Variable Step-Size Adaptive Algorithms for Acoustic Echo Cancellation in Hands-Free Portable Devices

submitted by

Rania Eldeeb, B.A.Sc

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science
in
Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Faculty of Engineering and Design
Department of Systems and Computer Engineering

Carleton University
Ottawa, Ontario

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-94265-9

Our file Notre référence
ISBN: 978-0-494-94265-9

ABSTRACT

Hands-free communication is becoming more ubiquitous in many of today's practical devices such as mobile phones, tablets, teleconferencing systems, Voice over Internet Protocol (VoIP) conferencing systems, hearing aids, Bluetooth® accessories, and hands-free car kits. Therefore there is an increased demand for more robust acoustic echo cancellation and noise cancellation to provide higher quality conversations. Acoustic echo cancellers use adaptive filters to identify the acoustic echo path impulse response. Non-stationarity of the acoustic environment and long lengths of the echo paths impose challenges on the performance of adaptive algorithms.

This thesis proposes new techniques to improve the performance of adaptive algorithms in the context of acoustic echo cancellation by varying the step-size parameter in different ways. Two techniques are proposed and incorporated into existing variable step-size Normalized Least-Mean-Square (NLMS) based algorithms: the gradient-induced technique and the thresholding technique. In the gradient-induced technique, an estimate of the gradient of the coefficients' adaptation is incorporated into the filter coefficients' update rule. It is shown that the proposed technique improves the overall performance of selected Proportionate NLMS (PNLMS) based algorithms with a modest increase in complexity. It also allows algorithms to achieve comparable performance with less overall complexity when compared to more complex algorithms. The thresholding technique is incorporated into the NLMS algorithm and the Non-Parametric Variable Step-Size NLMS (NPVSS-NLMS) algorithm forming new classes of thresholded NLMS-based algorithms. Simulations show that the proposed thresholded algorithms outperform their original counterparts in terms of convergence speed and tracking capability with a modest increase in computations.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Rafik Goubran, for his invaluable guidance, discussions and comments, and continuous encouragement throughout the course of this thesis. I would also like to thank Payam Moradshahi, and Trevor Burton for their assistance with the experimental setup and data acquisition. Special thanks go to Dr. Vilas Joshi for his constructive comments, feedback, and suggestions. I would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC), and Carleton University for their financial support during this research.

I am mostly grateful to my amazing family; my mother, Dr. Nashwah Abdel-Khalik, for her unconditional love and endless support, my father, Dr. Fekry Eldeeb, for his inspiration and for ingraining me with the true value of education, my lovely sisters, Rowayda and Radia, and brother-in-law, Mohamed Beshr, for their patience and faith in me. Finally, I would like to sincerely thank all my friends, especially, Heba Eid, Sara El-Kady, and Hazem Ezzat for giving me joy, and motivation while working on this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADC: Analog-to-Digital Converter

AEC: Acoustic Echo Canceller

AMPNLMS: Adaptive Mu-law Proportionate Normalized Least-Mean-Squares

ASCMPNLMS: Adaptive Sparseness Controlled Mu-law Proportionate Normalized Least-Mean-Squares

ATNLMS: Adaptive Thresholded Normalized Least-Mean-Squares

BPF: Band Pass Filter

DFT: Discrete Fourier Transform

EPIR: Echo Path Impulse Response

ERLE: Echo Return Loss Enhancement

FIFO : First-In, First-Out

FIR: Finite Impulse Response

g-PNLMS: gradient-induced Proportionate Normalized Least-Mean-Squares

GGPNLMS: Generalized Gradient Proportionate Normalized Least-Mean-Squares

GUI: Graphical User Interface

HPF: High Pass Filter

IPNLMS: Improved Proportionate Normalized Least-Mean-Squares

EPIR: Impulse Response

IIR: Infinite Impulse Response

ISS: Individual Step-Size

LRM: Loudspeaker-Room-Microphone

LMS: Least-Mean-Squares

LPF: Low Pass Filter

mATNLMS: masked Adaptive Thresholded Normalized Least-Mean-Squares

MPNLMS: Mu-law Proportionate Normalized Least-Mean-Squares

mTNLMS: masked Thresholded Normalized Least-Mean-Squares

NLMS: Normalized Least-Mean-Squares

NPVSS: Non-Parametric Variable Step-Size

PNLMS: Proportionate Normalized Least-Mean-Squares

SCIPNLMS: Sparseness Controlled Improved Proportionate Normalized Least-Mean-Squares

SCMPNLMS: Sparseness Controlled Mu-law Proportionate Normalized Least-Mean-Squares

SCPNLMS: Sparseness Controlled Proportionate Normalized Least-Mean-Squares

SNR: Signal-to-Noise Ratio

TNLMS: Thresholded Normalized Least-Mean-Squares

VISS: Variable Individual Step-Size

VSS: Variable Step-Size

WGN: White Gaussian Noise

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 Overview

In a hands-free communication system, the far-end signal propagating from the loudspeaker is reflected from different surfaces and objects in the Loudspeaker-Room-Microphone (LRM) environment forming an acoustic echo signal [1]. The acoustic echo signal is an attenuated, delayed and distorted replica of the original far-end signal. This echo signal is picked up by the microphone and transmitted back to the far-end speaker disrupting the conversation. An acoustic echo canceller (AEC) is required to suppress the undesired echoes. A general scheme for echo cancellation is depicted in Figure 1.1. In full-duplex hands-free communication systems, an AEC is used at each end of the conversation such that speakers at both ends of a conversation can speak simultaneously.

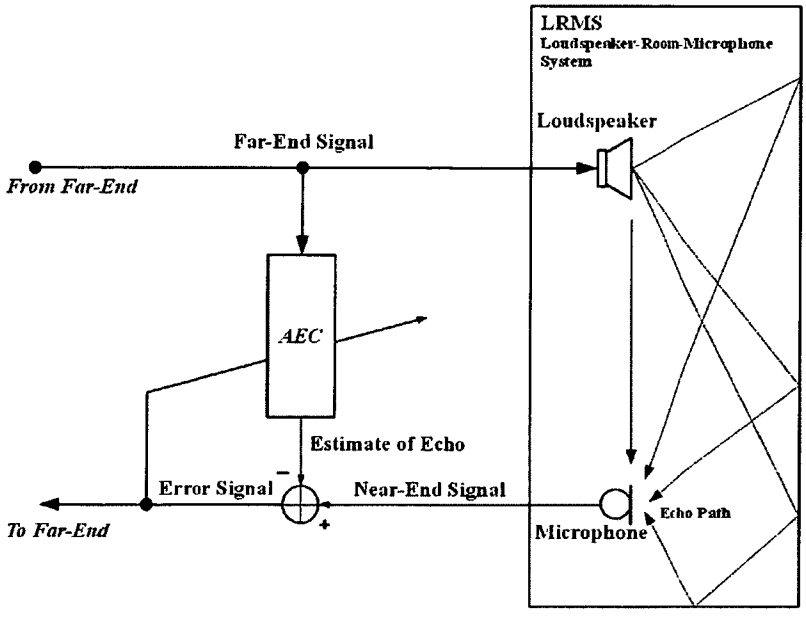


Figure 1.1: General scheme for acoustic echo cancellation.

The main component of acoustic echo cancellers is the adaptive filter. The adaptive filter is used to linearly model and estimate the unknown transfer function of the acoustic echo path referred to as the acoustic echo path impulse response (EPIR). This estimated impulse response is used to produce a replica of the echo signal which is then subtracted from the signal picked up by the microphone before it is transmitted back to the far-end speaker. Many algorithms have been proposed in the literature to improve the performance of adaptive filters which is limited by long lengths of acoustic echo paths, non-stationarities of the acoustic environment, and background noise [2]. One approach to improve the performance that is intensively researched is varying the adaptation step-size of the algorithm, which has an effect on the overall performance.

Various methods of varying the adaptation step-size have been proposed in the literature [3-23] all of which aim at achieving a high level of echo cancellation with a fast convergence speed under stationary conditions and fast tracking capability under non-stationary conditions. These algorithms present different ways to vary the adaptation step-size parameter. Some algorithms vary the step-size equally for all the filter coefficients [3-11], some assign a different constant step-size for each coefficient [12,13], and others vary a step-size for each coefficient differently [14-23]. However, each algorithm provides some trade-off between the amount of echo cancellation attainable and the computational complexity.

1.2 Problem Statement

Echo cancellation is basically a "system identification" problem, therefore one of the main goals for improving or developing new adaptive algorithms for AEC is the ability to accurately estimate the impulse response and rapidly adapt to its variations (usually few seconds).

The main focus of this research is to investigate the different methods of varying the step-size of adaptive algorithms, and to develop new techniques that provide at least the same amount of echo cancellation but with faster echo cancellation, i.e. faster convergence speed and higher tracking capability, under highly non-stationary environment conditions for hands-free portable devices. The developed techniques aim to improve the performance with a modest increase in computational complexity. Convergence is defined as the ability of the adaptive algorithm to adapt the filter coefficients from zero to their estimated values. Tracking capability in the context of this research is defined as the ability of the adaptive algorithm to adapt the filter coefficients to follow the changes in the impulse response under highly non-stationary conditions, i.e. highly variant impulse response. This is simulated by switching between impulse responses rather than having continuous changes over time.

1.3 Objectives

The main objectives of this thesis are:

- Investigate the performance of the different variable step-size adaptive algorithms in the literature for acoustical echo cancellation by using impulse responses of hands-free portable devices and use the results as a start point for research

- Develop new techniques for varying the step-size in adaptive algorithms
- Develop improved versions for state-of-the-art variable step-size algorithms using the new techniques
- Develop new variable step-size algorithms using the new techniques to achieve faster acoustic echo cancellation with modest increase in computational requirements
- Compare the performance of proposed and existing algorithms under stationary acoustic conditions (static/time-invariant echo path impulse response)
- Compare the performance of proposed and existing algorithms under non-stationary acoustic conditions (dynamic/time-variant echo path impulse response)

1.4 Thesis Contributions

The main contributions of this thesis work are:

- Measure acoustic echo path impulse responses of a commercial smartphone operating in hands-free mode under conditions that mimic realistic non-stationary scenarios.
- Compare the performance of various variable step-size adaptive algorithms that exist in the literature for realistic impulse response variation scenarios.
- Introduce the gradient-induced technique, a new variable step-size technique, and incorporate it in existing variable step-size adaptive algorithms in the literature to

provide improved performance in highly non-stationary environments with a relatively modest increase in computational complexity.

- Propose a modified variable step-size adaptive algorithm, which when combined with the gradient-induced technique, provides the best overall performance for acoustic echo path impulse responses.

- Introduce a new class of variable step-size algorithms, the Thresholded-Normalized Least-Mean-Squares algorithms, which are based on a thresholding technique that is incorporated in existing adaptive algorithms in the literature. This class of algorithms provides faster and higher acoustic echo cancellation with modest increase in computational requirements.

- Compare the performances of proposed and existing algorithms under different static and dynamic acoustical conditions using both measured and simulated impulse responses.

- Provide measurement and simulation environments by designing MATLAB GUIs to facilitate the capture and study of impulse responses and the comparison of the performances of various adaptive algorithms respectively.

1.5 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 reviews the background on acoustic echo cancellation, and provides a literature review on the various types of variable step-size adaptive algorithms that are most frequently used in echo cancellation. Chapter 3 discusses the experimental setup required to acquire the acoustic echo path

impulse responses of a smartphone operating in hands-free mode. The acquired impulse responses are used in the comparative simulations of the adaptive algorithms. Chapter 3 also presents the simulation environment, methodology, and the simulation conditions.

Chapter 4 investigates selected state-of-the-art variable step-size adaptive algorithms that are proposed in the literature. The Chapter provides a description of the underlying concepts of the algorithms, and presents a comparative performance evaluation using the acquired impulse responses. Chapter 5 first proposes a new technique, the gradient-induced technique that is incorporated into the various variable individual step-size adaptive algorithms presented to improve the performance under highly non-stationary conditions with a modest increase in computational complexity, and then proposes an improved version of a combination of selected variable step-size algorithms that are studied in Chapter 4. Comparative simulations and discussion of results are also presented in Chapter 5.

In Chapter 6, a new technique, the thresholding technique, is proposed and incorporated into the Normalized Least-Mean-Squares (NLMS). The technique is extended to the Non-Parametric Variable Step-Size NLMS (NPVSS-NLMS) algorithm in Chapter 7. Comparative performance evaluations for both static and dynamic acoustic conditions are presented for the proposed algorithms in Chapter 6 and Chapter 7 using modified versions of the acquired impulse responses.

Chapter 8 provides a conclusion and summarizes the findings of this research. It also points out possible directions for future research.

Appendices included present the listing of the different algorithms presented in this thesis as well as a simple analysis of their computational complexities. The concept of the each algorithm is also summarized in a table.

CHAPTER 2 BACKGROUND REVIEW

2.1 Adaptive Filters for Acoustic Echo Cancellation

The adaptive filter is the core component of the echo cancellation system. It is assumed that the unknown acoustic echo path can be reasonably modeled as a linear system. The Finite Impulse Response (FIR) filter structure is the most popularly used adaptive filter structure because it guarantees stability and is easily implemented **[24]**. Figure 2.1 shows the FIR filter structure. The filter consists of a weighting sequence (the coefficients of the estimated impulse response: $\hat{c}_0(n)$, $\hat{c}_1(n)$, ..., $\hat{c}_{L-1}(n)$) of length L that is multiplied by the delayed samples of the input signal $x(n)$ and added to give the output of the filter $\hat{y}(n)$, where n is the discrete-time index.

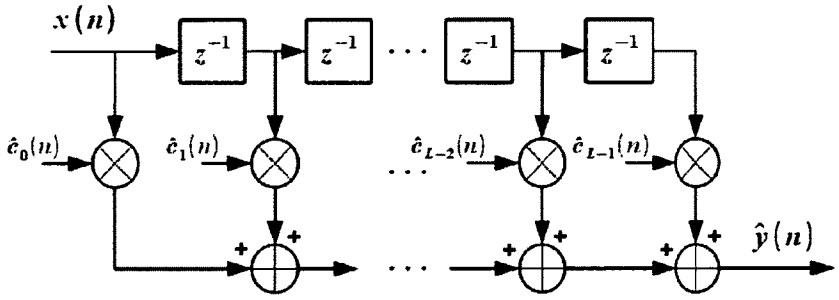


Figure 2.1: FIR filter structure.

Figure 2.2 and Table 2.1 present the echo cancellation system description and problem formulation respectively. The adaptive filter $\hat{c}(n)$ is adapting its coefficients to minimize the error $e(n)$ between its output $\hat{y}(n)$ and that of the unknown system $d(n)$, also called

desired signal. Adaptive algorithms are designed to perform this task as fast and as efficient as possible.

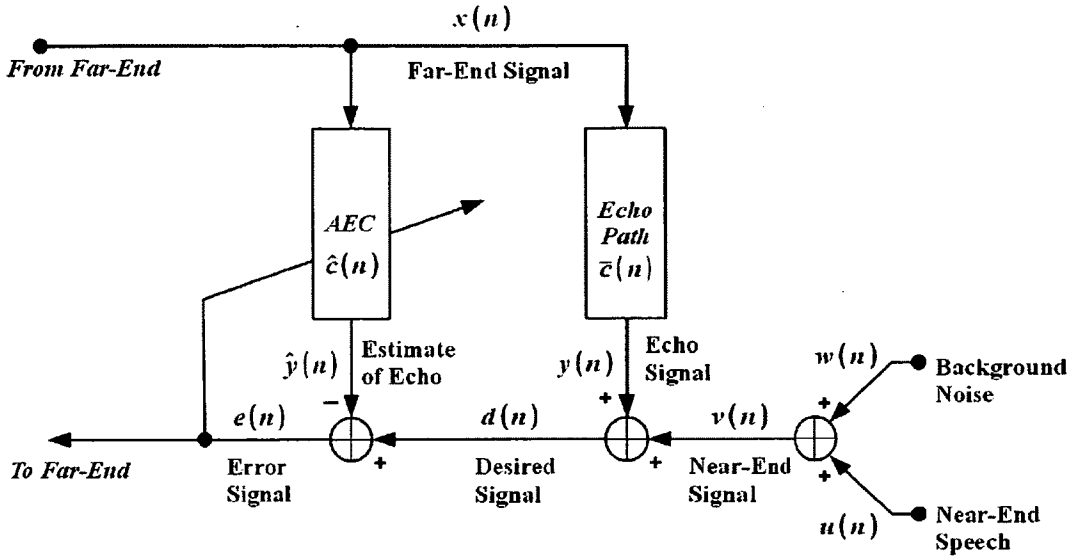


Figure 2.2: System description of acoustic echo cancellation.

Table 2.1: Problem formulation of acoustic echo cancellation.
$x(n)$ is the input signal. $\bar{x}(n)$ is the input vector of length L. $\bar{c}(n)$ is the echo path impulse response of the system (and is time-variant). $\hat{c}(n)$ is the estimated impulse response (i.e., the adaptive filter coefficients). $y(n) = \bar{c}^T(n)\bar{x}(n)$ is the echo signal. $\hat{y}(n) = \hat{c}^T(n)\bar{x}(n)$ is the estimated echo signal (i.e., the output of the adaptive filter). $v(n) = w(n) + u(n)$ is the near-end signal, where, $w(n)$ is the background noise, and $u(n)$ is the near-end speech signal. $d(n) = y(n) + v(n)$ is the reference signal which is the output of the unknown system (also known as the desired signal). The filter should adapt only when there is no near-end speech, i.e. $u(n) = 0$, therefore the desired signal considered in the design of adaptive algorithms is: $d(n) = y(n) + w(n)$. $e(n) = d(n) - \hat{y}(n) = d(n) - \hat{c}^T(n)\bar{x}(n)$ is the error signal (more formally known as the a priori error signal) n: the discrete-time index L: the length of the adaptive filter.

Although the acoustic echo cancellation problem may seem to be direct, in fact it is one of the most challenging applications of adaptive filtering due to several reasons. First, the echo path impulse response has the following characteristics:

- **Long length, *L***: a typical acoustic echo path has an impulse response of several hundred milliseconds, for a sampling rate of 8 kHz (or 16 kHz for high definition audio) the required adaptive filter length, L, is in the order of thousands of taps.

As the filter length increases, the maximum convergence rate of the adaptive algorithm decreases [**25**].

- **Fast time-varying, $\bar{c}(n)$**: the echo path impulse response $\bar{c}(n)$ is continuously and rapidly changing due to motion in the LRM environment. Variations in the temperature and pressure in the acoustic environment also affect the echo path impulse response. This fast time-variation of the impulse response requires fast tracking capabilities of the adaptive algorithm.

Second, the microphone picks up the echo signal $y(n)$ along with the near-end signal $v(n)$. The near-end signal can contain both the near-end speech $u(n)$ and background noise $w(n)$. The near-end speech acts as a disturbance signal while the background noise can be non-stationary. The adaptive filter is supposed to provide an estimate of the echo signal at its output and at the same time provide an estimate of the near-end speech signal to be transmitted to the far-end speaker. Last but not least, the input signal to the adaptive filter $x(n)$ is speech, which is a non-stationary and highly correlated signal. All these challenges impose special requirements for the adaptive algorithms that address the acoustic echo cancellation problem.

The adaptive algorithm of the AEC, therefore, should achieve the following performance characteristics:

- **Fast convergence rate and tracking capability**: to be able to dynamically model the long length and fast time-varying nature of the echo path impulse response.

- **Low misadjustment in steady-state**: to be able to attenuate the far-end speaker echoes by at least 25 dB (acoustic ITU recommendations).

- **Robustness against near-end signal variations:** to deal with background noise variations and double talk, a situation where speakers at both ends speak simultaneously.

- **Low computational complexity:** to provide efficient and low-cost real-time implementation, especially due to long filter lengths.

There is a trade-off between the performance requirements mentioned above; no algorithm in the literature satisfies all the previous requirements. Therefore the choice of an adaptive algorithm depends on the echo cancellation scenario subject to specific conditions and required performance measures.

2.2 Adaptive Filtering Algorithms

The Least-Mean-Squares (LMS) and Normalized Least-Mean-Squares (NLMS) algorithms [**26**,**27**] are the most widely used adaptive filters in practice due to their simplicity, low computational cost and robustness.

The NLMS algorithm has the advantage of being insensitive to the norm of the input vector since the coefficient-update is normalized by the power of the input vector. Since speech signals involve intervals of speech activity accompanied by intervals of pause, the NLMS algorithm exhibits faster convergence than the LMS algorithm for speech signals. However, the colored signal nature of speech signals limit the convergence speed of

NLMS algorithm; therefore more involved algorithms with better performance but at increased computational costs have been proposed.

Ozeki and Umeda [**28**] addressed this problem by developing the Affine Projection Algorithm (APA) which was derived as a generalization of the NLMS algorithm. It is a generalization in the sense that each coefficient-update vector in the NLMS algorithm is viewed as a one dimensional affine projection, while in the APA the projections are made in multiple dimensions using past data to form the solution space. As the projection dimension increases, the convergence rate increases but the computational complexity also increases.

In this thesis, the NLMS algorithm is used as a base algorithm for various approaches for performance improvements. The following section summarizes the LMS, and NLMS algorithms.

2.2.1 Standard Adaptive Filtering Algorithms

2.2.1.1 Least-Mean-Square (LMS) Algorithm

The LMS algorithm is a stochastic implementation of the steepest-descend algorithm [**26,27**], which searches for the optimal solution, $\bar{c}_{opt}$, by iteratively minimizing the mean-square error,

$$\min_{\bar{c}_{opt}} E\{\ |e(n)|^2\ \} \tag{2.1}$$

The $E\{.\}$ is the mathematical notation of the expected value. The LMS weight-update equation is

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{LMS}\,\bar{\boldsymbol{x}}(n)\,e(n) \tag{2.2}$$

where μ_{LMS} is the step-size parameter of the adaptive algorithm.

All signals are assumed to be real-valued for simplicity. A sufficient and necessary condition for the algorithm to converge is to have all the filter weights converge to the actual solution which leads to the following bounds on the step-size parameter:

$$0 < \mu_{LMS} < \frac{2}{\lambda_{max}} \tag{2.3}$$

where λ_{max} is the largest eigenvalue of the correlation matrix of the input vector, $\bar{\boldsymbol{x}}(n)$.

Algorithm Description

Table 2.2: The Least Mean-Square (LMS) algorithm.

Initialization:	$\hat{\boldsymbol{c}}(0) = \bar{\boldsymbol{0}}$
Parameters:	$0 < \mu_{LMS} < \frac{2}{\lambda_{max}}$
Error:	$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\bar{\boldsymbol{x}}(n)$
Filter Update:	$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{LMS}\,\bar{\boldsymbol{x}}(n)\,e(n)$

$n = 0, 1, \ldots, N-1$. where N is the number of iterations.

Computational complexity

For real-valued data, the LMS with a fixed step-size, μ_{LMS}, requires $2L + 1$ multiplications and $2L$ additions per iteration.

Advantages

The LMS is simple, therefore computationally cheap to implement and fast for real-time operation. It is also robust, and numerically stable.

Disadvantages

It suffers from slow convergence speed due to the dependency on the eigenvalue spread of the input signal's autocorrelation matrix. In other words the convergence is dependent on the variation in the input signal's power.

2.2.1.2 Normalized Least-Mean-Square (NLMS) Algorithm

The NLMS algorithm [**26,27**] can be derived from a constrained optimization problem, which states that, given the adaptive filter input vector $\bar{\boldsymbol{x}}(n)$ and the desired signal $\mathrm{d}(n)$, it is required to find the adaptive filter coefficients $\hat{\boldsymbol{c}}(n+1)$ that will minimize the squared Euclidean norm of the change in the filter coefficients, $\Delta\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n+1) - \hat{\boldsymbol{c}}(n)$, subject to the constraint $\bar{\boldsymbol{x}}^{\mathrm{T}}(n)\hat{\boldsymbol{c}}(n+1) = \mathrm{d}(n)$.

$$\min_{\hat{\boldsymbol{c}}(n+1)} \|\Delta\hat{\boldsymbol{c}}(n+1)\|^2 \tag{2.4}$$

$$subject\ to: \quad \bar{\boldsymbol{x}}^{T}(n)\hat{\boldsymbol{c}}(n+1) = d(n)$$

The NLMS weight-update equation is

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \frac{\mu_{NLMS}}{\bar{\boldsymbol{x}}^T(n)\bar{\boldsymbol{x}}(n) + \delta_{NLMS}}\bar{\boldsymbol{x}}(n)e(n) \tag{2.5}$$

where δ_{NLMS} is a regularization constant, which is a very small positive number with typical value equal to a constant multiplied by the variance of the input signal. It is used to prevent division by zero when the input is zero.

By examining a necessary but not sufficient condition for the stability of the LMS algorithm, we can derive a necessary condition for the stability of the NLMS algorithm. An intuitive condition for stability is to have

$$|\varepsilon(n)| < |e(n)| \tag{2.6}$$

where

$$\varepsilon(n) = d(n) - \bar{\boldsymbol{x}}^T(n)\hat{\boldsymbol{c}}(n+1) \tag{2.7}$$

is the a posteriori error signal, computed after the adaptation of the filter. By substituting equation (2.2) in equation (2.7), and using the condition (2.6) we find

$$0 < \mu_{LMS} < \frac{2}{\bar{\boldsymbol{x}}^T(n)\bar{\boldsymbol{x}}(n)} \tag{2.8}$$

By comparing the weight-update terms of the LMS and NLMS algorithms, we find that the step-size of the LMS, μ_{LMS}, is comparable to $\frac{\mu_{NLMS}}{\bar{\boldsymbol{x}}^T(n)\bar{\boldsymbol{x}}(n)+\delta_{NLMS}}$ in the NLMS update term. If we discard the regularization constant and take into account inequality (2.8) we find that a necessary but not sufficient condition for the stability of the NLMS algorithm is to have the normalized step-size parameter within the following bounds:

$$0 < \mu_{NLMS} < 2 \tag{2.9}$$

Algorithm Description

Table 2.3: The Normalized Least Mean-Square (NLMS) algorithm.	
Initialization:	$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$
Parameters:	$0 < \mu_{NLMS} < 2$
	$\delta_{NLMS} = cst.\sigma_x^2$
Error:	$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$
Filter Update:	$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \frac{\mu_{NLMS}}{\overline{\boldsymbol{x}}^T(n)\overline{\boldsymbol{x}}(n) + \delta_{NLMS}}\overline{\boldsymbol{x}}(n)e(n)$

Computational complexity

The NLMS algorithm requires $2L + 3$ multiplications, 2L + 3 additions, and 1 division per iteration.

Advantages

The NLMS has nearly the same convergence speed as the LMS but performs better when the input signal has a large dynamic range. This is because the NLMS is not sensitive to the variations in the norm of the input signal vector. Therefore the NLMS performs better with applications that involve speech signals such as in acoustic echo cancellation.

Disadvantages

The convergence speed and tracking ability of the NLMS deteriorates with long adaptive filter lengths. The NLMS performance also degrades with colored input signals, such as speech signals.

2.2.2 Literature Review

There are various areas of research for improving the performance of the LMS and NLMS algorithms. The approaches vary according to the area of application and the performance criteria under consideration. These include but not limited to using different filter structures, frequency-domain adaptive filtering, subband filter banks, and varying the step-size parameter. The adaptive Finite Impulse Response (FIR) filter is the most popular filter structure but Infinite Impulse Response (IIR) filters have also been investigated in the literature [**29**,**30**]. IIR filters have less computational cost; however, they require to be tested for stability. Frequency-domain implementation of FIR adaptive filters requires transferring the signals to the frequency domain using Discrete Fourier Transform (DFT) and the echoes are cancelled in the frequency domain [**31**]. There is a lot of literature on subband filter banks [**32**,**33**] where the input and desiredsignals are decomposed into subbands, decimated, and the adaptation is performed in each subband at a lower rate using shorter filters as opposed to one long FIR filter as in conventional adaptive filtering. Since the convergence rate of FIR adaptive filters is faster for white (uncorrelated) input signals and decreases for colored (correlated) signals, such as speech

inputs, lattice-type pre-whitening LMS/NLMS filters [**34**] are used to whiten the input signals to improve convergence speed.

The complexity of the NLMS increases as the length of the filter increases. Various algorithms that select only a subset of coefficients to be adapted have been proposed to provide good real-time performance for long impulse responses. These algorithms are referred to as the selective partial update algorithms. A tap selection scheme was proposed by Kawamura and Hatori in [**35**] in which at any given instant, only a subset of the taps are adapted and used to compute the filter output. The selection is based on the magnitudes of the taps, where taps with small magnitudes contribute the least to the output of the filter. The subset is updated on regular intervals of time. This technique allows covering long impulse responses efficiently. The update of the subset of taps is carried out using a first-in first-out (FIFO) queue of discarded tap positions such that the taps in the subset that have the smallest magnitudes are discarded and queued into the tail of the FIFO and replaced by other tap locations that are at the head of the FIFO. This technique is sensitive to the changes in the impulse response, but imposes the computational overhead of managing the FIFO.

A partial update NLMS algorithm was proposed by Dogancay et al. in [**36**] where the coefficients that are updated are selected based on the magnitudes of the corresponding inputs. The performance is as good as the full update NLMS algorithm. Aboulnasr et al. [**39**] proposed a selective partial update scheme where the coefficients are selected such that they minimize the mean-square error. The performance is close to that of the NLMS. Gordy et al. [**40**] proposed another partial selection update to reduce the complexity of the proportionate-NLMS algorithms where the coefficients are selected using a block-

based method. The algorithm combines proportional weighting of the input signal vector with fast ranking methods. The coefficients are divided into equal-sized blocks, and inside each block the coefficients are ranked according to the descending magnitudes of the corresponding inputs. A fixed number of coefficients are selected from all the blocks to be updated every iteration. More selective partial update schemes can be found in the literature [37,38].

In this work the focus will be on improvements of the LMS/NLMS algorithms, conceptually, in terms of varying the step-size parameter. This is an area that has been rigorously researched for the past couple decades. This section reviews the literature on the variable step-size adaptive algorithms. Other approaches in adaptive filtering are beyond the scope of this thesis.

There are several ways to vary the adaptation step-size parameter. The first approach is to vary the scalar adaptation step-size over time, $\mu(n)$. These algorithms are referred to as the **Variable Step-Size (VSS)** algorithms. There are many VSS algorithms proposed in the literature in which the scalar step-size parameter varies over time according to a specific criterion as will be covered in section 2.2.2.1. In the case of a scalar step-size parameter, the adaptation gain is distributed evenly among all the coefficients of the adaptive filter since the step-size parameter is the same for all the coefficients of the adaptive filter.

Another approach for varying the step-size is to vary a vector of adaptation step-sizes, $\bar{\boldsymbol{\mu}}(n)$, referred to as the individual step-sizes vector or μ-vector, such that there is an independent adaptation step-size for each coefficient of the adaptive filter. These

algorithms are, in general, referred to as the **Variable Individual Step-Size (VISS)** algorithms, and covered in section 2.2.2.3. In this case the adaptation gain is redistributed among the coefficients according to a specific profile, rule or criterion such that the individual step-size parameter is different for each coefficient of the adaptive filter. Table 2.4 summarizes the different approaches of varying the step-size parameter and lists the corresponding categories of various types of adaptive filter algorithms.

Table 2.4: Different approaches to varying the step-size parameter.

	Time-Invariant	**Time-Variant**
Scalar (Universal Step-Size)	μ Classical Algorithms	$\mu(n)$ Variable Step-Size (VSS) Algorithms
Vector (Individual Step-Size)	$\bar{\mu}$ Individual Step-Size (ISS) or μ-Vector Algorithms	$\bar{\mu}(n)$ Variable Individual Step-Size (VISS) or Variable μ-Vector Algorithms

Figure 2.3 shows a general block diagram of adaptive algorithms that use variable step-size, which is also referred to as gain control. Figure 2.4 shows a schematic drawing for the variation of the step-size for each category in Table 2.4.

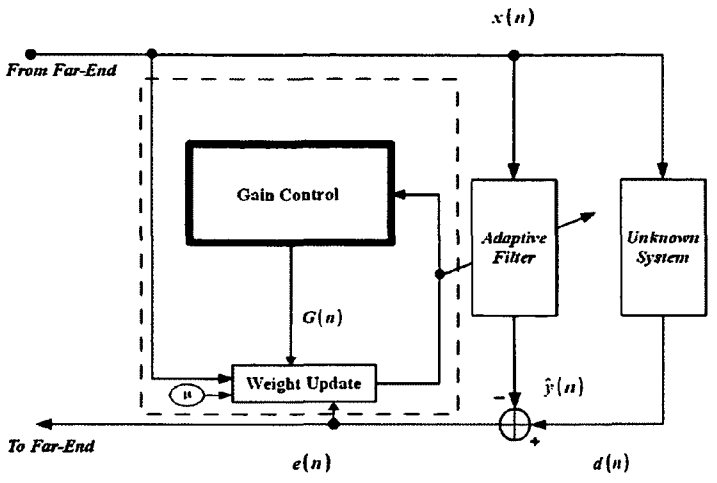


Figure 2.3: AEC block diagram showing the gain control module in variable step-size adaptive algorithms.

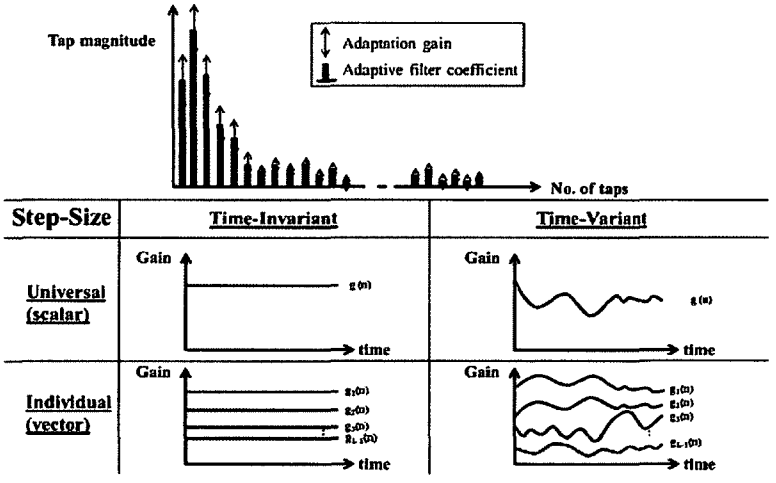


Figure 2.4: Schematic showing how the step-size varies for each coefficient in the different Variable Step-Size algorithm categories.

2.2.2.1 Variable Step-Size (VSS) Algorithms

The stability of the NLMS algorithm is governed by the step-size parameter. It is also well known that the step-size parameter, chosen within stability limits, causes a trade-off between high convergence speed and tracking ability on one hand when its value is large

and low misadjustment on the other hand when its value is small [**26**]. The steady-state performance is improved by reducing the step-size since it is equivalent to improving noise immunity by using a longer data averaging window from one aspect. In order to solve this performance conflict, and achieve good convergence and low misadjustment, many schemes have been proposed to control the step-size in the last couple decades as discussed in what follows. Figure 2.5 shows a block diagram for VSS algorithms.

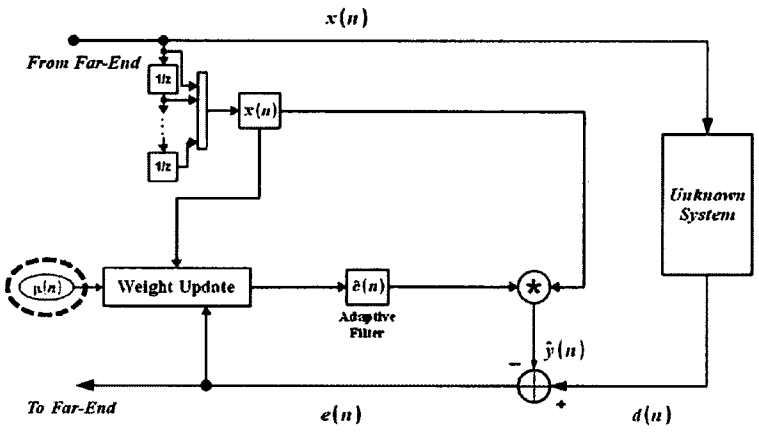


Figure 2.5: Block diagram for Variable Step-Size algorithms.

Harris et al. [**3**] proposed a VSS-LMS algorithm where the step-size is increased or decreased depending on the polarity of successive samples of the estimation error signal. A disadvantage pointed out by [**4**] is that this algorithm has instability since it uses the sign of the gradient component of the squared error. Although the error signal ideally should contain only the squared error, this signal is usually contaminated by the near-end signal. Therefore an incorrect sign is often detected causing undesirable change in the step-size. In the algorithm proposed by Kwong and Johnston [**5**], the step-size update

depends on the magnitude of the square of the instantaneous estimation error: $\mu(n+1) = \alpha\mu(n) + \gamma e^2(n)$ where $0 < \alpha < 1$ and $\gamma > 0$. The step-size increased or decreased as the mean-squared error increased or decreased. The algorithm thus tracks the changes in the system and provides small steady-state error. When the prediction error is large the tracking is fast, when the error is small the misadjustment is small, thus overcoming the performance conflict mentioned above. This idea could be easily extended to the NLMS algorithm and its variants. In all VSS algorithms, upper and lower boundaries for the step-size values are maintained to ensure stability and reasonable tracking performance.

Mathews and Xie [**6**] updated the step-size using a gradient descent algorithm. The update is proportional to the negative gradient of the squared estimation error associated with the previous step-size value. The algorithm is known as the Stochastic Gradient Algorithm with Gradient Adaptive Step-Size (SG-GAS). The algorithm had almost same convergence speed as the NLMS but better steady-state performance.

A disadvantage of these algorithms is that they are sensitive to noise. Sugiyama [**7**] showed that the performance of the SG-GAS [**6**] is affected by the uncorrelated noise in the environment and proposed using soft-clipping thresholds for the step-size update term such that the change in the step-size is limited to a fraction of the square of the previous step-size.

Aboulnasr and Mayyas [**8**] presented a robust variable step-size algorithm to overcome this problem. The derivation is based on the fact that the correlation between successive error samples is small when the error energy is small, and this correlation function is less sensitive to the noise than it is to the energy in the error. A time-averaged estimate of the

autocorrelation between successive error samples is thus used to control the step-size, $p(n) = \beta\, p(n-1) + (1-\beta)e(n)e(n-1)$. And the step-size update becomes $\mu(n+1) = \alpha\, \mu(n) + \gamma p^2(n)$. Simulations demonstrate significant convergence rate improvement as compared to [**5**] in high and low SNR conditions in stationary environments, and comparable performance with the standard LMS in non-stationary environments.

Mader et al. [**9**] proposed an optimal step-size per iteration for the NLMS algorithm that is based on minimizing the mean-squared deviation. Shin et al. [**10**] compares the various VSS-LMS algorithms proposed in [**5**], [**8**], [**41**], and [**9**], and also proposed a new VSS algorithm for NLMS that depends on finding the optimal step-size per iteration based on finding the largest decrease in the mean-squared deviation of the filter vector.

A common problem for all the previously mentioned VSS algorithms is that they depend on one or more parameter that is not easy to tune in practice. A Non-Parametric VSS-NLMS (NPVSS-NLMS) algorithm was proposed by Benesty et al. [**11**] whose performance does not depend on tuning of the parameters. The variable step-size is derived such that the a posterior error, $\varepsilon(n) = d(n) - \hat{\boldsymbol{c}}^T(n+1)\bar{\boldsymbol{x}}(n)$ is cancelled. Due to the presence of system noise, the practical approach was to have the power of the a posterior error approach the power of the system noise. The NPVSS-NLMS is extended in this thesis to achieve further performance improvements as will be investigated in Chapter 7.

2.2.2.2 Individual Step-size (ISS) Algorithms

The NLMS does not take into consideration the nature of the system being modeled, therefore algorithms have been developed that tend to exploit the static and dynamic characteristics of acoustic echo path impulse responses in order to be able to deal with the long lengths and rapidly time-varying nature of the acoustic echo path impulse responses and therefore improve both convergence and tracking performances.

One of these algorithms is the Exponential Step-Size NLMS (ESNLMS) algorithm proposed by Makino et.al. [**12**] and further examined by Yuan [**13**]. Makino et.al. [**12**] carried out extensive measurements of typical room impulse responses and found that acoustic echo path impulse responses have a common exponentially decaying characteristic. The ESNLMS makes use of this observation by using individual step-sizes for each coefficient, referred to as a μ-vector, rather than the scalar step-size μ for the entire adaptive filter. The μ-vector takes on an exponentially decaying profile. The decay factor depends on the reverberation time of the room. The authors proposed that the elements of the μ-vector should be varying and proportional to the mean-square error of the corresponding coefficients such that coefficients with large errors are updated with larger step-sizes. However, for practical reasons, a time-invariant μ-vector was used resulting in tripling of the convergence and tracking performances for white noise input and doubling for speech input when compared to NLMS performance. A problem with the ESNLMS algorithm (using a time- invariant μ-vector) is that a priori information about the room reverberation characteristics is required since the step-size profile is fixed.

Yuan [13] proposed a time-varying step-size profile that demonstrated only a small improvement in performance. This time-varying step-size profile makes the algorithm a Variable Individual Step-Size (VISS) algorithm. The VISS algorithms are reviewed in the next section.

2.2.2.3 Variable Individual Step-Size (VISS) Algorithms

Variable Individual Step-Size algorithms assign a different variable step-size for each filter coefficient. These algorithms make use of specific properties of the EPIR to design the update rule. The sparseness of the network EPIRs has inspired the idea of proportionate adaptation, since only a small number of coefficients have significant magnitudes while the rest have small magnitudes or are almost zero. Proportionate adaptation means that each coefficient is independently adapted in proportion to its estimated magnitude. The adaptation gain is redistributed such that larger coefficients receive more gain and adapt faster than smaller ones leading to an overall increase in the convergence rate. Acoustic EPIRs are generally less sparse than network EPIRs, and their sparseness depends on the amount of reflections and the intensity of reflections. The reflections, in general, depend on the reverberation time, the distance between the loudspeaker and microphone, and the changes in the temperature and pressure of the environment.

More than a decade ago, Duttweiler proposed the proportionate NLMS (PNLMS) algorithm [14] which exploits the sparseness nature of the EPIR. It relies only on the current estimate of the filter coefficients, without requiring any prior information about

the echo path. However, the calculation of the individual step-sizes was designed based on an intuitive way rather than using any optimization criteria and therefore the convergence of the PNLMS slows down after a fast initial phase. The PNLMS is also very sensitive to the sparseness degree of the echo path and the performance deteriorates and becomes even worse than that of the NLMS algorithm when the impulse response is not sparse, i.e. with acoustic impulse responses.

Many PNLMS-based algorithms have been proposed in the last decade to overcome these problems and improve the performance of the PNLMS for both sparse and non-sparse impulse responses. The PNLMS++ algorithm proposed by S. Gay in [**15**] alternates between NLMS and PNLMS coefficient updates and therefore it is less sensitive to the assumption of a sparse impulse response. Benesty et al. proposed the improved PNLMS (IPNLMS) algorithm [**16**] which uses an optimal rule for updating the coefficients by combining the proportionality rule of the PNLMS with the even gain distribution of the NLMS algorithm per iteration. Deng et al. proposed the mu-law PNLMS (MPNLMS) algorithm [**17,18**] using an optimal rule that is based on the steepest-descent method. The update rule is similar to that of the PNLMS except that the magnitudes of the coefficients are first compressed using a logarithmic function. Since the logarithm function is computationally expensive, the segment PNLMS (SPNLMS) algorithm [**17**] was also proposed as a cheaper implementation for the MPNLMS. However, the logarithmic function can be approximated and a Look Up Table can be used to reduce computation requirements. Moreover, logarithmic functions are not considered expensive in practice nowadays. The variation of the mu-law parameter of the MPNLMS was proposed by Wagner et al. in the adaptive MPNLMS (AMPNLMS) algorithm [**19**]. Loganathan et al.

proposed a set of sparseness controlled PNLMS algorithms in [**20**] where a measure of the sparseness of the current estimate of the impulse response was incorporated into the PNLMS, MPNLMS, and IPNLMS algorithms to give the sparseness controlled PNLMS (SCPNLMS), sparseness controlled MPNLMS (SCMPNLMS), and sparseness controlled IPNLMS (SCIPNLMS) algorithms respectively. Hoshuyama et al. proposed the generalized gradient PNLMS (GGPNLMS) algorithm [**21**] where the coefficient update rule depends on an estimate of the gradient of the coefficients with the goal to have higher tracking capability in non-stationary environments.

