institute of Research and Training (IRT), Hajee Mohammad Danesh Science and Technology University, Dinajpur-5200, Bangladesh, for helping me attain my aim. I appreciate my parents, friends, younger brothers, and well-wishers who have supported me along the voyage.

List of Contents

List of Figures

List of Tables

List of Abbreviations

ANC	Active Noise Control
AEC	Acoustic Echo Cancellation
LMS	Least Mean Square
NLMS	Normalized Least Mean Square
RLS	Recursive Least Square
FxNLMS	Filtered-X Normalized LMS
FxLMS	Filtered-X Normalized LMS
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
SNR	Signal to Noise Ration
CR	Convergence Rate

Abstract

Underwater environments pose a distinct set of challenges in comparison to terrestrial environments, demanding advanced solutions for data acquisition and control systems. The significance of accurate data readings cannot be overstated, as they directly influence the performance of controllers and augmented systems operating within these environments. To address this, the concept of active noise control (ANC) has emerged, focusing on the effective and adaptive management of both low- and high-frequency noise. This study focuses on creating a strong system to reduce noise and handle disruptions in underwater places using the proposed modified FxLMS algorithm to deal with the specific problems caused by noise, disturbances, and unpredictable changes in sensor readings underwater. The paper will comprehensively explore the formulation of the modified FxLMS algorithm, emphasising its parameters and underlying equations along with some concurrent adaptive algorithms. A thorough analysis and concise discussion of these equations will shed light on their role in achieving noise reduction and data refinement objectives. By leveraging the insights gained from the simulation results, the paper will demonstrate the effectiveness of the proposed model in noise reduction and data enhancement within underwater environments. The ability to achieve these objectives, particularly in the presence of unpredictable variations and noise, underscores the robustness and adaptability of the developed system.

.

Chapter 1

INTRODUCTION

1.1 Underwater Environment

Underwater noise refers to the complex array of sounds present in aquatic environments, primarily caused by natural and anthropogenic sources. These sources include marine life, geological activities, shipping vessels, industrial operations, and even human recreational activities. The study of underwater noise is crucial as it has significant implications for marine ecosystems, communication among marine species, and the accurate collection of scientific data.

Recovering noise-free signals underwater is considerably more challenging compared to terrestrial environments due to several key factors:

1. **Propagation of Sound**: Sound travels differently in water than in air. Water is denser and more efficient at conducting sound, allowing it to travel longer distances and carry more energy. This means that noise generated from distant sources can easily propagate over vast areas, making it difficult to isolate specific signals from the background noise.
2. **Signal Attenuation and Scattering**: As sound travels through water, it experiences attenuation (reduction in intensity) and scattering (directional changes) due to the interaction with particles and molecules in the water column. This phenomenon leads to the distortion of signals, making it challenging to recover the original noise-free signals accurately.
3. **Multiple Noise Sources**: Underwater environments host a multitude of noise sources, both natural and anthropogenic, operating simultaneously. Biological activities such as marine animal vocalisations, geological processes like underwater earthquakes, and human activities like shipping, drilling, and sonar operations all contribute to the acoustic landscape. Separating and filtering out specific signals amidst this cacophony of noise is a daunting task.
4. **Lack of Acoustic Barriers**: Unlike terrestrial environments where physical barriers like walls can help block or isolate noise sources, water lacks such barriers. Sound

waves can travel freely and spread in all directions, further complicating efforts to recover noise-free signals.

5. **Limited Sensor Performance**: Underwater sensors face challenges due to the harsh and corrosive nature of aquatic environments. Pressure, temperature, and fouling (the buildup of marine organisms or debris on sensor surfaces) can still impair the performance of underwater microphones (hydrophones) and other acoustic sensors despite advances in their development.
6. **Complex Data Processing**: The vast amount of collected acoustic data underwater requires sophisticated data processing techniques. Analysing and filtering this data to distinguish desired signals from the surrounding noise demands advanced algorithms and significant computational power.
7. **Environmental Variability**: Underwater environments are highly dynamic, with conditions changing rapidly due to factors such as currents, tides, and weather. These variations can introduce additional noise and complicate the task of recovering noise-free signals.

Efforts to recover noise-free signals underwater involve a combination of advanced signal processing techniques, acoustic modelling, and the deployment of specialised sensor arrays. Researchers are continually working to develop innovative solutions to mitigate the challenges posed by underwater noise, including real-time noise cancellation algorithms.

1.2 Active Noise Control (ANC)

Active Noise Control (ANC) is a sophisticated technology that holds significant promise for mitigating the challenges of recovering noise-free signals in underwater environments. Unlike passive noise reduction methods that rely on physical barriers or materials to attenuate noise, ANC operates in real-time to actively counteract noise by generating anti-noise signals. In underwater environments, where a complex mixture of natural and anthropogenic noise sources can overlap and interfere with desired signals, ANC's adaptability becomes a crucial advantage.

ANC systems typically consist of hydrophones that capture incoming noise and feed it to a control unit. This unit then processes the noise signals, generating anti-noise signals that are precisely tailored to the incoming noise's characteristics. When these anti-noise signals combine with the original noise, they effectively cancel each other out,

resulting in a noise-free acoustic environment. The basic concept was first introduced by Bernard widrow et al [1].

The ability of ANC to dynamically adjust its anti-noise signals based on real-time noise conditions is particularly well-suited for the underwater environment. The technology can adapt to changes in noise sources, intensity, and spatial distribution, providing a continuous and effective noise reduction solution. This adaptability is especially important in scenarios such as marine animal communication research, underwater surveillance, and the monitoring of underwater infrastructure. While challenges like sensor deployment, power supply, and algorithm complexity need to be addressed, ongoing advancements in ANC technology hold great promise for enhancing the recovery of noise-free signals in underwater environments. By actively countering the unique challenges posed by underwater noise, ANC contributes to improved data accuracy, communication reliability among marine species, and the overall understanding of aquatic ecosystems.

1.3 Background

Active Noise Control (ANC) uses feedforward to make a sound-cancelling pressure wave with the secondary/control speaker based on a reference signal measurement from the primary/disturbance speaker/source to lower the sound pressure at the error microphone (Fig. 1.1)

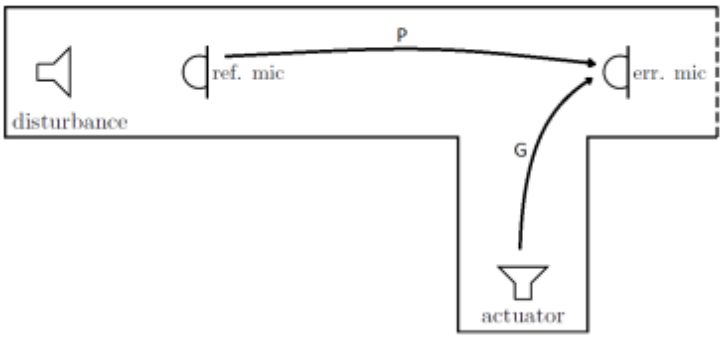


Figure 1: Diagram of a duct with a dusturbance speaker, a reference microphone, a noise cancellation actuator and an error microphone in which P denotes the primary and G the secondary path.

The adaptive filter is an essential part of ANC since it provides noise reduction without prior knowledge of the noise or signal. Traditional filters would cause distortion in the desired signal output. As a result, adaptive filters are appropriate in circumstances where communication and noise signals are random in character [2].

To increase the performance of ANC, various adaptive algorithms have been proposed. Among them, RLS (recursive least squares) and LMS (least mean squares)-based algorithms are popular due to their fewer complications [3]. Adaptive filtering involves adjusting the parameters of a filter in real-time to achieve a desired output based on input data. Both RLS and LMS algorithms are widely used for this purpose due to their distinct characteristics and advantages.

RLS is known for its optimal parameter estimation capabilities. It takes into account the entire history of input data, making it particularly effective in situations where the data is correlated or stationary [2]. This makes RLS well-suited for applications requiring accurate and precise parameter estimation, such as adaptive beamforming in communication systems, channel equalisation, and system identification. However, RLS has higher computational complexity compared to LMS, which can be a limitation in resource-constrained environments [4].

Reducing the computational burden associated with matrix inversion in RLS, YT Zhang proposed Fast RLS (FRLS) [5] that utilizes efficient matrix factorization techniques. This makes it more suitable for real-time implementations and resource-constrained environments. FRLS is particularly advantageous for applications that require quick and efficient adaptation, making it ideal for scenarios with rapidly changing input conditions. FRLS achieves computational efficiency through approximations, which can introduce errors and impact estimation accuracy to some extent compared to the standard RLS algorithm. But implementing FRLS with proper matrix factorization techniques requires careful consideration and may involve more complex design compared to the standard RLS.

Paulo Sergio Ramirez proposed a Lattice-Based Recursive Least Squares (RLS) Algorithm that utilizes lattice structures for efficient computation [6]. In this method, the input data is processed through a series of lattice stages, with each stage representing a different tapped delay line. The algorithm updates coefficients at each stage while minimizing the estimation error. The lattice structure reduces computational requirements compared to conventional RLS, making it suitable for real-time and resource-constrained applications. Lattice-based RLS handles high-dimensional data effectively, which is crucial in modern signal processing applications. However, while it reduces complexity compared to traditional RLS, the lattice-based approach might still be more complex than simpler adaptive filtering methods like the Least Mean Squares (LMS) algorithm. Implementing

and understanding the lattice-based approach might require a steeper learning curve compared to more straightforward algorithms.

The QR decomposition-based Recursive Least Squares (RLS) Algorithm uses QR decomposition to simplify RLS update computation [7]. The input data matrix is decomposed into an orthogonal matrix (Q) and an upper triangular matrix (R) using this procedure. The QR decomposition-based approach reduces computational complexity, especially in high-dimensional scenarios, and offers improved numerical stability compared to standard RLS algorithms. Although this approach can mitigate the sensitivity of RLS to data perturbations, leading to improved robustness, but may require additional memory to store the decomposition matrices, potentially affecting its applicability in memory-constrained environments. The initial decomposition step adds some computational overhead, which might impact the algorithm's performance during the initial phase.

Regularized RLS (RRLS) prevents overfitting and helps maintain stable filter adaptation. But the introduction of regularization might trade off some convergence speed compared to the standard RLS. Block RLS processes data in blocks rather than individual samples, reducing the overall computational load and memory requirements compared to traditional RLS when dealing with large datasets. But processing data in blocks introduces a delay in the adaptation process compared to standard RLS, which can impact the algorithm's ability to track rapidly changing system dynamics.

1.4 LMS based algorithms vs. RLS based algorithms

The convergence rate of LMS-based algorithms is significantly slower than that of the RLS algorithm. Although the RLS algorithm has an exceptional convergence rate, it cannot track the estimation because it is dependent on its model, input data, and, as the computation advances, the correlation matrix [8].

Whether the least mean squares (LMS)-based algorithm is better than the recursive least squares (RLS)-based algorithm depends on the specific context and requirements of the application. Both algorithms are widely used in adaptive filtering and have their own advantages and disadvantages.

1.4.1 Advantages of LMS over RLS:

1. **Simplicity and Lower Complexity**: LMS is generally simpler to implement and has lower computational complexity compared to RLS. It updates the filter

coefficients incrementally for each new data point, making it more suitable for real-time processing and applications with limited computational resources.

2. **Adaptation Speed**: LMS updates its coefficients with every new data point, which can lead to faster adaptation to changes in the data. This is particularly useful when the system dynamics are changing rapidly.
3. **Robustness to Outliers**: LMS tends to be less sensitive than RLS. It can handle situations where the data might contain occasional large errors without significantly affecting the parameter estimation.

1.4.2 Advantages of RLS over LMS:

1. **Optimal Parameter Estimation**: RLS provides optimal parameter estimation. It takes into account the entire history of data, leading to potentially more accurate estimates, especially when dealing with stationary or correlated data.
2. **Fewer Tunable Parameters**: RLS typically has fewer tuning parameters to set compared to LMS, which may simplify the algorithm setup and reduce the need for manual parameter tuning.

In summary, the choice between LMS and RLS depends on factors such as the desired level of accuracy, computational resources, speed of adaptation, noise characteristics, and the specific characteristics of the application.

LMS-based algorithms are often used in underwater environments because they can handle the unique challenges of underwater acoustic communication in a flexible and stable way. The underwater medium introduces severe signal propagation issues such as multipath reflections, signal attenuation, and high noise levels. LMS algorithms excel in these conditions as they update filter coefficients incrementally based on the most recent data points, enabling real-time adaptation to changing channel conditions. Their ability to track rapid variations in the channel, such as those caused by moving underwater vehicles or changing water conditions, makes them well-suited for dynamic underwater environments. The simplicity of LMS implementations aligns with the constraints often faced in underwater systems, where computational resources and power may be limited. Overall, LMS-based algorithms offer a practical and effective approach to mitigating the unique challenges of underwater communication, making them a popular choice in underwater acoustic signal processing and communication systems.

One key problem with LMS is its sensitivity to the scale of the input data, which can lead to slow convergence or even divergence if the step size is not appropriately adjusted. NLMS overcomes this problem by normalizing the step size based on the power of the input signal, resulting in a more consistent and stable convergence rate across a wide range of input magnitudes. NLMS tackles this problem by emphasizing larger updates for smaller error contributions, effectively reducing the impact of noise on the adaptation process. However, NLMS can be sensitive to noise and may require careful tuning of its parameters to achieve optimal performance. Using Past Weight Vectors and Regularization parameters, Manish D. Sawale and Ram N. Yadav proposed a new NLMS algorithm [9] that offers certain advantages but also presents notable drawbacks. On the positive side, leveraging past weight vectors can enhance convergence and tracking performance by incorporating historical information. However, this approach comes with several limitations. The utilization of past weight vectors and a regularization parameter can lead to increased computational complexity, which might hinder real-time processing in resource-constrained applications. Moreover, the integration of these additional components may introduce additional hyperparameters that need to be optimized, adding complexity to the algorithm's implementation and tuning process.

Normalized Least Mean Square (NLMS) algorithm based on the Kalman Filter framework provides certain advantages [10]. One notable benefit is the potential for improved tracking and adaptation in dynamic environments, as the Kalman Filter inherently accounts for time-varying characteristics. However, the combined algorithm's performance is heavily dependent on the accurate estimation of initial states and parameters, which can be challenging in practical scenarios.

In FXLMS (Filtered-x LMS), these issues are mitigated through the introduction of a secondary adaptive filter, often referred to as the "secondary path model" [11]. This model figures out the system's unwanted dynamics, like echoes or reverberations, and subtracts them from the primary output. This way, disturbances like these have less of an effect on the adaptation process. FXLMS also has a fractional delay component that helps fix phase mismatches between the primary and secondary paths. This makes the algorithm work better in situations where phase alignment is very important. By using these new ideas, FXLMS improves convergence speed, stability, and overall performance. This makes it a good improvement over NLMS in situations where system dynamics are complicated and phase differences are common.

However, the complete mathematical analysis of the FxLMS algorithm and the exact rules for the step size adjustment are not known at this time, likely due to its highly nonlinear properties [3]. FxLMS inherited the step size, which is the most inherent characteristic of the Least Mean Squares (LMS) algorithm, and it requires careful adjustment. Convergence is difficult due to the small step size required for a small excess mean square error. Large step sizes, which are necessary for rapid adaptation, may cause a loss of stability. Consequently, modifications to this algorithm are required, in which the step size varies during the adaptation process based on specific characteristics.

1.5 Motivation

The dynamic nature of underwater noise, influenced by factors like changing currents, varying noise source locations, and unpredictable marine activity, demands a solution that can respond in real-time to these fluctuations. In the literature, FxLMS is identified as a comparatively suitable method for noise cancellation of continuing noise, but it has limitations that must be addressed.

1.6 Objective

By addressing following objectives, the thesis aims to contribute to the field of noise cancellation and enhance our understanding of LMS-based algorithms' effectiveness in real-world applications.

- To simulate and compare the performance of various LMS-based methods, including: Least Mean Square (LMS), Normalized Least Mean Square (NLMS) algorithm, Filtered-x NLMS (FxNLMS) algorithm, Filtered-X LMS (FxLMS) algorithm.
- To investigate the rate of convergence for each of the mentioned algorithms and evaluate the signal-to-noise ratio (SNR) for the different LMS-based algorithms.
- To develop a modified Filtered-x LMS (FxLMS) algorithm for noise cancellation.
- To analyze and compare the performance of the modified FxLMS algorithm against the existing LMS-based methods in terms of both convergence rate and SNR.

1.7 Structure

The purpose of this study is to implement several adaptive filtering techniques for noise cancellation. The thesis comprises both a theoretical and a practical section to help the reader understand the proposed solution. In the theoretical part of the thesis, adaptive filtering theories and the proposed algorithm are explained. In the practical part, LMS,

NLMS, FxNLMS, and FxLMS, as well as the proposed FxLMS algorithm, are used in real-world situations.

