Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2007

Project Title:	**Adaptive Echo Cancellation**
Student:	**Pradeep Loganathan**
Course:	**ISE4**
Project Supervisor:	**Dr Patrick Naylor**
Second Marker:	**Dr Tania Stathaki**

Abstract

The effective removal of hybrid and acoustic echo, inherent within the telecommunication infrastructure, is the key to improving perceived voice quality standard. This project investigates adaptive filters and their use in system identification, specifically for the application of echo cancellation. The overall aim is to revoke echo on realistic test data exploiting particular algorithms to examine their properties and compare their effectiveness.

This thesis includes the implementations of two particular well-known approaches, the Normalised Least-Mean-Square (NLMS) and sparse adaptive filter methods such as proportionate-NLMS (PNLMS), improved-PNLMS and μ-PNLMS. Simulation studies are undertaken and subjective performance measures, including mean squared error (MSE), echo return loss enhancement (ERLE) and normalised projection misalignment (NPM), are applied to confirm the potential of the methods. Further analyses are made to look at their computational complexity and robustness to sparsely impulse response.

To improve convergence speed across different sparseness levels, three different approaches are proposed, namely Optimum-NLMS, Sparse-controlled-MPNLMS-1 and Sparse-controlled-MPNLMS-2.

Acknowledgement

I express my sincere gratitude to my project supervisor Dr. Patrick Naylor for his valuable guidance and support throughout the entire project. I appreciate his tolerance and thanks to him for his time spent on advising me and evaluating my work.

I am also indebted to Dr. Andy Khong for being my well-wisher and thank him for his constructive remarks and unflinching support throughout this course.

Finally, to make this acknowledgement complete, a special thanks to my parents and brothers for their unconditional love and support, this added an essence to my work.

Contents

Abstract 2

Acknowledgements 3

Contents 4

Abbreviations 7

Notations 8

CHAPTER - 1

1. INTRODUCTION

1.1 Overview…..…………………………………..…………. 9

1.2 Project Objectives…..…………………………………..……… 10

1.3 Thesis Organisation ....……………………………….... 10

CHAPTER – 2

2. ECHO CANCELLATION

2.1 Introduction…..…………………………………..……………..…. 12

2.2 Types of Echo…..……………………………………………..…... 12

2.3 Different Technologies in Echo Cancellation…..………………...…. 14

2.4 Acoustic Impulse response….……………………..…..…………….. 14

2.5 Adaptive Echo Cancellation Process…...………………………….. 16

2.6 Summary…..…………………………………..……………..……. 18

CHAPTER – 3

3. FUNDAMENTALS OF ADAPTIVE FILTERING

3.1 Introduction…..…………………………………..……………..… 19

3.2 Wiener Filters....................……………………………...……. 19

3.3 Mean-Square Error Surface……………………..……………..…. 22

3.4 Steepest-Descent Algorithm……………………..……………..… 23

3.5 Applications of Adaptive filters……………………….......…….25
3.6 Summary…..…………………………………..…………..……27

CHAPTER – 4

4. REVIEW OF EXISTING ADAPTIVE ALGORITHMS
4.1 Introduction…..…………………………………..…………….. 28
4.2 Gradient Based Algorithms………..……………….…….……….. 28
4.2.1 Least Mean Square……………………..………..…..... 29
4.2.2 Normalised Least Mean Square ……………………….….. 30
4.3 Sparse Algorithms…..……………………………………..…...…… 31
4.3.1 Sparseness Measure ………………………………………...32
4.3.2 Proportionate Normalised Least Mean Square……………...32
4.3.3 Improved Proportionate Normalised Least Mean Square …..33
4.3.4 μ- Proportionate Normalised Least Mean Square ……..…33
4.4 Performance Measures…..…………………………………..…...34
4.4.1 Mean Square Error…..……………………………………...34
4.4.2 Echo Return Lossless Enhancement…..……………………34
4.4.3 Normalised Projection Misalignment…..…………………..34
4.5 Computational Complexity…..……………………………………..35
4.6 Simulation Results…..……………………………………..…........37
4.6.1 Experiment Set-up…..………………………………..….…37
4.6.2 NLMS Performance…..…………………………….....…...41
4.6.3 PNLMS Performance…..……………………….…..…...44
4.6.4 IPNLMS Performance…..…………………..…..…………45
4.6.5 MPNLMS Performance…..………………………….…..… 46
4.7 Comparisons and Discussions…..………………………………..…47
4.8 Summary…..………………………………..…..….………49

CHAPTER – 5

5. NOVEL ALGORITHMS FOR SPARSE SYSTEMS
5.1 Introduction…..…………………………………………………. 50
5.2 Novel Algorithms…..…………………………………………… 50
5.2.1 Optimum-NLMS…..……………………………….……..… 50

5.2.2 Sparse Controlled MPNLMS 1…..………………….……… 53

5.2.3 Sparse Controlled MPNLMS 2…..………………………. 54

5.3 Computational Complexity…..…………………….………..… 55

5.4 Comparisons and Discussions…..………………….……..… 56

5.5 Summary…..……………………..……………………………. 57

CHAPTER – 6

6. CONCLUSION

6.1 Conclusion…………………………………………………..… 58

6.2 Future Work…..…………………………………..……...…… 59

REFERENCES …..……………………………..……..……….………. 61

APPENDIX

1. MATLAB Simulation Codes…..………………….…………..… 63

2. Coloured Signal Generation…..………………..……………..… 71

Abbreviations

AEC:	Acoustic echo cancellation
ERLE:	Echo return loss enhancement
FIR:	Finite impulse response
IIR:	Infinite impulse response
IPNLMS:	Improved Proportionate Normalised Least-mean-square
LMS:	Least-mean-square
MPNLMS:	μ- Law Proportionate Normalised Least-mean-square
MSE:	Mean Square Error
NLMS:	Normalised Least-mean-square
NPM:	Normalised Projection Misalignment
ONLMS:	Optimum Normalised Least-mean-square
PNLMS:	Proportionate Normalised Least-mean-square
PSTN:	Public switched telephone network
SC-MPNLMS:	Sparse Controlled μ- law Proportionate NLMS
SNR:	Signal-to-noise ratio
WGN:	White Gaussian noise

Notations

$[.]^T$	Matrix transpose operator
$E\{.\}$	Expectation operator
$\|.\|_2^2$	Squared *l*2-norm
n	Sample iteration
L	Length of adaptive filter
L_T	Length of transmission room impulse responses
L_R	Length of receiving room impulse responses
$s_T(n)$	Transmission room source
$s_R(n)$	Receiving room source
$y(n)$	Receiving room microphone signal
$\hat{y}(n)$	Adaptive filter output
$e(n)$	*a priori* error
$w(n)$	Uncorrelated measurement noise
$\mathbf{Q}(n)$	Diagonal tap selection control matrix
$J(n)$	Cost function
$h_i(n)$	ith channel near-end (receiving room) impulse response
$\hat{h}_i(n)$	ith channel estimated response
$\mathrm{x}_i(n)$	ith channel tap-input vector
$\mathbf{R}$	Input autocorrelation matrix
σ_{x^2}	Variance of zero mean input signal
δ	Regularization parameter
μ	Step-size parameter
ξ	Control parameter for non-stationary unknown impulse response

CHAPTER 1

Introduction

1.1 Overview

Digital Signal processing (DSP) is one of the fastest growing fields which can be found virtually in almost every engineering and scientific discipline, ranging from talking toys and music players to the challenges of next generation telecommunication infrastructure including 3G and wireless. Incorporating digital technology into the analogue world in which we live has increased the demand of DSP. This integration is accelerated by the flexibility and the usability of adaptive systems via digital manipulation.

Adaptive filtering has been extensively used in the context of many fields, including telecommunication, geophysical signal processing, biomedical signal processing, the elimination of radar clutter and sonar processing. In the field of digital communications, adaptive filtering approaches are very important and have been applied in many applications such as noise cancelling, adaptive beam forming and blind equalisation.

A specific application of noise cancelling, which is becoming prevalent in telecommunications, is echo cancellation. Several adaptive filtering algorithms have been proposed and investigated to overcome the echo problems in hands-free mobile and communication networks. For digital voice communications, such as distance learning education via video-conferencing, not only is accurate data transmission essential, superior audio quality in these multimedia functions is also required.

Therefore, in order to let users hear excellent sound performance and ensure call clarity, there is a great need to initially remove echoes.

1.2 Project Objectives

The aim of this project is to gain greater understanding into adaptive filters and their use in system identification, particularly for the application of echo cancellation. The overall aspire is to cancel echo on pragmatic test data employing particular algorithms to examine their properties and compare their effectiveness using simulations. However, the specific scope of this project is to implement and test using, two particular well-known approaches, the Normalised Least-Mean-Square (NLMS) and Sparse adaptive filter methods in order to inspect their computational complexity and speed of convergence to a stable error difference state within the MATLAB environment.

This study finally seeks to present three different novel adaptive algorithms for acoustic echo cancellation.

1.3 Thesis Organisation

The six chapters of this thesis are organised as follows:

Chapter 2 explains the different types of echo exists in telephony technology and describes the well known technologies that have been attempted up to the current generation.

Chapter 3 gives a detailed theoretical analysis of adaptive filtering. In addition Wiener filter is presented, establishing the optimum linear filter that can be sought in stationary environments. The concept of mean-square error surface is then discussed to analyse adaptive filters. It further introduces the classical steepest descent algorithm and finally a brief preamble of the four basic classes of adaptive filtering applications.

Chapter 4 reviews the main existing stochastic gradient based algorithms for sparse impulse responses. Different subjective performance measures are defined in order to compare each algorithm’s accuracy in predicting the unknown system and convergence speed. Computational complexities of those algorithms are considered for efficient implementation. Series of experiments are carried out to identify their performances within and across all the algorithms, by defining a realistic test data for single channel stereophonic AEC.

Chapter 5 proposes three different novel algorithms, which robust to sparse impulse response. The additional Computational complexity for each of them is reviewed and ultimately their performances are compared against the published algorithms.

Chapter 6 concludes remarks and recommends possible future work.

Appendix includes MATLAB codes for all the developed algorithms and for their experiments. It also includes a possible way to generate coloured signal in order to test the algorithms.

CHAPTER 2

Echo Cancellation

2.1 Introduction

Wireless phones are increasingly being regarded as essential communications tools due to their flexibility. As the use for the in-car hands free telephony has gained much popularity in recent years due to the rise in safety concerns and the need for supplying services to a user by an automated service delivery system, digital wireless subscribers are becoming ever more critical of the voice quality they receive from network providers. One factor that affects the voice quality across a wireless network is echo.

In telephone conversation, an echo is said to occur if the speaker hears repetition of his sound through the earpiece of his handset. This delayed replica is only noticeable if the amplitude of the echo is high or the time delay between the speech and the echo exceeds 35ms [1]. The study carried out at Bell laboratories found that echoes above 250ms can make it impossible to have natural conversation.

This chapter briefly discusses the different types of echoes that arise in the telephone communication, in order to describe the nature of the problem. The study also extends to the techniques that have been employed in the history to remove echoes from the channel.

2.2 Types of Echo

Acoustic echo is a type of echo which is produced by poor voice coupling between the earpiece and microphone in handsets and hands-free devices.

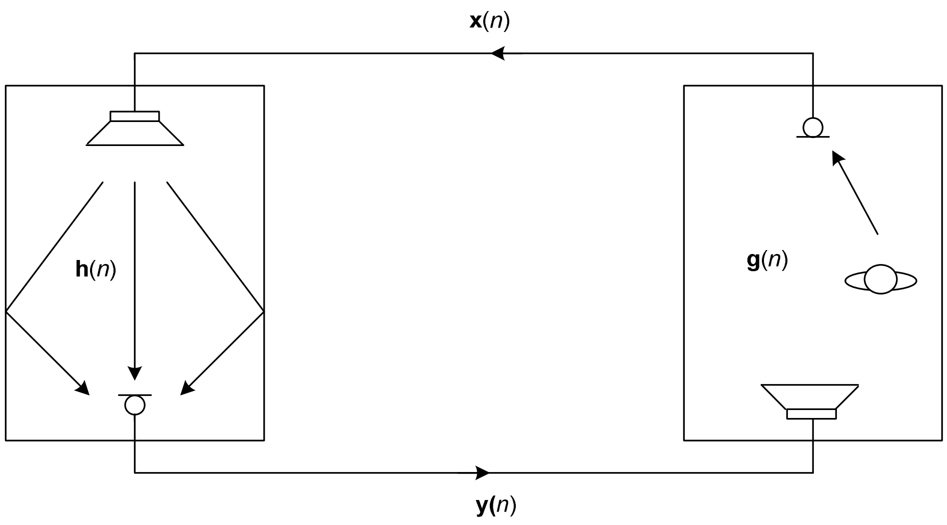


Figure 2.1: Illustration of acoustic echo

As shown in figure 2.1, sound signal, **x**(*n*), from a loudspeaker is heard by a listener, as intended. However, this same sound also is picked up by the microphone, both directly and indirectly, after bouncing off the wall. The result of this reflection is the creation of echo which is transmitted back to the far end and is heard by the talker as echo.

Hybrid echo is the other type of echo generated in the public-switched telephone network (PSTN) due to the impedance mismatch in the hybrid transformers. As illustrated in figure 2.2, when voice signals pass from the four-wire to the two-wire portion of the network, the energy in the four-wire section is reflected back to the speaker and create the echoed speech.

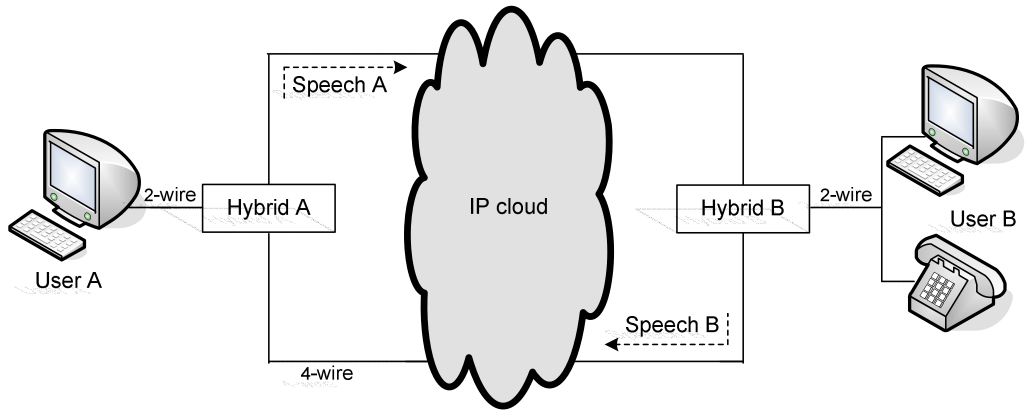


Figure 2.2: Illustration of hybrid echo

Due to the extent of those subjects, this project concerns only the acoustic echoes.

2.3 Technologies in Echo Cancellation [3]

Subscribers use speech quality as the benchmark for assessing the overall quality of a network. In the case of automated service delivery system, echoes can seriously degrade the performance of such systems. For this reason, the effective removal of echo inherent within the digital cellular infrastructure is the key to maintaining and improving perceived voice quality on a call.

In late 1950s, the emergence of echo suppression device facilitated the elimination of echoes generated in satellite communication. They were voice-activated switches that transmitted a voice path and then turned off to block any echo signal. The main issue with these gadgets is that they eliminated double-talk capabilities (parties at both ends can't speak simultaneously).

Nowadays, adaptive echo cancellers are used mostly in hands-free telephones where a loudspeaker and a microphone can be placed at a significant distance from each other. This approach utilized high-speed digital signal-processing (DSP) techniques to model and subtracts the echo from the return path and therefore, outperformed the suppression-based technique.

2.4 Acoustic Impulse Response

When a sound is generated in a room, the listener will first hear the sound via the direct path from the source. Shortly after, the listener will hear the reflections of the sound off the walls which will be attenuated, as shown in figure 2.3.

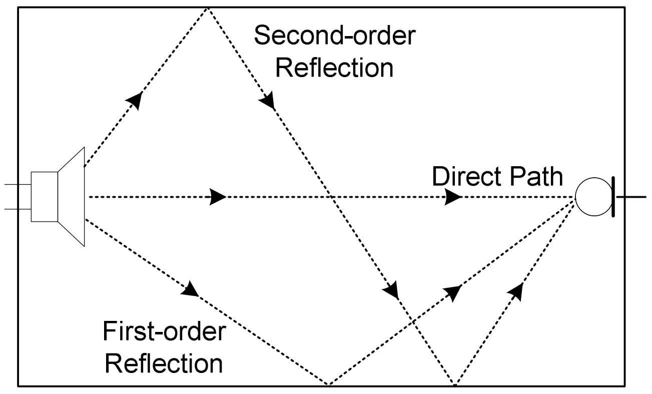


Figure 2.3: A visual example of how sound propagates through a room

Each reflection will then in turn be further delayed and attenuated as the sound is reflected again and again off the walls. Further examination of the impulse response of a room yields the observation that the sound decays at an exponential rate [3]. Therefore, the impulse response of the room shown above may be similar to figure 2.4.