All the PNLMS-based algorithms mentioned above have good performances with acoustic EPIRs and are therefore considered a start point of part of the research in this thesis. In Chapter 4, these PNLMS-based algorithms are outlined and have their performances and capabilities investigated for acoustic echo cancellation of hands-free portable devices using the acquired impulse responses presented in Chapter 3. These algorithms form a basis for developing new algorithms to improve the convergence and tracking performances as presented in Chapter 5.

Other PNLMS-based algorithms that are developed to work best with impulse responses exhibiting high sparseness are like the PNLMS with individual activation factors (IAF-PNLMS) proposed by das Chagas de Souza in [**22**] and the composite PNLMS (CPNLMS) proposed by Nekuii in [**23**]. These algorithms, however, are out of the scope of this thesis.

2.3 Performance Measures

There are different ways to measure an estimate of the amount of undesired echo signal attenuation. The most commonly used measures in the literature are explained in this section and employed in the Matlab GUI. However, only the Echo-Return Loss Enhancement (ERLE) is used in the comparative simulations in this thesis because it gives the best indication of echo attenuation, which has a direct relation to the quality of speech signals.

2.3.1 Mean-Square Error (MSE)

The error signal is the difference between the desired signal and the output of the filter, and the mean-square error is defined as:

$$MSE(n) = E\{[d(n) - \hat{y}(n)]^2\} \tag{2.10}$$

where $E\{.\}$ denotes mathematical expectation,

$$d(n) = y(n) + w(n) \tag{2.11}$$

$$d(n) = \bar{c}^T(n)\bar{x}(n) + w(n)$$

is the desired signal, and

$$\hat{y}(n) = \hat{c}^T(n)\bar{x}(n) \tag{2.12}$$

is the output of the adaptive filter. By substituting equations (2.11) and (2.12) in equation (2.10) we get:

$$MSE(n) = [\bar{c}^T(n) - \hat{c}^T(n)]^T R_x [\bar{c}^T(n) - \hat{c}^T(n)] + \sigma_w^2 \tag{2.13}$$

where

$$R_x = E\{\bar{x}(n)\bar{x}(n)^T\} \tag{2.14}$$

is the correlation matrix of the input signal $x(n)$ and

$$\sigma_w^2 = E\{w^2(n)\} \tag{2.15}$$

is the variance of the background noise.

As the adaptive filter, $\hat{c}^T(n)$, converges to the echo path impulse response, $\bar{c}^T(n)$, the $MSE(n)$ converges to the variance of the noise, σ_w^2,

$$\lim_{n\to\infty} MSE(n) = \sigma_w^2 \tag{2.16}$$

The MSE depends on the variance of the background noise, therefore it does not give a precise measure of the amount of echo attenuated.

2.3.2 Misalignment (Misalign.)

The misalignment specifies how well an adaptive filter converges to the impulse response of the unknown system. It is the one of the most widely used performance measure in the design of new adaptive algorithms in general. The misalignment is defined as:

$$Misalign(n) = \frac{\|\bar{c}^T(n) - \hat{c}^T(n)\|_2^2}{\|\bar{c}^T(n)\|_2^2} \tag{2.17}$$

or in dB,

$$Misalign(n) = 20\ log_{10} \frac{\|\bar{c}^T(n) - \hat{c}^T(n)\|_2}{\|\bar{c}^T(n)\|_2}\ (dB) \tag{2.18}$$

2.3.3 Echo-Return Loss Enhancement (ERLE)

The Echo-Return Loss Enhancement is the best performance measure used to give a precise estimate of the amount of echo attenuated, it is defined as:

$$ERLE(n) = \frac{E\{y^2(n)\}}{E\{[y(n) - \hat{y}(n)]^2\}} \tag{2.19}$$

or in dB,

$$ERLE(n) = 10\ log_{10} \frac{E\{y^2(n)\}}{E\{[y(n) - \hat{y}(n)]^2\}}\ (dB) \tag{2.20}$$

The ELRE does not depend on the background additive noise. The larger the ERLE, the more echo is attenuated, when $\hat{c}(n)$ converges to $\bar{c}(n)$, ERLE goes to infinity.

2.4 Double-Talk

Double-talk occurs when speakers on both ends speak at the same time, i.e. presence of near-end speech signal, $u(n)$, (see Figure 2.2). This situation affects the behavior of the adaptive filter to the extent that it may cause it to diverge from its impulse response estimation. Therefore, a double-talk detector (DTD) is usually employed with the echo canceller to stop the adaptation process during double-talk periods. Double-talk detectors are used to identify the presence of the near-end speech. There is a delay for any DTD to decide the presence of near-end speech and therefore the adaptive filter needs to be able to handle some double-talk without diverging away from its accurate estimation of the room impulse response, i.e. have robustness features. This imposes another challenge in the design of adaptive algorithms.

However, in this research it is assumed there is no double talk (i.e. $u(n) = 0$).

CHAPTER 3 DATA ACQUISITION AND SIMULATION ENVIRONMENT

This chapter first describes the experimental setup and measurement methodology for estimating the linear acoustic EPIRs. These EPIRs are used in all the comparative simulations of the adaptive algorithms presented in this thesis. The chapter then describes the simulation environment and methodology in terms of a Matlab GUI application. Finally, the simulation conditions for the various adaptive algorithms are summarized.

3.1 Echo Path Impulse Response (EPIR) Measurements

In order to be able to simulate the adaptive algorithms and compare their performances for acoustic echo cancellation in practical hands-free portable devices, actual acoustic EPIRs are required to be experimentally measured using commercial portable devices operating in hands-free mode.

3.2 Experimental Setup

The experimental setup is shown in Figure 3.1. Measurements were carried out using a commercial smartphone operating in hands-free mode. A single microphone and a single miniaturized loudspeaker are used in the hands-free mode. Each impulse response measurement involves playing a White Gaussian Noise (WGN) reference signal through

the loudspeaker of the smartphone into the LRM environment and measuring the reproduced echo signal using the microphone.

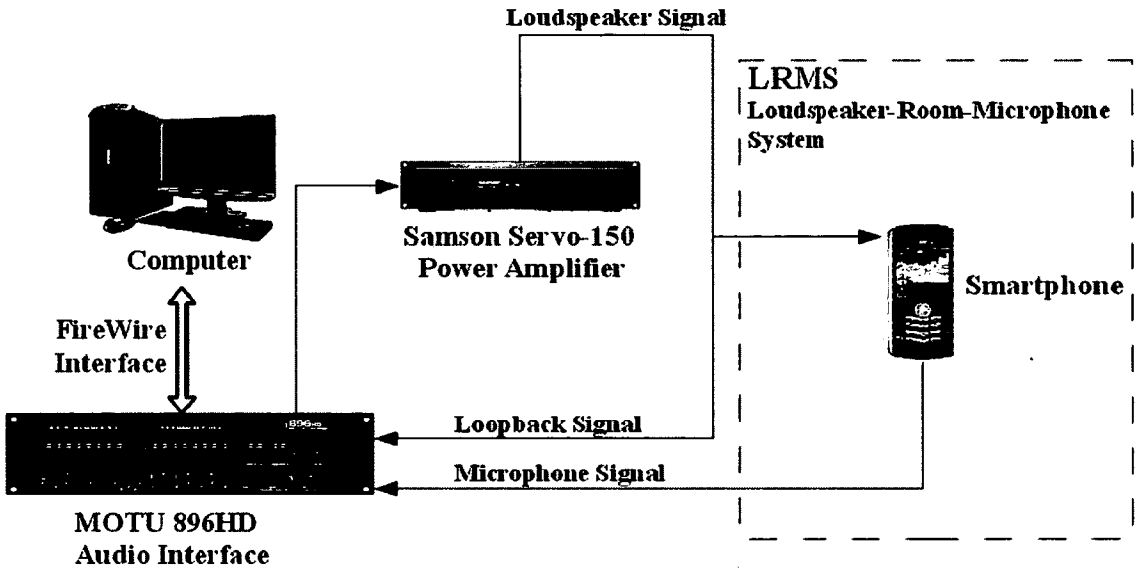


Figure 3.1: Experimental setup for echo path impulse response measurement.

The WGN reference signal is generated using Adobe Audition 3.0 software running on the computer and is then driven by a Samson Servo-150 power amplifier to the loudspeaker. The Samson amplifier is connected to the computer through a MOTU-896HD audio interface that is connected to the computer using FireWire cables. The microphone signal is digitized at a sampling rate of 48 kHz using the MOTU-896HD recording system with 24-Bit ADCs and the digitized signal is then captured by the Adobe Audition 3.0 software. The platform used for signal generation, acquisition and processing is a Dell Vostro 430, Intel® Core i7 desktop computer. The Adobe Audition 3.0 software was also used for control of playback and recording.

The MOTU-896HD audio interface is shown in Figure 3.2. This device consisted of 8 24-bit 192 kHz analog inputs and 2 analog outputs. All input and output ports have power control knobs to prevent output saturation and input clipping.

Figure 3.2: MOTU 896HD audio interface.

The Samson Servo-150 stereo power amplifier is shown in Figure 3.3. The amplifier has two channels; however, only one channel was used in the experiments. The amplifier was required because the internal amplifiers of the smartphone were not used since the connections were made directly to the miniaturized loudspeaker, and current is required to drive the signal.

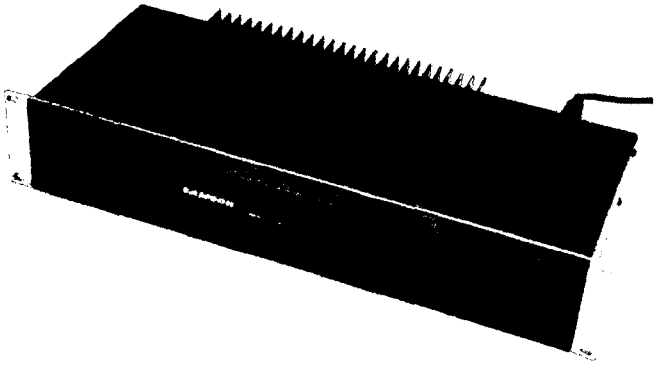


Figure 3.3: Samson Servo-150 power amplifier.

A loopback measurement of the loudspeaker signal was carried out, see Figure 3.1, to ensure that the loudspeaker signal played through the loudspeaker is not a distorted version of the one generated in the computer. This validates the experimental measurement by making sure that amplifications were in the linear range, no saturation or clipping has taken place at any of the devices' ports. Linearity of the loudspeaker was validated by listening to the signal played out and adjusting the volume at a moderate level. Visual inspection of the measured microphone signal showed no clipping as well.

Each pair of loudspeaker and microphone signals recorded is used to estimate an individual EPIR as described in the next section.

3.3 Linear Echo Path Impulse Response (EPIR) Estimation from Measured Signals

The estimation of the impulse response is a system identification problem that could be solved using adaptive filters. The linear acoustic EPIR between the loudspeaker and the microphone is estimated by slowly adapting a NLMS filter. The loudspeaker signal is used as the input to the adaptive filter and the microphone signal is used as the desired signal. This configuration is shown in Figure 3.4. To obtain an accurate estimation of the actual impulse response, first, a small step-size was used to guarantee a small final misalignment, and second, the NLMS algorithm ran for a large number of iterations. The effect of system non-linearities was assumed to be negligible and therefore all the linear impulse responses obtained are assumed to be an accurate estimation of the EPIRs.

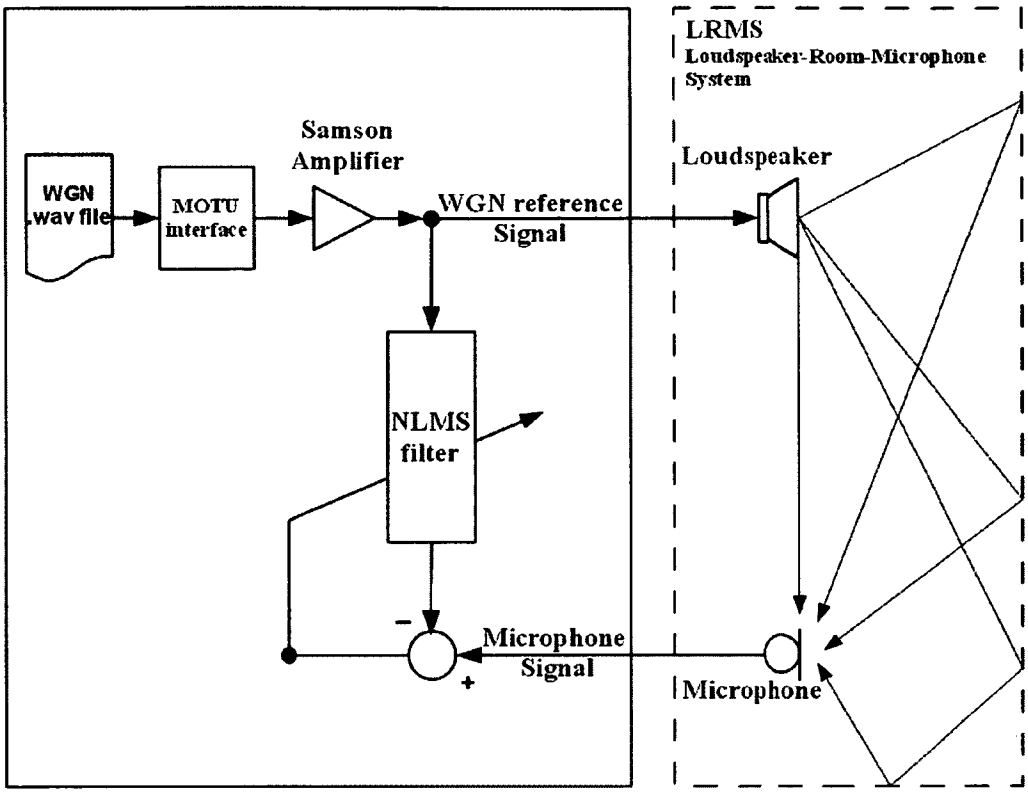


Figure 3.4: Identifying the linear echo path impulse response using a NLMS adaptive filter.

There are other methods for measuring the impulse response such as impulsive excitation, and maximum length sequence [**13**]. However, the adaptive filtering approach is used because it is simple, easy to implement, and provides an accurate linear model.

A thirty second WGN reference signal was played through the loudspeaker and picked up by the microphone. Both signals were sampled at 48 kHz and then decimated by a factor of three to 16 kHz sampling frequency. The signals were used to adapt an NLMS filter over 480000 iterations using a step-size, $\mu = 0.1$, the echo path length measured was 250 milliseconds resulting in a filter length of $L = 4000$ taps. The large number of

iterations resembles a very long averaging window to obtain a more accurate measurement of the impulse response (30 seconds long).

The loudspeaker signal level (controlled by the Adobe Audition 3.0 software and the Samson power amplifier) was set such that the loudspeaker performs in the linear range. The Samson amplifier was also operating in the linear range to avoid any non-linear amplifier saturation.

A MATLAB graphical user interface (GUI) was created to facilitate the estimation of different impulse responses using the recorded signals. A snapshot of the GUI is shown in Figure 3.5 below.

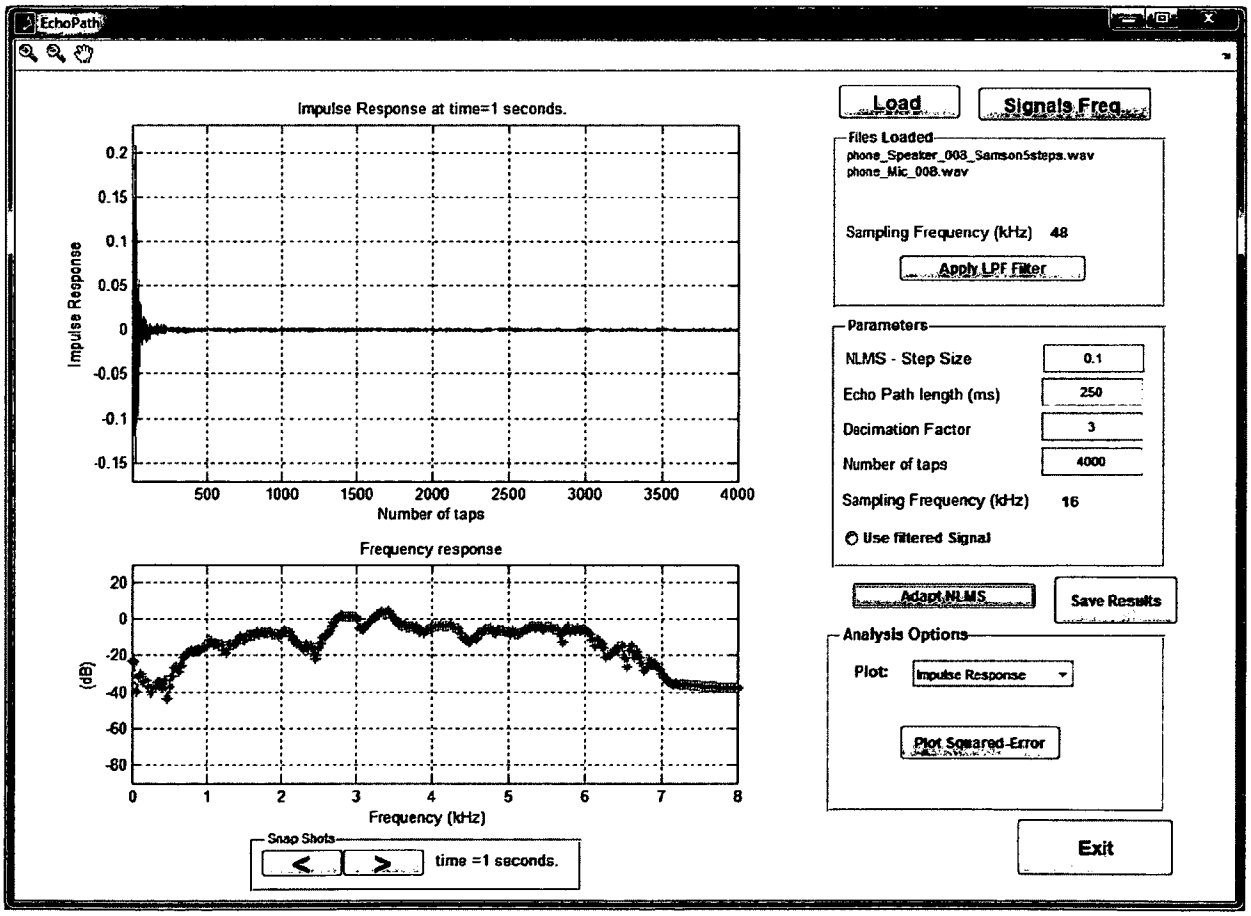


Figure 3.5: MATLAB GUI screen shot showing impulse response and frequency response of an acoustic EPIR.

The GUI has two plot areas on the left side: the top one is for the impulse response of the echo path and bottom one is for the frequency response. The parameters and other controls are on the right side. The following describes the steps for obtaining the impulse response:

- Loading two audio files, one pair of loudspeaker and microphone signals.
- Setting the NLMS adaptive filter step-size parameter and the echo path length. The decimation factor, echo path length, and number of taps (or filter length) are related and are automatically modified when one parameter is changed.
- Selection of whether to use the filtered signals or the original signals in the adaptation process. The filter used is a ChebyChev Type II low pass filter (LPF) with corner frequency 16 kHz, and order 50. Figure 3.6 shows a plot of the frequency spectrum for a sample of the original and filtered signals.
- Adaptation of the NLMS filter for the length of the audio signal, and saving the results to a .mat file.
- Setting the plotting analysis options to one of the following options:
    a) Impulse response at 1 second intervals
    b) Impulse response difference between 1 second intervals
    c) Minimum, maximum and average results (shown in Figure 3.7)
    d) Minimum, maximum and final results

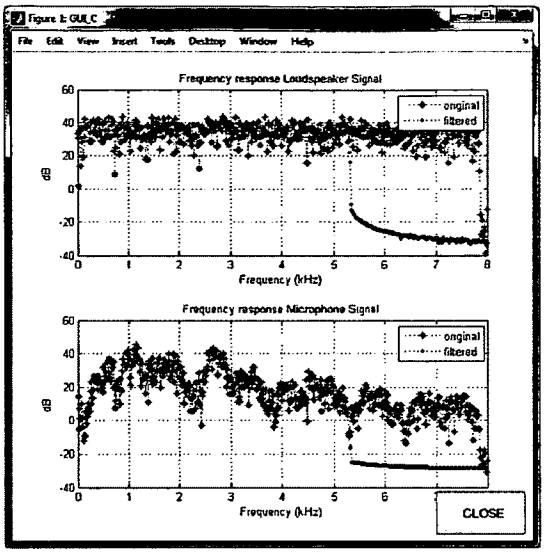


Figure 3.6: Frequency responses of original and filtered signals.

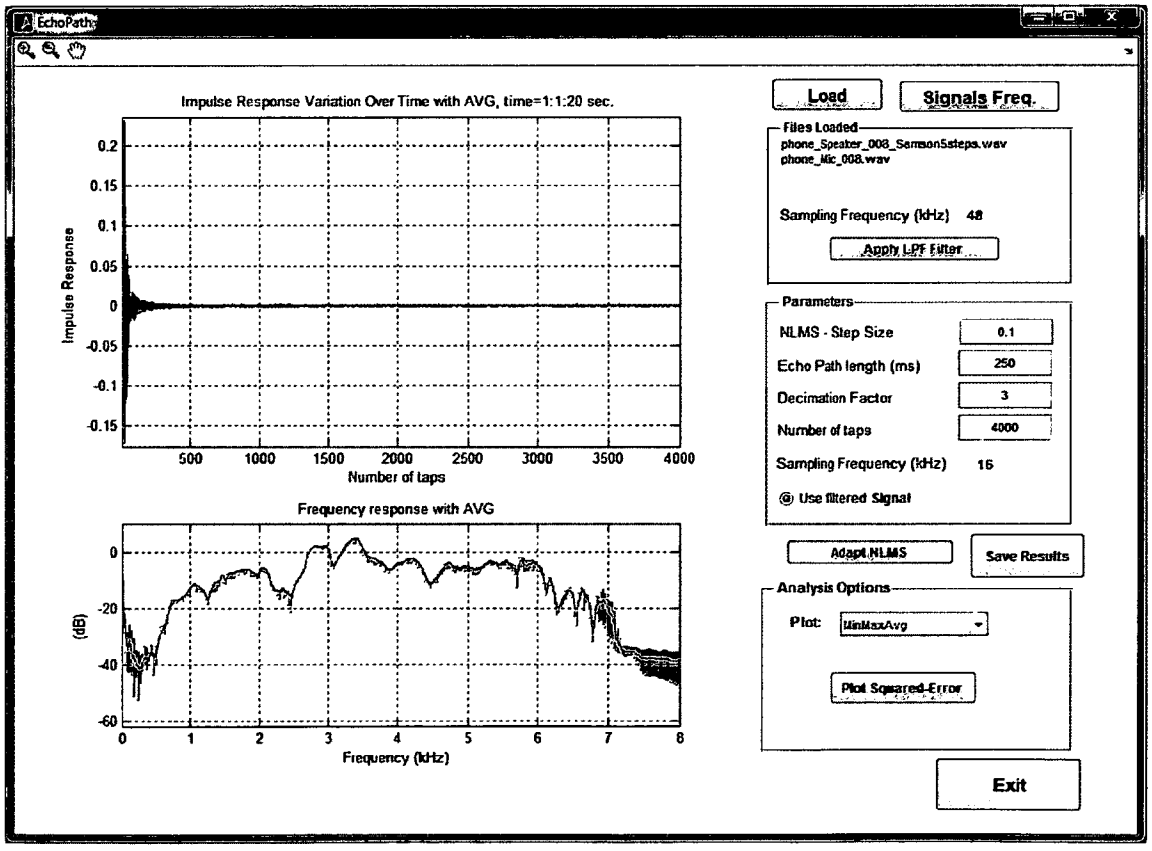


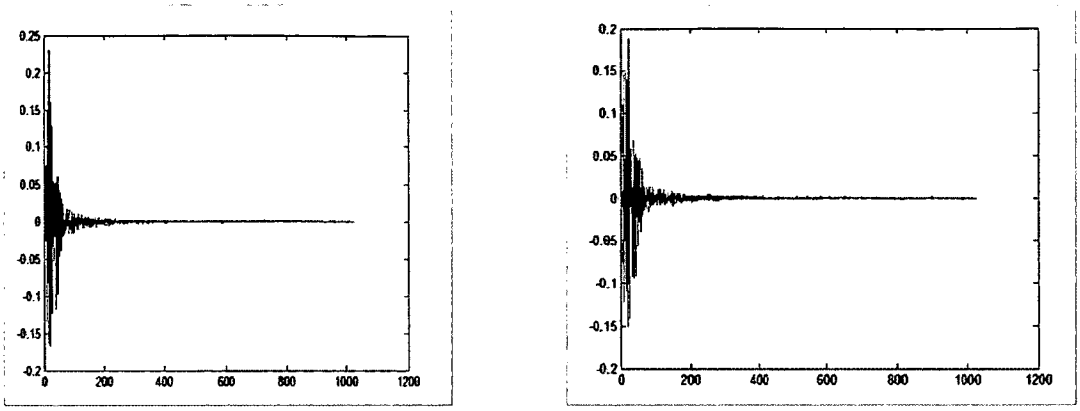
Figure 3.7: MATLAB GUI screen shot showing minimum, maximum and average of impulse response and frequency response estimations.

3.4 Acquired Echo Path Impulse Responses (EPIRs)

The EPIRs acquired and used to generate the microphone signals for the purpose of simulations are presented in this section. All the EPIRs obtained share a common general exponentially decaying profile with most of the differences occurring at the first part of the impulse response and with different intensities of the reflections along the echo path length. The obtained EPIRs are similar to typical acoustic EPIRs that are found in the literature [**42**,**43**].

The initial part of the impulse response is the result of the direct path between the loudspeaker and the microphone and the next segment is due to the early reflections from the smartphone's body and nearby objects and surfaces. Finally, the exponentially decaying tail is due to reflections from distant objects and the walls. Determination of the exact surface causing each reflection in the impulse response can be easily obtained by calculating the distance of each spike in the impulse response knowing that the speed of sound in air is 340 m/s.

The EPIRs in Figure 3.8 are measured with the smartphone placed inside an anechoic box. The anechoic box, shown in Figure 3.9, stops reflections and insulates external noise from being picked up by the microphone. This minimizes the effect of background noise on the accuracy of estimating the impulse response. The ERLE at which the impulse responses were acquired is approximately 40 dB. Only two impulse responses were acquired because the focus is on comparing a large number of adaptive algorithms, therefore we fix the impulse responses used in the experiments. Two impulse responses are required to be able to simulate a highly non-stationary by switching between them causing highly variant acoustic conditions.



a) Measured impulse response with phone lying on its back.

b) Measured impulse response with phone lying on its face.

Figure 3.8: Typical hands-free acoustic echo impulse responses of smartphone measured inside anechoic box.

These EPIRs depict the coupling between the loudspeaker and microphone of the smartphone with minimal echoes from the surroundings. As shown in Figure 3.8, only 1024 taps of the 4000 taps are used in the simulations, which represent 64 ms of the echo path.

Figure 3.9: Anechoic box.

3.5 Simulation Environment and Methodology

Another GUI was created to carry out the simulations and compare the various performances of the adaptive algorithms. A screen shot of the GUI is shown in Figure 3.10 below.

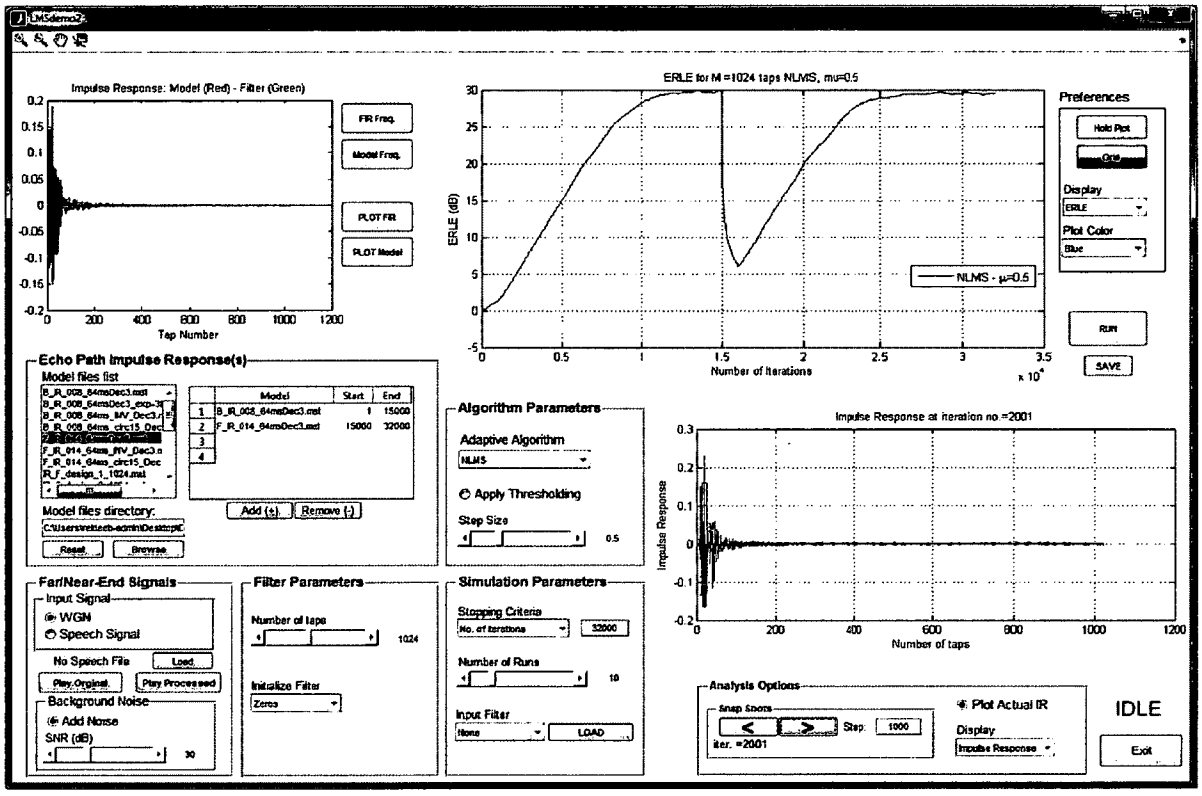


Figure 3.10: MATLAB GUI screen shot.

The following steps show how to use the GUI to run the simulations:

1) Load one or more **Echo Path Impulse Response** models:

The top left portion of the GUI is designed for the selection of either a single impulse response to simulate a static time-invariant echo path or a set of sequential impulse responses to simulate a dynamic time-variant echo path scenario. The impulse

response(s) selected are used to create artificial microphone signals by convolving the far-end loudspeaker signal with the selected impulse response(s).

- The top left plot, the *Impulse Response Model* plot, displays the model selected in the *Model Files List* just below. The buttons to the right of the plot are used to plot the impulse response of the selected model along with that of the adapted FIR filter (which is developed during the simulation) and the frequency response of the selected model along with that of the adapted FIR filter. Use the *Add(+)* button to add the impulse response model that is selected in the *Model Files List.*

- The *Models* table next to the *Model Files List* is used to display the name of the selected impulse response model(s) that are used in the simulation. The *Remove(-)* button is used to remove a model when it is selected in the *Models* table. When more than one model is loaded for simulation, the start and end iteration values associated with each impulse response are set in the *Models* table.

2) Set the **Far-End and Near-End Signals**:

- The far-end loudspeaker signal, which is also the input signal to the adaptive filter, is selected as a *WGN* signal or a *speech signal.* If a speech signal is selected, a speech file must be first loaded. The speech file can be played back before and after it is filtered.

- Since we assume there is no near-end speech (no double-talk), the output of the echo path is only corrupted by an independent WGN acting as

background noise (the near-end signal) by setting the *Add Noise* radio button and adjusting the power of the additive noise by setting the *Signal to Noise Ratio (SNR)*. The SNR is the power of the input signal relative to that of the additive background noise.

3) Set the **Filter Parameters**:

- *Number of Taps* (Filter Length) parameter is set using a sliding bar from 10 to 2048 tap.
- *Initialize filter* vector is selected using a dropdown menu with the following options:
    i. Zeros
    ii. Exponential decay profile
    iii. Random numbers
    iv. Random numbers with exponential decay profile.

4) Set the **Simulation Parameters**:

- *Stopping Criteria:* to stop the adaptation per run according to one of the following two options:
    i. *Number of iterations*: the adaptation per run stops after a specific number of iterations that is set in the edit box.
    ii. *Error threshold*: the adaptation per run stops when the error is smaller than a threshold value that is set in the edit box.

- *Number of runs:* the size of the ensemble to average the results, usually selected between 10 and 20 runs.

    i. An *Input Filter* is selected from the dropdown menu and loaded using the *Load* button. It is used to filter the input signal before adaptation. Four selections are available: Low Pass Filter (LPF), High Pass Filter (HPF), Band Pass Filter (BPF), or None.

5) Set the **Algorithm Parameters**:

- Select the *Adaptive Algorithm* from the pull down menu.
- Set the *Apply Thresholding* radio button if the thresholding technique is required (will be presented in Chapters 6 and 7)
- Set the *Step-size* parameter using the sliding bar to a value between 0 and 2.

6) Set the **Plotting Preferences**:

The top right portion has the *Preferences* section to select the *Performance Criteria* to be displayed (ERLE, Misalignment, MSE, or ERLE and MSE in an external figure). Different plotting color and pen can be selected, and the *Hold* button allows plotting more than one result to compare several simulations.

7) Run the simulation by clicking the *Run* button.

8) Analyze the results:

The top right plot shows the selected performance criteria of the simulated algorithm. The bottom right plot shows the adaptation of the FIR filter over time by navigating forward and backward over the **Snap Shots** using the arrow buttons. A step of the number of iterations can be set in the *Step* edit box. The bottom right plot shows only the impulse response snap shots of the last simulated algorithm averaged over the number of runs. The adapted impulse response can be compared against the actual impulse response model(s) by setting the *Plot Actual EPIR* radio button in the **Analysis Options** section.

3.6 Simulation Conditions

The two typical EPIRs in Figure 3.8 are used in the simulations in Chapter 4, and Chapter 5. Chapter 6 and Chapter 7 use modified versions of these two typical EPIRs that is presented in Chapter 6 to emphasize the effect of attenuated coefficients on the overall performance of the algorithms. When two EPIRs are used in the same simulation, they are used sequentially where each EPIR is used for half the number of iterations per run. This allows each algorithm to be tested for both stationary (static EPIR) and non-stationary (dynamic EPIR) room conditions. The stationary condition is attained by having the algorithm converge fully for the first impulse response to study the convergence speed and the steady-state performance. The non-stationary condition, on the other hand, is attained by changing the impulse response to study the tracking speed. The EPIR with the smartphone lying on its back (Figure 3.8.a) is always used first then

that with the smartphone lying on its face (Figure 3.8.b). The change in the EPIR represents a hugely variant echo path transfer function mimicking highly non-stationary conditions.

The length of all the adaptive filters was the same as the EPIRs, i.e., $L = 1024$ and they were all initialized by zeros. The SNR was 30 dB, the number of iterations was 32000 in Chapter 4, and Chapter 5 and 48000 in Chapter 6, and Chapter 7. The experiments were averaged over an ensemble of 16 independent runs. For a fair comparison, the normalized step-size parameter was set to $\mu_{NLMS} = 0.5$ for all algorithms. The NLMS regularization parameter was fixed to $\delta_{NLMS} = 1.\sigma_x^2$, where σ_x^2 is the variance of the input signal, and the regularization parameter for different algorithms is a function of δ_{NLMS}. The input signal (far-end signal) used is WGN. The performance measure used in all simulations is the ERLE (see section 2.3.3) because the amount of cancelled echo is the most important measure for evaluating speech quality improvement. The values of the parameters used in various algorithms are listed in the legend of the plots of the comparative simulations and mentioned in the text as well.

CHAPTER 4 Variable Individual Step-Size Adaptive Algorithms

This Chapter summarizes selected Proportionate NLMS-based algorithms that are the state-of-the-art in the literature, some of which were developed to improve the performance of acoustic echo cancellation in specific. These algorithms present different approaches to adjust the individual adaptation step-sizes to improve the convergence and tracking performances at the cost of increased complexity. After summarizing the selected PNLMS-based algorithms, they are then simulated to compare their ERLE performance for AEC using the measured EPIRs of the hands-free smartphone presented in Chapter 3. The algorithms are tested for a hugely variant echo path transfer function by using both impulse responses in Figure 3.8 sequentially.

4.1 Introduction to Proportionate NLMS Algorithms

Since acoustic echo paths are long in length, it is desirable to take into account the structure of the impulse response in order to improve the convergence rate and tracking performance of the NLMS algorithm as it is well known that it converges slowly for long filter lengths. Duttweiler proposed the proportionate NLMS (PNLMS) algorithm [**14**] that was designed to exploit the sparsity of network echo path impulse responses. In network echo path impulse responses, only a small number of coefficients of the adaptive filter are significantly different from zero, and are referred to as *active taps*, and all other coefficients are small values or zero, and are referred to as *inactive taps*. The idea of the

PNLMS is to update each coefficient independently of the other, by adjusting individual adaptation step-sizes in proportion to the magnitude of the estimated filter coefficients. The large coefficients thus converge faster initially increasing the overall initial convergence rate. Many algorithms based on the same principle have been proposed since then, with the goal of exploiting the structure of the acoustic echo path impulse response when adjusting the individual step-sizes.

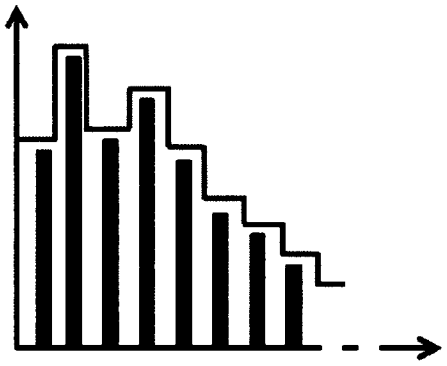


Figure 4.1: Schematic of an impulse response showing proportionate gain.

In order to independently vary individual step-sizes, a vector of the same length as the adaptive filter containing gain-control factors is used to redistribute the available adaptation gain,

$$\bar{g}(n) = [g_0(n) \quad g_1(n) \quad \dots \quad g_{L-1}(n)]^T \tag{4.1}$$

and it is plugged in the weight-update equation using a gain-control matrix,

$$\boldsymbol{G}(n) = diag\{\bar{\boldsymbol{g}}(n)\} \tag{4.2}$$

which is an LxL diagonal matrix containing the gain-control factors of $\bar{g}(n)$. The diag{ . } is the mathematical notation for diagonalization.

The general weight-update equation for VISS adaptive algorithms then becomes

$$\hat{c}(n+1) = \hat{c}(n) + \mu_{NLMS} \frac{\boldsymbol{G}(n)\bar{x}(n)e(n)}{\bar{x}^T(n)\boldsymbol{G}(n)\bar{x}(n) + \delta} \tag{4.3}$$

where μ_{NLMS} is the over-all step-size providing adaptation gain that will be distributed among filter taps using the gain-control matrix. The NLMS algorithm can be reformulated to use the gain-control matrix which in the case of the NLMS is assigned to the identity matrix,

$$\boldsymbol{G}_{NLMS}(n) = \boldsymbol{I} = \begin{pmatrix} 1 & 0 & & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & & 0 & 1 \end{pmatrix}_{LxL} \tag{4.4}$$

The following Figure shows a block diagram for proportionate gain control adaptive algorithms. Different PNLMS-based algorithms differ in the way the **G**(n) matrix is computed.

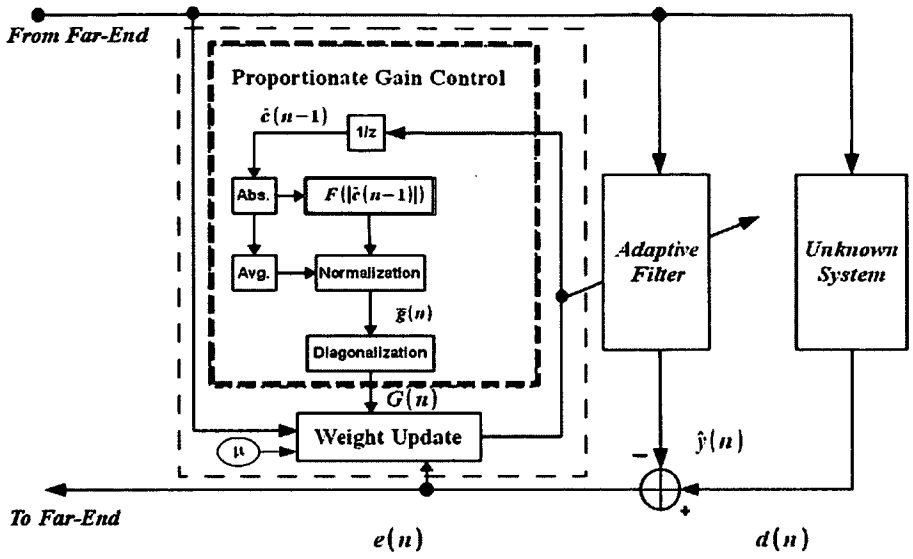


Figure 4.2: Block diagram of proportionate gain control adaptive algorithms.

4.2 Standard Proportionate NLMS Algorithms

4.2.1 Basic PNLMS Algorithm (PNLMS)

The PNLMS algorithm [**14**] assigns an individual step-size to each filter coefficient. The step-sizes are calculated using the magnitude of the last estimate of the filter coefficients such that the largest coefficients receive the largest step-sizes causing them to converge fast. The diagonal elements of the gain-control matrix are computed as follows:

$$\varphi(n) = \rho \, max\{\gamma, |\hat{c}_0(n)|, |\hat{c}_1(n)|, \ldots, |\hat{c}_{L-1}(n)|\} \tag{4.5}$$

$$\theta_l(n) = max\{\varphi(n), |\hat{c}_l(n)|\} \quad \forall l, \qquad l = 0, 1, \ldots L-1 \tag{4.6}$$

$$g_l(n) = \frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} \qquad l = 0, 1, \ldots L-1 \tag{4.7}$$

where the parameters ρ and γ are positive numbers. The parameter ρ prevents the coefficients from stalling at zero and it controls the amount of proportionality provided by the algorithm (the smaller ρ is, the larger the number of coefficients that adapt proportionally). The parameter γ is chosen as the smallest presentable number since its only effect is to force the initial update after a reset providing an equally distributed gain distribution initially like in the NLMS algorithm. The term $\varphi(n)$ is the minimum individual step-size that can be assigned to a filter coefficient. The individual step-sizes, $\theta_l(n)$, are then normalized by their average to provide an equal misalignment error as that of the NLMS algorithm.

The assignment of the individual step-sizes in the PNLMS divides the taps into mainly two groups per iteration, the first group is of the taps that have coefficient magnitudes greater than $\varphi(n) = \rho.|\hat{c}_{max}(n)|$ and the second group is of the taps with coefficient magnitudes smaller than this value. The first group is assigned individual step-sizes that are proportional to their coefficient magnitudes whilst the second group is assigned the same individual step-size with magnitude equal to $\varphi(n) = \rho.|\hat{c}_{max}(n)|$. This can be depicted by the schematic drawing shown in Figure 4.3,

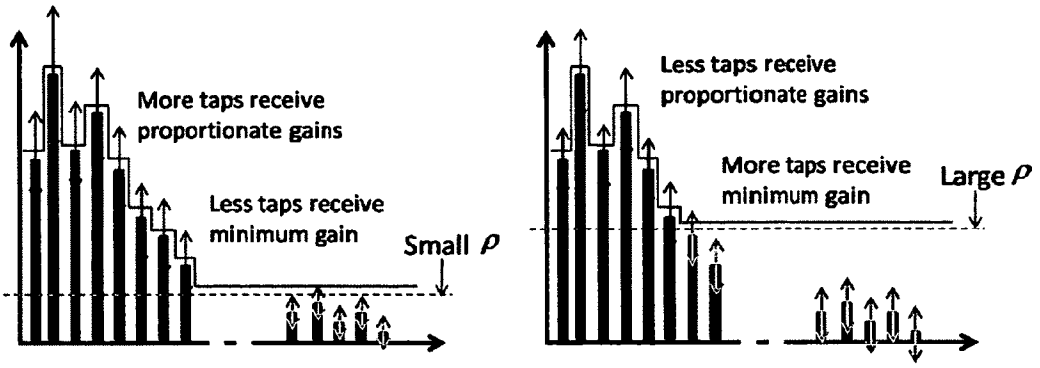
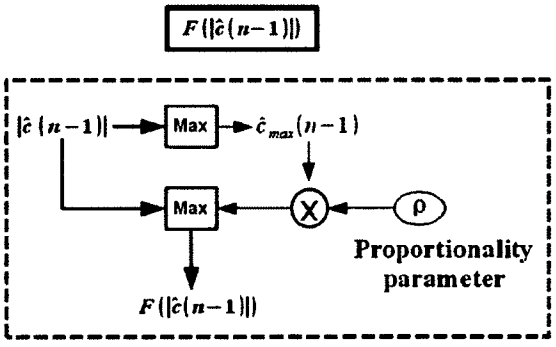


Figure 4.3: Schematic drawing of an impulse response showing the effect of the proportionality parameter on the gain distribution.