Chapter 2, "Physics of Sound," introduces the foundational concepts of sound waves, pressure, and their behavior. It covers the basics of acoustics, including the propagation of sound waves in various dimensions, the superposition principle, and the relationship between frequency, phase, and noise cancellation performance in Active Noise Cancellation (ANC) systems.

In Chapter 3, an introduction to adaptive filters is provided, as well as a description of the distinctions between the various adaptive filtering methods. The adaptive filtering theory described by Monson H. Hayes in the book “Statistical Digital Signal Processing and Modelling” [12] is used as the key reference.

Chapter 4 "Methods and Materials" introduces the Kalman Filter, a state estimation algorithm used for navigation and noise reduction, explaining its components. Proposed adjustments for noise reduction are discussed, including the use of Kalman gain instead of a fixed step size for FxLMS for effective signal denoising.

Chapter 5 "Results and Discussion" presents the outcomes of the proposed methodology for noise cancellation in terms of various adaptive filtering algorithms simulated in Matlab and the performance results, emphasizing comparison criteria such as convergence rate and signal-to-noise ratio (SNR) before and after filtering, showcasing the advantages of the Modified Filtered-x Least Mean Squares (FxLMS) algorithm in terms of faster convergence and efficient noise reduction.

Chapter 6, The last chapter summarizes the current work and performance findings.

Chapter 2

THEORETICAL VIEW OF ACTIVE NOISE CONTROL

2.1 Introduction

Active noise control (ANC) is the technique of cancelling sound waves with a compensation source. In the ideal case, the compensation signal would be of the same magnitude as the noise, but 180° different in phase, through out an entire sound field to be silenced. This chapter will cover the techniques used to achieve noise cancellation with real systems.

2.2 ANC structures

In this chapter, basic single channel ANC structures are introduced. These are the feedforward and feedback ANC structures. Then, adaptive filters are discussed, including why and how they are used in different ANC systems. Finally, the most common algorithms used to implement adaptive filters are covered.

2.2.1 Feedforward method

In the feedforward method a reference sensor is picking up a reference signal *x(n),* which is correlated with the unwanted noise *d(n)* [13]. This reference signal is used to produce a compensation signal played by a compensation source. The process is monitored with an error sensor, which can adaptively control processes used to create the compensation signal [13] [14]. This adaptation is commonly done with an adaptive filter, noted as W(z).

The acoustic domain path from the reference sensor to the error sensor is known as the primary path *P(z).* Most acoustic domain signal modifications are linear such as, temporal delay, magnitude alteration and phase modifications [13] [14]. Thus, the effects of the primary path can be represented with a linear filter.

The acoustic domain compensation signal produced by the ANC algorithm and the compensation source is noted as y(n) [13]. Signals *d(n)* and y(n) behave linearly in the acoustic domain and their residual is picked up by an error sensor. This residual error transferred to the electrical domain is noted as e(n) [14].

The path from the adaptive filter's output, back to the adaptation process as the error signal is known as the secondary path *S(z).* This path includes effects from digital analog converter, compensation source response, acoustic propagation path to error sensor, error sensor response, anti-aliasing filter and finally analog digital converter [13]. Figure 2 shows the complete feedforward ANC structure.

The sensors used for reference and error measurements are commonly microphones, but nonacoustic meters such as tachometers and optical sensors can be used as well [13]. However, broadband ANC is more commonly implemented with microphones, while other sensors are used in narrowband applications like engine noise or active vibration control [13]. The issues with microphones is that, they may suffer from acoustic feedback caused by compensation source signal leaking to reference microphone, whereas nonacoustic meters are less prone to feedback. Acoustic feedback can be combated by mechanical construction or by filtering out the compensation signal from the error microphone [13].

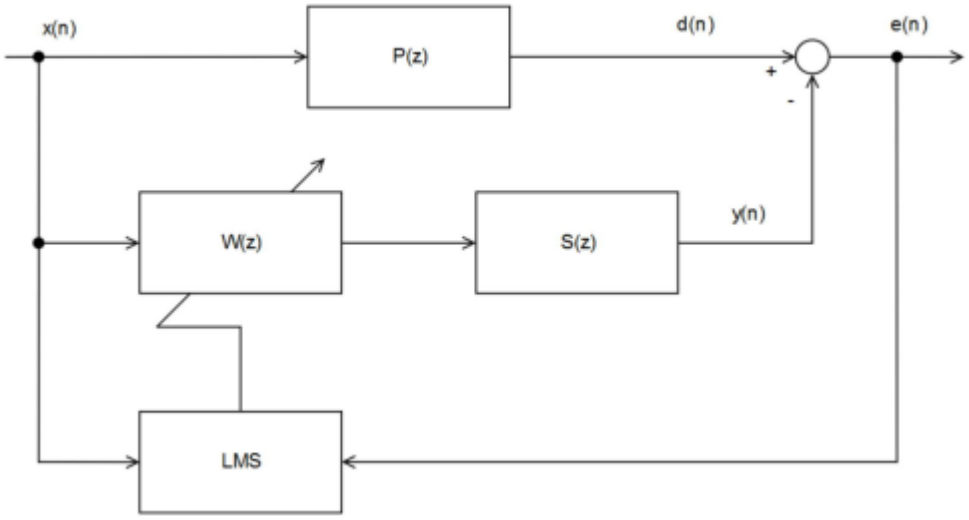


Figure 2: Feedforward ANC diagram. Figure adopted from *[13]*

2.2.2 Analog feedback method

In feedback ANC structure there is an error sensor, compensation source and a compensation signal processing block. Since no reference sensor is used, acoustic feedback issues caused by compensation signal leaking in to reference sensor are avoided. The lack of a reference sensor accentuates the effects of processing loop delay. In feedback ANC, compensation signal can be produced with analog circuits and adaptive processes [13].

Adaptive methods will be introduced in Chapter 3, while an example of the analog feedback system is introduced here.

Figure 13 shows the novel structure of a feedback ANC. The symbols used for noise signal *d(n)*, compensation signal y(n), secondary path *S(z)* and error signal e(n) are the same as in the feedforward structure.

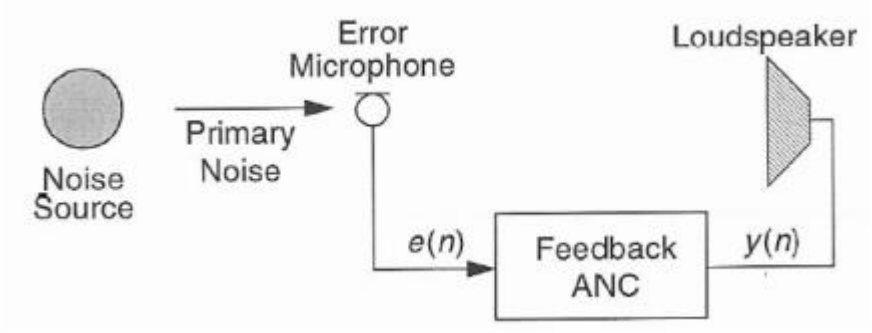


Figure 3: Feedback ANC diagram. Figure adopted from *[13]*.

Analog feedback control systems have been developed, by including a phase inverting amplifier of —A in the feedback loop, where A is the factor of amplification applied in the feedback loop. Performance of such structure is dependent on effects included in the secondary path, such as processing loop delay, overall phase response and acoustic domain propagation time from compensation source to error sensor. When frequencies get higher, smaller temporal errors cause larger shifts in phase. Since secondary path can never be flat in real applications, there is always a limit on how high frequencies can be attenuated with an analog feedback ANC system [13].

In practice, as frequencies get higher the 180° phase starts gradually turning in to positive feedback, resulting in an unstable system. An intuitive solution to this problem would be to limit the system to only allow stable frequencies. However, filters with sharp cutoff frequencies can have phase responses, which can make even low frequencies unstable. Sharper cutoffs also introduce more delay to the signal, further diminishing the performance of feedback ANC. For these reasons, moderate filter rolloffs are preferred and consequently the cutoff has to be set to a lower frequency, limiting usable band of a feedback system [13].

As mentioned, delay in the feedback loop has to be minimized. One way to limit the loop delay is to physically move the error sensor closer to the compensation source, effectively reducing the delay caused by secondary path's acoustic propagation time.

However, there is a limit on how close the error microphone can be to the compensation source. The radiation patter of a loudspeaker may not be as uniform and well behaved in the near field as in the far field [13].

2.2.3 Adaptive feedback method

Another way of producing compensation signal in feedback ANC is to use a linear predictor. Linear predictors work by calculating the next sample based on previous samples. Because there is no upstream of samples available in the feedback method, linear predictors are only capable of attenuating the periodic components in noise [11].

2.3 Adaptive filters

Adaptive filters change their coefficients according to their input, thus they are considered nonlinear [15]. However, at any given time instance the adaptation can be halted and those filter coefficients can be viewed as a linear filter. Some have used nonlinear filters for ANC but, acoustic domain processes are approximated linear in this thesis. Thus, the filters covered in this thesis are linear as well.

In the feedforward ANC method adaptive filters are used for unknown system modeling and in the feedback method for predicting future samples. Since the processes to be modeled are unknown, the adaptation has to be done iteratively [15]. Speed of the adaptation is related to the step size, commonly noted as μ. This section will focus more on why and how adaptive filters are used in ANC. Different algorithms used for adaptation are introduced in the following sections.

2.3.1 General block diagram of the adaptive filters:

In Figure 4, w represents the coefficients of the FIR filter tap weight vector, x(n) is the input vector sample, is a delay of one sample, y(n) is the adaptive filter output, d(n) is the desired echoed signal and e(n) is the estimation of the error signal at time n. The aim of an adaptive filter is to calculate the difference between the desired signal and the adaptive filter output, e(n).The error signal is fed back into the adaptive filter and its coefficients are changed algorithmically in order to minimize a function of this difference, which is known as the cost function [16].

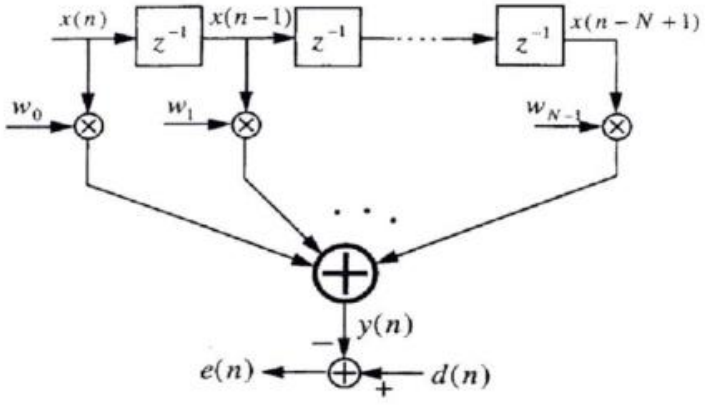


Figure 4: Block diagram of adaptive filter.

In the case of acoustic echo cancellation, the optimal output of the adaptive filter is equal in value to the unwanted echoed signal. When the adaptive filter output is equal to desired signal the error signal goes to zero. In the situation the echoed signal would be completely cancelled and the far user would not hear any of their original speech returned to them.

2.3.2 Applications of adaptive filters

a) System identification

System identification deals with the capability of an adaptive system to find FIR filter that best reproduces of another system, whose frequency response is unknown. The diagrammatical set up is shown in Figure 5.

When the adaptive system reaches its optimum value and the output is close to zero an FIR filter is obtained whose weights are the result of the adaptation process that is giving the same output as that of the 'unknown system' for the same input. In other words, the FIR filter reproduces the behavior of the 'unknown system' [17]. This design is said to be efficiently working when the frequency response of the system to be identified matches with that of a certain FIR filter. In case of unknown system having an all-pole filter, then the FIR filter will approach for the best result. The system output will never be zero but it may compromise reducing it by converging to an optimum weight vector. The frequency response of the FIR filter will try to get the best approximate out of it but not exactly equal to that of the 'unknown system.

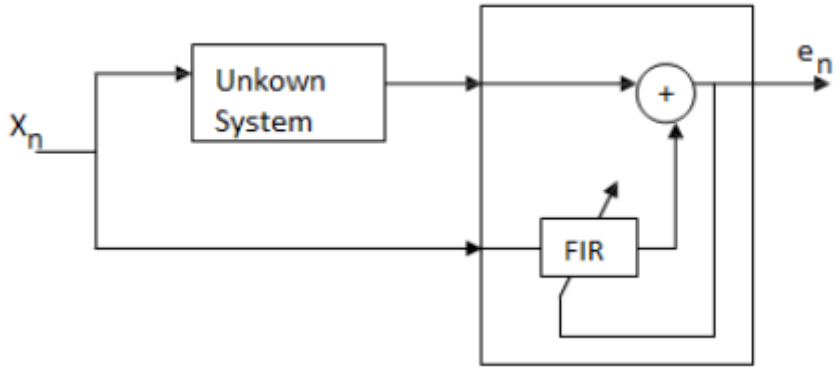


Figure 5: System identification via adaptive filter.

b) Noise cancellation in speech signals

Adaptive filtering can be extremely useful in cases where a speech signal is submerged in a very noisy environment with many periodic components lying in the same bandwidth as that of speech [17]. The design of adaptive noise canceller for speech signals consists of two inputs. The desired input consists of voice that is corrupted by noise (speech signal) and other reference input that contains noise which is related in some way to the desired input noise. The noise reference input is made as similar as that of the desired input noise by passing it to the system filter and that filtered version is subtracted from the desired input. Therefore, by removing the noise from the desired input signal the noise free signal is obtained. The setup is shown in Figure 6. From practical system noise is not completely removed but its level is reduced considerably.

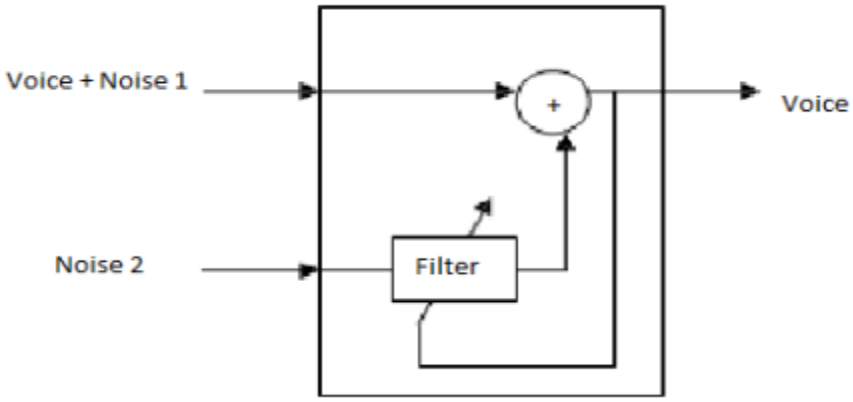


Figure 6: Noise cancellation via adaptive filter.

c) Signal prediction

Predicting signals may seem to be an impossible task, without some limiting assumptions. Assume that the signal is either steady or slowly varying over time, and

periodic over time as well. Here the function of the adaptive filter is to provide best prediction (in some sense) of the present value of a random signal. Accepting these assumptions, the adaptive filter must predict the future values of the desired signal based on past values. When **s**(k) is periodic signal and the filter is long enough to remember previous values, this structure with the delay in the input signal, can perform the prediction. This structure can also be used to remove a periodic signal from stochastic noise signals. The present value of the signal serves the purpose of a delayed response for the adaptive filter. Past values of the signal supply the input applied to the adaptive filter. Depending upon the application of interest, the adaptive filter output or the estimation (prediction) error may serve as the system output. In the first case, system operates as a predictor, in the latter case; it operates as a prediction error filter. The setup is shown in Figure 7.

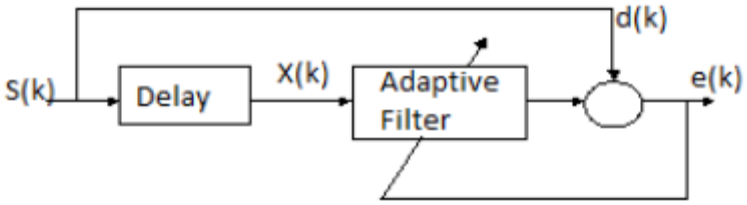


Figure 7: Predicting future values of a periodic signal.

d) Interference cancellation

In this application, adaptive filter is used to cancel unknown interference contained alongside an information signal component in a primary signal, with the cancellation being optimized in some sense in Figure 8. The primary signal serves as the desired response for the adaptive filter. A reference (auxiliary) signal is employed as the input to the adaptive filter. The reference signal is derived from the sensor or set of sensors located in relation to the sensors supplying the primary signal in such a way that the information signal component is weak or essentially undetectable [17].

