SWITCHING ADAPTIVE FILTER STRUCTURES FOR IMPROVED PERFORMANCE

by

Gaguk Zakaria

Thesis submitted to the Faculty of the
Bradley Department of Electrical Engineering
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirement for the degree of

Master of Science
in
Electrical Engineering

APPROVED:

A. A. (Louis) Beex, Chairman

R. L. Moose

Brian D. Woerner

December 1993
Blacksburg, Virginia

C.2

SWITCHING ADAPTIVE FILTER STRUCTURES FOR IMPROVED PERFORMANCE

by
Gaguk Zakaria

Committee Chairman: A. A. (Louis) Beex
Electrical Engineering

ABSTRACT

We describe an adaptive filter system that is able to switch between adaptive filter algorithms in order to produce fast convergence and low *mean square* error (MSE). The switching system employs two adaptive filters for two different tasks; one is intended to yield fast convergence, and logically called the "fast convergence structure", while the other is intended to give small MSE, and called the "low MSE structure".

The switching from one algorithm to the other is determined by the state of the system. For example, switching from the "fast convergence structure" to the "low MSE structure" happens if the former has reached its steady state according to some pre-defined criterion, while switching from the "low MSE structure" to the "fast convergence structure" happens if the former starts diverging according to some pre-defined criterion. We define here that an algorithm has reached its steady state if the average of the square of its output error is small and approximately constant for several iterations. After an algorithm has reached its steady state, not much additional error reduction can be obtained from it, so that there is no payoff using "the fast convergence structure", which is usually more computation intensive than the "low MSE structure". In this situation it would be better to use the Least Mean Squares

(LMS) algorithm as the "low MSE structure" because of its simplicity and its numerical robustness.

Experiments using the *recursive-least-squares-lattice* (RLSL) algorithm together with the LMS algorithm, the fast-transversal-filter (FTF) algorithm together with the LMS algorithm, and the gradient-adaptive-lattice (GAL) algorithm together with the LMS algorithm for a system identification application, in particular for echo cancellation, show the expected result of providing faster convergence and lower mean square error than would be possible with a single algorithm. The switching system demonstrates other important results: it can avoid the numerical instability of some algorithms, such as the RLSL and FTF algorithms, without adding any additional computations; it is able to handle a change in the unknown system, as long as it settles, without suffering a slow convergence rate caused by an incorrect initial condition; it is able to handle a change of the observation noise without facing a divergence problem; and it is able to produce an optimum result even for *ill-conditioned* input signals, i.e. the ratio of the maximum and minimum eigenvalue of the auto-correlation of the signal is high.

When switching to the "low MSE structure" we also apply a computationally reduced order technique, in which only the values of the impulse response that are greater than some threshold are used for computation. This technique is applied to the switching structure of the recursive-least-squares-lattice algorithm together with the LMS algorithm and exhibits fast convergence and low MSE even for *ill-conditioned* input signals. For the white Gaussian noise input, on the other hand, this technique yields a somewhat larger mean square error.

ACKNOWLEDGMENT

I would like to thank my advisor Dr. A. A. (Louis) Beex for his patience in revising my thesis, his helpful suggestions and directions that I needed to finish this thesis. I also would like to thank Dr. R. L. Moose and Dr. B. D. Woerner for serving on my committee.

I also would like to thank my mother, my brothers and my sister for supporting me. I wish to thank my special friends, Diah Lukmono and Yuda A. Anriany, who do not mind spending their time to discuss anything with me and always encourage me. I also would like to thank my lab mates in the DSP Research Lab. I wish to thank Dr. H. Chang who introduced me to the field of digital signal processing and adaptive filtering 6 years ago.

Finally, the most important thing, I would like to thank Allah, the Creator, for without His willing nothing can happen.

TABLE OF CONTENTS

CHAPTER ONE INTRODUCTION 1
CHAPTER TWO THEORETICAL BACKGROUND 6
2.1 Adaptive Filters 6
2.1.1 System Identification 7
2.1.2 Echo Cancellation 8
2.1.3 The Experimental Data Sets 12
2.2 The Transversal Least Mean Squares Algorithm 15
2.2.1 Convergence Indicator for the LMS Algorithm 20
2.2.2 Effect of the Initial Value 21
2.3 The Method of Least Squares 26
2.3.1 The Transversal Recursive Least Squares Algorithm 27
2.3.1.1 Convergence of the Tap-weight 31
2.3.1.2 The RLS Mean Square Error 32
2.3.2 The Fast Transversal Filter (FTF) Algorithm 34
2.3.2.1 Initialization of the FTF Algorithm 35
2.4 Lattice Structure 38
2.4.1 The Gradient Adaptive Lattice Joint Process Algorithm 40
2.4.1.1 The MSE for the Multi-stage GAL-JP Algorithm 43
2.4.2 The Recursive Least Squares Lattice Joint Process Algorithm 48
2.4.2.1 The MSE for the Multi-stage RLSL-JP Algorithm 50
2.5 Conversions from the Lattice Joint Process Structure to the Transversal Structure 55
CHAPTER THREE SIMULATION DESCRIPTION 60
3.1 Specification and Simulation Setup 60
3.1.1 Rate of Convergence 60
3.1.2 Cancellation at Steady-state 62
3.2 Reducing the Computational Order of the Adaptive Filter 64
3.3 Simulation 69
3.3.1 Convergence Test and Switching Threshold 70
3.3.2 Switching During Operation 71

CHAPTER FOUR SIMULATION RESULTS 78
4.1 Switching Using the RLSL-JP/TLMS Algorithm 78
4.1.1 Results Using the White Gaussian Noise Input 78
4.1.2 Results Using the Colored Noise Input 81
4.1.3 Results Using the Computationally Reduced Order Technique 85
4.2 Switching Using the FTF/TLMS Algorithm 90
4.2.1 Results Using the White Gaussian Noise Input 90
4.2.2 Results Using the Colored Noise Input 93
4.3 Switching Using the GAL-JP/TLMS Algorithm 98
4.3.1 Results Using the White Gaussian Noise Input 98
4.3.2 Results Using the Colored Noise Input 101
CHAPTER FIVE CONCLUSION AND FURTHER RESEARCH 105
5.1 Conclusion 105
5.2 Further Research 107
APPENDIX A 109
The Gain-Normalized Fast Transversal Filter Algorithm 109
APPENDIX B 111
MSE and Convergence Analysis for the GAL-JP Algorithm 111
APPENDIX C.1 125
The RLSL Algorithm using a Priori Error Feedback 125
APPENDIX C.2 128
MSE and Convergence Analysis for the RLSL-JP Algorithm 128
C.2.1 The MSE for the Multi-stage RLSL-JP Algorithm 136
APPENDIX D 139
Decibel Units 139
REFERENCES 140
VITA 144

LIST OF FIGURES

Figure 2.1.1 System Identification 8
Figure 2.1.2 Typical Telephone Connection with Echo Cancellers 10
Figure 2.1.3 Relation of Signals 10
Figure 2.1.4 Typical Impulse Response of the Echo-path 11
Figure 2.1.5 The AR Model of Order 2 14
Figure 2.2.1 The LMS Algorithm 15
Figure 2.2.2 Squared Error Averaging Circuit 22
Figure 2.2.3 The MSE for the LMS Algorithm with Zeros as the Initial Condition 23
Figure 2.2.4 The MSE for the LMS Algorithm with One as the Initial Condition 23
Figure 2.2.5 The MSE for the LMS Algorithm with the Optimum as the Initial Condition 24
Figure 2.3.1 Transversal Filter 27
Figure 2.3.2 Fast Transversal Filter 35
Figure 2.4.1.a Single Stage of Lattice Structure 38
Figure 2.4.1.b Lattice Joint Process Estimator 39
Figure 2.4.2 The MSE for the GAL-JP Algorithm for a Forgetting Factor of 0.999, and an Order of 32 45
Figure 2.4.3.a Trajectory of the Third Regression Coefficient 47
Figure 2.4.3.b A Closer Look at the Trajectory of the Third Regression Coefficient 47
Figure 2.4.4 The Conversion Factors of the RLSL-JP Algorithm for Different Stages 53
Figure 2.4.5 The MSE for the RLSL-JP Algorithm; α=.95 53
Figure 2.4.6.a Trajectory of the Regression Coefficient of the RLSL-JP Algorithm at Stage 2 54
Figure 2.4.6.b A Closer Look at the Trajectory of the Regression Coefficient of the RLSL-JP Algorithm 54
Figure 2.5.1 Backward Prediction Error Filter 56
Figure 2.5.2 Transversal Filter Equivalent to Lattice Joint Process 58
Figure 3.1.1 Attenuation in a Telephone System 61
Figure 3.1.2 Specification of Residual Echo Level 63
Figure 3.2.1.a The Fifth Order Filter 65

Figure 3.2.1.b The Computationally-Reduced Order Filter 65
Figure 3.3.1 Reaching the Convergence State 72
Figure 3.3.2 Gain Change of the System to be Identified 73
Figure 3.3.3 Diagram of the Switching Structure 76
Figure 4.1.1 Two Trial Average MSE for the RLSL-JP Algorithm 79
Figure 4.1.2 Two Trial Average MSE for the RLSL-JP/TLMS Algorithm 80
Figure 4.1.3 Ten Trial Average MSE for the RLSL-JP Algorithm
gain change at iteration 2,500 81
Figure 4.1.4 Ten Trial Average MSE for the RLSL-JP/TLMS algorithm
gain change at iteration 2,500 81
Figure 4.1.5 Five Trial Average MSE for the RLSL-JP Algorithm
colored noise input 82
Figure 4.1.6 Five Trial Average MSE for the RLSL-JP/TLMS Algorithm
colored noise input 83
Figure 4.1.7 Two Trial Average MSE for the RLSL-JP Algorithm
colored noise input, gain change at iteration 4,000 84
Figure 4.1.8 Two Trial Average MSE for the RLSL-JP/TLMS Algorithm
colored noise input, gain change at iteration 4,000 85
Figure 4.1.9 Five Trial Average MSE for the RLSL-JP/TLMS Algorithm
Computationally Reduced Order Technique 86
Figure 4.1.10 Two Trial Average MSE for the RLSL-JP/TLMS Algorithm
Computationally Reduced Order Technique; colored noise input 89
Figure 4.1.11 Impulse Response of the TLMS Algorithm
Computationally Reduced Order Technique; colored noise input 89
Figure 4.2.1 Ten Trial Average MSE for the FTF/TLMS Algorithm 90
Figure 4.2.2 TenTrial Average MSE for the FTF Algorithm 91
Figure 4.2.3 Ten Trial Average MSE for the FTF/TLMS Algorithm
gain change at iteration 4,000 92
Figure 4.2.4 Ten Trial Average MSE for the FTF Algorithm
gain change at iteration 4,000 93
Figure 4.2.5 Ten Trial Average MSE for the FTF/TLMS
colored noise input 94
Figure 4.2.6 Ten Trial Average MSE for the FTF Algorithm
colored noise input 94

Figure 4.2.7 Fifteen Trial Average MSE for the FTF/LMS Algorithm
colored noise input, gain change at iteration 2,500 96
Figure 4.2.8 Fifteen Trial Average MSE for the FTF/TLMS Algorithm
colored noise input, gain change at iteration 2,500 97
Figure 4.3.1 Eight Trial Average MSE for the GAL-JP/TLMS Algorithm 98
Figure 4.3.2 Eight Trial Average MSE for the GAL-JP Algorithm 99
Figure 4.3.3 Eight Trial Average MSE for the GAL-JP/TLMS Algorithm
gain change at iteration 4,000 100
Figure 4.3.4 Eight Trial Average MSE for the GAL-JP Algorithm
gain change at iteration 4,000 100
Figure 4.3.5 Eight Trial Average MSE for the GAL-JP/TLMS Algorithm
colored noise input 101
Figure 4.3.6 Eight Trial Average MSE for the GAL-JP Algorithm
colored noise input 102
Figure 4.3.7 Eight Trial Average MSE for the GAL-JP/TLMS Algorithm
colored noise input, gain change at iteration 6,500 103
Figure 4.3.8 Eight Trial Average MSE for the GAL-JP Algorithm
colored noise input, gain change at iteration 6,500 103

CHAPTER ONE

INTRODUCTION

Adaptive filtering presents an appropriate solution in signal processing where statistical information of the environment is not available [16]. Unlike the *fixed filter*, the adaptive filter develops its parameters *on-the-fly*, using the observable data, which gives it great flexibility to operate in an unknown environment.

There is an inherent problem facing the adaptive filter: an adaptive filter aimed at fast convergence usually exhibits large *mean square error* (MSE), while, on the other hand, an adaptive filter aimed at low MSE usually exhibits slow convergence. Several papers have been published regarding these issues, yet no satisfying results have emerged [5], [25], [38].

In this thesis, we try to solve the problem from a different angle. Instead of trying to improve an adaptive filter algorithm/structure, we develop an adaptive filter system, called a switching structure, consisting of two adaptive filter algorithms/structures. Each adaptive filter in this system is specified for a different task, one is intended to produce fast convergence, called the "fast convergence structure", and the other is intended to yield low MSE, called the "low MSE structure". The switching structure can choose one of the structures based on a predefined criterion. For example, during the transient state, the switching structure uses the "fast convergence structure" to model the unknown system as fast as possible. In other words, it tries to reduce the error as fast as possible. Then, after the "fast convergence structure" has converged, the

switching structure switches to the "low MSE structure" to get additional error reduction and to have a more stable algorithm.

The adaptive filter algorithms/structures used as the "fast convergence structure" are the adaptive filters that are less affected by the eigenvalue ratio of the input data. The adaptive filter algorithms/structures used as the "low MSE structure" are the adaptive filters that are numerically stable and, of course, produce low MSE. It turns out that *least-squares based* and *lattice based* adaptive filters are good choices for the "fast convergence structure" since their fast convergence characteristics are almost not affected by the eigenvalue ratio of the input signal, even though there can be problems with the numerical stability for implementations in finite word length. On the other hand, the LMS algorithm is a good choice for the "low MSE structure" since it is numerically stable and produces low MSE.

We simulate the switching structure for a system identification problem, i.e. echo cancellation. For this application the switching structure tries to model the unknown system using the input and output data of that system. Echo cancellation is usually used to remove the echo in telephone connections using satellite communication. As we know, the voices of the speakers on long distance connections via satellite very often are returned to them; they hear their own voices return, after being delayed, and this is very annoying. The adaptive filter is very suitable to this application since the characteristic of the telephone path that causes the echo changes from connection to connection.

The simulations use two sets of data: white Gaussian noise and colored noise. We also simulate the behavior of the switching structure when there is a

change in the unknown system. Then, we compare the merits of the switching structure with the non-switching structure for all those cases.

The switching structures that are simulated are the *recursive-least-squares-lattice joint process* (RLSL-JP) algorithm together with the LMS algorithm, the *fast-transversal-filter* (FTF) algorithm together with the LMS algorithm, and the *gradient-adaptive-lattice joint process* (GAL-JP) algorithm together with the LMS algorithm. The RLSL-JP, FTF, and GAL-JP algorithms are used as the "fast convergence structure" since they are known to yield fast convergence. On the other hand, the LMS algorithm is chosen as the "low MSE structure", since it is computationally simpler and numerically more robust, and produces low MSE.

When the switching structure switches from the "fast convergence structure" to the "low MSE structure", the parameters of the "fast convergence structure" should also be transferred. This transfer can be done directly if both structures are the same, say both structures are transversal filters. However, when the "fast convergence structure" is a lattice joint-process estimator and the "low MSE structure" is a transversal filter, i.e. a transversal LMS, we can not transfer the parameters directly because they are different. Here we introduce a method to convert the *regression coefficients* of the lattice joint process estimator to the tap-weights of the transversal filter.

After the switching structure reaches its steady state the "low MSE structure" is used. If suddenly the error increases, the switching structure has to figure out whether this is caused by a change in the unknown system or caused by an increase in the observation noise, i.e. the presence of *double talk* in the

echo cancellation application. In this situation, the switching structure enters a transition period.

During the transition period, both structures are working. The "fast convergence structure" tries to figure out what causes this error, while the "low MSE structure" works as a fixed filter, i.e. the adaptation is inhibited, and the output of the switching structure is taken from both structures with some weighting factor. Using this scheme, if it turns out that the change is caused by an increase in the observation noise, the switching structure is still able to model the unknown system using the coefficients which are already in the "low MSE structure". On the other hand, if the change is caused by the unknown system, the switching structure, then, tries to converge again to the new unknown system; the switching structure knows that the change is caused by a change in the unknown system if the "fast convergence structure" is converging to the new unknown system.

The decision whether the switching structure has to choose the "fast convergence structure" or the "low MSE structure" is made by the end of the transition period. The switching structure chooses the "low MSE structure" and operates as a fixed filter if the error increase is caused by an increase of the observation noise, or chooses the "fast convergence structure" if the error increase is caused by a change in the unknown system.

The rest of the thesis is organized as follows: Chapter Two gives the mathematical analysis as well as demonstrations/illustrations of the analytical and experimental result for the adaptive filter algorithms/structures that are used in this thesis. Chapter Three describes the simulation methods for the switching

structure adaptive filter. Chapter Four shows the simulation results and, finally, Chapter Five concludes the thesis and proposes further research.

CHAPTER TWO

THEORETICAL BACKGROUND

This chapter covers the characteristics of adaptive filter structures and algorithms. The discussion emphasizes the *rate of convergence* and *mean square error* (MSE) since these two characteristics are the main reasons for switching from one structure to another. The performance comparison among structures and algorithms is given at the end of this chapter. This performance comparison is limited, in that it is based on the system identification problem as it applies to echo cancellation.

2.1 Adaptive Filters

A **filter** is defined as a system that processes a stochastic input in order to produce a desired output or to extract predefined values, i.e. it passes signals having frequency f_1 only. An **adaptive filter** is a filter that can self-adjust its parameters based on its input signal and the desired signal. In other words, the adaptive filter defines its correct parameters "on-the-fly," starting from any initial value. Because of this self-defining characteristic, the adaptive filter is very suitable for applications where complete knowledge of the statistical characteristics of the environment is not available.

We now review some notions that are used to characterize the merit of adaptive filter algorithms. The *rate of convergence* is the number of iterations required by an algorithm in order for it to reach within some percentage of its final value. In the echo cancellation problem, CCITT Recommendation G.165 [3] defines the rate of convergence as the ability of an echo canceller to reduce the

echo level to a certain value, i.e. -27 dBm0, in a certain time, i.e. 500 ms [3]; the term dBm0 is explained in Appendix D. *Misadjustment* measures in percent the deviation of the *mean square error*, as an ensemble average, from the minimum value as produced by the optimal Wiener filter. *Robustness* describes the performance of an algorithm under ill-conditioned input signal conditions. *Numerical stability* determines the performance of an algorithm under finite word-length conditions.

The adaptive filter used here is a *linear adaptive filter*. Any non-linearity that can not be modeled by the linear system will be considered to produce extraneous noise.

2.1.1 System Identification

One application of adaptive filters lies in the area of system identification. The system identification problem is indicated in Figure 2.1.1. Given the input u(n) and the output d(n) of the unknown system, the task is to develop an adaptive filter in such a way that it generates an output $\hat{d}(n)$, for the same input as the input of the unknown system, that is close to the output of the unknown system in the sense of some predefined criterion, such as minimal mean-square error (MMSE). The output of the unknown system is usually termed the desired signal and the output of the adaptive filter is termed the estimate of the desired signal.

The adaptive filter uses the error between the desired signal and the estimate of the desired signal to change its parameters based on a certain algorithm. The structure of the adaptive filter does not have to be the same as the structure of the unknown system.

One example of the use of system identification is echo cancellation in telephone systems, which will be the main topic of this thesis.

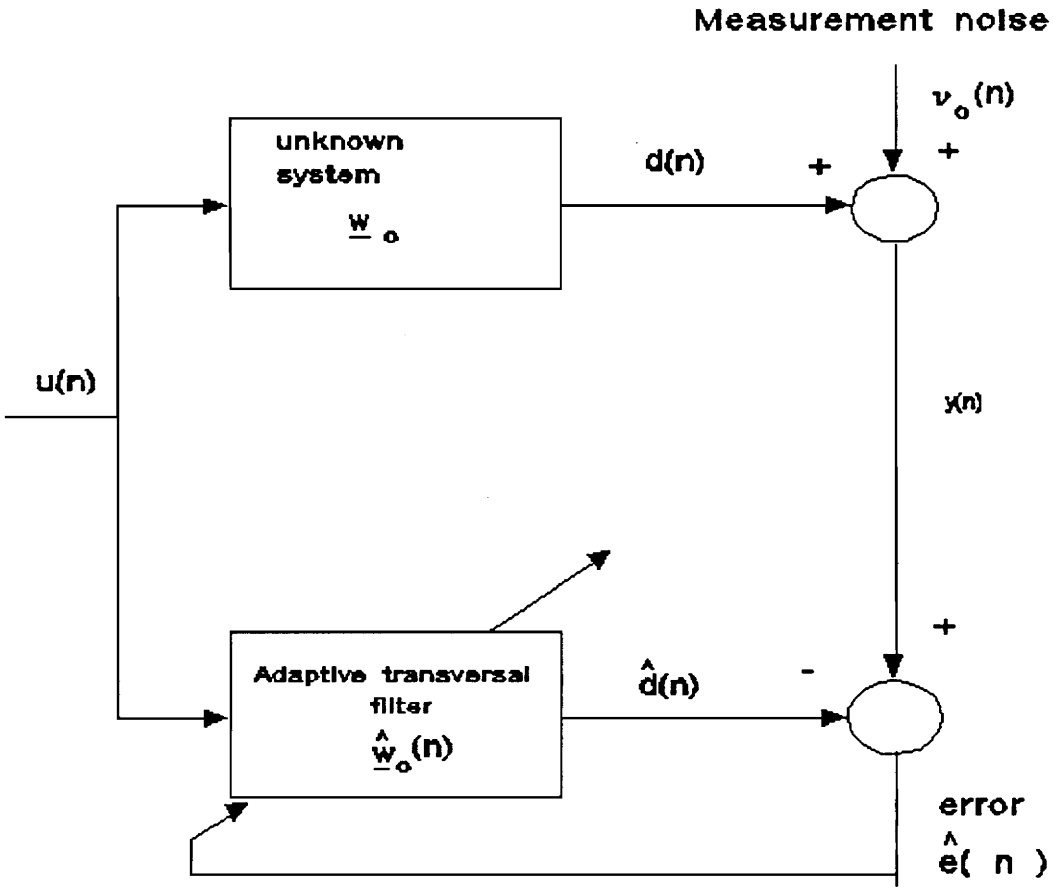

Figure 2.1.1 System Identification

2.1.2 Echo Cancellation

Every telephone connection will experience echo problems; echo is a signal that is transmitted back to its sender. The echo problem is very annoying, especially in long-distance connections such as occur in satellite communication in which one full-hop or *round-trip* delay is about 540 ms [37]. The typical configuration of a telephone connection is shown in Figure 2.1.2.

The best method to date to remove this echo is to use an echo canceller. The echo canceller tries to model the hybrid circuit including the transmission path and then generates a replica of the echo signal using the received signal from the far-end; the echo canceller on the left side in Figure 2.1.2 will cancel the echo signal that comes from the right side. By subtracting the replica of the echo $\hat{d}(n)$ from the echo signal y(n), as shown in Figure 2.1.3, it is expected that the echo can be prevented from going back to its sender. The echo signal is expressed as:

$$y(n) = d(n) + \upsilon(n) + r(n) \tag{2.1.1}$$

$$d(n) = \underline{w}^{H}(n)\underline{u}(n) \tag{2.1.2}$$

where u(n) is the received signal, d(n) is the desired signal, $\upsilon(n)$ is the observation noise, $\underline{w}(n)$ is the system function of the echo-path, and r(n) is the near-end signal.