The advantage of the PNLMS is that it requires less a priori information about the echo path, since the update depends only on the current filter estimates. This is true when compared to other non-uniform gain-allocation algorithms such as the algorithm proposed by Makino *et al.* [**28**] in which the individual tap adaptation gains fall off exponentially to match a measured exponential decay envelope of measured acoustic echo path impulse responses. The PNLMS algorithm is listed in Table A.2.



Ad-hoc rule based on use of
maximum function

Figure 4.4: Block diagram of gain determination in PNLMS algorithm.

4.2.2 PNLMS++ Algorithm (PNLMS++)

The PNLMS++ [15] is a variant of the PNLMS algorithm. The gain-control vector is assigned alternatively between that of the PNLMS algorithm for odd-numbered iterations and that of the NLMS for even-numbered iterations. The PNLMS++ converges and tracks at a rate near to the faster of the two algorithms. Another way to implement the PNLMS++ is to have the PNLMS adaptation once every k-iterations, which is more appropriate for non-sparse EPIRs such as acoustic EPIRs.

The PNLMS++ is less sensitive to the assumption of a sparse impulse response as is the case in the PNLMS; therefore it may perform better than the PNLMS for acoustic EPIRs, and on average it has a computational complexity less than the PNLMS. The PNLMS++ algorithm is listed in Table A.3.

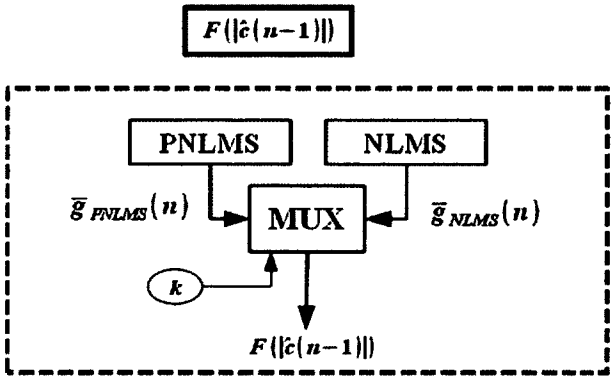


Figure 4.5: Block diagram of gain determination in PNLMS++ algorithm.

4.2.3 Improved PNLMS Algorithm (IPNLMS)

Since the PNLMS was designed to work best with sparse impulse responses, the update rule for the individual step-sizes was focused on converging the few large taps fast (by using the maximum function). This update rule is abrupt and does not work well for distributing the available adaptation gain optimally for non-sparse or denser impulse responses such as acoustic EPIRs.

In the IPNLMS [**16**], an optimal rule is used to better exploit the shape of the estimated impulse response such that the convergence rate and tracking performance are better than both the NLMS and PNLMS algorithms for both sparse and non-sparse impulse responses. The IPNLMS algorithm still makes use of the proportionality idea in the PNLMS but combines it with the NLMS update, which is an even distribution,

$$g_l(n) = \frac{1-\alpha}{2L} + (1+\alpha)\frac{|\hat{c}_l(n)|}{2\|\hat{\mathbf{c}}(n)\|_1 + \epsilon} \tag{4.8}$$

$$l = 0, 1, \ldots L-1$$

where α, $-1 \leq \alpha < 1$, is a parameter that controls the amount of proportional versus even gain distribution: for $\alpha = -1$ the IPNLMS becomes the NLMS algorithm (good for non-sparse EPIRs), for $\alpha \approx 1$ the IPNLMS behaves like the PNLMS (good for sparse EPIRs). The parameter ϵ is a small positive number used to avoid dividing by zero initially when the adaptive filter is reset to zero. The IPNLMS algorithm is listed in Table A.4.

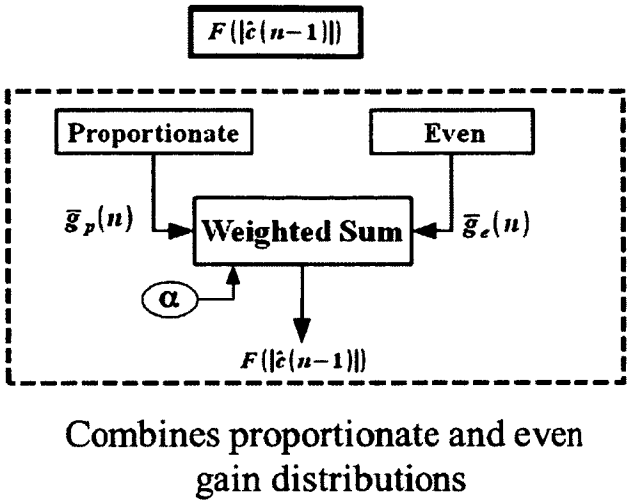


Figure 4.6: Block diagram of gain determination in IPNLMS algorithm.

4.2.4 Simulation of PNLMS, PNLMS++, and IPNLMS

The PNLMS algorithm (see Table A.2) is used as a benchmark to compare the performance of newly developed PNLMS-based algorithms. We compare the ERLE performance of the PNLMS with the NLMS as well as the PNLMS++ (see Table A.3) in Figure 4.7. The parameters used in the PNLMS are set to $\delta_{PNLMS} = \frac{\delta_{NLMS}}{L} = \frac{cst.\sigma_x^2}{L}$, $\rho = \frac{5}{L}$, and $\gamma = 0.001$ [14].

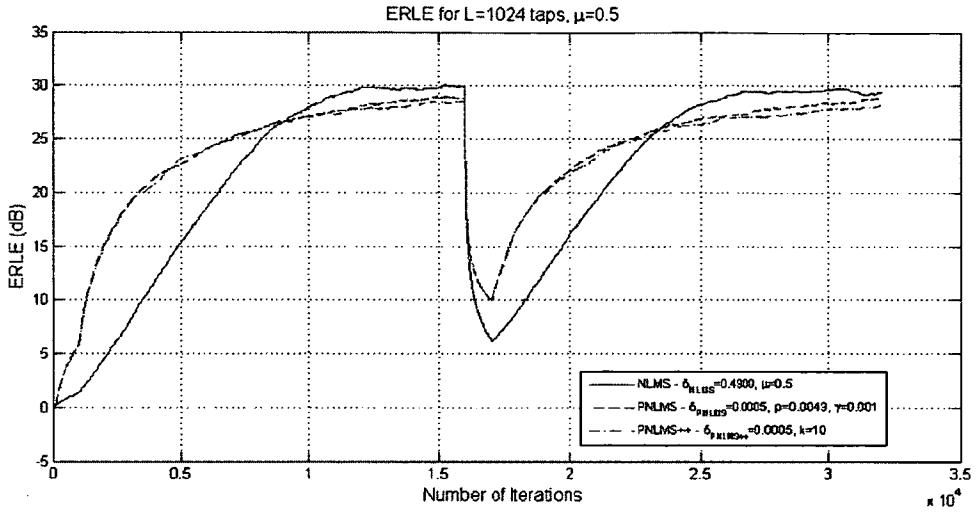


Figure 4.7: Comparative simulation of the NLMS, PNLMS and PNLMS++.

The PNLMS is compared with the NLMS to show the improvement in performance due to the proportionate gain distribution idea. The PNLMS outperforms the NLMS in terms of convergence rate and tracking capability but after an initial phase of faster convergence and faster tracking it slows down and has a lower steady-state performance.

In the PNLMS, more emphasis is given for the update of the large coefficients and this leads to the initial fast convergence rate. However, the convergence rate of the PNLMS slows down after the initial phase because the small coefficients adapt at a rate slower than that of the large coefficients, and the overall convergence rate becomes even slower than that of the NLMS algorithm for acoustic EPIRs. This is also because acoustic EPIRs have a larger number of small-valued taps that are receiving gains less than what they would receive in the NLMS algorithm. This is the result of using an update rule that is not based on any optimal criterion but rather on an ad-hoc way in the PNLMS algorithm.

The PNLMS++ simulation result plotted in Figure 4.7 uses the parameter k set to 10, i.e. one out of every ten iterations uses a PNLMS adaptation and the other nine uses NLMS adaptation. This allows having more NLMS adaptations than PNLMS since the acoustic EPIRs used are non-sparse. The ERLE performance of the PNLMS++ is the same as that of that PNLMS, and the steady-state performance of the PNLMS++ is still worse than that of the NLMS.

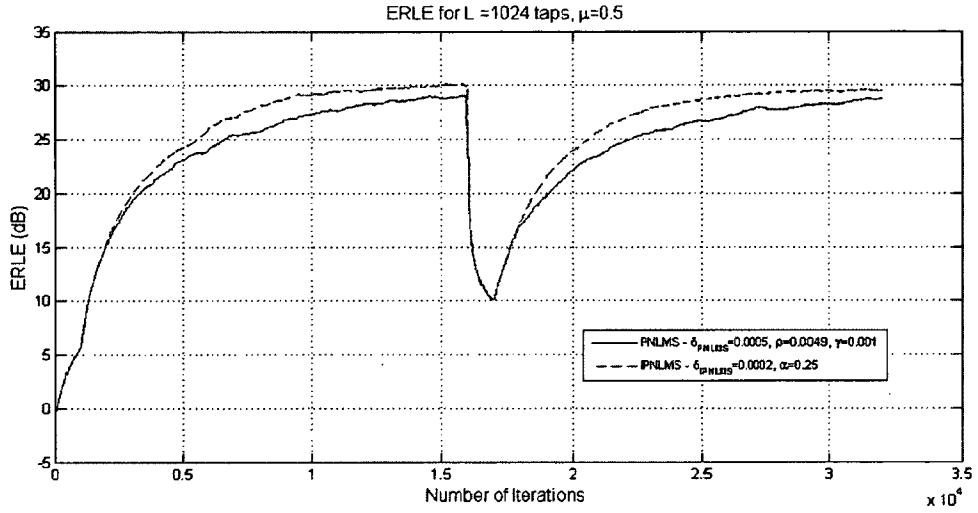


Figure 4.8: Comparative simulation of the PNLMS and IPNLMS.

The IPNLMS (see Table A.4) uses a more optimal update rule to better exploit the shape of the estimated impulse response, and therefore it is more robust with non-sparse EPIRs. The parameters in the IPNLMS were set to $\delta_{IPNLMS} = \delta_{NLMS} \frac{1-\alpha}{2L}$, and $\alpha = 0.25$ **[16]**. As shown in Figure 4.8, the IPNLMS performs better than the PNLMS in terms of convergence rate, tracking capability, and it also converges to the same steady-state level as the NLMS with the acoustic EPIRs. There is an insignificant increase in the amount of

additions required by the IPNLMS, and it does not require any comparison operations when compared to the PNLMS.

4.3 Mu-law Proportionate NLMS Algorithms

4.3.1 Mu-law PNLMS Algorithm (MPNLMS)

The Mu-law PNLMS proposed by Deng in [**17**,**18**] also targets changing the ad-hoc update rule of the PNLMS by finding a new optimal criterion to adjust the individual step-sizes. This optimal rule is based on the steepest-descent method [**26**,**27**]. It is derived such that all the coefficients attain a converged value within a ξ-vicinity of their optimal values (Wiener filter solution [**26**,**27**]) after a minimal number of iterations, where ξ is a small positive number. The derived individual step-sizes update rule is almost the same as that of the PNLMS except it uses the logarithm of the magnitude of the coefficients instead of the magnitudes of the coefficients directly. The following equations summarize the gain-control in the MPNLMS:

$$F(|\hat{c}_l(n)|) = ln(1 + \beta|\hat{c}_l(n)|), \qquad l = 0, 1, \ldots, L-1. \tag{4.9}$$

$$\varphi(n) = \rho max\{\gamma, F(|\hat{c}_0(n)|), F(|\hat{c}_1(n)|), \ldots, F(|\hat{c}_{L-1}(n)|)\} \tag{4.10}$$

$$\theta_l(n) = max\{\varphi(n), F(|\hat{c}_l(n)|)\} \quad \forall l, \qquad l = 0, 1, \ldots L-1 \tag{4.11}$$

$$g_l(n) = \frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} \qquad l = 0, 1, \ldots L-1 \tag{4.12}$$

where the function $F(|\,.\,|)$ is the mu-law function used for non-uniform compression in telecommunication applications [**44**], and the parameter $\beta = 1/\xi$ where ξ is a very small

positive number that depends on the measurement of the noise level. The MPNLMS algorithm is listed in Table A.5.

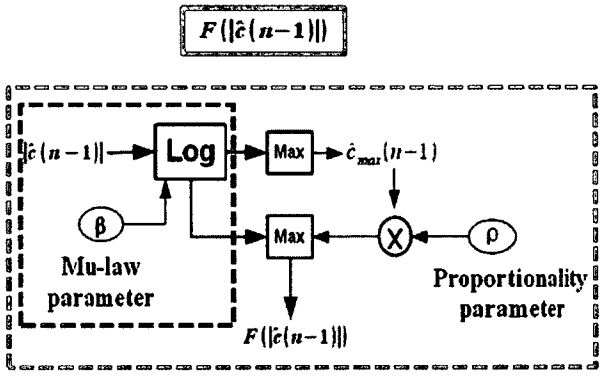


Figure 4.9: Block diagram of gain determination in MPNLMS algorithm.

4.3.2 Segment PNLMS Algorithm (SPNLMS)

To reduce the computational load of the mu-law function of the MPNLMS algorithm, piecewise linear functions are used to approximate it. The Segment PNLMS algorithm [**17**] is considered a cheaper implementation of the MPNLMS algorithm. The following two-segment functions were used in the literature [**17,18**]:

$$F(|\hat{c}_l(n)|) = \begin{Bmatrix} 200|\hat{c}_l(n)| & |\hat{c}_l(n)| < 0.005 \\ 1 & otherwise \end{Bmatrix} \tag{4.13}$$

$$F(|\hat{c}_l(n)|) = \begin{Bmatrix} 600|\hat{c}_l(n)| & |\hat{c}_l(n)| < 0.005 \\ 3 & otherwise \end{Bmatrix} \tag{4.14}$$

The larger the slope of the initial segment the more emphasis is put for the small coefficients. The SPNLMS algorithm is listed in Table A.6.

4.3.3 Adaptive Mu-law PNLMS Algorithm (AMPNLMS)

The performance of the MPNLMS can be improved by varying the mu-law parameter β [19], where $\beta = 1/\xi$ and ξ is a measure of the size of neighborhood that the coefficients are to converge within. The parameter ξ depends on the measurement of the system noise. The idea in the AMPNLMS is to start with a large value for ξ and then decrease it as time proceeds to reduce the ξ-vicinity that the coefficients converge within. This allows the AMPLMS reach the steady-state performance of the NLMS as time proceeds but at a faster rate. The value of $\xi(n)$ is determined using an estimate of the MSE as follows:

$$\hat{\sigma}_e^2(n+1) = \varepsilon\, \hat{\sigma}_e^2(n) + (1-\varepsilon)e^2(n) \tag{4.15}$$

$$\xi(n+1) = \sqrt{\frac{\hat{\sigma}_e^2(n+1)}{\nu\, L\sigma_x^2}} \tag{4.16}$$

$$\beta(n+1) = \frac{1}{\xi(n+1)} = \frac{1}{\sqrt{\frac{\hat{\sigma}_e^2(n+1)}{\nu\, L\sigma_x^2}}} \tag{4.17}$$

where $0 < \varepsilon \leq 1$, and ν is a constant parameter used to relate the current estimate of the MSE to the variance of the input signal. The AMPNLMS algorithm is listed in Table A.7.

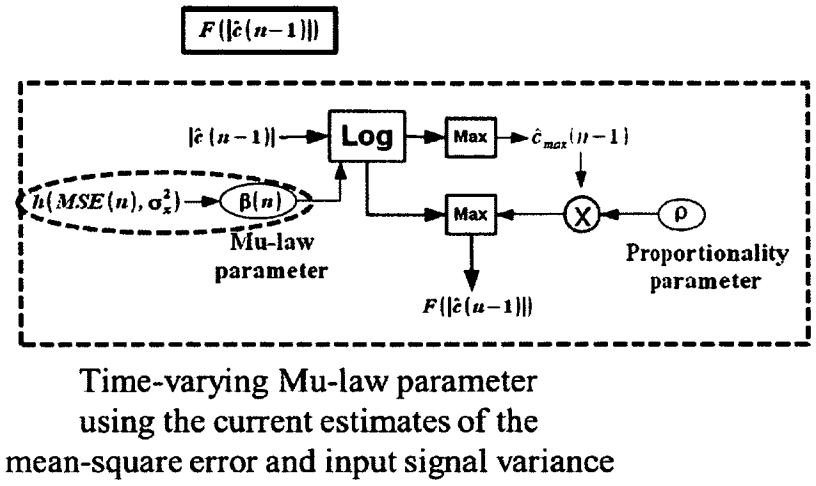


Figure 4.10: Block diagram of gain determination in AMPNLMS algorithm.

4.3.4 Simulation of MPNLMS, SPNLMS, and AMPNLMS

The mu-law algorithms are compared with previously mentioned algorithms in this section. Figure 4.11 shows that the MPNLMS (see Table A.5) outperforms both the PNLMS and the IPNLMS in terms of the convergence rate, and tracking capability. The parameters in the MPNLMS are selected as $\delta_{MPNLMS} = \delta_{NLMS}$, and $\beta = 1000$ is a good choice since the noise level smaller than -60 dB is negligible (i.e. $\xi = 0.001$) [17]. Unlike the PNLMS, as shown in Figure 4.11, the convergence rate of the MPNLMS does not slow down after the initial phase; it stays fast until the adaptation process reaches steady-state. However, the MPNLMS does not reach the same steady-state level (30 dB) of the NLMS when compared to the IPNLMS. This is because there is a limit for the density of the impulse response after which the steady-state performance of the NLMS becomes better than the MPNLMS **[18]** and acoustic EPIRs are dense.

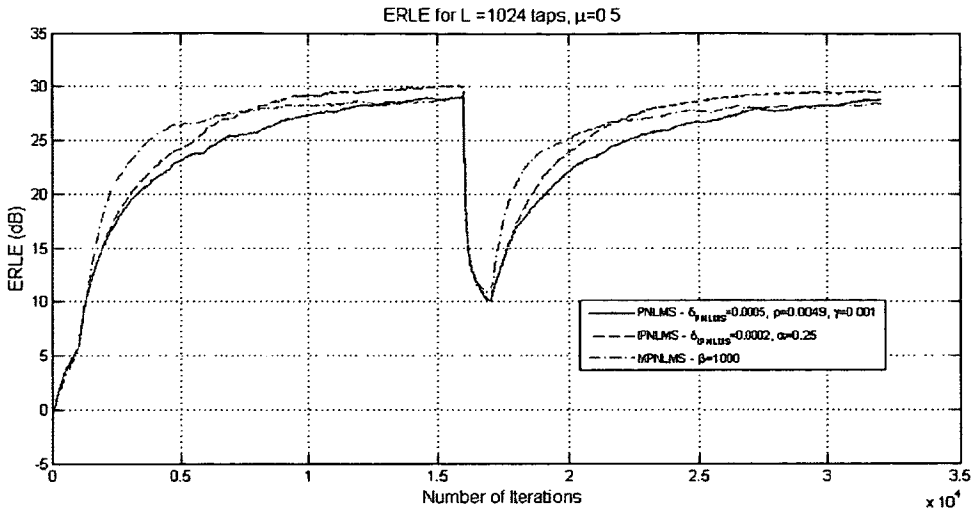


Figure 4.11: Comparative simulation of the PNLMS, IPNLMS, and MPNLMS.

The SPNLMS (see Table A.6) has much less computational complexity than the MPNLMS due to approximating the logarithmic function by using a two-segment function. Figure 4.12 shows that the initial rate of convergence of the SPNLMS is the same as that of the MPNLMS but it then the convergence rate slows down and the ERLE of the SPNLMS reaches a lower steady-state level. Since the SPNLMS is good at tracking, it may serve a cheap and efficient adaptive algorithm for highly non-stationary acoustic environments. Both two-segment functions used in the literature give similar performance using the measured smartphone EPIRs that are used in the simulations.

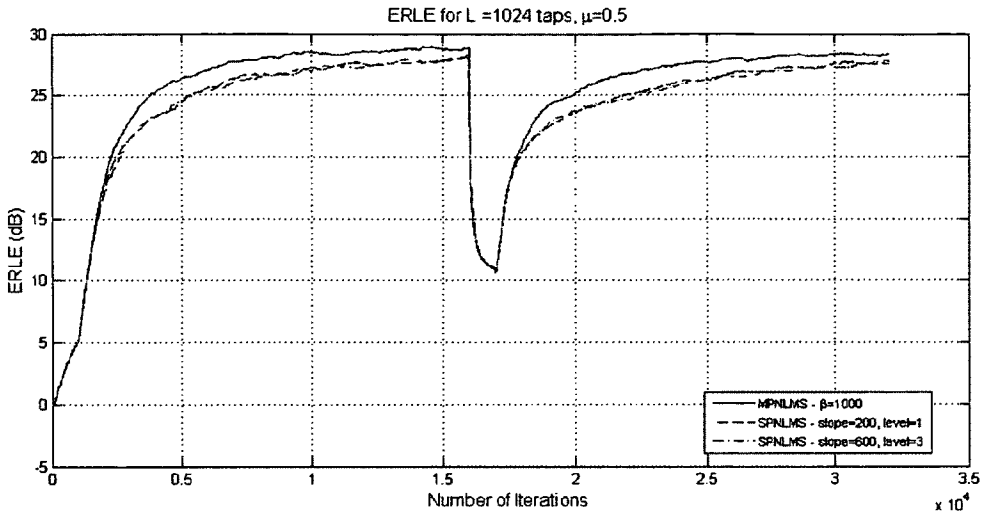


Figure 4.12: Comparative simulation of the MPNLMS, and SPNLMS.

In the AMPNLMS (see Table A.7) the parameters are set to the values: $\delta_{AMPNLMS} = \delta_{NLMS}$, $\varepsilon = 0.99$ and $\nu = 1000$ **[19]**. Figure 4.13 shows that the AMPNLMS has a minute improvement in the convergence rate for the EPIRs of the smartphone and has a lower steady-state performance similar to that of the MPNLMS when compared to the NLMS algorithm (30 dB).

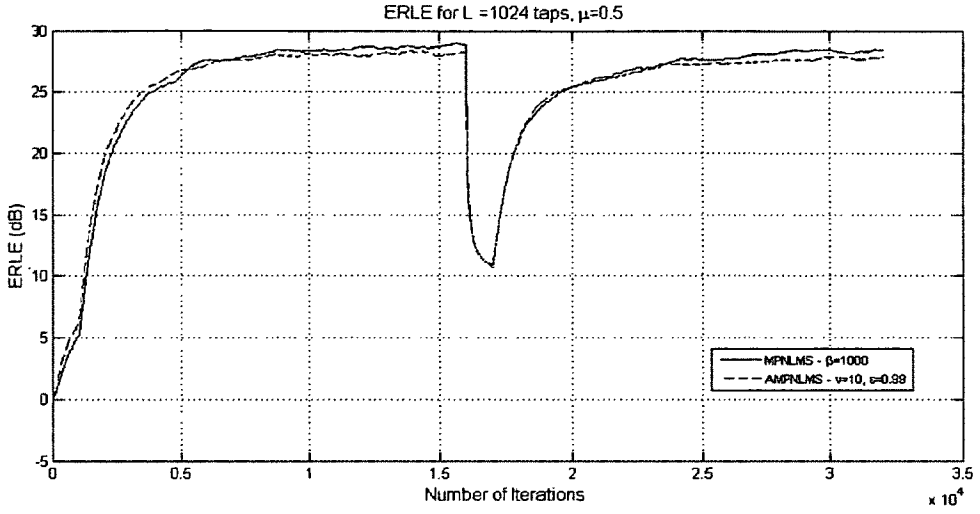


Figure 4.13: Comparative simulation of the MPNLMS, and AMPNLMS.

4.4 Sparseness Controlled Proportionate NLMS Algorithms

The sparsity of the EPIRs is exploited more explicitly in the Sparseness Controlled PNLMS algorithms presented in this section. Acoustic EPIRs are quasi-sparse (i.e. not sparse but not very dense as well), and more dispersive than network EPIRs (i.e. significant taps are not necessarily confined to certain locations). Network EPIRs are sparse and have the most significant taps located at specific sections of the impulse response. A novel approach was proposed in [**20**] that uses a sparseness measure of the current estimate of the adaptive filter coefficients to control the update of the individual step-sizes. This sparseness controlled technique was incorporated into the PNLMS, MPNLMS, and IPNLMS to give the SCPNLMS, SCMPNLMS, and SCIPNLMS algorithms respectively.

The sparseness measure uses the l_1-norm and l_2-norm of the current estimate of the adaptive filter [**45**,**46**],

$$\xi(n) = \xi\left(\hat{\boldsymbol{c}}(n)\right) = \frac{L}{L-\sqrt{L}}\left\{1 - \frac{\|\hat{\boldsymbol{c}}(n)\|_1}{\sqrt{L}\|\hat{\boldsymbol{c}}(n)\|_2}\right\} \tag{4.18}$$

where $0 \leq \xi(n) \leq 1$.

Sparse impulse responses have large $\xi(n)$ values and non-sparse impulse responses have small $\xi(n)$ values. Impulse responses with sparseness value of $\xi(n) < 0.4$ are considered non-sparse.

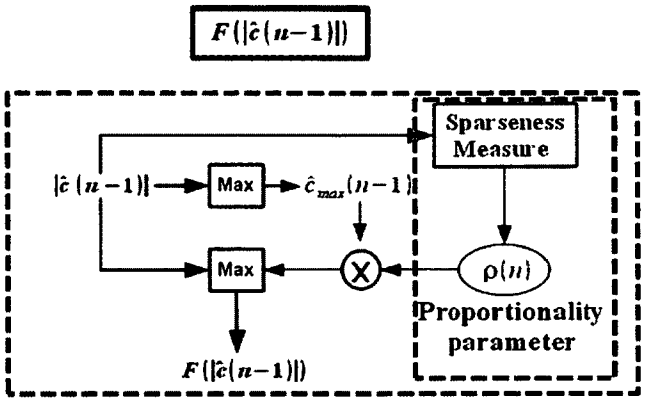
4.4.1 Sparseness Controlled PNLMS Algorithm (SCPNLMS)

The proportionality parameter, ρ, in the PNLMS affects the convergence of the algorithm differently for sparse and non-sparse impulse responses. A small ρ value is required for sparse impulse responses because it increases the proportionality in the update of the individual step-sizes, and a large ρ value is required for non-sparse impulse responses because more filter coefficients are updated evenly. Therefore the sparseness measure $\xi(\mathrm{n})$ is incorporated into ρ for both the PNLMS and MPNLMS algorithms as follows

$$\rho(n) = e^{-\lambda\xi(\hat{\boldsymbol{c}}(\boldsymbol{n}))} \tag{4.19}$$

where λ is a positive number.

Since the l_2-norm in the denominator is equal to zero initially, $\|\hat{\boldsymbol{c}}(n)\|_2 = 0$, therefore to avoid division by zero, $\rho(n) = {}^{5}/_{L}$ is used for the first L iterations after a reset, i.e. $n < L$, and then $\rho(n)$ is computed using $\xi(\hat{\boldsymbol{c}}(\boldsymbol{n}))$ for $n \geq L$ [**20**]. The SCPNLMS algorithm is listed in Table A.8.

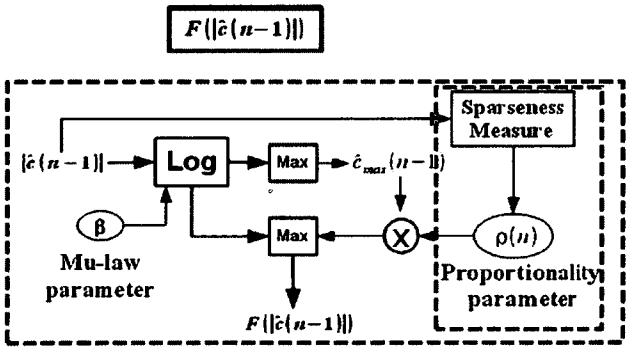


Time-varying proportionality parameter using a measure of sparseness

Figure 4.14: Block diagram of gain determination in SCPNLMS algorithm.

4.4.2 Sparseness Controlled Mu-law PNLMS Algorithm (SCMPNLMS)

The sparseness measure ξ(n) is incorporated into ρ using equation (4.19) with the MPNLMS exactly the same as with the PNLMS. The SCMPNLMS algorithm is listed in Table A.9.



Time-varying proportionality parameter using a measure of sparseness

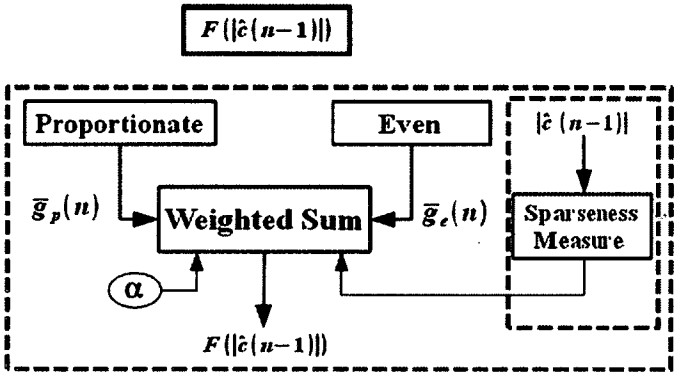
Figure 4.15: Block diagram of gain determination in SCMPNLMS algorithm.

4.4.3 Sparseness Controlled Improved PNLMS Algorithm (SCIPNLMS)

The sparseness measure is incorporated differently in the IPNLMS because the update rule in the IPNLMS involves two terms,

$$g_l(n) = \frac{1-\alpha}{2L}\left(\frac{1-0.5\,\xi(\hat{c}(n))}{L}\right) + (1+\alpha)\left(\frac{1+0.5\,\xi(\hat{c}(n))}{L}\right)\frac{|\hat{c}_l(n)|}{2\|\hat{c}(n)\|_1+\epsilon} \tag{4.20}$$

The SCIPNLMS algorithm is listed in Table A.10.



Another weighting factor
using a measure of sparseness

Figure 4.16: Block diagram of gain determination in SCIPNLMS algorithm.

4.4.4 Simulation of SCPNLMS, SCMPNLMS, and SCIPNLMS

The regularization parameter is set to $\delta_{SCPNLMS} = \frac{\delta_{NLMS}}{L}$, and λ is set to 6 since it provides good convergence rate for non-sparse impulse responses [**20**]. The SCPNLMS (see Table A.8) has almost the same performance as the PNLMS with a slight improvement in the steady-state performance as shown in Figure 4.17.

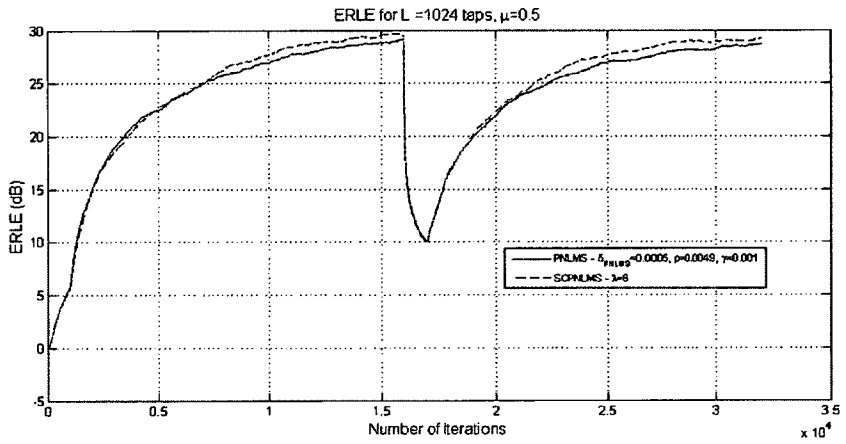


Figure 4.17: Comparative simulation of the PNLMS, and SCPNLMS.

The IPNLMS still outperforms the SCIPNLMS (see Table A.10) for acoustic EPIRs as shown in Figure 4.18. However, the SCIPNLMS also reaches the same steady-state as the NLMS.

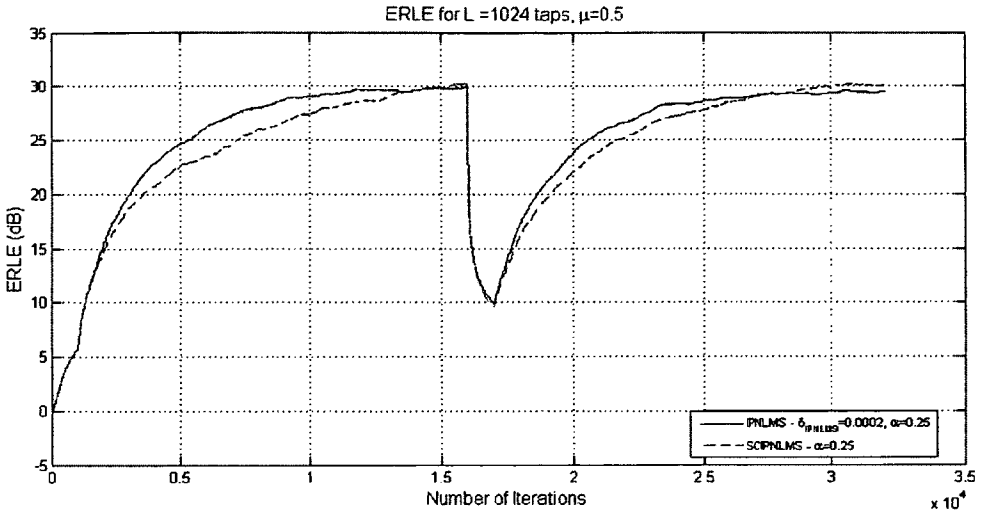


Figure 4.18: Comparative simulation of the IPNLMS, and SCIPNLMS.

The regularization parameter in the SCMPNLMS is set to $\delta_{SCMPNLMS} = \delta_{NLMS}$, and λ is set to 6 in one simulation and 4 in another [**20**]. The SCMPNLMS (see Table A.9) shows an improvement in the steady-state ERLE performance when compared to the MPNLMS in Figure 4.19 when λ is set to 4. The tracking capability of both algorithms is comparable.

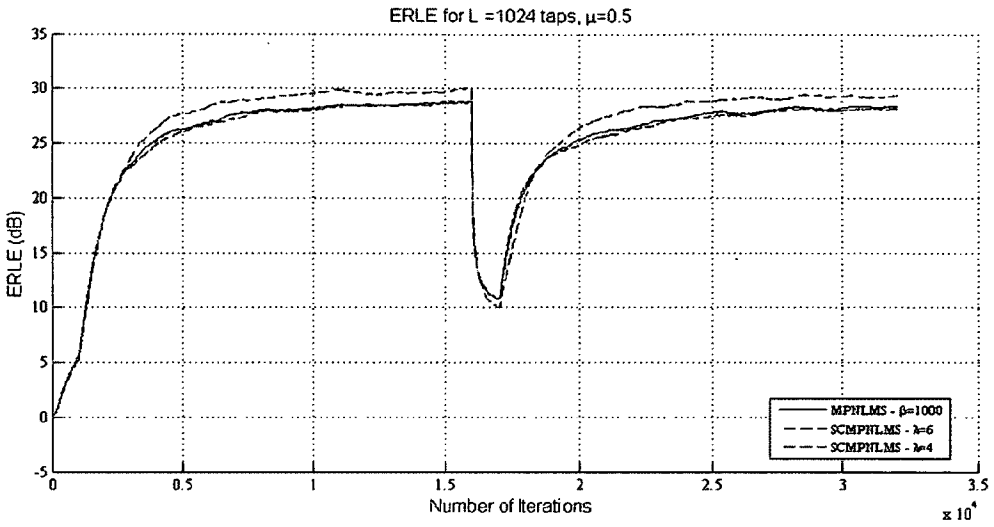


Figure 4.19: Comparative simulation of the MPNLMS, and SCMPNLMS.

4.5 Generalized Gradient PNLMS Algorithm (GGPNLMS)

4.5.1 Algorithm Description

The tracking performance of adaptive algorithms is very important in fast changing acoustic environments which are mainly caused by the motion of objects. The Generalized Gradient PNLMS algorithm proposed by Hoshuyama et al in [**21**] tries to mimic the changes in the acoustical environment by controlling the individual step-sizes based on an estimate of the gradient of the filter coefficients' adaptation. The gradient of a coefficient's adaptation is computed as the difference between the current estimate of the coefficient's magnitude and a time-averaged magnitude with a delay. The main concept here is to depend more on the time-varying component of the filter coefficient adaptation to enhance the tracking capability of the algorithm rather than depending on

the magnitude of the coefficient which includes both time-variant and time-invariant components. The following equations summarize the gain-control of the GGPNLMS:

$$\tilde{\boldsymbol{c}}(n) = \eta\, \tilde{\boldsymbol{c}}(n-1) + (1-\eta)\hat{\boldsymbol{c}}(n-1) \tag{4.21}$$

$$\hat{\boldsymbol{s}}(n) = \hat{\boldsymbol{c}}(n-1) - \gamma\, \tilde{\boldsymbol{c}}(n-1) \tag{4.22}$$

$$\tilde{\boldsymbol{s}}(n) = \epsilon\, \tilde{\boldsymbol{s}}(n-1) + (1-\epsilon)\, abs\{\hat{\boldsymbol{s}}(n)\} \tag{4.23}$$

$$g_l(n) = \frac{1-\beta}{2}\frac{\tilde{s}_l(n-1)}{\sum_{l=0}^{L-1}\tilde{s}_l(n-1)} + \frac{\beta}{2L} \qquad l = 0, 1, \ldots L-1 \tag{4.24}$$

A time-averaged estimate of the coefficients' magnitude, $\tilde{\boldsymbol{c}}(n)$, as computed in equation (4.21) and then is subtracted from the current estimate of the coefficients, $\hat{\boldsymbol{c}}(n)$, to find an estimate of the gradient, $\hat{\boldsymbol{s}}(n)$, as in equation (4.22). A time-averaged estimate of the magnitude of the gradient is then computed in equation (4.23). This time-averaged gradient is then used in the individual step-size update rule as shown in equation (4.24). Since the gradient has no time-invariant components, The GGPNLMS should provide faster tracking performance.

The parameters η and ϵ are forgetting factors used to control the size of the time averaging window, where $0 \leq \eta \leq 1$, and $0 \leq \epsilon \leq 1$. The parameter γ is a correction factor for finding the time-averaged estimate of the coefficients, $\tilde{\boldsymbol{c}}(n)$. This time-averaged estimate of the coefficients contains the time-invariant component which is subtracted from the current filter coefficient estimate to find the gradient. The parameter β, $-1 \leq \beta < 1$, is used to control the proportionality to the gradient for gain distribution similar to the role of the parameter α in the IPNLMS update rule in equation (4.8). An

obvious drawback of the GGPNLMS is the increased computational complexity and memory requirements. The GGPNLMS algorithm is listed in Table A.11.

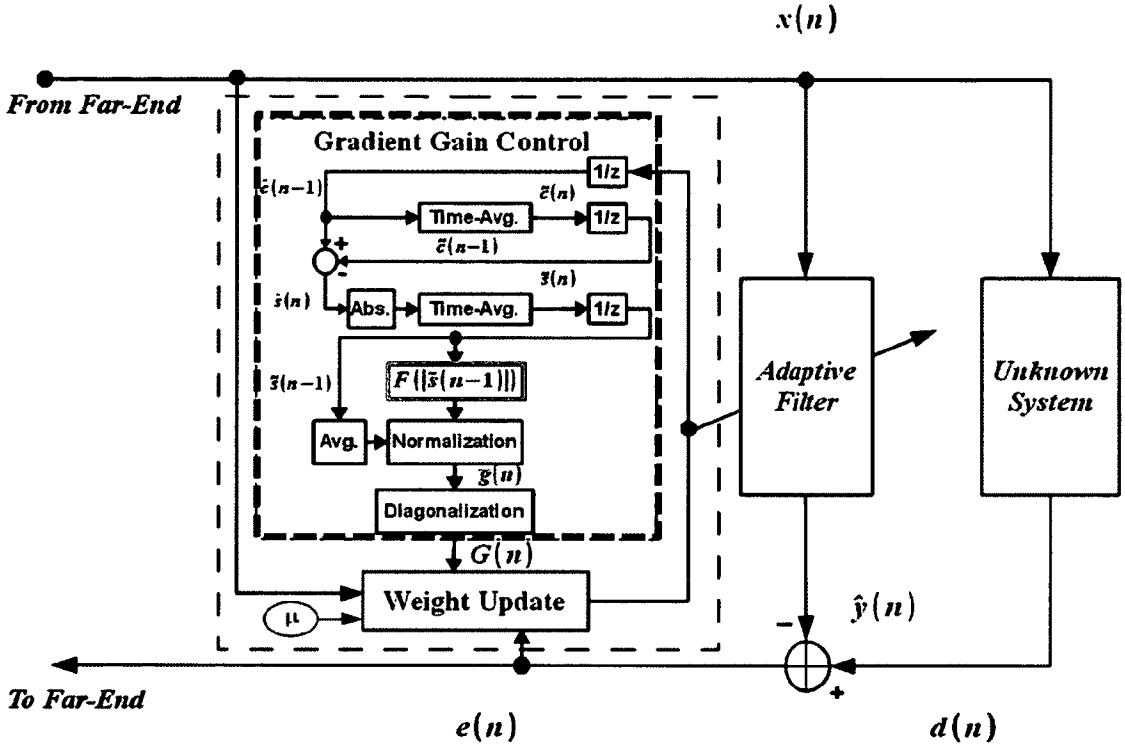
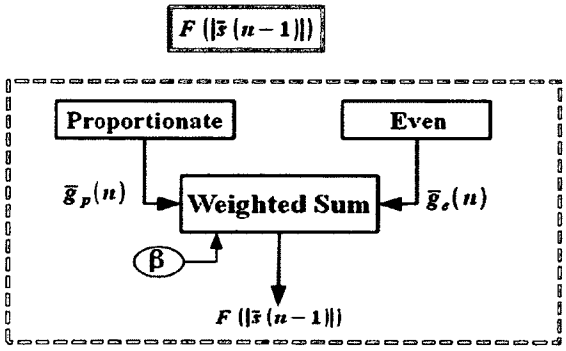


Figure 4.20: Block diagram of gradient gain control adaptive algorithms.



Combines gradient (proportionate to gradient) and even gain distributions

Figure 4.21: Block diagram of gain determination in GGPNLMS algorithm.

4.5.2 Simulation of GGPNLMS

The GGPNLMS (see Table A.11) uses an estimate of the gradient of the coefficients' adaptation to improve the tracking speed. The GGPNLMS parameters are set to $\delta_{GGPNLMS} = \frac{\delta_{NLMS}}{L}, \eta = 0.9999$, $\epsilon = 0$, and $\gamma = 1$ [21]. Figure 4.22 shows a comparison of the GGPNLMS with the IPNLMS and the AMPNLMS. The GGPNLMS has a slightly better tracking speed than the IPNLMS, but the AMPNLMS outperforms both the GGPNLMS and the IPNLMS in terms of convergence speed and tracking capability. The GGPNLMS and the IPNLMS, on the other hand, reach the steady-state performance of the NLMS but the AMPNLMS does not.