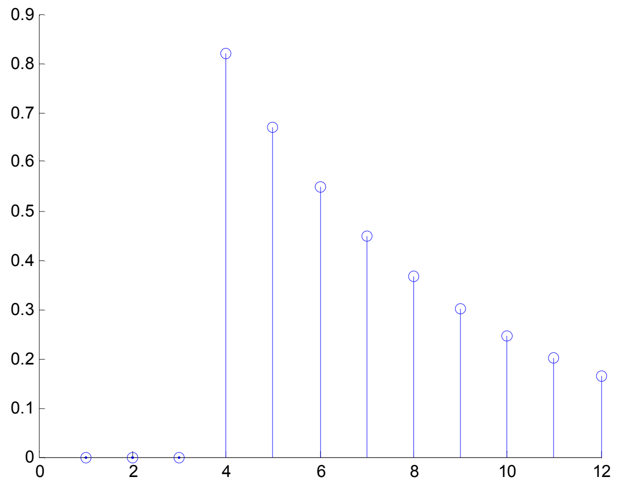


Figure 2.4: Impulse response of the room shown in figure 2.3

The echoes effects can be reduced by having absorbers around the wall. In the case, the impulse response has less active coefficients, as depicted in figure 2.5. The latter impulse response is said to be more sparse system than the former, due to the majority of its filter taps are inactive.

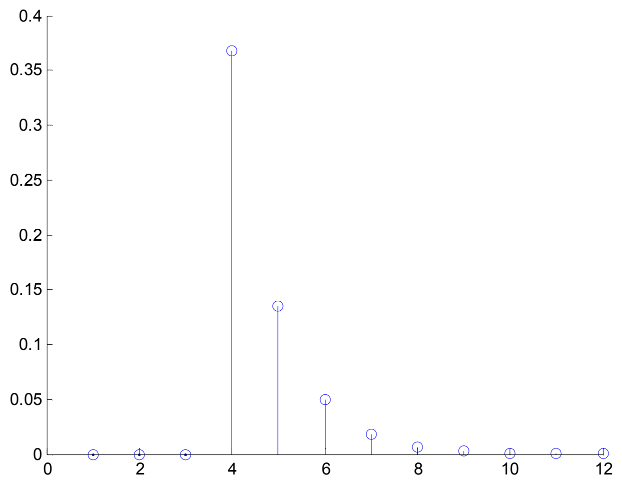


Figure 2.5: Sparse impulse response of the room in the presence of echo absorbers.

2.4 The Adaptive Echo Cancellation Process

Figure 2.6 shows an acoustic echo canceller set-up by employing an adaptive filter.

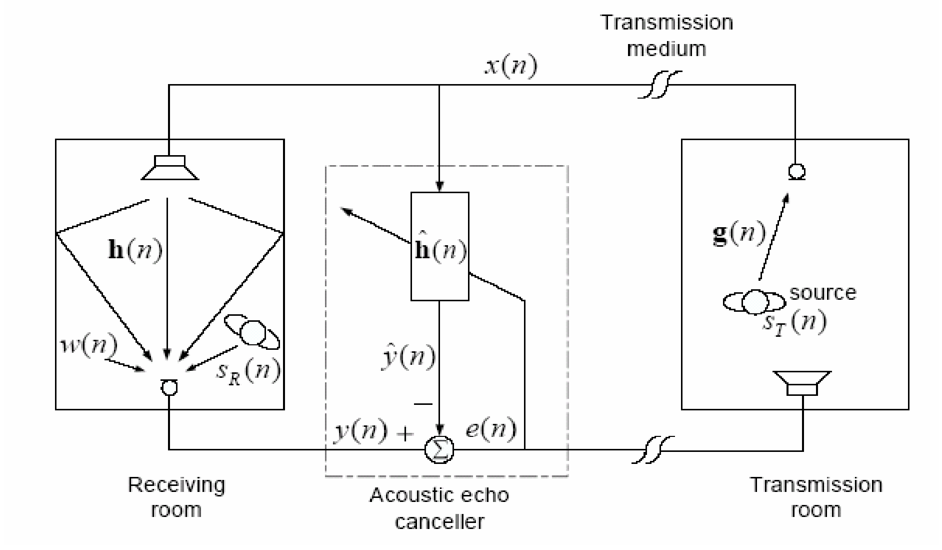


Figure 2.6: Single channel echo cancellation [2]

In this report, $[\ ]^T$ denotes matrix transpose and $E[\ ]$ signifies mathematical expectation operator. Scalars are also indicated in plain lowercase, vectors in bold lowercase and matrices in bold uppercase.

Notations and definitions:

$\mathbf{g}(n)$ = impulse response of transmission room

$=[g_0(n)\ g_1(n)\ ...\ g_{L_t-1}(n)]^{\mathrm{T}}$ where L_t is the length of $\mathbf{g}(n)$

$\mathbf{h}(n)$ = impulse response of receiving room

$= [h_0(n)\ h_1(n)\ ...\ h_{L_r-1}(n)]^{\mathrm{T}}$ where L_r is the length of $\mathbf{h}(n)$

$\hat{\mathbf{h}}(n)$ = impulse response of the adaptive filter

$=[\hat{h}_0(n)\ \hat{h}_1(n)\ ...\ \hat{h}_{L-1}(n)]^{\mathrm{T}}$ where L is the length of $\hat{\mathbf{h}}(n)$

$\mathbf{x}(n)$ = input signal to the adaptive filter and the receiving room system

$=[x(n)\ x(n\text{-}1)\ ...\ x(n\text{-L+1})]^{\mathrm{T}}$

$s_T(n)$ = transmission room source signal

$s_R(n)$ = receiving room source signal

$w(n)$ = noise signal in receiving room

The receiver attached in the transmission room (right hand side in figure 2.3) picks up the a time varying signal $\mathbf{x}(n)$ from a speech source $s_T(n)$ (far-end speaker) via impulse response of the transmission room $\mathbf{g}(n)$. The input signal $\mathbf{x}(n)$ is then transmitted to the loudspeaker in the near-end receiving room. The receiving room's microphone receives the desired signal $y(n)$ which is the convoluted sum of the input signal and the impulse response of the receiving room $\mathbf{h}(n)$ along with near-end speech signal and some additive noise. Therefore,

$$y(n) = \mathbf{h}^{\mathrm{T}}(n)\mathbf{x}(n) + w(n) + \mathrm{s}_{\mathrm{R}}(n) \tag{2.1}$$

In absence of echo canceller, the received signal $y(n)$ will be transmitted back to the origin with some delay. In the presence of an adaptive echo canceller, its objective is to estimate $\mathbf{h}(n)$ by taking into account the error signal $e(n)$ at each iteration, where the $e(n)$ is defined as

$$\begin{aligned} e(n) &= \text{Output from the receiving room system - Output from the adaptive filter} \\ &= y(n) - \hat{y}(n) \qquad (2.2) \\ &= [\mathbf{h}^{\mathrm{T}}(n) - \hat{\mathbf{h}}^{\mathrm{T}}(n)]\,\mathbf{x}(n) + w\,(n) + \mathrm{s}_{\mathrm{R}}(n) \end{aligned}$$

In order to simplify the mathematical derivations of algorithms without loss of generality the following assumptions are made throughout this project:

- The length of $\mathbf{h}\,(n)$, L_{r} is same as the length of $\hat{\mathbf{h}}\,(n)$, L. In a reality, the length of the adaptive filter is less than the receiving room impulse responses. This is due to the fact that the computational complexity of an adaptive algorithm increases monotonically with the length of the adaptive filter. Therefore, L must be long enough to achieve a low system mismatch and computational complexity.
- There is no noise signal in the receiving room, $w\,(n) = 0$
- There is no source signal in the receiving room, $s_R\,(n) = 0$, i.e., no double-talk is present.
- A transversal finite impulse response (FIR) filter configuration is used, as shown in figure 2.7, due to its stability characteristics.

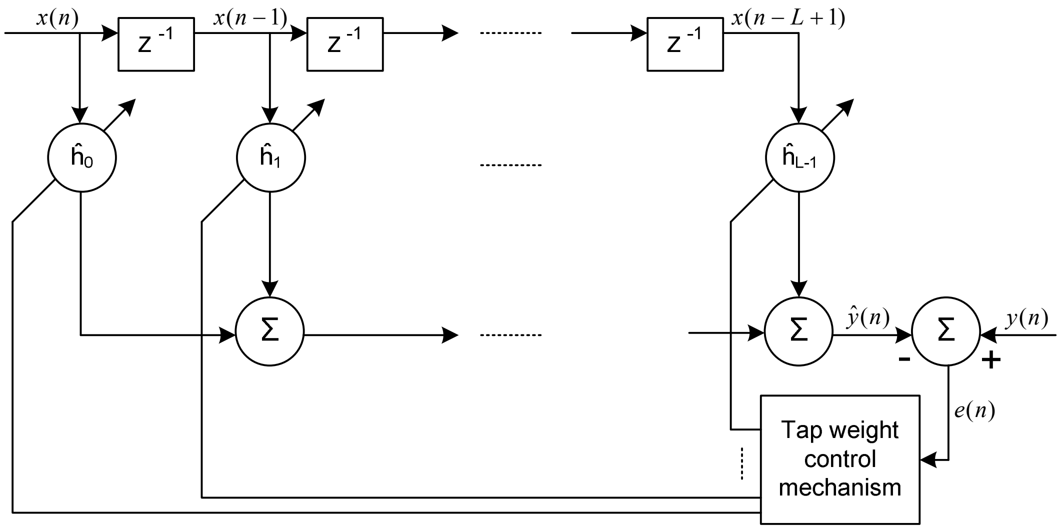


Figure 2.7: Adaptive transversal FIR filter

For effective echo cancellation, $e(n)$ must be significantly smaller in each iteration, as the filter coefficients converges to the unknown true impulse response $\mathbf{h}(n)$. Several adaptive algorithms are available for the weighs update and they generally exchange increased complexity for improved performance.

2.5 Summary

Echo cancellers can be potentially employed in telecommunication systems so that the undesired echoes, both acoustic and hybrid, can be diminished. The formation of single channel acoustic echo canceller has been looked at in some detail, where the functioning of the adaptive filter has been studied as a black box. The next chapter explains the theory behind the updates process of filter coefficients.

CHAPTER 3

Fundamentals of Adaptive Filtering

3.1 Introduction

This chapter includes a review of the concepts that are directly relevant to adaptive filtering, in order to find the true impulse response of the receiving room. It starts by studying Wiener filters, the way to get the optimum solution, and then addressed the potential issues involved in the process. The Steepest decent algorithm, the recursive way to reach a sub-optimum solution is also studied by having knowledge in tracing their performance. Finally, the four classes of adaptive filters are discussed using block diagrams.

3.2 Wiener Filters [5]

Filters are normally designed using frequency domain concepts, but Wiener filters are developed using time-domain concepts, by realising the simplified system as shown in figure 3.1.

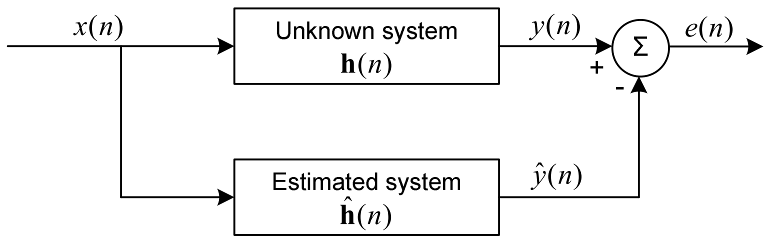


Figure 3.1: Linear conceptual system for system estimation

The quality of filter estimation is a function of the error signal, as shown by (2.2), which is the difference between the true system and the estimated system. This function can be considered as a cost incurred when the estimation is incorrect.

Therefore, a best choice for cost function is the mean-square error (MSE), since it is always positive and non-decreasing with acquired error.

$$\mathcal{J}(n) = E\{e^2(n)\} \tag{3.1}$$

Thus the optimum filter is defined as the filter of the set of all possible linear filters which minimises the MSE. Assuming that $e^2(n)$, $\mathbf{x}(n)$ and $y(n)$ are statistically stationary, the definition of the $e(n)$ can be substituted into (3.1) to yield an expression for the MSE cost function

$$\begin{aligned}
\mathcal{J}(n) &= E\{e^2(n)\} \\
&= E\{[y(n) - \hat{y}(n)][y(n) - \hat{y}(n)]\} \\
&= E\{[y(n) - \hat{\mathbf{h}}^{\mathrm{T}}(n)\mathbf{x}(n)][y(n) - \mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{h}}(n)]\} \\
&= E\{y^2(n) - y(n)\hat{\mathbf{h}}^{\mathrm{T}}(n)\mathbf{x}(n) - y(n)\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{h}}(n) + \hat{\mathbf{h}}^{\mathrm{T}}(n)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{h}}(n)\} \\
&= E\{y^2(n)\} - E\{y(n)\hat{\mathbf{h}}^{\mathrm{T}}(n)\mathbf{x}(n)\} - E\{y(n)\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{h}}(n)\} + E\{\hat{\mathbf{h}}^{\mathrm{T}}(n)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{h}}(n)\}
\end{aligned}$$

Since the $\hat{\mathbf{h}}(n)$ is not a random variable as the filter weights are not adjusted in this discussion, the cost function can be written as:

$$\begin{aligned}
\mathcal{J}(n) &= E\{y^2(n)\} - \hat{\mathbf{h}}^{\mathrm{T}}E\{y(n)\mathbf{x}(n)\} - E\{y(n)\mathbf{x}^{\mathrm{T}}(n)\}\hat{\mathbf{h}} + \hat{\mathbf{h}}^{\mathrm{T}}E\{\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\}\hat{\mathbf{h}} \\
&= E\{y^2(n)\} - 2\mathbf{p}^{\mathrm{T}}\hat{\mathbf{h}} + \hat{\mathbf{h}}^{\mathrm{T}}\mathbf{R}\hat{\mathbf{h}}
\end{aligned} \tag{3.2}$$

where $\mathbf{p}$ is the L-by-1 cross-correlation vector between the input signal and the desired signal defined as follows:

$$\mathbf{p} = E\{y(n)\mathbf{x}(n)\} = E\begin{bmatrix} y(n)x(n) \\ y(n)x(n-1) \\ \vdots \\ y(n)x(n-L+1) \end{bmatrix}$$

and $\mathbf{R}$ is the L-by-L auto-correlation matrix of the tap inputs in the transversal filter and can be defined as:

$$\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\} = E\begin{bmatrix} x(n)x(n) & x(n)x(n-1) & \cdots & x(n)x(n-L+1) \\ x(n-1)x(n) & x(n-1)x(n-1) & \cdots & x(n-1)x(n-L+1) \\ \vdots & \vdots & \ddots & \vdots \\ x(n-L+1)x(n) & x(n-L+1)x(n-1) & \cdots & x(n-L+1)x(n-L+1) \end{bmatrix}$$

Thus, for the FIR filter the MSE cost function has a quadratic form in the impulse response vector $\hat{\mathbf{h}}(n)$ and the minimum can be obtained by setting the partial derivatives of $\mathcal{J}$, with respect to each tap coefficient, to zero, as below.

$$\begin{aligned}\nabla\mathcal{J} &= \frac{\partial\mathcal{J}}{\partial\hat{\mathbf{h}}} = \begin{bmatrix}\frac{\partial\mathcal{J}}{\partial\hat{h}_0} & \frac{\partial\mathcal{J}}{\partial\hat{h}_1} & \cdots & \frac{\partial\mathcal{J}}{\partial\hat{h}_{L-1}}\end{bmatrix}^{\mathrm{T}} \\ &= 2\mathbf{R}\hat{\mathbf{h}} - 2\mathbf{p} \\ &= 0\end{aligned} \tag{3.3}$$

The optimum impulse response $\hat{\mathbf{h}}_{opt}$ which minimises the MSE is thus the solution to a set of L simultaneous linear equation.

$$\mathbf{R}\hat{\mathbf{h}}_{opt} = \mathbf{p} \tag{3.4}$$

If the power spectral density of the input signal $\mathbf{x}(n)$ has non-zero frequencies, the $\mathbf{R}$ matrix is positive definite and hence is non-singular [4]. Therefore, inverse of $\mathbf{R}$ exists and the unique optimum impulse response is given by

$$\hat{\mathbf{h}}_{opt} = \mathbf{R}^{-1}\mathbf{p} \tag{3.5}$$

The filter defined by (3.5) is the Wiener FIR filter or Levinson filter

This method provides minimum MSE and therefore helps to estimate the unknown room impulse response more accurately. However, this approach is not appropriate in dealing with nonstationarity signals like speech signals and spectral characteristics or autocorrelation and cross-correlation are unknown. In such situations “estimate and plug” approach can be used, whereby the filter first estimates the statistical parameters and then exploits the results for computing the filter coefficients. But, this procedure is not favourable in real-time operation because of requiring complex and costly hardware arrangement. Instead, adaptive approach can be applied on the Wiener equation to obtain solution close to optimum.

3.3 Mean-Square Error Surface

It can be seen from (3.2) that if the tap inputs of the transversal filter, $\hat{\mathbf{h}}$, and the desired response are jointly stationary, the mean square error is a quadratic function of $\hat{\mathbf{h}}$. For a given fixed $\hat{\mathbf{h}}$, the MSE can be expressed as [5]

$$\mathcal{J} = \sigma_y^2 - 2\mathbf{p}^{\mathrm{T}}\hat{\mathbf{h}} + \hat{\mathbf{h}}^{\mathrm{T}}\mathbf{R}\hat{\mathbf{h}} \tag{3.6}$$

where σ_y^2 is the variance of the desired signal, $y(n)$, assuming it has zero-mean. It forms a $(L+1)$ dimensional hyperparaboloid surface and has only positive values (i.e. convex surface). Figure 3.2 shows the parabolic surface by using only two weights for $\hat{\mathbf{h}}$, for a numerical example.