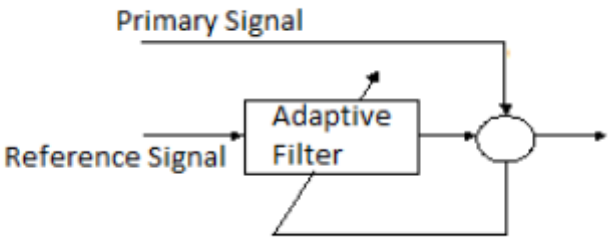


Figure 8: Interference cancellation model via adaptive filter.

e) Channel equalization

In communication channels such as wireless, telephone and optical channels are affected by inter-symbol interference (ISI). The channel bandwidth becomes inefficient, without the utilization of channel equalization. Channel equalization is a process of compensating for the effects caused by a band-limited channel, hence enabling higher data rates [18]. These effects are due to the out-of-boundary transmission medium and the multipath effects in the radio channel. A typical communication system is depicted in Figure 9.

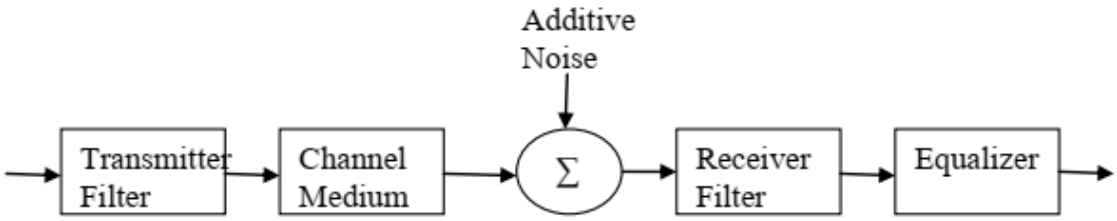


Figure 9: A baseband communication system.

In the receiver the equalizer is incorporated by introducing inter-symbol interference to the channel. The equalizer output transfer function is directly inverse to the channel transfer function estimate.

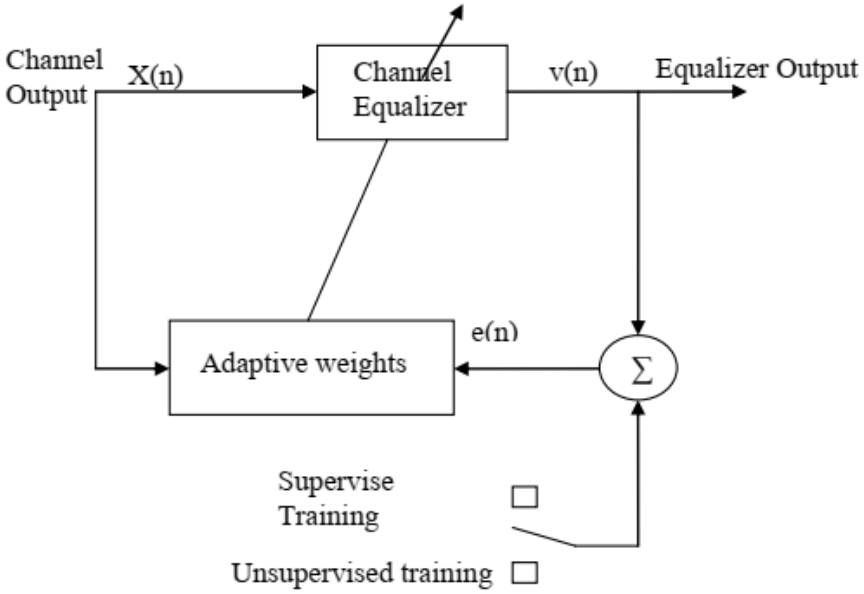


Figure 10: Adaptive equalizer.

The equalizer is designed to be adaptive to the channel variation in the transmission of high speed data over a band limited channel. The equalizer is recursively updated by an

adaptive algorithm based on the observed channel output for reconstructing the output signal. The configuration of an adaptive equalizer is depicted in Figure 14.

2.3.3 Filter types

Adaptive filters are implemented with digital filters [13]. Digital filters are either finite impulse response (FIR) filters, meaning the output of the filter is produced from a weighted sum of the previous input samples or infinite impulse response (IIR) filters, where the output is a weighted sum of previous input and output samples [15].

Adaptation algorithms have been developed for both FIR and IIR filters.

2.3.3.1 FIR Adaptive filters

Finite Impulse Response FIR filters as the name suggests, have an impulse response with finite length. A non-recursive filter has no feedback and its input-output relation is given in (2.3.1) by the linear constant coefficient difference equation.

$$y(n) = \sum_{k=0}^{q} b_n(k)x(n-k) \qquad 2.3.1$$

The output y(n) of a non-recursive filter is independent of the past output values, it is a function only of the input signal x(n) and the filter coefficient b(k), where k=0,1,…,q. The response of such a filter to an impulse consists of a finite sequence of q+1 samples, where q is the filter order. A direct-form FIR adaptive filter for estimating a desired signal d(n) from the related input signal x(n) is illustrated in the next figure.

Finding the coefficient vector w_n at sample n that produces the least amount of mean-square error is the objective of the process of creating the FIR adaptive filter. In equation (2.3.2), the filter output y(n) of a FIR adaptive filter is calculated in order to estimate a desired signal d(n) based on a related signal x(n). This estimation is performed in order to find the distance between the two signals [12].

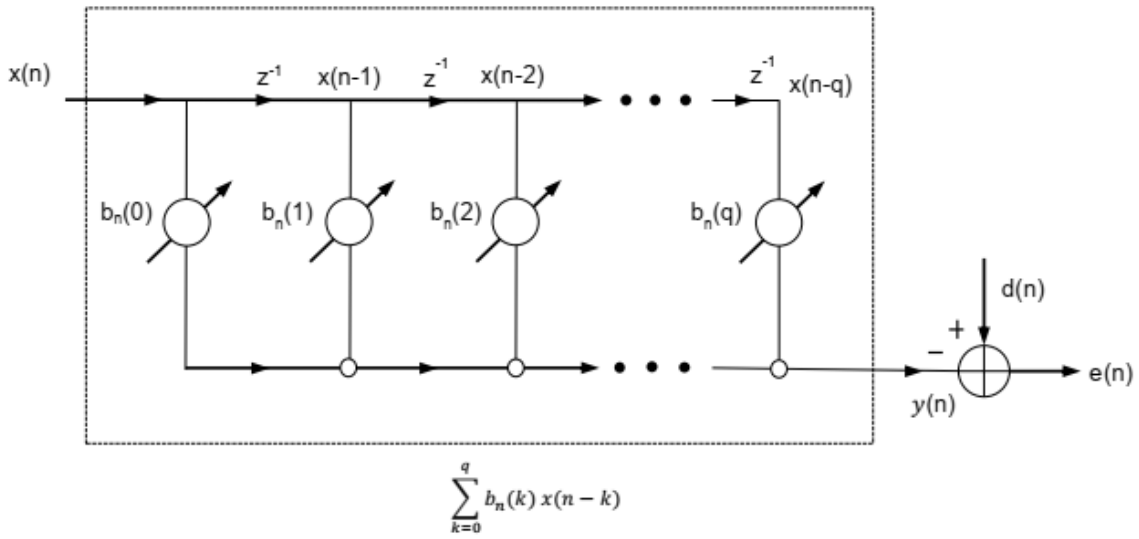


Figure 11: Block diagram of a direct-form FIR adaptive filter (Source: modified from *[12]*)

$$y(n) = \sum_{k=0}^{q} w_n(k)x(n-k) = w_n^T x(x) \qquad 2.3.2$$

It is assumed that both signals x(n) and d(n) are non-stationary signals and the goal is to find the coefficient vector at time n that minimizes the mean-square error.

$$\xi(n) = E\{|e(n)|^2\} \qquad 2.3.3$$

The error signal in (2.3.4) is calculated from the difference between the filter output signal y(n) and the desired signal d(n).

$$e(n) = d(n) - y(n) = d(n) - w_n^T x(n) \qquad 2.3.4$$

To find the filter coefficients that minimize the mean-square error it is necessary to set the derivative $\xi(n)$ equal to zero with respect to $wn*$(k) for k = 0,1…, q and $*$ represents the complex conjugate, which leads to the result**.**

$$E\{e(n)x^*(n-k)\} = 0 \qquad 2.3.5$$

Substituting equation (2.3.4) in equation (2.3.5), it becomes (2.3.6) [12]

$$E\left\{\left[d(n)-\sum_{k=0}^{q} w_n(k)x(n-k)\right]x^*(n-k)\right\}=0 \qquad 2.3.6$$

FIR filters are commonly used for noise cancellation applications. The FIR filter in its non-recursive form is always stable. FIR filters can have a linear phase response and they can be set up in order introduce no phase distortion to the signal. The FIR filter can have any structure, like direct form, cascade form or lattice form, but the most common form is the direct form, also known as transversal structure.

2.3.3.2 IIR Adaptive filters

An IIR filter can have an infinite number of coefficients in its impulse response. IIR filters feature feedback that goes from the output to the input, and the output is a function of both the most recent input samples and those that came before them [19]. The linear constant coefficient difference equation of the IIR filter is (2.3.7).

$$y(n)=\sum_{k=1}^{p} a_n(k)y(n-k)+\sum_{k=0}^{q} b_n(k)x(n-k) \qquad 2.3.7$$

where a_n (k) and b_n (k) are the coefficients of the adaptive filter at sample n. The output sample y(n) depends on past output samples y(n-k), as well as recent and past input samples x(n-k), that is known as the IIR filter's feedback. Shown in the following figure is the block diagram of an IIR adaptive filter.

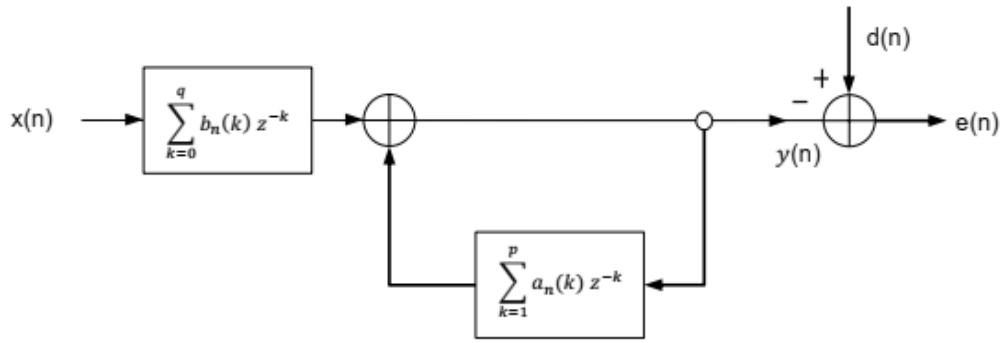


Figure 12: Block diagram of an IIR adaptive Filter.

In order to minimize the mean-square error $\xi(n) = E\{|e(n)|2\}$, where e(n) is the difference between the desired process d(n) and the output of the adaptive filter y(n), it is necessary to define some vectors [12]. The filter coefficient vector Θ is represented as

$$\Theta = \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a(1), a(2), \dots, a(p) \\ b(0), b(1), \dots, b(q) \end{bmatrix}^T \quad 2.3.8$$

The data vector z(n) in (2.3.9) denotes the aggregate data vector and contains the past output samples as well as recent and past input samples.

$$z(n) = \begin{bmatrix} y(n-1) \\ x(n) \end{bmatrix} = \begin{bmatrix} y(n-1,), y(n-2,), \dots, y(n-p) \\ x(n-p), x(n), \dots, x(n-q) \end{bmatrix}^T \quad 2.3.9$$

The output of the filter in (2.3.10) can be expressed in terms of the filter coefficients Θ and the data vector z(n)

$$y(n) = a_n^T y(n-1) + b_n^T x(n) = \Theta^T z(n) \quad 2.3.10$$

With the feedback coefficients an(k) the mean-square error is no longer quadratic, $\xi(n)$ may have multiple local minima and maxima [12] [20]. The gradient vector must be set to zero

$$E\{e(n)\nabla e^*(n)\} = 0 \quad 2.3.11$$

Since e(n) = d(n) – y(n) then

$$E\{e(n)\nabla y^*(n)\} = 0 \quad 2.3.12$$

Differentiating $\nabla y*(n)$ with respect to $a*(k)$ and $b*(k)$ results in

$$\begin{aligned} \frac{\delta y^*(n)}{\delta a^*(k)} &= y^*(n-k) + \sum_{k=1}^{p} a^*(k) * \frac{\partial y^*(n-k)}{\partial a^*(k)} \; ; k = 1,2, \dots, p \\ \frac{\delta y^*(n)}{\delta b^*(k)} &= y^*(n-k) + \sum_{k=0}^{q} b^*(k) * \frac{\partial y^*(n-k)}{\partial b^*(k)} \; ; k = 0,1, \dots, q \end{aligned} \quad 2.3.13$$

Finally, combining the equations leads to

$$E\left\{e(n)\left[y(n-k)+\sum_{k=1}^{p}a(k)*\frac{\partial y(n-k)}{\partial a(k)}\right]^{*}\right\}=0\,;k=1,2,\dots,p$$
$$E\left\{e(n)\left[x(n-k)+\sum_{k=0}^{q}b(k)*\frac{\partial y(n-k)}{\partial b(k)}\right]^{*}\right\}=0\,;k=0,1,\dots,q \qquad 2.3.14$$

Since the equations are nonlinear, they are difficult to solve for the optimum filter coefficients. Due to the nonlinearity, the solution may not be unique, therefore the solution will rather correspond to a local rather than a global minimum.

The strength of the IIR filters comes from the feedback procedure, but the disadvantage of it is that the IIR filter becomes unstable or poor in performance if it is not well designed. A common form of the recursive IIR filter is the lattice structure.

IIR filters construct their output from past input and output samples and require less coefficients. This comes at the cost of not being able to have linear phase response and having to check the stability of the filter. If the output samples being fed back to the filter are amplified by their coefficients, the filter's output can grow exponentially and the filter can become unstable. IIR filters have been used in adaptive filters, but as the filter coefficients adapt, the IIR structure stability has to be constantly monitored. In practice, this reduces the size of the step size used and thus the convergence is slower [13].

FIR filters require more coefficients compared to IIR filters, but they are always stable. All linear behavior in phase and magnitude can be modeled with FIR filters. Even the response of an IIR filter can be reproduced with a FIR filter, by taking sufficient amount of coefficients from the impulse response of a given IIR filter. In this thesis only adaptive FIR filters were investigated.

Chapter 3

ADAPTIVE FILTERING ALGORITHMS

3.1 Adaptive filtering algorithms

Adaptive algorithms are used to modify the coefficients of the digital filter in such a way that the error signal is minimized in accordance with some criterion. There are several variants of adaptive filtering algorithm to choose from. Most of them can be categorized as least mean square (LMS) and recursive least square (RLS) based. An overview of the different types can be seen in Figure 13.

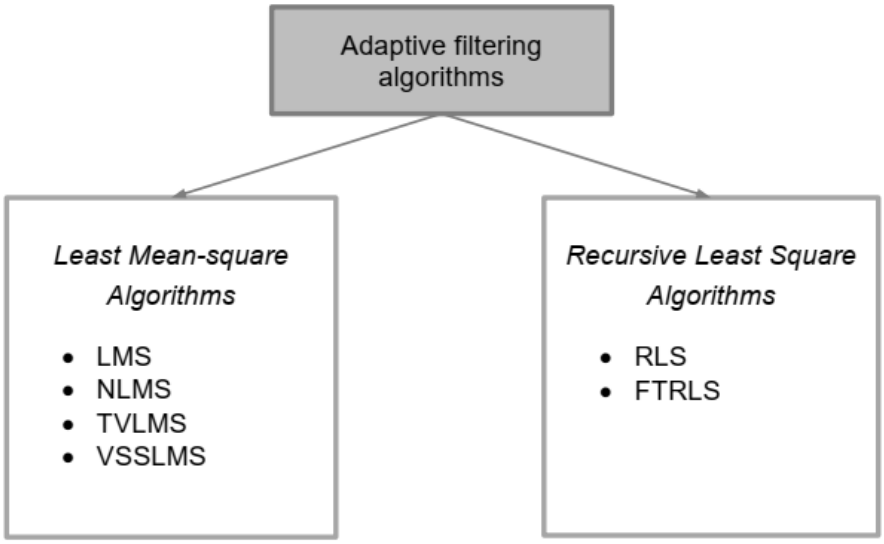


Figure 13: Types of Adaptive Filtering Algorithms.

LMS algorithms, are like smart tools that adjust certain settings in a digital filter to make sure the difference between what we want and what we actually get becomes as small as possible [21]. There are different types of these LMS variants, such as NLMS, TVLMS, and VSSLMS. These algorithms use certain methods to make these adjustments, kind of like following a path that guides them based on the current mistake.