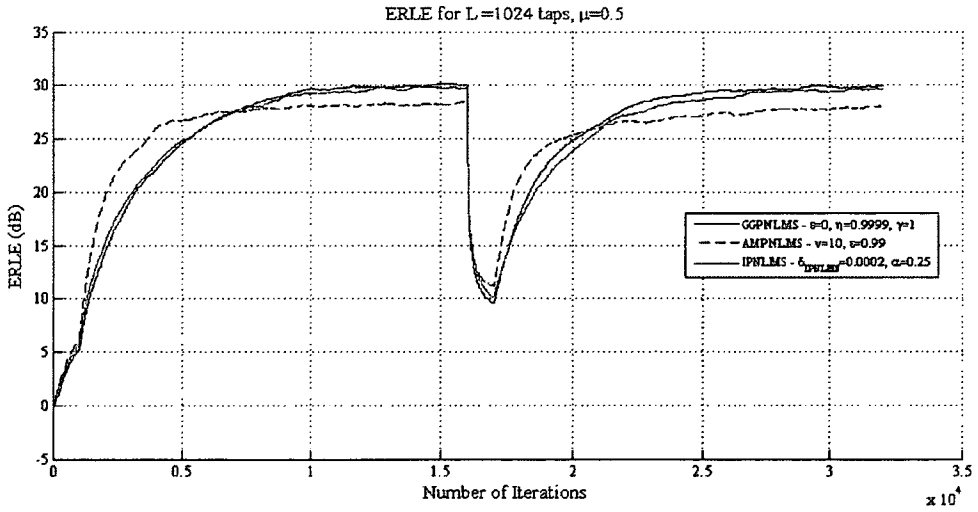


Figure 4.22: Comparative simulation of the GGPNLMS, IPNLMS, and AMPNLMS.

Figure 4.23 shows that the SCMPNLMS has an overall better performance than the GGPNLMS and AMPNLMS using the measured hands-free smartphone acoustic EPIRs.

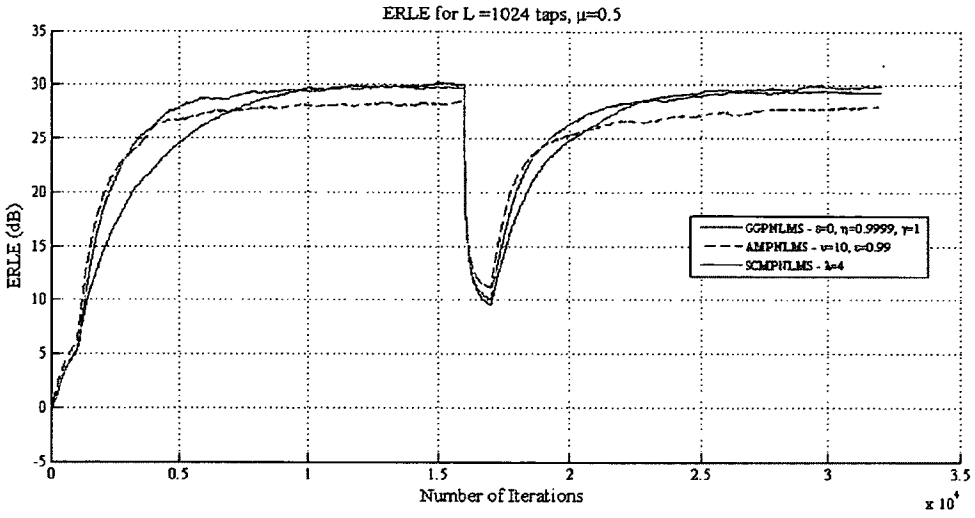


Figure 4.23: Comparative simulation of the GGPNLMS, SCMPNLMS, and AMPNLMS.

4.6 Conclusion

In this chapter, a set of selected state-of-the-art PNLMS-based algorithms are summarized in terms of their different approaches to updating individual step-sizes for adaptive filtering. The algorithms are simulated for AEC using the measured EPIRs of smartphone operating in hands-free mode that are presented in Figure 3.8. The ERLE performances of the algorithms are compared in terms of the convergence rate, tracking capability and steady-state performance.

The performance of the algorithms improves as more sophisticated update rules are used but at the expense of increased computational complexity. The PNLMS showed improved initial convergence and tracking performance when compared to the NLMS. However, the PNLMS++ did not show any improvement over the PNLMS for the measured EPIRs. The IPNLMS has an overall better performance than the PNLMS and reaches the same steady-state ERLE level as that of the NLMS. The Mu-law PNLMS

algorithms have in general better convergence and tracking performance when compared to the PNLMS and IPNLMS but a worse steady-state level due to the increased density of the EPIRs. Sparseness Controlled PNLMS algorithms did not show much significant improvement in performance except for an improvement in the steady-state performance of the SCMPNLMS when compared to the MPNLMS. The AMPNLMS showed a better convergence rate and tracking capability when compared to the GGPNLMS for the measured EPIRs, however, its steady-state performance is worse. The SCMPNLMS proved to have the best overall ERLE performance for the smartphone EPIRs.

CHAPTER 5 Proposed Variable Individual Step-Size Adaptive Algorithms for Acoustic Echo Cancellation in Non-Stationary Environments

In this Chapter, we first introduce a new technique, the gradient-induced technique, and incorporate it into the PNLMS algorithm to give the gradient-induced PNLMS (g-PNLMS) algorithm. The g-PNLMS is designed to provide improved tracking performance in highly non-stationary acoustic environments. The technique is also incorporated into various PNLMS-based algorithms presented in Chapter 4. We then present a modified VISS adaptive algorithm, the Adaptive Sparseness Controlled Mu-law PNLMS (ASCMPNLMS) algorithm. Finally, we present the simulation results for all the algorithms using the measured smartphone EPIRs (presented in Chapter 3).

5.1 Gradient Induced PNLMS Algorithms (g-PNLMSs)

5.1.1 A Non-Stationary Acoustic Environment Scenario

The acoustic environment that is in proximity to the portable device usually contains objects in continuous motion. For example, when the device is held in hands during a hands-free call or when its orientation changes as it is placed on a surface, or mounted on a holder. These changes affect the acoustic EPIR greatly especially at the initial reflections segment which constitutes the echo signals with the largest magnitudes. In

such a highly non-stationary environment the quality of the speech during the conversation might be vastly degraded. An adaptive filter that has good tracking capabilities is required to be able to track these changes in the impulse response as fast as possible.

The schematic diagram in Figure 5.1 shows a case of how the filter taps of an acoustic EPIR may vary in a highly non-stationary environment. It shows that taps with relatively large magnitudes would also have significant variations, whereas other taps with small magnitudes may vary insignificantly. During convergence of the filter from reset at initialization, the taps with relatively larger magnitudes require larger step-sizes to converge to their final values, however, during tracking the taps with the largest variations in magnitude require the largest step-sizes to reach their new values as fast as possible. Since the acoustic EPIR is non-sparse, this means that there is a large number of taps that have significant magnitudes and need to be tuned. This poses a challenge during tracking for adaptive algorithms used for AEC. Therefore, a new approach to distribute the available adaptation gain more efficiently during tracking is required.

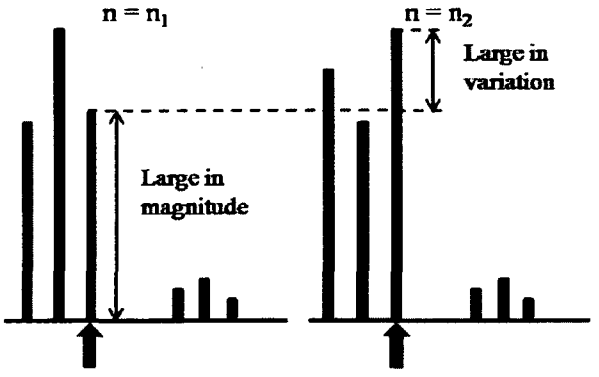


Figure 5.1: Schematic drawing of an impulse response variation scenario showing taps with large magnitudes undergoing large variations overtime.

Figure 5.2 shows the typical curve of the ERLE performance for an adaptive algorithm. The dotted line shows how the tracking needs to be improved when a change in the impulse response takes place such as in highly non-stationary acoustic environments.

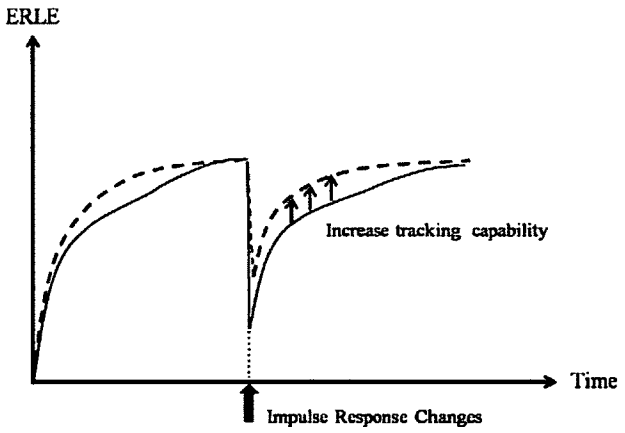


Figure 5.2: Typical ERLE performance curve for an adaptive algorithm showing how tracking performance needs to be improved in highly non-stationary acoustic environments.

Figure 5.3 shows a controlled simulation of an impulse response change scenario that is designed to mimic the highly non-stationary environment described above. The magnitudes of the taps in the initial reflections segment of the echo path can be varied significantly by shuffling the taps randomly.

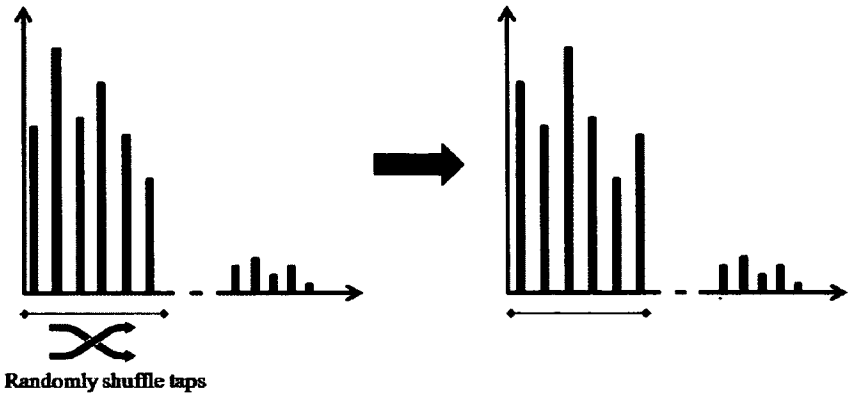
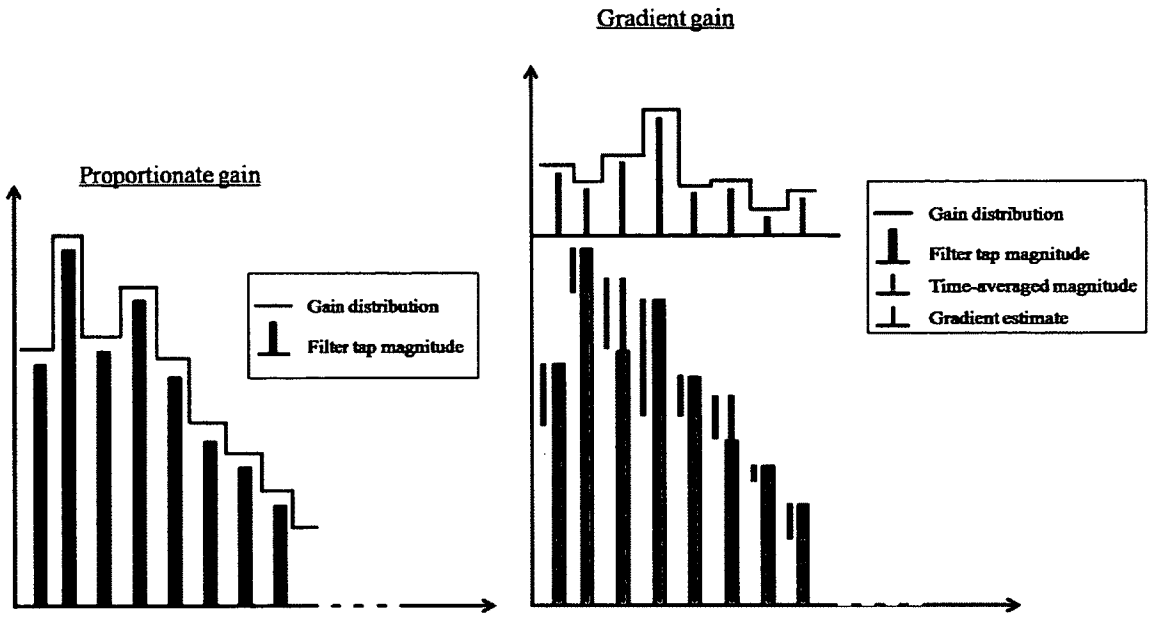


Figure 5.3: A controlled simulation of an impulse response variation scenario where the taps in the initial reflections segment undergo huge variations.

5.1.2 Algorithm Description

The variation of individual step-sizes to distribute the available gain among the coefficients can be categorized into two different approaches: 1) depending on the current estimate of the magnitude of the filter coefficients, maybe referred to as *proportionate-gain* distribution, and 2) depending on the estimate of the variation in the estimated magnitude, i.e. an estimate of the gradient, and maybe referred to as *gradient-gain* distribution. Figure 5.4 presents a schematic showing the gain distribution in the two categories.



a) Proportionate gain distribution.

b) Gradient gain distribution.

Figure 5.4: A schematic of an impulse response showing how adaptation gain is distributed in different VISS algorithms.

Most PNLMS-based algorithms proposed in the literature, and summarized in Chapter 4, are from the first category, maybe referred to as proportionate-gain VISS algorithms, such as the PNLMS [**14**], IPNLMS [**16**], MPNLMS [**17**], and SCPNLMSs [**20**]. A block diagram for the gain-control in this category is shown in Figure 5.5 where the function $F(|\hat{c}(n-1)|)$ is a general function that differs from an algorithm to the other as summarized in Chapter 4, and $\hat{c}(n-1)$ is the estimate of the magnitude of the filter coefficients.

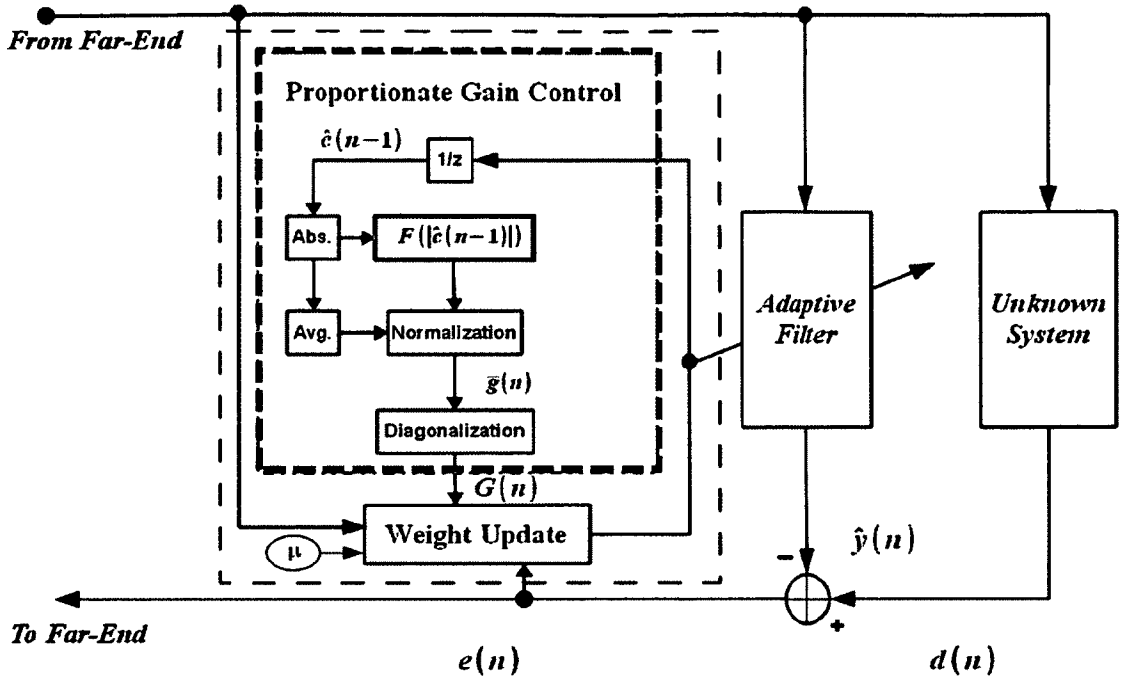


Figure 5.5: General block diagram of VISS algorithms that use the magnitude of the coefficients to distribute the adaptation gain.

The GGPNLMS [21] is considered from the second category, and maybe referred to as a gradient-gain VISS algorithm. A block diagram in Figure 5.6 shows the gain-control for the second category, where again the function $F(|\tilde{s}(n-1)|)$ can, in general, take different forms, and $\tilde{s}(n-1)$ is an estimate of the gradient.

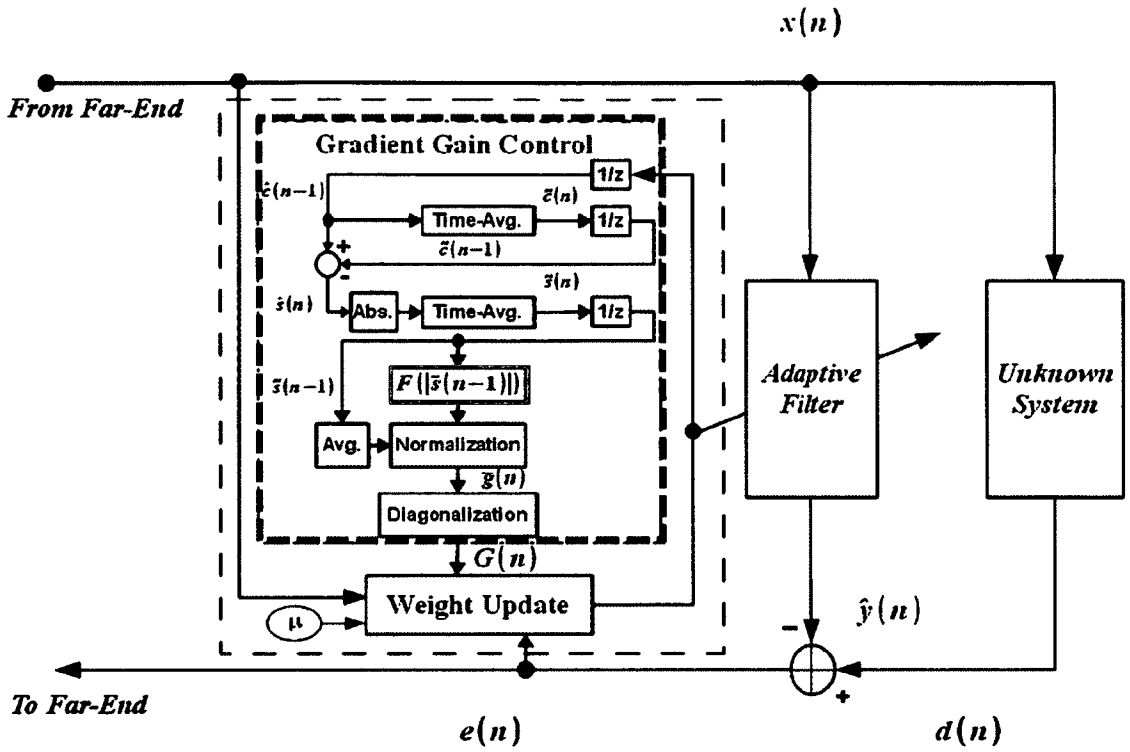


Figure 5.6: General block diagram of VISS algorithms that use the gradient of the coefficients to distribute the adaptation gain.

The goal in designing a new VISS adaptive algorithm in this chapter is to be able to provide fast tracking performance in highly non-stationary acoustic environments. Since the filter coefficients that have large magnitudes are more probable to vary in large amounts it becomes intuitive to use both the current estimate of the magnitude as well as the estimate of the gradient to distribute the adaptation gain. Figure 5.7 shows a block diagram for the proposed gradient-induced VISS algorithms. The proportionate-gain and the gradient-gain vectors are combined using a weighted sum where the contribution of each vector is controlled by a constant parameter, r, where $0 < r < 1$.

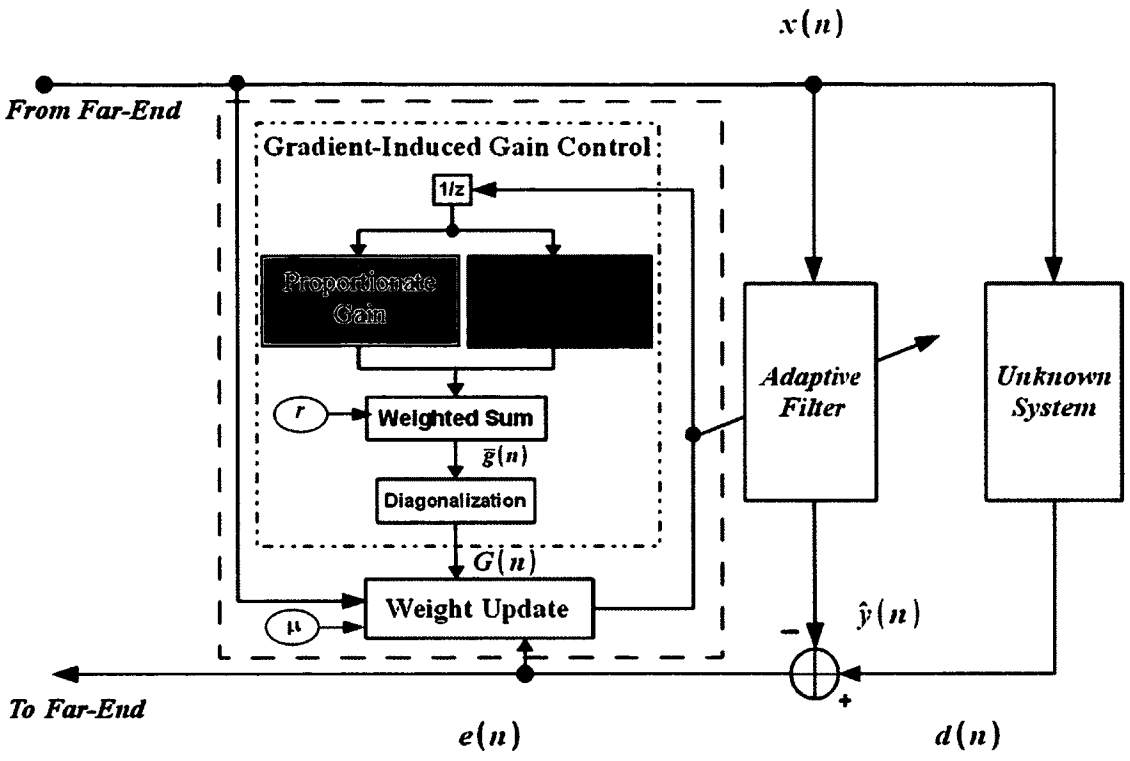


Figure 5.7: Block diagram of the proposed gradient-induced VISS algorithm that uses both proportionate and gradient gains to distribute the adaptation gain.

The idea of employing the gradient of the coefficients' adaptation in the update-rule was first suggested by Hoshuyama et al. in the GGPNLMS [21] which resulted in an improved tracking performance. The update rule for gain distribution in the GGPNLMS, however, relies on the estimate of the gradient only (equation (4.24)), but not on the magnitude of the coefficients.

In the gradient-induced PNLMS algorithms (g-PNLMSs), we propose a new update rule that relies in part on the estimate of the gradient of the coefficients, the gradient-gain, and in part on the magnitude of the coefficients, the proportionate-gain. The gain-control vector is computed as follows,

$$g_l(n) = r\frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} + (1-r)\frac{\Psi_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\Psi_l(n)} \tag{5.1}$$

$$l = 0, 1, \ldots L-1$$

where θ_l is the proportionate-gain at tap location l (as used in Chapter 4), Ψ_l is the gradient-gain at tap location l, and r is the constant parameter, where $0 < r < 1$.

The gradient-gain vector is computed as follows,

$$\Psi_l(n) = max\{\rho(n)\, max\{\gamma, \tilde{s}_0(n), \tilde{s}_1(n), \ldots, \tilde{s}_l(n)\}, \tilde{\boldsymbol{s}}(n)\} \tag{5.2}$$

$$\forall l, \qquad l = 0, 1, \ldots L-1$$

where $\tilde{\boldsymbol{s}}(n)$ is the time-averaged estimate of the magnitude of the gradient and is computed as follows,

$$\tilde{\boldsymbol{s}}(n) = \; abs\{\,\hat{\boldsymbol{c}}(n-1) - \;\tilde{\boldsymbol{c}}(n-1)\} \tag{5.3}$$

which is deduced by setting the parameters to $\gamma = 1$, $\epsilon = 0$ in equations (4.22) and (4.23). $\tilde{\boldsymbol{c}}(n)$ is the time-averaged estimate of the filter coefficients and is given by equation (4.21) in the GGPNLMS algorithm (see section 4.5).

The proportionate-gain, θ_l, may, in general, take on any of the various proportionate-gain updates of different PNLMS-based algorithms, extending the gradient-induced PNLMS (g-PNLMS) to other PNLMS-based algorithms. The special cases for the IPNLMS

(equation (4.7)) and the SCIPNLMS (equation (4.20)) have the following gain-control vectors

$$g_l(n) = r\left(\frac{1-\alpha}{2L} + (1+\alpha)\frac{|\hat{c}_l(n)|}{2\|\hat{\mathbf{c}}(n)\|_1 + \epsilon}\right) + (1-r)\frac{\Psi_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\Psi_l(n)} \tag{5.4}$$

$$l = 0, 1, \ldots L-1$$

for the gradient-induced IPNLMS (g-IPNLMS), and

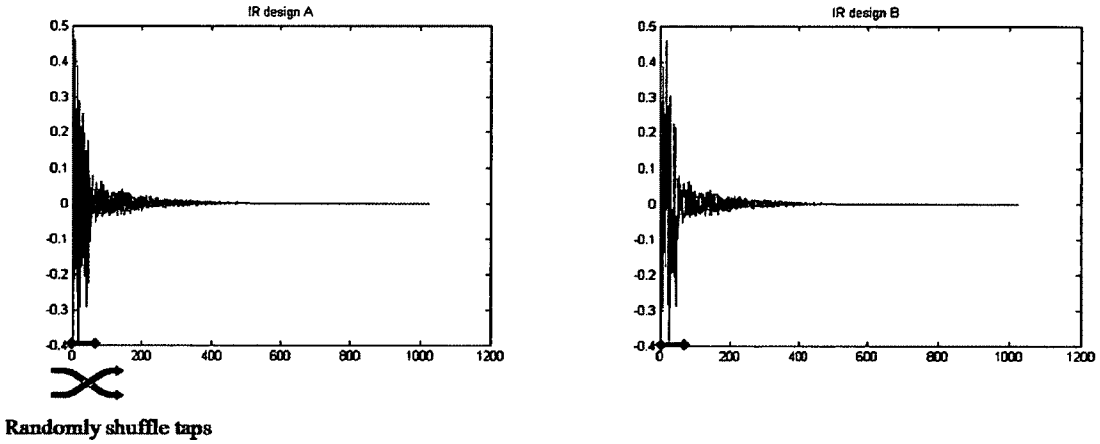
$$g_l(n) = r\left(\frac{1-\alpha}{2L}\left(\frac{1-0.5\,\xi(\hat{\mathbf{c}}(n))}{L}\right) + (1+\alpha)\left(\frac{1+0.5\,\xi(\hat{\mathbf{c}}(n))}{L}\right)\frac{|\hat{c}_l(n)|}{2\|\hat{\mathbf{c}}(n)\|_1 + \epsilon}\right) + (1-r)\frac{\Psi_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\Psi_l(n)} \tag{5.5}$$

for the gradient-induced SCIPNLMS (g-SCIPNLMS).

The gradient-induced technique requires an additional L additions, $2L+1$ multiplications, 2 divisions, and $2L$ comparisons per iteration. The total computational complexity depends on the choice of a PNLMS-based algorithm to compute the proportionate-gain vector.

5.1.3 Preliminary Results

A controlled simulation was carried out to mimic an impulse response change scenario where the magnitudes of the taps in the initial reflections of the echo path vary significantly. The simulated impulse responses shown in Figure 5.8 have the typical acoustic EPIR structure; initial random reflections and a general exponentially decaying profile. Two impulse responses are designed to simulate a highly non-stationary tracking scenario as the impulse response changes from the first to the second impulse response. The second impulse response, shown in Figure 5.8.b, is produced by shuffling the taps of the initial segment of the impulse response in Figure 5.8.a randomly. This segment is approximately 3 ms in length.



a) Simulated impulse response showing the initial 3 ms segment that is randomly shuffled.

b) Modified impulse response showing the segment of shuffled taps.

Figure 5.8: Simulated impulse responses mimicking huge variations in the magnitudes of taps in the initial reflections segment.

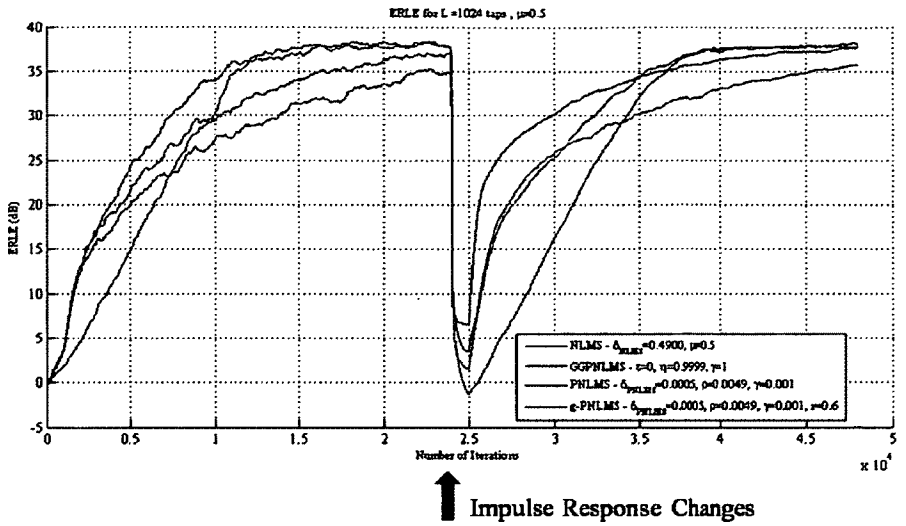


Figure 5.9: Comparative simulation of the NLMS, PNLMS, GGPNLMS, and g-PNLMS using the simulated impulse responses in Figure 5.8.

The ERLE simulation results for the NLMS, PNLMS, GGPNLMS, and g-PNLMS using the simulated impulse responses are shown in Figure 5.9. The results show that the g-PNLMS outperforms all other algorithms in terms of tracking capability. The GGPNLMS is faster than the g-PNLMS in terms of the convergence speed since the magnitudes of the coefficients are reset to zero at initialization; therefore by depending solely on the gradient-gain, the GGPNLMS converges faster than the g-PNLMS algorithm.

5.2 Adaptive Sparseness Controlled Mu-law PNLMS Algorithm (ASCMPNLMS)

5.2.1 Algorithm Description

Varying the constant algorithm parameters has been a successful approach to improve the performance of existing adaptive algorithms. Examples include varying the mu-law parameter, β, in the MPNLMS to give the Adaptive MPNLMS (AMPNLMS) algorithm, see equation (4.17). Another example is varying the proportionality parameter, ρ, in the PNLMS and MPNLMS to give the SCPNLMS and SCMPNLMS respectively (see equation (4.19)). These parameters are made time-variant by depending on the characteristics of different system signals. This dependency in turn makes the adaptation process more sensitive to the changes in the environment such as the variations in background noise and the dynamics of the echo path impulse response. Moreover, the automation of the algorithm parameters makes the algorithm require no a priori information about the system and improves the performance and robustness of the algorithm.

In this section, we present a new adaptive algorithm by automatically tuning the scaling parameter λ in the SCMPNLMS (see equation (4.19)) that affects the algorithm performance depending on the impulse response characteristics. Since the selection of large positive values for the parameter λ (typically 4 or 6 in the SCMPNLMS) provides good convergence rate for non-sparse impulse responses **[20]**, the profile of the change in λ was limited between the values 8 and 4.

By making λ a variable we get

$$\rho(n) = e^{-\lambda(n)\xi(n)} \tag{5.6}$$

Figure 5.10 shows a 3-D plot of the dependency of $\rho(n)$ on $\lambda(n)$ and $\xi(n)$. A notable observation is that $\rho(n)$ is in general large for non-sparse impulse responses (impulse responses with small sparseness values, $\xi(n) < 0.4$), and this is because the algorithm needs to be less proportional since there are more significant coefficients that require more even distribution of the available gain when the impulse response is non-sparse.

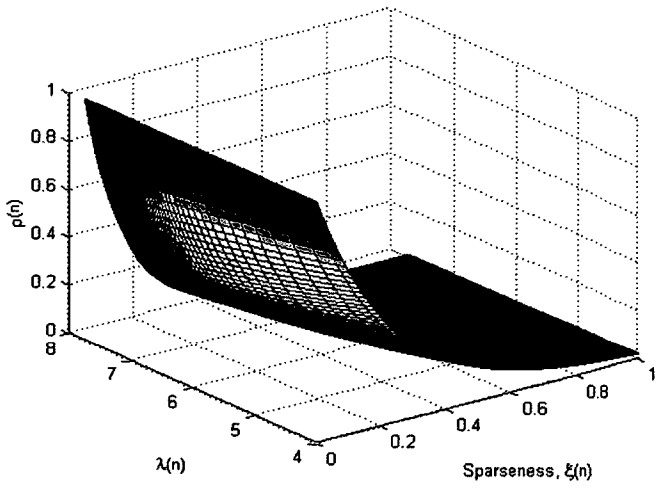


Figure 5.10: The dependency of the parameter $\rho(n)$ on the sparseness of the impulse response, $\xi(n)$, and the parameter $\lambda(n)$ as given by equation (5.6).

This can be explained by examining the effect of $\rho(n)$ on the distribution of the gain in PNLMS algorithms as shown in Figure 5.11. When $\rho(n)$ is small, the largest coefficients receive larger proportionate gains and the small coefficients receive smaller equal gains, i.e. the algorithm becomes more proportionate, and therefore most of the available adaptation gain is consumed as proportionate gain and smaller coefficients are deprived.

When $\rho(n)$ is large, more adaptation gain is used equally rather than proportionally. When the filter coefficients are not yet converged, a small $\rho(n)$ is required to provide more proportionality and therefore allow the most significant taps to converge fast initially to improve the overall initial convergence rate. As the filter coefficients converge overtime to their actual values, $\rho(n)$ should increase to allow more taps of non-sparse impulse responses to converge.

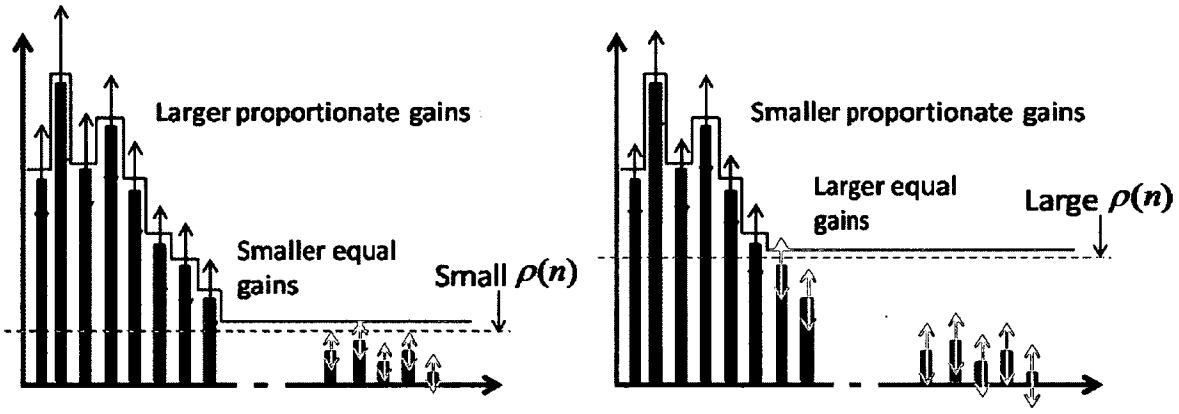


Figure 5.11: Schematic showing the effect of $\rho(n)$ on the distribution of adaptation gain in PNLMS algorithms.

Figure 5.12 shows the relationship between $\rho(n)$ and $\lambda(n)$ for a non-sparse impulse response with a sparseness value of $\xi(n) = 0.4$. It shows that $\lambda(n)$ should decrease as the filter converges overtime so that $\rho(n)$ will be increasing.

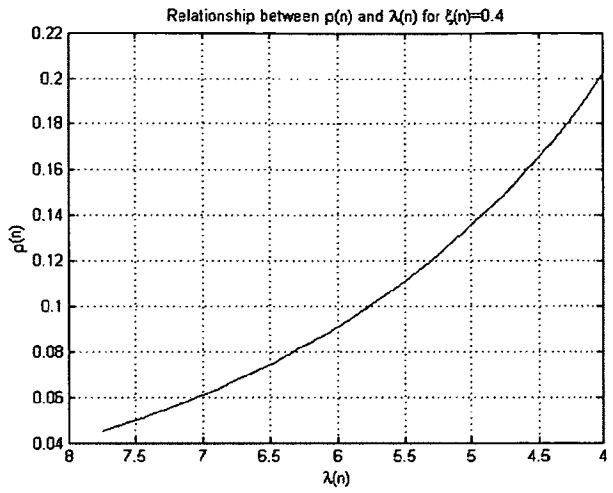


Figure 5.12: The relationship between $\rho(n)$ and $\lambda(n)$ for a non-sparse impulse response with $\xi(n) = 0.4$

Since the MSE decreases as the filter converges, then the parameter λ is varied using an estimate of the MSE as in the following equations:

$$\lambda(n) = 4 * exp(\frac{-1}{\varsigma\, \hat{\sigma}_e^2(n)}) + 4 \tag{5.7}$$

where $\hat{\sigma}_e^2(n)$ is the current estimate of the MSE and is computed as in equation (4.15), and ς is a large positive scaling factor.

This relation is plotted in Figure 5.13, where λ decreases as the current estimate of the MSE decreases over time.

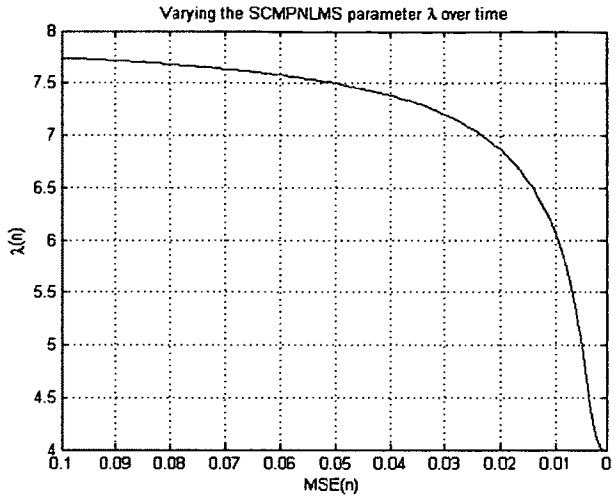


Figure 5.13: The profile of the change in λ over time.

As the MSE decreases over time, and $\lambda(n)$ decreases as well, $\rho(n)$ increases to provide more adaptation gain for a larger number of small-valued coefficients to converge after the fewer number of large-valued coefficients have converged in the initial phase.

The variation of λ of the SCMPNLMS is combined with the variation of the mu-law parameter, β, as in the AMPNLMS [**19**] (see equation (4.17)) to give the Adaptive Sparseness Controlled Mu-law PNLMS (ASCMPNLMS) algorithm. The ASCMPNLMS algorithm is summarized in Table B.1.

5.2.2 Preliminary Results

The parameters in the ASCMPNLMS are set to $\gamma = 0.001$, $\varepsilon = 0.99$, $\nu = 1000$ [**19**], $\delta_{ASCMPNLMS} = \delta_{NLMS}$, and choosing $\varsigma = 150$ provided the required scaling of the MSE.

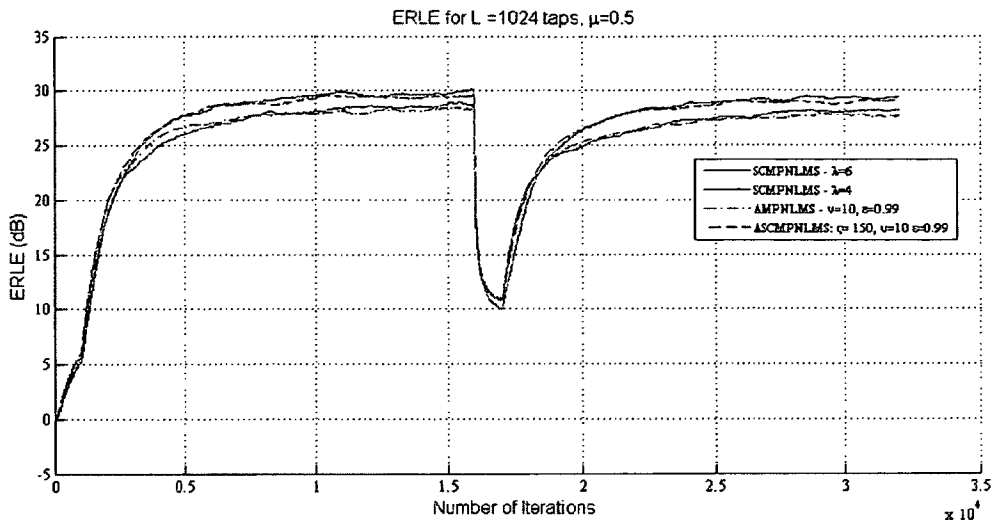


Figure 5.14: Comparative simulation of the SCMPNLMS, AMPNLMS, and ASCMPNLMS using the measured impulse responses in Figure 3.8.

Figure 5.14 show the simulation results for the SCMPNLMS, AMPNLMS, and ASCMPNLMS using the measured impulse responses in Figure 3.8. The overall performance of the ASCMPNLMS algorithm is comparable to the SCMPNLMS when $\lambda = 4$ for the measured impulse responses, with only a slight improvement in tracking capability. The steady-state of the ASCMPNLMS is higher than the AMPNLMS because the small-valued coefficients have converged more adequately.

5.3 Simulation Results using Measured Impulse Responses

5.3.1 Parameter Selection for g-PNLMS

The only parameter in the gradient-induced algorithms that need to be tuned is r that controls the contribution from the proportional-gain and gradient-gain to the gain-control vector. Figure 5.15 and Figure 5.16 show simulation results for the PNLMS, and for the

g-PNLMS with different values for r. As r increases to 0.6, the convergence speed, and tracking performance as well as the steady-state improve. Any further increase has no significant improvement and the tracking starts to slow down slightly with $r = 0.8$. The best overall performance is provided using $r = 0.6$.

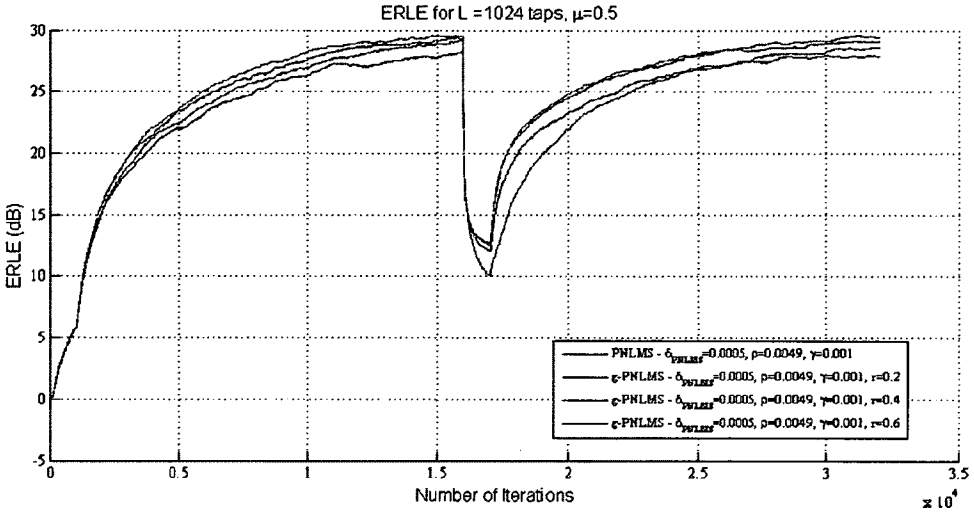


Figure 5.15: Comparative simulations for the PNLMS, and the g-PNLMS using small values for r.