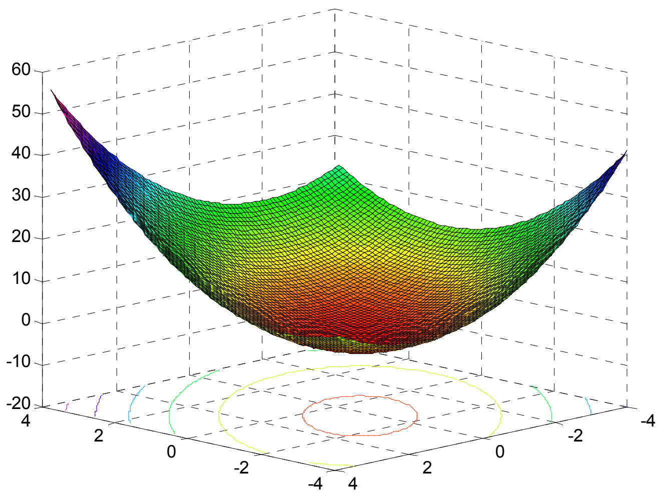


Figure 3.2: Error performance surface of the two tap transversal filter.

The minimum point of the error surface represents the Wiener solution, where the cost function shown by (3.6) attains its minimum value. At this point the minimum MSE value can be evaluated by substituting the optimum value from (3.5) into (3.6) as follows:

$$\begin{aligned}\mathcal{J}_{\min} &= \sigma_y^2 - 2\mathbf{p}^{\mathrm{T}}\hat{\mathbf{h}}_{opt} + \hat{\mathbf{h}}_{opt}^{\mathrm{T}}\mathbf{R}\hat{\mathbf{h}}_{opt} \\ &= \sigma_y^2 - 2\mathbf{p}^{\mathrm{T}}\hat{\mathbf{h}}_{opt} + \hat{\mathbf{h}}_{opt}^{\mathrm{T}}\mathbf{R}\hat{\mathbf{h}}_{opt} \\ &= \sigma_y^2 - 2\mathbf{p}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{p} + \mathbf{p}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{p} \\ &= \sigma_y^2 - \mathbf{p}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{p}\end{aligned}$$

The intersection of a plane parallel to the $\hat{\mathbf{h}}$ plane consists of an ellipse representing equal MSE contours as depicted in figure 3.3. The $\mathcal{J}_{\min}$ defines the centre of those contours.

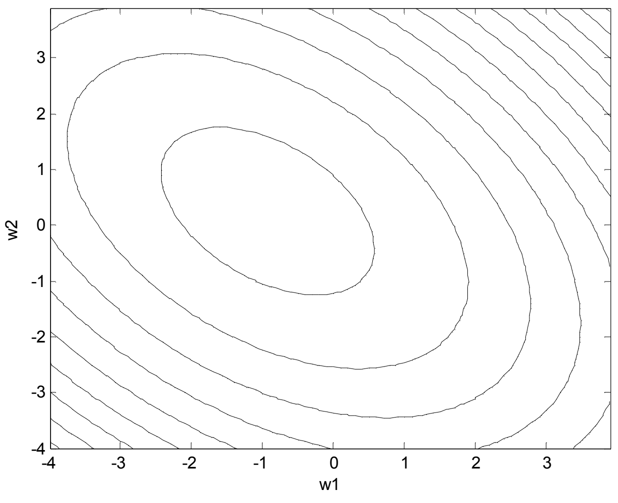


Figure 3.3: Contour plot of the error performance surface shown in figure 3.2

3.4 Steepest-Descent Method

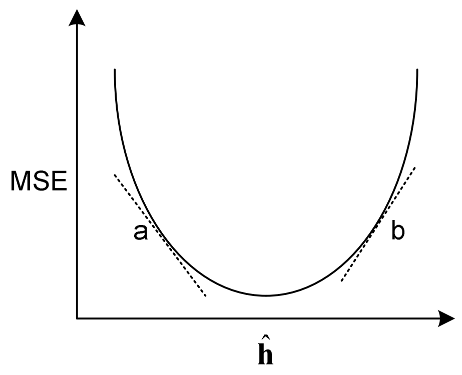
The Wiener equation, (3.5), can be solved using Gaussian elimination, but the computation needs to be done every time the statistics of the input signal changes. Therefore, the engaged computational complexity makes it impossible in the real applications. The method of steepest decent is an efficient gradient type iterative technique that has been employed to optimise cost function.

The process starts with a guess for the tap weights, $\hat{\mathbf{h}}$, and calculates the associated MSE value using (3.6). Then the gradient vector, $\nabla\mathcal{J}$ which is in the direction of the greatest rate of change of the MSE cost function, is calculated using (3.3). Finally, a scaled version of the gradient, $\mu\nabla\mathcal{J}$, is subtracted from the guess to form new and improved tap weights, as follows.

$$\hat{\mathrm{h}}_{i+1} = \hat{\mathrm{h}}_i - \mu\nabla\mathcal{J}_i \tag{3.7}$$

where $\nabla\mathcal{J}_i = \frac{\partial\mathcal{J}}{\partial\hat{\mathbf{h}}} = 2\mathbf{R}\hat{\mathbf{h}}_i - 2\mathbf{p}$ (3.8)

The step size or the convergence factor, μ, must be a small positive scalar in order to give smaller MSE at each iteration until it reaches the minimum. Figure 3.4 is used to explain the theory behind for this gradient vector subtraction.



If MSE of the initial guess = 'a'
Then gradient at 'a' = negative
Therefore, $\hat{\text{h}}_{i+1} = \hat{\text{h}}_i + \mu\nabla\mathcal{J}_i$
$\Rightarrow$ $\hat{\text{h}}$ moves in +ve direction towards the minima.

If MSE of the initial guess = 'b'
Then gradient at 'b' = positive
Therefore, $\hat{\text{h}}_{i+1} = \hat{\text{h}}_i - \mu\nabla\mathcal{J}_i$
$\Rightarrow$ $\hat{\text{h}}$ moves in –ve direction towards the minima.

Figure 3.4: Explains steepest decent equation, (3.7)

The activity diagram that summarised the procedures involves in the steepest decent algorithm can be shown in figure 3.5.

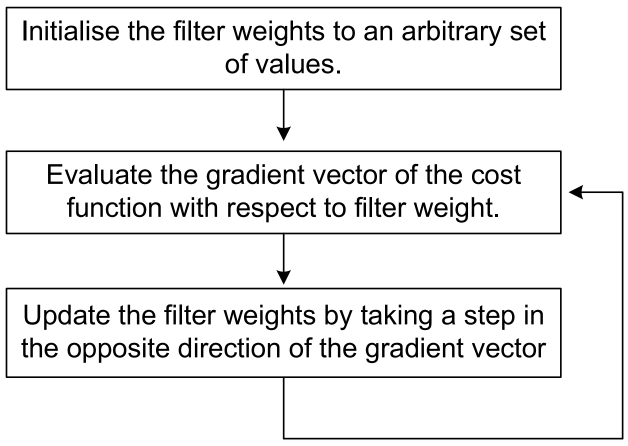

Initialise the filter weights to an arbitrary set of values.
Evaluate the gradient vector of the cost function with respect to filter weight.
Update the filter weights by taking a step in the opposite direction of the gradient vector

Figure 3.5: procedures involve in steepest decent algorithm

According to the quadratic relation between the MSE and the filter coefficient, it is guaranteed that the error performance surface always has global minima and therefore, eventually reaches the minimum point after some iteration. The step size must be lies within the range

$$0 < \mu < \frac{1}{\lambda_{\max}} \tag{3.9}$$

where $\lambda_{\max}$ is the largest eigen-value of the auto-correlation matrix, **R**.

3.6 Applications of Adaptive Filters

An adaptive filter is a digital filter that adapts its performance continuously by updating the weighting of past iteration for smoothing or forecasting purposes. The concept is very popular in most areas, including Control Engineering, military applications and communication systems, due to its ability to work in an unknown environment.

We looked at system identification class of adaptive filter for the application of acoustic echo cancellation, where the system is realised as in figure 3.6. The filter finds a linear model, $\hat{\mathbf{h}}$, that closely match with the unknown system, $\mathbf{h}$. Both $\hat{\mathbf{h}}$ and $\mathbf{h}$ are driven by same input. The adaptive system is then used the output of the unknown system, $y(n)$, as desired response and moved to a better prediction by calculating the error, $e(n)$, between its output and the desired response at each iteration.

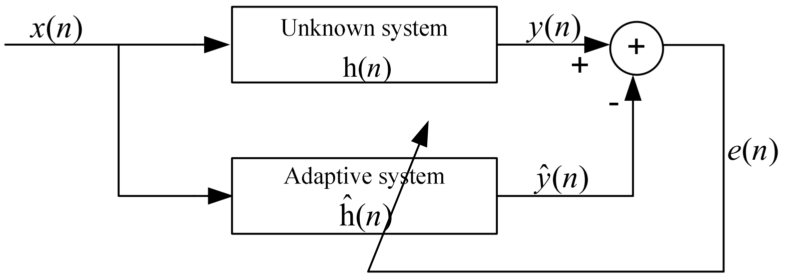


Figure 3.6: Conceptual adaptive system for identification [5]

Adaptive filters are also used to provide an inverse model that represents the best fit to an unknown noisy system. In this class of application, the adaptive system is said to be converged when the transfer function of the adaptive system is close to the reciprocal of the unknown system's transfer function. The adaptive filter setup for this purpose is illustrated in figure 3.7.

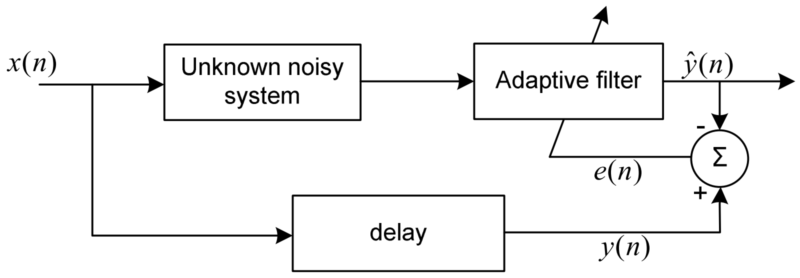


Figure 3.7: Conceptual adaptive system for inverse modelling [5]

Another application of the adapter filter is for signal enhancement, where the primary signal consists of a desired signal $x(n)$ that is corrupted by an additive noise $w_1(n)$. The input signal to the adaptive filter $w_2(n)$ is correlated with $w_1(n)$ but uncorrelated with $x(n)$. The arrangement shown in figure 3.8 demonstrates this type of application and can be found in hearing aids and noise cancellation in hydrophones.

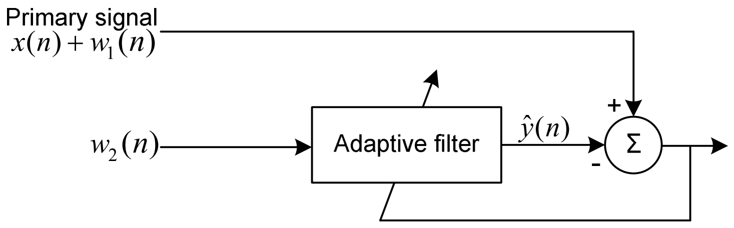


Figure 3.8: Conceptual adaptive system for signal enhancement [5]

In signal prediction application, the adaptive filter input consists of a delayed version of the desired signal as illustrated in figure 3.9. This filter is able to predict the present sample of the input signal using past values of the signal. This type is employed to estimate the speech parameters in speech coding prediction.

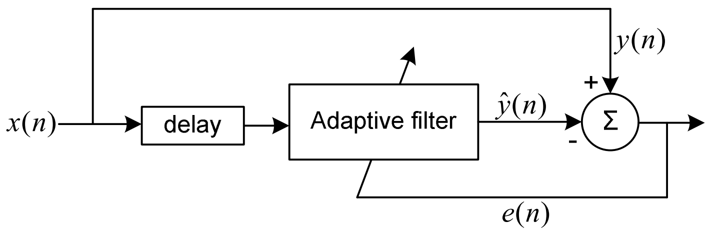


Figure 3.9: Conceptual adaptive system for signal prediction [5]

3.7 Summary

Wiener filters can be employed to find the optimal filter which is, when applied to the input signal produces a signal that is close to the desired signal. But, this approach requires knowledge of certain statistical information of the input signal and needs to perform heavy computation each time the statistics changes. The well-known recursive optimization technique, steepest descent, can be applied to the Wiener filter, and eventually converge to the optimum solution from an arbitrary starting point on the error performance surface. It provides improved solution by progressing towards the minimum point on the error performance surface as the number of iterations increases.

To be complete, the four types of adaptive filtering types have been looked with aid of their conceptual model diagrams.

CHAPTER 4

Review of existing adaptive algorithms

4.1 Introduction

The method of steepest descent avoids the direct matrix inversion inherent in the Wiener equation, (3.5), explicit knowledge of the statistics of the input signal, is still required, according to (3.8).

This chapter starts by studying the most common approach to adaptive filtering, the stochastic gradient based algorithms. Several modifications to this algorithm, are made in order to cope with practical constraints, and are discussed in later sections. Since a wide variety of algorithms are available in the literature, this chapter defines and analyses their performances using simulation results of three different subjective measures and computational requirements.

4.2 Stochastic Gradient Based Algorithms

Stochastic gradient based algorithms do not provide an exact solution to the problem of minimising the MSE as the steepest descent approach, rather approximates the solution. However, the requirement for stationary input or knowledge of auto-correlation matrix, **R**, and the cross-correlation vector, **p**, in steepest descent approach are circumvented in this algorithm.

This type of algorithms are widely used in various applications of adaptive filtering due to its low computational simplicity, proof of convergence in the stationary environment, unbiased convergence in the mean to the Wiener solution and stable behaviour in the finite-precision arithmetic implementations [6].

4.2.1 Least Mean Square (LMS) [5]

The weight update equation for LMS replaces the iterative step in the steepest descent algorithm with a time recursion and the gradient with an estimate of the gradient, as in (4.1).

$$\hat{\mathbf{h}}(n+1) = \hat{\mathbf{h}}(n) - \mu \hat{\nabla}\mathcal{J}(n) \tag{4.1}$$

where the vector $\hat{\mathbf{h}}(n)$ is an estimate of the Wiener filter, $\hat{\mathbf{h}}_{opt}$, at time n and the vector $\hat{\nabla}\mathcal{J}(n)$ is an estimation of the gradient of MSE cost function at the point where the impulse response is $\hat{\mathbf{h}}(n)$. The exact gradient $\nabla\mathcal{J}(n)$ is defined in similar manner to (3.8).

$$\begin{aligned}\nabla\mathcal{J}(n) &= \frac{\partial\mathcal{J}(n)}{\partial\hat{\mathbf{h}}(n)} \\ &= \frac{\partial}{\partial\hat{\mathbf{h}}(n)} E\{[y(n) - \hat{\mathbf{h}}^{\mathrm{T}}(n)\mathbf{x}(n)]^2\} \\ &= E\{2[y(n) - \hat{\mathbf{h}}^{\mathrm{T}}(n)\mathbf{x}(n)][-\mathbf{x}(n)]\} \\ &= -2E\{[\mathbf{x}(n)][y(n) - \hat{\mathbf{h}}^{\mathrm{T}}(n)\mathbf{x}(n)]\}\end{aligned} \tag{4.2}$$

Therefore, the n^{th} time sample estimate of the gradient, $\hat{\nabla}\mathcal{J}(n)$, is defined by taking the ensample average of the $(n+1)^{\mathrm{th}}$ time sample of (4.2).

$$\begin{aligned}\hat{\nabla}\mathcal{J}(n) &= -2[\mathbf{x}(n+1)][y(n+1) - \hat{\mathbf{h}}^{\mathrm{T}}(n)\mathbf{x}(n+1)] \\ &= -2\mathbf{x}(n+1)e(n+1)\end{aligned} \tag{4.3}$$

Substitution of (4.3) into (4.1) yields

$$\hat{\mathbf{h}}(n+1) = \hat{\mathbf{h}}(n) + 2\mu\mathbf{x}(n+1)e(n+1) \tag{4.4}$$

The resulting gradient based algorithm is known as the least-mean-square algorithm, since it seeks to minimise the MSE. The convergence factor, μ, must be chosen in the range as defined in (3.9) for the steepest descent case. The realisation of the LMS algorithm for a delay line input $x(n)$ is illustrated in figure 4.1, where the required multiplication and addition operations are shown in detail.

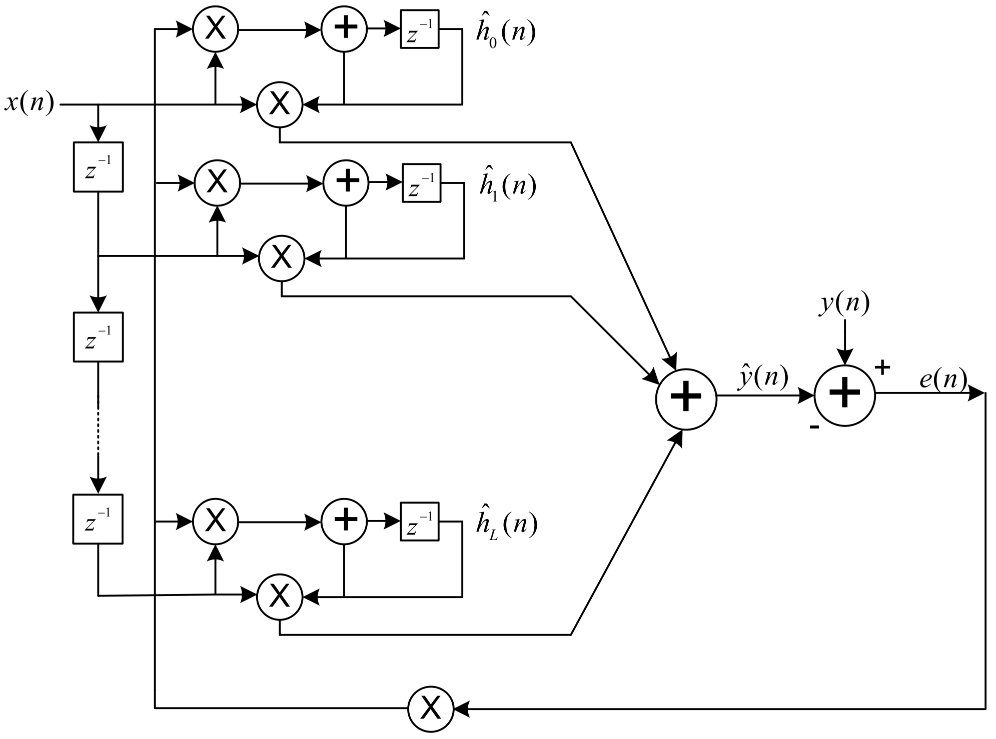


Figure 4.1: LMS adaptive FIR filter [6]

4.2.2 Normalised Least Mean Square (NLMS) [5]

NLMS filter is structurally same as that of LMS, but it differs in the way that the taps weights are updated. In the LMS algorithm the weight adjustment is directly proportional to the tap input vector, $\mathbf{x}(n)$, according to (4.4). Therefore, when the $\mathbf{x}(n)$ vector is large, the LMS suffers from a gradient noise amplification problem.