The replica of the echo signal generated by the echo canceller, $\hat{d}(n)$, is expressed as :

$$\hat{d}(n) = \hat{\underline{w}}^{H}(n)\,\underline{u}(n) \tag{2.1.3}$$

where $\hat{\underline{w}}^{H}(n)$ is the estimate of the system function of the echo-path, and $\hat{d}(n)$ is the estimate of the echo signal d(n) which here is the desired signal (to be cancelled).

A typical impulse response of the echo-path is as shown in Figure 2.1.4. It usually has an all-pass characteristic [10]. The pure delay region of the first few samples represents the distance between the echo canceller and the hybrid. The non-zero impulse response usually lasts about 2 ms (16 samples at 8 kHz sampling frequency) [7].

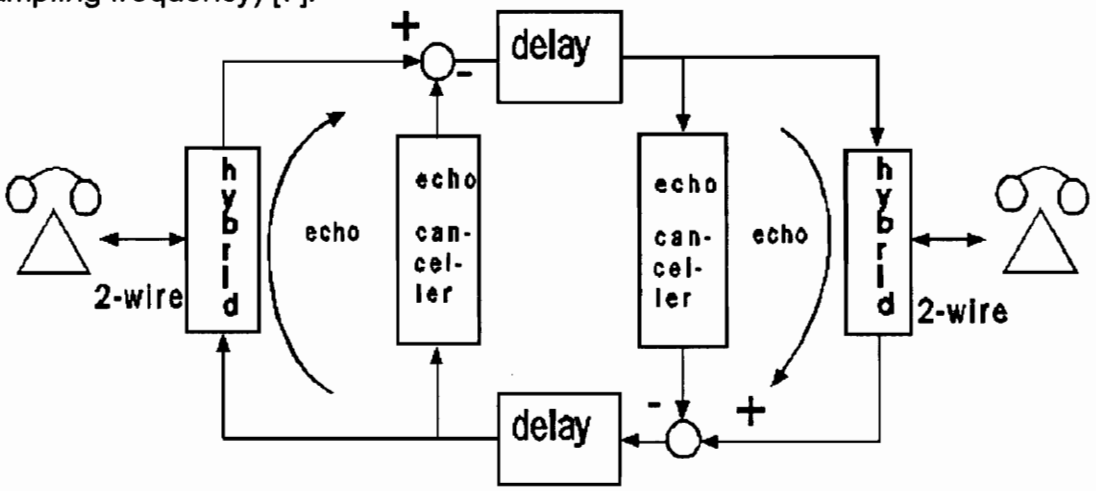

Figure 2.1.2 Typical Telephone Connection
with Echo Cancellers

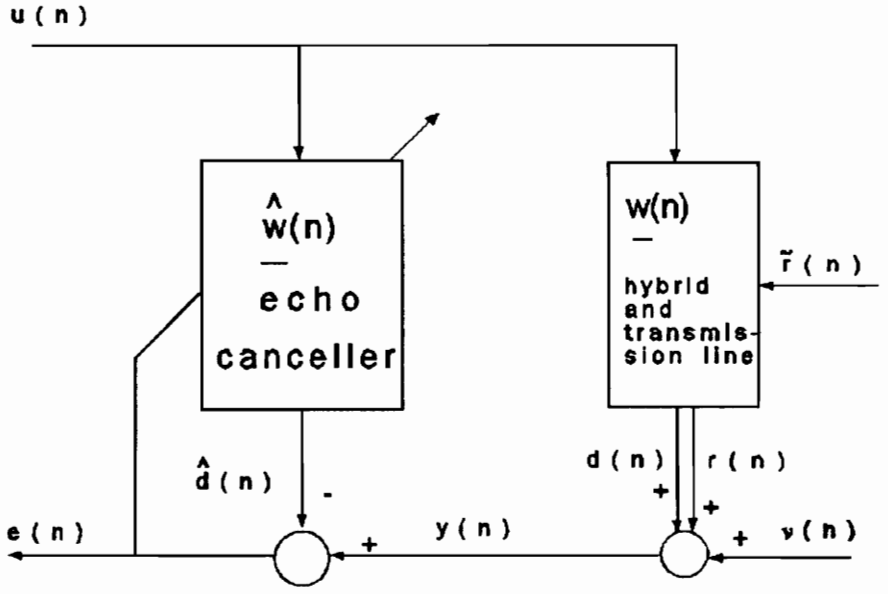

Figure 2.1.3 Relation of Signals

The characteristic of the hybrid, including the transmission line, is different from connection-to-connection, so that the use of an adaptive filter in echo cancellation is well suited to the application.

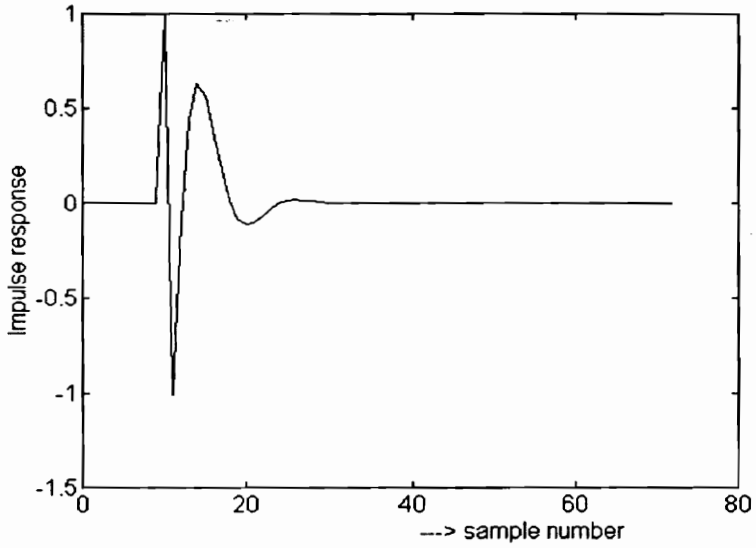

Figure 2.1.4 Typical Impulse Response of the Echo-path

The dynamic range between weak and strong voice signals on the telephone system is about 30 dB [7]; the weak and strong signals mentioned here refer to the signal strength under different connection condition, some connections produce very strong signals, whereas other connections produce weak signals. This could be a problem for some algorithms, such as the least-mean-square (LMS) algorithm, in that its performance is dependent on the eigenvalue or the power of the input signal. As will be mentioned in Section 2.2, the step-size μ of the LMS algorithm should be less than 2/power of the input signal, so that in order to assure convergence for any signal, the step-size of the

LMS algorithm should be set in reference to the weakest signal, otherwise there is the possibility that for some connections the LMS algorithm will diverge.

As soon as the near-end signal r(n) appears, the adaptation should be inhibited, otherwise the echo canceller algorithm will diverge. This occurrence is called the "double talk" condition. To monitor this condition, an echo canceller is usually furnished with a double talk detector.

2.1.3 The Experimental Data Sets

The adaptive filter algorithms, and their features, used in this thesis will be illustrated using several sets of simulated data. For simulations, the adaptive filter algorithms are used for system identification problems as shown in Figure 2.1.1. The plant to be identified is an all-pass type of plant; this is to resemble the echo path. The plant is specified as a pure delay followed by a second-order all-pass filter section with poles at $0.75\ e^{\pm j2\pi/6}$ and zeros at $1.3333e^{\pm j2\pi/6}$. The pure delay models the delay of the echo path and the all-pass filter models the hybrid. The all-pass filter is designed to have a variable gain. The impulse response of the above plant is shown in Figure 2.1.4.

Two kinds of data sets will be used: white Gaussian noise and colored noise with an eigenvalue ratio of 100. The white Gaussian noise is used in the simulations because the standard [3] is based on a white Gaussian noise input instead of speech. On the other hand, the colored noise is used mainly to test the algorithms/structures because the performance of some algorithms/structures is affected by colored noise.

The white Gaussian noise is generated by the MATLAB program routine "randn", with zero mean and unit variance. MATLAB has the capability to

generate white Gaussian noise with a different seed or initial value. This is useful for ensemble averaging of several independent trials. However, we can also define the seed itself, so that we can compare different algorithms based on exactly the same input noise signal. For future use, the white Gaussian noise input data will be referred to as **Data Set 1**.

The colored noise data is generated by passing zero mean white Gaussian noise through a second-order AR process as shown in Figure 2.1.5. The variance of the white Gaussian noise is set such that the output of the AR process will have unit variance theoretically. By using different AR parameters, we can generate colored noise with eigenvalue ratio of 100. For future use, the colored noise data with an eigenvalue ratio of 100 is called **Data Set 2**.

The AR process is expressed as:

$$u(n) = -a_1 u(n-1) - a_2 u(n-2) + v(n) \qquad (2.1.4)$$

where v(n) is a zero mean white Gaussian input with variance σ_v^2, a_1 and a_2 are the parameters of the AR process, and a_0 is set to 1.

The parameters for **Data Set 2** are [16]:

$$\underline{a} = [1 \; -.1960 \; -.8] \; ; \; \sigma_v^2 = 0.014$$

The theoretical output variance of Data Set 2 is .9820 with the eigenvalue ratio $\chi(R)$ of 99; the eigenvalue ratio is the ratio of the maximum and minimum eigenvalues of the auto-correlation matrix of the input data:

$$\chi(R) = \frac{\lambda_{max}}{\lambda_{min}} \tag{2.1.5}$$

$$R = E\left[\underline{u}\underline{u}^H\right] \tag{2.1.6}$$

$$\underline{u} = [u(n),\ u(n-1), \cdots u(n-N+1)]^H \tag{2.1.7}$$

where λ_{max} and λ_{min} are the maximum and minimum eigenvalues of the auto-correlation matrix R respectively. If Data Set 2 is used for an adaptive filter order N, the eigenvalue ratio $\chi(R)$ will be different, depending on N. The following are the theoretical eigenvalue ratios $\chi(R)$ for N equal to 64 and 256:

N	64	256
$\chi(R)$	1.387×10^4	4.872×10^4

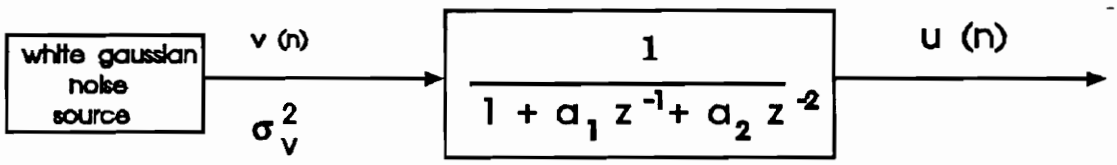

Figure 2.1.5 The AR Model of Order 2

2.2 The Transversal Least Mean Squares Algorithm

The least mean squares (LMS) algorithm is based on the use of the instantaneous value of the gradient, as an estimate of the gradient, to recursively compute the solution of the Wiener Filter. The LMS algorithm is depicted in Figure 2.2.1 and is expressed as follows [16]:

$$y(n) = \hat{\underline{w}}^H(n)\,\underline{u}(n) \tag{2.5.a}$$

$$e(n) = d(n) - y(n) \tag{2.5.b}$$

$$\hat{\underline{w}}(n+1) = \hat{\underline{w}}(n) + \mu\,\underline{u}(n)\,e^*(n) \tag{2.5.c}$$

where d(n) is the desired signal, y(n) is the estimate of the desired signal, e(n) is the difference between the desired signal and its estimate, $\hat{\underline{w}}(n)$ is the estimate of the tap-weight of the unknown system, $\underline{u}(n)$ is the tap-input vector, and μ is the step-size parameter. The LMS algorithm in (2.5) can be used for *complex* as well as *real* systems. A *complex* system means that the data and the tap-weight vector consist of complex numbers, and a *real* system means that the data and the tap-weight vector consist of real numbers.

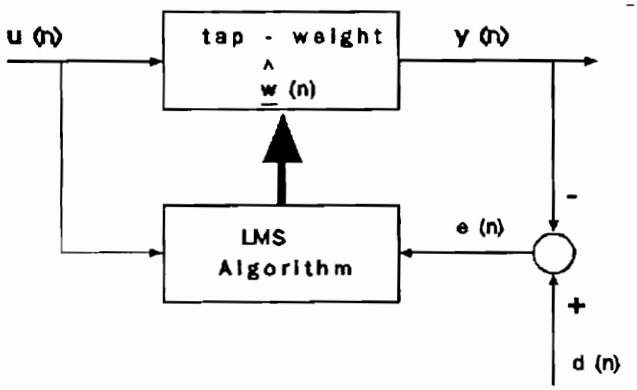

Figure 2.2.1 The LMS Algorithm

In order for the tap-weight to converge in the mean, μ should satisfy:

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{2.6}$$

where λ_{max} is the maximum eigenvalue of the input auto-correlation matrix R which is defined as:

$$R = E\left[\underline{u}\,\underline{u}^{H}\right] \tag{2.7}$$

Note that convergence in the mean implies that the mean of the tap-weight estimate $\hat{\underline{w}}(n)$ converges to the optimum Wiener solution $\underline{w}_{opt}$ [16].

We know that λ_{max} is less than $\sum_{i=1}^{M}\lambda_i$, which is equivalent to M times the auto-correlation of the tap-input vector at lag zero, σ_n^2, or

$$\sum_{i=1}^{M}\lambda_i = M\sigma_n^2 \tag{2.8}$$

Considering that M times the auto-correlation at lag zero, $M\sigma_n^2$, is equivalent to the power of the input signal [16], so that by replacing λ_{max} in (2.6) with the input power, we get

$$0 < \mu < \frac{2}{\text{total input power}} \tag{2.9}$$

Furthermore, the step-size μ defined in (2.9) is always less than μ defined in (2.6), meaning that (2.9) always gives μ that makes the LMS algorithm converge.

The maximum time constant τ_{max} for convergence in the mean [16], is

$$\tau_{max} = \frac{-1}{\ln(1-\mu\lambda_{min})} \tag{2.10.a}$$

For μ small we can approximate (2.10.a) with:

$$\tau_{max} \approx \frac{1}{\mu \lambda_{min}} \tag{2.10.b}$$

By replacing μ in (2.10.b) with the upper limit in (2.6) we get:

$$\tau_{max} > \frac{\lambda_{max}}{2\lambda_{min}} = \frac{\chi(R)}{2} \tag{2.11}$$

where $\chi(R)$ is the ratio (condition number) of the maximum and minimum eigenvalues of the auto-correlation matrix R.

The MSE of the LMS algorithm [16] is:

$$J(n) = \sum_{i=1}^{M} \gamma_i c_i^n + \frac{J_{min}}{1 - \sum_{i=1}^{M} \frac{\mu \lambda_i}{(2 - k\mu\lambda_i)}} \tag{2.12}$$

where k is equal to 1 for the complex LMS algorithm and k is equal to 2 for the real LMS algorithm, and J_{min} is the minimum mean square error produced by the optimal Wiener Filter.

The constants c_i are the eigenvalues of matrix C, which is a diagonal matrix derived from the *eigendecomposition* of matrix B:

$$B = GC\,G^H$$

G is an orthonormal matrix consisting of the orthonormal *eigenvectors* $\underline{g}$, and G^H denotes the Hermitian transpose of matrix G. B is an M-by-M matrix with elements:

$$b_{ij} = \begin{cases} (1 - \mu\lambda_i)^2 + k\mu^2\lambda_i^2 & i = j \\ \mu^2\lambda_i\lambda_j & i \neq j \end{cases} \qquad (2.14)$$

where k is equal to 1 for the complex LMS algorithm, and k is equal to 2 for the real LMS algorithm.

We now derive γ_i. Thereto define the weight error vector $\underline{\varepsilon}(n)$ as:

$$\underline{\varepsilon}(n) = \underline{\hat{w}}(n) - \underline{w}_{opt} \qquad (2.15)$$

and define K(n) as the variance of $\underline{\varepsilon}(n)$:

$$K(n) = E\left[\underline{\varepsilon}(n)\underline{\varepsilon}^H(n)\right] \qquad (2.16)$$

Now multiply K(n) with Q^H on the left side and Q on the right side to yield:

$$X(n) = Q^H K(n)\, Q \tag{2.17}$$

where Q is obtained from an eigendecomposition of auto correlation matrix R such that

$$\Lambda = Q^H R Q$$

in which Λ is an diagonal matrix. Usually X(n) in (2.17) is not a diagonal matrix, consisting of column vectors $x_i(n)$. It has been shown [16] that

$$\gamma_i = \underline{\lambda}^T g_i^T g_i \left[x(0) - x(\infty)\right] \tag{2.18}$$

$$\underline{x}(\infty) = \mu^2 J_{min} (I - B)^{-1} \underline{\lambda} \tag{2.19}$$

where x(0) is the initial value of x(n), x(∞) is the final value of x(n), and I is the M-by-M identity matrix.

The range of μ that guarantees that the algorithm converges in the mean-square is

$$\sum_{i=1}^{M} \frac{\mu \lambda_i}{(2 - k\mu \lambda_i)} < 1 \tag{2.22}$$

where k is equal to 1 for the complex LMS algorithm, and k is equal to 2 for the real LMS algorithm. In order for the LMS algorithm to converge, μ should fulfill

(2.6) as well as (2.22). Note that convergence in the mean square implies that the final value of the mean square error, $J(\infty)$, is finite [16].

The LMS algorithm is very simple and has good *numerical properties*, i.e. the LMS algorithm has not been reported as having divergence problems caused by quantization errors. However, the LMS algorithm convergence rate is very dependent on the eigenvalue ratio of the auto-correlation matrix R. In other words, for ill-conditioned input signals, i.e. $\chi(R)$ large, the LMS algorithm will take a long time to converge as indicated by (2.11).

When using a large μ, the algorithm will converge fast as indicated by (2.10), but the MSE will be large as indicated by (2.12). In order to get fast convergence and small MSE, we can use a time-varying μ. At the beginning of the iterations, a large μ is used, so that the LMS algorithm will converge fast. After the algorithm has converged substantially, a small μ is used, and this will give a small MSE. However the choice of a large μ can cause divergence of the algorithm for some input signals, if in this condition μ violates (2.6) or (2.22). This can happen in an application where the input signal strength varies greatly from one condition to another, such as in the echo cancellation application.

2.2.1 Convergence Indicator for the LMS Algorithm

The estimation error of the LMS algorithm is very noisy, so that it is difficult to monitor whether the algorithm has converged or not. One way to check for convergence of the algorithm is by computing an average squared estimation error. This is done by summing the current squared estimation error and M-1 previous squared errors and then dividing this sum by M.

In order to reduce the computation, the M-1 previous values of the squared estimation error are stored, and added to the square of the current estimation error (the oldest value of the squared estimation errors is automatically removed as depicted in Figure 2.2.2), and the sum of these errors is divided by M. By using this scheme, we only add one multiplication to the total computational load at every iteration. These procedures can be expressed as follows:

$$e_T^2(n) = \sum_{i=0}^{M-1} e^2(n-i) \tag{2.24}$$

$$e_{ave}^2(n) = \frac{e_T^2(n)}{M} \tag{2.25}$$

where M is the length of the averaging interval.

After the algorithm converges, the difference of the average squared estimation error between two successive iterations is very small; in other words, the average squared error will be almost constant. We therefore define the convergence indicator $\gamma(n)$ as:

$$\gamma(n) = 1 - e_{ave}^2(n) \tag{2.26}$$

2.2.2 Effect of the Initial Value

Recall the *weight error vector* $\underline{\varepsilon}(n+1)$ defined in (2.15) as

$$\underline{\varepsilon}(n+1) = \underline{\hat{w}}(n+1) - \underline{w}_{opt}$$

By using (2.5.c) in (2.15), where e(n) is replaced by (2.5.b), we can rewrite (2.15) as

$$\underline{\varepsilon}(n+1) = \left(I - \mu \underline{u}(n)\underline{u}^{H}(n)\right) \underline{\varepsilon}(n) + \mu \underline{u}(n)\, e^{*}(n) \tag{2.26}$$

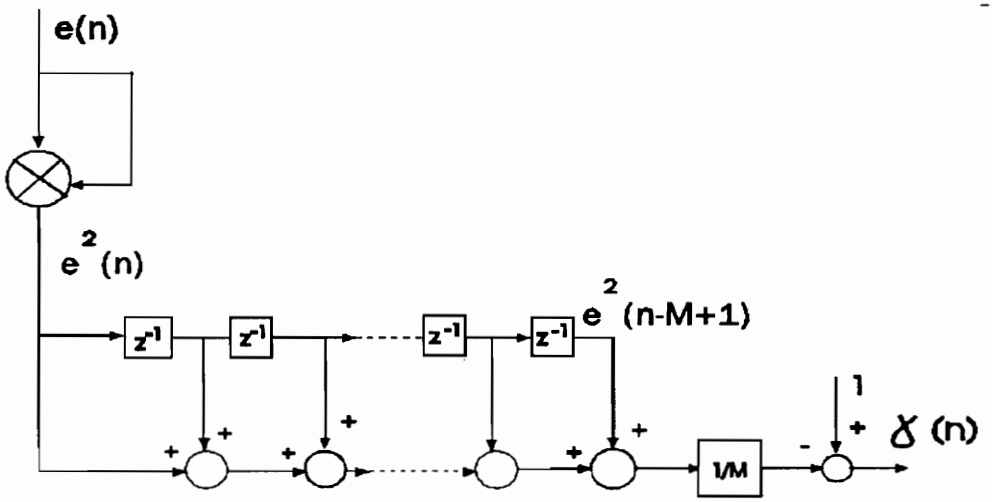

Figure 2.2.2 Squared Error Averaging Circuit

The expected value of the weight error vector, by assuming that $\underline{u}(n)$ and $e^{*}(n)$ are orthogonal, is:

$$E\left[\underline{\varepsilon}(n+1)\right] = (I - \mu R)\underline{\varepsilon}(n) \tag{2.27}$$
$$= (I-\mu R)^{n}\, \underline{\varepsilon}(0)$$

where

$$\underline{\varepsilon}(0) = \hat{\underline{w}}(0) - \underline{w}_{opt}$$

and $\hat{\underline{w}}(0)$ is the initial value of the tap-weight.

The choice of $\underline{\varepsilon}(0)$ will determine the rate of convergence. If $\underline{\hat{w}}(0) = \underline{w}_{opt}$, then $\underline{\varepsilon}(0) = 0$, and the algorithm has converged immediately. However, if the initial tap-weight is far from the optimum value, then convergence will usually take quite a long time. Experiments with different initial values confirm this. For initial conditions zero, one, and optimum, the results are given in Figures 2.2.3, 2.2.4, and 2.2.5 respectively. These experiments use Data Set 1 with a filter order of 64 and μ of 0.005. The MSE is taken as the ensemble average of 10 independent trials.