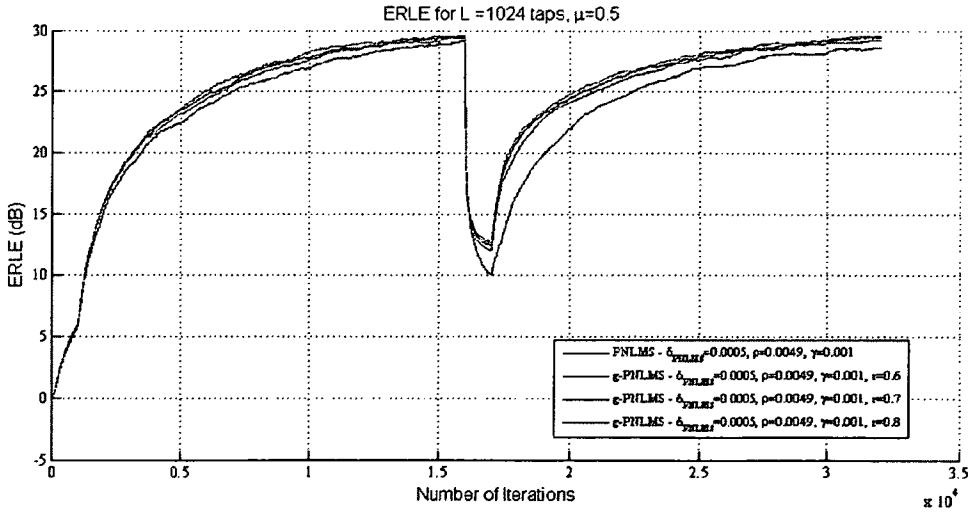


Figure 5.16: Comparative simulations for the PNLMS, and the g-PNLMS using large values for r.

5.3.2 Results

5.3.2.1 Comparison of g-PNLMS with GGPNLMS

Figure 5.17 shows a comparison between the g-PNLMS and the GGPNLMS. The results show that the gradient-induced PNLMS outperforms the GGPNLMS in terms of tracking capability; however the g-PNLMS slows down as the filter converges and the GGPNLMS achieve a higher steady-state level.

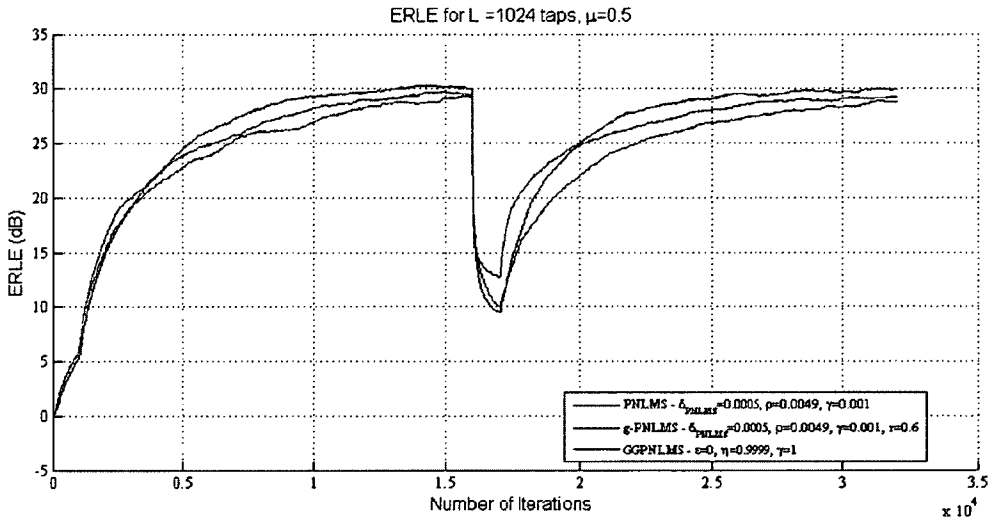


Figure 5.17: Comparative simulation of the PNLMS, g-PNLMS, and GGPNLMS.

5.3.2.2 Comparison of PNLMS-Based Algorithms with their Gradient-Induced Counterparts

The following comparative simulations show the improvement of using the gradient induced technique (see section 5.1) with various VISS algorithms. The g-PNLMS shown in Figure 5.18 has better convergence speed and a significant improvement in the

tracking capability. The steady-state level is also higher than that of the PNLMS by few decibels.

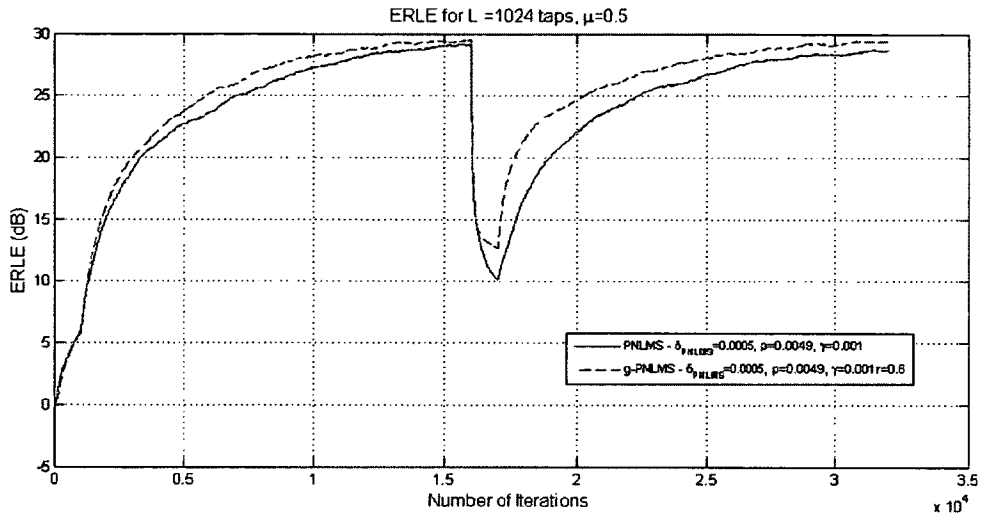


Figure 5.18: Comparative simulation of the PNLMS, and g-PNLMS.

The g-IPNLMS also shows an improvement in the tracking performance when compared to the IPNLMS as shown in Figure 5.19.

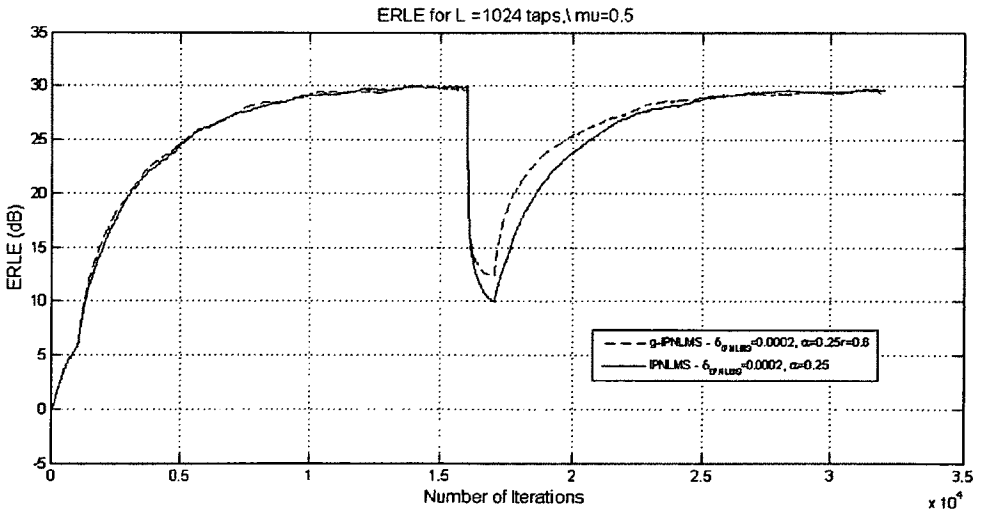


Figure 5.19: Comparative simulation of the IPNLMS, and g-IPNLMS.

The g-MPNLMS shows only a slight improvement in the tracking capability as shown in Figure 5.20, however it provides better steady-state level than the MPNLMS.

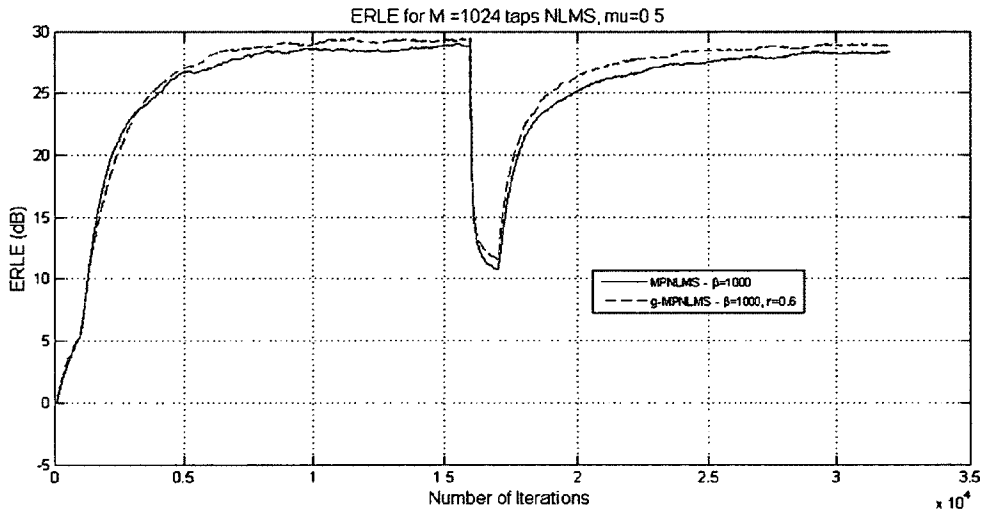


Figure 5.20: Comparative simulation of the MPNLMS, and g-MPNLMS.

The g-AMPNLMS provides a similar improvement as the g-MPNLMS provides when compared to their original counterparts, as shown in Figure 5.21.

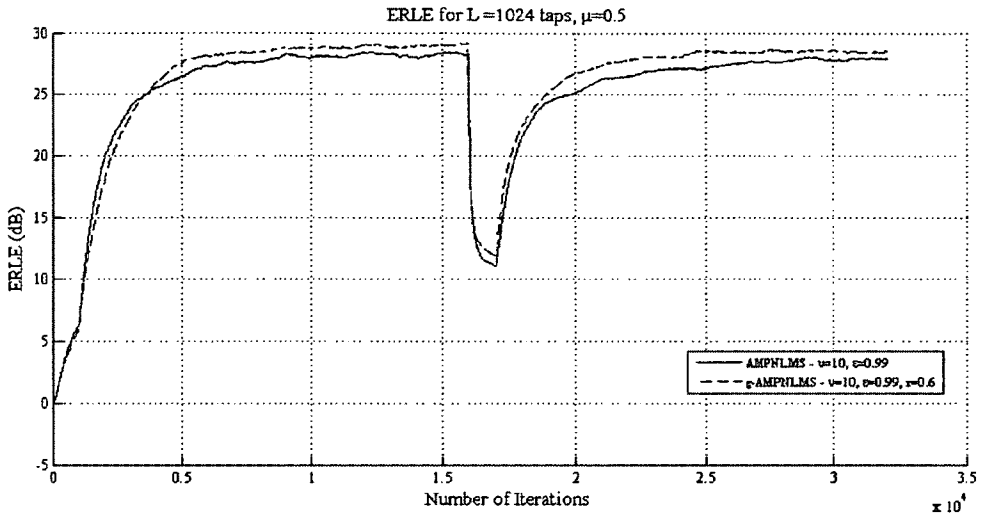


Figure 5.21: Comparative simulation of the AMPNLMS, and g-AMPNLMS.

In Figure 5.22 the g-SPNLMS shows a relatively significant improvement in the overall performance and specifically in terms of the tracking speed.

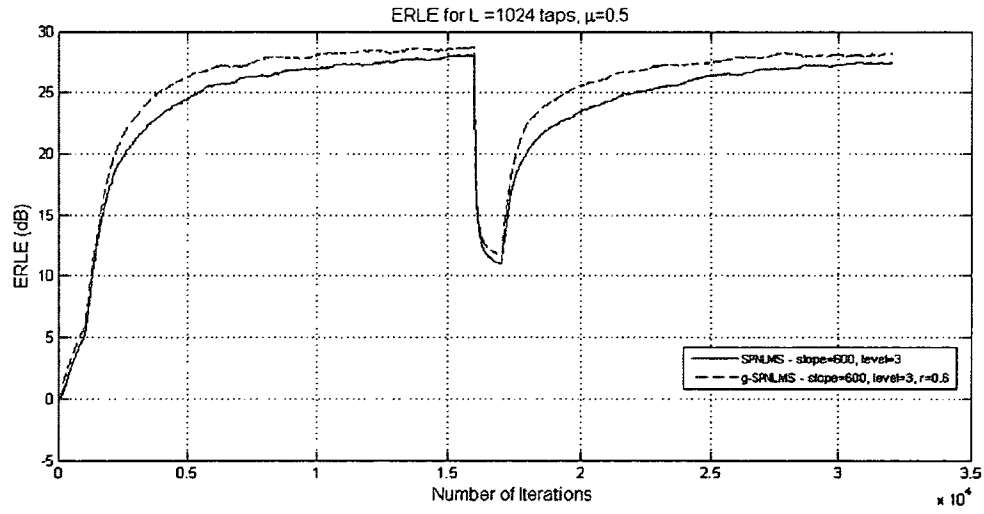


Figure 5.22: Comparative simulation of the SPNLMS, and g-SPNLMS.

The gradient-induced technique improved the tracking performance of all sparseness controlled algorithms: SCPNLMS, SCIPNLMS, and SCMPNLMS as shown in the following three figures.

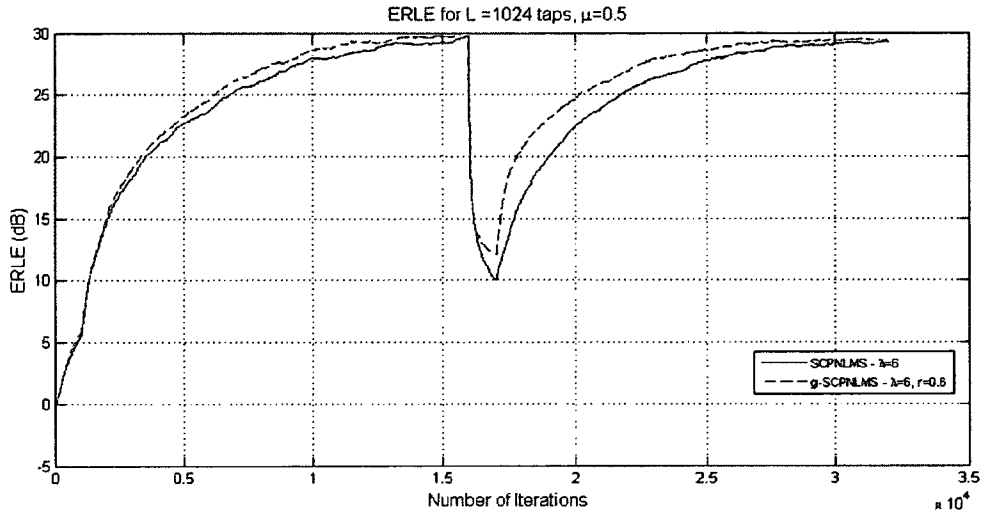


Figure 5.23: Comparative simulation of the SCPNLMS, and g-SCPNLMS.

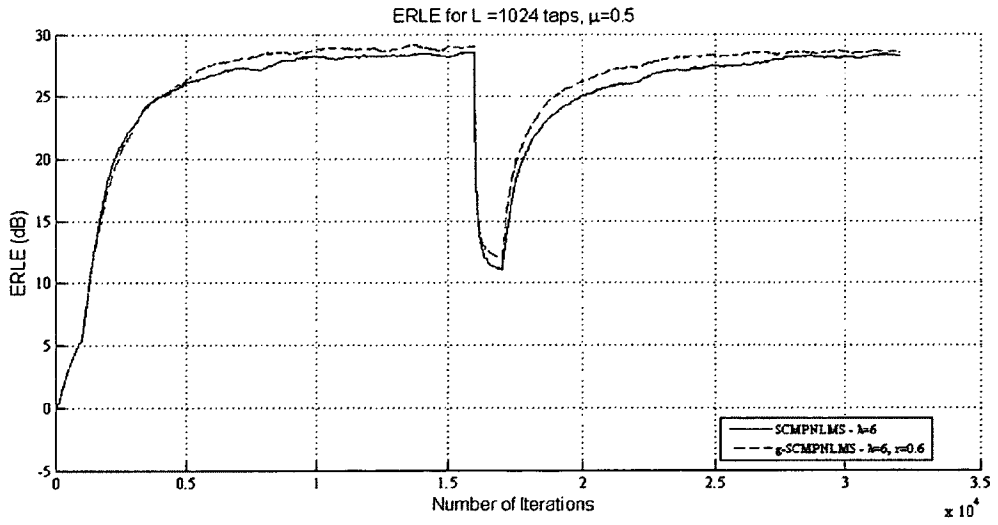


Figure 5.24: Comparative simulation of the SCMPNLMS, and g-SCMPNLMS with $\lambda = 6$.

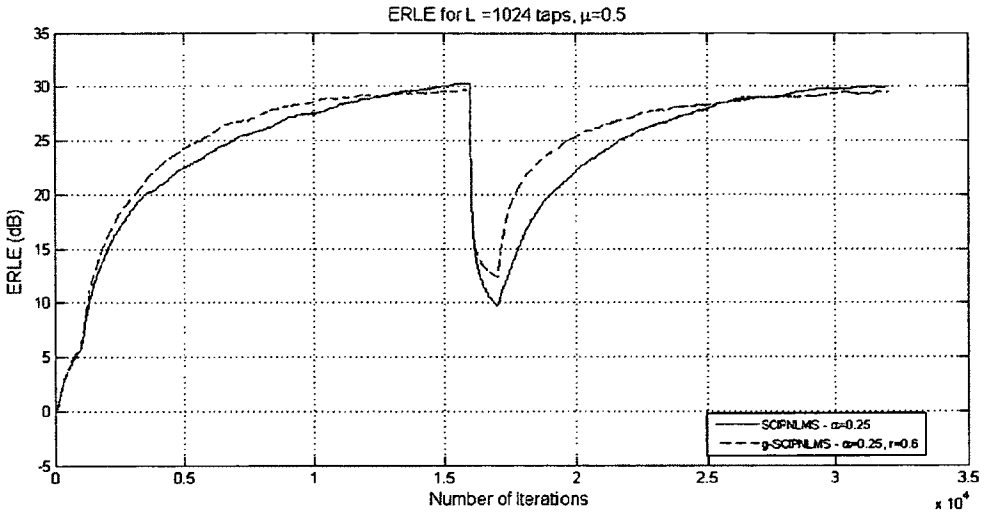


Figure 5.25: Comparative simulation of the SCIPNLMS, and g-SCIPNLMS.

The g-SCMPNLMS has the same performance as the SCMPNLMS when $\lambda = 4$ as shown in Figure 5.26.

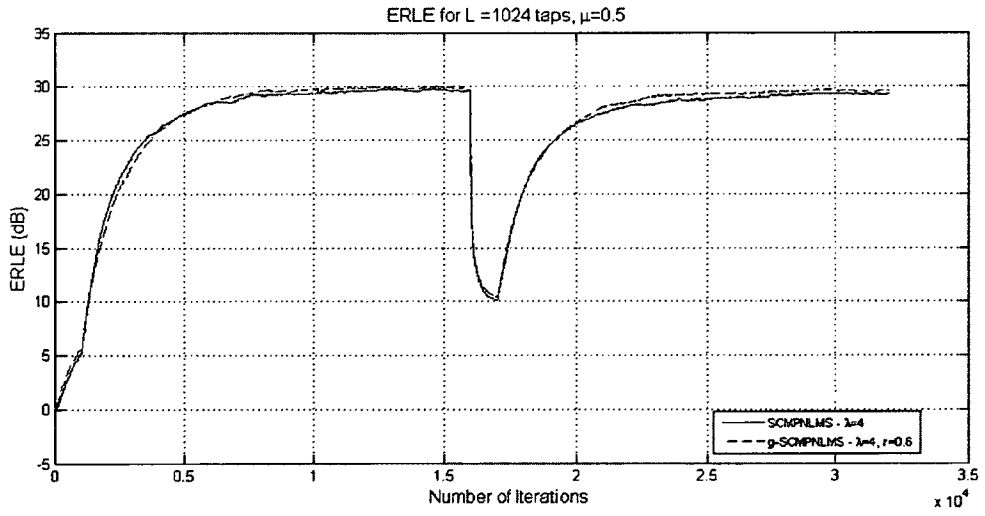


Figure 5.26: Comparative simulation of the SCMPNLMS, and g-SCMPNLMS with $\lambda = 4$.

Also only a small improvement in tracking capability is evident for the g-ASCMPNLMS as shown in Figure 5.27.

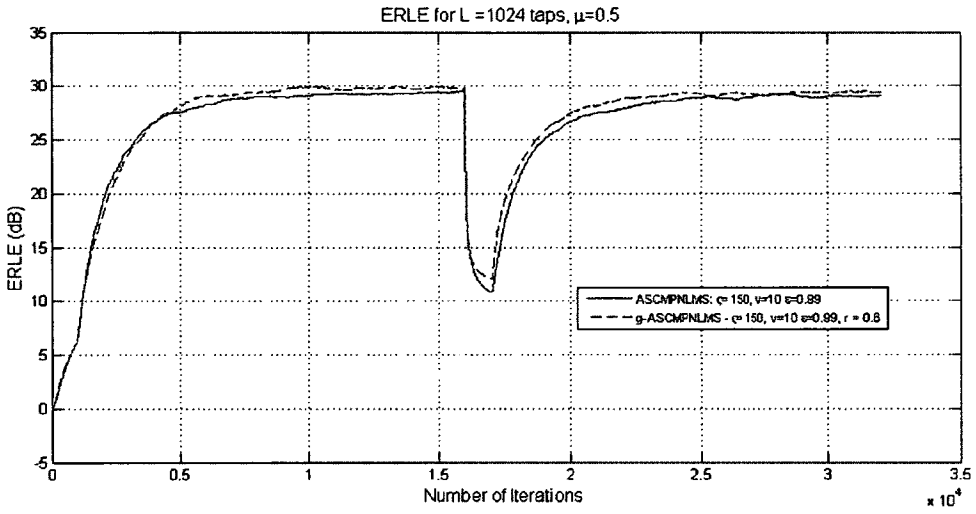


Figure 5.27: Comparative simulation of the ASCMPNLMS, and g-ASCMPNLMS.

5.3.2.3 Comparison of Gradient-Induced PNLMS-Based Algorithms

Figure 5.28 compares the g-MPNLMS, g-SCIPNLMS, and g-SCMPNLMS showing that the g-SCMPNLMS has the best steady-state performance and same convergence speed as the g-MPNLMS, and the g-SCIPNLMS has the best tracking capability. The g-MPNLMS has the best overall performance but a slightly lower steady-state performance.

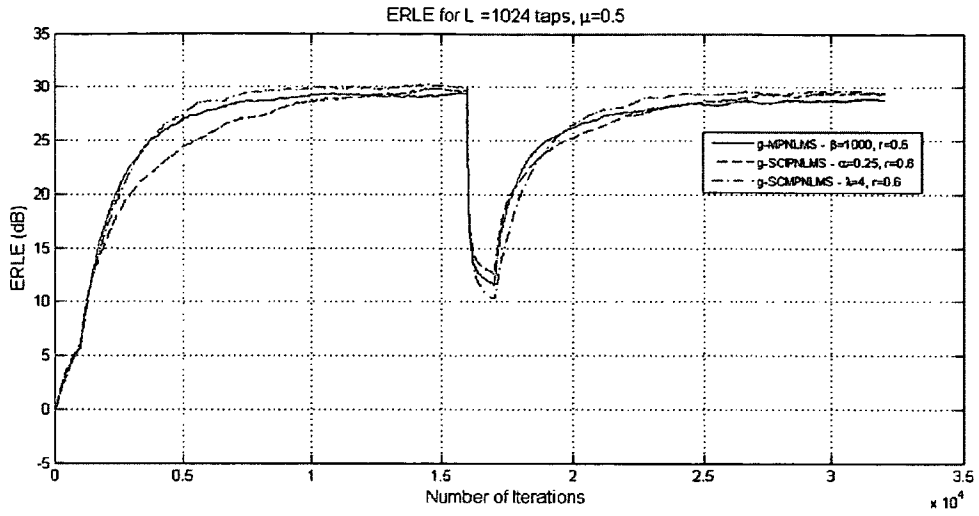


Figure 5.28: Comparative simulation of the g-MPNLMS, g-SCIPNLMS, and g-SCMPNLMS.

The g-ASCMPNLMS is compared to the g-MPNLMS, g-SCIPNLMS, and g-SCMPNLMS in Figure 5.29. The results show that the g-ASCMPNLMS has the best overall tracking performance; however, it is the most computationally expensive algorithm.

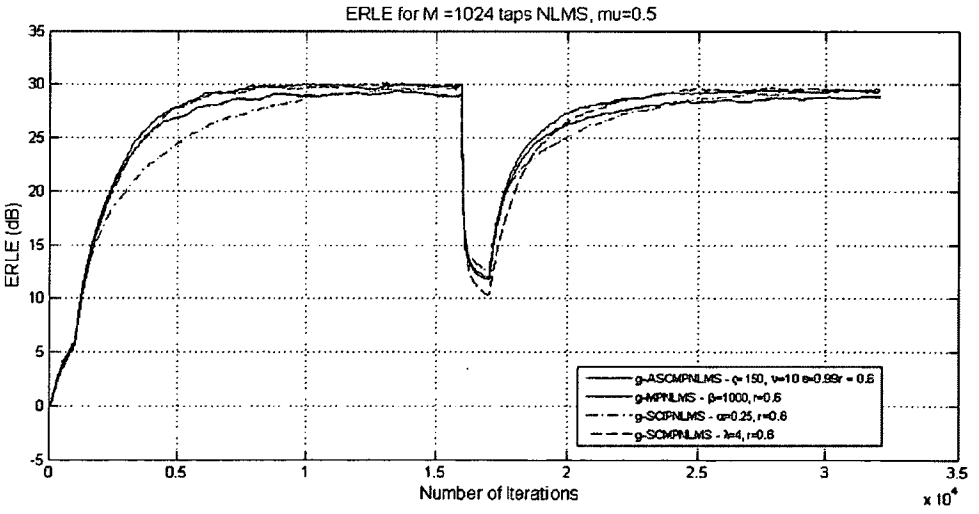


Figure 5.29: Comparative simulation of the g-ASCMPNLMS, g-MPNLMS, g-SCIPNLMS, and g-SCMPNLMS.

5.3.2.4 Comparison of g-SPNLMS with MPNLMS and AMPNLMS

Figure 5.30 shows that the g-SPNLMS has a comparable performance as the MPNLMS with even better tracking capability. The MPNLMS algorithm may require more computations because of the logarithmic function whereas the g-SPNLMS does not. The g-SPNLMS requires $3L + 1$ additions, $8L + 3$ multiplications, 4 divisions, and $4L$ comparisons. The MPNLMS requires $3L + 1$ additions, $6L + 2$ multiplications, 2 divisions, $2L$ comparisons, and L logarithms. The implementation of the g-SPNLMS may be cheaper than that of the MPNLMS providing a more efficient algorithm since it provides the same performance as the MPNLMS.

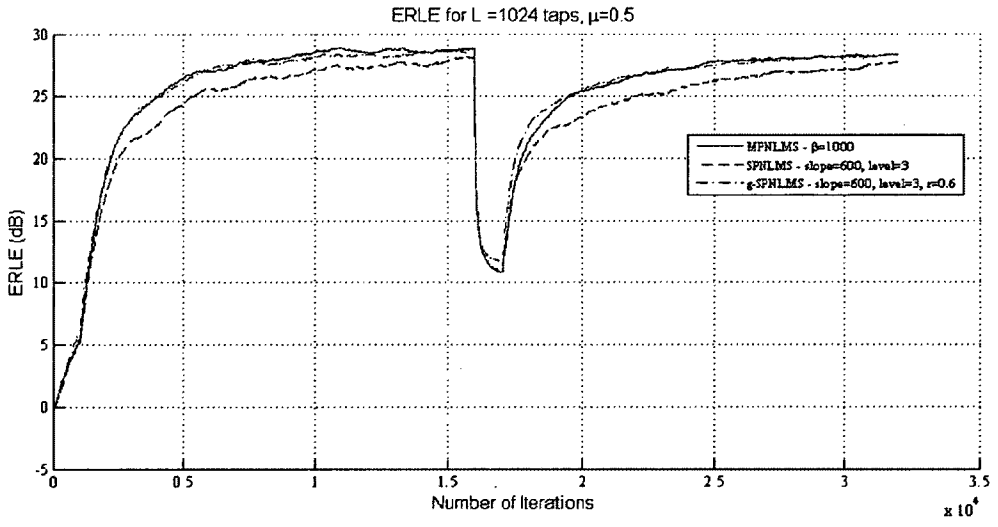


Figure 5.30: Comparative simulation of the MPNLMS, SPNLMS, and g-SPNLMS.

A similar deduction can be drawn for the AMPNLMS as shown in Figure 5.31.

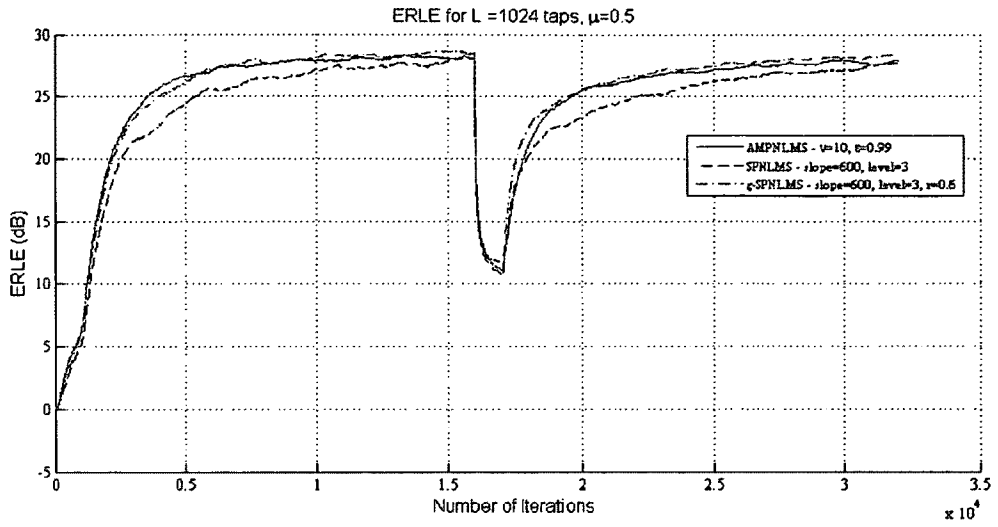


Figure 5.31: Comparative simulation of the AMPNLMS, SPNLMS, and g-SPNLMS.

5.4 Conclusion

In this chapter, we introduced the gradient-induced PNLMS (g-PNLMS) algorithm that was designed to provide fast tracking performance in highly non-stationary acoustic environments. The g-PNLMS algorithm outperformed both the PNLMS and the GGPNLMS using simulated and measured impulse responses. The gradient-induced technique was incorporated into other PNLMS-based algorithms and proved to improve the tracking performances of most algorithms. The g-SPNLMS algorithm provided better overall performance when compared to the MPNLMS and AMPNLMS, and it does not require the expensive logarithmic computations needed in the MPNLMS and the AMPNLMS. The Adaptive Sparseness Controlled Mu-law PNLMS (ASCMPNLMS) was presented and compared with the SCMPNLMS and the AMPNLMS algorithms. The g-ASCMPNLMS provided the best overall performance but at increased computational complexity.

CHAPTER 6 Proposed Thresholded-NLMS Adaptive Algorithms

This chapter proposes a new method for improving the performance of AECs by the use of a thresholding technique with the NLMS algorithm. The idea could be extended to other NLMS-based algorithms. Two algorithms are proposed; the basic Thresholded-NLMS (TNLMS) and the masked Thresholded-NLMS (mTNLMS). Two impulse responses are simulated for the purpose of comparing the ERLE performance of the proposed algorithms to the NLMS algorithm.

6.1 The NLMS Algorithm in Acoustic Echo Cancellation

Since acoustic EPIRs are very long (several hundred milliseconds), the echoes due to reflections from distant objects are relatively much smaller than echoes due to initial reflections. However, due to the large number of small echoes from distant objects, these echoes contribute significantly to the deterioration of the speech conversation. Therefore proper tuning of small valued coefficients of the adaptive filter is required. The NLMS algorithm distributes the available adaptation gain equally among all the filter coefficients. This is depicted by the equal lengths of the green arrows shown in Figure 6.1. The schematic drawing also emphasizes the tuning of a small filter tap, which has an actual magnitude that is relatively smaller than the equally distributed adaptation gain of the NLMS algorithm.

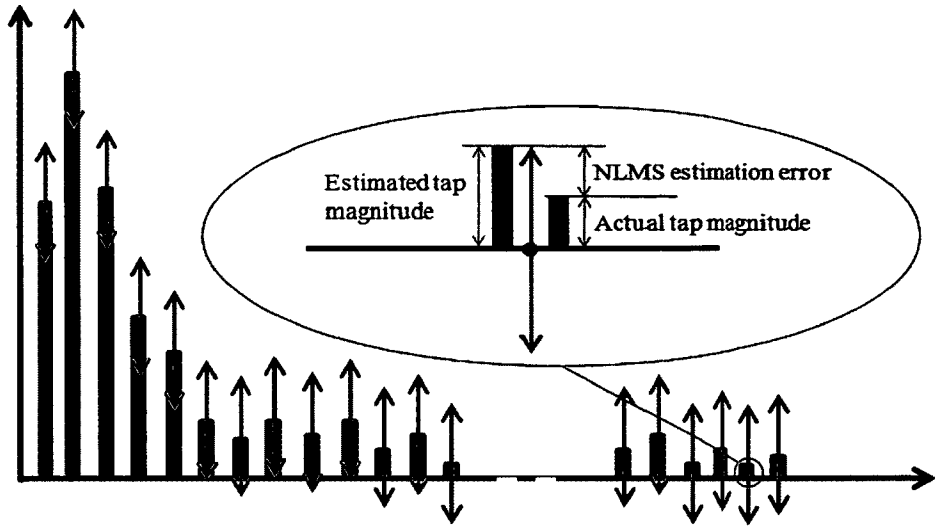


Figure 6.1: Impulse response schematic showing equal NLMS gain distribution and the NLMS estimation error due to tuning of small coefficients.

The estimated tap value, as a result, will fluctuate around its actual magnitude and introduce an NLMS estimation error as shown in Figure 6.1. Therefore, it is important to draw some attention to the estimation errors resulting from the adaptation of the small valued coefficients of the adaptive filter. This is elaborated in more details in the following analysis.

The individual tap weight-update equation for the NLMS algorithm can be written as follows:

$$\hat{c}_l(n+1) = \hat{c}_l(n) + \frac{\mu\, e(n)x(n-l)}{\sum_{l=0}^{L-1} x^2(n-l) + \delta_{NLMS}} \tag{6.1}$$

$$l = 0, 1, \ldots L-1$$

The weight-update per tap is:

$$weight\ update\ per\ tap = \frac{\mu\, e(n)x(n-l)}{\sum_{l=0}^{L-1} x^2(n-l) + \delta_{NLMS}} \tag{6.2}$$

where l is the filter tap location. The weight-update per tap is mainly governed by an equal gain distribution that is multiplied by the individual input signal sample at each tap to give a different weight-update for each tap,

$$weight\ update\ per\ tap = equal\ gain\ .\ x(n-l) \tag{6.3}$$

$$= \frac{\mu\, e(n)}{\sum_{l=0}^{L-1} x^2(n-l) + \delta_{NLMS}} x(n-l)$$

This is shown by different lengths of orange arrows in Figure 6.2.

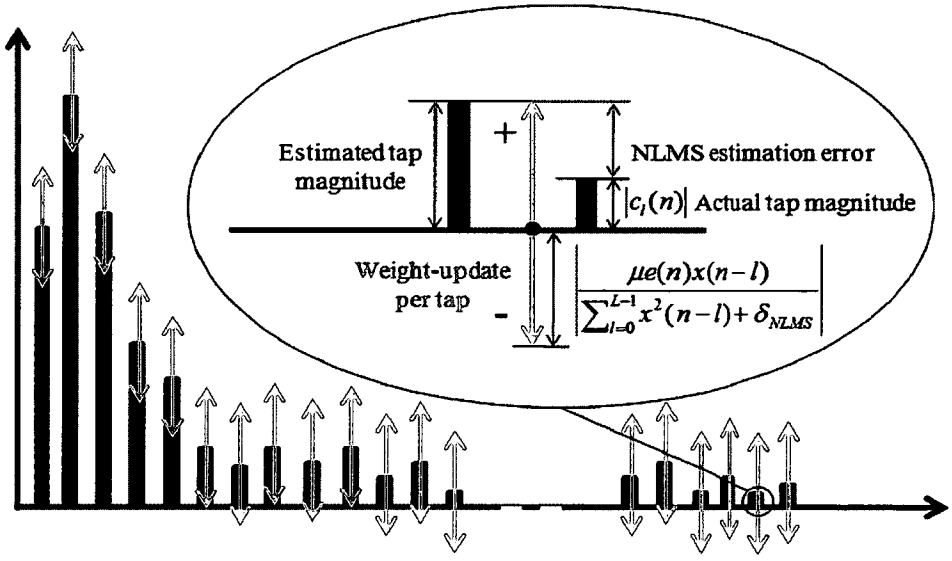


Figure 6.2: Impulse response schematic emphasizing the weight-update of a small tap and the resulting NLMS estimation error.

The issue of tuning small-valued coefficients can be stated more accurately as follows: if the actual magnitude of the tap, $|c_l(n)|$, is relatively smaller than the weight-update per tap as shown in Figure 6.2, an NLMS estimation error results. When there is a large number of taps that have actual magnitudes relatively smaller than their weight-update, the cumulative estimation error degrades the convergence speed and tracking capability, and increases the steady-state error.

Tuning of coefficients with small magnitudes slows down the convergence speed and tracking because the available adaptation gain is not used efficiently. Moreover, the cumulative estimation error increases the steady-state error because the relatively small coefficients have estimation errors and therefore the echo signals are not properly cancelled. Therefore a new technique is required to redistribute the available adaptation gain to coefficients with more significant magnitudes in a more efficient way and at the same time reduce the error resulting from estimating the small-valued coefficients.

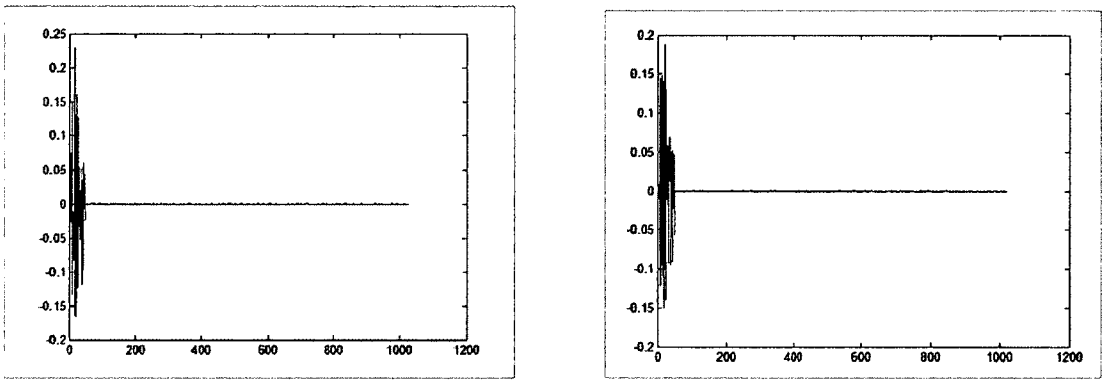
6.2 A Controlled Acoustic Echo Path Impulse Response Scenario

A controlled simulation of EPIRs was carried out to mimic a situation where there are a significant number of coefficients that have small magnitudes such that they contribute significantly to the echo signal. This situation may occur in real life due to attenuated reflections coming from distant objects.

The first 50 taps of the measured EPIRs shown in Figure 3.8 were used to simulate the initial reflections in the simulated EPIRs, and the remaining taps were divided into three sets: 1) taps of magnitudes equal to or smaller than a threshold value 2) taps of

magnitudes larger than the threshold, and 3) taps of magnitude zero. The locations of the taps in each set were randomly assigned between $l = 50$ and $l = 1024$. The threshold was calculated using the formula: $threshold = fraction * \frac{\mu}{L} = 0.4 * \frac{0.5}{1024} \approx 2 \times 10^{-4}$. This value is comparable to a fraction of the update of each tap in the NLMS algorithm that uses a normalized step-size of 0.5.

Figure 6.3.a and Figure 6.3.b show two artificial EPIRs created using the measured impulse responses in Figure 3.8.a and Figure 3.8.b respectively.



a) Modified impulse response with the smartphone lying on its back.

b) Modified impulse response with the smartphone lying on its face.

Figure 6.3: Simulated EPIRs using the measured impulse responses in Figure 3.8.

6.3 Thresholded-NLMS Algorithms

This section presents the incorporation of a thresholding technique in the NLMS algorithm to improve the overall performance of adaptive filters for AEC. Two algorithms are proposed: the basic Thresholded-NLMS (TNLMS) and the masked Thresholded-NLMS (mTNLMS).

The Thresholded-NLMS algorithms have two goals: 1) reduce the estimation error resulting from tuning of the small-valued coefficients and therefore improve the steady-state performance, 2) use the available adaptation gain more efficiently to increase the convergence and tracking speeds by adapting taps with more significant magnitudes faster.

6.3.1 Basic Thresholded-NLMS Algorithm (TNLMS)

The basic idea depends on dividing the filter coefficients into two groups: *active* taps and *inactive* taps based on comparing the coefficients' current estimated magnitudes to a threshold value. If the current estimate of a coefficient is smaller than the threshold value, it is set to zero, therefore becomes an inactive tap and its adaptation is omitted. The NLMS estimation error is therefore replaced by an omission error which depends solely on the actual magnitude of the inactive tap, as shown in Figure 6.4.

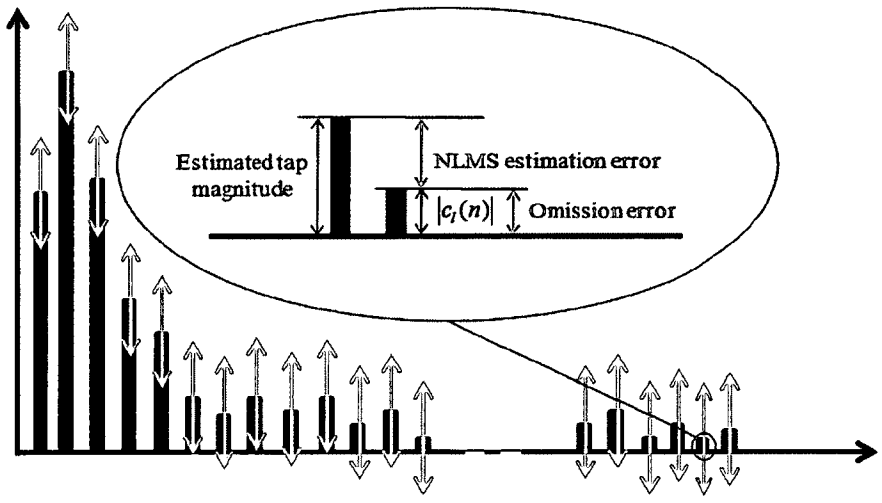


Figure 6.4: Schematic of an impulse response showing the NLMS estimation error and the omission error resulting from thresholding relatively small coefficients.

Figure 6.5 and Figure 6.6 show block diagrams for the NLMS and TNLMS algorithms respectively to show the modifications of incorporating the thresholding technique.