Recursive adaptive filtering algorithms like RLS and FTRLS find coefficients that minimize a weighted linear least squares cost function for input signals. This differs from least mean-squares LMS, which reduces mean-square error. [22].

LMS-based algorithms are popular since they can adapt and withstand underwater acoustic communication challenges. Details are discussed in chapter 1. The underwater medium introduces severe signal propagation issues such as multipath reflections, signal attenuation, and high noise levels. LMS algorithms excel in these conditions as they update filter coefficients incrementally based on the most recent data points, enabling real-time adaptation to changing channel conditions, and also makes them well-suited for dynamic underwater environments.

We will look into different LMS based methods, like the Least Mean Square(LMS) itself, Normalized Least Mean Square(NLMS) algorithm, Filtered-X NLMS (FxNLMS) algorithm and Filtered-X LMS (FxLMS) algorithm, and compared how well they work.

3.2 LMS algorithm

Least mean square (LMS) is an algorithm used to calculate adaptive filter coefficients. It can be used to model unknown systems, such as the primary path of a feedforward ANC system. The LMS algorithm uses gradient descent, a technique where it takes steps in the direction opposite to the gradient, helping it locate a nearby lowest point [15]. Due to it being light to calculate and simple to implement, the LMS algorithm has established itself as the standard algorithm used in adaptive filter applications.

The LMS algorithm works by approximating the true gradient with the mean of squared error. The resulting equation for adapting filter coefficients for the next iteration can be represented as

$$w(k+1) = w(k) + 2\mu\, ex(k) \qquad 3.2.1$$

where $w(k+1)$ is a vector containing the updated filter coefficients, *w(k)* is a vector containing the current adaptive filter coefficients, μ the factor of step size taken in the direction of the gradient, e the instantaneous error which remains constant for all filter coefficients throughout the iterations and $x(k)$ is a buffer containing the most recent reference signal samples [15].

Figure 14 shows the LMS algorithm implemented with a **FIR** filter, in a block diagram form. The most resent reference signal sample $x(k)$ is fed to the algorithm and all previous samples of x are passed along the chain of unit delays z^{-1}. The result of the adaptive FIR filter's output, denoted as y(k), is determined by multiplying each sample of the filter, represented as $x(k-n)$, with its corresponding weight, $w_n(k)$, and then adding

them together [15]. Mathematically this can be written as $y(k) = \sum_{n=0}^{N} w_n x(k-n)$ or just $w^T x$.

The error sample *e(k)* is calculated by summing *y(k)* and noise sample *d(k)*, which is then multiplied by 2μ for the adaptation process. Finally, all coefficients are moved in the direction of negative gradient $2\mu e x(k)$ as shown in equation 6. [15]

Step size μ determines adaption speed. Small μ take longer to obtain ideal filter coefficients but are more stable.

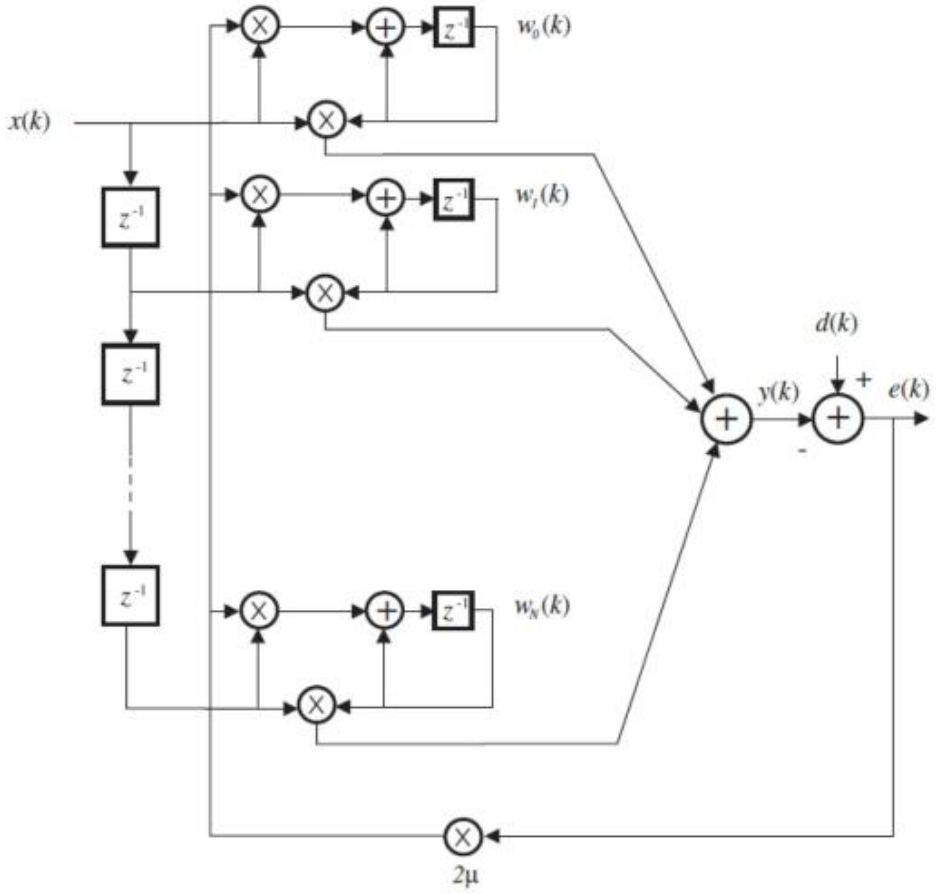


Figure 14: Block diagram of the LMS algorithm implementation with a FIR filter. Diagram adapted from [15].

3.2.1 Implementation of the LMS algorithm:

The following equation illustrates LMS components.

- The output signal y(n) is computed by a standard FIR filter:

$$y(n) = \sum_{i=0}^{M-1} w_i(n) * x(n-i) \qquad 3.2.2$$

- The error signal equals the difference between the reference signal $\boldsymbol{d(n)}$ and filter output:

$$e(n) = d(n) - y(n) \qquad 3.2.3$$

- After each iteration on which the sample of the error signal is computed, the filter coefficients w(n) are updated for n=0,1,2…

$$w(n+1) = w(n) + \mu\, e(n)\, x*(n) \qquad 3.2.4$$

The step size μ influences how quickly the coefficients come together. The resulting output signal, y(n), and the error signal, e(n), are expressed using equations (3.2.2) and (3.2.3) respectively.

3.2.2 Advantages and disadvantages of LMS algorithm

- The LMS algorithm has a straightforward implementation and requires minimal computational resources.
- The LMS algorithm has overall better convergence, meaning it can converge to the optimal solution under certain conditions.
- While the LMS algorithm generally converges quickly, it can exhibit slow convergence in certain scenarios. This can occur when dealing with highly correlated input signals or when the step size parameter is not appropriately selected.
- If the initial values of LMS filter are far from the optimal solution, it may take longer to converge or even converge to a suboptimal solution.

3.2.3 Computational complexity of LMS algorithm

In each iteration of the LMS algorithm, the total number of multiplications is 2M, where M is the number of filter taps.

3.3 NLMS algorithm

The LMS is based on steepest descent algorithm. LMS algorithm is used widely for different applications such as channel equalization and echo cancellation. This algorithm is used due to its computational simplicity.

One of the main drawbacks of the LMS algorithm is that it uses the same step size every time it makes changes. This can be tricky when the input signal keeps changing. To solve this, we can change the step size as needed using time varying approach. The NLMS method is like a smarter version of LMS. It figures out a different step size for each change it makes. This step size depends on how much energy the input signal has. So, when the input signal is stronger, NLMS takes smaller steps, and when it's weaker, it takes bigger steps. This helps NLMS work better with changing signals.

In the NLMS algorithm, step size towards the gradient is normalized with the sum of all reference signal samples squared, which can be represented as,

$$\mu = \frac{\mu}{\alpha + x^T x}$$

where α is a small value, used to prevent division with zero in the case of all zero buffer of *x.* Since NLMS scales the steps size to match the magnitude of reference signal, it only affects the filter coefficient adaptation.

In real applications size of the reference cannot be controlled, NLMS algorithm has to be used. If NLMS is not used while the reference signal levels change, it has the same effect as changing the LMS algorithm's step size [13]. This means, low reference signal levels will adapt slower and high levels faster or even turn the system unstable.

3.3.1 Implementation of the NLMS algorithm

- The output of the received signal is calculated as

$$y(n) = \sum_{i=0}^{N-1} w(n)x(n-i) = w^T(n)x(n) \qquad 3.3.1$$

- An error signal is the difference between the reference signal and the filter output

$$e(n) = d(n) - y(n) \qquad 3.3.2$$

- The step size is calculated as

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

- The filter tap weights are updated for the next iteration

$$w(n+1) = w(n) + \mu(n)e(n)x(n) \qquad 3.3.3$$

3.3.2 Advantages and Disadvantages of NLMS algorithm

- NLMS algorithm having low computational complexity, with good convergence speed.
- It has minimum steady state error.
- Sensitivity to rapid changes in the input signal, leading to potential instability.
- Difficulty in handling correlated input signals, affecting adaptation performance.
- The step size selection process can be challenging and critical for optimal performance.

3.3.3 Computational Complexity of NLMS algorithm

- Computational Complexity of NLMS is 3N+1, which is N times as much multiplying as LMS, where N is the length of the coefficient vector.

3.4 FxNLMS algorithm

The FxNLMS (Filtered-x Normalized Least Mean Squares) algorithm is an adaptive filtering algorithm that combines the benefits of the NLMS algorithm with an additional filtering step to enhance its performance in various scenarios.

In the FxNLMS algorithm, the input signal is filtered by an FIR filter to create a filtered reference signal. This filtered reference signal is then used to adaptively adjust the filter coefficients based on the error between the desired signal and the filter output. The key idea behind FxNLMS is that by using the filtered reference signal, the algorithm can effectively adapt to changes in the input signal and improve its ability to track dynamic variations.

The FxNLMS update equation involves the adjustment of the filter coefficients in proportion to the error signal and the filtered reference signal. Importantly, the step size parameter is normalized by the energy of the filtered reference signal to ensure stable and

controlled adaptation. Additionally, a small positive constant is introduced to prevent division by zero.

3.4.1 Implementation of the FxNLMS algorithm

The FxNLMS algorithm equations are as follows:

- **Filtering the Reference Signal:** FxNLMS introduces a reference signal **u(n)** which is filtered through an FIR filter **F** to create a filtered reference signal $u_f(n)$:

$$u_f(n) = F \cdot u(n)$$

where **F** is the FIR filter coefficients and **u(n)** is the original reference signal.

- **Error Calculation:** The error signal **e(n)** is the difference between the desired signal **d(n)** and the output **y(n)**:

$$e(n) = d(n) - y(n)$$

- **Update Coefficients:** The coefficients **w(n)** of the adaptive filter are updated using the FxNLMS update equation:

$$w(n+1) = w(n) + \frac{\mu}{\epsilon + u_f^T(n)u_f(n)} \cdot e(n)u_f(n) \qquad 3.4.1$$

Where,

- w(n) is the coefficient vector at time n.
- u_f is the filtered reference signal.
- e(n) is the error signal.
- mu is the step size parameter.
- epsilon is a positive constant.

This algorithm adaptively adjusts the filter coefficients based on the error between the desired signal and the filter output. The filtering of the reference signal **u(n)** with the FIR filter F provides improved tracking and convergence behavior compared to the basic NLMS algorithm.

3.4.2 Advantages and disadvantages of FxNLMS algorithm

- FxNLMS's filtered reference signal accelerates convergence and improves tracking in changing environments.

- The algorithm's filtering step enhances noise suppression and improves signal-to-noise performance.
- The filtered reference signal requires additional memory storage.
- May not be as effective in handling highly nonlinear systems or signals.

3.4.3 Computational complexity of FxNLMS algorithm

The filtering operation and coefficient update are performed for each of the N iterations, resulting in a computational complexity of O(LN). Compared to the basic NLMS algorithm, the FIR filtering step adds some overhead, but the benefits of improved convergence and noise robustness may justify the increased complexity.

3.5 FxLMS algorithm

The optimal solution for adaptive filter is W(z) = *P(z)/S(Z).* Since *S(z)* is constantly part of the ANC system it would be more efficient if W(z) would not have to adapt to reversing relatively constant effects of *S(z)* [13].

An intuitive solution to this would be to create an inverse filter for *S(z)0.* However, in all real application *S(z)* produces latency, from steps such as acoustic flight time between compensation source and error sensor. Thus, in order to create an inverse filter for *S(z)*, the inverse filter would be required to reverse latency, i.e. predict future samples. This is not possible to be implemented unless, W(z) has modeled some latency, which can compensate for the needed prediction.

Effects of *1/S(z)* can be removed from *W(z)*, by placing an estimate of secondary path $\hat{S}(z)$, prior to the LMS algorithm [23]. Since $\hat{S}(z)$ is placed in front of *x(n)* the algorithm is called filtered x LMS (FxLMS). $\hat{S}(z)$, can be modeled offline, prior to starting the system. It has been shown that offline model of *S(z)* does not have to be perfect. In testing, the FxLMS algorithm was able to converge even with 90° phase error. Figure 15 shows a block diagram of a feedforward FxLMS system.

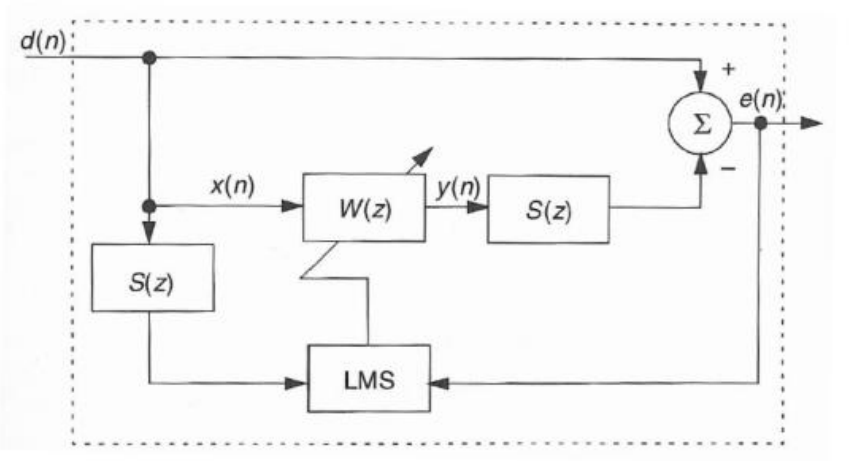


Figure 15: Block diagram of a FxLMS feedforward ANC system. Diagram adapted from [13].

If the secondary path of a feedback system can be modeled, a reference signal correlated with *d(n)* can be synthesized. The synthesized reference can then be used to simulate a feedforward type structure. In a feedback ANC system, only e(n) and y(n) are known. Since, *d(n) = e(n) + s(n) * y(n),* a reference signal *x(n)* can be created as $x(n)=\hat{d}(n)= e(n) + \hat{s}(n) * y(n)$, where $\hat{d}(n)$ is an approximation of the noise signal and $\hat{s}(n)$ the modeled secondary path. This system can function as a feedforward system, however since the reference is synthesized and not measured, there is no advantage of a physical look-ahead from having a microphone further away from the silenced zone. Figure 13 shows a block diagram of the introduced feedback FxLMS system.

Why $\hat{S}(z)$ is added to the adaptation algorithm can be understood as the following. Since *W(z)* and S(z) are both linear, the order in which signals are processed through them does not make a difference. If *S(z)* was before *W(z)* in 8 block diagram, it could be easier to see that, if effects of S(z) were also applied to the adaptation algorithm, it would be effectively the same as applying *S(z)* to both paths. Because *S(z)* cannot be removed from the system, adding $\hat{S}(z)$ prior to the adaptation algorithm includes the same filtering effects to both paths. Once W has converged the *W(z)S(z) = P(z)* or *W(z)S(z) = —P(z),* depending on whether or not a separate inverter was used.