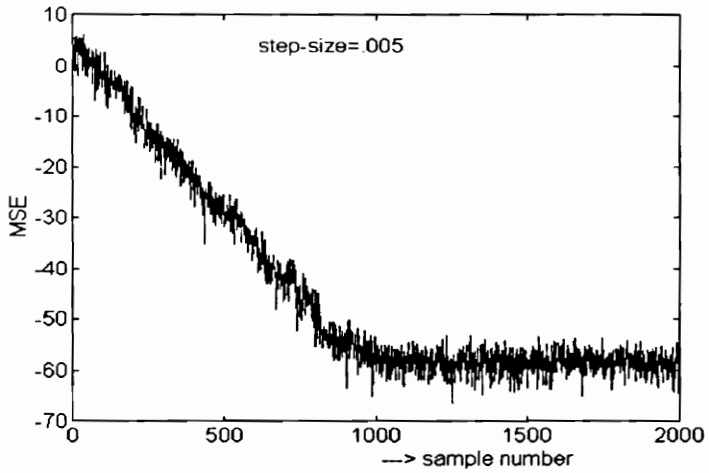

Figure 2.2.3 The MSE for the LMS Algorithm with Zeros as the Initial Condition

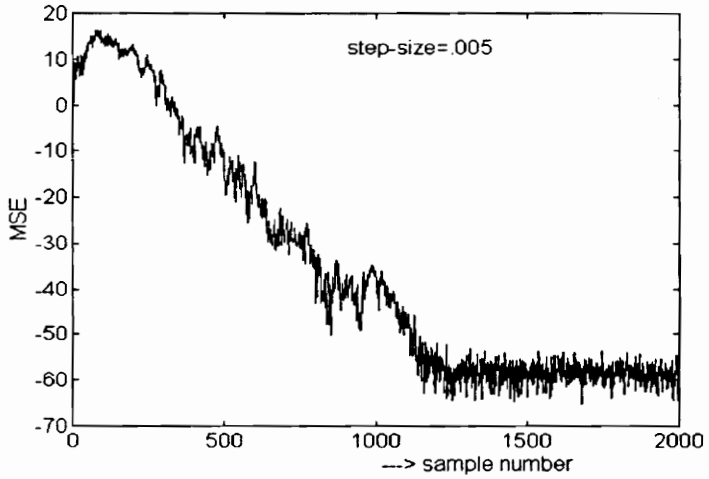

Figure 2.2.4 The MSE for the LMS Algorithm with One as the Initial Condition

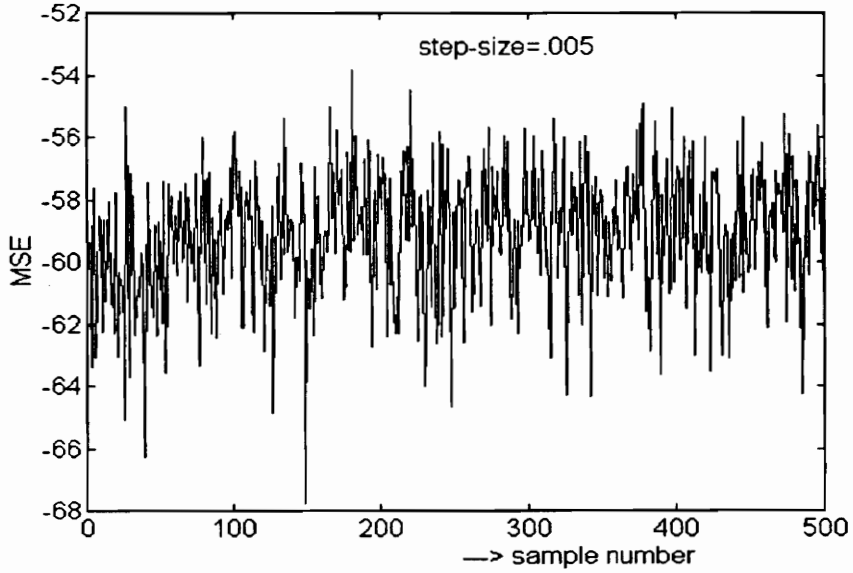

Figure 2.2.5 The MSE for the LMS Algorithm with the Optimum as the Initial Condition

Since an initial condition of zero does not show any peculiar convergence behavior such as when the initial condition is one, for practical purposes it will be convenient to start the algorithm from initial condition zero. However, if the algorithm is given an initial condition that is close to the optimum value, as happens when we switch from another algorithm, the convergence rate will be faster than if the algorithms were started at an initial tap-weight of zero. This is exactly the behavior we want to exploit when switching from the "fast convergence structure" to the "low MSE structure", thereby averting the slow convergence of the latter.

Furthermore, if the initial tap-weight is extracted from another algorithm, which is done during operation, the M tap-input of the latter algorithm should be the same as the M tap-input of the previous algorithm; otherwise, the latter algorithm will experience a transient condition, and if this happens, the error will

go up because the estimate of the desired response will be different from the desired response.

2.3 The Method of Least Squares

The *method of least squares* uses a time average of the available data, instead of an ensemble averaged statistical characterization of the data, to solve the linear filtering problem. As a result, the least squares approach gives a better solution than that resulting from approximation approaches such as in the LMS or gradient algorithm [16]. The *cost function* $\xi(n)$of the least squares method is:

$$\xi(n) = \sum_{i=1}^{n} |e(i)|^2 \tag{2.3.1}$$

where n is the time index, e(i) is the difference between the desired signal d(i) and the estimate of the desired signal $\hat{y}(i)$ at time i. As shown in Figure 2.3.1, the error e(i) is defined as:

$$\begin{aligned} e(i) &= d(i) - \hat{y}(i) \\ &= d(i) - \underline{w}^H(n)\, \underline{u}(i) \end{aligned} \tag{2.3.2}$$

where $\underline{w}(n)$ is the *tap-weight* vector at time n, defined as:

$$\underline{w}(n) = \left[w_0(n), w_1(n), \ldots, w_{M-1}(n)\right]^T \tag{2.3.3}$$

$\underline{w}(n)$ is fixed during the observation interval $1 \le i \le n$, and $\underline{u}(i)$ is the *tap-input vector at time i*, defined as:

$$\underline{u}(i) = [u(i), u(i-1), \ldots, u(i-M+1)]^T \qquad 1 \le i \le n \qquad (2.3.4)$$

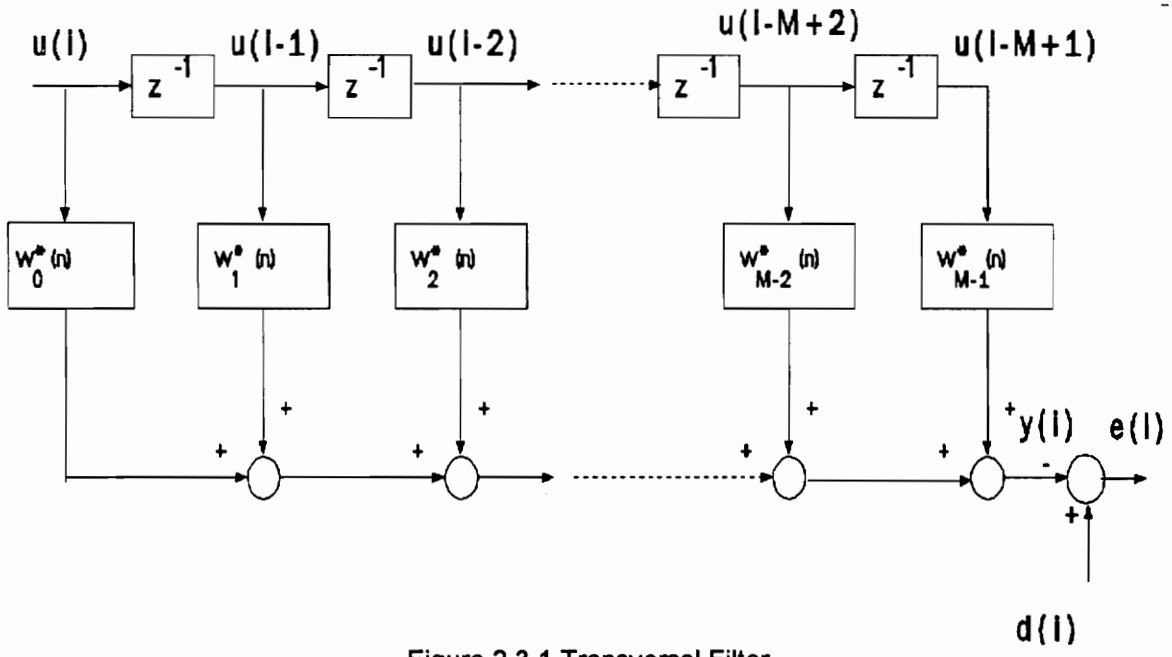

Figure 2.3.1 Transversal Filter

2.3.1 TheTransversal Recursive Least Squares

The *recursive least squares* (RLS) algorithm solves the least squares approach recursively by way of minimizing a *weighted* version of the *cost function* $\xi(n)$. We follow a fairly standard treatment [16] of the RLS algorithm to set the stage for further work. The commonly used weight or forgetting factor μ is usually in the form of *exponential weighting*, so that the cost function $\xi(n)$ is written as [16]:

$$\xi(n) = \sum_{i=1}^{n} \mu^{n-i} |e(i)|^2; \qquad 0 < \mu \le 1 \qquad (2.3.5)$$

For μ equal to 1, the algorithm uses all the past data equally and the criterion corresponds to that in (2.3.1). In other words, the case of μ equal to 1 corresponds to *linearly increasing memory*. For μ <1, the algorithm forgets the past data more and more.

The optimum values of the tap-weight vector $\underline{w}(n)$ are obtained from the *normal equation:*

$$\Phi(n)\ \hat{\underline{w}}(n) = \underline{\theta}(n) \tag{2.3.6}$$

where $\Phi(n)$ is the M-by-M correlation matrix of the input signal, defined as:

$$\Phi(n) = \sum_{i=1}^{n} \mu^{n-i}\ \underline{u}(n)\underline{u}^{H}(n) \tag{2.3.7}$$

and $\underline{\theta}(n)$ is an M-by-1 cross-correlation vector between the input signal and the desired signal, defined as :

$$\underline{\theta}(n) = \sum_{i=1}^{n} \mu^{n-i}\underline{u}(i)\ d^{*}(n) \tag{2.3.8}$$

In recursive form, $\Phi(n)$ and $\underline{\theta}(n)$can be written as :

$$\Phi(n) = \mu\,\Phi(n-1) + \underline{u}(n)\ \underline{u}^{H}(n) \tag{2.3.9}$$

$$\underline{\theta}(n) = \mu\,\underline{\theta}(n-1) + \underline{u}(n)\ d^{*}(n) \tag{2.3.10}$$

Using the *matrix inversion lemma* [16], the inverse of $\Phi(n)$ can be written in a recursive form as:

$$\Phi^{-1}(n) = \mu^{-1}\Phi^{-1}(n-1) - \frac{\mu^{-2}\Phi^{-1}(n-1)\,\underline{u}(n)\underline{u}^{H}(n)\Phi^{-1}(n-1)}{1+\mu^{-1}\underline{u}^{H}(n)\Phi^{-1}(n-1)\underline{u}(n)} \tag{2.3.11}$$

Letting $P(n) \triangleq \Phi^{-1}(n)$, we can rewrite (2.3.11) as:

$$P(n) = \mu^{-1}P(n-1) - \frac{\mu^{-2}P(n-1)\,\underline{u}(n)\underline{u}^{H}(n)P(n-1)}{1+\mu^{-1}\underline{u}^{H}(n)P(n-1)\underline{u}(n)} \tag{2.3.12}$$

$$= \mu^{-1}P(n-1) - \mu^{-1}\underline{k}(n)\underline{u}^{H}(n)P(n-1) \tag{2.3.13}$$

where $\underline{k}(n)$ is an M-by-1 vector defined as

$$\underline{k}(n) = \frac{\mu^{-1}P(n-1)\,\underline{u}(n)}{1+\mu^{-1}\,\underline{u}^{H}(n)P(n-1)\underline{u}(n)} \tag{2.3.14}$$

The *time update* of the tap-weight vector can now be derived. Multiplying (2.3.6) with $\Phi^{-1}(n)$ from the left, we have

$$\hat{\underline{w}}(n) = \Phi^{-1}(n)\underline{\theta}(n) \tag{2.3.15}$$

and then by applying (2.3.10), (2.3.12), (2.3.14), and (2.3.15), we have the time update of the tap-weight vector $\hat{\underline{w}}(n)$ as:

$$\hat{\underline{w}}(n) = \hat{\underline{w}}(n-1) + \underline{k}(n)\alpha^{*}(n) \qquad (2.3.16)$$

where $\alpha(n)$ is the *a priori error*

$$\alpha(n) = d(n) - \hat{\underline{w}}^{H}(n-1)\,\underline{u}(n) \qquad (2.3.17)$$

The RLS algorithm is initialized by defining :

$$\Phi(0) = \delta I \qquad (2.3.18)$$

where I is an M-by-M identity matrix, and $0 < \delta \leq 1$, δ is a small positive number, something like 0.1. Consequently,

$$P(0) = \delta^{-1} I \qquad (2.3.19)$$

For $\mu<1$, the initial value of $\Phi(0)$ will not affect the steady state value of the algorithm because $\Phi(0)$ will be zero, as proven below:

$$\Phi(n) = \mu\,\Phi(n-1) + \underline{u}(n)\,\underline{u}^{H}(n) \qquad (2.3.20.a)$$

$$= \mu^{n-1}\,\Phi(0) + \sum_{i=0}^{n} \mu^{i}\,\underline{u}(n-i)\underline{u}^{H}(n-i) \qquad (2.3.20.b)$$

where for $n \rightarrow \infty$, the first term of (2.3.20.b), which corresponds to the initial value, will vanish.

2.3.1.1 Convergence of the Tap-weight Vector in the Mean

For the RLS algorithm with $\mu < 1$, the expected value of the estimate of the tap-weight vector can be represented by [16]:

$$E\left[\hat{\underline{w}}(n)\right] = \underline{w}_{opt} - \mu^n \delta \Phi^{-1}(n)\underline{w}_{opt} \qquad (2.3.21)$$

As $n \rightarrow \infty$, the second term of (2.3.21) goes to zero. In other words, the expected value of the tap-weight exponentially approaches the optimum tap-weight.

The independence of the *weighted* RLS algorithm from the *eigenvalue ratio* can be shown as follows. The correlation matrix $\Phi(n)$ defined in (2.3.7) is bounded if the outer product of the input vector u(i) satisfies [20]

$$\upsilon I \leq \frac{1}{N-M}\sum_{i=M}^{N}\underline{u}(i)\underline{u}^H(i) \leq \rho I$$

for all i, N-M is fixed, I is an identity matrix , $0 < \rho < \infty$, and $\upsilon > 0$. Define the square of the tap-weight vector error as $\sigma_\varepsilon(n)$

$$\sigma_\varepsilon(n) = \underline{\varepsilon}^H(n)\Phi(n)\underline{\varepsilon}(n) \qquad (2.3.22)$$

where the tap-weight error vector $\underline{\varepsilon}(n)$ is defined as

$$\underline{\varepsilon}(n) = \underline{w}(n) - \underline{w}_{opt} \qquad (2.3.23)$$

and $\Phi(n)$ is defined in (2.3.20.a). Then we can define the difference of $\sigma_\varepsilon(n-1)$ with $\sigma_\varepsilon(n)$ as

$$\Delta(n) = \sigma_\varepsilon(n) - \sigma_\varepsilon(n-1) \tag{2.3.24}$$

By substituting (2.3.20.a) and (2.3.22) into (2.3.24), we have [20]

$$\Delta(n) \leq (\mu-1)\underline{\varepsilon}^H(n-1)\Phi(n-1)\underline{\varepsilon}(n-1) \tag{2.3.25.a}$$

$$\leq (\mu-1)\sigma_\varepsilon(n-1) \tag{2.3.25.b}$$

and by replacing $\Delta(n)$ with (2.3.24), we get

$$\sigma_\varepsilon(n) \leq \mu\sigma_\varepsilon(n-1) \tag{2.3.26.a}$$

$$\leq \mu^n\sigma_\varepsilon(0) \tag{2.3.26.b}$$

$$\leq \mu^n \underline{\varepsilon}^H(0)\Phi(0)\underline{\varepsilon}(0) \tag{2.3.26.c}$$

Eq.(2.3.26.c) shows that $\sigma_\varepsilon(n)$ decays exponentially to zero. Since $\Phi(n)$ is a positive definite matrix, when $\sigma_\varepsilon(n)$ decays to zero the only component of $\sigma_\varepsilon(n)$ that decays to zero is the tap-weight error vector $\underline{\varepsilon}(n)$ as shown in (2.3.22). This means also that the tap-weight vector $\underline{w}(n)$ converges exponentially to $\underline{w}_{opt}$. Eq.(2.3.26.c) also indicates that the convergence of the weighted RLS algorithm is not dependent on the eigenvalue ratio.

2.3.1.2 The RLS Mean Square Error

The MSE of the RLS algorithm is formulated as [16]:

$$J(n) = \sigma_n^2 + \mathrm{tr}\left[RK(n-1)\right] \tag{2.3.27}$$

where σ_n^2 is the variance of the observation noise, K(n-1) is the weight-error correlation matrix, i.e. $E[\underline{\varepsilon}(n)\underline{\varepsilon}^H(n)]$, for $\underline{\varepsilon}(n)$ defined in (2.3.23), and R is the M-by-M ensemble-averaged correlation matrix of the input signal.

For large n, by using the *law of large numbers* we get [16]:

$$J(n) = \sigma_n^2 + J_{ex}(n) \tag{2.3.28}$$

where the *excess mean-squared error* $J_{ex}(n)$ satisfies

$$J_{ex}(n) = \mathrm{tr}\{E[RK(n)]\} \tag{2.3.29}$$

The final value of $J_{ex}(n)$ can be written as

$$J_{ex}(\infty) = \sigma_n^2 \frac{1-\mu}{1+\mu} M \tag{2.3.30}$$

and then we can define the *Misadjustment* $\mathcal{M}$:

$$\mathcal{M} = \frac{J_{ex}(\infty)}{\sigma_n^2} = \frac{1-\mu}{1+\mu} M \tag{2.3.31}$$

where M is the order of the filter.

Unlike the LMS algorithm, the RLS algorithm is less affected by an ill-conditioned input signal as indicated in (2.2.26). Furthermore, the RLS algorithm converges in the mean square in about 2M iterations [16], [30].

2.3.2 The Fast Transversal Filter (FTF) Algorithm

Even though the RLS algorithm avoids computing the inverse of a matrix, its computational load is still relatively heavy; on the order of M^2, where M is the order of the filter. Reducing the computational load to on the order of M leads to so-called fast algorithms. Such a fast algorithm is the *fast transversal filter* (FTF) algorithm.

The FTF algorithm consists of four different transversal filters driven by the same input data, as shown in Figure 2.3.2. Filter 1 and filter 2 solve the forward and backward prediction error problems respectively. Filter 3 computes the gain vector $\underline{k}(n)$ of the RLS algorithm. Filter 4 computes the *a priori* and *a posteriori* errors and also adapts the tap-weight vector. The combination of filters 1 to 4 produces the exact solution of the RLS problem at all times [16]. The complete FTF algorithm is given in Appendix A.

The FTF algorithm introduces the conversion factor γ (n) that relates the a priori error $\alpha(n)$ and the a posteriori error e(n). This scheme reduces the computation of the RLS solution, and also the $\gamma(n)$ can be used as a convergence indicator.

The FTF algorithm solves the RLS problem, and fundamentally therefore exhibits the same performance as the standard RLS algorithm such as rate of convergence and misadjustment. However, finite word-length experiments show that the FTF algorithm suffers from numerical instability [16], [36].

2.3.2.1 Initialization of the FTF Algorithm

The FTF algorithm can be started by initializing some of the parameters as follows [1]:

$a_M(0) = [1\ 0 \cdots 0]$

$c_M(0) = [0\ 0 \cdots 1]$

$\gamma_M(0) = 1$

$F_M(0) = \mathbf{B}_M(0) = \omega$, ω is a small positive number (I used 0.09)

$\underline{k}_M(0) = [0\ 0 \cdots 0]$

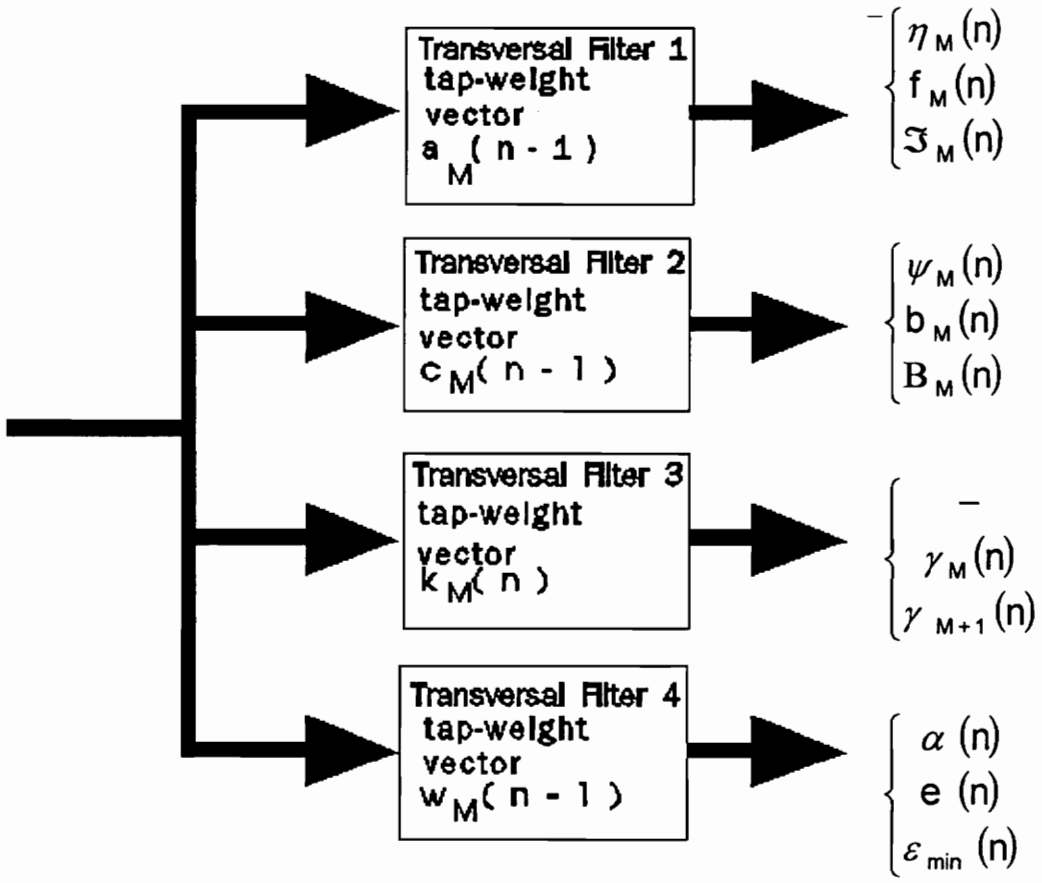

Figure 2.3.2 Fast Transversal Filter

Actually, there is no exact guidance on how to pick ω. However, if we use the soft constraint (explained below) to initialize the FTF algorithm with the tap-weight vector of zeros, this initialization is equal to the soft constraint initialization of the RLS algorithm and the recommended value of ω is any positive number as long as it is small compared to $.001\sigma_u^2$, where σ_u^2 is the variance of a data sample u(n) [16].