To overcome this problem, the adjustment applied to the tap weight vector at each iteration is normalised with respect to the squared Euclidean norm of $\mathbf{x}(n)$ [5].

$$\hat{\mathbf{h}}(n+1) = \hat{\mathbf{h}}(n) + 2\mu \frac{\mathbf{x}(n+1)e(n+1)}{\| \mathbf{x}(n+1) \|_2^2 + \delta_{\text{NLMS}}} \qquad (4.5)$$

where $\delta_{\text{NLMS}} = \sigma_x^2$, the variance of the input signal.

δ_{NLMS} is the regularization parameter which prevents division by zero during initialization when $\mathbf{x}(n) = \mathbf{0}$. In order to maintain the stability the step size must be in the range

$$0 < \mu < 2\frac{E\{|\mathbf{x}(n+1)|^2\}\mathcal{D}(n+1)}{E\{|e(n+1)|^2\}}$$

where $E\{|\mathbf{x}(n)|^2\}$ - power of the tap inputs

$E\{|e(n)|^2\}$ - power of the error signal, and

$\mathcal{D}(n)$ - mean square deviation

4.3 Sparse Algorithms

A sparse impulse response has most of its components with small or zero magnitude and can be found in telephone networks. Due to the presence of bulk delays in the path only 8-10% exhibits an active region. Figure 4.2 shows a typical sparse impulse response, which can be realised in reality.

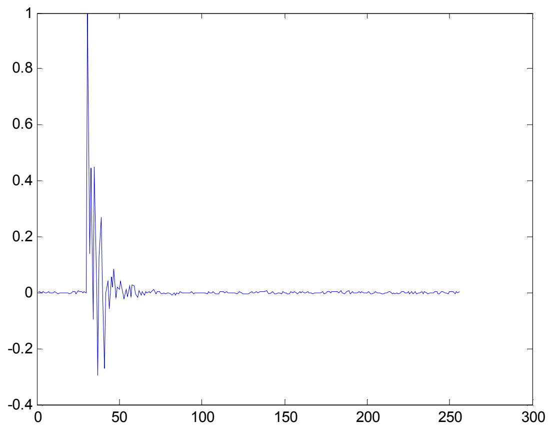


Figure 4.2: An example of a sparse impulse response exists in telephone networks.

The NLMS algorithm does not take into account this feature when it presents in a system and therefore performs inadequately. This is because [7]

- The need to adapt a relatively long filter
- The unavoidable adaptation noise occur at the inactive region of the tap weights

While in the NLMS, the adaptation step is same for all components of the filter, in the sparse algorithms, such as PNLMS, IPNLMS and MPNLMS, adaptive step sizes are calculated from the last estimate of the filter coefficients in an efficient way that the step size is proportional to the size of the filter coefficients. This is resulted to adjust the active coefficients faster than the non-active ones. Therefore, the overall convergence time is reduced.

4.3.1 Sparseness Measure

Degree of sparseness can be qualitatively referred as a range of strongly dispersive to strongly sparse [7]. Quantitatively, the sparseness of an impulse response can be measured by the following sparseness measure [8][9].

$$\xi(\mathbf{h}) = \frac{L}{L\text{-}\sqrt{L}}\left[1-\frac{\|\mathbf{h}(n)\|_1}{\sqrt{L}\,\|\mathbf{h}(n)\|_2}\right] \tag{4.6}$$

where $|\mathbf{h}(n)\|_1 = \sum_{l=0}^{L-1}|h_l(n)|$,

$|\mathbf{h}(n)\|_2 = \sqrt{\sum_{l=0}^{L-1}h_l^2(n)}$, and

L is the length of filter **h**. The measure takes values between 0 and 1 where the lower bound can be obtained by using the uniform filter $[1\ ...1]^{\mathrm{T}}$ and the upper limit can be achieved by using the Dirac filter $[1\ 0\ ...0]^{\mathrm{T}}$.

4.3.2 Proportionate-Normalised Least Mean Square (PNLMS) [10][11]:

In order to track sparse impulse response faster Proportionate NLMS (PNLMS) was introduced from the NLMS equation. The coefficient update equation of the PNLMS is slightly differ from NLMS with the extra step size update matrix **Q** as shown below and the rest of the terms are carried over from NLMS.

$$\hat{\mathbf{h}}(n+1) = \hat{\mathbf{h}}(n) + \mu\frac{\mathbf{Q}(n)\mathbf{x}(n+1)e(n+1)}{\mathbf{x}^{\mathrm{T}}(n+1)\mathbf{Q}(n)\mathbf{x}(n+1)+\delta_{\mathrm{PNLMS}}} \tag{4.7}$$

where $\delta_{\mathrm{PNLMS}} = \delta_{\mathrm{NLMS}}/L$, and the diagonal matrix

$$\mathbf{Q}(n) = \operatorname{diag}\{q_0(n) \quad q_1(n) \quad \cdots \quad q_{L\text{-}1}(n-1)\}$$

$$= \begin{bmatrix} q_0(n) & 0 & \cdots & 0 \\ 0 & q_1(n) & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & q_{L\text{-}1}(n-1) \end{bmatrix}$$

controls the step size. These control matrix elements can be expressed as

$$q_l(n) = \frac{k_l(n)}{\frac{1}{L}\sum_{i=0}^{L-1}k_i(n)} \tag{4.8}$$

$$k_l(n) = \max\left\{\rho\times\max\left\{\gamma,|\hat{\mathrm{h}}_0(n)|...|\hat{\mathrm{h}}_{L\text{-}1}(n)|\right\},|\hat{\mathrm{h}}_l(n)|\right\} \text{ with } l = 0,\ 1...\ L\text{-}1 \tag{4.9}$$

Parameters ρ and γ have typical values of $5/L$ and 0.01 [7], respectively. The former prevents coefficients from stalling when they are much smaller than the largest coefficient and the latter prevents $\hat{h}_l(n)$ from stalling during initialisation stage.

4.3.3 Improved Proportionate Normalised Least Mean Square (IPNLMS) [7]

An improvement of PNLMS is the IPNLMS algorithm, which employs a combination of proportionate (PNLMS) and non-proportionate (NLMS) updating technique, with the relative significance of each controlled by a factor α. (4.9) is updated as follows for the IPNLMS approach.

$$k_l(n) = \frac{1-\alpha}{2L} + (1+\alpha)\frac{|h_l(n)|}{2\,|\mathbf{h}(n)\|_1 + \in} \tag{4.10}$$

where $\in$ is a small positive number. Results from [7] shows that the good choice for α are 0, -0.5 and -0.75. The regularisation parameter should be taken as follows, in order to achieve the same steady state misalignment compare to that of NLMS using same step size.

$$\delta_{\text{IPNLMS}} = \frac{1-\alpha}{2L}\delta_{\text{NLMS}} \tag{4.11}$$

(4.10) is made up of the sum of two terms, where the first is a constant and the second term is a function of the weight coefficients. It can be noticed that, when α = -1 the second term in (4.10) becomes zero and therefore the *k* becomes 1/*L*. It means that the same update will be made to all filter coefficients regardless of their individual magnitudes. So, for this α value IPNLMS performs as NLMS. For α close to unity, the second term dominates the equation and as a result it behaves as PNLMS.

4.3.4 μ- Proportionate Normalised Least Mean Square (MPNLMS) [12]

The MPNLMS algorithm calculates the optimal proportionate step size in order to achieve the fastest convergence during the whole adaptation process until the adaptive filter reaches its steady state. The definition for $k_l(n)$ of MPNLMS is differed as follows from that of previous proportionate algorithms.

$$k_l(n) = \max\left\{\rho \times \max\left\{\gamma, \mathrm{F}(|\hat{h}_0(n)|) ... \mathrm{F}(|\hat{h}_{L\text{-}1}(n)|)\right\}, \mathrm{F}(|\hat{h}_l(n)|)\right\} \tag{4.12}$$

where $\mathrm{F}(|\hat{\mathrm{h}}_l(n)|) = \dfrac{\ln(1+\mu|\hat{\mathrm{h}}_l(n)|)}{\ln(1+\mu)}$, (4.13)

$|\hat{\mathrm{h}}_l(n)| \le 1$, and $\mu = 1/\varepsilon$

The constant 1 inside the logarithm is to avoid obtaining negative infinity at the initial stage when $|\hat{\mathrm{h}}_l(n)| = 0$. The denominator $\ln(1+\mu)$ normalizes $\mathrm{F}(|\hat{\mathrm{h}}_l(n)|)$ to be in the range [0, 1]. The vicinity ε is a very small positive number, and its value should be chosen based on the noise level. ε = 0.001 is a good choice for echo cancellation, as the echo below -60 dB is negligible.

4.4 Performance Measures

The choice of one algorithm over the wide variety of others needs to be addressed to differentiate it from the rest, so that one can pick a right algorithm for his particular application. The following three measures deal with different concepts in applications akin to echo cancellation.

4.4.1 Mean Square Error (MSE)

MSE is one of the ways to define an objective function that satisfies the optimality and non-negativity properties. It is the expected value of the square of the error and can be seen from (4.14) that the lower MSE value is favourable.

$$\mathrm{MSE}(n) = E\{e^2(n)\} \tag{4.14}$$

4.4.2 Echo Return Lossless Enhancement (ERLE)

It measures the attenuation of the echo signals in an Acoustic Echo Cancellation system. It can be witnessed from (4.15) that a higher ERLE corresponds to higher reduction in echo.

$$\mathrm{ERLE}(n) = 10 \times \log_{10} \frac{y^2(n)}{e^2(n)} \mathrm{dB} \tag{4.15}$$

4.4.3 Normalised Projection Misalignment (NPM) [13]

The normalized projection misalignment measures the closeness of the estimated impulse response ($\hat{\mathbf{h}}(n)$) to that of the unknown impulse response ($\mathbf{h}(n)$).

It can be expressed as follows:

$$\mathrm{NPM}(n) = 20 \times \log_{10}\left(\frac{1}{\|\mathbf{h}\|} \left\| \mathbf{h} - \frac{\mathbf{h}^{\mathrm{T}}\hat{\mathbf{h}}(n)}{\hat{\mathbf{h}}^{\mathrm{T}}(n)\hat{\mathbf{h}}(n)} \hat{\mathbf{h}}(n) \right\| \right) \mathrm{dB}$$

where the denominator is defined as the squared l_2-norm operator. To achieve a good performance, the misalignment must be close to zero, which is the case when the length of unknown filter (L_R) is close to that of adaptive filter (L). It is interesting to note that when the filter has only one tap the term inside the logarithm becomes zero and therefore yields negative infinity for NPM.

4.5 Computational Complexity

It is also necessary to examine the computational complexity of an algorithm. Although many factors contribute to the complexity of an algorithm, the relative complexity of the four algorithms in terms of the total number of additions, multiplications, divisions and logarithms per iteration is assessed from figure 4.3 to 4.6, along with their MATLAB implementation code on the left.

The followings should be noted:

- The comparison between two numbers takes one subtraction. In this content, subtraction is counted as addition.
- The computation of the norm2 $|\mathbf{h}(n)\|_2^2$ requires two multiplications and one addition using the following recursive method.

$$\mathbf{h}(n)\|_2^2 = \mathbf{h}(n-1)\|_2^2 + (1-\lambda)x^2(n) \quad \text{where } \lambda \text{ is the forgetting factor [2]}$$

```
%                                                              _|_+_|_x_|_/_|
for n= 1: length(input);%                                       |   |   | - |
    x            = [input(n); x(1:end-1)];%                     | 1 | - | - |
    output(n)    = weight' * x;%                                | - | L | - |
    e(n)         = d(n) - output(n);%                           | 1 | - | - |
    weight       = weight + (mu/(epsilon+x'*x))*x*e(n);%|L+2|L+3| 1 |
end
```

Figure 4.3: MATLAB implementation code of NLMS and related computations

```
%                                                         _|_+_|_x_ |_/_|
for n= 1: length(input);%                                  |   |     |   |
   x         = [input(n); x(1:end-1)];%                    | 1 | -   | - |
   output(n)= weight' * x;%                                | - | L   | - |
   e(n)      = d(n) - output(n);%                          | 1 | -   | - |
   a         = max([0.01;abs(weight)]);%                   | L | -   | - |
   k         = max((5/length(h))*a,abs(weight));%          | L | L   | 1 |
   Q         = length(h)*(k/sum(k));%                      | L | L   | L |
   weight  = weight+((mu*e(n)*(Q.*x))/((x'*(Q.*x))+ep))%|L+1|3L+3| 1 |
end
```

Figure 4.4: MATLAB implementation code of PNLMS and related computations

```
%                                                     _|_+_ |_x_ |_/_|
for n= 1: length(input);%                              |     |     |   |
  x          = [input(n); x(1:end-1)];%                | 1   | -   | - |
  output(n)  = weight' * x;%                           | -   | L   | - |
  e(n)       = d(n) - output(n);%                      | 1   | -   | - |
  Q1  = (1-a)/(2*length(h));%                          | 1   | 1   | 1 |
  Q   = Q1+(((1+a)*abs(weight))/((2*sum(abs(weight)))+eps));
   %                                                   |3L+3| L+1| L  |
  weight=weight+((mu*e(n)*(Q.*x))/((x'*(Q.*x))+ep));%|L+1  |3L+3| 1  |
end
```

Figure 4.5: MATLAB implementation code of IPNLMS and related computations

```
%                                                _|_+_ |_x_ |_/_|log|_
for n= 1: length(input);%                         |     |     |   |     |
  x          = [input(n); x(1:end-1)];%           | 1   | -   | - | -   |
  output(n)= weight' * x;%                        | -   | L   | - | -   |
  e(n)       = d(n) - output(n);%                 | 1   | -   | - | -   |
  delta      = 1000;%                             | -   | -   | - | -   |
  F = log(1+delta*(abs(weight)))/constant;     %  | L   | L   | 1 | L   |
  a = max([0.01;F]);%                             | L   | -   | - | -   |
  k = max((5/length(h))*a,F);%                    | L   | L   | 1 | -   |
  Q = length(h)*(k/sum(k));%                      | L   | L   | L | -   |
  weight=weight+((mu*e(n)*(Q.*x))/((x'*(Q.*x))+ ep))%
%                                                 |L+1  |3L+3| 1 | -   |
end
```

Figure 4.6: MATLAB implementation code of MPNLMS and related computations

The following table summarised the computational operations for the all four algorithms. It can be noticed that the overall computational complexity is increased or stayed same when the improvement is made. To compensate this issue their performances must be significantly higher.

	NLMS	PNLMS	IPNLMS	MPNLMS
Addition	L+4	4L +3	4L +7	5L +3
Multiplication	2L +3	6L+3	5L +5	7L +3
Division	1	L+2	L+2	L+3
Logarithm	0	0	0	L

Table 4.1: Computational complexities of the four algorithms.

4.6 Simulation Results

Individual results for the four algorithms that described in sections 4.2.2 and 4.3 in an echo cancellation experiment are presented and discussed in this section using two different types of input signals and 4 different levels of sparse measures. All of the simulations are performed using synthetic data via MATLAB and the implementations codes can be found in appendix.

4.6.1 Experiment Setup

Figure 4.7 shows the signals necessary for this experiment. In order to obtain more accurate results, compared to the practical situation, the input signals are chosen realistically as follows.

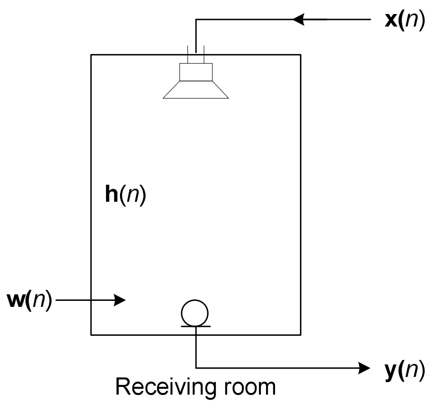


Figure 4.7: Illustrates the required signals for the echo cancellation experiment

The source signal, **x**(n), is generated using the `randn` command with the vector size of L_s where L_s = (sampling frequency * simulation time). 8 kHz is chosen as the sampling frequency and 2.5 seconds is for the simulation time.