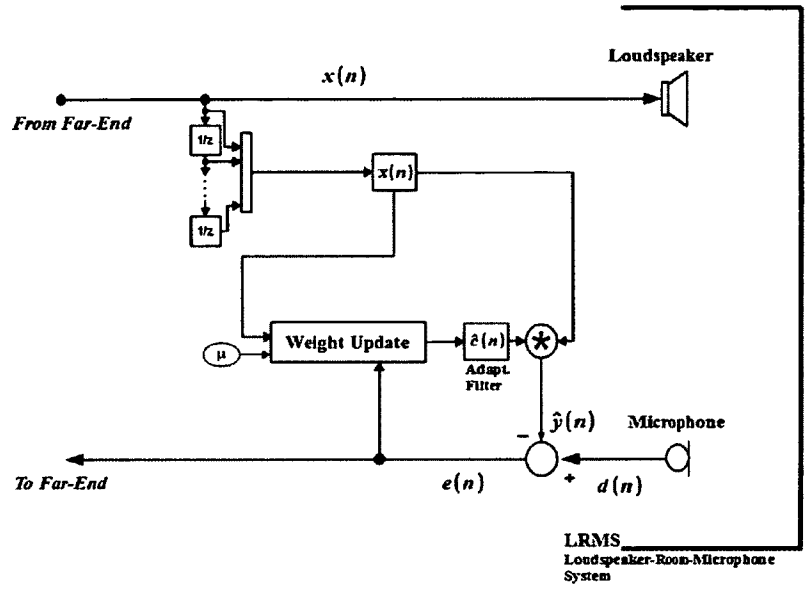


Figure 6.5: Block diagram for the NLMS algorithm.

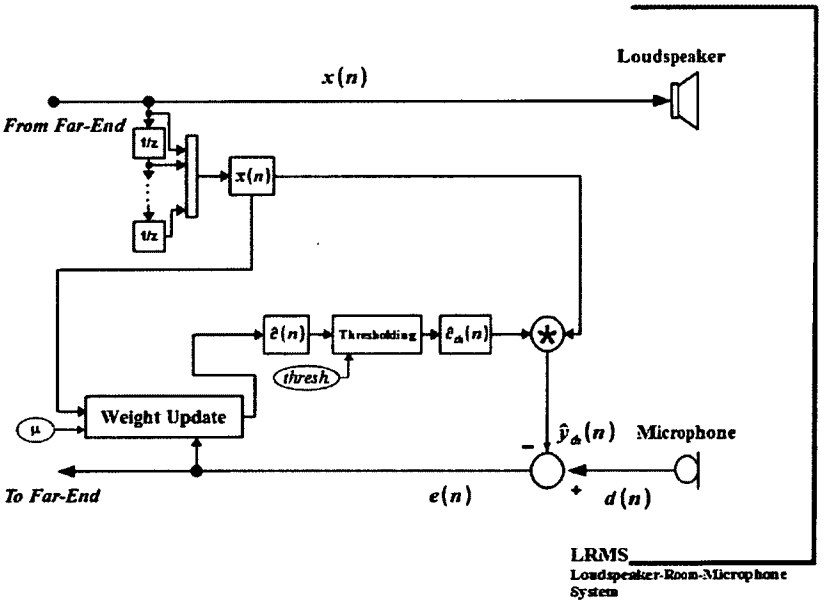


Figure 6.6: Block diagram for the TNLMS algorithm.

The threshold value is deduced from the weight-update per tap. If we assume $x(n)$ to be WGN with zero mean and variance σ_x^2 then we can approximate the summation in the denominator of equation (6.2) by

$$\sum_{l=0}^{L-1} x^2(n-l) \approx L\sigma_x^2 \tag{6.4}$$

Therefore the weight-update per tap becomes approximately governed by $\frac{\mu}{L}$, giving rise to the following threshold formula:

$$threshold = error\ ratio * \frac{\mu}{L} \tag{6.5}$$

where the error ratio is a fraction, where error ratio < 0.5, to guarantee a smaller cumulative omission error than a cumulative NLMS estimation error, thus providing an improvement in echo cancellation performance.

Thresholding, depicted in Figure 6.7, is carried out every iteration by comparing the current estimate of the coefficients to the threshold value,

$$c_l(n) = \begin{cases} c_l(n) & |c_l(n)| > threshold \\ 0 & otherwise \end{cases} \quad \forall l, \tag{6.6}$$

$$l = 0, 1, \ldots L-1$$

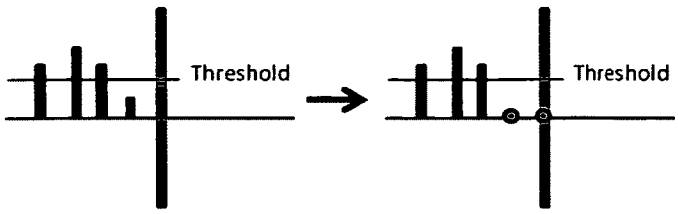


Figure 6.7: Schematic of an impulse response showing thresholding applied to taps with magnitudes smaller or equal to the threshold.

Thresholding takes place before the output of the adaptive filter is computed, therefore inactive taps do not contribute to the output signal, $\hat{y}_{th.}(n)$, of the thresholded adaptive filter, $\hat{c}_{th.}(n)$, as shown in the block diagram of the TNLMS in Figure 6.6 where,

$$\hat{y}_{th.}(n) = \hat{c}^T{}_{th.}(n)\bar{x}(n) \tag{6.7}$$

This results in a more accurate error signal which in return improves the weight-update. The TNLMS algorithm is listed in Table B.2.

6.3.2 Masked Thresholded-NLMS (mTNLMS)

Since the output of the adaptive filter is a function of the active taps only, the input signal samples at the inactive tap locations have no effect on the output of the adaptive filter. The masked-TNLMS algorithm proposes an approach to increase the adaptation step-size by attenuating the contribution of the input signal samples at the locations of the inactive taps. This in turn decreases the power of the adaptive filter input signal that is used to normalize the NLMS step-size.

A weighting mask vector, $\bar{m}(n)$, is deduced from the inactive tap locations using the thresholding of the filter coefficients,

$$m_l(n) = \begin{Bmatrix} 1 & |c_l(n)| > threshold \\ \tau & otherwise \end{Bmatrix} \quad \forall l, \tag{6.8}$$

$$l = 0, 1, \ldots L-1$$

where $0 < \tau < 1$ is the masking factor, and used to attenuate the contribution of the input signal at the inactive tap locations by applying it to the filter input vector using element-by-element array multiplication (Hadamard product),

$$masked\ input\ = \bar{\acute{x}}(n)$$

$$\bar{\acute{x}}(n) = \bar{m}(n) \circ \bar{x}(n) \tag{6.9}$$

$$= [m_0(n) \quad m_1(n) \ldots \quad m_{L-1}(n)]^T$$

$$\circ\, [x(n) \quad x(n-1) \ldots \quad x(n-L+1)]^T$$

$$= [m_0(n)x(n) \quad m_1(n)x(n-1) \ldots \quad m_{L-1}(n)x(n-L+1)]^T \tag{6.10}$$

The mTNLMS step-size becomes

$$\mu_{mTNLMS}(n) = \frac{\mu_{NLMS}}{\bar{\acute{x}}^T(n)\bar{\acute{x}}(n) + \delta_{NLMS}} \tag{6.11}$$

which is effectively a larger step-size because μ_{NLMS} is divided by a smaller power of the masked input vector,

$$\frac{\mu_{NLMS}}{\sum_{l=0}^{L-1} \acute{x}^2(n-l) + \delta_{NLMS}} \geq \frac{\mu_{NLMS}}{\sum_{l=0}^{L-1} x^2(n-l) + \delta_{NLMS}} \tag{6.12}$$

This increases the convergence rate and tracking performance by giving larger adaptation gains to active taps. Another advantage is that the mTNLMS is sensitive to the changes in the echo path impulse response thus improving the tracking performance in non-stationary environments.

Figure 6.8 shows a block diagram for the mTNLMS algorithm which is also listed in Table B.3.

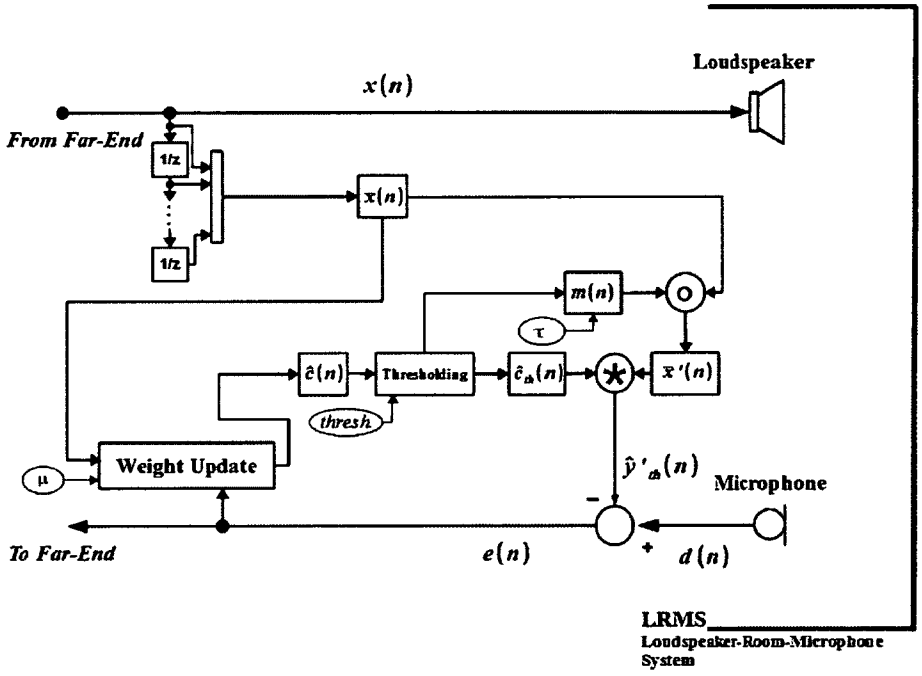


Figure 6.8: Block diagram for the mTNLMS algorithm.

6.4 Simulation Results

6.4.1 Parameter Selection

There are two parameters selected in the Thresholded-NLMS algorithms; the first is the error ratio which is used in both TNLMS (see Table B.2) and mTNLMS (see Table B.3), and the second parameter is the masking factor which is used in the mTNLMS only.

Figure 6.9 shows the simulation results for the TNLMS algorithm for different values of the error ratio. The results show that the convergence speed and the tracking capability of the TNLMS are slightly affected by the choice of the error ratio. The steady-state, on the other hand, improves as the error ratio increases up to 0.4 (at around 34 dB) after which it degrades. When the error ratio is set to 0.5, the steady-state level is approximately 32 dB. This is because of two reasons, first, the thresholded taps that have magnitudes equal to half their weight-updates have their estimation errors equal to their omission errors, which means there is no reduction in the overall error by thresholding them. The second reason is that more taps that carry significantly important information about the echo path are being sacrificed by being thresholded, which leads to an even worse steady-state level when compared to the results with an error ratio of 0.4. The performance degrades more with larger values for the error ratio, as shown in Figure 6.10, where the steady-state becomes comparable to the NLMS with the error ratio set to 0.6.

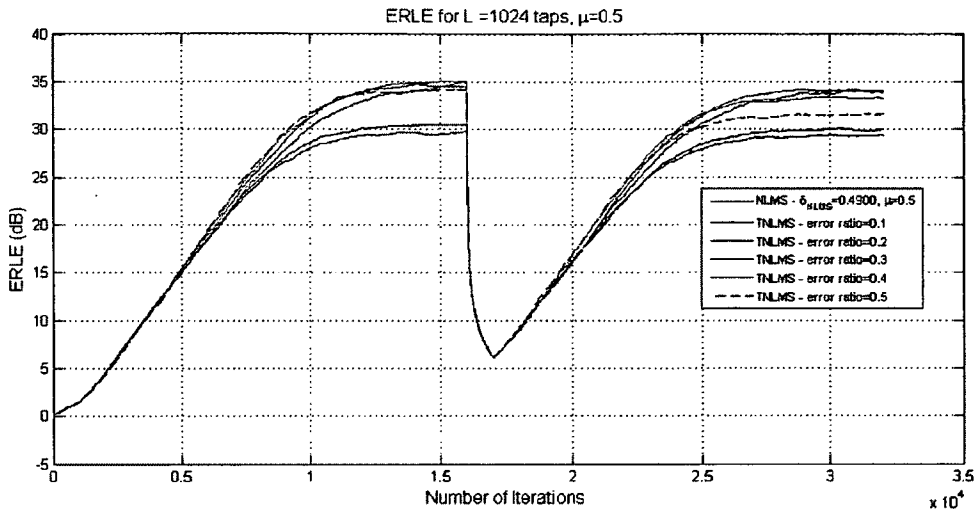


Figure 6.9: Comparative simulation of TNLMS for small values of the error ratio.

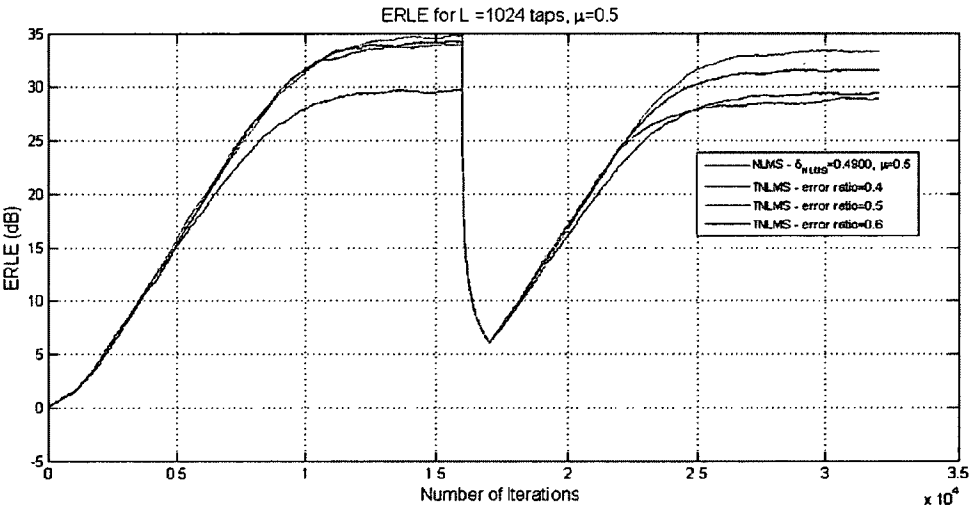


Figure 6.10: Comparative simulation of TNLMS for large values of the error ratio.

Simulations for the mTNLMS algorithm were carried out for different values of the error ratio as well with the masking factor kept constant at $\tau = 0.5$. Figure 6.11 shows the simulation results for the mTNLMS algorithm for values of the error ratio from 0.1 to 0.6. The plots show that the overall performance improves significantly as the error ratio increases from 0.1 to 0.6. This is because as the error ratio increases there are fewer

active taps to converge, and also with the masking of more inactive taps, the mTNLMS step-size, $\mu_{mTNLMS}(n)$ given by equation (6.11)), increases as well leading to an overall increase in the convergence speed and tracking capability.

Figure 6.12 shows results for larger values of the error ratio, from 0.4 to 0.7, where it shows that the improvement in the convergence speed and tracking capability are not significant, and on the other hand, the steady-state level also degrades only slightly. This is because the further increase in μ_{mTNLMS} causes some taps to have larger omission errors than estimation errors. A good choice for the error ratio in the mTNLMS algorithm is 0.4 or 0.5 as can be deduced from Figure 6.12.

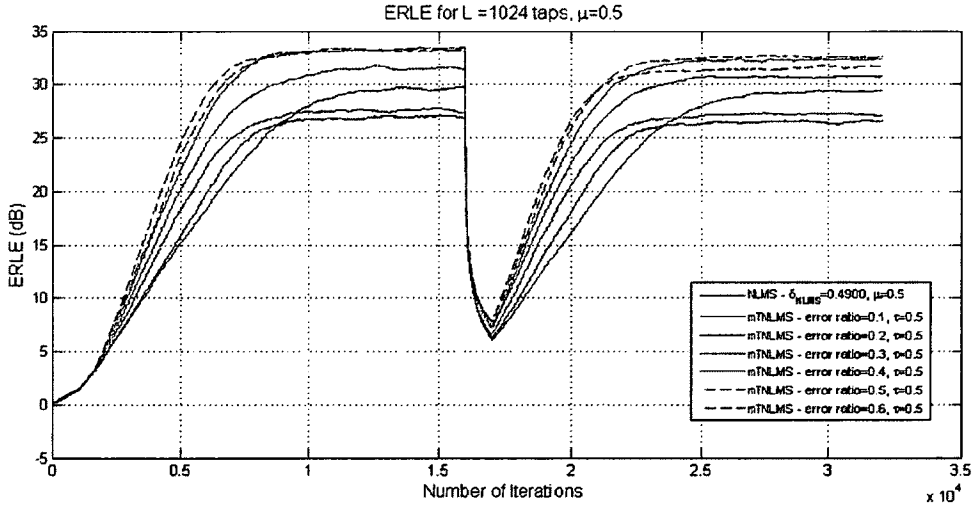


Figure 6.11: Comparative simulation of mTNLMS for small values of the error ratio with constant masking factor.

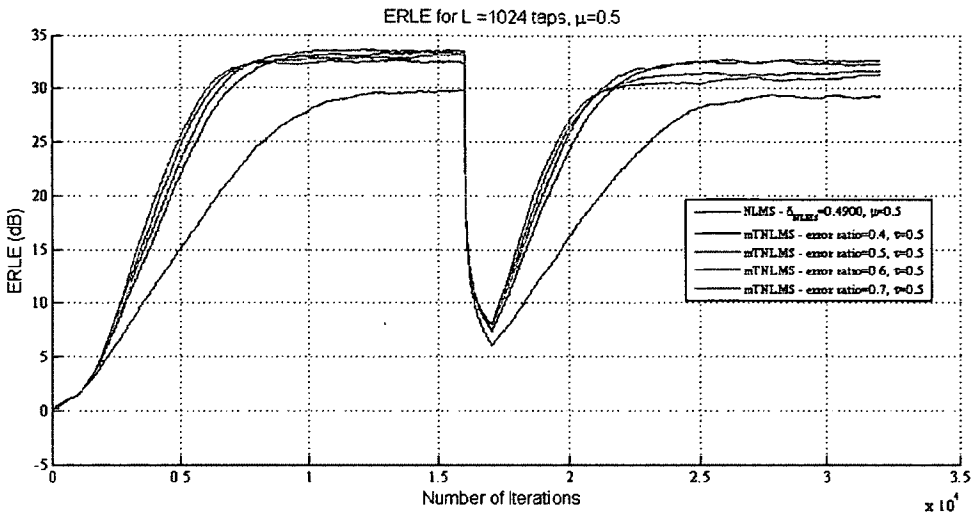


Figure 6.12: Comparative simulation of mTNLMS for large values of the error ratio with constant masking factor.

Figure 6.13 compares the ERLE steady-state performance of TNLMS and mTNLMS using the impulse response in Figure 6.3.b for different values of the error ratio and a masking factor of $\tau = 0.5$ for the mTNLMS. The figure shows that the two curves meet at an error ratio of 0.45, but the decrease in the steady-state of the TNLMS between 0.4 and 0.45 is larger than the increase in the steady-state of the mTNLMS between the same values. Therefore, a good selection for the error ratio value to use when comparing different Thresholded-NLMS algorithms is 0.4. This value provides good overall performance for both the TNLMS and mTNLMS algorithms. Table 6.1 also shows the numerical values of the steady-state performances plotted in Figure 6.13.

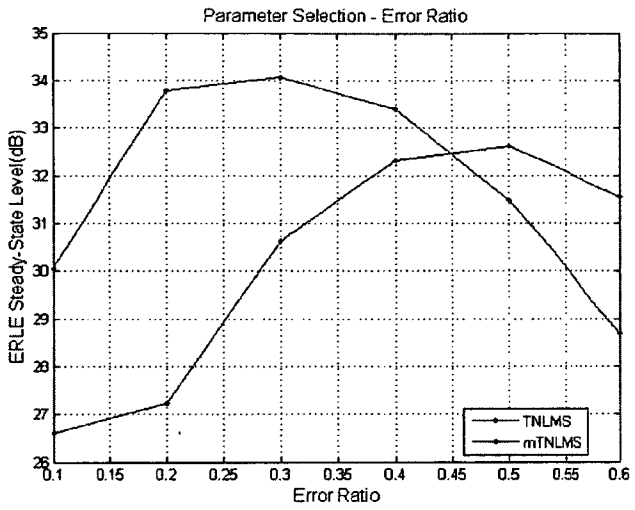


Figure 6.13: ERLE steady-state performance of TNLMS and mTNLMS for different values of the error ratio, with $\tau = 0.5$ using the impulse response in Error! Reference source not found..**b**

Table 6.1: ERLE steady-state performance of the TNLMS and mTNLMS for different error ratio values.						
Error Ratio	0.1	0.2	0.3	0.4	0.5	0.6
TNLMS	30.05	33.79	34.07	33.4	31.48	28.7
mTNLMS	26.6	27.23	30.63	32.32	32.61	31.56

The second parameter selected is the masking factor, denoted by τ, which is used in the mTNLMS algorithm only. The error ratio is fixed at 0.4 in the following simulations. Figure 6.14 and Figure 6.15 show simulation results for different values of the masking factor. When the masking factor is increased from 0.01 to 0.4 as shown in Figure 6.14, the performance improves greatly. This is because when τ is very small the mTNLMS step-size, $\mu_{mTNLMS}(n)$ given by equation (6.11)), increases greatly due to a much smaller

masked input power in the denominator. This causes the weight-update per tap to increase accordingly and the estimation error of most of the taps becomes large since they cannot converge to their actual values leading to a big misadjustment error. This analysis is similar to the behaviour of the NLMS algorithm with a large step-size. As τ increases the weight-update per tap becomes comparable to the magnitudes of the taps and therefore they converge leading to an overall improvement in performance.

The performance of the mTNLMS improves beyond that of the NLMS in terms of convergence speed, tracking capability, and steady-state when τ is equal to or larger than 0.4 as shown in Figure 6.15.

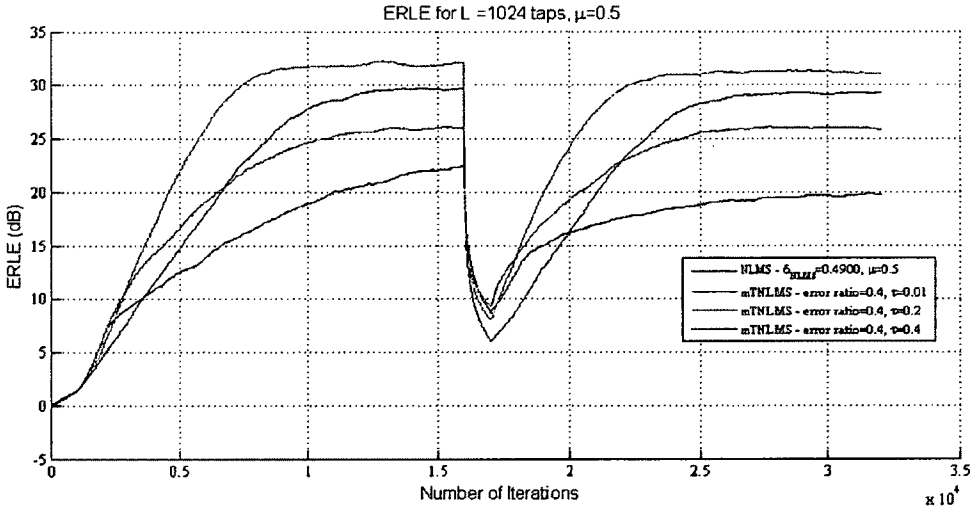


Figure 6.14: Comparative simulation of mTNLMS for small values of the masking factor.

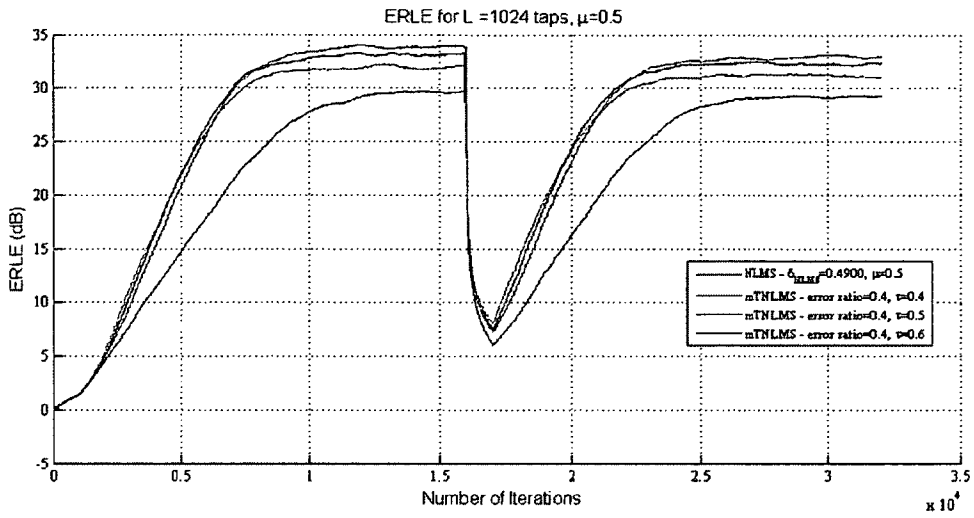


Figure 6.15: Comparative simulation of mTNLMS for large values of the masking factor.

A good choice for the masking factor is $\tau = 0.5$ which provides a good balance between good convergence speed and tracking capability on one hand and good steady-state performance on the other hand as can be deduced by visual inspection of Figure 6.15.

6.4.2 Results

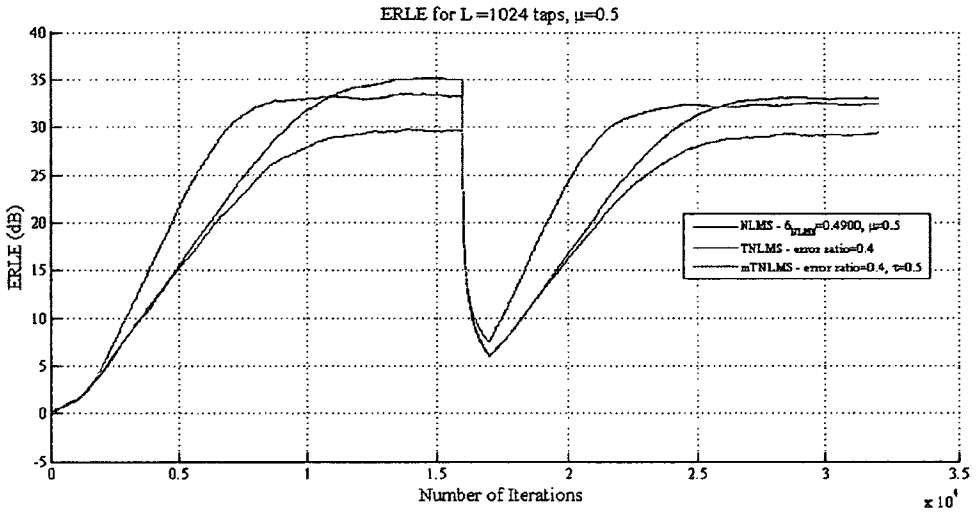


Figure 6.16: Comparative simulation of the NLMS, TNLMS, and mTNLMS using the impulse responses in Figure 6.3.

Figure 6.16 displays the ERLE simulation results for the NLMS, TNLMS, and mTNLMS algorithms. The error ratio used is 0.4, and the masking factor is set to $\tau = 0.5$.

As a first observation, the steady-state performance of the TNLMS and mTNLMS algorithms exceed that of the NLMS algorithm (which had an asymptotic ceiling of 30 dB SNR), because the cumulative estimation error of the inactive taps is replaced by a smaller cumulative omission error. The steady-state ERLE of the TNLMS is approximately 5 dB higher than the NLMS while that of the mTNLMS is approximately 4 dB higher The steady-state level of the TNLMS is slightly greater than that of the mTNLMS because as the step-size increases in the mTNLMS and the threshold value is fixed, some taps that were active in the TNLMS become inactive in the mTNLMS because their actual magnitudes become relatively smaller than their weight-updates.

The initial convergence speed of the TNLMS algorithm is the same as the NLMS, but it improves slowly over time. The mTNLMS, on the other hand, shows a significant improvement in the initial convergence speed due to the increase in the mTNLMS step-size. The convergence of TNLMS and mTNLMS exceed the NLMS by an average of 5 dB and 8 dB on average respectively at 10,000 iterations.

The tracking performance of the Thresholded-NLMS algorithms shows a similar improvement like the initial convergence where the mTNLMS drops to approximately 1.5 dB less than the two other algorithms when the EPIR is changed.

6.5 Conclusion

A thresholding technique is developed and incorporated into the NLMS algorithm resulting in the Thresholded-NLMS algorithms. These algorithms are proposed to solve the problem of the estimation errors resulting from tuning the small-valued coefficients using relatively larger individual step-sizes, and using the adaptation gain more efficiently. Two simulated impulse responses are used to emphasize the effect of small-valued coefficients on the ERLE performance of adaptive filters for AEC. The algorithms are simulated and compared showing that the proposed Thresholded-NLMS algorithms outperform the NLMS by approximately 4-5 dB on average in terms of steady-state performance and 5-8 dB on average in terms of convergence and tracking speeds.

CHAPTER 7 Proposed Thresholded Non-Parametric Variable Step-Size NLMS Algorithms

The thresholding technique proved to be successful when there was a significant number of taps that have actual magnitudes relatively smaller than their weight-updates. Since the weight-updates are governed by the step-size, this situation could also exist when the value of the step-size is large in variable step-size algorithms, which occurs during initial convergence and during tracking. Since the threshold value also depends on the step-size, it was intuitive to investigate the thresholding technique with a variable step-size algorithm. The threshold value will be varying as the step-size varies resulting in a dynamic grouping of the filter taps into active and inactive taps as they converge.

This chapter first summarizes the Non-Parametric Variable Step-Size NLMS algorithm **[11]** (NPVSS-NLMS) and then investigates extending the thresholding idea to the NPVSS-NLMS algorithm. Two types of algorithms are proposed: the Thresholded NPVSS-NLMS algorithms (NPVSS-TNLMS, and NPVSS-mTNLMS), and the Adaptive Thresholded NPVSS-NLMS algorithms (NPVSS-ATNLMS, and NPVSS-mATNLMS). The two simulated impulse responses in Figure 6.3 are used in the simulations in this chapter to compare the ERLE performance of the proposed algorithms to the NPVSS-NLMS algorithm. For simulations under stationary room conditions (static EPIR) the impulse response in Figure 6.3.b is used. For simulations under non-stationary room conditions to test the tracking capability of the algorithms, both impulse responses in Figure 6.3 are used sequentially.

7.1 Non-Parametric Variable Step-Size NLMS Algorithm (NPVSS-NLMS)

7.1.1 Algorithm Description

The nonparametric VSS-NLMS (NPVSS-NLMS) algorithm was proposed by Benesty et al. [**11**]. The variable step-size was derived based on cancelling the a posteriori error, $\varepsilon(n) = d(n) - \hat{\boldsymbol{c}}^T(n+1)\overline{\boldsymbol{x}}(n)$. Due to the presence of system noise, the practical approach was to have the power of the a posteriori error approach the power of the system noise:

$$E\{\,\varepsilon^2(n)\,\} = \sigma_w^2, \quad \forall n \tag{7.1}$$

$$\varepsilon(n) = d(n) - \hat{\boldsymbol{c}}^T(n+1)\overline{\boldsymbol{x}}(n)$$

where σ_w^2 is the power of environment noise (estimated or known). The derivation in [**11**] resulted in the NPVSS step-size is defined by

$$\mu_{NPVSS}(n) = \begin{Bmatrix} \beta(n) & if\ \hat{\sigma}_e(n) \geq \sigma_w \\ 0 & otherwise \end{Bmatrix} \tag{7.2}$$

where

$$\beta(n) = [\delta + \overline{\boldsymbol{x}}^T(n)\overline{\boldsymbol{x}}(n)]^{-1}\left[1 - \frac{\sigma_w}{\epsilon + \hat{\sigma}_e(n)}\right] \tag{7.3}$$

$$\beta(n) = \mu_{NLMS}(n)\,\alpha_{NPVSS}(n)$$

ϵ is a small positive number to avoid dividing by zero and $0 \leq \alpha_{NPVSS}(n) \leq 1$ is the normalized step-size. $\hat{\sigma}_e^2(n)$ is the current estimate of the mean-square error, and is computed as follows

$$\hat{\sigma}_e^2(n) = \lambda\, \hat{\sigma}_e^2(n-1) + (1-\lambda)\, e^2(n) \tag{7.4}$$

where λ is the exponential window factor defined by

$$\lambda = 1 - \frac{1}{KL}, K \geq 2 \tag{7.5}$$

The NPVSS-NLMS algorithm is listed in Table A.12.

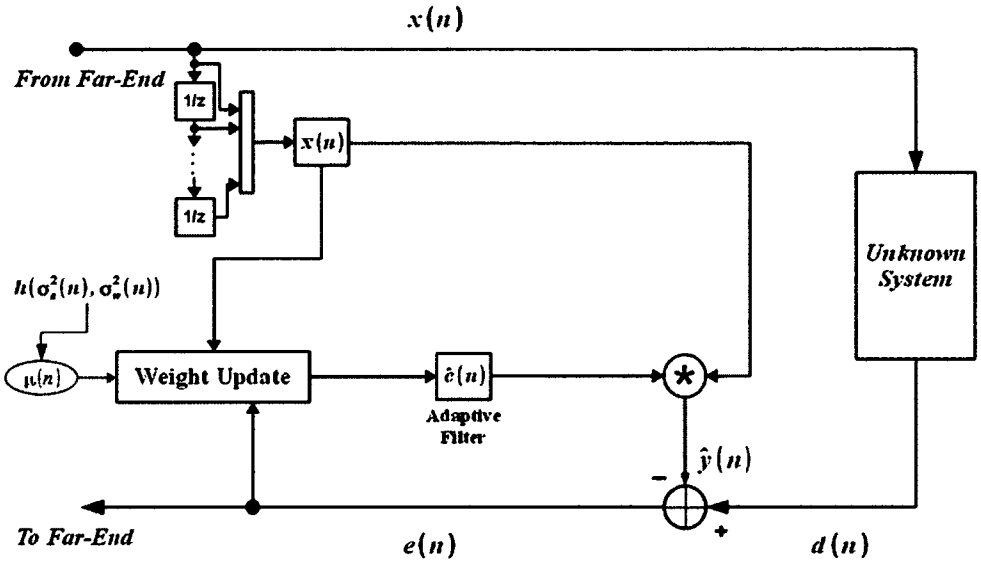


Figure 7.1: Block diagram for the NPVSS-NLMS algorithm.

7.1.2 Simulation of NPVSS-NLMS

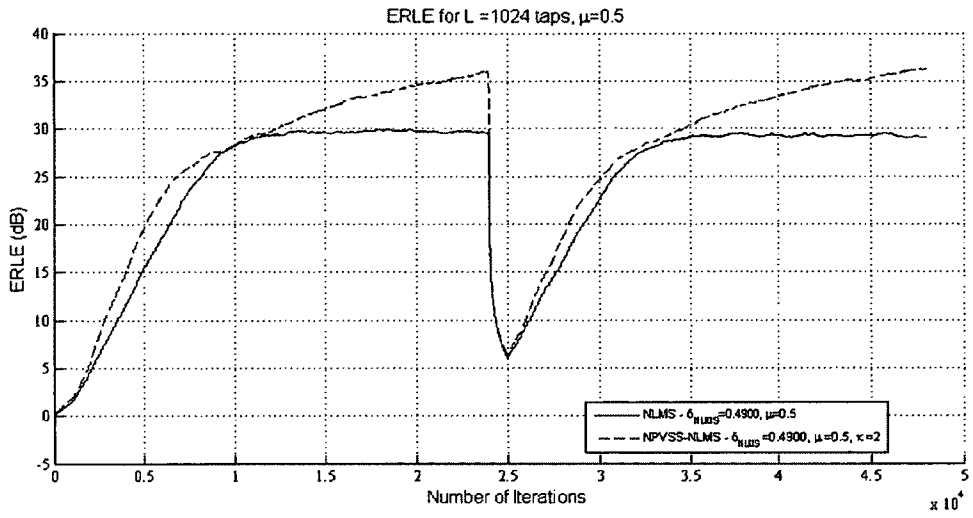


Figure 7.2: Comparative simulation of the NLMS and NPVSS-NLMS.

The parameter K is set to 2 for white input signal [**11**]. The results in Figure 7.2 show a slightly increase in the initial convergence speed and tracking of the NPVSS-NLMS as compared to the NLMS. However, convergence and tracking of the NPVSS-NLMS algorithm continues past the steady-state level of the NLMS (which levels off at 30 dB). This is explained by the following discussion on the NPVSS-NLMS algorithm (which is listed in Table A.12). When the step-size is large initially, the weight-update per tap is also large, and this is advantageous for adapting the large-valued taps first at a high rate, then as the step-size decreases overtime, the weight-update per tap also decreases which is again advantageous for adapting the small-valued taps at lower rates. This guarantees that the cumulative estimation error is kept minimal throughout the adaptation process in the NPVSS-NLMS. The adaptation continues until the power of the a posteriori error becomes smaller than the power of the system noise.

7.2 Thresholded Non-Parametric Variable Step-Size NLMS Algorithms

7.2.1 Algorithm Description

The threshold becomes a function of the time-varying NPVSS step-size,

$$threshold(n) = error\ ratio * \frac{\mu_{NPVSS}(n)}{L} \tag{7.6}$$

When the filter is not converged in the NPVSS-NLMS, the estimate of the mean-square error is large, therefore the value of the step-size is large and the value of the threshold is large as well. This means that during initial convergence or tracking when the filter is not converged; only a small number of taps are active and use the available adaptation gain (which is also large due to the large step-size) to converge fast. The convergence of the active taps in turn causes the estimate of the mean-square error to decrease, therefore the value of step-size is decreased and the value of the threshold is also decreased, allowing more and more inactive taps to become active taps and adapt. The process continues until the step-size value becomes almost constant as the filter coefficients approach a steady-state and the threshold becomes constant too where a group of inactive taps may remain inactive eventually.

The thresholding of the filter coefficients in the NPVSS-TNLMS and NPVSS-mTNLMS is governed by the same equation as in the Thresholded-NLMS algorithms, equation (6.6), and the masking operation in the NPVSS-mTNLMS is also governed by the same

equations as in the mTNLMS, in equations (6.8) and (6.9). The step-size in the NPVSS-mTNLMS becomes

$$\mu_{NPVSS-mTNLMS}(n) = \begin{cases} \beta_{mTNLMS}(n) & if\ \hat{\sigma}_e(n) \geq \sigma_w \\ 0 & otherwise \end{cases} \tag{7.7}$$

where

$$\beta_{mTNLMS}(n) = \left[\delta + \bar{\boldsymbol{x}}^T(n)\bar{\boldsymbol{x}}(n)\right]^{-1}\left[1 - \frac{\sigma_w}{\epsilon + \hat{\sigma}_e(n)}\right] \tag{7.8}$$

and $\bar{\boldsymbol{x}}$ is the masked input vector, given by equation (6.9).

The NPVSS-TNLMS algorithm is listed in Table B.4, and the NPVSS-mTNLMS algorithm is listed in Table B.5.

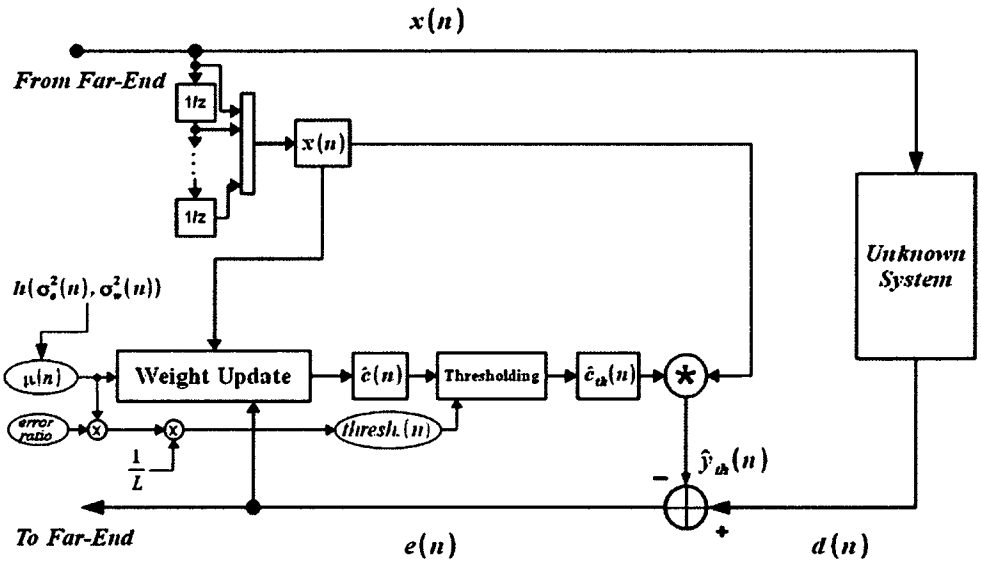


Figure 7.3: Block diagram for the NPVSS-TNLMS algorithm.

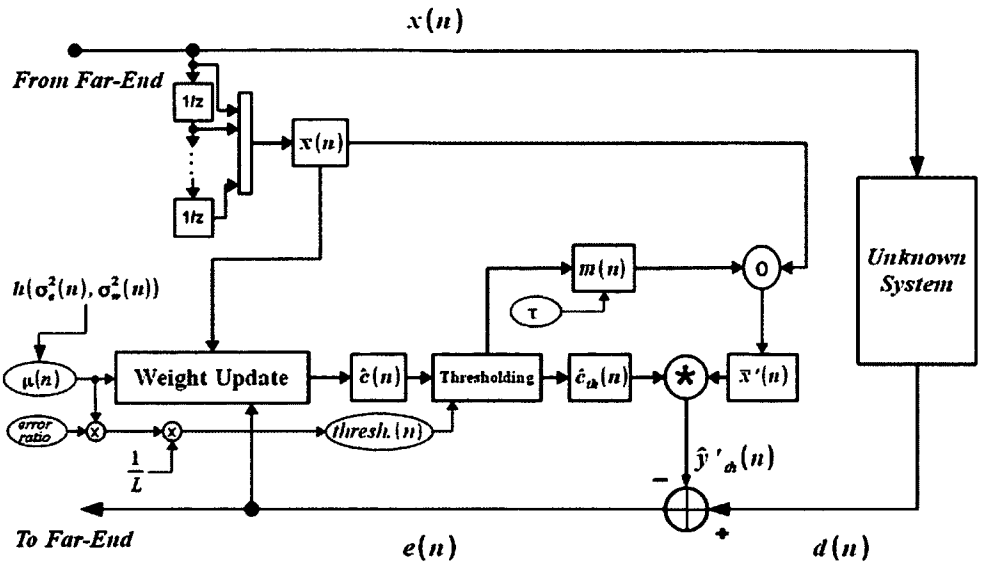


Figure 7.4: Block diagram for the NPVSS-mTNLMS algorithm.

7.2.2 Simulations of the Thresholded NPVSS-NLMS Algorithms

7.2.2.1 Parameter Selection

Figure 7.5 shows the simulation results for the NPVSS-TNLMS algorithm for different values of the error ratio . The results show that as the error ratio increases above 0.2, the steady-state levels off at 35 dB. This is because the continuous convergence of the small coefficients in the NPVSS-NLMS ceases by thresholding them and therefore do not contribute to the echo cancellation. The convergence speed improves as the error ratio increases until it become 0.5, after which it becomes the same. The best convergence performance for the NPVSS-TNLMS is provided by an error ratio of 0.4, but the steady-state level is less than that of the NPVSS-NLMS.