If we start the FTF algorithm using some arbitrary initial condition, such as when we switch from another algorithm to the FTF algorithm, we should use the *soft constraint.* This is done by appending (M+1) terms to the tap-input data and the desired signal as follows. Suppose at time n we switch to the FTF algorithm and at that time we already have the tap-input data

$$\underline{u}(n) = [u(n-M), u(n-M+1),\cdots, u(n)]$$

and the desired response

$$\underline{d}(n) = [d(n-M), d(n-M+1),\cdots, d(n)]$$

Assume that we have stored the last M values of the desired response, so that when we switch to the FTF algorithm we append the tap-input data $\underline{u}(n)$ with M+1 terms of data as follows

$$\text{tap-input } \underline{u}(n) \rightarrow \left\{ \overbrace{\omega^{-\frac{1}{2}},\ 0 \cdots,\ 0,\ 0}^{\text{M+1 terms}},\ u(n-M),\ u(n-M+1),\ \cdots,\ u(n) \right\} \qquad (2.3.32)$$

↑

Time n

and we also append the desired response $\underline{d}(n)$ with M+1 terms as follows:

$$d(n) \rightarrow \left\{ \overbrace{\omega^{-\frac{1}{2}} w_{o0},\ \omega^{-\frac{1}{2}} w_{o1},\ \cdots,\ \omega^{-\frac{1}{2}} w_{oM-1}, 0}^{\text{M+1 terms}},\ d(n-M),\ d(n-M+1),\ \cdots,\ d(n) \right\}$$

↑ (2.3.33)

Time n

where $\underline{w}_o$ is the tap-weight vector used as the initial value of the FTF algorithm, and ω is a positive number to ensure a *quadratic cost function*. So, during the initialization from time interval n up to n+M, the FTF algorithm does not use the tap-input $\underline{u}(n)$ nor the desired signal $\underline{d}(n)$ yet, it uses the appended terms of $\underline{u}(n)$ and $\underline{d}(n)$ instead.

The soft constraint can also be used when we want to change the weighting factor μ during operation of the FTF.

By using the soft constraint, the *cost function* is modified to :

$$\xi(n) = \mu^n \omega \left\| \underline{w}(n) - \underline{w}_{opt} \right\|^2 + \sum_{i=1}^{n} \mu^{n-i} \left| e(i) \right|^2 \qquad (2.3.34)$$

For n large and $\mu < 1$, the first term of (2.3.34) will vanish so that the cost function eventually gives the RLS solution.

2.4 Lattice Structure

Unlike the transversal or tapped-delay line filter structure, the lattice structure provides the potential to orthogonalize the input signal. This makes the lattice structure based algorithm less dependent on the eigenvalue ratio $\chi(R)$ of the input signal. This also implies that it may converge faster than transversal filter based algorithms [14], [15], [28]. Furthermore, lattice structure algorithms show better numerical properties in terms of both numerical stability and numerical accuracy. On the other hand, lattice structure based algorithms need more computations per iteration than transversal filter based algorithms [33].

The structure of the Lattice Joint Process Estimator is shown in Figure 2.4.1.

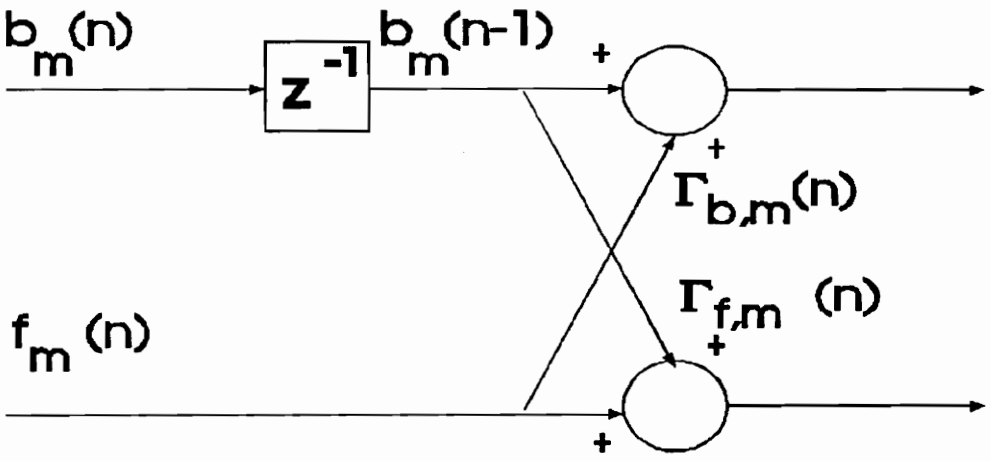

Figure 2.4.1 Single Stage of Lattice Structure

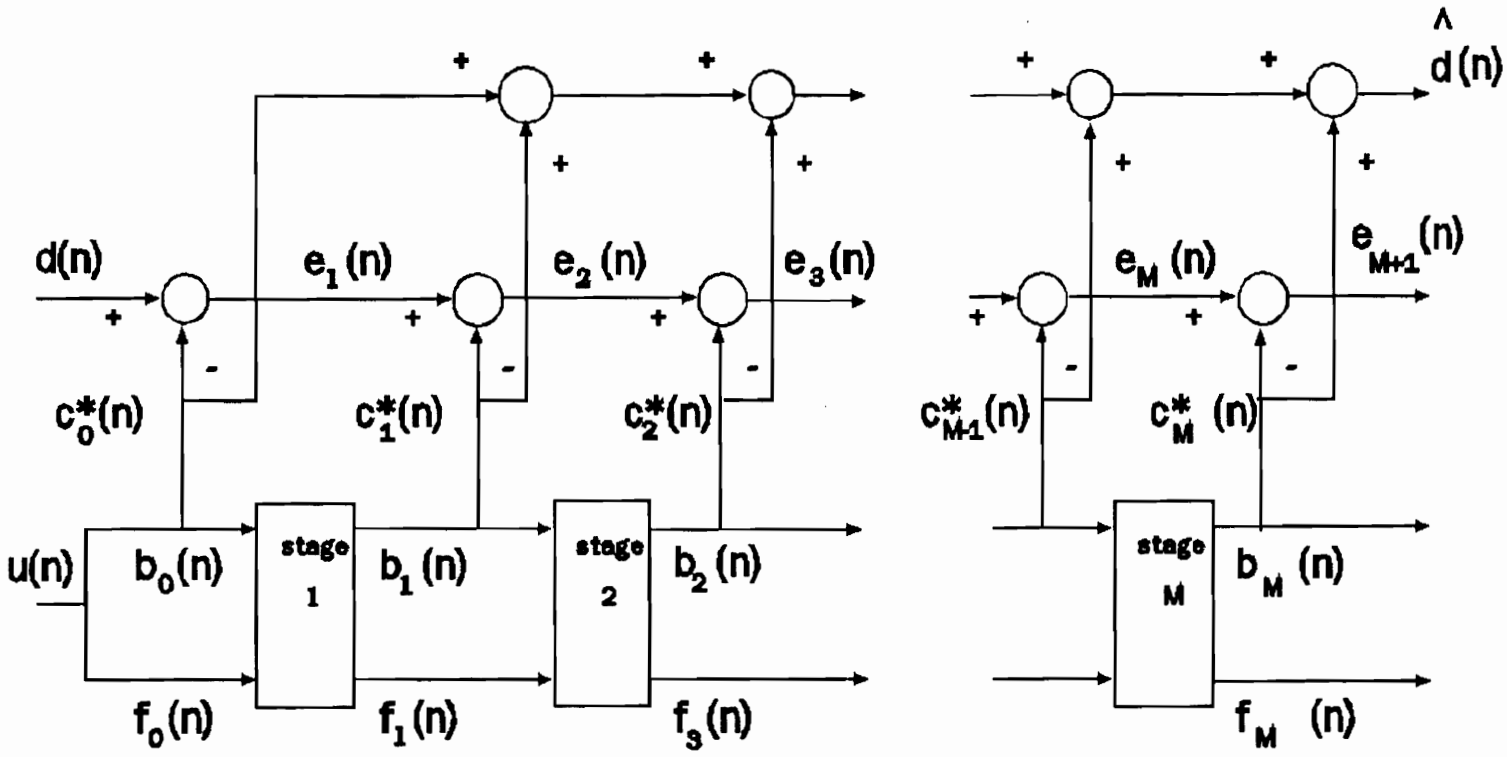

Figure 2.4.1.b Lattice Joint Process Estimator

2.4.1 The Gradient Adaptive Lattice Joint Process Algorithm

The Gradient Adaptive Lattice Joint Process (GAL-JP) algorithm uses the gradient algorithm to adapt its *regression coefficients* or the *ladder weights*. The GAL-JP adaptation algorithm is summarized in (B.1.a) through (B.1.g) in Appendix B.

The MSE at *stage* m, $J_{m+1}(n)$, is computed as the variance of the estimation error at the output of stage m:

$$J_{m+1}(n) = E\left[e_{m+1}(n)e^{*}_{m+1}(n)\right] \tag{2.4.1}$$

where $e_{m+1}(n)$ is the estimation error at the output of stage m, defined as:

$$e_{m+1}(n) = e_m(n) - c^{*}_m(n)b_m(n) \tag{2.4.2}$$

Defining the regression coefficient error at stage m as:

$$\varepsilon_m(n+1) = c_m(n+1) - c_{m\,opt} \tag{2.4.3}$$

and the variance of the regression-coefficient error at stage m, $K_m(n+1)$, as:

$$K_m(n+1) = E\left[\varepsilon_m(n+1)\varepsilon^{*}_m(n+1)\right]$$

$$= K_m(n)B_{m+1}(n+1) + P_{m+1}(n+1) \tag{2.4.4}$$

where, from (B.13),

$$B_{m+1}(n+1) = 1 - \frac{4E\left[b_m^2(n)\right]}{E\left[\mu_{m+1}(n+1)\right]} + \frac{12\left(E\left[b_m^2(n)\right]\right)^2}{E\left[\mu_{m+1}(n+1)\mu_{m+1}(n+1)^*\right]} \tag{2.4.5}$$

and from (B.14)

$$P_{m+1}(n+1) = \frac{4E\left[e^*_{m+1\,opt}e^*_{m+1\,opt}\right]E\left[b_m^2(n)\right]}{E\left[\mu_{m+1}(n+1)\mu_{m+1}(n+1)^*\right]} \tag{2.4.6}$$

Substituting (2.4.2) into (2.4.1), and using (2.4.4) yields the MSE at stage m $J_{m+1}(n+1)$ as:

$$J_{m+1}(n+1) = J_{m+1\,opt} + J_{m+1\,exc}(n+1) \tag{2.4.7}$$

where $J_{m+1\,opt}$ is the optimum MSE at stage m defined in (B.21) as

$$\begin{aligned} J_{m+1\,opt} &= E\left[e_{m+1\,opt}(n+1)e^*_{m+1\,opt}(n+1)\right] \\ &= \sigma_n^2 \end{aligned} \tag{2.4.8}$$

Note that σ_n^2 is the observation noise, and $J_{m+1\,exc}(n+1)$ is the excess noise at stage m, defined in (B.22) as:

$$J_{m+1\,exc}(n+1) = K_m(n+1)E\left[b_m(n)b_m^*(n)\right] \tag{2.4.9}$$

At large n, we have as in (B.23), that

$$E\left[b_m(n+1)b_m(n+1)^*\right] = \sigma_{b_m}^2 \tag{2.4.10}$$

and from (B.26) and (B.27), we have the mean and the variance of $\mu_{m+1}(n)$, which are:

$$E\left[\mu_{m+1}(n)\right] = \alpha^{n-1}\,\mu_{m+1}(0) + 2\sigma_{b_m}^2\,\frac{1-\alpha^n}{1-\alpha} \tag{2.4.11}$$

$$E\left[\mu_{m+1}(n)\mu_{m+1}^{\cdot}(n)\right] = \alpha^{2(n-1)}\mu_{m+1}^2(0) + 4\alpha^{n-1}\mu_{m+1}(0)\sigma_{b_m}^2\,\frac{1-\alpha^n}{1-\alpha} + 12\sigma_{b_m}^4\left(\frac{1-\alpha^n}{1-\alpha}\right)^2 \tag{2.4.12}$$

As $n \to \infty$, we have from Appendix B the following final-values:

$$E\left[\mu_{m+1}(n)\right] \to 2\sigma_{b_m}^2\,\frac{1}{1-\alpha} \tag{2.4.13}$$

$$E\left[\mu_{m+1}(n)\mu_{m+1}^{\cdot}(n)\right] \to 4\sigma_{b_m}^4\left(\frac{1}{1-\alpha}\right)^2 \tag{2.4.14}$$

$$B_{m+1}(n+1) \to 2-4\alpha+3\alpha^2 \tag{2.4.15}$$

$$P_{m+1}(n+1) \to \frac{\sigma_n^2(1-\alpha)^2}{\sigma_{b_m}^2} \tag{2.4.16}$$

$$K_m(n+1) \to \frac{\sigma_n^2(1-\alpha)}{\sigma_{b_m}^2(1+\alpha)} \tag{2.4.17}$$

so that as $n\to\infty$, the final value of the MSE as in (B.35) is

$$J_{m+1}(n+1) \to \sigma_n^2 + \frac{\sigma_n^2(1-\alpha)}{(1+\alpha)} \tag{2.4.18}$$

where α is the forgetting factor.

2.4.1.1 The MSE for Multi-stage GAL-JP Algorithm

If we write the error of m stages of the GAL-JP algorithm with respect to the desired response, we can rewrite (2.4.2) as:

$$e_1(n+1) = d(n+1) - c_0^*(n+1)b_0(n+1)$$

$$e_2(n+1) = e_1(n+1) - c_1^*(n+1)b_1(n+1)$$

$$= d(n+1) - c_0^*(n+1)b_0(n+1) - c_1^*(n+1)b_1(n+1)$$

for stage m, the error is

$$e_{m+1}(n+1) = d(n+1) - c_1^*(n+1)b_1(n+1) - \cdots - c_m^*(n+1)b_m(n+1)$$

$$= d(n+1) - \sum_{i=0}^{m} c_i^*(n+1)b_i(n+1) \qquad (2.4.19)$$

where d(n+1) is the desired response. Replacing the regression coefficient $c_i(n+1)$ with the tap-weight error vector $\varepsilon_i(n+1)$ as in (2.4.3), we can rewrite $e_{m+1}(n+1)$ as:

$$e_{m+1}(n+1) = e_{m+1\,opt} - \sum_{i=0}^{m} \varepsilon_i^*(n+1)b_i(n+1) \qquad (2.4.20)$$

The MSE of the m-stage GAL-JP algorithm, $J_{m+1}(n)$, is:

$$J_{m+1}(n+1) = E\left[e_{m+1}(n+1)e^{*}_{m+1}(n+1)\right] \tag{2.4.21}$$

and from (B.47) in Appendix B, we have for the MSE of the m-stage GAL-JP algorithm:

$$J_{m+1}(n+1) = \sigma_n^2 + \left[\sum_{i=0}^{m} K_i(n+1)E\left[b_i(n+1)b_i^{*}(n+1)\right]\right] \tag{2.4.22}$$

where $K_i(n+1)$ is the variance of the regression-coefficient error defined as in (2.4.4), and $\varepsilon_i(n+1)$ is the regression-coefficient-error vector defined as in (2.4.3).

As $n\rightarrow\infty$, by substituting the final value of $K_i(n+1)$ as in (B.35) into (2.4.22), we get the final MSE $J_{m+1\ opt}(n+1)$ of the m-stage GAL-JP algorithm, as we have in Appendix B (B.48):

$$J_{m+1}(n+1) = \sigma_n^2 + \frac{\sigma_n^2(1-\alpha)}{(1+\alpha)}(m+1) \tag{2.4.23}$$

Equation (2.4.23) indicates that the MSE of the GAL-JP algorithm is affected by the order, the higher the order, the higher the MSE. This is true if the order of the lattice is the same as or higher than the order of the identified system, so that the optimum error $J_{m+1\ opt}$ is contributed by the observation noise only. On the other hand, if the order of the lattice is smaller than the order of the identified system, the optimum error $J_{m+1\ opt}$ does not come from observation noise only but also from the underdetermined parametric estimation, so that it is

possible, and even likely, that the MSE of a lower order GAL-JP is higher than the MSE of a higher order one.

The GAL-JP is simulated using Data Set 1. The order of the GAL-JP algorithm is 32 and the forgetting factor α is 0.999. The graph of an ensemble average of 20 independent trials with 4,900 iterations for each trial is shown in Figure 2.4.2, along with the theoretical value.

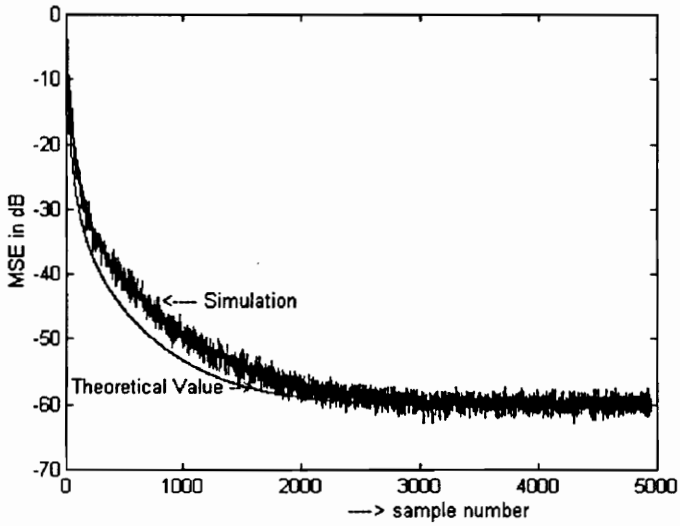

Figure 2.4.2 The MSE for the GAL-JP Algorithm for a Forgetting Factor of .999, and an Order of 32

The MSE for the simulation, which is calculated by taking the average of the last 900 samples of the squared error, is 1.0484×10^{-6}, while the theoretical value of the MSE is1.0165×10^{-6}. So, the simulation result agrees well with the theoretical value for the MSE. Furthermore, the final value of the MSE can be computed from (2.4.23) and it gives the value 1.0053×10^{-6}. This means that for $n \rightarrow \infty$ the GAL-JP algorithm will eventually reach the final MSE value of 1.0053×10^{-6}.

Figure 2.4.2 also shows that the theoretical MSE value converges faster than the simulation result. This is because the theoretical MSE value is computed by assuming that the prediction sections of the GAL-JP algorithm have already converged, in other words the *backward prediction errors* $b_m(n+1)$ have already reached their final values.

By taking the expected value of the regression-coefficient error at stage m, $\varepsilon_m(n+1)$ in (B.5.), and assuming that $b_m(n)$ and $e_{m+1\ opt}$ are uncorrelated, we obtain:

$$E\left[\varepsilon_m(n+1)\right] \approx \left[1 - \frac{2E\left[b_m(n)\, b_m^*(n)\right]}{E\left[\mu_{m+1}(n)\right]}\right] E\left[\varepsilon_m(n)\right] \qquad (2.4.24)$$

By iterating (2.4.17) for large n , we get, as in (B.40),

$$E\left[\varepsilon_m(n+1)\right] = \left[1 - \frac{2\,E\left[b_m(n)\, b_m^*(n)\right]}{E\left[\mu_{m+1}(n)\right]}\right]^n \varepsilon_m(0) \qquad (2.4.25)$$

The time constant τ_{m+1} for stage m of the GAL-JP algorithm is then,

$$\tau_{m+1} = \frac{-1}{\ln\left(1 - \frac{2\,E\left[b_m(n)\, b_m^*(n)\right]}{E\left[\mu_{m+1}(n)\right]}\right)} \qquad (2.4.26)$$

Figure 2.4.3 shows that the trajectory of the expected value of the third regression coefficient $c_3(n)$ is close to the theoretical value. The experiment is the same as that used to simulate the MSE above. Observe that the

convergence behavior of $c_3(n)$ is like the overal convergence in Figure 2.4.2, i.e. complete convergence after about 2,000 iterations.

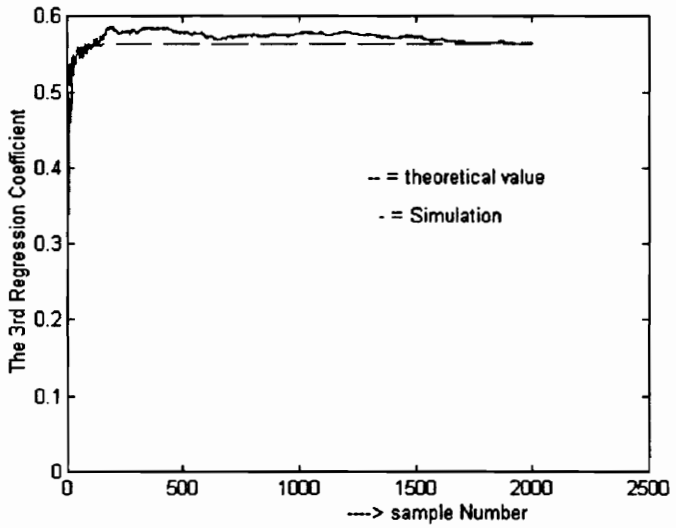

Figure 2.4.3.a Trajectory of the Third Regression Coefficient $c_3(n)$

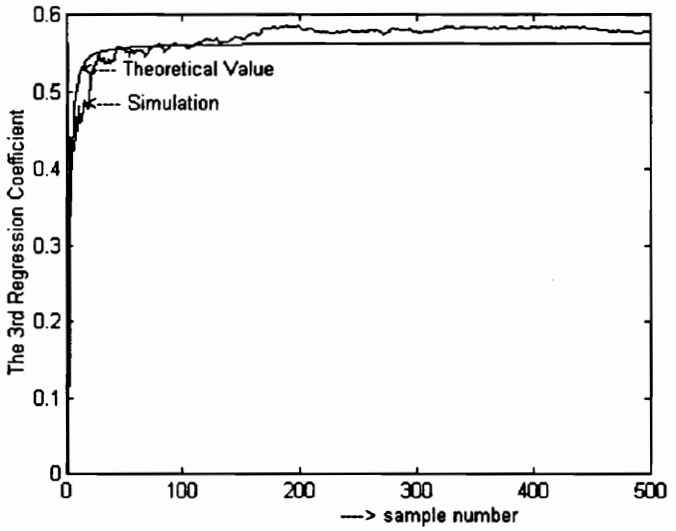

Figure 2.4.3.b A Closer Look at the Trajectory of the Third Regression Coefficient $c_3(n)$

2.4.2 The Recursive Least Squares Lattice Joint Process Algorithm

In the *recursive least-squares lattice joint process* (RLSL-JP) algorithm, the regression coefficients are adapted with the goal to minimize the *sum of weighted squared errors*. There are two types of RLSL-JP algorithm: the first type uses *a priori* errors and the second type uses *a posteriori* errors. Furthermore, each of these can be adapted using error feedback (direct update), or without error feedback (indirect update). The direct update gives better numerical accuracy [26] as well as better numerical performance in a finite word length environment [31]. Also, since the RLSL-JP algorithm solves the exact least squares problem, it gives better performance than the GAL-JP algorithm which uses a gradient approximation in computing the parameters [31].

For some applications, such as echo cancellation and adaptive equalization, the RLSL-JP algorithm using a posteriori errors can not be used [33], because the *regression coefficient* $c_m(n)$ instead of $c_m(n-1)$ has to be used to compute the estimate of the desired response, yet $c_m(n)$ can not be computed prior to the availability of the desired response d(n) [33]. Therefore, only the RLSL-JP algorithm using a priori estimation errors with error feedback will be considered here.

The structure of the RLSL-JP algorithm is the same as the structure of GAL-JP as shown in Figure 2.4.1. The complete summary of the RLSL-JP algorithm is given in Appendix C.1.