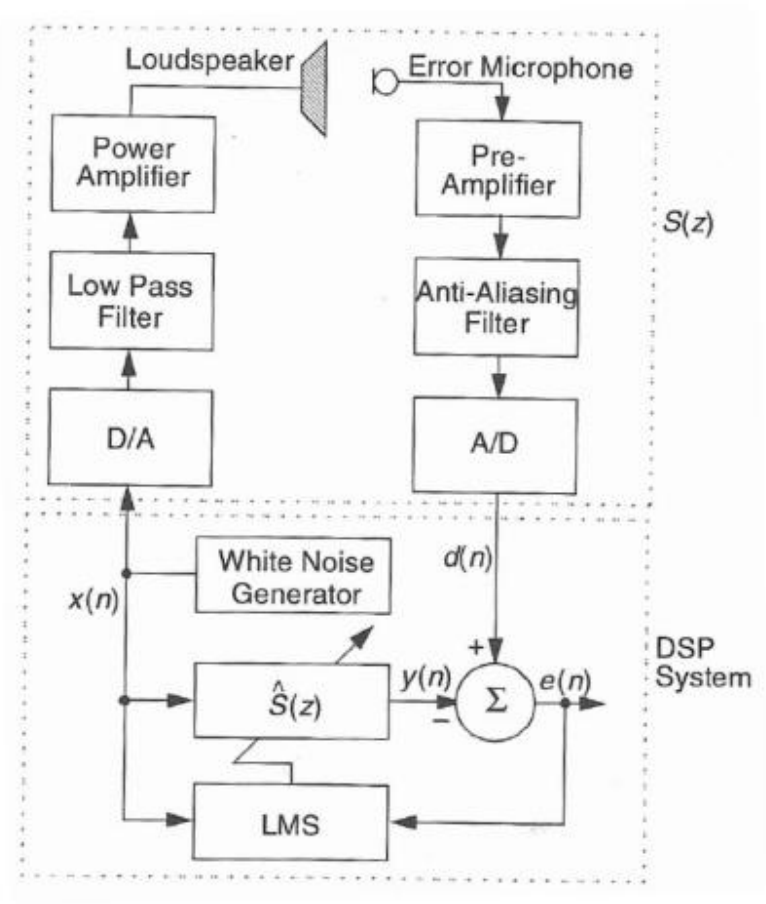


Figure 16: Block diagram of an offline calculation setup for estimating system's secondary path. Diagram adapted from [13].

One method for realizing the offline estimation is to internally create a broadband signal, such as white noise, to be used as a reference signal. The broadband signal can be passed to the adaptive filter and the compensation source. Once the adaptation has converged, the adaptive filter has created an estimate of S(z). Since only a compensation source and error sensor are used, this method can be realized for both, feedforward and feedback systems. Figure 16 shows the described algorithm, with secondary path effects shown in the upper dotted line rectangle.

FxLMS extends the LMS method by adding a secondary path model. This extra model helps create an adaptive filter to eliminate unwanted noise or interference. It works by using a reference signal that matches the noise, which goes through the adaptive filter to create a guess of the noise. This estimated noise is then subtracted from the main input signal to get the signal we want. FxLMS is really good at getting rid of noise and is often used for active noise control.

3.5.1 Advantages and disadvantages of FxLMS algorithm

- Effective in reducing the impact of a known reference noise signal on the adaptive filter output.
- Can adjust the filter coefficients to be balanced to track changes in the reference noise signal.
- Performance heavily relies on the accuracy of the reference noise signal estimation.
- Limited effectiveness when dealing with non-stationary or time-varying noise signals.

3.5.2 Computational complexity of the FxLMS algorithm

The computational complexity of the FxLMS algorithm is similar to that of the LMS algorithm, with additional computations for the reference signal filtering. The complexity per iteration can be expressed as O(M + N), where M is the number of filter taps and N is the length of the reference signal.

3.6 Comparing Adaptive Filtering Algorithms:

In the realm of adaptive filtering algorithms, various methods have been developed to efficiently adapt to changing environments and process data in real-time applications. This table summarize comprehensive comparison of four widely used adaptive filtering techniques that already discussed above: Least Mean Squares (LMS), Recursive Least Squares (RLS), Normalized Least Mean Squares (NLMS), Filtered-x Normalized Least Mean Squares (FxLMS) and Filtered-x Least Mean Squares (FxLMS). Each algorithm has its unique strengths and limitations, making it crucial for engineers and researchers to understand their characteristics thoroughly.

The comparison on Table 1 focuses on key criteria to evaluate the algorithms' performance, encompassing Computational Complexity, Convergence Rate, Signal-to-Noise Ratio (SNR) handling capabilities, Cost considerations, overall Effectiveness in various applications, and Limitations that may affect their practical utility.

Table 1: Summary of different Adaptive Filtering Algorithms.

Parameter / Algorithms	LMS (Least Mean Squares):	NLMS (Normalized Least Mean Squares)	FxNLMS (Filtered-x Normalized Least Mean Squares)	FxLMS (Filtered-x Least Mean Squares)
Computational Complexity	Low	Low	Low to Moderate	Moderate
Convergence Rate	Slower compared to RLS and Affine projection	Slower compared to LMS	Similar to LMS	Moderate
SNR	Moderate performance	Moderate performance	Good performance when used for active noise control	Good performance
Cost	Low	Low	Moderate	Moderate
Effectiveness	Simple and widely used, suitable for basic adaptive filtering tasks	Improved stability and robustness compared to LMS	Effective in active noise control applications, cancels noise or interference	Fast convergence, suitable for sparse and time-varying environments
Limitation	Slower convergence compared to other algorithms, may struggle with highly dynamic environments	Slightly slower convergence than LMS, may not be as effective in highly dynamic environments	Limited to active noise control applications, may not be as versatile as other algorithms for general adaptive filtering tasks	May not be as effective in handling highly nonlinear systems or signals.

ANC aims to reduce unwanted noise by generating an anti-noise signal that cancels out the incoming noise, leading to a quieter and more pleasant environment. Though FxLMS may not be as versatile as other algorithms for general adaptive filtering tasks but

excels in this domain due to its adaptability and real-time response capabilities. It efficiently tracks changes in noise characteristics by adaptively updating its filter coefficients, enabling precise estimation of the anti-noise signal required for effective noise cancellation.

One of FxLMS's key strengths lies in its ability to accurately model the secondary path between the noise source and the error microphone. By employing a filtered reference signal that accounts for the secondary path's transfer function, FxLMS effectively addresses phase mismatches and non-linearities, which are common challenges in ANC systems. This advanced modeling ensures that the anti-noise signal is appropriately tailored to cancel the specific noise components, resulting in superior noise reduction performance.

The adaptability and robustness of FxLMS make it a favored choice for various active noise control implementations, ranging from aircraft cabins and car interiors to industrial facilities. Its ability to deliver high-quality noise cancellation while efficiently utilizing computational resources makes it well-suited for real-time and resource-constrained applications.

Overall, FxLMS's effectiveness in active noise control applications has positioned it as a prominent algorithm in the pursuit of creating quieter and more comfortable environments, significantly enhancing the acoustic experience for occupants in diverse settings.

3.7 Problems findings in FxLMS

The stability of the Filtered-x Least Mean Squares (FxLMS) algorithm is indeed highly dependent on the step size, also known as the learning rate or adaptation constant. The step size plays a critical role in determining how quickly the algorithm adapts its filter coefficients to track changes in the system being controlled.

To understand the relationship between the step size and stability in FxLMS, let's delve into the algorithm's functioning. FxLMS is an adaptive filtering technique commonly used in applications like active noise control, echo cancellation, and system identification. It operates by continuously adjusting the filter coefficients to minimize the error between the desired output and the actual output.

The step size, denoted by the symbol μ (mu), represents the magnitude of the update applied to the filter coefficients during each iteration of the algorithm. A larger step size implies that the filter coefficients will be adjusted more aggressively, leading to faster

convergence, which means the algorithm adapts rapidly to changes in the environment or input signals.

However, a very large step size can lead to instability in the FxLMS algorithm. When the step size is too big, the algorithm becomes overly sensitive to variations in the input signal or noise, causing the filter coefficients to fluctuate significantly with each iteration. This oscillatory behavior can lead to divergence, where the algorithm fails to converge to a stable solution and instead produces unstable and unpredictable outputs.

On the other hand, a very small step size can result in slow convergence, where the algorithm takes a long time to reach a satisfactory solution. This slow convergence may limit the algorithm's ability to effectively track rapid changes in the system or noise characteristics.

3.7.1 Solving Idea

Selecting the right step size plays a critical role in ensuring stability and achieving optimal performance in the FxLMS algorithm. However, a dynamic step-size technique can offer a more elegant solution. By dynamically adjusting the step size during the filtering process, the algorithm can adapt to variations in the system dynamics, noise levels, and other environmental changes in real-time.

That's where the Kalman Filter steps in – it's like a fast learner that predicts how things will change and can helps the noise-canceling algorithms adjust quickly. By teaming up, they make sure the noise gets removed even when things are tricky, like when the noise or the signal changes suddenly. We got better rate of convergence and Signal to noise ratio (SNR) as well.

The upcoming chapter will elaborate kalman filter and also the idea of merging the Kalman gain into FxLMS and how this integration enhances the algorithm's performance.

Chapter 4

METHODS AND MATERIALS

4.1 Kalman Filter

In Kalman filter, filtering means actually estimating the state vector at the present time which is based upon the past observed data. Prediction, on the other hand, is the estimation of the state vector at a future time [24]. Most state estimation algorithms are based on the Kalman filter. We employ this state estimation solely to acquire the best possible solution from the numerous measures. The Kalman filter takes into account all measurement data that is sent into it over time, not simply the most recent batch of measurements. It is more of an estimating algorithm than a filter. It also keeps real-time estimates of a variety of factors. Estimates are updated using a succession of noise measurements. It employs knowledge of the deterministic and random aspects of system parameters and measurements to derive the best estimates possible from the available data. It is also called Bayesian estimation technique [24]. This recursive technique is more efficient for real-time applications such as navigation since only new measurement data needs to be handled on each iteration. We can get rid of the previous measurement data.

The Kalman filter is a versatile tool for estimating variables in various processes. It estimates the states of a linear system, minimizing estimation error variance. Widely used in embedded control systems, it maintains uncertainties and correlations between parameter errors for optimal data weighting. Unlike non-recursive methods, Kalman filters iteratively update estimates using prior data [24].

4.2 Elements of Kalman filter

a) State vector:

It is a set of parameters which describe a system, known as states, which the Kalman filter estimates. Each state may be constant or may be time varying. For many navigation applications, the state includes the components of position or position error. Velocity, altitude and navigation sensor error states may also be estimated. Along with the state vector there is an error covariance matrix which represents the uncertainties in the Kalman filter's state estimate and degree of correlation between errors in those

estimates. This correlation information within the error covariance matrix is important for the following 3 reasons.

a) It enables the error distribution of the state estimates to be completely represented.
b) There is not always sufficient information from the measurement to estimate the Kalman filter states independently. The correlation information enables estimates of linear combinations of these states to be maintained while awaiting further measurement information.
c) Correlations between errors can build up over the integral between measurements. Understanding this lets us figure out one error from another. Since the Kalman filter works in steps, we must set the initial values of the state and uncertainty matrix, often from another process [25].

b) System model:

This model is also called the process model or time propagation model which describes how Kalman filter states and error covariance matrix vary with time. Example- A position state will vary with time as integral of a velocity state, the position uncertainty will increase with time as the integral of velocity uncertainty, the position and velocity estimation errors will become more correlated. The system model is deterministic for the states, as it is based on known properties of the system. A state uncertainty should also be increasing with time to account for unknown changes in the system which causes the state estimate to go out of data in the absence of new measurement information. These changes may be unmeasured dynamics or random noise on an instrument output [25]. Example- A velocity uncertainty must be increasing over time if acceleration is unknown. This variation over the true values of the states is called as system noise or process noise and its assumed random properties are usually defined by K.F designer.

c) Measurement vector:

It is a set of simultaneous measurements of properties of system which are functions of state vector. Along with the measurement vector is a measurement noise covariance matrix that describes the statistics of noise on the measurement. For many applications, new measurement information is input to K.F at regular intervals. But in some other cases the time interval between measurements can be irregular.

d) Measurement model:

It describes how the measurement vector varies with the function of true state vector in the absence of measurement noise. The Kalman filter is a set of mathematical equations which provides us an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in various aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown to us.

4.3 Kalman Filter Algorithm

The Kalman Filter estimates a process simply by using feedback control like form. The operation may be described as the process is estimated by the filter at some particular point of time and the feedback is obtained in the form of noisy measurements. The Kalman filter equations can be divided into two categories: time update equations and measurement update equations. To obtain the a priori estimates for the next time step the time update equations project forward (in time) the current state and error covariance estimates. The measurement update equations get the feedback to obtain an improved *a* posteriori estimate which incorporates a new measurement into the a priori estimate [26].

4.3.1 System Model

The Kalman filter estimates the state of a discrete-time process governed by the linear stochastic difference equation.

$$x_k = Fx_{k-1} + Bu_{k-1} + w_{k-1}$$

where F is the state transition matrix. x_{k-1} is the previous state vector. B is the control-input matrix applied to the control vector u_{k-1} and w_{k-1} is the process noise vector.

The process model is combined with the measurement model to describe the state-measurement relationship at time step k:

$$z_k = Hx_k + v_k$$

Here z_k is the measurement vector, H is the measurement matrix, and v_k is the measurement noise vector. In different literature, "measurement" is often called "observation" [27] [28]. Note that subscripts to these matrices are omitted here by assuming that they are invariant over time as in most applications. Although the covariance

matrices are supposed to reflect the statistics of the noises, the true statistics of the noises is not known or not Gaussian in many practical applications. Therefore, Q and R are usually used as tuning parameters that the user can adjust to get desired performance.

4.3.2 Kalman filter equations

Kalman filter is popular for having easy computation, memory requirements and good capability on overcoming noises. It is state technique estimation that can extract information from noisy data [29]. So, a Kalman Filter is the best way to reduce noise on sensor readings in general, especially when how often the noise might happen on the sensor reading is unknown. There are various types of Kalman Filter, such as standard Kalman Filter [30], Extended Kalman Filter, Unscented Kalman Filter [31] etc. Standard Kalman Filter is the simplest while the other types are modified for more complicated tasks. The paper will use standard Kalman filter since it contains enough part of equation for noise reducing.

Kalman filter algorithm consists of two stages: prediction and update. Note that the terms "prediction" and "update" are often called "propagation" and "correction," respectively, in different literature. The Kalman filter algorithm is summarized as follows:

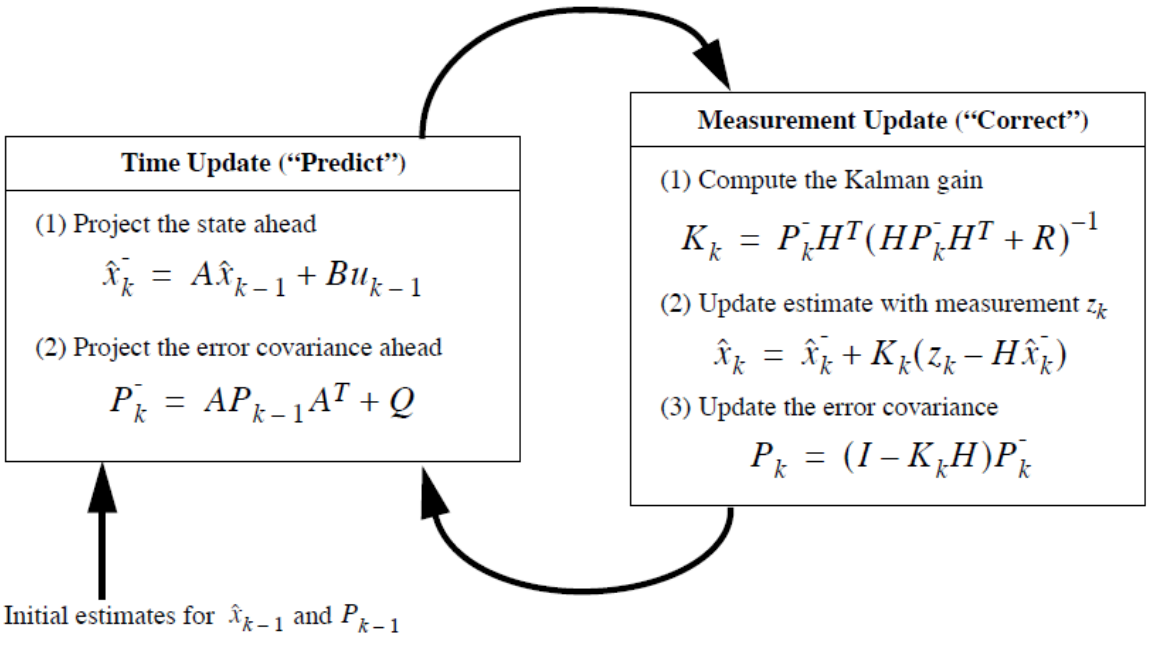


Figure 17: A complete picture of the operation of the Kalman filter.

Predict:

$$\hat{x}_{t|t-1} = F_t \hat{x}_{t-1|t-1} + B_t u_t \quad 4.3.1$$

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + Q_t \quad 4.3.2$$

Update:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - H_t \hat{x}_{t|t-1}) \quad 4.3.3$$

$$K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1} \quad 4.3.4$$

$$P_{t|t} = (1 - K_t H_t)\, P_{t|t-1} \quad 4.3.5$$

where x is estimated state, F is state transition matrix, u is control variables, B is control matrix, P is state variance matrix, Q is process variance matrix, y is measurement variables, H is measurement matrix, K is Kalman gain, R is measurement matrix, $t|t$ is current time period, $t\text{-}1|t\text{-}1$ is previous time period, and $t|t\text{-}1$ is intermediate steps.