The receiving room impulse response, **h**(n), is generated synthetically using the method proposed in [10], so that the sparseness can be controlled easily. The method starts by defining vector $\mathbf{u}$

$$\mathbf{u}_{L\times 1} = [\mathbf{0}_{L_p\times 1} \quad 1 \quad e^{-1/\psi} \quad e^{-2/\psi} \quad \cdots \quad e^{-(L_u-1)/\psi}]^{\mathrm{T}}$$

where the leading zeros with length L_p models the length of the bulk delay and $L_u = L - L_p$ is the length of the decaying window which can be controlled by ψ. Smaller the ψ value yields more sparse system. The synthetic impulse response can then be expressed as

$$\mathbf{h}(n) = \begin{bmatrix} \mathbf{0}_{L_p\times L_p} & \mathbf{0}_{L_p\times L_u} \\ \mathbf{0}_{L_u\times L_p} & \mathbf{B}_{L_u\times L_p} \end{bmatrix} \mathbf{u} + \mathbf{p}$$

where $\mathbf{B}_{L_u\times L_p} = \mathrm{diag}\{\mathbf{b}\}$, and vector $\mathbf{b}$ and $\mathbf{p}$ are zero-mean white Gaussian noise vectors with length L_u and L, respectively. $\mathbf{p}$ is used to model the merging inactive regions in the pulse response. Figure 4.9 shows two impulse responses that can be attained using this approach, by setting the impulse length L=256, the bulk delay length L_p=30 and ψ to infinity (non-sparse) and 8 (sparse).

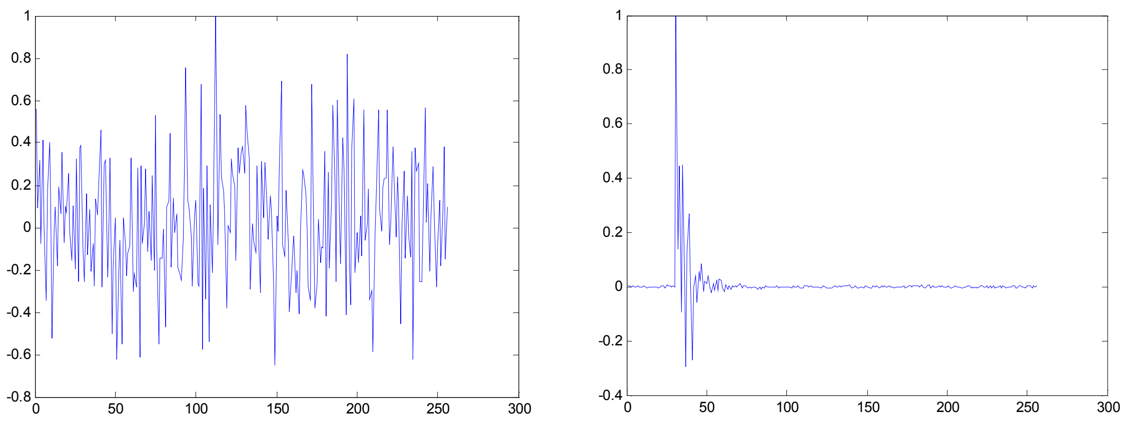


Figure 4.9: Non-sparse (left) and Sparse (right) impulse responses using ψ = infinity and ψ =8, respectively.

The input source signal, **x**(*n*), is filtered through the in-built FIR filter (i.e. `filter` command with denominator coefficients set to 1) using the generated impulse response, **h**(*n*) (numerator coefficients). A white Gaussian noise signal, **w**(*n*), with 30 dB SNR (signal-to-noise ratio) is added to the filtered signal to generate the output signal, y(*n*).

The source signal, **x**(*n*), is now fed as the input signal to the adaptive filter, whereas the y(*n*) is used as the desired signal. There is a trade-off in choosing the optimum step size, as higher the value results to converge quicker, but may be far more apart (mis-adjustment) from the Wiener optimum solution.

In most of the analysis the convergence speed is measured using the time to reach the -20 dB level of the NPM. Previous studies in the concept of echo cancellation have proved that at the -20 dB of NPM level echo canceller gives satisfactory performance. In order to simplify calculations, adaptive filter with 256 taps is used throughout the project, even though 1024 taps are typically required [7].

A series of experiments are carried for the four algorithms, namely NLMS, PNLMS, IPNLMS and MPNLMS, separately, in order to understand their theories. And in each experiment the adaptive process is repeated 10 times to obtain the ensample average of the error and the NPM values. These values are then averaged over 100 blocks to get smooth graphs. Simulation results are presented under each algorithm and their strengths and weaknesses are identified.

The step size parameter for the adaptive process is chosen in such a way that all the algorithms achieve same steady state for the NPM measure. This is achieved by setting a sensible step size for NLMS and then step size for others are obtained by attaining the same steady state as NLMS.

__**Experiment 1**: Trade-off in choosing the step size__

The simulations start to study the trade-off in choosing the best step size for its adaptive process. Figure 4.10 shows the plot of the NPM value of NLMS algorithm against different step sizes, when the filter coefficients are initialised to zeros. Different step sizes can be chosen randomly, but by theory it must be in the range between 0 and 1.

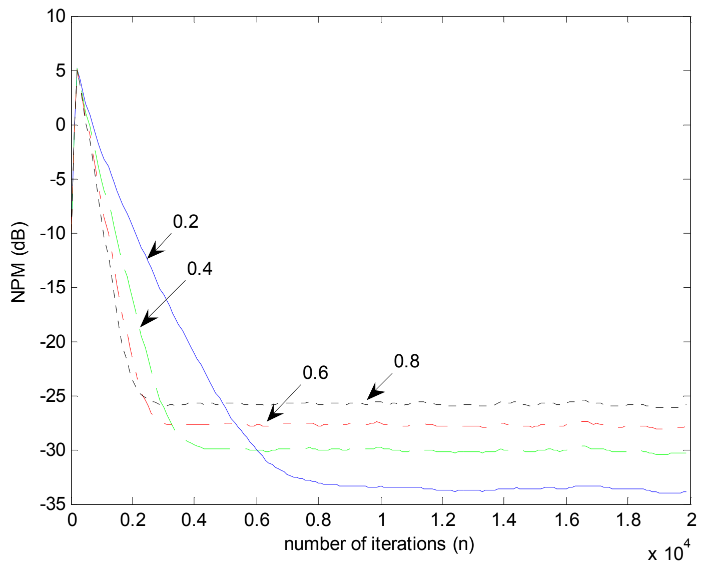


Figure 4.10: plot of NPM value vs. different step sizes, for non-sparse system.

It can be seen from figure 4.10 that the smaller step size, for example 0.2, exploits better performance in converging to the true system, about -33 dB for this example. But it takes longer to reach the -20 dB point, about 8000 iterations as in this case. While for 0.8, the steady state is achieved at -25 dB and it intersects the -20 dB line in about 2000 iterations.

Out of these four values, 0.4 is a good choice for the step size, since it gives a satisfactory performance in the presence of 30 dB SNR level and also by taking into account the involved trade-off between mis-adjustment and the convergence speed.

Experiment 2: Find step size for all algorithms to achieve same NPM steady state.

This experiment is carried out by setting 0.4 to the step size of NLMS, and tuned the others step sizes to achieve the same NPM steady state. It is found that, for a system with randomly chosen sparseness level, the steady state is achieved by using 0.4 for PNLMS and IPNLMS and 0.3 for MPNLMS. Figure 4.11 shows the steady convergence of the algorithms.

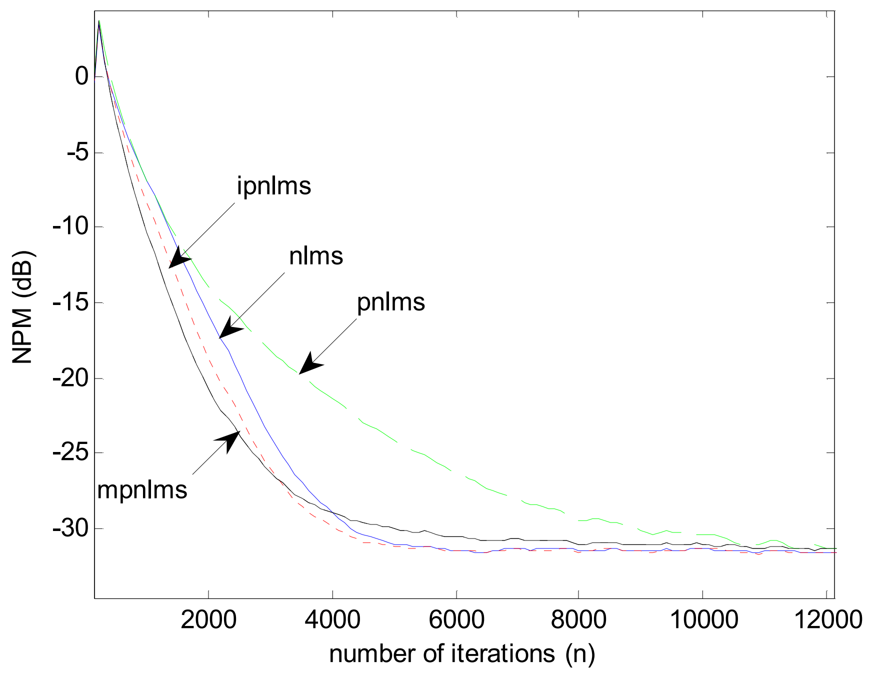


Figure 4.11: Same NPM steady state convergence of NLMS, PNLMS, IPNLMS and MPNLMS algorithms.

4.6.2 NLMS Performance

Experiment 3: Performance of NLMS in different sparse systems.

Further experiment is carried out to find the performance of NLMS in different sparse systems. Figure 4.12 shows the MSE, ERLE and NPM results for this experiment, where the different sparseness is obtained using ψ = Infinity, ψ = 100, ψ = 40 and ψ = 8, respectively. The four different impulse responses are shown in different colours on the bottom right hand side of figure 4.12.

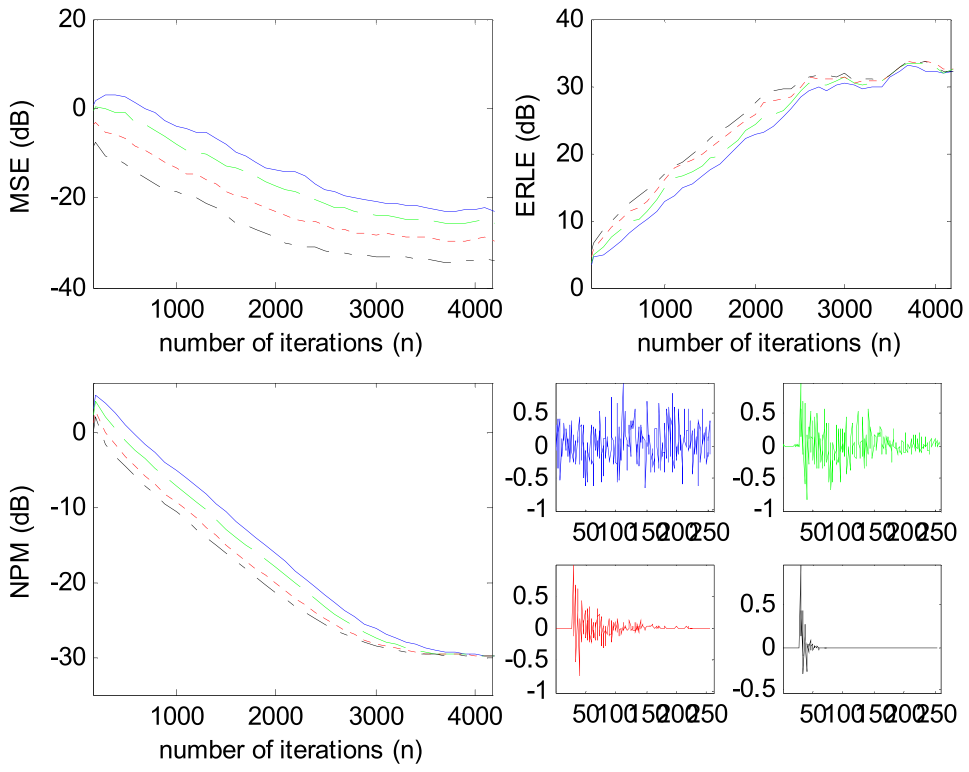


Figure 4.12: plot of MSE, ERLE and NPM for different sparse systems controlled by ψ = Infinity, ψ = 100, ψ = 40 and ψ = 8.

Figure 4.12 shows that the MSEs for the different sparse systems are not same at steady state. In the sparse case (black plot) it has less error between the adaptive filter output and the desired response. Since the error signal is sent back to the origin, sparse response is favourable here, compare to the non-sparse system. Moreover, the ERLE and NPM become better as sparseness is increased. This is because, as the impulse response becomes sparser (i.e. less active taps), more coefficients are already reached their optimum (or close to optimum) values even at the beginning of the adaptive process, due to the zero initialisation.

Experiment 4: Effect of sparseness on rate of convergence for NLMS

100 different sparseness measures (ξ) are obtained (covers about 62% of the sparseness range) by adjusting the standard-deviation of the inactive taps. The time taken to cut the desired NPM level is plotted against ξ in figure 4.13. The time taken is calculated by dividing the number of iterations to reach the point by the sampling frequency.

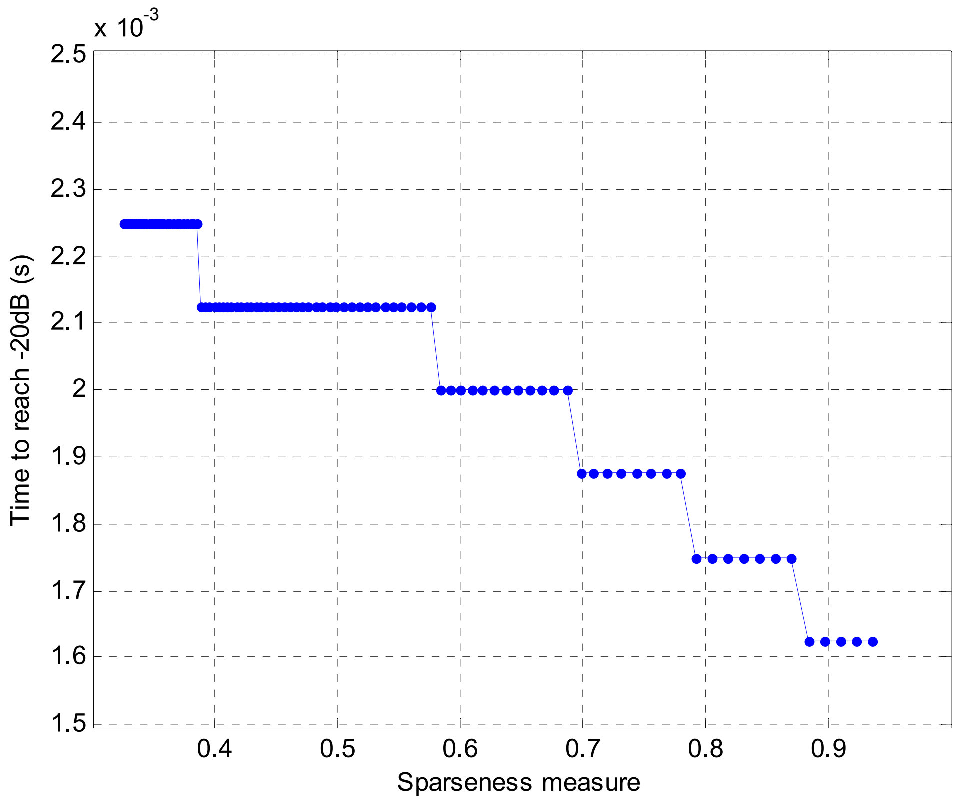


Figure 4.13: plot of Time to reach the -20dB NPM against different sparseness measures of 100 systems.

Figure 4.13 proves that there is not a strict linear relationship between the convergence speed and sparseness measure. But the stair case shape graph proves that within some segments of sparseness measure the convergence speed is exactly same and there is a constant drop across these segments. The mathematical analysis from [12] says that the convergence speed for NLMS is dependant only on the largest distance between the optimum tap weight and its initial value. Within the segments, the experimental observation obeys the theory, but not consistently throughout the all sparse systems, even though the all time highest tap weight is 1. This experiment suggests that the sparseness measure also plays a role in finding the exact convergence speed.

4.6.3 PNLMS Performance

<u>**Experiment 5**: Performance of PNLMS in different sparse system</u>

The experiment 3 procedures are repeated by using the PNLMS algorithm and the simulation result is presented in figure 4.14.