The expected value of the conversion factor as expressed in (C.2.8) Appendix C.2 is

$$E[\gamma_m(n)] = \alpha^m \gamma_0(n) \qquad (2.4.2.1)$$

Define the *regression coefficient error* at stage m as:

$$\varepsilon_m(n) = c_m(n) - c_{m\,opt} \tag{2.4.2.3}$$

where $c_{m\,opt}$ is the optimum value of the regression coefficient at stage m. The expected value of the *regression coefficient error* at stage m can be written in recursive form, as shown in (C.2.18), under the assumption that the previous stages have converged:

$$E[\varepsilon_m(n)] = \alpha E[\varepsilon_m(n-1)]$$

$$= \alpha^n \varepsilon_m(0) \tag{2.4.2.4}$$

The time constant for convergence in the mean, τ_m, is :

$$\tau_m = \frac{-1}{\ln \alpha} \tag{2.4.2.5}$$

The error at the output of stage m, $e_{m+1}(n)$, is defined as:

$$e_{m+1}(n) = e_m(n) - c_m^*(n-1)\psi_m(n)$$

$$= e_{m+1\,opt}(n) - \varepsilon_m^*(n-1)\psi_m(n) \tag{2.4.2.6}$$

where $\varepsilon^{*}_{m}(n-1)$ is defined in (2.4.2.3), and the MSE at stage m, $J_{m+1}(n)$, is defined as:

$$J_{m+1}(n) = E\left[e_{m+1}(n)e^{*}_{m+1}(n)\right] \tag{2.4.2.7}$$

Replacing $e_{m+1}(n)$ as in (2.4.2.6) and assuming that the previous stages have converged, $J_{m+1}(n)$ can be expressed, as derived in Appendix C.2, as

$$J_{m+1}(n) = E\left[e_{m+1\,opt}(n)e^{*}_{m+1\,opt}(n)\right] + E\left[\left|\psi_{m}(n)\right|^{2}\right] K_{m}(n-1) \tag{2.4.2.8}$$

where under the optimum condition, with the length of the lattice longer than the autocorrelation depth of the input signal, $E[e_{m+1\,opt}(n)e^{*}_{m+1\,opt}(n)]$ is equal to the *observation noise* σ_{n}^{2}, and the variance of the regression coefficient error $K_{m}(n-1)$ as in (C.2.25) is

$$K_{m}(n-1) = \alpha^{2n} K_{m}(0) + \sigma_{n}^{2}(1-\alpha)^{2} \sum_{i=0}^{n-1} \frac{\alpha^{2(n-1-i)}}{E\left[\left|\psi_{m}(i)\right|^{2}\right]} \tag{2.4.2.9}$$

The MSE for large n, as derived in (C.2.30), is

$$J_{m+1}(n) = \sigma_{n}^{2}\left(\frac{2}{1+\alpha}\right) \tag{2.4.2.10}$$

2.4.2.1 The MSE for the Multi-stage RLSL-JP Algorithm

If we write the error of m stages of the RLSL-JP algorithm with respect to the desired response, we can rewrite (2.4.2.6), as in (C.2.32) as:

$$e_{m+1}(n) = d(n) - \sum_{i=0}^{m} c_i^*(n-1)\,\psi_i(n) \qquad (2.4.2.11)$$

where d(n) is the desired response. Substituting for the regression coefficient $c_i(n-1)$ as in (2.4.2.3), we can rewrite $e_{m+1}(n)$ as:

$$e_{m+1}(n) = e_{m+1\,opt}(n) - \sum_{i=0}^{m} \varepsilon_i^*(n-1)\,\psi_i(n) \qquad (2.4.2.12)$$

The MSE for the m-stage RLSL-JP algorithm, $J_{m+1}(n)$, is:

$$J_{m+1}(n) = E\left[e_{m+1}(n)e_{m+1}^*(n)\right] \qquad (2.4.2.13)$$

and from (C.2.37) in Appendix C.2, we have for the MSE of the m-stage RLSL-JP algorithm

$$J_{m+1}(n) = \sigma_n^2 + \left[\sum_{i=0}^{m} K_i(n-1)E\left[\left|\psi_i(n)\right|^2\right]\right] \qquad (2.4.2.14)$$

where $K_i(n-1)$ is the variance of the regression-coefficient error defined as in (2.4.2.9)

At large n so that

$$E\left[\psi_i(n+1)\psi_i^*(n+1)\right] = E\left[\psi_i(n)\psi_i^*(n)\right]$$

$$= \sigma_{\psi_i}^2 \qquad (2.4.2.15)$$

we can write (2.4.2.14) as

$$J_{m+1}(n) = \sigma_n^2 + \left[\sum_{i=0}^{m} K_i(n-1)\sigma_{\nu_i}^2\right] \tag{2.4.2.16}$$

At large n, by using the final value of $K_i(n-1)$ as in (C.2.29) in (2.4.16), we get the final MSE $J_{m+1}(n)$ for the m-stage RLSL-JP, as we have in (C.2.38):

$$J_{m+1}(n) = \sigma_n^2 + \frac{\sigma_n^2(1-\alpha)}{(1+\alpha)}(m+1) \tag{2.4.17}$$

A simulation for the order 32 RLSL-JP algorithm using Data Set 1 is performed. The trajectory of the conversion factors for different stages is shown in Figure 2.4.4. This plot verifies (2.4.2.4) that the final value of the conversion factor is dependent on the stage m as well as on the forgetting factor α. The MSE is shown in Figure 2.4.5, along with the theoretical value. The MSE of the simulation is an ensemble average of 20 independent trials with 1,000 iterations for each trial. The theoretical MSE for iteration 1,000 is 1.8625×10^{-6} and the MSE of the simulation, which is calculated by taking the average value of the last 500 points of the squared error, is 1.989×10^{-6}. This graph shows a good agreement between the simulation and the theoretical value. Furthermore, the trajectory of the second regression coefficient is shown in Figures 2.4.6.a and 2.4.6.b along with the theoretical value. The trajectory of the theoretical value also agrees well with the simulation.

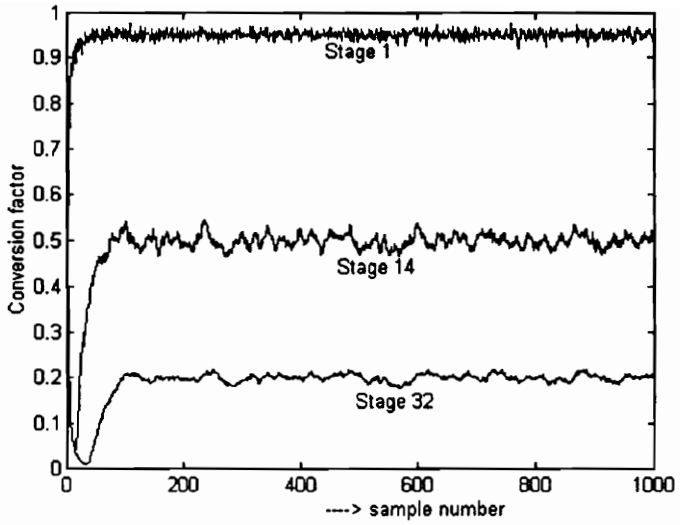

Figure 2.4.4 The Conversion Factors of the RLSL-JP Algorithm for Different Stages; α = .95 and the order of 32

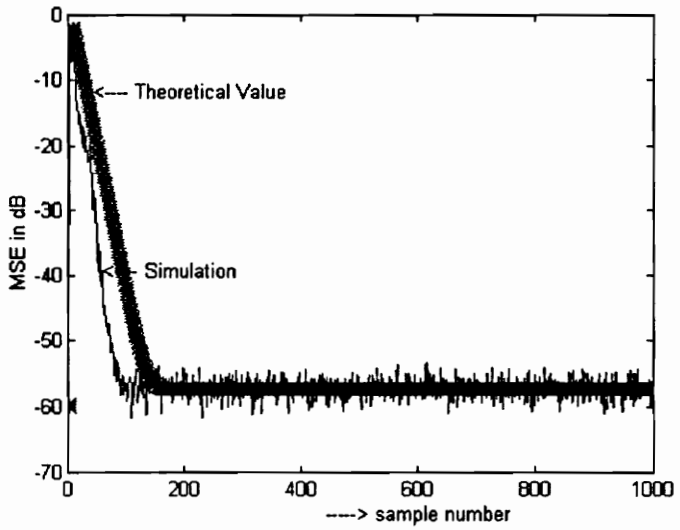

Figure 2.4.5. The MSE for the RLSL-JP Algorithm; α = .95, and the order of 32

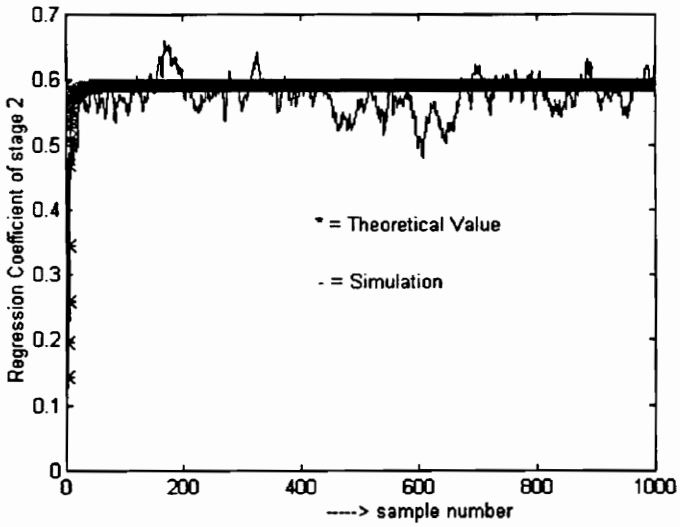

Figure 2.4.6.a Trajectory of the Regression Coefficient of the RLSL-JP Algorithm at Stage 2

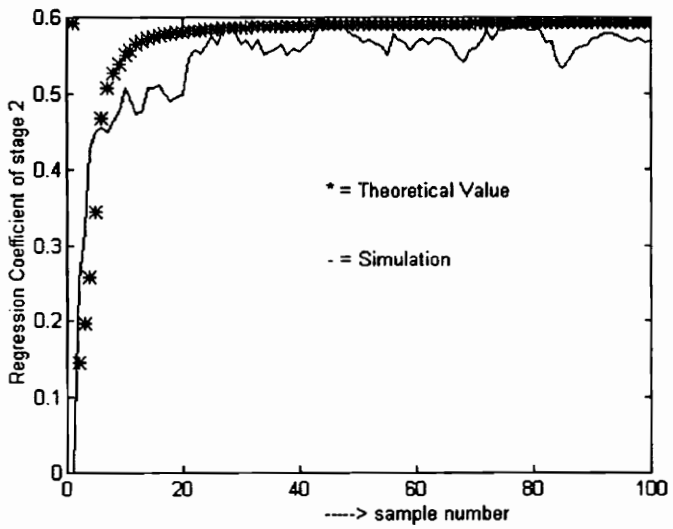

Figure 2.4.6.b A Closer Look at the Trajectory of the Regression Coefficient of theRLSL-JP Algorithm at Stage 2

2.5 Conversions from the Lattice Joint Process Structure to the Transversal Structure

This section explains the procedures for converting from the lattice structure joint process estimator to the transversal filter and vice versa. This conversion is needed when, during operation, we want to switch from one structure to another and we want to have a smooth transition process.

We assume that the conversion from a lattice-structure-based algorithm to a transversal-structure-based algorithm is done after the former algorithm has converged. This implies that the conversion takes place under the condition of stationarity of the input data. Furthermore, because the coefficients of the transversal structure are derived by converting the coefficients of the lattice structure, it acts as a continuation process of the lattice structure, and hence the transversal-structure based algorithm will not experience the usual convergence period or transient condition. The subsequent derivation of the conversion from a lattice structure to a transversal structure is thus based on the stationary input assumption.

The transversal backward prediction error filter is shown in Figure 2.5.1, where the *backward prediction error* of order M, $b_M(i)$, is expressed as:

$$b_M(i) = \underline{a}_{M,k}^H(n)\, \underline{u}_{M+1}(i) \qquad i>M,\ M=0,1,2\ldots,\ \text{and } 1<i<n, \tag{2.5.1}$$

$\underline{a}_{M,k}(n)$ are the *coefficients* of the backward prediction error filter, defined as

$$\underline{a}_{M,k}(n) = \left[a_{M,M}(n),\ a_{M,M-1}(n),\ \ldots,\ 1\right]^T,$$

and $\underline{u}_{M+1}(i)$ is the tap-input vector, defined as:

$$\underline{u}_{M+1}(i) = [u(i), u(i-1), \ldots, u(i-M)]^T$$

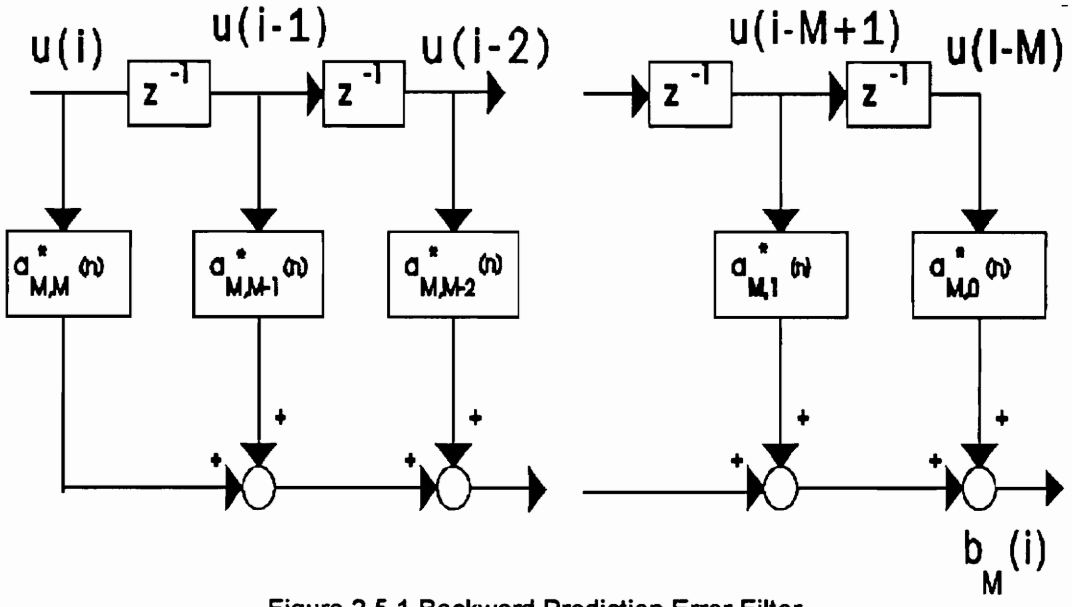

Figure 2.5.1 Backward Prediction Error Filter

Using (2.5.1), we can compute the *backward prediction error* for any order of the backward prediction error filter. In matrix form, the *backward prediction error* vector $\underline{b}_M(i)$ can be written as:

$$\underline{b}_M(i) = T(n)\, \underline{u}_{M+1}(i) \qquad (2.5.2)$$

where $\underline{b}_M(i)$ is the M-by-1 backward prediction error vector, defined as:

$$\underline{b}_M(i) = [b_0(i), b_1(i), \ldots, b_M(i)]^T$$

T(n) is an M-by-M conversion matrix defined as

$$T(n) = \begin{bmatrix} 1 & 0 & . & 0 \\ a_{1,1}(n) & 1 & . & 0 \\ . & . & . & . \\ a_{M,M}(n) & a_{M,M-1}(n) & . & 1 \end{bmatrix}$$

and $\underline{a}_{M,k}(n)$ are the coefficients of the backward prediction error filter of order M.

For a lattice structure with the forward reflection coefficients $\Gamma_{f,k}$ and the backward reflection coefficients $\Gamma_{b,k}$, the *forward prediction error filter* coefficient vector $\underline{g}_{m,k}$ and the *backward prediction error filter* coefficient vector $\underline{a}_{m,k}$ are related as follows [16]:

$$\underline{g}_m(n) = \begin{bmatrix} \underline{g}_{m-1}(n) \\ 0 \end{bmatrix} + \Gamma_{f,m} \begin{bmatrix} 0 \\ \underline{a}_{m-1}(n-1) \end{bmatrix} \tag{2.5.3}$$

$$\underline{a}_m(n) = \begin{bmatrix} 0 \\ \underline{a}_{m-1}(n-1) \end{bmatrix} + \Gamma_{b,m} \begin{bmatrix} \underline{g}_{m-1}(n) \\ 0 \end{bmatrix} \tag{2.5.4}$$

where m= 1,..., M, $g_0(n) = 1$, $a_0(n) = 1$, and n is the time index.

Using (2.5.3) and (2.5.4) we can compute the backward prediction error filter coefficients recursively. In the case that the forward and backward reflection coefficients are assumed to be the same, such as in the *gradient adaptive lattice,* we can still use (2.5.3) and (2.6.4) with $\Gamma_{f,m}$ and $\Gamma_{b,m}$ taken to be the same.

The structure of the *lattice joint process estimator* is shown in Figure 2.4.1.b. The estimate of the desired response $\hat{d}(n)$ is

$$\hat{d}(n) = \underline{c}_M^H(n)\, \underline{b}_M(n) \tag{2.5.5}$$

where $\underline{c}_M(n)$ are the regression coefficients of the lattice joint process estimator defined as:

$$\underline{c}_M(n) = \left[c_0(n),\ c_1(n),\ \ldots,\ c_M(n)\right]^T$$

The transversal filter that is equivalent to the *lattice joint process estimator* is shown in Figure 2.5.2. The estimate of the desired response $\hat{d}(n)$ produced by the transversal filter is:

$$\hat{d}(n) = \underline{w}_M^H(n)\, \underline{u}_{M+1}(n) \qquad (2.5.6)$$

where $\underline{w}_M(n)$ are the coefficients of the transversal filter defined as:

$$\underline{w}_M(n) = \left[w_0(n),\ w_1(n),\ \ldots,\ w_M(n)\right]^T$$

By equating (2.5.5) and (2.5.6), and using (2.5.2) to replace $\underline{b}_M(n)$, we get:

$$\underline{w}_M^H(n)\, \underline{u}_{M+1}(n) = \underline{c}_M^H(n)\, T(n)\underline{u}_{M+1}(n) \qquad (2.5.7)$$

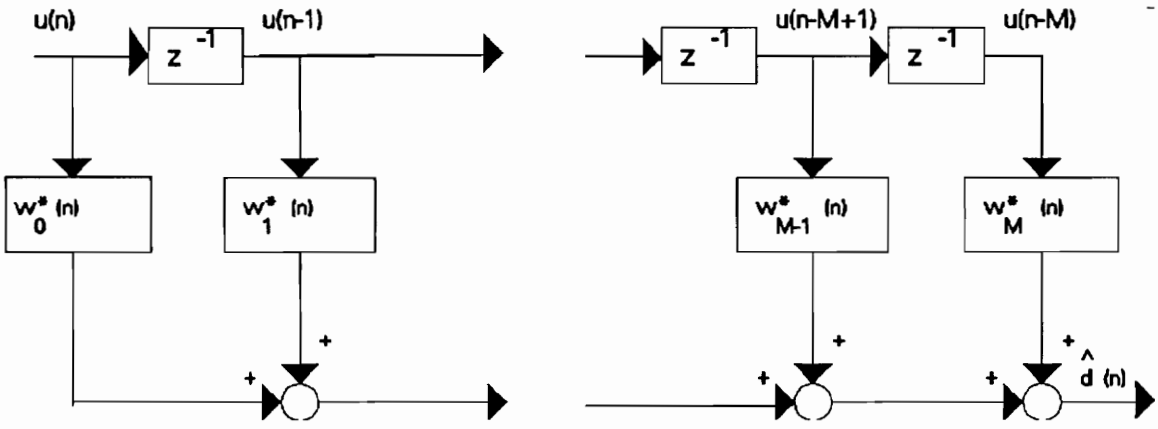

Figure 2.5.2 Transversal Filter Equivalent to Lattice Joint Process

Equation (2.5.7) holds for vector $\underline{u}_{M+1}(n)$ at any time n, therefore, the tap-weights of the transversal filter and the regression coefficients of the *lattice joint process estimator* are related as:

$$\underline{w}_M^H(n) = \underline{c}_M^H(n)\, T(n) \qquad (2.5.8)$$

Equation (2.5.8) is used to convert the regression coefficients of the *lattice joint process estimator* $\underline{c}_M(n)$ to the tap-weights of the transversal filter $\underline{w}_M(n)$.

In the adaptive filtering context, where the input data changes every time sample during the operation of the lattice structure, the tap-input vector $\underline{u}_{M+1}(n)$ of the transversal filter structure has to be filled with recent data samples even if the coefficient vector $\underline{w}_M(n)$ is not being adapted, otherwise the transversal filter structure will experience a transient condition when we switch from the lattice algorithm to the transversal algorithm.

CHAPTER THREE

SIMULATION DESCRIPTION

This chapter describes the simulation set up for switching between structures based on the echo cancellation application, whereas the simulation results will be presented in Chapter 4. This chapter is organized as follows: the first subsection gives a brief description of the specification of an echo canceller based on CCITT Recommendation G.165 [3]. The second subsection describes methods for reducing the order of an adaptive filter, which is important in decreasing the computational load. The third subsection explains the simulation procedures.

3.1 Specification and Simulation Setup

Most echo cancellers nowadays follow the specification of CCITT Recommendation G.165 [3]. The simulation here will use two important specifications of CCITT Rec. G.165 concerning the *rate of convergence* and *cancellation at steady-state*. We will discuss these two issues only because they relate directly to the merit of the adaptive filter. The other specifications such as distortion, leak-rate, etc., are very much dependent on the details of the hardware implementation of the echo canceller, which is outside the scope of the present investigation.

3.1.1 Rate of Convergence

CCITT Recommendation G.165 [3, pp.224] describes the rate of convergence as:

" the interval between the instant a defined test signal is applied to the receive-in port of an echo canceller with the estimated echo path impulse response initially set to zero, and the instant the returned echo level at the send-out port reaches a defined level"

The rate of convergence is specified as 500 ms for the echo canceller to have a minimum total attenuation of 27 dB. The total attenuation includes hybrid attenuation, echo attenuation, and nonlinear processing attenuation as shown in Figure 3.1.1. The non-linear processing block represents a non-linear device that is used to suppress the echo signal even further. Since we use a linear adaptive filter, any non linearity in the system to be identified consequently can not be modeled and comes out as an extraneous error. This error hopefully can be rejected by the non-linear device. One such non-linear device is a threshold circuit. Whenever the output error of the adaptive filter is less than its threshold value, this error will be suppressed even further, however if the error of the adaptive filter is bigger than its threshold value, this error is just passed through.

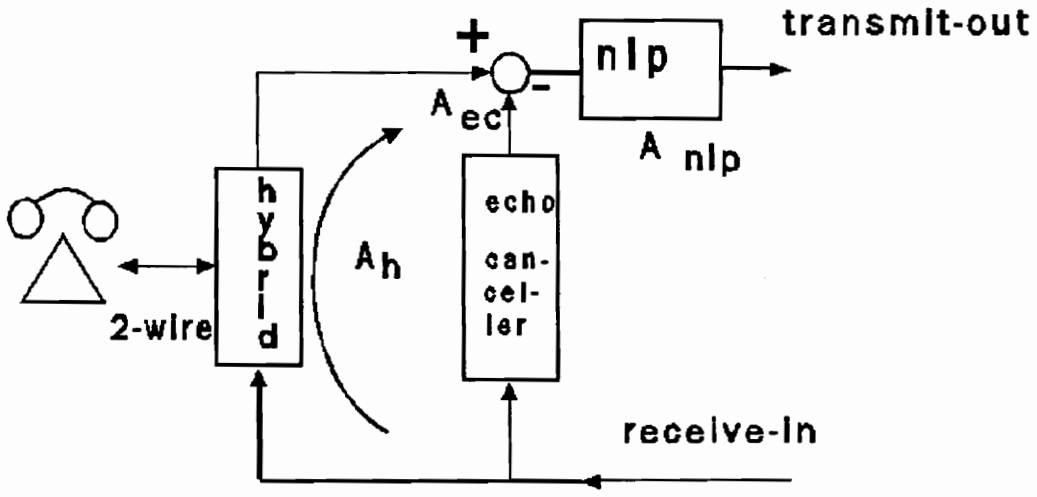

Figure 3.1.1 Attenuation in a Telephone System

In equation form, we have:

$$A_t = A_{ec} + A_h + A_{nlp} \quad (3.1.1)$$

where A_t is the total attenuation, A_h is the attenuation due to the hybrid, A_{ec} is the attenuation provided by the echo canceller, and A_{nlp} is the attenuation provided by the non-linear processing device. CCITT Recommendation G.165 [3] specifies that the echo canceller should fulfill the recommendation based on the hybrid attenuation A_h of 6 dB or greater.