In the above equations, the hat operator ^ means an estimate of a variable. That is, $\hat{x}$ is an estimation of *x*. The superscripts – and þ denote predicted (prior) and updated (posterior) estimates, respectively. The predicted state estimate is evolved from the updated previous updated state estimate. The new term P is called state error covariance. It encrypts the error covariance that the filter thinks the estimate error has. Note that the covariance of a random variable x is defined as $cov(x) = E[(x - \hat{x})(x - \hat{x})^T]^T$ where E denotes the expected (mean) value of its argument. One can observe that the error covariance becomes larger at the prediction stage due to the summation with Q, which means the filter is more uncertain of the state estimate after the prediction step.

In the update stage, the measurement residual $\tilde{y}_k$ is computed first. The measurement residual, also known as innovation, is the difference between the true measurement, z_k, and the estimated measurement, $H\tilde{x}_k$. The filter estimates the current measurement by multiplying the predicted state by the measurement matrix. The residual, $\tilde{y}_k$, is later then multiplied by the Kalman gain, K_k, to provide the correction, $K_k\tilde{y}_k$, to predicted estimate $\tilde{x}_k$. After it obtains the updated state estimate, the Kalman filter calculates the updated error covariance, P_k , which will be used in the next time step.

Note that the updated error covariance is smaller than the predicted error covariance, which means the filter is more certain of the state estimate after the measurement is utilized in the update stage.

The process noise covariance matrix, *Q*, and measurement noise covariance matrix, *R*, can be constructed following the real noise statistics described above to get the best performance. However, have in mind that in real applications, we do not know the real statistics of the noises and the noises are often not Gaussian. Common practice is to conservatively set *Q* and *R* slightly larger than the expected values to get robustness.

Q and *R* are constant for every time step. The more uncertain your initial guess for the state is, the larger the initial error covariance should be. A single run is not sufficient for verifying the statistic characteristic of the filtering result because each sample of a noise differs whenever the noise is sampled from a given distribution, and therefore, every simulation run results in different state estimate.

In real applications, one will be able to acquire only the estimated covariance. Also, getting a good estimate of *Q* and *R* is often difficult.

4.4 Modification of Kalman filter

The flowchart of the Kalman filter algorithm is shown in Fig. 17 alongside equations (4.3.1)–(4.3.5). It can be adjusted depending on the system's complexity and purpose. Alfian Ma'arif et al. suggested adjusting the Kalman Filter algorithm to reduce sensor reading noise [32].

a) Predicting the state: In this stage, equation no. (4.3.1) are modified by providing the score $F_t = 1$ because there is no state transition. The adjusted equation is-

$$x_{t|t-1} = x_{t-1|t-1} \tag{4.4.1}$$

b) Predicting the error: Since $F_t = 1$, then (4.3.2) becomes

$$P_{t|t-1} = P_{t-1|t-1} + Q_t \tag{4.4.2}$$

c) Updating the state value: From (4.3.3), $H_t = 1$ since the sensor data that will be filtered is only consisted of one sensor reading. Hence, the equation can be written as-

$$x_{t|t} = x_{t|t-1} + K_t(y_t - x_{t|t-1}) \quad 4.4.3$$

d) Calculating the gain of Kalman: Since H_t=1, then (4.3.4) can be written as-

$$K_t = P_{t|t-1}(P_{t|t-1} + R)^{-1} \quad 4.4.4$$

e) Updating the error value: Since H_t=1, then (4.3.5) can be written as-

$$P_{t|t} = (1 - K_t)\, P_{t|t-1} \quad 4.4.5$$

After the adjustments are done, the Kalman Filter equation for reducing the noise of sensor reading can be rewritten. The Kalman gain (at eq. 4.4.4) is the weight given to the measurements and current-state estimate and can be "tuned" to achieve a particular performance.

In this paper, we opted to utilize the Kalman gain to replace the traditional step-size μ within the FxLMS algorithm. By doing so, we achieved a more flexible and adaptive step size that adjusts according to the characteristics of the signal components. This departure from a fixed step-size approach allows us to enhance the algorithm's responsiveness and effectiveness in managing diverse signals, enabling better noise reduction and signal extraction.

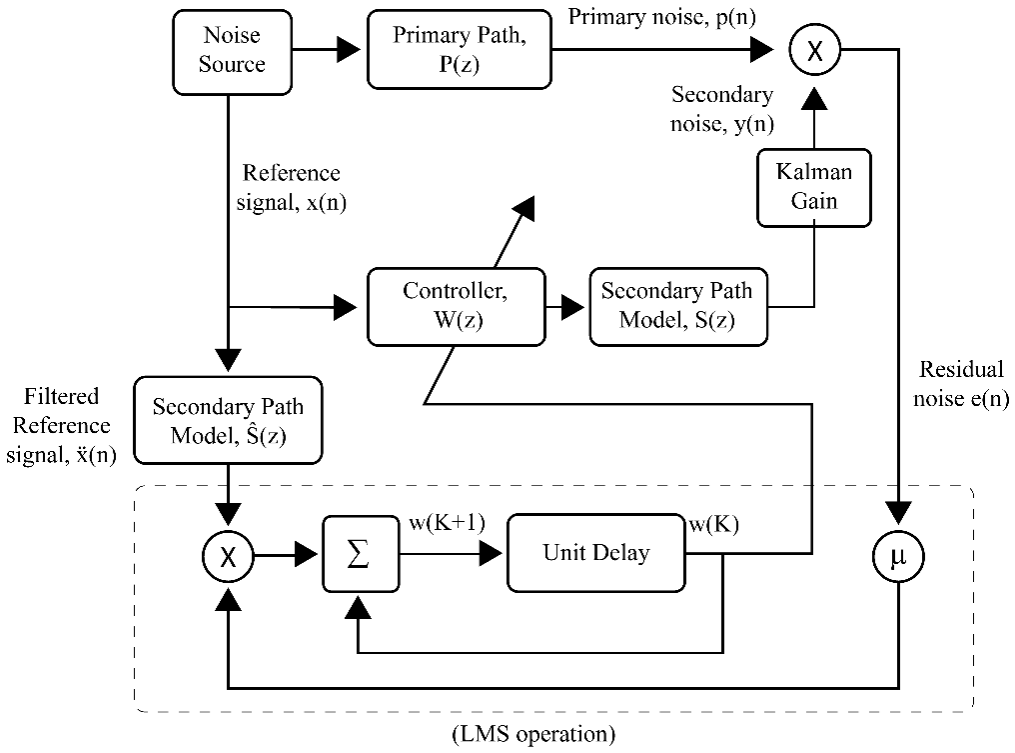


Figure 18: Proposed modified FxLMS algorithm.

In Figure 18. the flowchart of our modified FxLMS algorithm outlines the sequential steps undertaken to implement the Kalman gain as a replacement for the conventional step-size. This change depends on how good the measurements are and helps the method adjust better to different parts of the signal.

Imagine a situation where we have a signal, let's call it x(n), going from a source to a sensor through a fluid (like water or air), represented by P(z). But there's some unwanted noise, p(n), that the sensor picks up. To get rid of this noise, we create another kind of 'noise', y(n), using a controller called W(z). The idea is to make this new 'noise' interfere with the original signal x(n) in a way that cancels out the unwanted noise. This works best if the controller W(z) is like a copy of the fluid medium P(z) that the signal is passing through. Least Mean Square used to adjust the controller's settings. But here's the catch: there's also another fluid medium, S(z), between the controller and the sensor. We call this the secondary path. So, to make everything work correctly, we need to adjust for this secondary path as well, and estimate its effect, which we'll call $\hat{S}(z)$.

4.4.1 Values of Q and R

As we replaced step size, μ with Kalman-gain in the original FxLMS algorithm, we needed to declare some necessary variables to calculate Kalman-gain out of the noisy

signal. Alfian Ma'arif et al. proposed some adjustments [32] on Q (process noise covariance) and R (measurement noise covariance) during calculating Kalman-gain for sensor readings as well. The value of Q and R are chosen according to the system operations. Covariance Q and R states may not be in general observable but the measurements should be related to the states [33].

Q, the process noise covariance, contributes to the overall uncertainty. When Q is large, the Kalman Filter more closely tracks large changes in the data than when Q is small. The measurement noise covariance R determines how much information is used from the measurement. When R is large, the Kalman Filter considers the measurements to be inaccurate. The three images below visualize the positional data. The red lines represent the measurement data, the green lines are the estimated states. [34]

Q small R large | Q and R equal | Q large R small

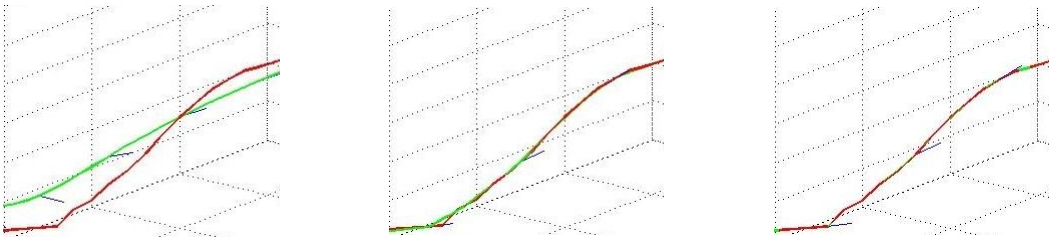


Figure 19: Relations between Q and R [34]

We need to balance between Q and R according to our needs. The vast majority of the noise estimation methods have been designed under the assumption of the uncorrelated state and measurement noise [35]. For example, if kalman used in tracking cars on a road, then the constant velocity model should be reasonably good, and the entries of Q should be small. Else if it is used tracking people's faces, they are not likely to move with a constant velocity, so the Q need to cranked up [36]. The ratio of R and Q values are crucial. The correct choice would be directly responsible for the filter performance and form the basic question of filter design [37].

In [32], The author performed denoising using a Kalman filter. They found that the greater the disparity between R and Q, the greater the mean error values. Furthermore, regardless of the values of R and Q, the mean error is roughly the same (as shown in Fig.19). Based on their findings, the mean error values for the parameters that most closely match the original data are between 40 and 55 (see table below).

Table 2: Ratio between R and Q and their yielding mean error.

No. of Analysis	Kalman Filter Parameter Value		R and Q Ratio	Mean Error
	R	Q		
1	1	1	1	26.0677
2	1	0.1	10	44.7392
3	1	0.01	100	53.4466
4	10	0.1	100	53.4541
5	100	0.1	1000	56.9959

Through their experimental findings, it was demonstrated that for effective signal denoising, the Kalman filter achieves optimal outcomes when the ratio between R and Q is maintained at 100:1. Consequently, we adopted the same 100:1 ratio for R and Q in our own operations.

Chapter 5

RESULTS AND DISCUSSION

5.1 Introduction

The proposed methodology is described in the previous chapter. The obtained result and finding of our study are introduced here with proper diagrams, pictures, and tables.

5.2 Comparison Criteria

In order to be able to compare the proposed algorithm with some adaptive filtering algorithms discussed in chapter 4, some characteristics must be defined which can be evaluated for each algorithm. For the comparison two performance criteria are used in the study: The rate of convergence and the signal-to-noise ratio (SNR) after filtering.

In many noise cancellation applications, a higher rate of convergence is desired. For satisfactory performance in noise cancellation, a high rate of convergence allows the algorithm to adapt rapidly to a stationary environment of unknown statistics, but the convergence speed is not independent from other performance characteristics [38]. There will be a trade-off in other performance criteria for an improved convergence rate and there will be a reduced convergence performance for an increase in other performance. In some applications, the system stability will drop when the rate of convergence is decreased, causing the system more likely to diverge rather than converging to a proper solution. To ensure a stable system, the parameters that affect the rate of convergence must be within certain limits. Each algorithm works on different methods for noise cancellation and reaches system stability in different ways.

5.2.1 The Rate of Convergence

The rate of convergence is defined as the number of adaptation cycles required for the algorithm to converge from some initial condition to its steady-state or close enough to an optimum, means how quickly an algorithm learns to minimize the difference between a desired signal and the signal it produces, like the optimum Wiener solution in the mean-square error sense [3]. This is crucial in applications like noise cancellation or signal enhancement, where timely adaptation matters.

To calculate the rate of convergence, one common approach is to track a metric like the Mean Square Deviation (MSD) or Mean Squared Error (MSE) between the desired and actual signals. By observing how this metric changes over iterations, you can infer the convergence speed. The Rate of Convergence can be quantified with the following equation:

Rate of Convergence $= -\frac{1}{N}\sum_{n=1}^{N}\ln(|e(n)|^2)$

Where: N is the number of iterations, $e(n)$ is the instantaneous error signal at iteration n.

Where N is the number of iterations and e(n) is the error signal at iteration n. The equation essentially calculates the average logarithm of squared error values.

However, the rate of convergence isn't uniform across algorithms. Depending on each algorithm, the rate of convergence is influenced by different factors like step size, input signals, and system characteristics impact it.

5.2.2 2.4.3 Signal-to-noise ratio SNR

The signal-to-noise ratio SNR is another important performance criterion in adaptive noise cancellation and describes the relationship between the strength of the input signal and the noise signal. The SNR is defined in (5.2.1) by the ratio of the signal power to the noise power and is often expressed in decibel.

$$SNR_{dB} = 10log_{10}\frac{S}{N} \qquad 5.2.1$$

In order to compare the different adaptive filtering algorithms in the efficiency of noise cancellation, the so-called improvement SNR level in (5.2.2) is used, which is the difference between the input and output SNR [39].

$$SNR_{imp} = SNR_{out} - SNR_{in} \qquad 5.2.2$$

Therefore, the SNR is calculated before and after applying the adaptive filter. The signal-to-noise ratio SNR in decibels is computed by the ratio of the summed squared magnitude of the signal to that of the noise. The input SNR is the ratio between the power of input signal and power of noise at the input

$$SNR_{in} = 10log_{10}\frac{\sum_n x(n)^2}{\sum_n v_1(n)^2} \quad 5.2.3$$

where x(n) is the noise-corrupted signal and v1 is the noise sequence. As there is no information about the noise signal, it is not possible to calculate exactly the input SNR, it can only be estimated from the sinusoid. The output SNR has to be higher than the input SNR, which indicates the success of noise removal. A lower value of the output SNR compared with the input SNR means that the filtering process introduces more noise instead of reducing noise. The output SNR is the ratio between the power of the filtered signal and power of the noise at output.

$$SNR_{out} = 10log_{10}\frac{\sum_n y(n)^2}{\sum_n e(n)^2} \quad 5.2.4$$

where y(n) is the output signal of the adaptive filter and e(n) is the noise signal. A large value of the output SNR is desirable, which indicates that the adaptive filter can remove a large amount of noise and is able to produce an accurate estimate of the desired signal. The signal-to-noise ratio increases when the output noise power decreases. Minimizing the output power causes the filtered signal to be perfectly noise-free [39].

5.3 Result and Analysis

First section of our code generates an input signal consisting of two sinusoids. The sample rate was 1000Hz. Frequency of two sinusoids are 50Hz and 40Hz respectfully. A random noise is generated with SNR value 20. After initial stage, various adaptive filter algorithms (LMS, NLMS, FxLMS, FxNLMS, and proposed modified FxLMS) are used to the noisy signal.

An input signal is generated which is a composition of two sinusoidal waves. The signal is sampled at a rate of 1000Hz (samples per second), ensuring that the signal is well-represented in the digital domain. The frequencies of the two sinusoids are chosen as 50Hz and 40Hz, resulting in a complex signal that exhibits both of these frequencies. Typical frequencies associated with underwater acoustics are between 10 Hz and 1 MHz. The propagation of sound in the ocean at frequencies lower than 10 Hz is usually not possible without penetrating deep into the seabed, whereas frequencies above 1 MHz are rarely used because they are absorbed very quickly [40].

To simulate real-world scenarios, random noise is generated. The Signal-to-Noise Ratio (SNR) value is set to 20dB, indicating the ratio of the signal power to the noise power. This creates a noisy version of the original signal, which is essential for testing the performance of the adaptive filter algorithms in a noisy environment.

After this initial setup, several adaptive filter algorithms are applied alongside proposed modified FxLMS algorithm. Figure (a) shows the original signal and a noisy version of it.