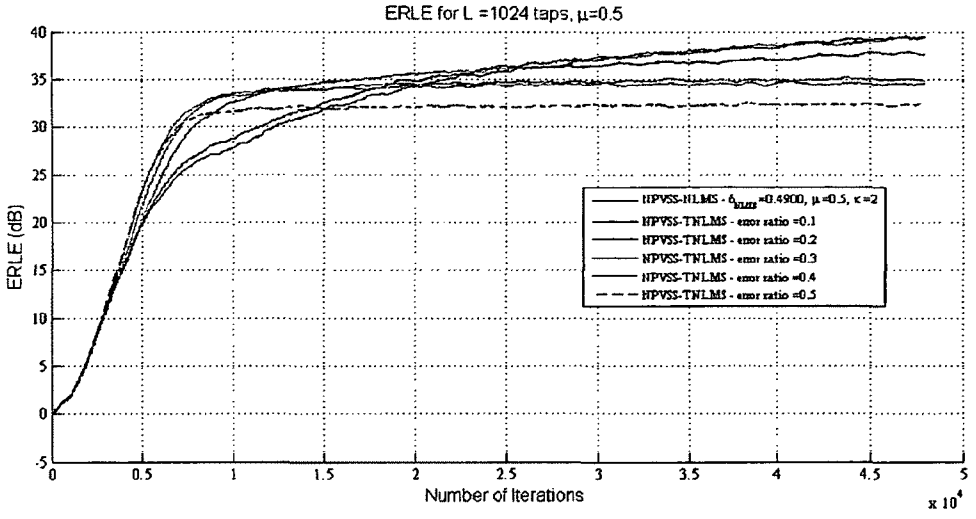


Figure 7.5: Comparative simulation of NPVSS-TNLMS for different values of the error ratio.

The NPVSS-mTNLMS algorithm is simulated for different values of the error ratio using a masking factor set to $\tau = 0.5$. Figure 7.6 shows that a value of 0.4 for the error ratio also provides a fair convergence performance but with a lower steady-state level, 35 dB, as compared to the NPVSS-NLMS, which increases to approximately 39 dB.

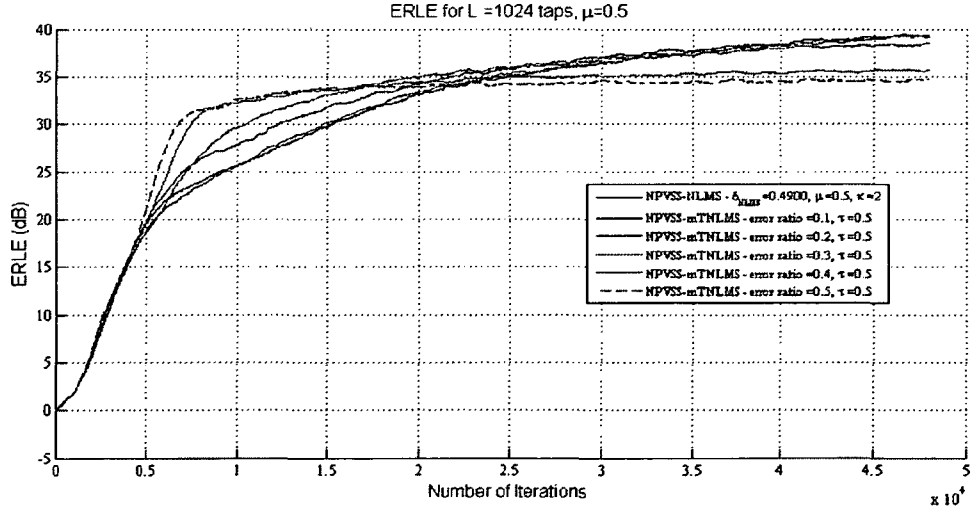


Figure 7.6: Comparative simulation of NPVSS-mTNLMS for different values of the error ratio with constant masking factor.

Figure 7.7 shows the simulation results for the NPVSS-mTNLMS algorithm for different values of the masking factor using an error ratio of 0.4. The results show that as the masking factor increases the convergence speed increases. For values larger than 0.4, the steady-state level decreases slightly and levels off at approximately 35 dB. A masking factor value of $\tau = 0.6$ provides the best overall performance. Any further increase in the masking factor provides no significant improvement. Here again, the steady-state levels off because the small coefficients are not allowed to converge to their actual values due to thresholding.

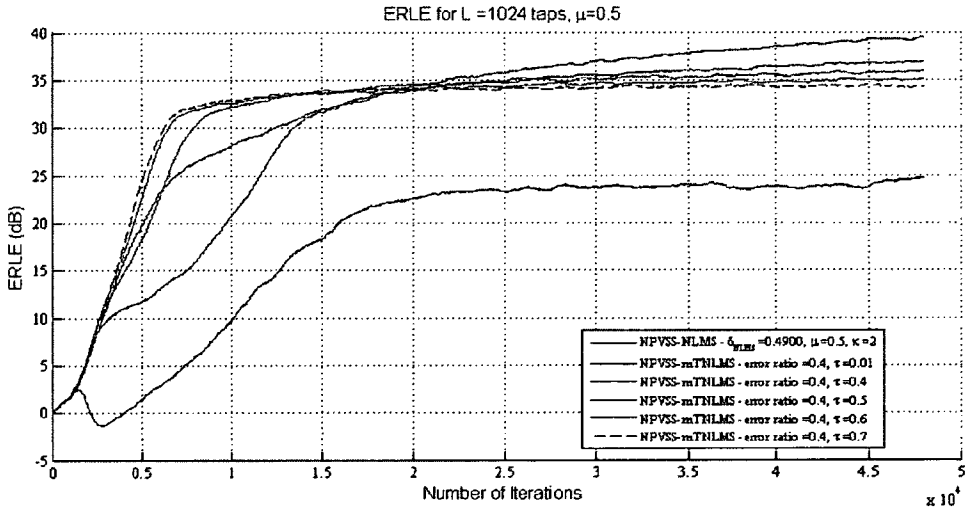


Figure 7.7: Comparative simulation of NPVSS-mTNLMS for different values of the masking factor with an error ratio of 0.4.

7.2.2.2 Results

Figure 7.8 shows the comparative simulation for the NPVSS-NLMS, NPVSS-TNLMS, and NPVSS-mTNLMS under stationary (static EPIR) room conditions using an error ratio of 0.4 and a masking factor of $\tau = 0.6$. Both Thresholded-NPVSS-NLMS

algorithms show the same improvement in the convergence speed over the NPVSS-NLMS algorithm, with an average of 5 dB more convergence at 10,000 iterations. The NPVSS-mTNLMS provides a slightly better steady-state performance as compared to the NPVSS-TNLMS, both of which are less than that of the NPVSS-NLMS algorithm.

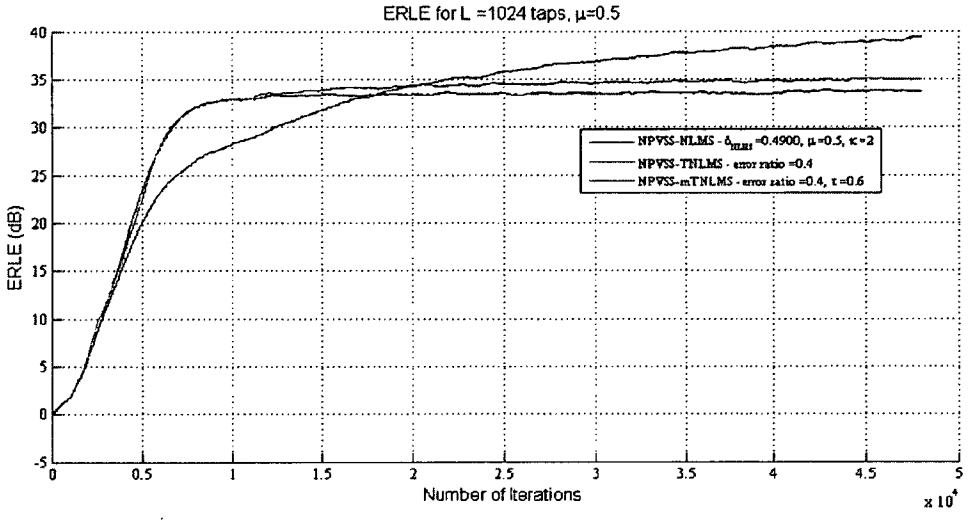


Figure 7.8: Comparative simulation of the NPVSS-NLMS, NPVSS-TNLMS, and NPVSS-mTNLMS.

Figure 7.9 shows the results using the same conditions but under non-stationary (dynamic EPIR) room conditions. The NPVSS-mTNLMS provides a higher tracking capability than the NPVSS-TNLMS as well as a higher steady-state level after tracking. The higher tracking can be explained by having larger step-size due to dividing it by the smaller masked input power.

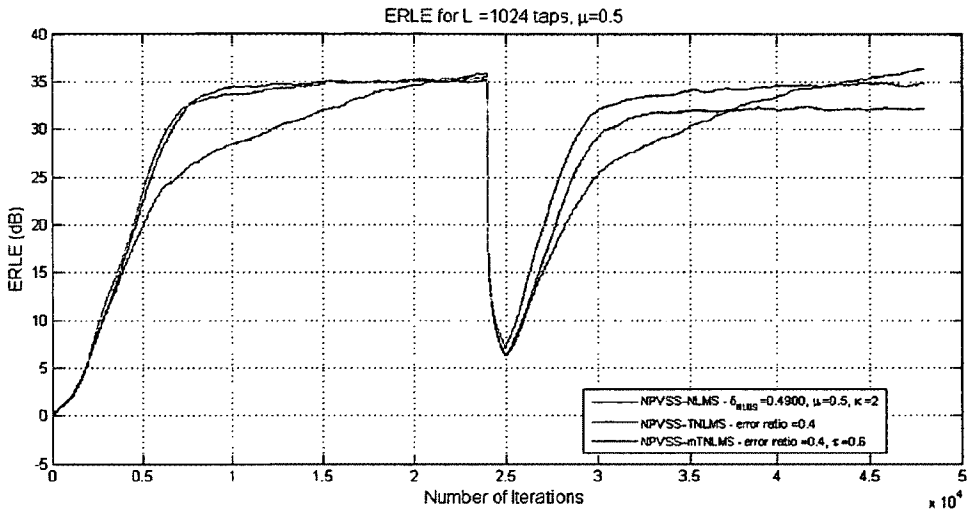


Figure 7.9: Comparative simulation of the NPVSS-NLMS, NPVSS-TNLMS, and NPVSS-mTNLMS.

7.3 Adaptive Thresholded Non-Parametric Variable Step-Size NLMS Algorithms (NPVSS-ATNLMS, NPVSS-mATNLMS)

7.3.1 Algorithm Description

The simulation results for the parameter selection of the error ratio, presented in section 7.2.2.1, show that as the error ratio increases the convergence speed and tracking capability improve. If the error ratio is large during initial convergence and tracking and then decreases over time as the filter converges, fewer taps will be allowed to converge first at a high rate and then more taps start to adapt as the error ratio decreases overtime. Coefficients with small magnitudes will be allowed to converge as well and therefore contribute to the cancellation of the echoes and eventually improving the steady-state performance.

In the Adaptive Thresholded NPVSS-NLMS algorithms, the variation in the error ratio is proposed by the following equation

$$error\ ratio(n) = \vartheta * \mu_{NPVSS}(n) \tag{7.9}$$

where ϑ is a fraction in the range $0 < \vartheta < 1$. The NPVSS-ATNLMS algorithm is listed in Table B.6, and the NPVSS-mATNLMS is listed in Table B.7.

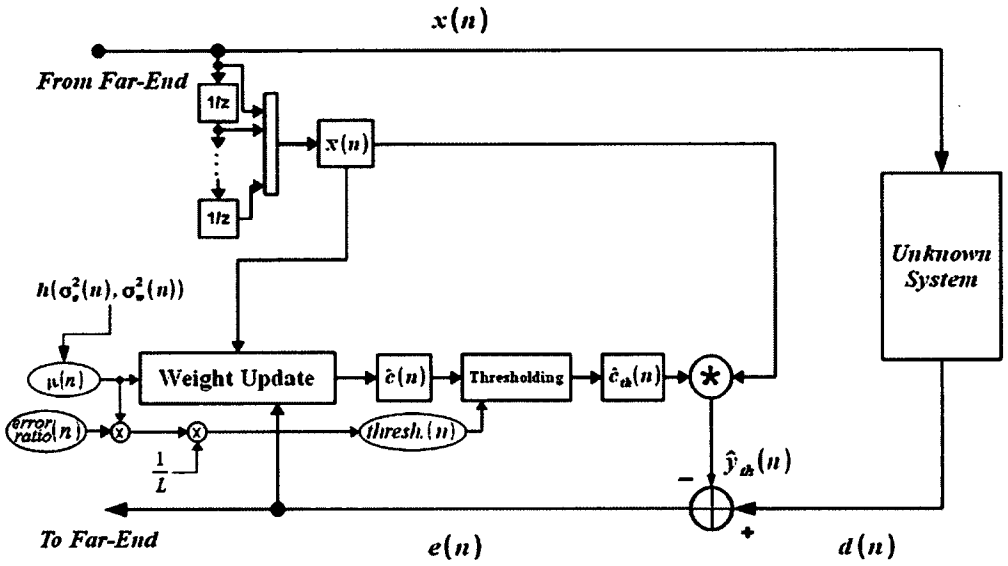


Figure 7.10: Block diagram for the NPVSS-ATNLMS algorithm.

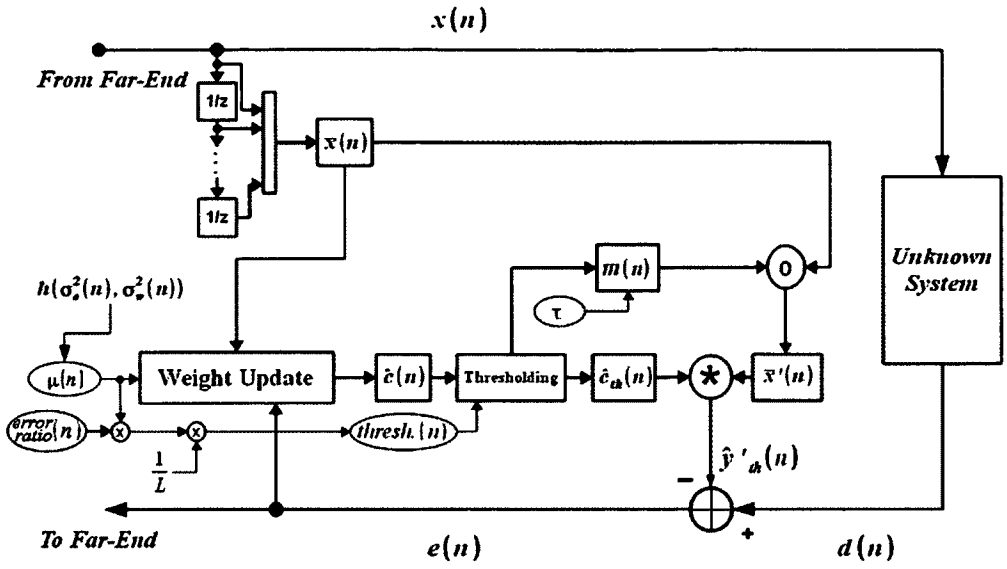


Figure 7.11: Block diagram for the NPVSS-mATNLMS algorithm.

7.3.2 Simulations of Adaptive Thresholded NPVSS-NLMS Algorithms

7.3.2.1 Parameter Selection

The error ratio and the masking factor are varied to find the optimal values to use with the NPVSS-ATNLMS and NPVSS-mATNLMS algorithms. Figure 7.12 shows the results for simulating the NPVSS-ATNLMS using ϑ= 0.3, 0.5 and 0.7. The convergence of the NPVSS-ATNLMS continues giving higher ERLE overtime similar to the NPVSS-NLMS algorithm. The initial convergence speed with ϑ= 0.5 is the highest and comparable to that of the NPVSS-TNLMS with an error ratio of 0.4. The NPVSS-ATNLMS provides both faster initial convergence similar to the NPVSS-TNLMS and continuous convergence similar to the NPVSS-NLMS.

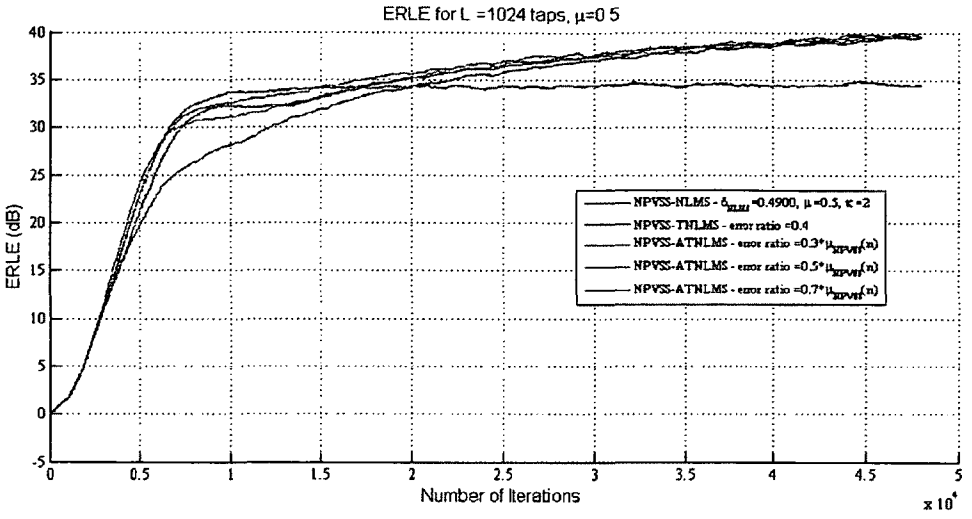


Figure 7.12: Comparative simulation of NPVSS-NLMS, NPVSS-TNLMS, and NPVSS-ATNLMS with different values of the error ratio for the NPVSS-ATNLMS.

Figure 7.13 shows simulation results for the NPVSS-mATNLMS with ϑ= 0.5, 0.7 and 0.9 with a fized masking factor of 0.6. The best overall performance is provided with ϑ= 0.7 where the initial convergence is even slightly better than that of the NPVSS-mTNLMS. After the initial phase of convergence, the NPVSS-mATNLMS slows down as compared to the NPVSS-mTNLMS but it eventually continues converging similar to the NPVSS-NLMS algorithm.

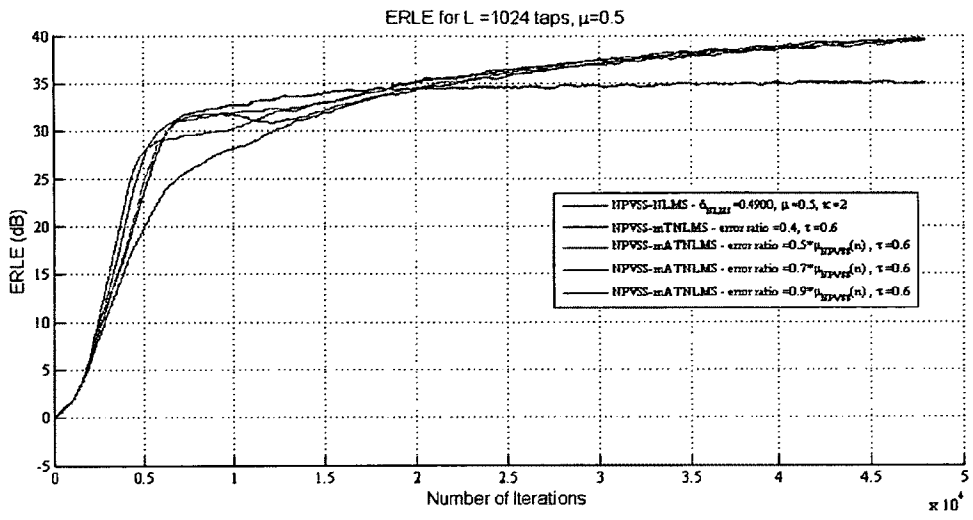


Figure 7.13: Comparative simulation of NPVSS-NLMS, NPVSS-mTNLMS, and NPVSS-mATNLMS with different values of the error ratio for the NPVSS-mATNLMS.

Figure 7.14 and Figure 7.15 show simulation results for the NPVSS-mATNLMS for different values of the masking factor. As the masking factor increases from 0.4 to 0.7 the convergence speed increases, as shown in Figure 7.14, and then as the masking factor increases further, as shown in Figure 7.15, the initial convergence rate decreases slightly. The best overall performance is provided by a masking factor of 0.7.

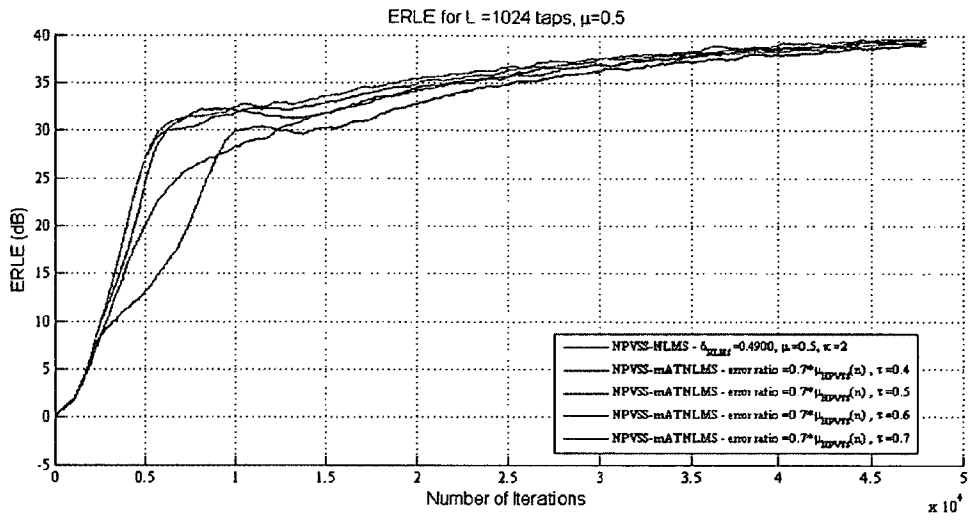


Figure 7.14: Comparative simulation of NPVSS-NLMS, NPVSS-TNLMS, and NPVSS-ATNLMS with different values of the masking factor for the NPVSS-ATNLMS.

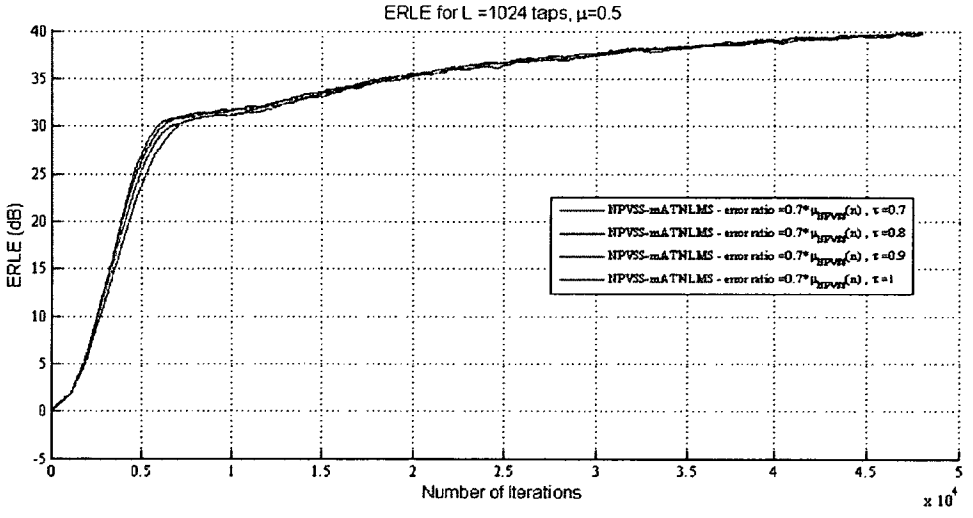


Figure 7.15: Comparative simulation of NPVSS-ATNLMS with more different values of the masking factor.

7.3.2.2 Results

Figure 7.16 shows the comparative simulation for the NPVSS-NLMS, NPVSS-TNLMS, and NPVSS-ATNLMS under stationary (static EPIR) room conditions using the

parameter values as selected in the previous section and shown in the legend of the plot. The NPVSS-ATNLMS has the same initial convergence speed as the NPVSS-TNLMS and continues to converge as the NPVSS-NLMS algorithm. The NPVSS-ATNLMS provides an overall improvement as compared to the other two algorithms.

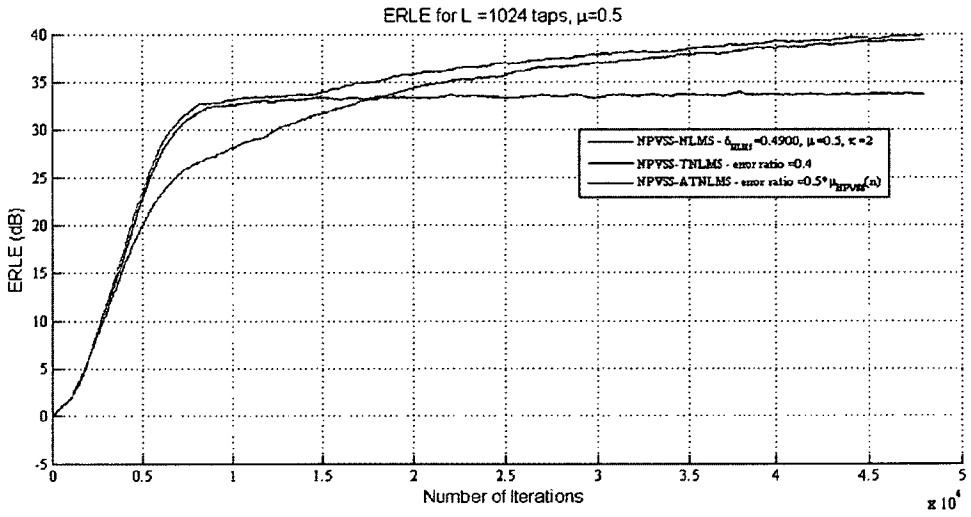


Figure 7.16: Comparative simulation of the NPVSS-NLMS, NPVSS-TNLMS, and NPVSS-ATNLMS.

Figure 7.17 shows the results for the NPVSS-NLMS, NPVSS-mTNLMS, and NPVSS-mATNLMS under stationary (static EPIR) room conditions. The NPVSS-mATNLMS has a slightly faster initial convergence as compared to the NPVSS-mTNLMS and continues to converge like the NPVSS-NLMS algorithm providing a better overall performance as compared to the other two algorithms.

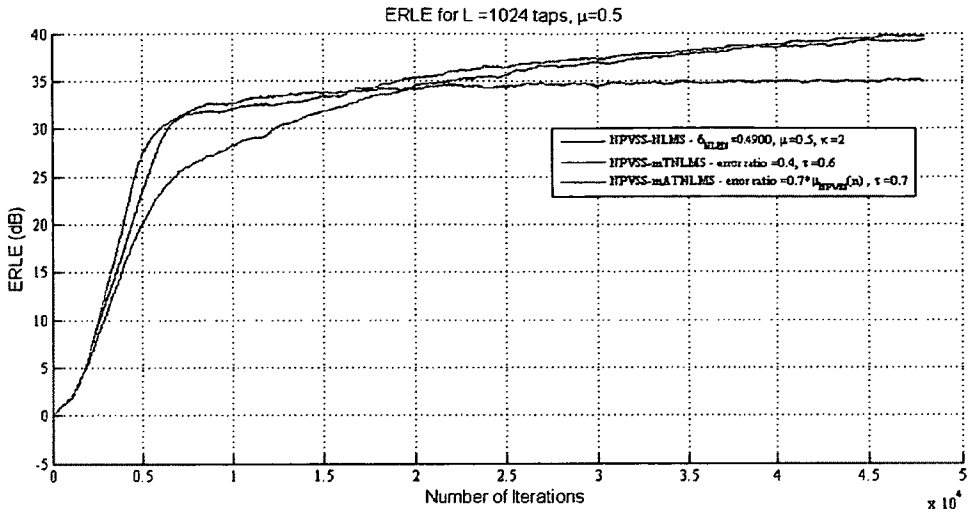


Figure 7.17: Comparative simulation for the NPVSS-NLMS, NPVSS-mTNLMS, and NPVSS-mATNLMS.

The NPVSS-ATNLMS and the NPVSS-mATNLMS algorithms are compared to the NPVSS-NLMS in Figure 7.18. The NPVSS-mATNLMS provides a slightly faster initial convergence, but the performances are comparable. The Adaptive Thresholded NPVSS-NLMS algorithms in general provide a significant improvement, an average of 6 dB, in the convergence speed when compared to the NPVSS-NLMS algorithm.

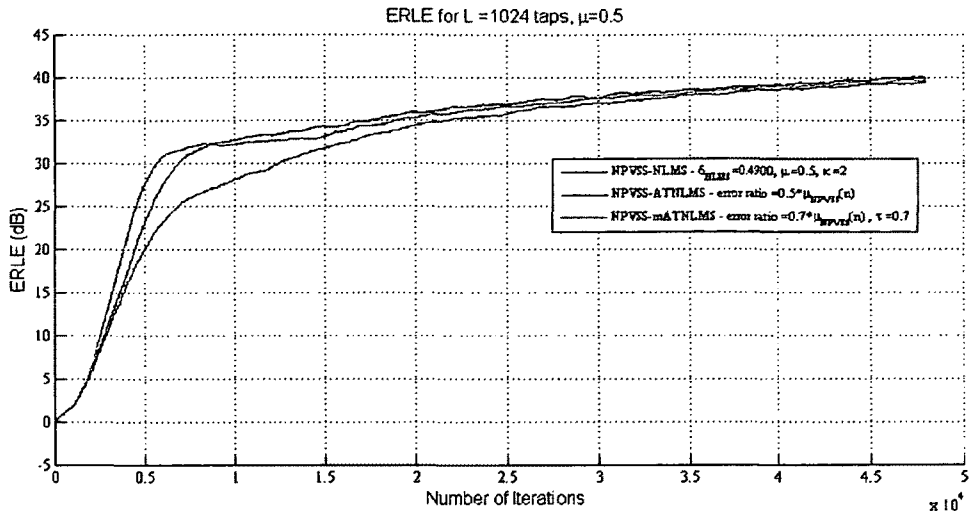


Figure 7.18: Comparative simulation for the NPVSS-NLMS, NPVSS-ATNLMS, and NPVSS-mATNLMS.

Figure 7.19, Figure 7.20, and Figure 7.21 show the same comparisons for the algorithms but under non-stationary (dynamic EPIR) room conditions.

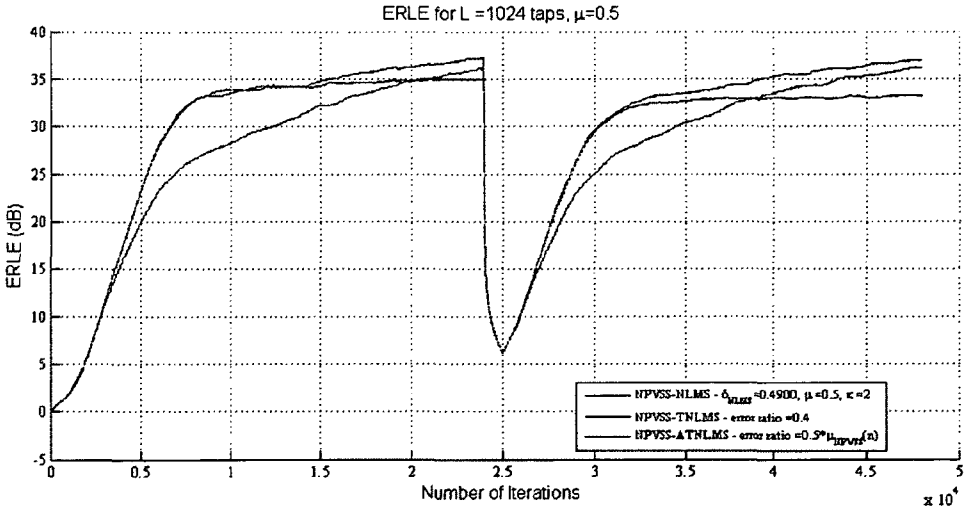


Figure 7.19: Comparative simulation of the NPVSS-NLMS, NPVSS-TNLMS, and NPVSS-ATNLMS.

The initial tracking capability of the NPVSS-ATNLMS is the same as that of the NPVSS-TNLMS, as shown in Figure 7.19. Tracking of the NPVSS-ATNLMS continues slightly faster than the NPVSS-NLMS algorithm.

The NPVSS-mATNLMS has a comparable initial tracking performance as well when compared to the NPVSS-mTNLMS as shown in Figure 7.20, and the tracking continues with a slightly higher speed than that of the NPVSS-NLMS algorithm.

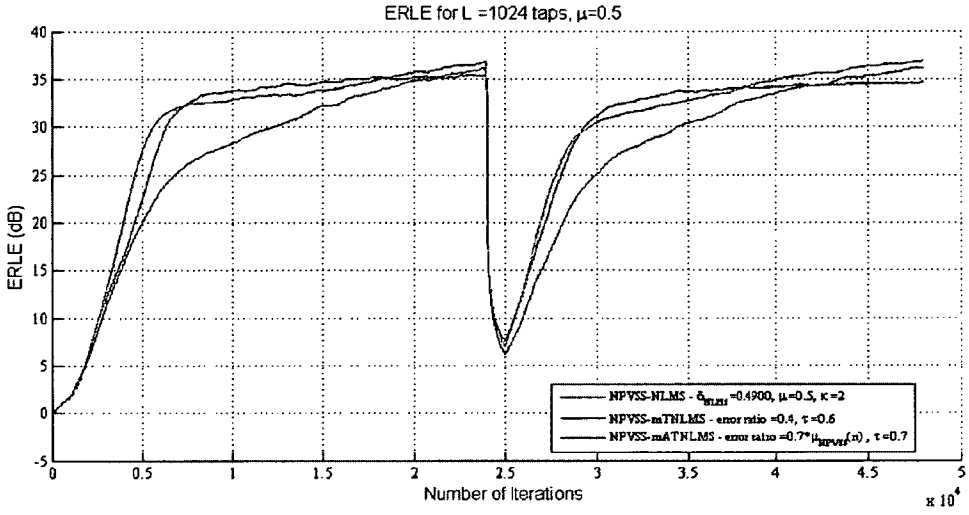


Figure 7.20: Comparative simulation for the NPVSS-NLMS, NPVSS-mTNLMS, and NPVSS-mATNLMS.

The NPVSS-mATNLMS provide a slightly faster initial tracking performance when compared to the NPVSS-ATNLMS which slows down at a later phase. The later tracking of both Adaptive Thresholded NPVSS-NLMS algorithms is the same.

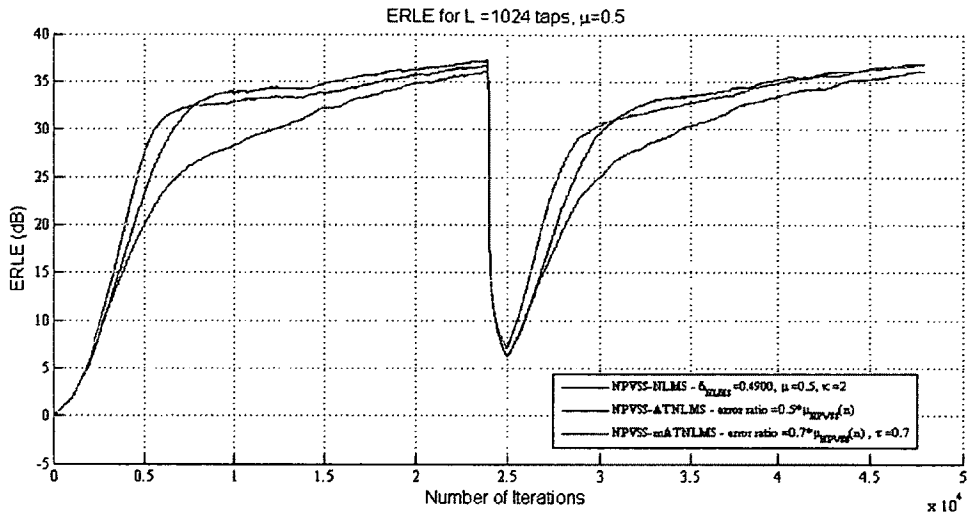


Figure 7.21: Comparative simulation for the NPVSS-NLMS, NPVSS-ATNLMS, and NPVSS-mATNLMS.

7.4 Conclusion

The thresholding technique is incorporated into the NPVSS-NLMS algorithm resulting in the development of the Thresholded NPVSS-NLMS algorithms (NPVSS-TNLMS, and NPVSS-mTNLMS). Moreover, by varying the error ratio parameter that is used to compute the threshold, the Adaptive Thresholded NPVSS-NLMS algorithms (NPVSS-ATNLMS, and NPVSS-mATNLMS) are developed. Comparative simulations are carried out for the proposed algorithms and compared with each other and with the NPVSS-NLMS algorithm under both stationary (static/time-invariant EPIR) and non-stationary (dynamic/time-variant EPIR) acoustical conditions using simulated impulse responses. Results show that the proposed Thresholded-NPVSS-NLMS algorithms outperform the NPVSS-NLMS by approximately 5 dB on average in terms of convergence rate, and 5-7

dB on average in terms of tracking capability. The steady-state performance of the Thresholded-NPVSS-NLMS, however, levels off earlier at the NLMS steady-state level.

The Adaptive Thresholded NPVSS-NLMS algorithms in general provide an average of 6 dB improvement in the convergence speed when compared to the NPVSS-NLMS algorithm and an average of 3 dB improvement in the tracking capability. The steady-state performance of the Adaptive Thresholded NPVSS-NLMS algorithms does not level off early like the Thresholded NPVSS-NLMS algorithms; it continues to increase same as the NPVSS-NLMS algorithm.

CHAPTER 8 CONCLUSIONS

This work investigated different types of variable step-size adaptive algorithms in the context of acoustic echo cancellation using impulse responses of hands-free portable devices. New techniques were developed to improve the performance of adaptive filters in terms of convergence speed, tracking capability and steady-state performance under both stationary (by using time-invariant impulse responses) and highly non-stationary environment conditions (by using time-variant impulse responses). The developed techniques are incorporated into existing adaptive algorithms with the aim to improve the performance with a modest increase in computational complexity. The next section summarizes the findings of this research.

8.1 Summary of Results

Acoustic echo path impulse responses were measured using a commercial smartphone operating in hands-free mode (see section 3.4). Two orientations for the smartphone were used: on its face and on its back. The results showed that most of the significant differences between the impulse responses of the smartphone occurred at the initial reflections segment. The two impulse responses were used sequentially in the simulations of adaptive algorithms to mimic a highly non-stationary acoustic environment due to a hugely variant impulse response.

A set of selected state-of-the-art PNLMS-based algorithms were summarized in terms of their different approaches to updating individual step-sizes for adaptive filtering, and had their echo return level enhancement performance compared. The SCMPNLMS algorithm

has the best overall performance but maybe relatively computationally expensive due to logarithmic computations.

The gradient-induced technique, (see section 5.1) was proposed to provide fast tracking performance in highly non-stationary acoustic environments. The technique uses both the estimate of the gradient of the coefficients and the magnitude of the coefficients in the individual step-size update rule to distribute the adaptation gain among the filter coefficients. The gradient-induced technique was incorporated into the PNLMS and other PNLMS-based algorithms that were presented in Chapter 4. The g-PNLMS algorithm outperformed both the PNLMS and the GGPNLMS using simulated and measured impulse responses. The gradient-induced technique also proved to improve the tracking performances of most other PNLMS-based algorithms. Moreover, the g-SPNLMS algorithm provided better tracking and steady-state performance when compared to the MPNLMS and AMPNLMS. This is an advantage since the g-SPNLMS does not require the logarithmic computations needed in the MPNLMS and the AMPNLMS.

The ASCMPNLMS was developed as a modification of existing individual variable step-size algorithms. The gradient-induced technique was also incorporated into the ASCMPNLMS and compared to other algorithms. The g-ASCMPNLMS provided the best overall performance compared to all other algorithms presented in Chapter 5 but at the expense of increased computational complexity.

The Thresholded-NLMS algorithms were developed in Chapter 6 by incorporating a thresholding technique into the NLMS algorithm. These algorithms are proposed to solve two problems: 1) reduce the estimation errors resulting from tuning of the large number

of coefficients with small magnitudes using relatively larger individual step-sizes, 2) and use the adaptation gain more efficiently to adapt coefficients with larger magnitudes using larger gains. Two simulated impulse responses were used to emphasize the effect of coefficients with small magnitudes on the ERLE performance of the NLMS and Thresholded-NLMS algorithms for AEC. The algorithms are simulated and compared showing that the proposed Thresholded-NLMS algorithms outperform the NLMS by approximately 4-5 dB on average in terms of steady-state performance and 5-8 dB on average in terms of convergence and tracking speeds.

The thresholding technique was also incorporated into the NPVSS-NLMS algorithm resulting in the development of the Thresholded NPVSS-NLMS algorithms (NPVSS-TNLMS, and NPVSS-mTNLMS) in section 7.2. Comparative simulations in section 7.2.2.2 showed that the proposed Thresholded-NPVSS-NLMS algorithms outperform the NPVSS-NLMS by approximately 5 dB on average in terms of convergence rate, and 5-7 dB on average in terms of tracking capability. The steady-state performance of the Thresholded-NPVSS-NLMS, however, levels off earlier at the NLMS steady-state level.

Adaptive Thresholded NPVSS-NLMS algorithms (NPVSS-ATNLMS, and NPVSS-mATNLMS) were also developed in section 7.3 by varying a constant parameter that is used to compute the threshold. The Adaptive Thresholded NPVSS-NLMS algorithms in general provided an average of 6 dB improvement in the convergence speed when compared to the NPVSS-NLMS algorithm and an average of 3 dB improvement in the tracking capability. The steady-state performance of the Adaptive Thresholded NPVSS-NLMS algorithms did not level off at a lower ERLE level like the Thresholded NPVSS-NLMS algorithms; it continued to increase similar to the NPVSS-NLMS algorithm.

8.2 Recommendations for Future Research

There are several directions in which this research could continue. The performance of the proposed algorithms could be evaluated using speech signal input as the far-end signal rather than WGN to investigate the effect of speech signal non-stationary aspects and verify the improvement in performances in a more realistic hands-free communication system. The effects of signal coloration can be investigated by using colored noise as well which can be easily obtained by simulation. The behavior of algorithms will differ relatively using colored and non-stationary speech signals.

More impulse response scenarios can be recorded and used to test the capabilities of the existing and proposed variable step-size adaptive algorithms to deal with different types and levels of non-stationary environments.

The NPVSS could be combined with the various PNLMS-based algorithms, and the thresholding technique could be extended to the PNLMS and other PNLMS-based algorithms. Moreover, the NPVSS along with thresholding techniques could be incorporated into PNLMS-based algorithms.

The proposed algorithms could be implemented in real-time on Digital Signal Processing (DSP) hardware. The computational complexities could be also compared by considering a specific platform for implementation. This also allows evaluating the performance of the algorithms in realistic hands-free portable devices, where attention must be drawn to numerical limitations of the algorithms if they were to be implemented on fixed-point DSP hardware. Fixed-point computations may introduce errors, and the performance of

different algorithms will differ based on the equations of each algorithm and how they are implemented in fixed-point.

The selection of an algorithm to suits an AEC designer's requirements depends on the specific environment, application area, and the performance measure of interest. Some applications might require high ERLE, others require low MSE or misalignment. This idea of algorithm selection can be extended to creating algorithm selection modules to select the algorithm based on the context or surrounding environment.

APPENDICES

Appendix A Variable Step-Size Algorithms Description

A.1 The NLMS Algorithm

Table A.1: The NLMS algorithm with gain-control factors.

Initialization: $\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$

Parameters: $0 < \mu_{NLMS} < 2$

$\delta_{NLMS} = cst.\sigma_x^2$

Error: $e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$

Gain-control:

$$\boldsymbol{G}(n) = \boldsymbol{I} = \begin{pmatrix} 1 & 0 & & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}_{LxL}$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{NLMS}}$$

Computational complexity

The term $\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)$ can be computed using element-by-element product between $\overline{\boldsymbol{x}}(n)$ and $\overline{\boldsymbol{g}}(n)$ which requires L multiplications. Similarly, the term $\overline{\boldsymbol{x}}^T(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) = \sum_{l=0}^{L-1} x^2(n-l)\, g_l(n)$ requires $2L$ multiplications and $L-1$ additions. The filter update equation requires in total $2L$ additions, $4L+1$ multiplications and 1 division. The

practical complexity of the NLMS algorithm is as stated in Chapter 2; however, the analysis here is required for the proceeding sections of various PNLMS-based algorithms.

A.2 The PNLMS Algorithm

Table A.2: The PNLMS algorithm with gain-control factors.