For this simulation, the specification will be tightened so that the minimum total attenuation of at least 27 dB is provided by the echo canceller only. That is, we set

$$A_t = A_{ec} \quad (3.1.2)$$

This also means that the error between the desired signal and the estimate of the desired signal is at most -27 dB in less than 500 ms (4,000 iterations at an 8 kHz sampling rate).

3.1.2 Cancellation at Steady-state

CCITT Recommendation G.165 [3, pp.223] defines the cancellation at steady-state as the final cancellation of an echo canceller so that its output consists of the residual echo only; the final cancellation includes the hybrid attenuation. The residual echo level should satisfy Figure 3.1.2 (this figure is taken from Figure 7/G.165 Recommendation G.165). Figure 3.1.2 indicates that the attenuation of an echo canceller, including the hybrid attenuation of 6 dB, is a minimum of 18 dB for the lowest received-in test signal of -30 dBm0, up to a

minimum of 26 dB attenuation for the highest received-in test signal of -10 dBm0. The steady state should be attained within at most 2 seconds (16,000 iterations at an 8 kHz sampling rate).

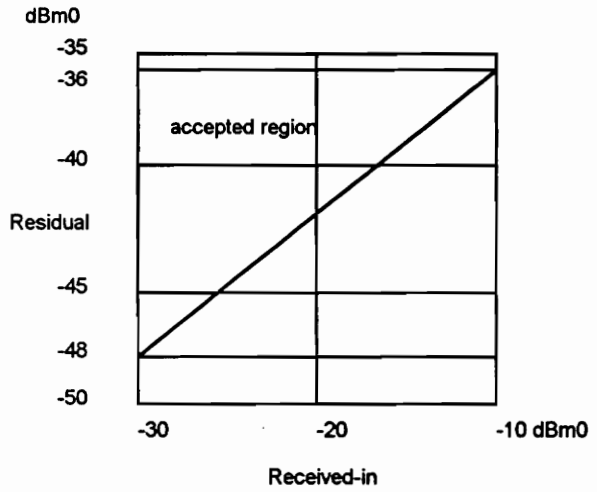

Figure 3.1.2 Specification of Residual Echo Level

The reason for choosing the test signal of -10 dBm0, meaning the level of the test signal is 10 dB below the nominal level of the receive path, is to prevent the possibility that the test signal exceeds the allowable level of the receive path, which can produce *clipping*. If *clipping* happens the actual level of the test signal is not known, and hence the measurement will not be valid. On the other hand, the minimum test signal is 30 dB below the nominal level which is hopefully not too small, so that the measurement can still be made accurately.

For this simulation, the attenuation requirement will be tightened up so that the adaptive filter by itself provides the minimum required attenuation of 18 dB up to 26 dB of attenuation for the specified received-in signal levels above.

3.2 Reducing the Computational Order of the Adaptive Filter

We know that the lower the order of the adaptive filter the faster the computation and the lower the MSE. Since the impulse response of the echo-path has a pure delay followed by the impulse response of the hybrid, we can reduce the computational order of the adaptive filter used to model the impulse response of the hybrid, if we can identify this delay. Furthermore, the impulse response of the hybrid itself also has some values equal to or close to zero, so that if we can pinpoint these zeros, we can reduce the computational order of the adaptive filter even further.

The pure delay of the echo-path can be identified from the cross-correlation of the echo signal and the received-in signal. However, from this procedure, we can identify the length of the delay only and not the impulse response values that equal zero.

The second way to reduce the order is by putting a threshold on the estimated impulse response values. We use only the impulse response values that exceed the threshold, along with the corresponding input signal samples, to cancel the echo signal. For the simulation, the second method will be used.

The order reduction technique is illustrated in Figures 3.2.1.a and b. Suppose the threshold is zero. In Figure 3.2.1.a we see that the second and the fourth tap-weight are non-zero so that by using the computational order reduction technique, we use only the second and the fourth tap-weight and the filter structure now becomes computationally equivalent to a second order filter as shown in Figure 3.2.1.b.

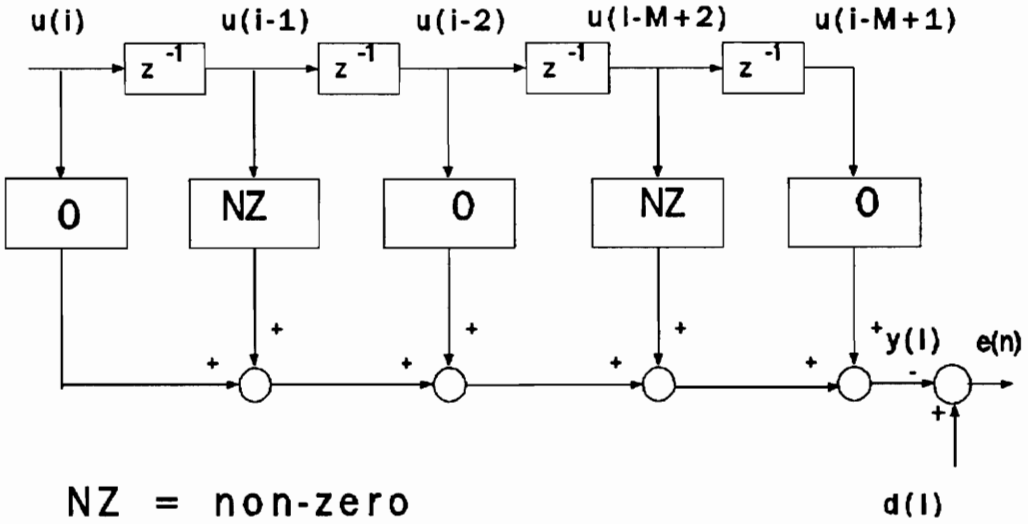

Figure 3.2.1.a The Fifth Order Filter

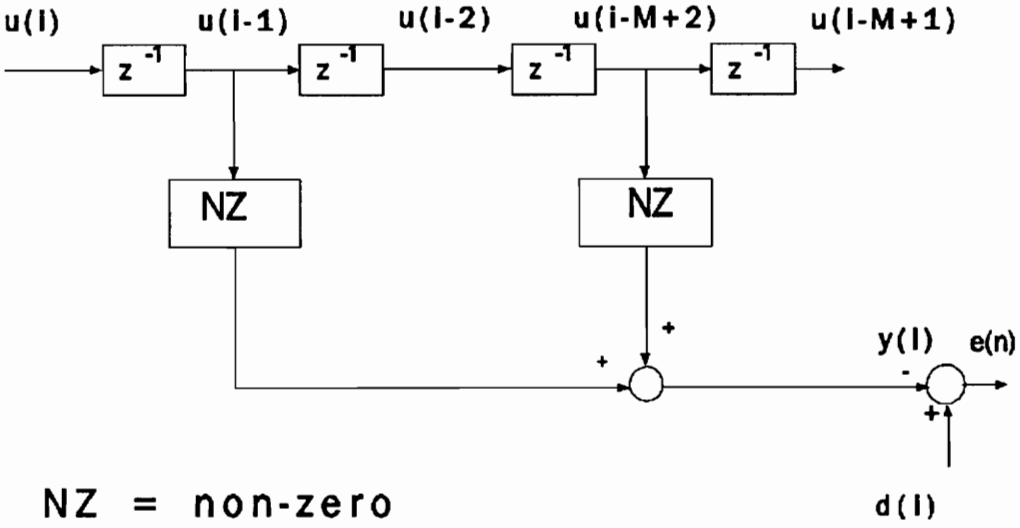

Figure 3.2.1.b The Computationally-Reduced Order Filter

In order to see the effects of the computationally reduced order filter on the MSE, we expand the derivation of the MSE for the LMS algorithm. Recall the tap-weight error vector $\underline{\varepsilon}(n)$ in (2.15) as

$$\underline{\varepsilon}(n) = \underline{\hat{w}}(n) - \underline{w}_{opt} \tag{3.2.1}$$

and the variance of the tap-weight error vector K(n) in (2.16) as:

$$K(n) = E\left[\underline{\varepsilon}(n)\underline{\varepsilon}^{H}(n)\right] \tag{3.2.2}$$

If Q is a unitary transform matrix for the auto-correlation matrix of the tap-input data R, we have:

$$\Lambda = Q^{H}RQ \tag{3.2.3}$$

By multiplying K(n) with Q, we have

$$X(n) = Q^{H}K(n)Q \tag{3.2.4}$$

From [16] we know the excess squared error $J_{ex}(n)$ as

$$J_{ex}(n) = tr(RK(n)) \tag{3.2.5}$$

By replacing R and K(n) in (3.2.5) with Λ and X(n), respectively, we can rewrite (3.2.5) as:

$$J_{ex}(n) = tr(Q\Lambda Q^{H}QX(n)Q^{H}) \tag{3.2.6}$$

since Λ is a diagonal matrix, we can write $J_{ex}(n)$ in (3.2.6) as

$$J_{ex}(n) = \sum_{i=1}^{M} \lambda_i x_i(n) \tag{3.2.7}$$

where λ_i are the eigenvalues of R, $x_i(n)$ are the diagonal elements of the matrix X(n), and M is the order of the filter.

Now, to see the effects of the computationally reduced order, let us take a look at an example of a third order filter with the coefficients

$$\underline{w}(n) = \left[w_1(n),\ 0,\ w_2(n)\right]^H \tag{3.2.8}$$

By definition in (3.2.1), the tap weight error vector is

$$\underline{\varepsilon}(n) = \left[\varepsilon_0(n),\ 0,\ \varepsilon_2(n)\right]^H \tag{3.2.9}$$

and the variance of the tap-weight error vector K(n) is

$$K(n) = E\begin{bmatrix} \varepsilon_0\varepsilon_0^* & 0 & \varepsilon_0\varepsilon_1^* \\ 0 & 0 & 0 \\ \varepsilon_1\varepsilon_0^* & 0 & \varepsilon_1\varepsilon_1^* \end{bmatrix} \tag{3.2.10}$$

Suppose the matrix Q consists of q_{ij}, so that by substituting the matrix Q and K(n) into (3.2.4), with the time index n dropped for convenience, we obtain

$$X = \begin{bmatrix} q_{11}^* & q_{21}^* & q_{31}^* \\ q_{12}^* & q_{22}^* & q_{32}^* \\ q_{13}^* & q_{23}^* & q_{33}^* \end{bmatrix} \begin{bmatrix} \varepsilon_0\varepsilon_0^* & 0 & \varepsilon_0\varepsilon_1^* \\ 0 & 0 & 0 \\ \varepsilon_1\varepsilon_0^* & 0 & \varepsilon_1\varepsilon_1^* \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \tag{3.2.11}$$

Generally as the q_{ij} are not zero, X(n) will not have a zero column vector $x_j(n)$ and, as a result, the excess squared error $J_{ex}(n)$, as defined in (3.2.7), consists of summations as the full third order filter. Therefore the computationally reduced order filter does not reduce the excess squared error in the sense that the number of constituting summations remains the same.

However, if we look at (3.2.11), the value of the $x_j(n)$ will be smaller than the value of the $x_j(n)$ when there is no zero coefficient, because if we multiply out (3.2.11), some of the multiplications will give zero values. Consequently, this fact raises the possibility that the computationally reduced order procedure can lower the excess squared error $J_{ex}(n)$.

Furthermore, if Q is a diagonal matrix, the second column of matrix X(n) derived from (3.2.10) will be zero, in which case the excess squared error $J_{ex}(n)$ will be

$$J_{ex}(n) = \lambda_1 x_1(n) + \lambda_3 x_3(n)$$

Therefore, $J_{ex}(n)$ for the computationally reduced order filter could be lower than $J_{ex}(n)$ for the full third order filter. The matrix Q can be a diagonal matrix if the matrix R is a diagonal matrix, which is true if the input data is white Gaussian noise. In short, for a white Gaussian noise input, the computationally reduced order technique will possibly give a smaller MSE.

Since we use two structures, when switching, the computationally reduced order technique is applied to the "low MSE structure". The "fast convergence structure" will identify the whole impulse response or parameters of the echo-path including the delay, and then only non-zero parameters, i.e., the

parameters that are greater than the threshold, are transferred to the "low MSE structure".

If the threshold is not zero we have to be careful, because if the threshold is too large, there are some significant impulse response values which will not be used, and as a result, the computational order reduction technique would give a higher MSE instead. This is the same as the effect of model mismatch.

3.3 Simulation

For the simulation there are several performance characteristics that need to be examined: the rate of convergence, the final MSE, the effect of changes in the echo-path, the effect of the computationally order reduced filter, and the effect of different input signals.

From now on, the "**fast convergence structure**" term will refer to the adaptive filter that is used at the beginning of operation which handles the convergence state or the *fast changing* situation and the "**low MSE structure**" term will refer to the adaptive filter that gives fine cancellation or cancellation at steady state. Note, *fast changing* can be caused by a change in the system to be identified, i.e., a new echo-path, or caused by a change (increase) in the observation noise, i.e., the appearance of a near-end signal; any signal that comes at the desired signal side which is not the desired signal is called observation noise since it disturbs the adaptation process.

In order to be able to accommodate the echo-path delay of 32 ms, the adaptive filter, more precisely the "fast convergence structure" will be designed to have an order of 256 which is equivalent to 32 ms at an 8 kHz sampling rate; the 32 ms delay includes the impulse response of the hybrid. However, the length of the computationally reduced order filters, which is the "low MSE

structure", will vary depending on the number of the non-zero tap-weights which are transferred from the "fast convergence structure".

The "fast convergence structure" will be designed to give the fastest convergence, with the consequence of producing relatively high MSE. The "low MSE structure" on the other hand, will be chosen to produce the lowest MSE, with the consequence of exhibiting relatively slow convergence. The use of both algorithms successively in time is expected to produce fast convergence and low MSE.

3.3.1 Convergence Test and Switching Threshold

Switching from the "fast convergence structure" to the "low MSE structure" will be done if the former structure has already converged according to some pre-defined criterion. The convergence criterion is based on the convergence of the error; the algorithm is considered to have converged if the difference of the averages of the squared error over consecutive intervals falls inside a pre-defined range.

There are two steps involved in deciding whether the "fast convergence structure" has converged: the first step is to compute the average of the squared error of the "fast convergence structure", $e_1^2(n)$, at every iteration which is detailed in Section 2.2.1, and the second step is to compute the difference or the ratio of the average squared error between two consecutive iterations $\Delta_T(n)$, that is:

$$\Delta_T(n) = \frac{e_1^2(n)}{e_1^2(n-1)} \qquad (3.3.1)$$

where $e_1^2(n-1)$ is the average squared error at iteration n-1, and $e_1^2(n)$ is the average squared error at iteration n.

If

$$1-\delta < \Delta_T(n) < 1+\delta, \qquad \delta = .02 \tag{3.3.2}$$

for P consecutive iterations; then the algorithm is considered to have converged and the system switches to the "low MSE structure"; P will be defined later depending on the structure and the algorithm used. Before the adaptive filter has converged, the total output error $e_T(n)$ is taken from the output error of the "fast convergence structure" $e_1(n)$.

The idea of monitoring $\Delta_T(n)$ for P consecutive iterations is to make sure that the error $e_1^2(n)$ is within the value that is considered "flat", as shown in Figure 3.3.1. When the system switches to the "low MSE structure", the parameters of the "fast convergence structure" needed by the second structure are also transferred, after being transformed if necessary. The "low MSE structure", then, starts working to reduce the error even further and now the total output error $e_T(n)$ is taken from the error of the second structure $e_2(n)$.

3.3.2 Switching During Operation

There are two major changes that should be anticipated by the adaptive filter, or system, during operation. Those are a fast change caused by a complete change of the system to be identified, and a fast change caused by the observation noise.

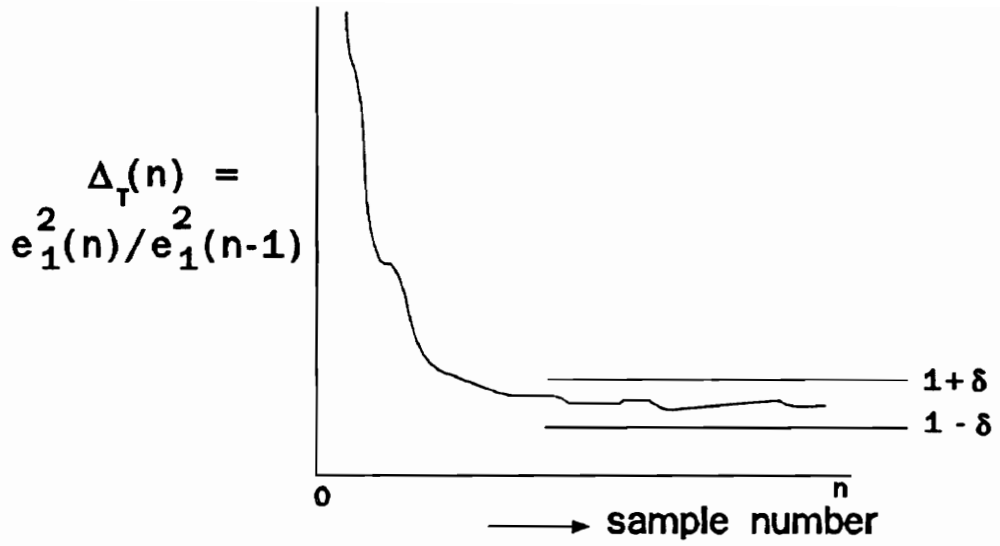

Figure 3.3.1 Reaching the Convergence State

A fast change in the system to be identified is made by changing the gain A of the impulse response of the system from 1 to -1. Before the change, the system to be identified has the input-output relation:

$$y = A(\underline{w}^H \underline{x})$$

where $\underline{w}$ is the tap-weight vector, $\underline{x}$ is the tap-input vector, and A is the gain which is set to 1. For the system change, the gain A is moved to -1, so that the impulse response after the change is the opposite of the impulse response before the change. However, the change of gain A from 1 to -1 is not done instantaneously. The change follows the path shown in Figure 3.3.2. The start of the system change can be made at any sample number.

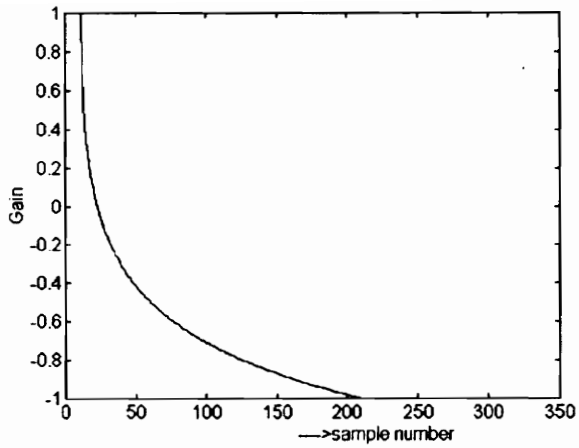

Figure 3.3.2 Gain Change of the System to be Identified

The switched adaptive filter system will consider that there is a fast change if

$$\Delta_T(n) > \gamma, \qquad \gamma = 1.1 \tag{3.3.3}$$

and if this happens, the "low MSE structure" stops adaptation, meaning it acts as a fixed filter, and the system enters a transition period.

During the transition period, both structures are running. The "fast convergence structure" is working to figure out whether the change is caused by a change in the system to be identified, or caused by an increase of the observation noise. Meanwhile, the "low MSE structure" is working as a *fixed filter* using the old parameters. The total output error $e_T^2(n)$ is a weighted sum of the output error of the "fast convergence structure" $e_1^2(n)$ and the "low MSE structure" $e_2^2(n)$, that is

$$e_T^2(n) = (1-w_o^k)e_1^2(n) + w_o^k\, e_2^2(n) \tag{3.3.4}$$

where w_e is the weighting factor and k is the iteration number in the transition period. The latter produces a smooth transition from using $e_2^2(n)$ to $e_1^2(n)$.

The switching procedures using the above transition have some advantages. First, if the change is caused by an increase in the observation noise, the echo can still be canceled by the "low MSE structure" using the old parameters since the system to be identified itself does not change, while the "fast convergence structure" is figuring out the cause of the change; adaptation should be inhibited if the observation error is large in order to prevent the algorithm from diverging. If the "fast convergence structure" has not converged by the end of the transition period, then the change is considered to have been caused by an increase in the observation noise and the "low MSE structure" continually works as a *fixed filter* to cancel the echo until the change has disappeared, which is indicated by a decrease of the output error of the "low MSE structure" $e_2^2(n)$, and then the "low MSE structure" becomes an adaptive filter again.

Second, if the change is caused by a change of the system to be identified, the "fast convergence structure" will be converging, which is indicated by a decrease of its output error $e_1^2(n)$. If this happens, the "low MSE structure" can stop running and the "fast convergence structure" will continue to adapt after the transition period has elapsed, and eventually reach convergence again. Finally, the system will switch back to the "low MSE structure" after the "fast convergence structure" has converged again.

The weighting factor w_e should be chosen such that it will not go to zero very fast or very slowly. The reason is that, if the weighting factor w_e vanishes

very quickly, then the role of the "fast convergence structure" filter will be nullified. If, on the other hand, the weighting factor w_e diminishes very slowly, then the role of the "low MSE structure" will be nullified. This also means that the value of w_e will depend on the length of the transition period because, as indicated by (3.3.4), the weighting factor w_e is raised to the power of the transition iteration number, k.