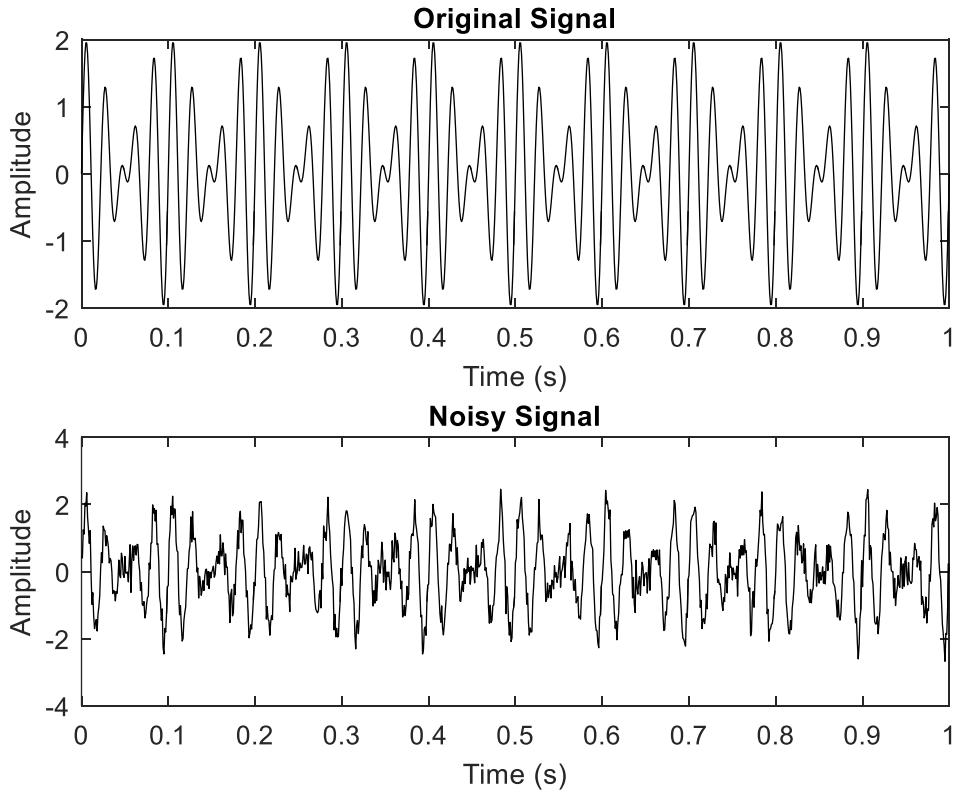
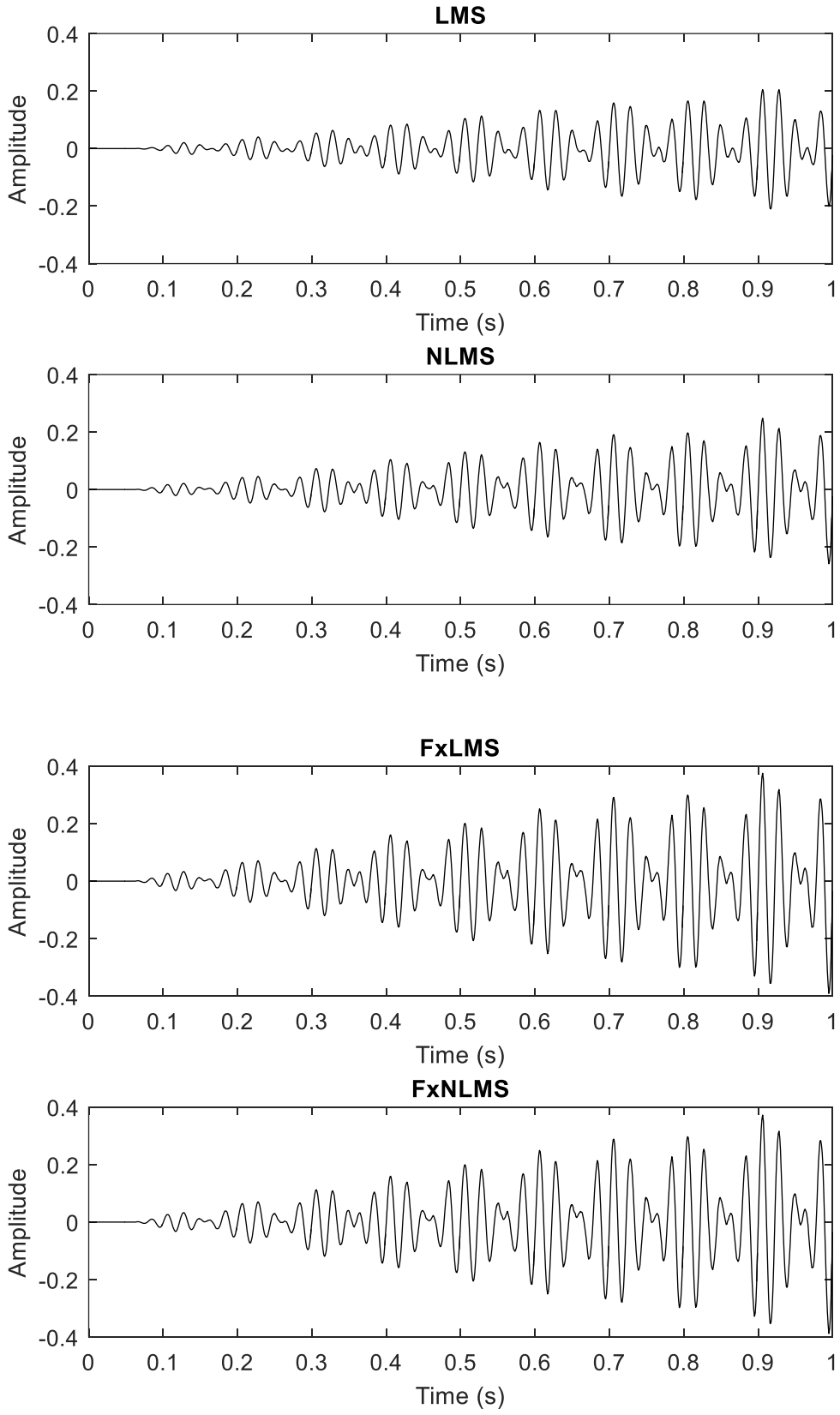


Figure 20: Matlab simulated signal and noise.

Several adaptive filters (LMS, NLMS, FxNLMS, FxLMS and modified FxLMS) are executed on the noisy input signal. These algorithms have been employed to enhance the quality of the signal by attenuating the unwanted noise components. After processing the input signal using each algorithm, the resulting output signals are visualized in figure (b).


LMS
Amplitude
Time (s)
NLMS
Amplitude
Time (s)
FxLMS
Amplitude
Time (s)
FxNLMS
Amplitude
Time (s)

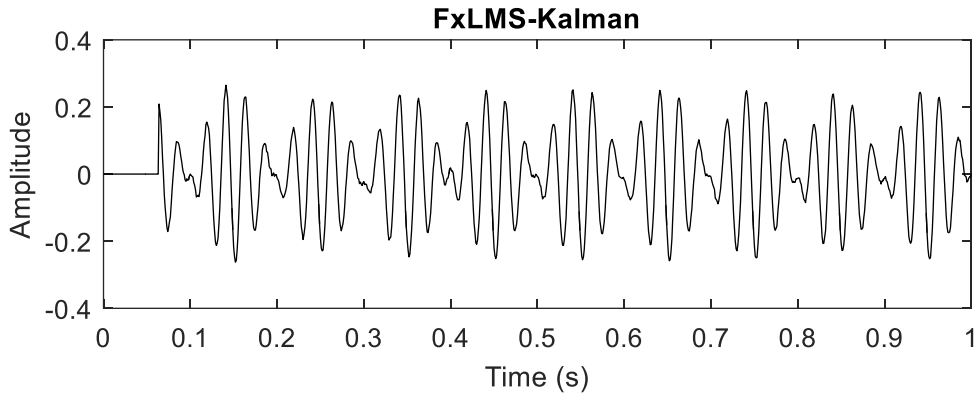


Figure 21: Simulation of different adaptive filtering algorithm and modified FxLMS.

This visualization provides a direct comparison of the outputs produced by the different adaptive filter algorithms. Each algorithm's output is represented as a separate subplot within figure (b), allowing for an easy and immediate assessment on which algorithm produces the cleanest and most accurate output in terms of noise reduction and signal preservation.

The convergence rates of each adaptive algorithm are presented below.

Table 3: Rate of Convergence of various Adaptive Filters.

Algorithm	**Rate of convergence**
Least Mean Square (LMS)	0.9022
Normalized LMS (NLMS)	0.8841
Filtered X LMS (FxLMS)	0.82741
Filtered X NLMS (FxNLMS)	0.8287
Proposed FxLMS	**1.007**

The Modified Filtered-x Least Mean Squares (FxLMS) algorithm showcases a notable advantage in terms of convergence rate, enabling it to swiftly adapt to dynamic and noisy systems, resulting in a more efficient noise reduction process. Unlike the other adaptive filter algorithms in consideration, the Modified FxLMS algorithm demonstrates a remarkable ability to rapidly adjust its filter coefficients during the initial iterations. a separate subplot within figure (b) are presented to visualize the comparisons of rate of convergence.

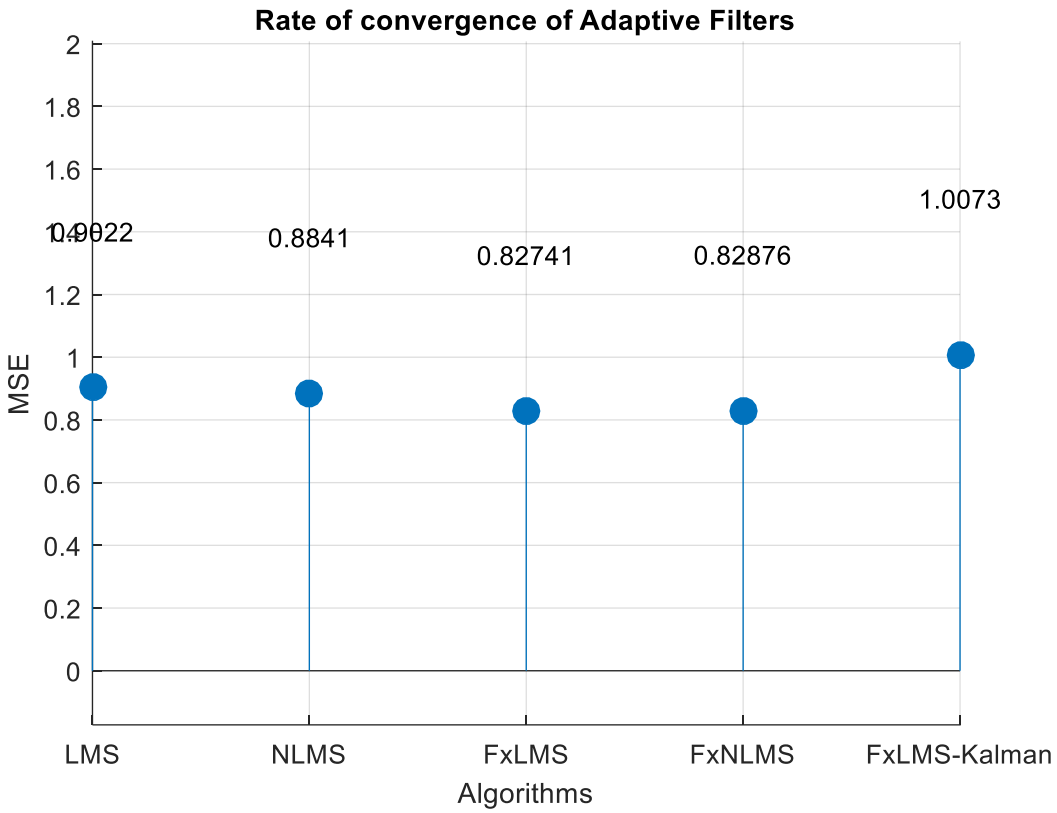


Figure 22: Rate of Convergence of various Adaptive Filters

In the attached table and graph below, displays the SNR performance of various adaptive algorithms including LMS, NLMS, FxLMS, FxNLMS and proposed modified FxLMS algorithm. Notably, the modified algorithm exhibits a distinctive advantage in terms of Signal-to-Noise Ratio (SNR) enhancement.

Table 4: SNR Comparisons of various Adaptive Filters.

Algorithm	**SNR**
Least Mean Square (LMS)	7.3912
Normalized LMS (NLMS)	8.0765
Filtered X LMS (FxLMS)	11.3217
Filtered X NLMS (FxNLMS)	11.2546
Proposed FxLMS	**11.8788**

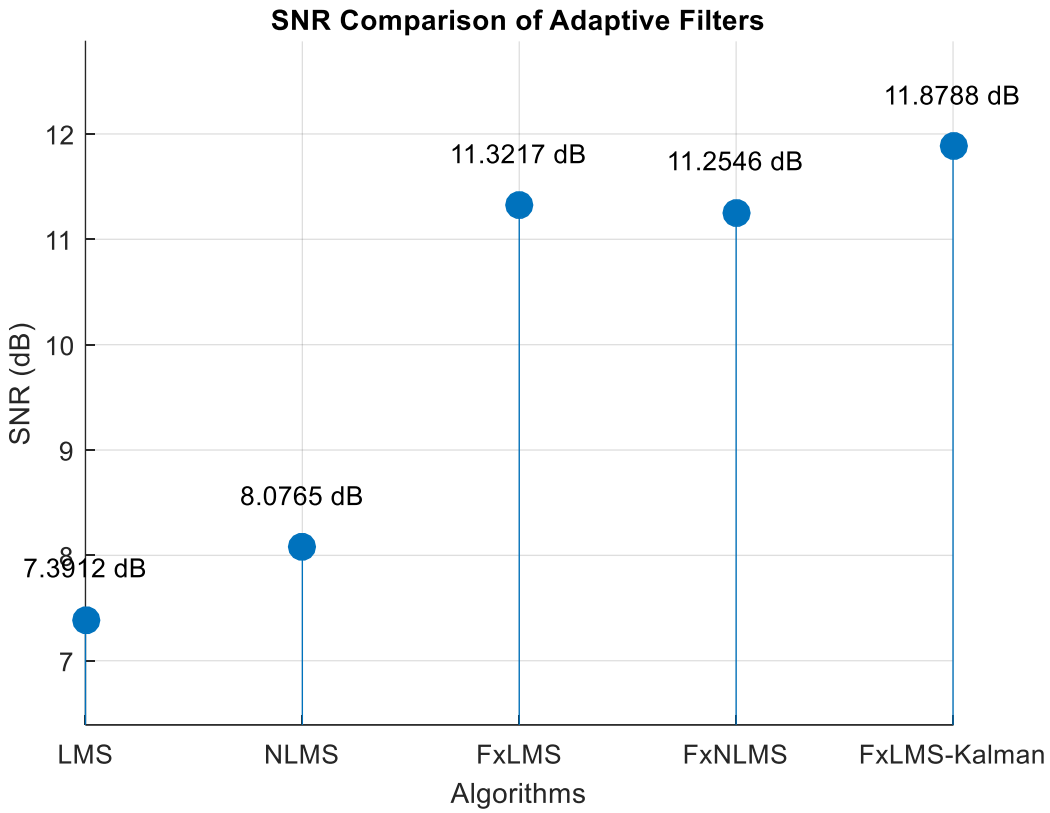


Figure 23: SNR Comparisons of various Adaptive Filters.

Among the algorithms showcased, modified algorithm stands out by consistently delivering a higher SNR value. While these algorithms have their own merits, our innovation seems to have harnessed a unique capability for optimizing SNR, which holds great promise for applications demanding superior noise reduction and signal fidelity.

These results not only validate the potential of our modified algorithm but also open doors for further exploration and refinement. The apparent superiority in convergence rate and SNR performance signifies a significant stride towards enhancing adaptive algorithms in practical contexts where noise mitigation is pivotal.

Chapter 6

CONCLUSION

6.1 Summary of thesis

This thesis embarked on a comprehensive exploration and comparative analysis of various adaptive algorithms for system identification and noise cancellation. The aim was to identify an algorithm that not only showcases impressive convergence rates but also excels in enhancing the signal-to-noise ratio (SNR) of the system under consideration. Through rigorous theoretical examination and simulation-based experimentation, the study has successfully achieved these objectives and has shed light on the promising potential of the modified FXLMS algorithm.

The comparative study revealed crucial insights into the strengths and limitations of several popular adaptive algorithms, including the LMS, NLMS, RLS, and APA algorithms. Each algorithm showcased unique attributes in terms of convergence speed and robustness, laying the groundwork for a comprehensive understanding of their applicability in different scenarios.