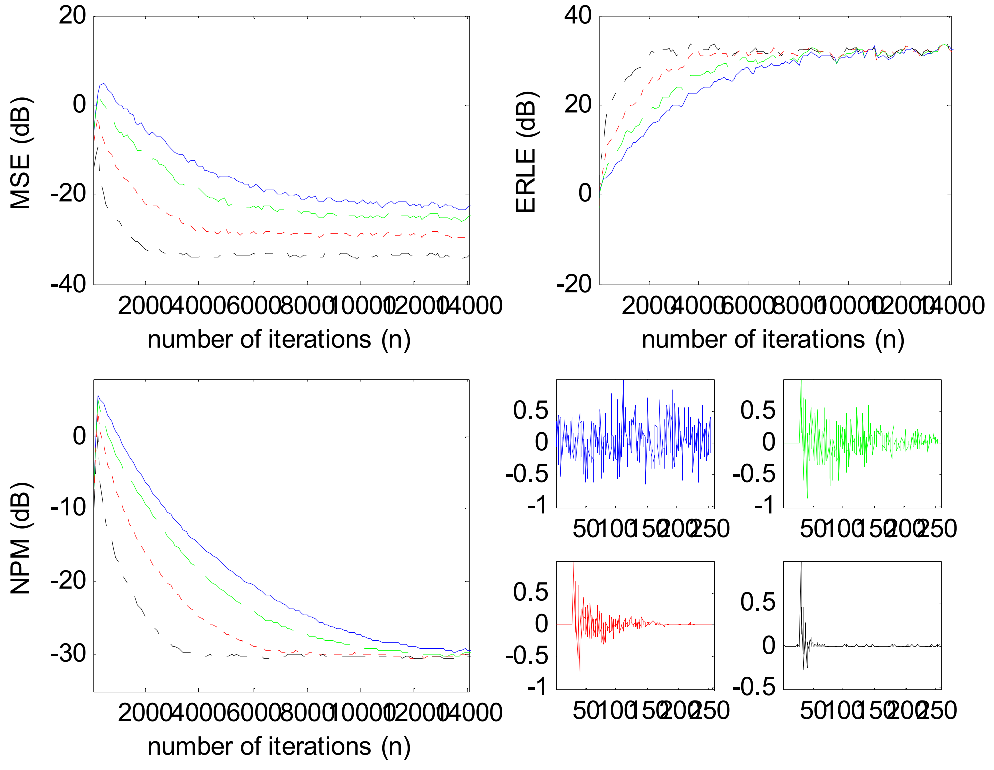


Figure 4.14: plot of MSE, ERLE and NPM for different sparse systems controlled by ψ = Infinity, ψ = 100, ψ = 40 and ψ = 8

MSE, ERLE and NPM develop into better performance as the system gets sparser for same reason as in the case of NLMS. In addition, the curves start off steeply and slow down dramatically after the initial period, which is hinting its ability to provide fast initial convergence. The deviation between the NPM curves gets larger as the system changes towards sparse. It is evidenced that the relative performance of PNLMS in sparse system is higher and as a result it could be favourable in cancelling echoes in sparse situations.

4.6.4 IPNLMS Performance

Experiment 6: <u>Performance of IPNLMS in different sparse system</u>

The controlling factor α is chosen to be -0.5 in this simulation, since it gives better performance in both sparse and non-sparse responses, compared to -0.75 and 0. Figure 4.15 is obtained by performing the procedures of the experiment 3 for the case of IPNLMS algorithm.

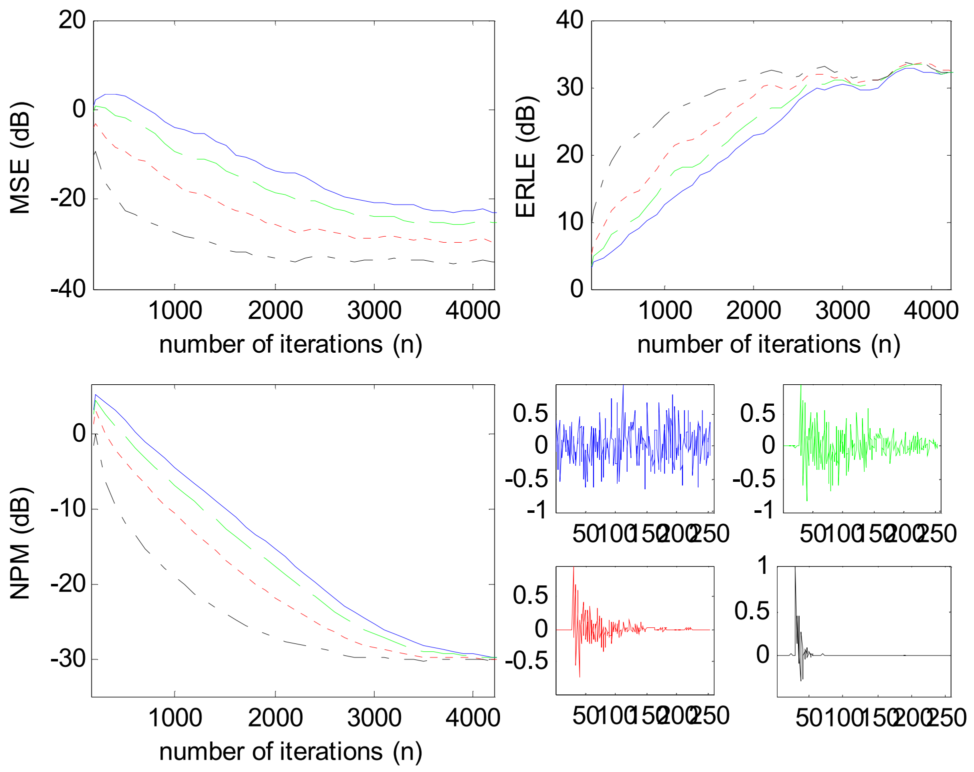


Figure 4.15: plot of MSE, ERLE and NPM for different sparse systems controlled by ψ = Infinity, ψ = 100, ψ = 40 and ψ = 8

It can be seen from figure 4.15 is that more favourable result is obtained for MSE, ERLE and NPM as the system gets sparser. The performance in the very sparse system, in particular, is very appreciative for IPNLMS, and therefore it can also be utilized in sparse systems. A point to note here is that, further experiments using α = -0.75 and α = 0 are carried out and the former gives better performance for non-sparse system and the latter gives better performance for sparse system.

4.6.5 MPNLMS Performance

<u>**Experiment 7**: Performance of the MPNLMS in different sparse system</u>

Experiment 3 is repeated for MPNLMS algorithm and the resulting graph is plotted in figure 4.16.

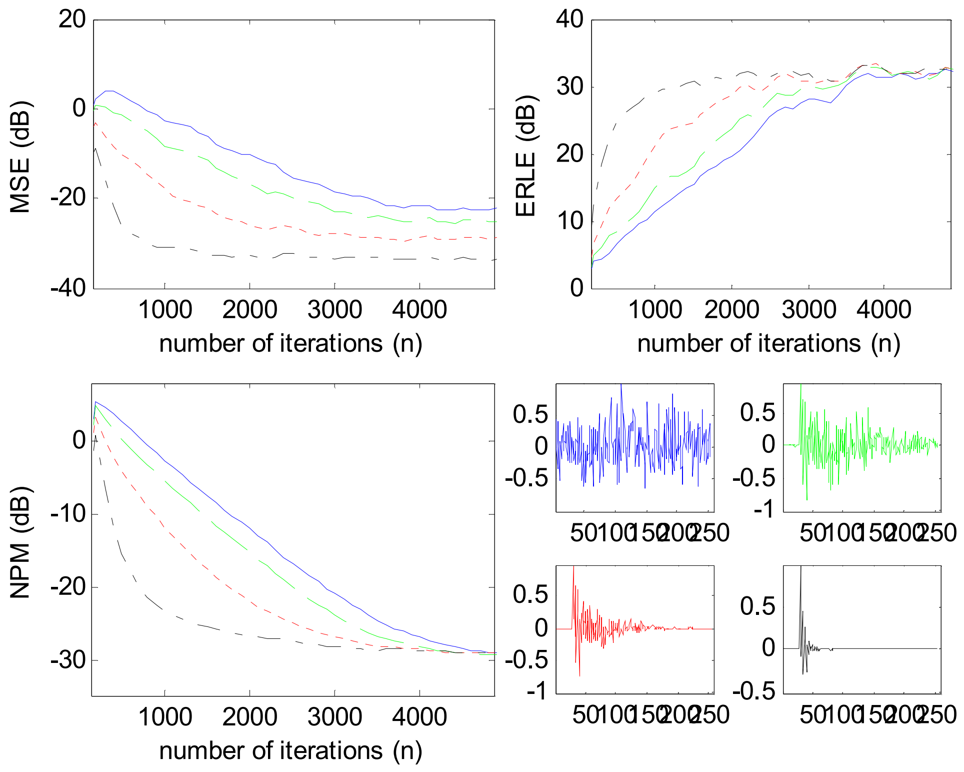


Figure 4.16: plot of MSE, ERLE and NPM for different sparse systems controlled by ψ = Infinity, ψ = 100, ψ = 40 and ψ = 8

In very sparse system the performance of the MPNLMS is very affective, as all three performances give the steepest graphs as a result of earliest convergence. However, its performance is degraded in non-sparse system, due to the effect of the logarithm in the tap weight update equation.

4.7 Comparisons and discussions

Effect of sparseness on rate of convergence for NLMS, PNLMS, IPNLMS and MPNLMS is studied comparatively by repeating the experiment 4 for all the algorithms and the resulting graph is plotted in figure 4.17.

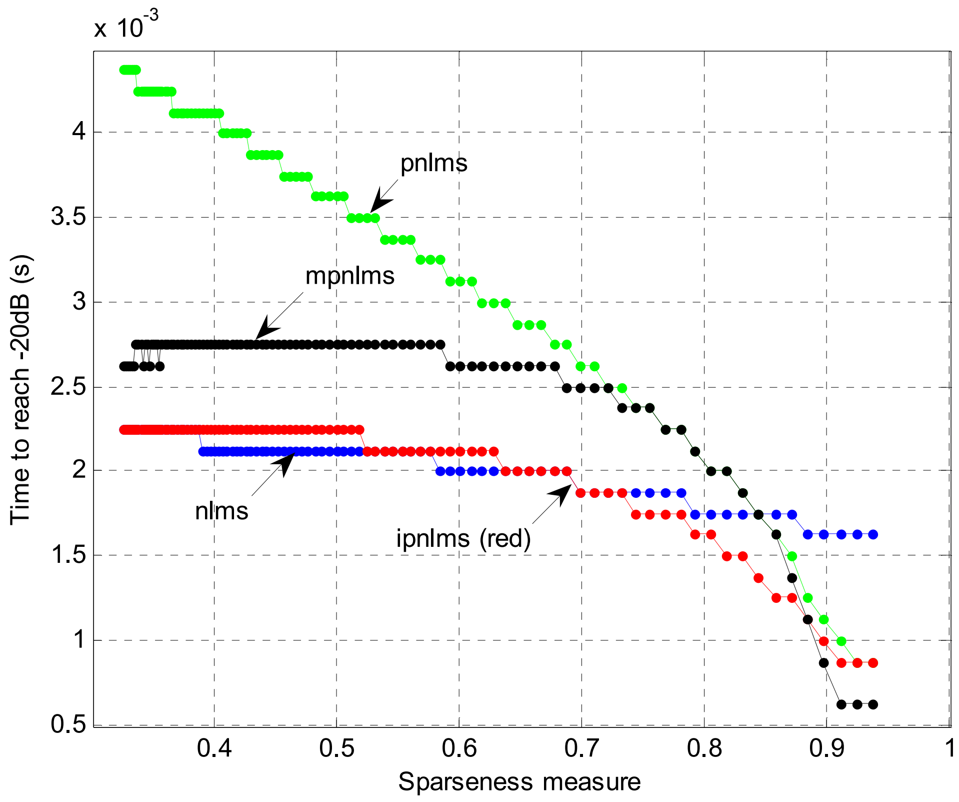


Figure 4.17: Plot of Time to reach the -20dB NPM against different sparseness measures of 100 systems for NLMS, PNLMS, IPNLMS and MPNLMS.

Figure 4.17 tells that there may be a second order relationship between the convergence speed and the sparseness measure. When going from non-sparse to sparse the speed of convergence is reduced significantly by following an increasing negative gradient type of curve (convex). In non-sparse system (ξ~0.3), NLMS performs the best and PNLMS gives the worst performance. On the other end, MPNLMS performs best whereas NLMS performs badly.

For NLMS, constant tap weight is applied to all the taps regardless of their magnitudes. In non-sparse system, since the taps are initialised to zeros, in early stages its error is high. As a result it step size becomes higher and consequently converges quickly. For the sparse case, since most of the taps are in their optimum, the resulting error is smaller. So that it relative convergence speed slows down as the step size is reduced.

For PNLMS in non-sparse case, half of its active regions get the **Q** (refer (4.7) factors below 1 and the lower active regions get the **Q** factors above 1. In sparse systems all the active regions get the **Q** factors much higher than 1, according to (4.8). Therefore, its convergence speed gets faster when it slides towards sparse systems (ξ~0.8). As it can be seen from figure 4.17 is that PNLMS is favourable only in sparse systems.

Out of these four algorithms, IPNLMS presents good performance over all sparse levels. The reason is simply because it is a combination of the best non-sparse algorithm (NLMS) and the second best sparse algorithm (PNLMS).

Performance of MPNLMS stays same up-until the sparse is significantly developed (ξ~0.6). And then it follows steepest convex curve shape. This is resulted from the effect of logarithm, when the tap weights get smaller the rate of reduction of their F values (refer (4.13)) increase. Therefore, when the system's impulse response changes from non-sparse to sparse, the denominator of (4.8) (i.e. average of F) reduces very quickly and as a result the rate of convergence speed increases dramatically.

All these observations are hinting that including the sparseness measure in the weight update equation may help to improve their performance throughout all circumstances.

4.8 Summary

The process involved in the three different approaches in system identification may be summarised using figure 4.18. The Wiener filter reaches the optimum solution in one step, whereas the steepest descent and the stochastic gradient methods reach a sub-optimum solution by taking calculated and estimated steps, respectively.

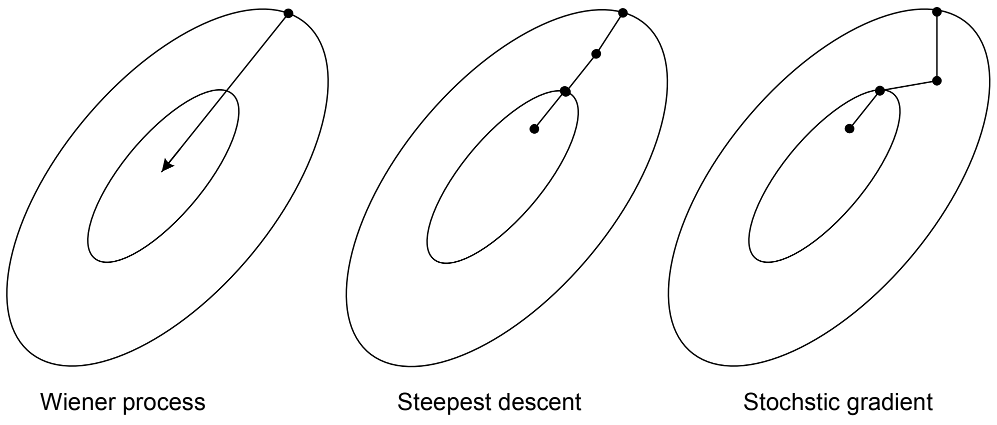


Figure 4.18: Summarised the process of the three well-known approaches in system identification using contour plots.

The performances of four stochastic gradient based algorithms, NLMS, PNLMS, IPNLMS and MPNLMS, have been looked for a range of different sparse systems, separately and comparatively. The results showed that NLMS is better in non-sparse system, whereas the MPNLMS is superior for sparse impulse response. Furthermore, none of them perform well across all sparseness level.

CHAPTER 5

Novel Algorithms for Sparse Systems

5.1 Introduction

In reality, the impulse response of the receiving room does not follow a same path. The path may vary with time due to change in room characteristics, including temperature, pressure and movement of talker. Therefore, there is a need for an algorithm which can work effectively and robust to sparse impulse response for AEC.

This chapter includes three different novel techniques that can be applied to overcome the above challenge. It starts by describing the algorithms and then looks at their computational complexity. Finally, experimental results for these algorithms are compared with the published algorithms.

5.2 Novel Algorithms

5.2.1 Optimum Normalised Least Mean Square (ONLMS)

It is well known that the NLMS algorithm obtains very fast convergence for non-sparse impulse response due to its convergence process being essentially independent on the individual tap weights. On the other hand MPNLMS performs better in sparse system by using an optimum technique proposed in [12].

In order to exploit the advantages of theses main types of adaptive algorithm, a switching structure which combines the above two is proposed. This is motivated by the switching technique applied in PNLM++. The description of the algorithm follows as below.