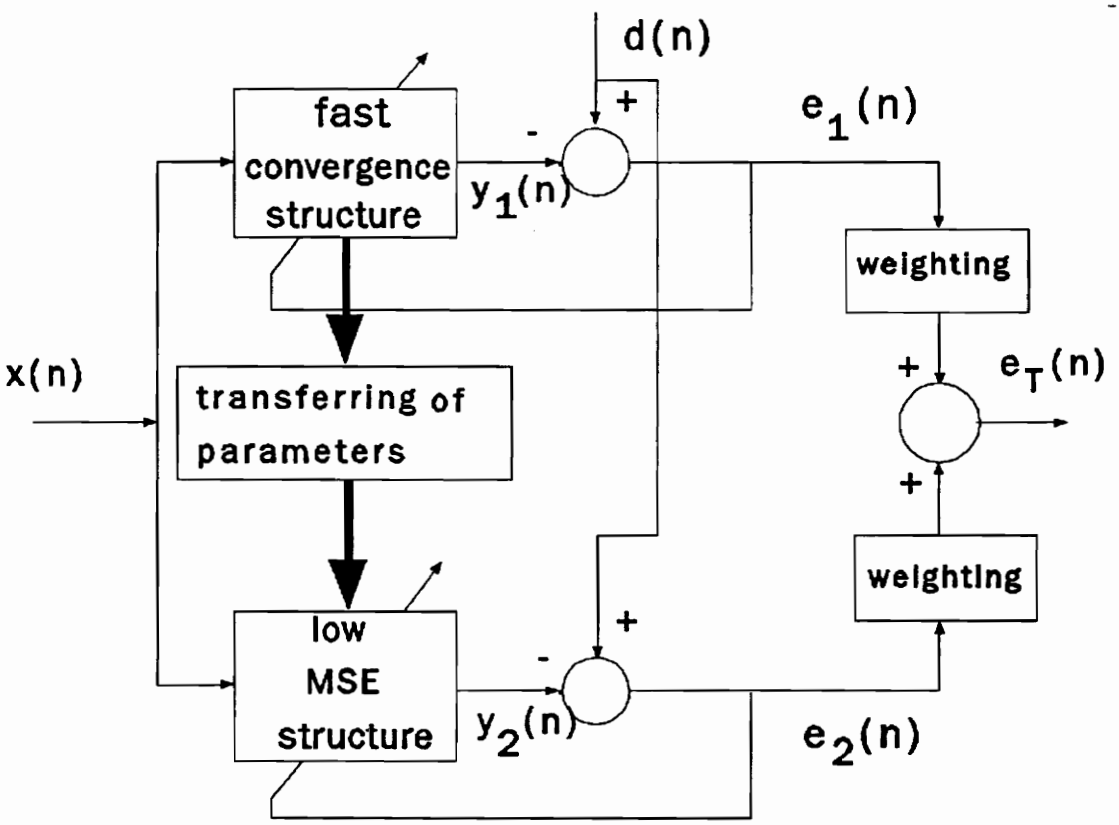

Figure 3.3.3 Diagram of the Switching Structure

In summary, the procedures for switching are as diagrammed in Figure 3.3.3:

1. at the beginning of operation the "fast convergence structure" is working, the total output error $e_T(n)$ is based on the error of the "fast convergence structure" $e_1(n)$, and the average of the square of $e_1(n)$ is monitored;
2. if the "fast convergence structure" has converged as explained in Section 3.3.1, the system switches to the "low MSE structure". The tap-weights of the "fast convergence structure" are transformed, if necessary, and then copied into the "low MSE structure". The total output error $e_T(n)$ is taken from the error of the "low MSE structure" $e_2(n)$, and the average of the square of $e_2(n)$ is monitored;
3. if there is a change in the desired signal d(n), which is indicated by an increase of the error $e_T(n)$ as explained in Section 3.3.2, the adaptation of the "low MSE structure" is inhibited. The system will enter the transition period in which the "fast convergence structure" and the "low MSE structure" are running in parallel;
4. if, at the end of the transition period, the average square of the output error of the "fast convergence structure" $e_1(n)$ decreases, meaning that the "fast convergence structure" is converging, then the "fast convergence structure" will take over because it is then assumed that there is a change in the system to be identified, and the process is as described under 1. If, however, the average squared output error of the "fast convergence structure" $e_1(n)$ does not decrease, this means that the "fast convergence structure" is not converging, so that most probably the change is caused by the appearance of a near-end signal, while the system to be identified did not change. If this happens, the "low MSE structure" will take over and continuously work as a *fixed filter* using the old parameters and the total output $e_T(n)$ will be taken

from the error of the "low MSE structure," $e_2(n)$. By using this scheme, the echo can still be cancelled because the system to be identified actually remains the same; the large error is caused by the near-end signal.

For switching between FIR structures, the "low MSE structure" is always the transversal LMS algorithm because the transversal LMS algorithm is the most robust and exhibits the lowest MSE.

We briefly tried to use an adaptive IIR structure, but it turned out to converge very slow; taking more than 100,000 iterations, even with the initial parameters close to the optimal solution. Also, very often it diverged if the order was a little higher, for example more than 10. The problem comes from the choice of the convergence control parameter, i.e. the step-size. There is still not a good method to choose the convergence control parameter. So, we decided not to use the IIR adaptive filter.

The switching structures that will be investigated are, for transversal filters: FTF/TLMS, and for switching from lattice to transversal filters: GAL/TLMS and RLSL/TLMS; where the / indicates the switching structure and prefix T denotes a transversal filter.

CHAPTER FOUR

SIMULATION RESULTS

The simulation procedures explained in Chapter 3 are applied to several switching structures in this chapter. The input data used in this simulation will be the white Gaussian noise specified as Data Set 1 and the colored noise specified as Data Set 2. The eigenvalue ratio $\chi(R)$ of Data Set 2 depends on the order of the adaptive filter. For an adaptive filter of order 64, $\chi(R)$ is 1.387×10^4 and for an adaptive filter of order 256, $\chi(R)$ is 4.872×10^4.

4.1 Switching Using the RLSL-JP/TLMS Algorithm

4.1.1 Results Using the White Gaussian Noise Input

For the experiment using Data Set 1, which is a zero mean Gaussian noise with unit variance, as the input, the RLSL-JP algorithm is used either as the "fast convergence structure" of the RLSL-JP/TLMS algorithm or as a stand-alone algorithm with the *forgetting factor* α equal to 0.95. This value will give fast convergence for the RLSL-JP algorithm, without causing problems of divergence soon thereafter. Moreover, the TLMS algorithm as the "low MSE structure" will be used with a small step-size μ of .0005 in order to give low MSE.

The result for the RLSL-JP algorithm is shown in Figure 4.1.1 and the result for the RLSL-JP/TLMS algoritm is shown in Figure 4.1.2. Because the computation is very slow, the ensemble average of the RLSL-JP/TLMS is taken only for two independent trials.

We see from Figure 4.1.1 that the RLSL-JP algorithm converges at about iteration 400, and so does the RLSL-JP/TLMS algorithm as shown in Figure 4.1.2. However, by comparing Figures 4.1.1 and 4.1.2, we see that the RLSL-JP/TLMS algorithm in Figure 4.1.2 produces a lower MSE of 1.2355×10^{-6} or -59 dB than the RLSL-JP algorithm with MSE of 9.3114×10^{-6} or -50 dB; the MSE is taken by averaging the last 1,000 samples of the ensemble average of the squared error. The theoretical final value MSE for the RLSL-JP algorithm, using (2.4.17), is -51.14 dB. The additional error reduction of the RLSL-JP/TLMS algorithm is contributed by the TLMS algorithm.

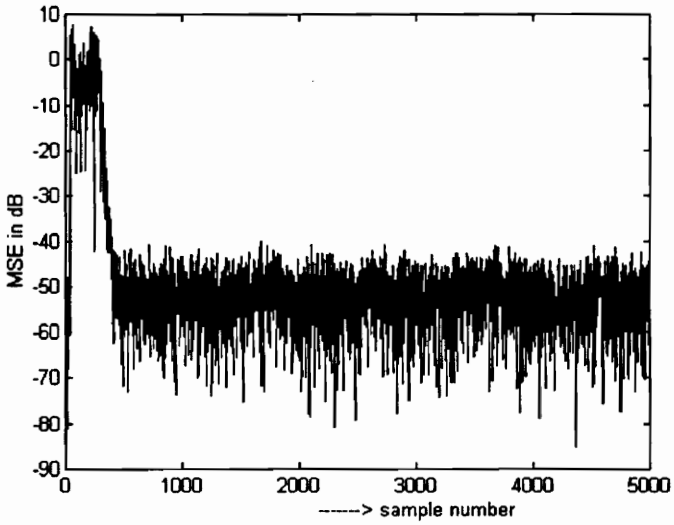

Figure 4.1.1 Two Trial Average MSE for the RLSL-JP Algorithm
α=.95

If we use a sampling time of 125 micro seconds per sample (8 kHz), convergence of the RLSL algorithm, which takes about 400 iterations, takes about 50 ms. This is faster than the convergence rate specified by CCITT Rec. G.165 [3] of 500 ms. Furthermore, the steady state cancellation for the RLSL-JP/TLMS algorithm is about 59 dB and for the RLSL-JP algorithm about 50 dB, both of which far exceed the CCITT Rec. G. 165 requirement of 26 dB.

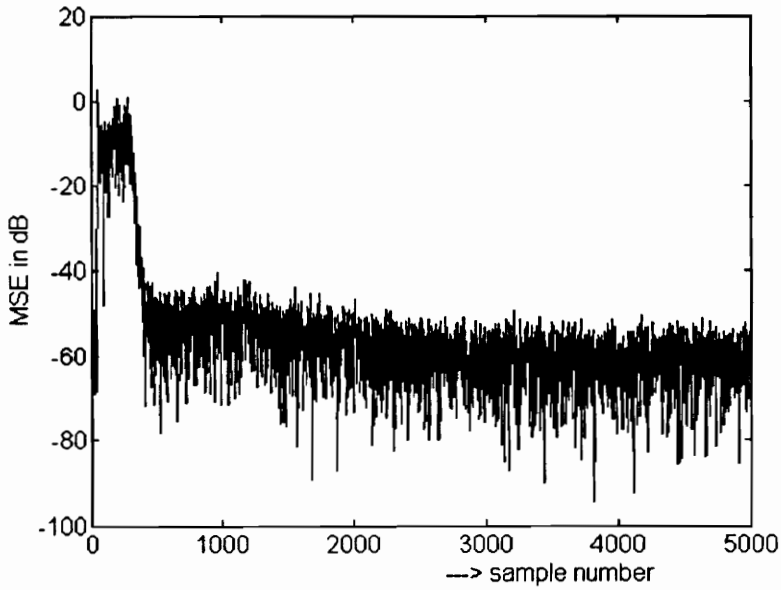

Figure 4.1.2 Two Trial Average MSE for the RLSL-JP/TLMS Algorithm α=.95 and μ =.0005

Next we look at the performance of the RLSL-JP algorithm and the RLSL-JP/TLMS algorithm when the system to be identified changes during operation. The change is made after both algorithms have converged, at iteration 2,500. The change is made by changing the gain of the system to be identified as explained in Section 3.3.2. The result for the RLSL-JP algorithm is shown in Figure 4.1.3 and the result for the RLSL-JP/TLMS is shown in Figure 4.1.4.

By comparing Figure 4.1.3 and 4.1.4, we see no significant difference in convergence. However, we see again there is reduction in MSE for the switching structure as seen in Figure 4.1.4.

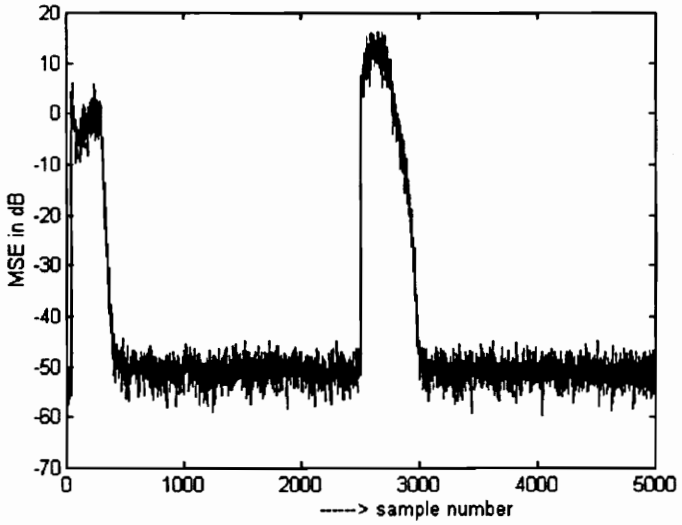

Figure 4.1.3 Ten Trial Average MSE for the RLSL-JP Algorithm
α=.95, gain change at iteration 2,500

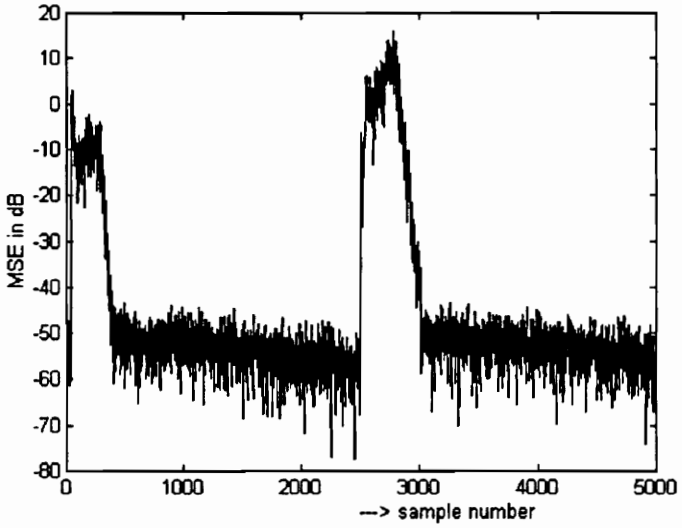

Figure 4.1.4 Ten Trial Average MSE for the RLSL-JP/TLMS Algorithm
α=.95, μ =.0005, gain change at iteration 2,500

4.1.2 Results Using the Colored Noise Input

The next simulation uses Data Set 2, in which the input signal is a colored noise having eigenvalue ratio $\chi(R)$ of 4.872×10^4 since the order of the filter is

256. The result for the RLSL-JP algorithm is shown in Figure 4.1.5, and the result for the RLSL-JP/TLMS algorithm is shown in Figure 4.1.6. For this experiment, the step-size μ is set to .0002 because a value of μ greater than .0002 causes the LMS algorithm to diverge, and the forgetting factor of the RLSL-JP algorithm is set to 0.95. Both algorithms, the RLSL-JP/TLMS algorithm and the RLSL-JP algorithm, converge at about the same rate, at about iteration 400.

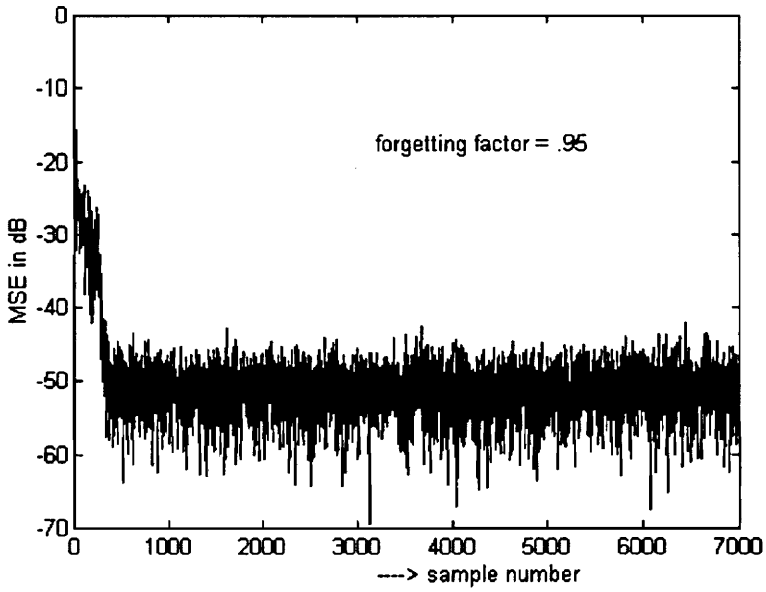

Figure 4.1.5 Five Trial Average MSE for the RLSL-JP Algorithm colored noise input

By comparing Figure 4.1.5 and Figure 4.1.6, we see that the final MSE's are almost the same. The MSE for the RLSL-JP algorithm is 9.833×10^{-6} or -50 dB, and the MSE for the RLSL-JP/TLMS algorithm is 7.1923×10^{-6} or -51.4 dB; the MSE is calculated by averaging the last 1,000 points of the ensemble average of the squared error. The smaller MSE for the RLSL-JP/TLMS is contributed by the TLMS algorithm as the "low MSE structure".

Moreover, the steady state cancellation of the RLSL-JP/TLMS algorithm is about 51.4 dB, which easily exceeds the CCITT Rec. G.165 requirements, and so does the steady state cancellation for the RLSL-JP algorithm.

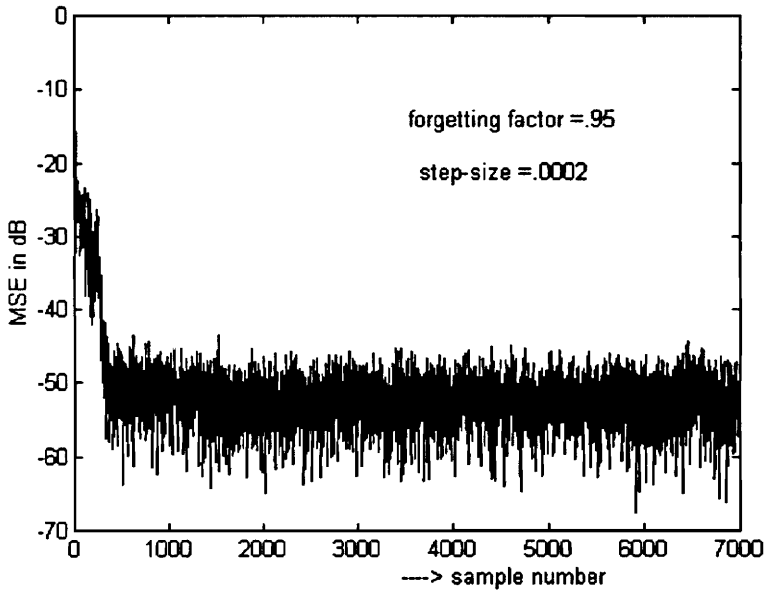

Figure 4.1.6 Five Trial Average MSE for the RLSL-JP/TLMS Algorithm colored noise input

Next we investigate the effect of a change in the system to be identified during operation. The system is run using Data Set 2 and after the system has converged, at iteration 4,000 the system to be identified changes by changing its parameters as explained in Section 3.3.2. This change causes large errors, so that the adaptive filter sees that there is a new system to be identified. Since the adaptive filter is already in the steady state, this condition forces the adaptive filter to converge from the arbitrary initial values which are not in the direction of the optimal values.

For the RLSL-JP algorithm, the result is shown in Figure 4.1.7 and for the RLSL-JP/TLMS algorithm the result is shown in Figure 4.1.8. By comparing Figure 4.1.7 and 4.1.8, we see that if we use the RLSL-JP/TLMS algorithm, the

peak error at the change is about 10 dB smaller than the peak error if we use the RLSL-JP algorithm,

In conclusion, the experiment with the colored noise input shows that the convergence rate of the RLSL-JP algorithm is almost not affected by the ill-conditioned input data. Whether we use the white Gaussian noise in Section 4.1.1 or the colored noise in Section 4.1.2 as the input, the RLSL-JP algorithm converges at about the same rate. Moreover, the RLSL-JP/TLMS algorithm easily exceeds the CCITT Rec. G.165 requirement regarding the convergence rate and the steady state cancellation, and so does the RLSL-JP algorithm.

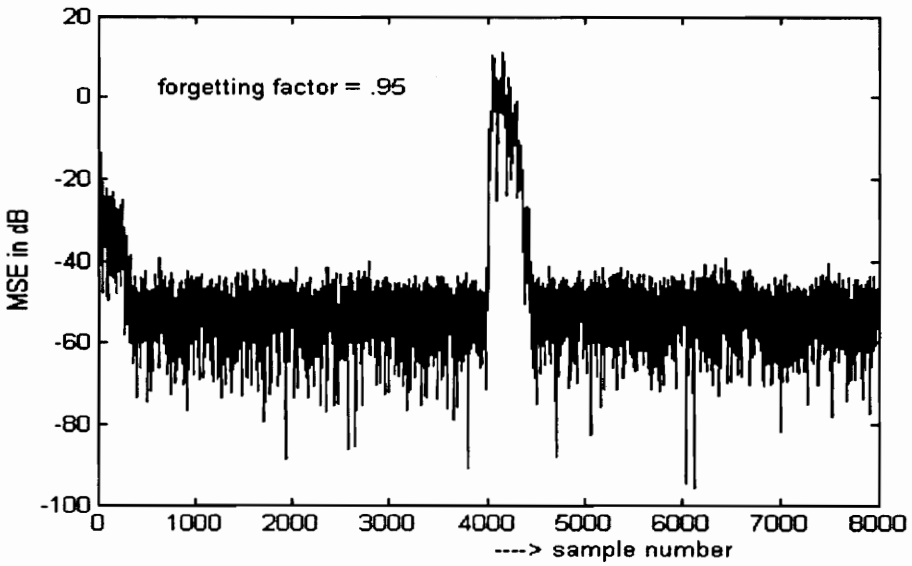

Figure 4.1.7 Two Trial Average MSE for the RLSL-JP Algorithm
colored noise input, gain change at iteration 4,000

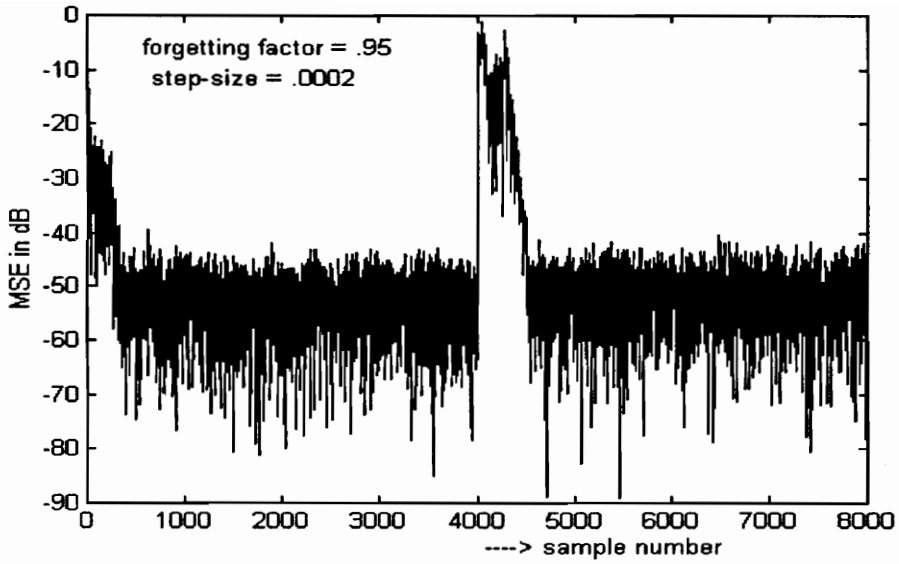

Figure 4.1.8 Two Trial Average MSE for the RLSL-JP/TLMS Algorithm colored noise input, gain change at iteration 4,000

4.1.3 Results Using the Computationally Reduced Order Technique

In this section, we apply the computationally reduced order technique for the RLSL-JP/TLMS algorithm. As described in Section 3.2, this technique is applied to the TLMS algorithm as the "low MSE structure", that is, after the RLSL-JP algorithm as the "fast convergence structure" has converged, only coefficients which are greater than the threshold are transferred to the TLMS structure. The threshold is chosen as 0.2% of the biggest value of the coefficients.

Since the "order" of the resulting TLMS structure is smaller, here we can use a step-size μ somewhat bigger than the one which is used in ordinary TLMS (TLMS without computationally reduced order). For the white Gaussian input, the step-size is 0.0007. The MSE curve is shown in Figure 4.1.9. The MSE, which is taken as the ensemble average of the last 2,000 samples of the squared error, is 1.789×10^{-6} or -57.5 dB. We see that the computationally reduced order

technique yields a somewhat higher MSE than the switching without computationally reduced order which is 1.2355×10^{-6} or -59 dB.