However, upon modifying the FXLMS algorithm, it became evident that the proposed enhancements brought forth a remarkable leap in performance. The simulations conducted under matlab simulated noise conditions consistently demonstrated that the modified FXLMS algorithm outperformed its counterparts. The algorithm not only achieved faster convergence rates but also exhibited exceptional noise cancellation capabilities, leading to significantly improved SNR values. This superior performance can be attributed to the carefully designed modifications that harness the inherent strengths of the FXLMS framework while addressing its limitations.

The modified FXLMS algorithm has the potential to enhance the efficiency and effectiveness of active noise control, thereby contributing to improved user experiences and system performance.

6.2 Future Development

It is worth noting that while this thesis has made significant strides in evaluating and enhancing adaptive algorithms, further research avenues remain open. Exploring variations of the modified FxLMS algorithm, investigating its performance across a broader

range of scenarios, and delving deeper into its theoretical underpinnings could provide valuable insights for future studies.

In conclusion, by looking at theories, making changes to how the algorithm works, and testing it in simulations, we have discovered that the modified FXLMS algorithm has a lot of potential. This research adds to the growing knowledge about how to make filters that adjust automatically and sets the groundwork for new ways to improve how signals are processed and systems are understood.

As the field of these automatic algorithms keeps growing, the things we've learned here can help other scientists and people who use these algorithms. They can use this knowledge to create even better filters for lots of different uses. This will make signals clearer, lower the effects of unwanted noise, and make systems work better overall.

References

[1] J. G. Bernard widrow, "Adaptive Noise Cancelling: Principles and Aplications," *IEEE proceeding*, vol. 63, no. 12, 1975.

[2] M. M. a. K. R. Dewasthale, "Acoustic Noise Cancellation Using Adaptive Filters: A Survey," in *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies*, 2014.

[3] S. S. Haykin, Adaptive filter theory, Pearson Education India, 2008.

[4] J. J. a. J. J. M. Rajesh., "Active noise reduction of automotive HVAC system using filtered LMS [Part-1 sound measurement]," in *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, 2012.

[5] Z. Y. Y. F. Ma WK, "A fast recursive-least-squares adaptive notch filter and its applications to biomedical signals," *Medical & Biological Engineering & Computing*, vol. 37, no. 1, pp. 99-103, 1999.

[6] P. S. Ramirez, "Adaptive Lattice-Based RLS Algorithms," in *Adaptive Filtering: Algorithms and Practical Implementation*, Boston, MA, Springer US, 2002, pp. 235-286.

[7] P. S. Ramirez, "QR-Decomposition-Based RLS Filters," *The Kluwer International Series in Engineering and Computer Science*, vol. 694, pp. 309-359, 2002.

[8] V. a. J. S. N. a. Y. T. Madravam, "An Analysis of LMS, NLMS and RLS Filters with Application to DOA Tracking," in *2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2020.

[9] R. N. Y. Manish D. Sawale, "Improved Normalized Least Mean Square Algorithm Using Past Weight Vectors and Regularization Parameter," in *Communications in Computer and Information Science*, Berlin, 2011.

[10] P. a. G. J. Lopes, "New Normalized LMS Algorithms Based on the Kalman Filter," in *IEEE International Symposium on Circuits and Systems*, 2007.

[11] I. a. A. W. Ardekani, "FxLMS-based Active Noise Control : A Quick Review," in *Proceedings of 2011 Asia Pacific Signal and Information Processing Association Annual (APSIPA) Summit and Conference*, Xi'an, China, 2011.

[12] M. H. Hayes, Statistical Digital Signal. Processing and Modeling, John Wiley & Sons, 1996.

[13] S. M. a. D. R. M. V. 4. Kuo, Active noise control systems, vol. 4, New York: Wiley, 1996.

[14] P. A. N. S. J. Elliott, The active control of sound, 1992.

[15] P. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 2nd Edition, New York.: Kluwer Academic Publishers, 2008.

[16] M. a. V. V. a. L. T. Makundi, "Closed-form design of tunable fractional-delay allpass filter structures," in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems*, 2001.

[17] R. a. S. G. a. D. B. a. g. s. Kaur, "Acoustic Echo Cancellation Using Modified Variable Step Size Least Mean Square Adaptive Algorithm," in *Conference: International conference on computer applications*, 2012.

[18] S. Qureshi, "Adaptive equalization," *Proceedings of the IEEE,* vol. 73, no. 9, pp. 1349-1387, 1985.

[19] S. Vaseghi, "Digital Filters," in *Multimedia Signal Processing: Theory and Applications in Speech, Music and Communications,*, John Wiley & Sons,, 2007, pp. 111-117.

[20] S. D. Stearns, "Error surfaces of recursive adaptive filters," *IEEE Transactions on Circuits and Systems,* vol. 28, no. 6, pp. 603-606, 1981.

[21] MathWorks, "Compare RLS and LMS Adaptive Filter Algorithms," [Online]. Available: https://www.mathworks.com/help/dsp/ug/compare-rls-and-lms-adaptive-filter-algorithms.html. [Accessed 15 01 2023].

[22] D. K. G. V. K. G. A. Tajammul, "Analysis of Noise Signal Cancellation using LMS, NLMS and RLS Algorithms of Adaptive Filter," *Advanced Research in Electrical and Electronic Engineering,* vol. 2, pp. 50-53, 2015.

[23] D. R. Morgan, "An analysis of multiple correlation cancellation loops with a filter in the auxiliary path," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vols. ASSP-28, pp. 454-467, 1980.

[24] M. Barkat, Signal Detection and Estimation, Second Edition, Artech, 2005.

[25] G. B. Greg Welch, An Introduction to the Kalman Filter, Chapel Hill, United States: University of North Carolina, 1995.

[26] D. F. Elliott, Handbook of Digital Signal Processing, Orlando: Academic Press, 1987.

[27] G. M. Siouris, An Engineering Approach to Optimal Control and Estimation Theory, Wiley and Sons, 1996.

[28] V. K. I. a. S. M. K. D. G. Manolakis, Statistical and Adaptive Signal Processing: Spectral Estimation, Signal Modeling, Adaptive Filtering and Array Processing, New York: McGraw-Hill, 2000.

[29] S. B. D. S. F. a. K. P. Y. Pei, "An elementary introduction to Kalman filtering," *Communications of the ACM,* vol. 62, no. 11, p. 122–133, 2019.

[30] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Fluids Engineering, Transactions of the ASME,* vol. 82, no. 1, pp. 35-45, 1960.

[31] N. K. R. A. M. K. N. Kumari, "Use of Kalman Filter and Its Variants in State Estimation: A Review," in *Artificial Intelligence for a Sustainable Industry 4.0,* Springer International Publishing, 2021, pp. 213-230.

[32] I. A. A. N. R. I. A. Alfian Ma'arif, "Kalman Filter for Noise Reducer on Sensor Readings," *Signal and Image Processing Letters,* vol. 1, no. 2, pp. 50-60, 2019.

[33] M. R. Ananthasayanam, "Tuning of the Kalman Filter Using Constant Gains," in *Introduction and Implementations of the Kalman Filter*, IntechOpen, 2018.

[34] D. K. Buckl, "Sensor Fusion using the Kalman Filter," TUM: The Entrepreneurial University, 2005.

[35] J. a. S. O. a. K. O. a. H. J. Dunik, "Noise covariance matrices in state-space models: A survey and comparison of estimation methods—Part I," *International Journal of Adaptive Control and Signal Processing*, vol. 31, 2017.

[36] Cyberdyne, "Kalman filter in computer vision: the choice of Q and R noise covariances," 2014. [Online]. Available: https://stackoverflow.com/questions/21245167/kalman-filter-in-computer-vision-the-choice-of-q-and-r-noise-covariances. [Accessed June 2022].

[37] S. Srinivasan, "The Kalman Filter: An algorithm for making sense of fused sensor insight," Towards Data Science, 2018.

[38] S. Jimaa, "Convergence Evaluation of a Random Step-Size NLMS Adaptive Algorithm in System Identification and Channel Equalization," in *Adaptive Filtering*, InTech, 2011.

[39] M. H. A.-U. Noman Q. Al-Naggar, "Performance of Adaptive Noise Cancellation with Normalized Last-Mean-Square Based on the Signal-to-Noise Ratio of Lung and Heart Sound Separation," *Journal of Healthcare Engineering*, vol. 2018, 2018.

[40] H. M. P. D. W. Muhammad Zainuddin, Signal processing for marine acoustic and dolphin using matlab, LAP LAMBERT Academic Publishing, 2016.

[41] K. a. A. T. Mayyas, "Comparative Study of the Filtered-X Lms and Lms Algorithms With Undermodelling Conditions," *International Journal of Modelling and Simulation*, vol. 22, pp. 159-166, 2002.

[42] B. P. &. P. P. HIRAVE, "FXLMS Algorithm for Feed Forward Active Noise Cancellation," in *First International Conference on Advances in Computer, Electronics and Electrical Engineering - CEEE*, 2012.

[43] S. G. a. V. K. Gupta, "A Review on Filtered-X LMS Algorithm," *International Journal of Signal Processing Systems* , vol. 4, no. 2, pp. 172-176, 2016.

[44] W. A. Iman ArdekaniIman, "FxLMS-based active noise control: A quick review," in *Asia Pacific Signal and Information Processing Association Annual (APSIPA) Summit and Conference*, Xi'an, China, 2011.

[45] S. a. K. Y. a. K. N. Makino, "Exponentially weighted stepsize NLMS adaptive filter based on the statistics of a room impulse response," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 101-108, p. 1, 1993.

[46] M. M. Sondhi, "An Adaptive Echo Canceller," *Bell System Technical Journal*, vol. 46, no. 3, pp. 497-511, 1967.

Appendix

Appendix I: Implementation code

```
% Generate input signal
fs = 1000; % Sample rate (Hz)
t = 0:1/fs:1; % Time vector (1 second)
f1 = 50; % Frequency of sinusoid 1 (Hz)
f2 = 40; % Frequency of sinusoid 2 (Hz)
x = sin(2*pi*f1*t) + sin(2*pi*f2*t); % Input signal

% Add noise to the input signal
SNR = 10; % Signal-to-Noise Ratio (dB)
noise = randn(size(x)); % Gaussian noise
noise = noise / norm(noise); % Normalize noise
noise_power = 10^(-SNR/20) * norm(x) / norm(noise); %
Calculate noise power
x_noisy = x + noise_power * noise; % Noisy input signal

% Apply LMS algorithm
filter_order = 64; % Filter order
mu_lms = 0.01; % LMS step size
[y_lms, ~] = lms(x_noisy, x, filter_order, mu_lms); %
Apply LMS

% Apply NLMS algorithm
mu_nlms = 0.01; % NLMS step size
[y_nlms, ~] = nlms(x_noisy, x, filter_order, mu_nlms); %
Apply NLMS

% Apply FxLMS algorithm
mu_fx = 0.01; % FxLMS step size
[y_fx, ~] = fxlms(x_noisy, x, filter_order, mu_fx); %
Apply FxLMS

% Apply FxNLMS algorithm
mu_fxn = 0.01; % FxNLMS step size
[y_fxn, ~] = fxnlms(x_noisy, x, filter_order, mu_fxn); %
Apply FxNLMS

% Apply fxlms_kalman algorithm
kalman_gain = 100;
[y_fxkal, ~] = fxlms_kalman(x_noisy, x, filter_order,
kalman_gain); % Apply FxNLMS

%% Calculate MSE

% Calculate MSE for LMS
mse_lms = mean((y_lms - x').^2);

% Calculate MSE for NLMS
```

```
mse_nlms = mean((y_nlms - x').^2);

% Calculate MSE for FxLMS
mse_fx = mean((y_fx - x').^2);

% Calculate MSE for FxNLMS
mse_fxn = mean((y_fxn - x').^2);

% Calculate MSE for FxLMS-Kalman
mse_fxkal = mean((y_fxkal - x').^2);

% Plot signals

% plot of Original Signal
figure(1)
subplot(2,1,1)
plot(t, x,'k');
title('Original Signal');
xlabel('Time (s)');
ylabel('Amplitude');
%axis([0 1000 -3 3])
subplot(2,1,2);
plot(t, x_noisy,'k');
title('Noisy Signal');
xlabel('Time (s)');
ylabel('Amplitude');

figure (2);
subplot(5,1,1);
plot(t, y_lms,'k');
title('LMS');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(5,1,2);
plot(t, y_nlms,'k');
title('NLMS');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(5,1,3);
plot(t, y_fx,'k');
title('FxLMS');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(5,1,4);
plot(t, y_fxn,'k');
title('FxNLMS');
xlabel('Time (s)');
ylabel('Amplitude');
```

```
subplot(5,1,5);
plot(t, y_fxkal,'k');

SNR_fx = snr_norm(x, y_fxkal);
title(['FxKalman Output (SNR: ', num2str(SNR_fx), '
dB)']);

xlabel('Time (s)');
ylabel('Amplitude');

figure(3)
subplot(2,1,1)
plot(t, y_fx,'k');
title('FxLMS');
xlabel('Time (s)');
ylabel('Amplitude');
%axis([0 1000 -3 3])
subplot(2,1,2);
plot(t, y_fxkal,'k');
title('FxLMS-Kalman');
xlabel('Time (s)');
ylabel('Amplitude');


% Print rate of convergence (MSE) for each algorithm
disp(['MSE for LMS: ', num2str(mse_lms)]);
disp(['MSE for NLMS: ', num2str(mse_nlms)]);
disp(['MSE for FxLMS: ', num2str(mse_fx)]);
disp(['MSE for FxNLMS: ', num2str(mse_fxn)]);
disp(['MSE for FxLMS-Kalman: ', num2str(mse_fxkal)]);

% Calculate SNR for each algorithm
SNR_lms = snr_norm(x, y_lms);
SNR_nlms = snr_norm(x, y_nlms);
SNR_fx = snr_norm(x, y_fx);
SNR_fxn = snr_norm(x, y_fxn);
SNR_fxkal = snr_norm(x, y_fxkal);

% Print SNR for each algorithm
disp(['SNR for LMS: ', num2str(SNR_lms), ' dB']);
disp(['SNR for NLMS: ', num2str(SNR_nlms), ' dB']);
disp(['SNR for FxLMS: ', num2str(SNR_fx), ' dB']);
disp(['SNR for FxNLMS: ', num2str(SNR_fxn), ' dB']);
disp(['SNR for FxLMS-Kalman: ', num2str(SNR_fxkal), '
dB']);


% Create a waveform plot for SNR values
algorithms = {'LMS', 'NLMS', 'FxLMS', 'FxNLMS', 'FxLMS-
Kalman'};
```

```
snr_values = [SNR_lms, SNR_nlms, SNR_fx, SNR_fxn,
SNR_fxkal];

figure;
hold on;

% Plot stem markers
stem(1:length(algorithms), snr_values, 'filled',
'MarkerSize', 10);

% Annotate SNR values above the markers
for i = 1:length(algorithms)
    text(i, snr_values(i) + 0.5, [num2str(snr_values(i)),
' dB'], 'HorizontalAlignment', 'center');
end

set(gca, 'XTick', 1:length(algorithms), 'XTickLabel',
algorithms);
title('SNR Comparison of Adaptive Filters');
xlabel('Algorithms');
ylabel('SNR (dB)');
grid on;
ylim([min(snr_values) - 1, max(snr_values) + 1]);

hold off;

% Print SNR values
disp('SNR values:');
for i = 1:length(algorithms)
    disp([algorithms{i}, ': ', num2str(snr_values(i)), '
dB']);
end

% Create a waveform plot for MSE values
algorithms = {'LMS', 'NLMS', 'FxLMS', 'FxNLMS', 'FxLMS-
Kalman'};
mse_values = [mse_lms, mse_nlms, mse_fx, mse_fxn,
mse_fxkal];

figure;
hold on;

% Plot stem markers
stem(1:length(algorithms), mse_values, 'filled',
'MarkerSize', 10);

% Annotate MSE values above the markers
for i = 1:length(algorithms)
    text(i, mse_values(i) + 0.5, [num2str(mse_values(i))],
'HorizontalAlignment', 'center');
end
```

```
set(gca, 'XTick', 1:length(algorithms), 'XTickLabel',
algorithms);
title('Rate of convergence of Adaptive Filters');
xlabel('Algorithms');
ylabel('MSE');
grid on;
ylim([min(mse_values) - 1, max(mse_values) + 1]);

hold off;

% Print MSE values
disp('MSE values:');
for i = 1:length(algorithms)
    disp([algorithms{i}, ': ', num2str(mse_values(i))]);
end

%######### End of Code ######
```

Appendix II: Publication

Rahman, Md. Mostafizur and Sarker, Sumonto and Islam, Md. Mehedi, "Underwater Active Noise Cancellation Combining Kalman Filter with FxLMS", *Journal of Information Hiding and Multimedia Signal Processing*, 2023, Vol. 14, No. 01, pp. 31-40, https://doi.org/10.5281/zenodo.7741880