As showed in [12], the tap adaptation equation of coefficients can be written as

$$\hat{\mathbf{h}}(n) = [1-(1-\lambda)^{n}]\hat{\mathbf{h}}_{opt} \tag{5.1}$$

where λ = step size $\times$ variance of input

Let n_p is the convergence time for the p^{th} coefficient to reach the $\in$- vicinity of its optimal value for a give small positive number $\in$. Then, the coefficient convergence time for the p^{th} coefficient is calculated as

$$\begin{aligned}
\hat{\mathrm{h}}_{opt,p} - [1-(1-\lambda)^{n_p}]\hat{\mathrm{h}}_{opt,p} &= \in \\
(1-\lambda)^{n_p}\hat{\mathrm{h}}_{opt,p} &= \in \\
\frac{\hat{\mathrm{h}}_{opt,p}}{\in} &= \frac{1}{(1-\lambda)^{n_p}} \\
\ln\frac{\hat{\mathrm{h}}_{opt,p}}{\in} &= \ln\frac{1}{(1-\lambda)^{n_p}} \\
\ln\frac{\hat{\mathrm{h}}_{opt,p}}{\in} &= -n_p \ln(1-\lambda) \\
n_p &= \ln\frac{\hat{\mathrm{h}}_{opt,p}}{\in} \Big/ \ln\frac{1}{(1-\lambda)}
\end{aligned} \tag{5.2}$$

Since $\in$ and λ are constants, the convergence time is only dependent on its optimal magnitude. Therefore the overall convergence time for the NLMS algorithm is defined as

$$n_{NLMS} = \ln\frac{\hat{\mathrm{h}}_{opt,\max}}{\in} \Big/ \ln\frac{1}{(1-\lambda)} \tag{5.3}$$

For $0 < \lambda < 1$ the denominator can be estimated as

$$\begin{aligned}
\ln\frac{1}{(1-\lambda)} &= \ln 1 - \ln(1-\lambda) \\
&\approx 0 - (-\lambda) = \lambda
\end{aligned}$$

Now, equation 5.3 is simplified to

$$n_{NLMS} \approx \frac{1}{\lambda}\ln\frac{\hat{\mathrm{h}}_{opt,\max}}{\in} \tag{5.4}$$

For the proportionate algorithms, a step size control factor g_p needs to be assigning to all coefficients in order to get different step sizes to different coefficients. Following from equation 5.3, coefficient convergence time for the p^{th} coefficient is calculated as

$$n_p \approx \frac{1}{\lambda g_p} \ln \frac{\hat{h}_{opt,\max}}{\in} \tag{5.5}$$

But the overall convergence time in this case is the sum of the individual tap weights. So, it can be defined as

$$n_{MPNLMS} \approx \sum_{p=1}^{L} \frac{1}{\lambda g_p} \ln \frac{\hat{h}_{opt,\max}}{\in} \tag{5.6}$$

But, the step size control vector must satisfy the following condition.

$$\sum_{p=1}^{L} g_p = L \tag{5.7}$$

Substituting equation 5.7 into 5.6 yields:

$$n_{MPNLMS} \approx \frac{1}{\lambda L} \sum_{p=1}^{L} \ln \frac{\hat{h}_{opt,\max}}{\in} \tag{5.8}$$

Since, NLMS performs better than MPNLMS in non-sparse system, it follows that

$$\begin{gathered} n_{NLMS} < n_{MPNLMS} \\ \frac{1}{\lambda} \ln \frac{\hat{h}_{opt,\max}}{\in} < \frac{1}{\lambda L} \sum_{p=1}^{L} \ln \frac{\hat{h}_{opt,\max}}{\in} \\ \ln \frac{\hat{h}_{opt,\max}}{\in} < \frac{1}{L} \sum_{p=1}^{L} \ln \frac{\hat{h}_{opt,\max}}{\in} \end{gathered} \tag{5.9}$$

By testing the above condition at each iteration, the algorithm switches to NLMS adaptive process if it satisfies the condition, otherwise employs MPNLMS technique. In reality, since the optimum tap weights are not available, the switching condition must be adapted as follows

$$\text{constant} \times \ln \frac{\hat{h}_{opt,\max}}{\in} < \frac{1}{L} \sum_{p=1}^{L} \ln \frac{\hat{h}_{opt,\max}}{\in} \tag{5.10}$$

Let the left hand side of equation 5.10 be ‘a’ and the right hand side be ‘b’, then the operation of the ONLMS can therefore be summarised as in figure 5.1.

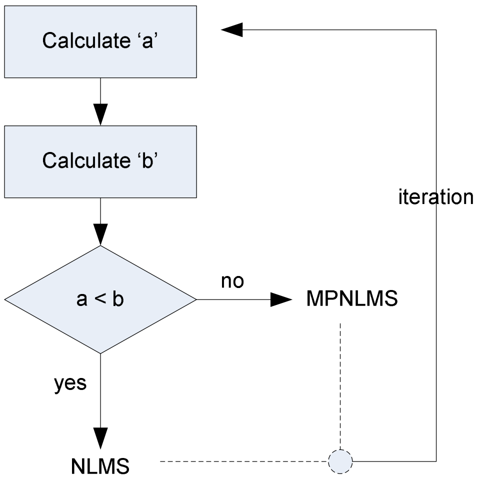


Figure 5.1: The flow chart of the ONLMS algorithm

5.2.2 Sparse-controlled MPNLMS 1 (SC-MPNLMS-1)

The experimental studies from section 4.6 tell that for efficient convergence speed the sparseness measure needs to be added in the weight update equation. In non-sparse system, the denominator of the $q_l(n)$ factor (in equation 4.8) for MPNLMS becomes higher and therefore gives less emphasise for the $q_l(n)$. As a result it convergence slows down for the non-sparse system.

In order to study this issue further, an experiment is carried out to see any relation between sparseness measure and the denominator of the $q_l(n)$. The table 5.1 summarised the results.

sparseness measure (ξ)	**denominator of the $q_l(n)$**
0.9272	0.1978
0.8423	0.2244
0.6949	0.3124
0.49	0.4877
0.4241	0.5311
0.3787	0.5594
0.2618	0.6516
0.2173	0.7341

Table 5.1: Summarised the results of denominator of the MPNLMS’s $q_l(n)$ vector for different sparseness measure.

As it can be seen from table 5.1 is that the sparseness is inversely proportional to the denominator of the $q_l(n)$ vector. The top three sparse measures (ξ) in the table result better performance for MPNLMS and the remaining ones (shaded in table) give worse performance than that of the NLMS. Therefore, in order to balance the effect of the denominator, the following modification is made to the $q_l(n)$

$$q_l(n) = \frac{k_l(n)}{\max(k_l(n), \xi)} \tag{5.11}$$

This modification allows to set $q_l(n)$ to 1 for the tap weights meeting the following condition

$$k_l(n) > \xi \tag{5.12}$$

In the non-sparse system above condition may be true for all the tap weights and as a result the above modification allows behaving as a NLMS algorithm. On the other hand, for sparse case the $q_l(n)$ denominator chooses the sparseness measure except for the few active taps with high magnitudes.

5.2.3 Sparse-controlled MPNLMS 2 (SC-MPNLMS-2)

A different technique can be followed by employing the SC-IPNLMS approach [10] to the MPNLMS. It is also taking into account the sparseness of an impulse response. As MPNLMS performs badly in sparse systems, the proposed SC-MPNLMS-2 algorithm improves the performance of MPNLMS by emphasising the proportionate term if the impulse response is significantly non-sparse. The computation of the $q_l(n)$ can be expressed as

$$q_l(n) = \frac{[1 - 0.5\hat{\xi}(n)]}{L} + [1 + 0.5\hat{\xi}(n)] \frac{k_l(n)}{\sum_{i=0}^{L-1} k_i(n)} \tag{5.13}$$

For relatively less sparse impulse responses, the SC-MPNLMS-2 allocates a higher weighting to the first term. In order to avoid the dividing by zero or a small number in the computation of sparseness measure at the early stages of the adaptive process, this adapting process can be employed for $n \geq L$. For $n < L$ the elements of the $q_l(n)$ is computed using the actual MPNLMS equation.

5.3 Computational Complexity

Achieving slightly better performance by carrying out heavier computations is not acceptable in this context. Additional computations need to be done on top of the MPNLMS computations is analysed below the three novel algorithms.

5.3.1 Optimum Normalised Least Mean Square (ONLMS)

Computation for the testing condition is needs to be done in addition to the MPNLMS computation. Even though the right hand side of the equation 5.10 involves logarithmic calculation for L times, this data is readily available from the computation of the F factor, as in the equation 4.13. Therefore, an addition of some more memory is required to save the value. However, the extra computation to calculate the left hand side of the equation and then comparison can not be avoidable. Whenever it switches to perform NLMS adaptation process its computation is reduced compare to performing MPNLMS all time.

5.3.2 Sparse-controlled MPNLMS 1 (SC-MPNLMS-1)

The additional complexity arises from the computation of the sparseness measure and comparison function. Given that $L/(L-\sqrt{L})$ can be computed off line, remaining norm1 and norm2 functions are needed to compute, which will take additional $2L$ additions.

5.3.3 Sparse-controlled MPNLMS 2 (SC-MPNLMS-2)

On top of the MPNLMS computation, the sparseness measure computation needs to be done as stated in the SC-MPNLMS-1 process. Other than that L+2 multiplications 2 additions and one division is required to calculate the weighting vector, according to equation 5.13.

5.4 Comparisons and Discussions

The NLMS, PNLMS, IPNLMS, MPNLMS, SC-MPNLMS-1 and SC-MPNLMS-2 algorithms are analysed experimentally in this section.

The step size for the both SC-MPNLMS are adjusted in order to achieve the same steady state NPM as that for NLMS, PNLMS, IPNLMS and MPNLMS. These correspond to $\mu_{\text{SC-MPNLMS-1}} = \mu_{\text{SC-MPNLMS-2}} = 0.4$. Figure 5.2 shows convergence result for all these published and novel algorithms.

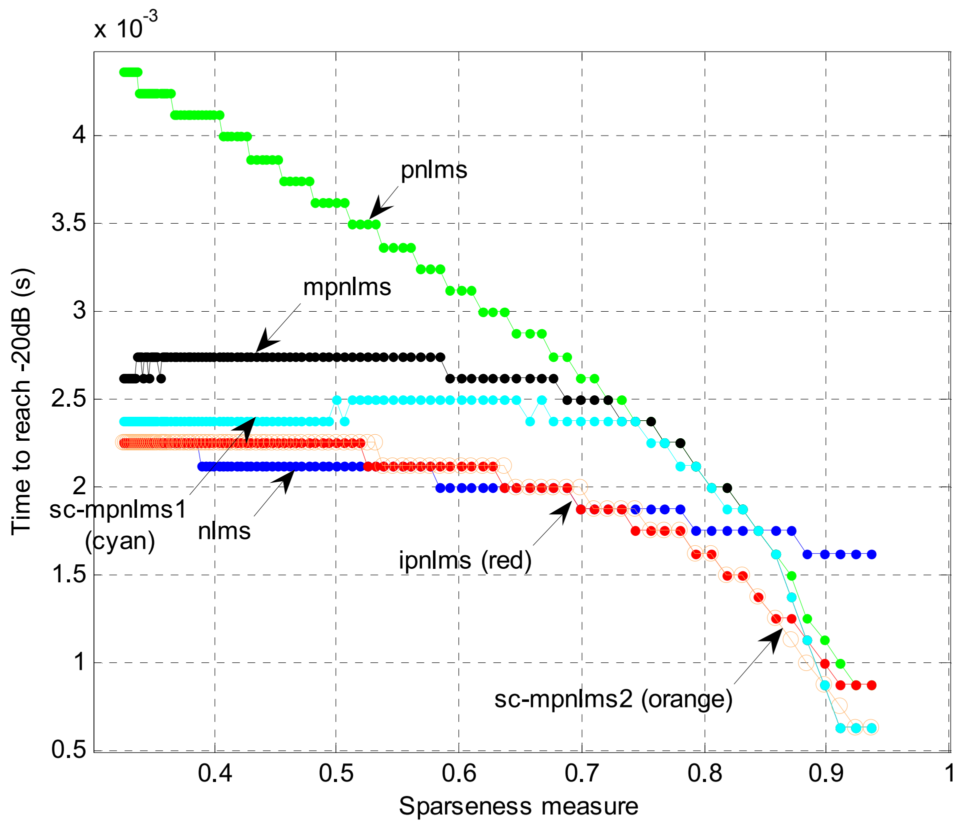


Figure 5.2: Plot of Time to reach the -20dB NPM against different sparseness measures of 100 systems for NLMS, PNLMS, IPNLMS, MPNLMS, SC-MPNLMS-1 and SC_MPNLMS-2.

SC-MPNLMS-1 performs almost similar to MPNLMS in sparse system and gives acceptable performance close NLMS and IPNLMS in sparse system. This is because in non-sparse system all (or most) of the $k_l(n)$ are greater than ξ, therefore $q_l(n) = 1$ as in the case of NLMS.

SC-MPNLMS-2 gives an overall best performance throughout all values of ξ. It achieves this desired result by taking into account the ξ, which was a forgotten factor in the main existing algorithms.

5.5 Summary

It has been shown that attaching the sparseness measure (ξ) into the tap update equation can exhibit promising performance in all ξ level for AEC, in terms of NPM. But, extra computations need to be done to achieve this.

CHAPTER 6

Conclusion

6.1 Conclusion

This project has addressed the significant problem caused by undesirable echoes that result from coupling between the loud speakers and microphones in the near end room. The research for this project focuses on the development of the adaptive filtering algorithms for sparse and non-sparse systems, emphasising on the achievement of fast convergence rate with relatively low computational cost.

The trade-off between convergence speed and the steady state misalignment is an important issue in this context, and can be balanced by choosing a sensible step size for the adaptive processes.

A series of experiments carried out both within and across NLMS, as well as a few other proportionate techniques, namely PNLMS, IPNLMS and MPNLMS, help to investigate their strengths and weaknesses. NLMS gives better performance in non-sparse system, whereas MPNLMS performs well in sparse impulse response. The combination of non-sparse and sparse technique, IPNLMS, exhibits an overall of better performance in all sparse levels. This identified an important factor – the sparseness measure (ξ), which affects their convergence speed.

By introducing ξ together with employing MPNLMS approach, adaptive algorithms for acoustic echo cancellation can achieve fast convergence and robustness to sparse impulse response. The proposed algorithms, known as SC-MPNLMS-1 and SCMPNLMS-2, take into account this factor differently via the coefficient update function. Simulation results show that the SC-MPNLMS-2 exhibits more robustness to sparse systems than the other techniques. Though their computational complexities are slightly higher than the existing main stochastic algorithms, both new algorithms perform better in all ξ levels.

6.2 Future Works

The work within this thesis could be further extended in a number of directions, as listed below.

- Early works for this project is found that the different initialisations give different convergence time. The figure 6.1 shows the convergence speed of the NLMS adaptive process when the tap weights are initialised to a) zeros, b) ones and c) random numbers. Therefore, optimum weights initialisation should be further investigated.

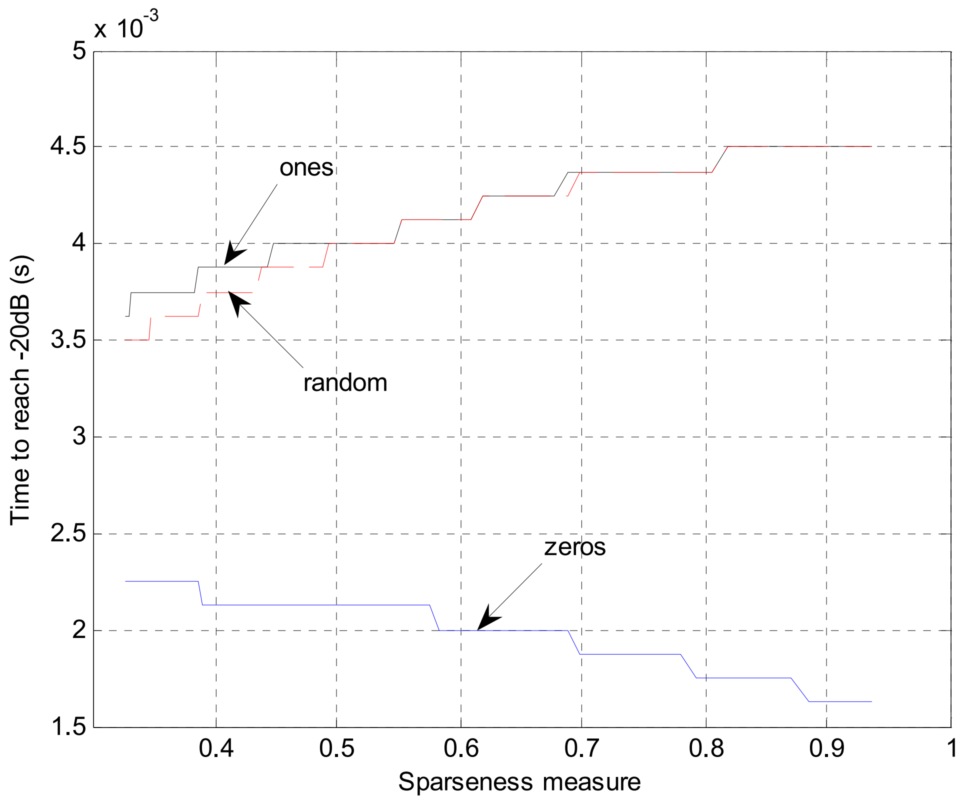


Figure 6.1: Time to reach the -20 dB NPM level vs. the sparseness measure using different taps initialisations.

- The theory behind the ONLMS algorithm provides good reasoning to obtain optimum performance independent on the sparse level. 3.33 for the constant in the test condition provides a good result over NLMS and PNLMS, although, further exploration is needed to make the constant value optimum, so that its performance is optimum throughout all sparse measure (ξ).

- Throughout this project all the algorithms are tested using zero mean white Gaussian noise. It is preferred to check these algorithms with coloured signal. In order to compare their performances with each other the step sizes for all algorithms need to be set to achieve same steady state NPM level. Some testing is already initiated by using the coloured signal presented in Appendix 2 and found that the step size of NLMS and IPNLMS should be set to 0.4, while 0.6 for PNLMS.

Reference

Thesis

[1] Sunit Patel, *Echo cancellation for sparse systems with applications to IP gateways*, MSc thesis, 2004.

[2] Andy W.H. Khong, *Adaptive Algorithms Employing Tap Selection For Single Channel and Stereophonic Acoustic Echo Cancellation*, Thesis, 2006.

Websites

[3] *Echo Cancellation* – International Engineering Consortium, web Proforums http://www.iec.org/online/tutorials/echo_cancel/topic03.html

Textbooks

[4] Bernard Mulgrew, Colin F N Cowan, *Adaptive filters and equalisers,* Kluwer Academic Publishers, Boston/Dordrecht/London, 1988.

[5] S. Haykin, *Adaptive Filter Theory*, 4th edition, Information and System Sciences series. Prentice Hall, 2002.

[6] Paulo S Diniz, *Adaptive filtering – Algorithms and Practical Implementations,* Second Edition, Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[7] Patrick A Naylor, Jingjing Cui, Mike Brookes, *Adaptive algorithms for sparse echo cancellation,* Signal Processing, Elsevier.

[8] P O Hoyer, “Non-negative matrix factorization with sparseness constraints”, *Journal of Machine learning research*, vol. 5, Nov. 2004

[9] J Benesty, Y A Huang, J Chen and P A Naylor, “Adaptive algorithmds for the identification of sparse impulse responses”, *in selected methods for acoustic echo and noise control,* 2006, ch. 5.