Initialization: $\hat{\boldsymbol{c}}(0) = \overline{\boldsymbol{0}}$

Parameters: $0 < \mu_{NLMS} < 2$

$$\delta_{PNLMS} = \frac{\delta_{NLMS}}{L} = \frac{cst.\sigma_x^2}{L}$$

$$\rho = \frac{5}{L}, \qquad \gamma = 0.001$$

Error: $e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$

Gain-control: $\varphi(n) = \rho\, max\{\gamma, |\hat{c}_0(n)|, |\hat{c}_1(n)|, \ldots, |\hat{c}_{L-1}(n)|\}$

$$\theta_l(n) = max\{\varphi(n), |\hat{c}_l(n)|\} \quad \forall l, \qquad l = 0, 1, \ldots L-1$$

$$g_l(n) = \frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} \qquad l = 0, 1, \ldots L-1$$

$$\boldsymbol{G}(n) = diag\{\overline{\boldsymbol{g}}(n)\}$$

Filter Update: $$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{PNLMS}}$$

Computational complexity

The PNLMS requires $2L + 1$ additions, $5L + 2$ multiplications, 2 divisions, and $2L$ comparisons [**20**].

A.3 The PNLMS++ Algorithm

Table A.3: The PNLMS++ algorithm with gain-control factors.

Initialization:
$$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$$

Parameters:
$$0 < \mu_{NLMS} < 2$$

$$\delta_{PNLMS} = \frac{\delta_{NLMS}}{L} = \frac{cst.\sigma_x^2}{L}$$

$$\rho = \frac{5}{L}, \qquad \gamma = 0.001$$

Error:
$$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$$

Gain-control:
$$\varphi(n) = \rho max\{\gamma, |\hat{c}_0(n)|, |\hat{c}_1(n)|, \dots, |\hat{c}_{L-1}(n)|\}$$

$$\theta_l(n) = max\{\varphi(n), |\hat{c}_l(n)|\} \quad \forall l, \qquad l = 0, 1, \dots L-1$$

$$g_l(n) = \frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} \qquad l = 0, 1, \dots L-1$$

$$\boldsymbol{G}(n) = \begin{Bmatrix} I & n\ even\ (or\ n\ mod\ k\ != 0) \\ diag\{\overline{\boldsymbol{g}}(n)\} & n\ odd\ (or\ every\ kth\ iteration) \end{Bmatrix}$$

$$\delta = \begin{Bmatrix} \delta_{NLMS} & n\ even\ (or\ n\ mod\ k\ != 0) \\ \delta_{PNLMS} & n\ odd\ (or\ every\ kth\ iteration) \end{Bmatrix}$$

Filter Update:
$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{PNLMS}}$$

Computational complexity

The average computational complexity of the PNLMS++ is less than that of the PNLMS, and the maximum is the same as the PNLMS.

A.4 The IPNLMS Algorithm

Table A.4: The IPNLMS algorithm with gain-control factors.

Initialization:	$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$
Parameters:	$0 < \mu_{NLMS} < 2$
	$\delta_{IPNLMS} = \delta_{NLMS}\,\frac{1-\alpha}{2L} = cst.\,\sigma_x^2\,\frac{1-\alpha}{2L}$
	$-1 \leq \alpha < 1$
	$\epsilon > 0$
Error:	$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$
Gain-control:	$g_l(n) = \frac{1-\alpha}{2L} + (1+\alpha)\frac{\lvert\hat{c}_l(n)\rvert}{2\lVert\hat{\boldsymbol{c}}(n)\rVert_1 + \epsilon} \qquad l = 0, 1, \ldots L-1$
	$\boldsymbol{G}(n) = diag\{\overline{\boldsymbol{g}}(n)\}$
Filter Update:	$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{IPNLMS}}$

Computational complexity

The IPNLMS requires $3L + 2$ additions, $5L + 2$ multiplications, and 2 divisions per iteration [**20**].

A.5 The MPNLMS Algorithm

Table A.5: The MPNLMS algorithm with gain-control factors.

Initialization: $\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$

Parameters: $0 < \mu_{NLMS} < 2$

$\delta_{MPNLMS} = \delta_{NLMS} = cst.\sigma_x^2$

$\rho = \frac{5}{L}, \qquad \gamma = 0.01$

$\beta = 1000$

Error: $e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$

Gain-control: $F(|\hat{c}_l(n)|) = ln(1 + \beta|\hat{c}_l(n)|), \qquad l = 0, 1, \ldots, L-1.$

$\varphi(n) = \rho max\{\gamma, F(|\hat{c}_0(n)|), F(|\hat{c}_1(n)|), \ldots, F(|\hat{c}_{L-1}(n)|)\}$

$\theta_l(n) = max\{\varphi(n), F(|\hat{c}_l(n)|)\} \quad \forall l, \qquad l = 0, 1, \ldots L-1$

$g_l(n) = \frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} \qquad l = 0, 1, \ldots L-1$

$\boldsymbol{G}(n) = diag\{\overline{\boldsymbol{g}}(n)\}$

Filter Update: $\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{MPNLMS}}$

Computational complexity

The mu-law function may require more computations due to the logarithm function used.

The MPNLMS requires L logarithm computations, L multiplications, and L additions

more than the PNLMS algorithm per iteration. Therefore in total it requires $3L + 1$ additions, $6L + 2$ multiplications, 2 divisions, $2L$ comparisons, and L logarithms [**20**]. Since the MPNLMS maybe computationally expensive, an approximation of the logarithm function is proposed in the segment-PNLMS algorithm covered in the next section.

A.6 The SPNLMS Algorithm

Table A.6: The SPNLMS algorithm with gain-control factors.

Initialization:

$$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$$

Parameters:

$$0 < \mu_{NLMS} < 2$$

$$\delta_{SPNLMS} = \delta_{NLMS} = cst.\sigma_x^2$$

$$\rho = \frac{5}{L}, \qquad \gamma = 0.01$$

Error:

$$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$$

Gain-control:

$$F(|\hat{c}_l(n)|) = \begin{Bmatrix} 200|\hat{c}_l(n)| & |\hat{c}_l(n)| < 0.005 \\ 1 & otherwise \end{Bmatrix},$$

$$l = 0, 1, \ldots, L-1.$$

$$\varphi(n) = \rho\, max\{\gamma, F(|\hat{c}_0(n)|), F(|\hat{c}_1(n)|), \ldots, F(|\hat{c}_{L-1}(n)|)\}$$

$$\theta_l(n) = max\{\varphi(n), F(|\hat{c}_l(n)|)\} \quad \forall l, \qquad l = 0, 1, \ldots L-1$$

$$g_l(n) = \frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} \qquad l = 0, 1, \ldots L-1$$

$$\boldsymbol{G}(n) = diag\{\overline{\boldsymbol{g}}(n)\}$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{SPNLMS}}$$

Computational complexity

The SPNLMS requires less than L extra multiplications and comparisons as compared to the MPNLMS, therefore it requires in total $2L + 1$ additions, $6L + 2$ multiplications, 2 divisions, and $2L$ comparisons [17].

A.7 The AMPNLMS Algorithm

Table A.7: The AMPNLMS algorithm with gain-control factors.

Initialization:

$$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$$

Parameters:

$$0 < \mu_{NLMS} < 2$$

$$\delta_{AMPNLMS} = \delta_{NLMS} = cst.\,\sigma_x^2$$

$$\rho = \frac{5}{L}, \qquad \gamma = 0.01$$

$$0 < \varepsilon \leq 1, \qquad \nu = 1000$$

Error:

$$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$$

Gain-control:

$$\hat{\sigma}_e^2(n+1) = \varepsilon\,\hat{\sigma}_e^2(n) + (1-\varepsilon)e^2(n)$$

$$\beta(n+1) = \frac{1}{\sqrt{\frac{\hat{\sigma}_e^2(n+1)}{\nu L \sigma_x^2}}}$$

$$F(|\hat{c}_l(n)|) = ln(1 + \beta(n+1)|\hat{c}_l(n)|), \qquad l = 0, 1, \ldots, L-1.$$

$$\varphi(n) = \rho max\{\gamma, F(|\hat{c}_0(n)|), F(|\hat{c}_1(n)|), \ldots, F(|\hat{c}_{L-1}(n)|)\}$$

$$\theta_l(n) = max\{\varphi(n), F(|\hat{c}_l(n)|)\} \quad \forall l, \qquad l = 0, 1, \ldots L-1$$

$$g_l(n) = \frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} \qquad l = 0, 1, \ldots L-1$$

$$\boldsymbol{G}(n) = diag\{\overline{\boldsymbol{g}}(n)\}$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{AMPNLMS}}$$

Computational complexity

The AMPNLMS algorithm requires five multiplications, one addition, and a square root operation more than the MPNLMS algorithm per iteration.

A.8 The SCPNLMS Algorithm

Table A.8: The SCPNLMS algorithm with gain-control factors.

Initialization: $\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$

Parameters: $0 < \mu_{NLMS} < 2$

$$\delta_{SCPNLMS} = \frac{\delta_{NLMS}}{L} = \frac{cst.\sigma_x^2}{L}$$

$$\gamma = 0.01$$

$$\lambda = 6$$

Error: $e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$

Gain-control:

$$\xi\left(\hat{\boldsymbol{c}}(n)\right) = \frac{L}{L-\sqrt{L}}\left\{1 - \frac{\|\hat{\boldsymbol{c}}(n)\|_1}{\sqrt{L}\|\hat{\boldsymbol{c}}(n)\|_2}\right\}, \qquad n \geq L$$

$$\rho(n) = \begin{cases} 5/L & n < L \\ e^{-\lambda\xi(\hat{\boldsymbol{c}}(\boldsymbol{n}))} & n \geq L \end{cases}$$

$$\varphi(n) = \rho(n)\, max\{\gamma, |\hat{c}_0(n)|, |\hat{c}_1(n)|, \ldots, |\hat{c}_{L-1}(n)|\}$$

$$\theta_l(n) = max\{\varphi(n), |\hat{c}_l(n)|\} \quad \forall l, \qquad l = 0, 1, \ldots L-1$$

$$g_l(n) = \frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} \qquad l = 0, 1, \ldots L-1$$

$$\boldsymbol{G}(n) = diag\{\overline{\boldsymbol{g}}(n)\}$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^{\boldsymbol{T}}(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{SCPNLMS}}$$

Computational complexity

Given a certain λ value, $\rho(n)$ (given by equation (4.19)) can be computed using a look-up-table defined for $0 < \xi(n) \leq 1$. The additional complexity arises from the computation of the sparseness measure, $\xi(\hat{\boldsymbol{c}}(n))$ where the l-norms require $2L$ additions and L multiplications. The total computations are $4L + 2$ additions, $6L + 4$ multiplications, 3 divisions, and $2L$ comparisons per iteration.

A.9 The SCMPNLMS Algorithm

Table A.9: The SCMPNLMS algorithm with gain-control factors.

Initialization: $\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$

Parameters: $0 < \mu_{NLMS} < 2$

$\delta_{SCMPNLMS} = \delta_{NLMS} = cst.\sigma_x^2$

$\gamma = 0.01$

$\beta = 1000$

Error: $e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$

Gain-control: $F(|\hat{c}_l(n)|) = ln(1 + \beta|\hat{c}_l(n)|), \qquad l = 0, 1, \ldots, L-1.$

$$\xi\left(\hat{\boldsymbol{c}}(n)\right) = \frac{L}{L-\sqrt{L}}\left\{1 - \frac{\|\hat{\boldsymbol{c}}(n)\|_1}{\sqrt{L}\|\hat{\boldsymbol{c}}(n)\|_2}\right\}, \qquad n \geq L$$

$$\rho(n) = \begin{cases} {}^{5}/_{L} & n < L \\ e^{-\lambda\xi(\hat{\boldsymbol{c}}(n))} & n \geq L \end{cases}$$

$$\varphi(n) = \rho(n)\, max\{\gamma, F(|\hat{c}_0(n)|), F(|\hat{c}_1(n)|), \ldots, F(|\hat{c}_{L-1}(n)|)\}$$

$$\theta_l(n) = max\{\varphi(n), F(|\hat{c}_l(n)|)\} \quad \forall l, \qquad l = 0, 1, \ldots L-1$$

$$g_l(n) = \frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} \qquad l = 0, 1, \ldots L-1$$

$$\boldsymbol{G}(n) = diag\{\overline{\boldsymbol{g}}(n)\}$$

Filter Update: $$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^{\boldsymbol{T}}(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{SCMPNLMS}}$$

Computational complexity

The total computations required per iteration are $5L + 2$ additions, $7L + 4$ multiplications, 3 divisions, $2L$ comparisons, and L logarithms.

A.10 The SCIPNLMS algorithm

Table A.10: The SCIPNLMS algorithm with gain-control factors.

Initialization:

$$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$$

Parameters:

$$0 < \mu_{NLMS} < 2$$

$$\delta_{SCIPNLMS} = \delta_{NLMS}\,\frac{1-\alpha}{2L} = cst.\,\sigma_x^2\,\frac{1-\alpha}{2L}$$

$$-1 \le \alpha < 1$$

$$\epsilon > 0$$

Error:

$$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$$

Gain-control:

$$\xi\big(\hat{\boldsymbol{c}}(n)\big) = \frac{L}{L-\sqrt{L}}\left\{1 - \frac{\|\hat{\boldsymbol{c}}(n)\|_1}{\sqrt{L}\|\hat{\boldsymbol{c}}(n)\|_2}\right\}, \qquad n \ge L$$

$$g_l(n) = \frac{1-\alpha}{2L}\left(\frac{1 - 0.5\,\xi\big(\hat{\boldsymbol{c}}(n)\big)}{L}\right) + (1+\alpha)\left(\frac{1 + 0.5\,\xi\big(\hat{\boldsymbol{c}}(n)\big)}{L}\right)\frac{|\hat{c}_l(n)|}{2\|\hat{\boldsymbol{c}}(n)\|_1 + \epsilon}$$

$$l = 0, 1, \ldots L-1$$

$$\boldsymbol{G}(n) = diag\{\overline{\boldsymbol{g}}(n)\}$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\,\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^{\boldsymbol{T}}(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{SCIPNLMS}}$$

Computational complexity

The l_1-norm is already computed for the IPNLMS in equation (4.20), therefore the SCIPNLMS requires only an additional $L + 3$ additions, $L + 6$ multiplications, and 1 division. Therefore the total number of computations required per iteration is $4L + 5$ additions, $6L + 8$ multiplications, and 3 divisions [**20**].

A.11 The GGPNLMS algorithm

Table A.11: The GGPNLMS algorithm with gain-control factors.

Initialization: $\hat{\boldsymbol{c}}(0) = \overline{\boldsymbol{0}}$

Parameters: $0 < \mu_{NLMS} < 2$

$$\delta_{GGPNLMS} = \frac{\delta_{NLMS}}{L}$$

$$\eta = 0.9999\,, \qquad \epsilon = 0, \qquad \gamma = 1$$

Error: $e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$

Gain-control:

$$\tilde{\boldsymbol{c}}(n) = \eta\,\tilde{\boldsymbol{c}}(n-1) + (1-\eta)\hat{\boldsymbol{c}}(n-1)$$

$$\hat{\boldsymbol{s}}(n) = \hat{\boldsymbol{c}}(n-1) - \gamma\,\tilde{\boldsymbol{c}}(n-1)$$

$$\tilde{\boldsymbol{s}}(n) = \epsilon\,\tilde{\boldsymbol{s}}(n-1) + (1-\epsilon)\,abs\{\hat{\boldsymbol{s}}(n)\}$$

$$g_l(n) = \frac{1-\beta}{2}\frac{\tilde{s}_l(n-1)}{\sum_{l=0}^{L-1}\tilde{s}_l(n-1)} + \frac{\beta}{2L} \qquad l = 0, 1, \ldots L-1$$

$$\boldsymbol{G}(n) = diag\{\overline{\boldsymbol{g}}(n)\}$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{GGPNLMS}}$$

Computational complexity

The GGPNLMS requires $7L$ additions, $10L + 1$ multiplications, and 2 divisions per iteration.

A.12 The Non-Parametric Variable Step-Size NLMS Algorithm

Table A.12: The NPVSS-NLMS algorithm with gain-control factors.

Initialization:	$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$
	$\hat{\sigma}_e^2(0) = 0$
Parameters:	$0 < \mu_{NLMS} < 2$
	$\delta_{NLMS} = cst.\sigma_x^2$
	$\lambda = 1 - \frac{1}{KL}, K \geq 2$
Error:	$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$
Gain-control:	$\hat{\sigma}_e^2(n) = \lambda\, \hat{\sigma}_e^2(n-1) + (1-\lambda)\, e^2(n)$
	$\beta(n) = [\delta_{NLMS} + \overline{\boldsymbol{x}}^T(n)\overline{\boldsymbol{x}}(n)]^{-1} \left[1 - \frac{\sigma_w}{\epsilon + \hat{\sigma}_e(n)}\right]$
	$\mu_{NPVSS}(n) = \begin{Bmatrix} \beta(n) & if\ \hat{\sigma}_e(n) \geq \sigma_w \\ 0 & otherwise \end{Bmatrix}$
Filter Update:	$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NPVSS}(n)\overline{\boldsymbol{x}}(n)e(n)$

Computational Complexity

The NPVSS-NLMS requires $4L + 5$ multiplications, $4L + 3$ additions, 2 divisions, and 1 square-root per iteration.

Appendix B Proposed Variable Step-Size Algorithms Description

B.1 The ASCMPNLMS Algorithm

Table B.1: The ASCMPNLMS algorithm with gain-control factors.

Initialization:

$$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$$

Parameters:

$$0 < \mu_{NLMS} < 2, \qquad \delta_{ASCMPNLMS} = \delta_{NLMS} = cst.\sigma_x^2$$

$$\gamma = 0.001,\ \varepsilon = 0.99,\ \nu = 1000, \varsigma = 150$$

Error:

$$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$$

Gain-control:

$$\hat{\sigma}_e^2(n+1) = \varepsilon\,\hat{\sigma}_e^2(n) + (1-\varepsilon)e^2(n)$$

$$\beta(n+1) = \frac{1}{\sqrt{\frac{\hat{\sigma}_e^2(n+1)}{\nu L \sigma_x^2}}}, \qquad \lambda(n+1) = 4 * exp\left(\frac{-1}{\varsigma\,\hat{\sigma}_e^2(n+1)}\right) + 4$$

$$\xi\left(\hat{\boldsymbol{c}}(n)\right) = \frac{L}{L-\sqrt{L}}\left\{1 - \frac{\|\hat{\boldsymbol{c}}(n)\|_1}{\sqrt{L}\|\hat{\boldsymbol{c}}(n)\|_2}\right\}, \qquad n \geq L$$

$$\rho(n) = \begin{cases} 5/L & n < L \\ e^{-\lambda(n+1)\xi(\hat{\boldsymbol{c}}(n))} & n \geq L \end{cases}$$

$$F(|\hat{c}_l(n)|) = ln(1 + \beta(n+1)|\hat{c}_l(n)|), \qquad l = 0, 1, \ldots, L-1$$

$$\varphi(n) = \rho(n)\, max\{\gamma, F(|\hat{c}_0(n)|), F(|\hat{c}_1(n)|), .., F(|\hat{c}_{L-1}(n)|)\}$$

$$\theta_l(n) = max\{\varphi(n), F(|\hat{c}_l(n)|)\} \quad \forall l, \qquad l = 0, 1, \ldots L-1$$

$$g_l(n) = \frac{\theta_l(n)}{\frac{1}{L}\sum_{l=0}^{L-1}\theta_l(n)} \quad l = 0, 1, .. L-1$$

$$\boldsymbol{G}(n) = diag\{\overline{\boldsymbol{g}}(n)\}$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\boldsymbol{G}(n)\overline{\boldsymbol{x}}(n) + \delta_{ASCMPNLMS}}$$

Computational complexity

The computations required are the sum of those of the SCMPNLMS and the additional computations of the AMPNLMS when compared to the MPNLMS. $\lambda(n)$ can be computed using a look-up-table. Therefore ASCMPNLMS requires in total $5L + 3$ additions, $7L + 9$ multiplications, 3 divisions, $2L$ comparisons, and L logarithms per iteration.

B.2 Thresholded NLMS Algorithms

Table B.2: The TNLMS algorithm with gain-control factors.

Initialization:	$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$
Parameters:	$0 < \mu_{NLMS} < 2$
	$\delta_{NLMS} = cst.\sigma_x^2$
	$error\ ratio = 0.4$
	$thresh. = error\ ratio * \frac{\mu}{L}$
Gain-control:	$c_l(n) = \begin{Bmatrix} c_l(n) & \lvert c_l(n)\rvert > thresh. \\ 0 & otherwise \end{Bmatrix} \forall l, \quad l = 0 \ldots L-1$
Error:	$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$
Filter Update:	$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS}\frac{\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\overline{\boldsymbol{x}}(n) + \delta_{NLMS}}$

Table B.3: The mTNLMS algorithm with gain-control factors.

Initialization:	$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$
Parameters:	$0 < \mu_{NLMS} < 2$
	$\delta_{NLMS} = cst.\sigma_x^2$
	$error\ ratio = 0.4, \qquad \tau = 0.5$
	$thresh. = error\ ratio * \frac{\mu}{L}$
Gain-control:	$m_l(n) = \begin{Bmatrix} 1 & \lvert c_l(n)\rvert > thresh. \\ \tau & otherwise \end{Bmatrix} \forall l, \qquad l = 0 \ldots L-1$
	$\overline{\boldsymbol{x}}(n) = \overline{\boldsymbol{m}}(n) \circ \overline{\boldsymbol{x}}(n)$
	$c_l(n) = \begin{Bmatrix} c_l(n) & \lvert c_l(n)\rvert > thresh. \\ 0 & otherwise \end{Bmatrix} \forall l, \qquad l = 0 \ldots L-1$
Error:	$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$
Filter Update:	$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NLMS} \frac{\overline{\boldsymbol{x}}(n)e(n)}{\overline{\boldsymbol{x}}^T(n)\overline{\boldsymbol{x}}(n) + \delta_{NLMS}}$

B.3 Thresholded NPVSS-NLMS Algorithms

Table B.4: The NPVSS-TNLMS algorithm with gain-control factors.

Initialization:

$$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$$

$$\hat{\sigma}_e^2(0) = 0$$

Parameters:

$$0 < \mu_{NLMS} < 2$$

$$\delta_{NLMS} = cst.\sigma_x^2$$

$$\lambda = 1 - \frac{1}{KL}, K \geq 2$$

$$error\ ratio = 0.4$$

Gain-control:

$$\hat{\sigma}_e^2(n) = \lambda\, \hat{\sigma}_e^2(n-1) + (1-\lambda)\, e^2(n)$$

$$\beta(n) = [\delta_{NLMS} + \overline{\boldsymbol{x}}^T(n)\overline{\boldsymbol{x}}(n)]^{-1} \left[1 - \frac{\sigma_w}{\epsilon + \hat{\sigma}_e(n)}\right]$$

$$\mu_{NPVSS}(n) = \begin{Bmatrix} \beta(n) & if\ \hat{\sigma}_e(n) \geq \sigma_w \\ 0 & otherwise \end{Bmatrix}$$

$$thresh.(n) = error\ ratio * \frac{\mu_{NPVSS}(n-1)}{L}$$

$$c_l(n) = \begin{Bmatrix} c_l(n) & |c_l(n)| > thresh.(n) \\ 0 & otherwise \end{Bmatrix} \forall l, \qquad l = 0 \ldots L-1$$

Error:

$$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NPVSS}(n)\overline{\boldsymbol{x}}(n)e(n)$$

Table B.5: The NPVSS-mTNLMS algorithm with gain-control factors.

Initialization:

$$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$$

$$\hat{\sigma}_e^2(0) = 0$$

Parameters:

$$0 < \mu_{NLMS} < 2$$

$$\delta_{NLMS} = cst.\sigma_x^2$$

$$\lambda = 1 - \frac{1}{KL}, K \geq 2$$

$$error\ ratio = 0.4, \qquad \tau = 0.6$$

Gain-control:

$$\hat{\sigma}_e^2(n) = \lambda\, \hat{\sigma}_e^2(n-1) + (1-\lambda)\, e^2(n)$$

$$thresh.(n) = error\ ratio * \frac{\mu_{NPVSS}(n-1)}{L}$$

$$m_l(n) = \begin{Bmatrix} 1 & |c_l(n)| > thresh.(n) \\ \tau & otherwise \end{Bmatrix} \quad \forall l, \qquad l = 0 \ldots L-1$$

$$\overline{\boldsymbol{x}}(n) = \overline{\boldsymbol{m}}(n) \circ \overline{\boldsymbol{x}}(n)$$

$$\beta(n) = \left[\delta_{NLMS} + \overline{\boldsymbol{x}}^T(n)\overline{\boldsymbol{x}}(n)\right]^{-1} \left[1 - \frac{\sigma_w}{\epsilon + \hat{\sigma}_e(n)}\right]$$

$$\mu_{NPVSS}(n) = \begin{Bmatrix} \beta(n) & if\ \hat{\sigma}_e(n) \geq \sigma_w \\ 0 & otherwise \end{Bmatrix}$$

$$c_l(n) = \begin{Bmatrix} c_l(n) & |c_l(n)| > thresh.(n) \\ 0 & otherwise \end{Bmatrix} \forall l, \qquad l = 0 \ldots L-1$$

Error:

$$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NPVSS}(n)\overline{\boldsymbol{x}}(n)e(n)$$

B.4 Adaptive Thresholded NPVSS-NLMS Algorithms

Table B.6: The NPVSS-ATNLMS algorithm with gain-control factors.

Initialization:

$$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$$

$$\hat{\sigma}_e^2(0) = 0$$

Parameters:

$$0 < \mu_{NLMS} < 2$$

$$\delta_{NLMS} = cst.\sigma_x^2$$

$$\lambda = 1 - \frac{1}{KL}, K \geq 2, \qquad \tau = 0.7$$

$$error\ ratio(n) = \vartheta * \mu_{NPVSS}(n), \qquad \vartheta = 0.7$$

Gain-control:

$$\hat{\sigma}_e^2(n) = \lambda\, \hat{\sigma}_e^2(n-1) + (1-\lambda)\, e^2(n)$$

$$\beta(n) = [\delta_{NLMS} + \overline{\boldsymbol{x}}^T(n)\overline{\boldsymbol{x}}(n)]^{-1} \left[1 - \frac{\sigma_w}{\epsilon + \hat{\sigma}_e(n)}\right]$$

$$\mu_{NPVSS}(n) = \begin{Bmatrix} \beta(n) & if\ \hat{\sigma}_e(n) \geq \sigma_w \\ 0 & otherwise \end{Bmatrix}$$

$$thresh.(n) = error\ ratio * \frac{\mu_{NPVSS}(n)}{L}$$

$$c_l(n) = \begin{Bmatrix} c_l(n) & |c_l(n)| > thresh.(n) \\ 0 & otherwise \end{Bmatrix} \forall l, \qquad l = 0 \ldots L-1$$

Error:

$$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NPVSS}(n)\overline{\boldsymbol{x}}(n)e(n)$$

Table B.7: The NPVSS-mATNLMS algorithm with gain-control factors.

Initialization:

$$\hat{\boldsymbol{c}}(0) = \overline{\mathbf{0}}$$

$$\hat{\sigma}_e^2(0) = 0$$

Parameters:

$$0 < \mu_{NLMS} < 2$$

$$\delta_{NLMS} = cst.\sigma_x^2$$

$$\lambda = 1 - \frac{1}{KL}, K \geq 2$$

$$error\ ratio(n) = \vartheta * \mu_{NPVSS}(n), \quad \vartheta = 0.5$$

Gain-control:

$$\hat{\sigma}_e^2(n) = \lambda\, \hat{\sigma}_e^2(n-1) + (1-\lambda)\, e^2(n)$$

$$\beta(n) = \left[\delta_{NLMS} + \overline{\boldsymbol{x}}^T(n)\overline{\boldsymbol{x}}(n)\right]^{-1} \left[1 - \frac{\sigma_w}{\epsilon + \hat{\sigma}_e(n)}\right]$$

$$\mu_{NPVSS}(n) = \begin{Bmatrix} \beta(n) & if\ \hat{\sigma}_e(n) \geq \sigma_w \\ 0 & otherwise \end{Bmatrix}$$

$$thresh.(n) = error\ ratio * \frac{\mu_{NPVSS}(n)}{L}$$

$$m_l(n) = \begin{Bmatrix} 1 & |c_l(n)| > thresh.(n) \\ \tau & otherwise \end{Bmatrix} \quad \forall l, \qquad l = 0 \ldots L-1$$

$$\overline{\boldsymbol{x}}(n) = \overline{\boldsymbol{m}}(n) \circ \overline{\boldsymbol{x}}(n)$$

$$c_l(n) = \begin{Bmatrix} c_l(n) & |c_l(n)| > thresh.(n) \\ 0 & otherwise \end{Bmatrix} \forall l, \qquad l = 0 \ldots L-1$$

Error:

$$e(n) = d(n) - \hat{\boldsymbol{c}}^T(n)\overline{\boldsymbol{x}}(n)$$

Filter Update:

$$\hat{\boldsymbol{c}}(n+1) = \hat{\boldsymbol{c}}(n) + \mu_{NPVSS}(n)\overline{\boldsymbol{x}}(n)e(n)$$

Appendix C Variable Step-Size Algorithms Summary

Table C.1: Summary of the concept of Variable Step-Size algorithms.

#	Algorithm Name	Concept
1	Least-Mean-Squares (LMS)	A stochastic implementation of the steepest-descend algorithm [26,27], which searches for the optimal solution of the unknown system impulse response by iteratively minimizing the mean-square error
2	Normalized LMS (NLMS)	The step-size of the LMS is normalized with the power of the input signal for better performance with signals that have a large dynamic range such as speech.
3	Proportionate NLMS (PNLMS)	Each coefficient is updated independently of the other by assigning individual step-sizes and where each is adjusted in proportion to the magnitude of the estimated filter coefficient.
4	PNLMS plusplus (PNLMS++)	The coefficient update is alternated between NLMS and PNLMS. The PNLMS is used every k-iterations, where k is a positive integer.
5	Improved PNLMS (IPNLMS)	The individual step-size update rule combines the even distribution of the adaptation gain of the NLMS with a proportionate update term by adding them together.
6	Mu-law PNLMS (MPNLMS)	The individual step-sizes are updated in proportion to the mu-law function of the magnitude of the estimated filter coefficient. This update rule was derived such that all the coefficients attain a converged value within a ξ-vicinity of their optimal values (Wiener filter solution) after the same number of iterations, where ξ is a small positive number.
7	Segment PNLMS (SPNLMS)	Piecewise linear functions are used to approximate the logarithm function in the mu-law function used in the MPNLMS because it maybe computationally expensive.
8	Adaptive MPNLMS (AMPNLMS)	An estimate of the mean-square error is used to reduce the size of the neighborhood to which the coefficients converge within, i.e. reduce the ξ-vicinity of the coefficients' optimal values in the MPNLMS algorithm. This is achieved by varying the mu-law parameter of the MPNLMS.
9	Sparseness Controlled PNLMS	The sparsity of the echo path impulse responses is exploited and used to control the update of the individual step-sizes. This is achieved by using a measure of the sparseness of the current estimate of the adaptive filter

	(SCPNLMS)	coefficients to control the proportionality parameter of the PNLMS.
10	Sparseness Controlled MPNLMS (SCMPNLMS)	Similar to the SCPNLMS, a measure of the sparseness of the current estimate of the adaptive filter coefficients is used to control the proportionality parameter of the PNLMS.
11	Sparseness Controlled IPNLMS (SCIPNLMS)	A measure of the sparseness of the current estimate of the adaptive filter coefficients is employed in the individual step-size update rule of the IPNLMS to control the contribution of the proportionate and NLMS updates.
12	Generalized Gradient PNLMS (GGPNLMS)	The individual step-size update rule combines the NLMS update term with a proportionate update term that uses an estimate of the gradient of the filter coefficients' adaptation instead of their magnitudes, and adds the two terms together. This is meant to improve the tracking capability of the adaptive algorithm by depending on the time-varying component of the filter coefficient change rather than on the time-invariant component.
13	Adaptive SCMPNLMS (ASCMPNLMS)	The time-varying mu-law parameter of the MPNLMS and the time-varying proportionality parameter of the SCMPNLMS are combined with varying another constant parameter of the SCMPNLMS that is sensitive to the sparseness of the impulse response. The variation profile for this parameter is designed to provide good convergence performance for non-sparse impulse responses.
14	gradient induced PNLMSs (g-PNLMSs)	The individual step-size update rule is a weighted sum of a proportionate update term that uses an estimate of the gradient of the coefficients' adaptation (referred to as the gradient-gain) and a proportionate update term that uses the magnitude of the coefficients (referred to as the proportional-gain). The proportional-gain used could be any of the various PNLMS-based update rules leading to the various gradient-induced PNLMS algorithms.
15	Non-Parametric Variable Step-Size NLMS (NPVSS-NLMS)	A scalar variable step-size formula is derived such that the power of the a posteriori error approaches the power of the system noise. The a posteriori error is the error resulting from subtracting the output of the updated adaptive filter from the desired signal. The performance of the algorithm does not depend on tuning any of the parameters, therefore it is non-parametric.
16	Thresholded NLMS (TNLMS, NPVSS-TNLMS)	Coefficients that have actual magnitudes relatively smaller than their weight-update introduce a cumulative estimation error that degrades the overall tracking capability and increases the steady-state error. These relatively small taps, identified using a threshold value, are set to zero and their update is omitted. The threshold value is a fraction of the NLMS

		weight-update per tap governed by the step-size and the length of the filter. A variant algorithm is the NPVSS-TNLMS.
17	masked Thresholded NLMS (mTNLMS, NPVSS-mTNLMS)	A weighting mask vector is used to attenuate the contribution of the input signal at the inactive (or thresholded) tap locations. This effectively increases the normalized step-size for a fewer number of active taps since the step-size is normalized by a smaller input signal power. A variant algorithm is the NPVSS-mTNLMS.
18	Adaptive Thresholded NLMS (NPVSS-ATNLMS, NPVSS-mATNLMS)	The threshold value in the TNLMS is made time-varying by depending proportionally on an estimate of the mean-square error. When the filter is not converged and the MSE is large, the threshold value is large allowing a few taps with the largest magnitudes to converge initially at the fastest rate since the adaptation gain is distributed on few taps only. As the filter converges over time, the MSE and the threshold value decrease allowing an increasing number of taps, with the larger magnitudes first, to converge. The larger the coefficient magnitude is the earlier it starts converging and the higher the rate of convergence at onset. The rate of convergence of each tap decreases over time as more and more taps are taking shares in the adaptation gain. Variants of the ATNLMS include the masked Adaptive TNLMS (mATNLMS), NPVSS-ATNLMS, and NPVSS-mATNLMS.

REFERENCES

[1] J., Gänsler, T., Morgan, D.R., Sondhi, M.M., Gay, S.L. Benesty, *Advances in Network and Acoustic Echo Cancellation*. Berlin, Germany: Springer-Verlag, 2001.

[2] M. E. Knappe, "Acoustic Echo Cancellation: Performance and Structures," Carleton University, Ottawa, Canada, M. Eng. Thesis July 1992.

[3] R. Harris, D. Chabries, and F. Bishop, "A variable step (VS) adaptive filter algorithm," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, no. 2, pp. 309-316, Apr 1986.

[4] A. Sugiyama, M.N.S. Swamy, and E.I. Plotkin, "A fast convergence algorithm for adaptive FIR filters," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 1989*, vol. 2, Glasgow, Scotland, 23-26 May 1989, pp. 892-895.

[5] R.H. Kwong and E.W. Johnston, "A variable step size LMS algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 7, pp. 1633-1642, Jul 1992.

[6] V.J. Mathews and Z. Xie, "A stochastic gradient adaptive filter with gradient adaptive step size," *IEEE Transactions on Signal Processing*, vol. 41, no. 6, pp. 2075-2087, Jun 1993.

[7] A. Sugiyama, "An interference-robust stochastic gradient algorithm with a gradient-adaptive step-size," in *Proc. of the IEEE International Conference on Acoustics,*

Speech, and Signal Processing, ICASSP 1993, vol. 3, Minneapolis, MN, USA, 27-30 April 1993, pp. 539-542.

[8] T. Aboulnasr and K. Mayyas, "A robust variable step-size LMS-type algorithm: analysis and simulations," *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 631-639, Mar 1997.

[9] A. Mader, H. Puder, and G. U. Schmidt, "Step-size control for acoustic echo cancellation filters—an overview," *Signal Processing*, vol. 80, no. 9, pp. 1697-1719, Sep. 2000.

[10] Hyun-Chool Shin, A.H. Sayed, and Woo-Jin Song, "Variable step-size NLMS and affine projection algorithms," *IEEE Signal Processing Letters*, vol. 11, no. 2, pp. 132-135, Feb. 2004.

[11] J. Benesty, H. Rey, L.R. Vega, and S. Tressens, "A Nonparametric VSS NLMS Algorithm," *IEEE Signal Processing Letters*, vol. 13, no. 10, pp. 581-584, Oct. 2006.

[12] S. Makino, Y. Kaneda, and N. Koizumi, "Exponentially weighted stepsize NLMS adaptive filter based on the statistics of a room impulse response," *IEEE Transactions on Speech and Audio Porcessing*, vol. 1, no. 1, pp. 101-108, Jan 1993.

[13] H. Yuan, *Dynamic Behaviour of Acoustic Echo Cancellation.* Ottawa, Canada: M.Eng Thesis, Carleton University, 1994.

[14] D.L. Duttweiler, "Proportionate normalized least-mean-squares adaptation in echo cancelers," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 5, pp.

508-518, Sep. 2000.

[15] S.L. Gay, "An efficient, fast converging adaptive filter for network echo cancellation," in *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers*, Pacific Grove, California, 1-4 Nov. 1998, pp. 394-398.

[16] Jacob Benesty and Steven L. Gay, "An improved PNLMS algorithm," *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, pp. II-1881-II-1884, 13-17 May 2002.

[17] H. Deng and M. Doroslovacki, "Improving convergence of the PNLMS algorithm for sparse impulse response identification," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 181- 184, Mar. 2005.

[18] H. Deng and M. Doroslovacki, "Proportionate adaptive algorithms for network echo cancellation," *IEEE Transactions on Signal Processing*, vol. 54, no. 5, pp. 1794-1803, May 2006.

[19] K.T. Wagner and M.I. Doroslovacki, "Gain allocation in proportionate-type NLMS algorithms for fast decay of output error at all times," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2009*, Taipei, Taiwan, 19-24 April 2009, pp. 3117-3120.

[20] P. Loganathan, A.W.H. Khong, and P.A. Naylor, "A Class of Sparseness-Controlled Algorithms for Echo Cancellation," *IEEE Transactions on Audio, Speech, and*

Language Processing, vol. 17, no. 8, pp. 1591-1601, Nov. 2009.

[21] O. Hoshuyama, R.A. Goubran, and A. Sugiyama, "A generalized proportionate variable step-size algorithm for fast changing acoustic environments," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. (ICASSP '04)*, vol. 4, pp. iv-161- iv-164 , 17-21 May 2004.

[22] F. das Chagas de Souza, O.J. Tobias, R. Seara, and D.R. Morgan, "A PNLMS Algorithm With Individual Activation Factors," *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2036-2047, April 2010.

[23] M. Nekuii and M. Atarodi, "A fast converging algorithm for network echo cancellation," *IEEE Signal Processing Letters*, vol. 11, no. 4, pp. 427- 430, April 2004.

[24] Alan V. Oppenheim and Ronald W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[25] W. Kenneth Jenkins, Andrew W. Hull, Jeffrey C. Strait, Bernard A. Schnaufer, and Xiaohui Li, *Advanced Concepts in Adaptive Signal Processing*. Boston: Kluwer Academic Publishers, 1996.

[26] S.Haykin, *Adaptive Filter Theory*, 4th ed. Upper Saddle River, New Jersy: Prentice Hall, 2002.

[27] A. H. Sayed, *Fundametals of Adaptive Filtering*.: John Willey & Sons, Inc., 2003.

[28] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electron. Commun. Jpn.*, vol. 67-A, no. 5, pp. 19-27, May 1984.

[29] Fan Hong and W.K. Jenkins, "An investigation of an adaptive IIR echo canceller: advantages and problems," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 12, pp. 1819-1834, Dec 1988.

[30] J.J. Shynk, "Adaptive IIR filtering," *IEEE Transactions Acoustics, Speech, Signal Processing*, vol. 6, no. 2, pp. 4-21, April 1989.

[31] J.J. Shynk, "Frequency-domain and multirate adaptive filtering," *IEEE Signal Processing Magazine*, vol. 9, no. 1, pp. 14-37, Jan. 1992.

[32] B.N.M. Laska, R.A. Goubran, and M. Bolic, "Improved Proportionate Subband NLMS for Acoustic Echo Cancellation in Changing Environments," *IEEE Signal Processing Letters*, vol. 15, pp. 337-340, 2008.

[33] M. De Courville and P. Duhamel, "Adaptive filtering in subbands using a weighted criterion," *IEEE Transactions on Signal Processing*, vol. 46, no. 9, pp. 2359-2371, Sep. 1998.

[34] B. Friedlander, "Lattice filters for adaptive processing," *IEEE Proceedings* , vol. 70, no. 8, pp. 829- 867, Aug. 1982.

[35] S. Kawamura and M.Hatori, "A Tap Selection Algorithm for Adaptive Filters," in *Proceedings of IEEE International Conf. on Acoustics, Speech, and Signal*

Processing, Tokya, Japan, 1986, pp. 2979-2982.

[36] K. Dogancay and O. Tanrikulu, "Adaptive filtering algorithms with selective partial update," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 8, pp. 762-769, Aug. 2001.

[37] T. Aboulnasr and K. Mayyas, "Selective coefficient update of gradient-based adaptive algorithms ," *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-97*, vol. 3, pp. 1929-1932, 21-24 April 1997.

[38] J.D. Gordy, T. Aboulnasr, and M. Bouchard, "Reduced-complexity proportionate NLMS employing block-based selective coefficient updates," *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2008*, pp. 233-236, 2008.

[39] S.C. Douglas, "Adaptive filters employing partial updates," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 3, pp. 209-216, March 1997.

[40] T. Schertler, "Selective block update of NLMS type algorithms," *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. 1717-1720, 12-15 May 1998.

[41] D.I. Pazaitis and A.G. Constantinides, "A novel kurtosis driven variable step-size adaptive algorithm," *IEEE Transactions on Signal Processing*, vol. 47, no. 3, pp. 864-872, Mar 1999.

[42] J.D. Gordy and R.A. Goubran, "Fast system identification using affine projection and a critically sampled subband adaptive filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 4, pp. 1242-1249, Aug. 2006.

[43] T.G. Burton, R.A. Goubran, and F. Beaucoup, "Nonlinear System Identification Using a Subband Adaptive Volterra Filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 5, pp. 1389-1397, May 2009.

[44] L. Couch, *Digital and Analog Communication Systems*, 5th ed. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[45] A.W.H. Khong and P.A. Naylor, "Efficient Use Of Sparse Adaptive Filters," in *Proc. of the Fortieth Asilomar Conference on Signals, Systems and Computers, 2006. ACSSC '06.* , Pacific Grove, California, Oct. 29 2006-Nov. 1 2006, pp. 1375-1379.

[46] J. Benesty, Y. A. Huang, J. Chen, and P. A. Naylor, "Adaptive algorithms for the identification of sparse impulse responses," in *Selected Methods for Acoustic Echo and Noise Control, and Speech Processing*, E. Hänsler and G. Schmidt, Eds. New York: Springer, 2006, ch. 5, pp. 125-153.

[47] J.B. Evans, P. Xue, and B. Liu, "Analysis and implementation of variable step size adaptive algorithms," *IEEE Transactions on Signal Processing*, vol. 41, no. 8, pp. 2517- 2535, Aug 1993.

[48] C. Paleologu, S. Ciochina, and J. Benesty, "Variable Step-Size NLMS Algorithm for

Under-Modeling Acoustic Echo Cancellation," *IEEE Signal Processing Letters*, vol. 15, pp. 5-8, 2008.

[49] L. Hawker, J. Nobels C. E. Ranta, "Handheld electronic device having hidden sound openings offset from an audio source," US Patent 7,714,838 B2, Issue Date: May 11, 2010.

[50] H. Deng and M. Doroslovacki, "Wavelet-based MPNLMS adaptive algorithm for network echo cancellation," *EURASIP Journal on Audio, Speech, and Music Processing*, 2007.

[51] H. Deng and M. Doroslovacki, "Proportionate adaptive algorithms for network echo cancellation," *IEEE Transactions onSignal Processing*, vol. 54, no. 5, pp. 1794-1803, May 2006.