[10] Andy W.H. Khong, Patrick A Naylor, “Efficient use of Sparse Adaptive Filters,” *Signal Processing Letters*, IEEE.

[11] Donald L Duttweiler, “Proportionate Normalized Least-Mean Squares Adaptation in Echo Cancellers”, *Transactions on speech and Audio processing*, IEEE, Sep. 2000.

[12] Hongyang Deng, Doroslovacki, M., “Improving convergence of the PNLMS algorithm for sparse impulse response identification”, *Signal Processing Letters*, IEEE, March 2005.

[13] Dennis R. Morgan, “On the Evaluation of Estimated Impulse Responses”, *Signal Processing Letters*, IEEE, July 1998.

[14] Jacob Benesty and Steven L Gay, “An Improved PNLMS algorithm”, IEEE, 2002.

Appendix 1

MATLAB Simulation Codes

NLMS

```
function [e,npm] = nlms (input, d, mu, h)
%   d     = desired signal
%   L     = weight tap number
%   mu    = step size of LMS filter
%   x     = weight vector input

x         = zeros(length(h),1);
randn('state', 5);
weight    = 0*randn(length(h),1);
e         = zeros(length(input),1);
npm       = zeros(length(input),1);

epsilon   = sum(input.^2)/length(input);

for n= 1: length(input);
   x           = [input(n); x(1:end-1)];
   output(n)   = weight' * x;
   e(n)        = d(n) - output(n);
   weight      = weight + (mu/ (epsilon + x'*x)) * x * e(n);
   npm(n)                                  =       norm(h-
(((h'*weight)/(weight'*weight))*weight))/norm(h);
end
```

PNLMS

```
function [e,npm] = pnlms (input, d, mu, h);
%    [wevn,e] = nlms (input, d, mu, h);
%   d     = desired signal
%   L     = weight tap number
%   mu    = step size of LMS filter
%   x     = weight vector input

x         = zeros(length(h),1);
% randn('state', 5);
weight    = ones(length(h),1);
e         = zeros(length(input),1);
npm       = zeros(length(input),1);

epsilon   = sum(input.^2)/(length(input)*length(h));
```

```
for n= 1: length(input);
   x           = [input(n); x(1:end-1)];
   output(n)   = weight' * x;
   e(n)        = d(n) - output(n);

   a = max([0.01;abs(weight)]);
   k = max((5/length(h))*a,abs(weight));
   Q = length(h)*(k/sum(k));

   weight      = weight + ((mu*e(n)*(Q.*x))/((x'*(Q.*x))+ epsilon));

   npm(n)      = norm(h-
(((h'*weight)/(weight'*weight))*weight))/norm(h);
end
```

IPNLMS

```
function [e,npm] = ipnlms (input, d, mu, h, a);
%   d    = desired signal
%   L    = weight tap number
%   mu   = step size of LMS filter
%   x    = weight vector input

x        = zeros(length(h),1);
weight   = zeros(length(h),1);
e        = zeros(length(input),1);
npm      = zeros(length(input),1);

% a = -0.75;  % alpha (0,-0.5,-0.75)
epsilon  = (1-a)*(sum(input.^2)/(length(input)*2*length(h)));

for n= 1: length(input);
   x           = [input(n); x(1:end-1)];
   output(n)   = weight' * x;
   e(n)        = d(n) - output(n);

   Q                             =                              ((1-
a)/(2*length(h)))+(((1+a)*abs(weight))/((2*sum(abs(weight)))+eps));

   weight      = weight + ((mu*e(n)*(Q.*x))/((x'*(Q.*x))+ epsilon));

   npm(n)                                                 =       norm(h-
(((h'*weight)/(weight'*weight))*weight))/norm(h);
end
```

MPNLMS

```
function [e,npm] = mpnlms (input, d, mu, h,alp);
%   [wevn,e] = nlms (input, d, mu, h);
%   d    = desired signal
%   L    = weight tap number
%   mu   = step size of LMS filter
%   x    = weight vector input

x        = zeros(length(h),1);
```

```
weight   = zeros(length(h),1);
e        = zeros(length(input),1);
npm      = zeros(length(input),1);

%a = -0.5;  % alpha (0,-0.5,-0.75)
epsilon  = (1-alp)*(sum(input.^2)/(length(input)*2*length(h)));

for n= 1: length(input);
   x           = [input(n); x(1:end-1)];
   output(n)   = weight' * x;
   e(n)        = d(n) - output(n);

   delta =  1/0.001;
   F = log(1+delta*(abs(weight)))/log(1+delta);
   a = max([0.01;F]);
   k = max((5/length(h))*a,F);
   Q = length(h)*(k/sum(k));

   weight      = weight + ((mu*e(n)*(Q.*x))/((x'*(Q.*x))+ epsilon));

   npm(n)                                            =       norm(h-
(((h'*weight)/(weight'*weight))*weight))/norm(h);
end
```

SC-MPNLMS-1

```
function [e,npm] = scmpnlms1 (input, d, mu, h);
%   [wevn,e] = nlms (input, d, mu, h);
%   d    = desired signal
%   L    = weight tap number
%   mu   = step size of LMS filter
%   x    = weight vector input

x        = zeros(length(h),1);
weight   = zeros(length(h),1);
e        = zeros(length(input),1);
npm      = zeros(length(input),1);

a = 0;  % alpha (0,-0.5,-0.75)
epsilon  = (1-a)*(sum(input.^2)/(length(input)*2*length(h)));

for n= 1: length(input);
   x           = [input(n); x(1:end-1)];
   output(n)   = weight' * x;
   e(n)        = d(n) - output(n);

   delta =  1/0.001;
   F = log(1+delta*(abs(weight)))/log(1+delta);
   a = max([0.01;F]);
   k = max((5/length(h))*a,F);

   len = length(h);
   if n<len
       Q = len*(k/sum(k));
   else
       sp  = (len/(len-sqrt(len)))* (1-
(sum(abs(weight))/(sqrt(len)*sqrt(sum(abs(weight).*abs(weight)))))) ;
       Q = (k./max(k,sp));
   end;
   weight      = weight + ((mu*e(n)*(Q.*x))/((x'*(Q.*x))+ epsilon));

   npm(n)      = norm(h-
(((h'*weight)/(weight'*weight))*weight))/norm(h);
end
```

SC-MPNLMS-2

```
function [e,npm] = scmpnlms2 (input, d, mu, h);
%   [wevn,e] = nlms (input, d, mu, h);
%   d    = desired signal
%   L    = weight tap number
%   mu   = step size of LMS filter
%   x    = weight vector input

x        = zeros(length(h),1);
% randn('state', 5);
weight   = zeros(length(h),1);
e        = zeros(length(input),1);
npm      = zeros(length(input),1);

al = -0.5;  % alpha (0,-0.5,-0.75)
epsilon  = (1-al)*(sum(input.^2)/(length(input)*2*length(h)));

for n= 1: length(input);
   x           = [input(n); x(1:end-1)];
   output(n)   = weight' * x;
   e(n)        = d(n) - output(n);

   delta =  1/0.001;
   F = log(1+delta*(abs(weight)))/log(1+delta);
   a = max([0.01;F]);
   k = max((5/length(h))*a,F);

   len = length(h);

   if (n > length(h))
       sp                 =         (len/(len-sqrt(len)))*         (1-
(sum(abs(weight))/(sqrt(len)*sqrt(sum(abs(weight).*abs(weight)))))) ;
       Q = ((1+(0.5*sp))*(k/sum(k))) + ((1-(0.5*sp))/length(h));
   else
       Q = length(h)*(k/sum(k));
   end;
```

Sparse Impulse Response Generator

```
function h = SIRgen (Lh, Lp, tau, b, p)

Lu   = Lh - Lp;
f    = exp(-(1/tau)*[0:Lu-1]');
u    = [zeros(Lp,1); f];

B    = diag(b);
```

```
top  = [zeros(Lp,Lp) zeros(Lp,Lu)];
bot  = [zeros(Lu,Lp) B];

h    = [top; bot]*u+p;
h    = h./max(h);
```

Noise Generator With Specified SNR Level

```
function  [corruptsignal,  newnoise,  confirmSNR]=  gennoise(signal,
noise, SNR);
%
*********************************************************************
**
%   This  function  generates  a  new  corrupted  signal  based  on  a  given
SNR.
%   It also generates the new noise sequence as output.
%   New SNR is confirmed via confirmSNR.
%    Usage:  [newnoise,  corruptsignal,  confirmSNR]=  gennoise(signal,
noise, SNR);
%
*********************************************************************
**


signalpow=(signal'*signal)/length(signal);
noisepow= (noise'*noise)/length(noise);

newnoise= sqrt((signalpow/((10^(SNR*0.1))*noisepow)))*noise;

corruptsignal= signal+newnoise;
confirmSNR= 10*log10(signalpow/((newnoise'*newnoise)/length(noise)));
```

Run File To Plot T20 Vs. Sparseness Measure

```
clear all;

fs      = 8000;
SimTim  = 2.5;
mu      = 0.4;
SNR     = 30;

Lh      = 256;
Lp  = 30;
tau     = 8; % smaller tau, more sparse
colour  = ['b','m','r','k'];
wbar    = waitbar(0,'NLMS + PNLMS + IPNLMS + MPNLMS');

for count = 1:2 % 4 different algorithm
    ind = zeros(1,100);
    XX  = zeros(1,100);
      for cc = 1:100 % 100 different mu values 0.1 - 1.0

            randn('state', 6);
            B           = randn((Lh-Lp),1);

        con = (cc-1)* 0.005;
```

```
            randn('state', 16);
            P           = con * randn(Lh,1); % inactive taps
        XX(cc)  = std(P);

            h           = SIRgen (Lh, Lp, tau, B, P);

            for trial = 1:10 % 10 experiments
                waitbar((((cc-1)*10)+trial)/1000);
                  randn('state', 50+trial);
                  source  = randn(fs*SimTim,1);

                y             = filter(h,1,source);

                  randn('state', 30+trial);
                  noise   = randn(length(y),1);
                  y       = gennoise(y, noise, SNR);
                  yy(:,trial) = y;

                  if (count ==1)
                        [e,npm] = scmpnlms1 (source, y, 0.4, h);
                  else
                        [e,npm] = scmpnlms1 (source, y, 0.7, h);
                  end;

                  ee(:,trial) = e;
                nnpm(:,trial) = npm;
            end;

            e     = mean(ee')';
            y     = mean(yy')';
            npm   = mean(nnpm')';


            a = [1/3;2/3;3/3];

            NPM = zeros(length(npm)/100,1);
            for i=1:(length(npm)/100)
                data   = npm((i*100)-100+1:(i*100));
                NPM(i) = mean(data);
            end;


%           NPM = filter(a,1,NPM);
        z   = 20*log10(NPM);
        zz = find(z<=-20);  % finding the T20
        ind(cc) = zz(1);

    end;
    figure (1)
    plot(XX,ind/fs,colour(count));
    xlabel('standard deviation of inactive taps');
      ylabel('Time to reach -20dB (s)');
    hold on;
    grid on;

end;
close(wbar);
```

Run File to Plot MSE, ERLE and NPM

```
clear all;

fs      = 8000;
SimTim  = 2.5;
mu      = 0.3;
SNR     = 30;

Lh      = 256;
tau     = [Inf 100 40 8]; % smaller tau, more sparse
colour  = ['b','g','r','k','c'];
wbar     = waitbar(0,'NLMS + PNLMS + IPNLMS + MPNLMS');

for cc =1:4 % different sparse
      for count = 1:4 % different algorithm
            for trial = 1:10 % ensample
                waitbar((((cc-1)*50)+((count-1)*10)+trial)/200);
                  randn('state', 50+trial);
                  source  = randn(fs*SimTim,1);

                if (tau(cc) == Inf)
                    Lp  = 0;
                else
                    Lp  = 30;
                end;

                  randn('state', 6+trial);
                  B           = randn((Lh-Lp),1);
                randn('state', 16+trial);
                  P           = 0.005*randn(Lh,1);

                  h           = SIRgen (Lh, Lp, tau(cc), B, P);
                  hh(:,trial) = h;
                  y           = filter(h,1,source);

                  randn('state', 30+trial);
                  noise   = randn(length(y),1);
                  y       = gennoise(y, noise, SNR);
                  yy(:,trial) = y;

                  if (count ==1)
                        [e,npm] = nlms (source, y, mu, h);
                  elseif (count ==2)
                        [e,npm] = nlms (source, y, mu, h);
              elseif (count ==3)
                        [e,npm] = mpnlms (source, y, mu, h);
            elseif (count ==4)
                        [e,npm] = scmpnlmsx (source, y, mu, h);
                  else
                        [e,npm] = optimum (source, y, mu, h);
                  end;

                  ee(:,trial) = e;
                nnpm(:,trial) = npm;
            end;

            e     = mean(ee')';
            y     = mean(yy')';
            npm    = mean(nnpm')';
            h      = mean(hh')';

            MSE = zeros(length(e)/100,1);
            for i=1:(length(e)/100)
                data   = e((i*100)-100+1:(i*100)).^2;
                MSE(i) = mean(data);
            end;

            a = [1/3;2/3;3/3];
            MSE = filter(a,1,MSE);
            figure (cc)
            subplot 221;
            ax1  = 0:100:100*(length(MSE)-1);
            plot(ax1,10*log10(MSE),colour(count));
            xlabel('number of iterations (n)');
            ylabel('MSE (dB)');
            hold on;

            SERLE = zeros(length(e)/100,1);
            for i=1:(length(e)/100)
                E = e((i*100)-100+1:(i*100)).^2;
                Y = y((i*100)-100+1:(i*100)).^2;
                SERLE(i) = sum(Y)/sum(E);
            end;

            SERLE = filter(a,1,SERLE);
            subplot 222;
            ax2  = 0:100:100*(length(SERLE)-1);
            plot(ax2,10*log10(SERLE),colour(count));
            xlabel('number of iterations (n)');
            ylabel('ERLE (dB)');
            hold on;

            NPM = zeros(length(npm)/100,1);
            for i=1:(length(npm)/100)
                data   = npm((i*100)-100+1:(i*100));
                NPM(i) = mean(data);
            end;

            NPM = filter(a,1,NPM);
            subplot 223;
            ax3  = 0:100:100*(length(NPM)-1);
            plot(ax3,20*log10(NPM),colour(count));
            xlabel('number of iterations (n)');
            ylabel('NPM (dB)');
            hold on;
      end;

      figure (cc);
      axx(1) = subplot (2,2,1);
      axx(2) = subplot (2,2,2);
      axx(3) = subplot (2,2,3);
      linkaxes(axx,'x');

      subplot 224;
      plot(0:(Lh-1),h);
end;
```

Appendix 2

Coloured Signal

In order to generate the coloured signal (i.e. a signal with different levels of frequency), an all-pole IIR (Infinite Impulse Response) filter is used with a complex conjugate pair of poles. The position of the poles is chosen to be 0.9 from the origin of the z-plane to gain a small over-shoot at about 2 kHz. The numerator of the filter coefficient is chosen to be 1 and the denominator is obtained as [`1 -1.732 0.827`], using the MATLAB in-built filter design tool box, to meet the design specifications. Figure 4.8 shows the frequency response of the two types of **x**(*n*).

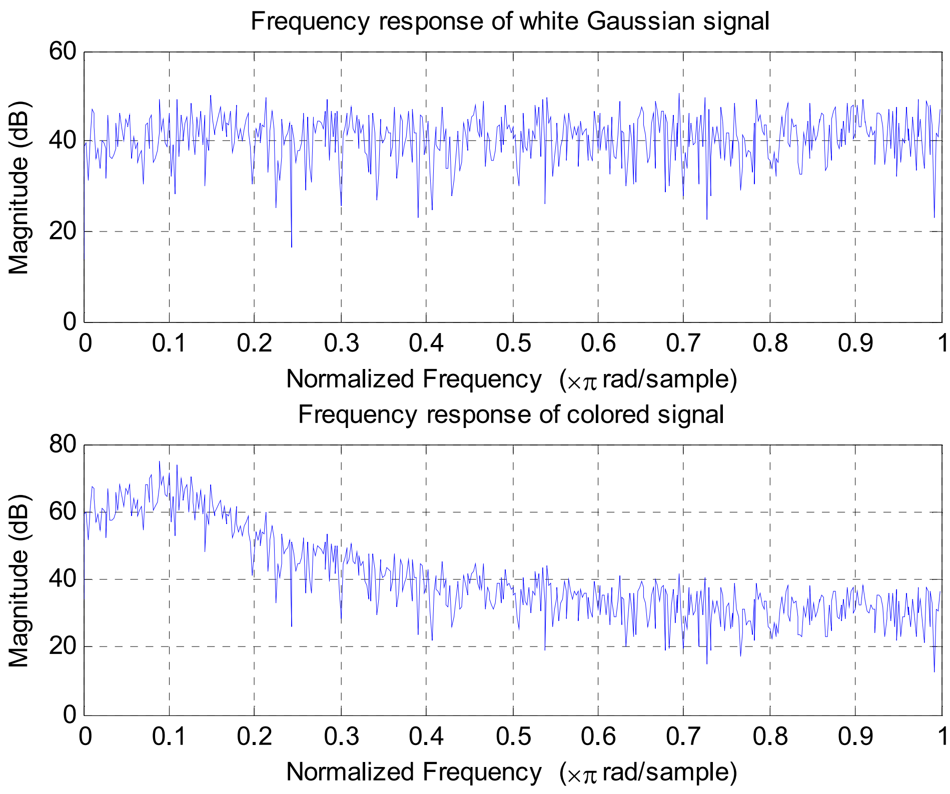


Figure 4.8: Frequency response of the two types of input signal, **x**(*n*).