From Figure 4.1.9 we see that there is an immediate drop of the MSE curve. What happens is that, when we apply the computationally reduced order technique, we "force" some values of the impulse response to zero. This is almost the same as making the adaptive filter reach its steady state value, if we assume that any values below the threshold will actually be zero when they have converged. If there are many zeros in the impulse response, there are also many zeros in (3.2.1) and (3.2.2). Moreover, this will make (3.2.4) also contains many zeros and as a result, the excess noise, $J_{ex}(n)$, in (3.2.7) will drop. In the experiment, there are only 23 out of the 256 values of the impulse response which are bigger than the threshold. This immediate drop of the MSE is shown in Figure 4.1.9 at about iteration 1,000.

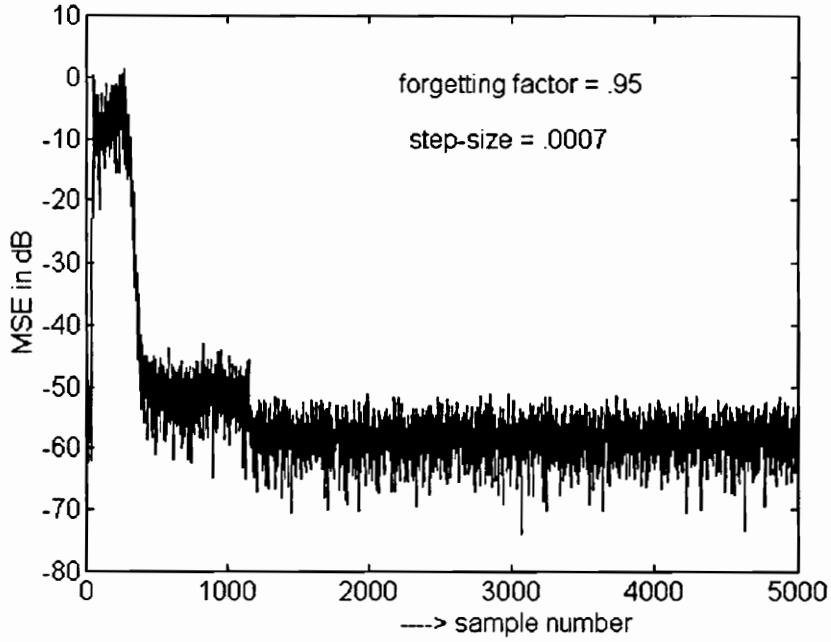

Figure 4.1.9 Five Trial Average MSE for the RLSL-JP/TLMS Algorithm with Computationally Reduced Order Technique

Furthermore, the immediate drop of the error can make the adaptation slow as indicated in (2.5.c) Section 2.2; a low error makes the TLMS algorithm, as the "low MSE structure", slow to reach its final value as indicated by the MSE which is somewhat bigger than the MSE for the TLMS algorithm without the computationally reduced order technique. Moreover, when we use the computationally reduced order technique, there is an error caused by mismatching, meaning there are some values which are actually non-zero which are forced to be zero, and this contributes to a higher MSE.

For the colored noise input with eigenvalue ratio $\chi(R)$ of 4.872×10^4, the step-size is 0.0003. This step-size is bigger than the step-size for the TLMS algorithm without the computationally reduced order technique of .0002, so that normally this will give a somewhat higher MSE. The MSE curve is shown in Figure 4.1.10. The MSE, which is taken as the ensemble average of the last 1,000 samples of the squared error, is 5.1815×10^{-6} or -52.8 dB and this is smaller than the MSE without the computationally reduced order technique as in Section 4.1.2. that is 7.1923×10^{-6} or -51.4 dB.

For the colored noise as an input, the computationally reduced order technique gives a smaller MSE. This is because for colored noise the TLMS algorithm, as the "low MSE structure", needs a long time to reach its steady state value, so that by applying the computationally reduced order technique, we "force" the TLMS algorithm to reach its steady state, meaning that any impulse response values below the threshold are considered to become zero as their steady state values, so that we replace them with zero. It is true that it can cause error. However, since if we use a small threshold the error is also small. The

result shows that for colored noise, the MSE is lower. So the experiment with a colored noise produces a better MSE.

For the colored noise input, unlike for the white Gaussian input, the drop in MSE is not very large as we see from Figure 4.1.10 where the MSE drop at about iteration 1,500 is barely visible. This is because the number of impulse response values greater than the threshold is 146 out of 256. From Figure 4.1.11 we can barely see the non-zero impulse response values at the beginning as well as at the tail of the impulse response. While the steady state value of these values is zero, we see that even when we apply the computationally reduced order technique, many of the impulse response values are still larger than the threshold. It simply takes longer in the colored noise case for the impulse response values to converge below the threshold.

In short, even if for the white Gaussian input the computationally reduced order technique gives a somewhat higher MSE, the computationally reduced order technique is still preferred because it can reduce the computational load, it gives a smaller MSE for the colored noise input, and we know that most of the signals in applications are colored. Since the computationally reduced order technique can not be applied for a non switching structure, we again have shown an advantage of the switching structure.

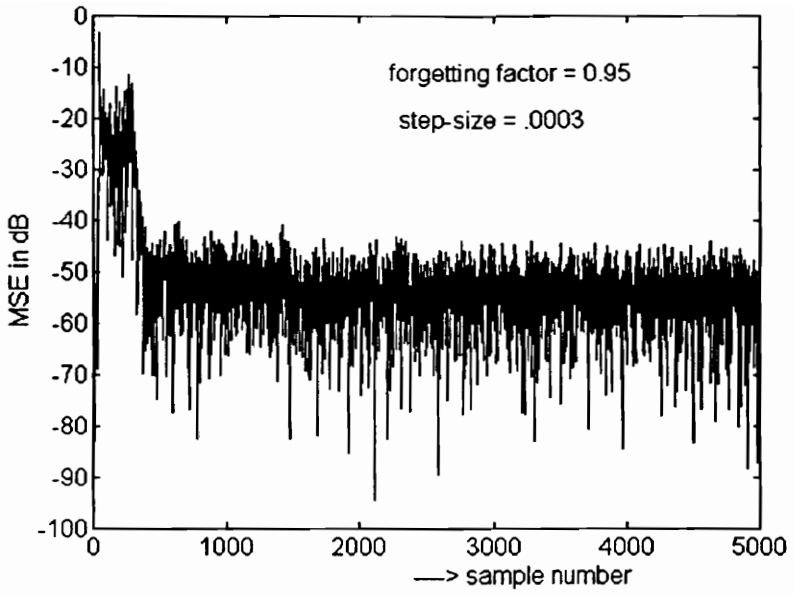

Figure 4.1.10 Two Trial Average MSE for the RLSL-JP/TLMS Algorithm with Computationally Reduced Order Technique, colored noise input

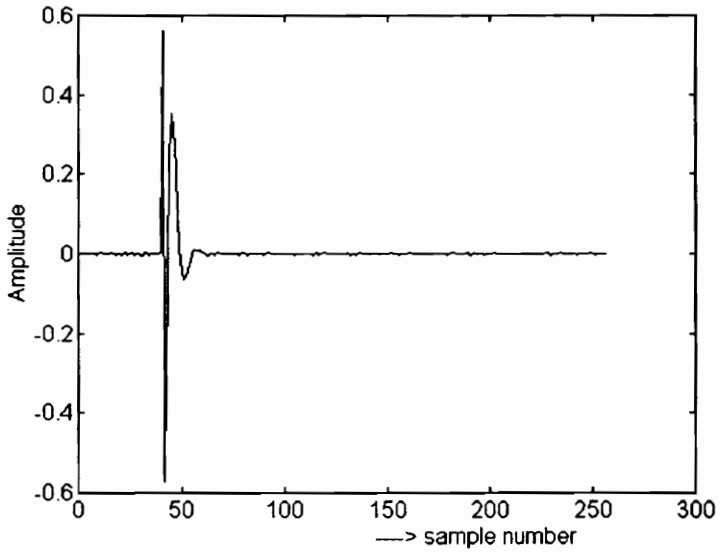

Figure 4.1.11 Impulse Response of the TLMS Algorithm with Computationally Reduced Order Technique, colored noise input

4.2 Switching Using the FTF/TLMS Algorithm

4.2.1 Results Using the White Gaussian Noise Input

For Data Set 1 as the input, the FTF algorithm, either as the non-switching structure or as the "fast convergence structure" in the switching structure, will be used with forgetting factor α equal to 0.999. This value will give fast convergence, without causing divergence. It turns out that for an order of 256, the FTF algorithm can converge with the forgetting factor α equal to .999 or larger, otherwise, the FTF algorithm will diverge. Meanwhile, the TLMS algorithm as the "low MSE structure" will be used with a small step-size μ of .0005 in order to give a low MSE. Moreover, the length of the interval, P, over which the error is considered flat, is 512. The result for the FTF/TLMS algorithm is shown in Figure 4.2.1. The same input data is applied to the FTF algorithm, and the result is shown in Figure 4.2.2.

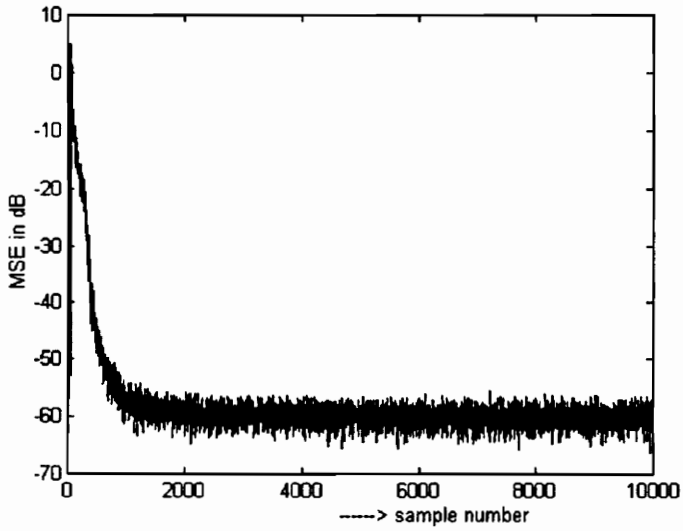

Figure 4.2.1 Ten Trial Average MSE for the FTF/TLMS Algorithm
α=.999, μ =.0005

Let us compare the performance of the FTF algorithm and the performance of the FTF/TLMS algorithm. By comparing Figure 4.2.1 for the FTF/TLMS algorithm and Figure 4.2.2 for the FTF algorithm, we can not see a clear difference. The MSE of the FTF/TLMS algorithm is 1.0238×10^{-6} or -59.8 dB and the MSE of the FTF algorithm is 1.011×10^{-6} or -59.9 dB; the MSE is calculated from the average of the last 2,000 samples of the squared error.

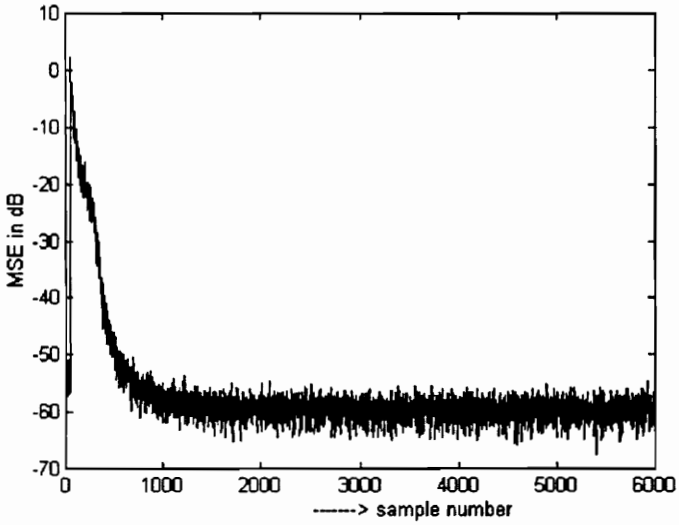

Figure 4.2.2 TenTrial Average MSE for the FTF Algorithm
α =.999

For the second experiment, we change the system to be identified at iteration 6,000. The transition period is made three times as long as the order of the filter and the transition weighting factor w_e is 0.994. After the change, the FTF/TLMS algorithm converges fast as indicated in Figure 4.2.3. From Figure 4.2.4, on the other hand, we see that after the change, the FTF algorithm converges very slowly. The slow convergence of the FTF algorithm after the change is because when the system to be identified changes, the FTF algorithm has already converged to the previously identified system, in other words, this

means that when the change occurs, the FTF algorithm is initialized with values which are not in the direction of the new system to be identified.

For the FTF/TLMS algorithm, when the change in the system to be identified occurs, it is still able to converge fast, because the FTF is reinitialized, although it is not as fast as at the beginning of the iterations. The reason is that, the change in the system to be identified is not instantaneous, it produces a transient condition, before settling at the final value as explained in Section 3.3.2. Therefore, when the change occurs, the FTF algorithm will try to follow this transient behaviour. Although the transient period is very short, it still produces some initial values to the FTF algorithm.

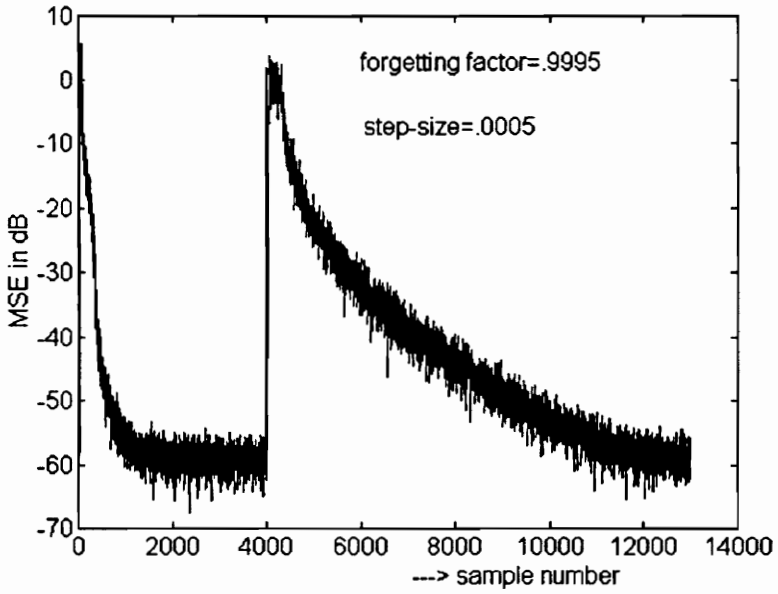

Figure 4.2.3 Ten Trial Average MSE for the FTF/TLMS Algorithm gain change at iteration 4,000

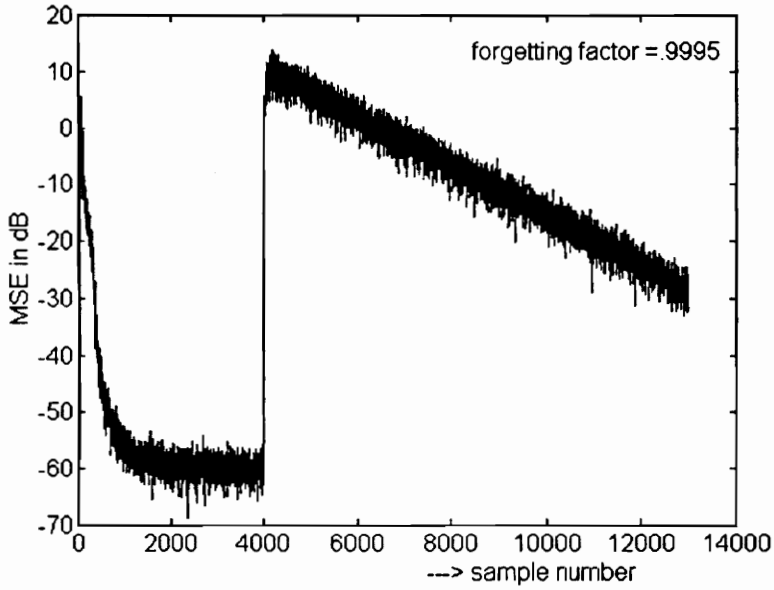

Figure 4.2.4 Ten Trial Average MSE for the FTF Algorithm gain change at iteration 4,000

4.2.2 Results Using the Colored Noise Input

Data Set 2, which is a colored noise with eigenvalue ratio $\chi(R)$ of 1.387×10^4, since the filter order is 64, is used for this simulation. We use a filter order of 64 since the computation for order 256 takes a long time. Also, most of the impulse response of the system to be identified, i.e. the echo path, consists of zeros, so that reducing the order does not produce a different result. By reducing the order here, we reduce the number of zeros in the impulse response of the system to be identified, and also reduce the order of the adaptive filter.

The FTF algorithm uses a forgetting factor of .9999 since the algorithm diverges for a forgetting factor less than that. For the FTF/TLMS algorithm, on the other hand, the forgetting factor is set at .999 and the step-size of the LMS algorithm is set at 0.0003. This is possible since for the switching structure, the system switches to the LMS algorithm after the FTF algorithm reaches convergence, so that the system switches to the LMS algorithm before the FTF

algorithm diverges. This scheme gives the FTF/TLMS algorithm faster convergence as shown in Figure 4.2.5 than the FTF algorithm as shown in Figure 4.2.6.

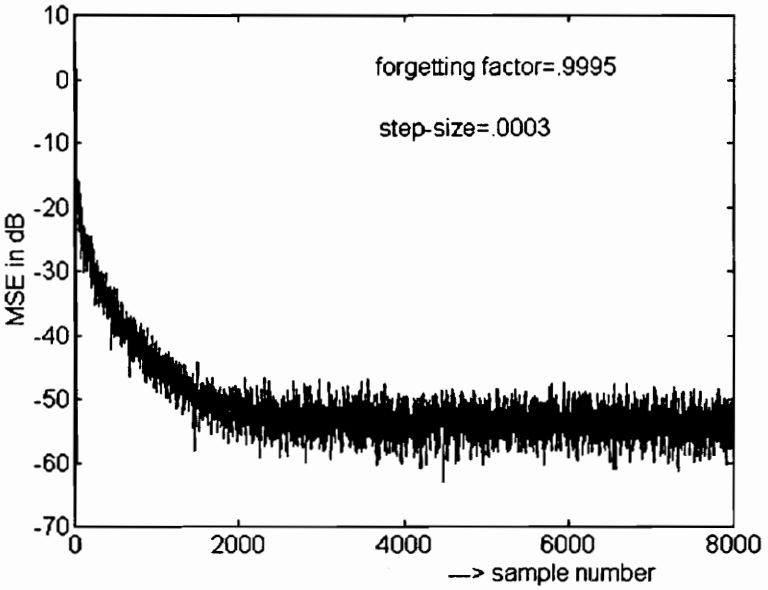

Figure 4.2.5 Ten Trial Average MSE for the FTF/TLMS Algorithm colored noise input

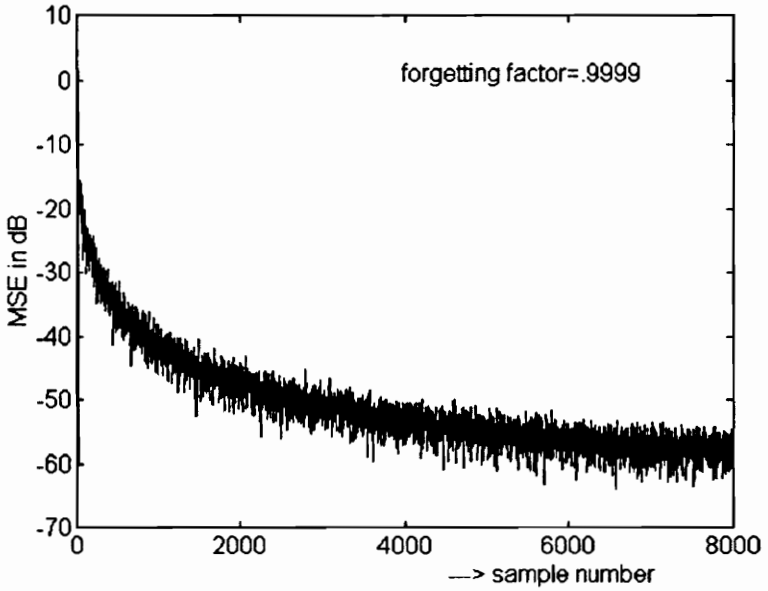

Figure 4.2.6 Ten Trial Average MSE for the FTF Algorithm colored noise input

We see from Figure 4.2.5, that the FTF/TLMS algorithm gives faster convergence than the FTF algorithm as shown in Figure 4.2.6; the FTF/TLMS algorithm reaches an MSE of about -53 db in about 1,500 iterations or 187 ms, while the FTF algorithm reaches an MSE of about -53 dB in about 3,000 iterations or 375 ms assuming the sampling time is 125 micro seconds. However, the FTF algorithm gives a lower MSE of 1.095×10^{-6} or -59 dB at iteration 6,000 than the MSE for the FTF/TLMS algorithm of 4.14×10^{-6} or -53 dB. Actually, the final MSE value of the LMS algorithm with the step-size .0003 is 1.009×10^{-6} or -59.96 dB and of the FTF algorithm is 1.0032×10^{-6} or -59.97 dB.

In fact, we can develop the FTF/FTF algorithm where the first FTF algorithm is used with a forgetting factor of .9995 to get fast convergence, and the second FTF algorithm is used with a forgetting factor of .9999 to get low MSE. The switching between the FTF algorithms can be done by using a *soft-constraint* as explained in Section 2.3.2.1. However, here we will most likely have problems of numerical instability. Furthermore, the MSE of the FTF algorithm with a forgetting factor of .9999 is not much different from the MSE for the TLMS algorithm with a forgetting factor of .0003, whereas the TLMS algorithm does not have numerical instability problems. In conclusion, the FTF/FTF algorithm is not too promising.

The FTF/TLMS algorithm gives better performance, in the sense of convergence, if there is a change in the system to be identified, as can be seen from comparing Figures 4.2.7 and 4.2.8 for the FTF/TLMS algorithm and the FTF algorithm respectively. The reason is that when there is a change, the FTF algorithm has to converge from an incorrect initial condition, while, on the other hand, the FTF/TLMS algorithm does not experience much of an initial condition

problem since it can reset the initial condition to zero, and start converging. However, the convergence after the change for both structures is slower than the convergence if the algorithms start from the zero initial condition. This is because the change in the system does not happen instantaneously, as explained in Section 3.3.2, there is some transient state which is short yet produces an incorrect initial condition.

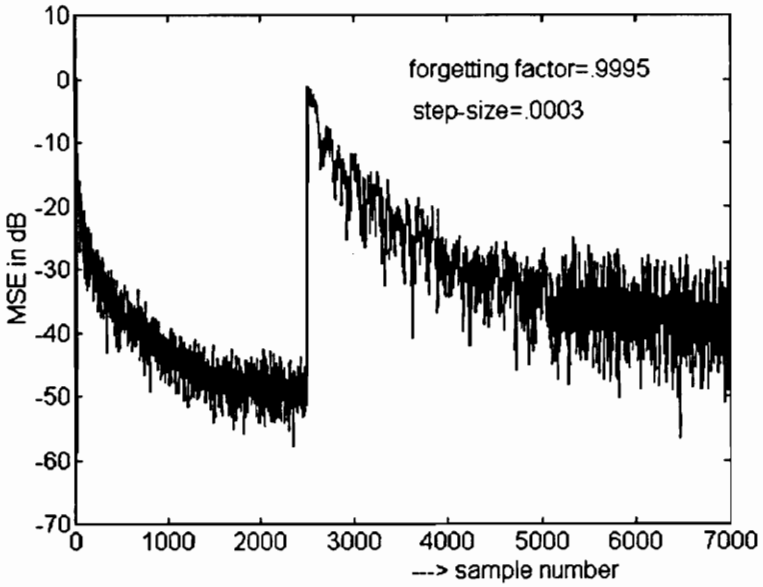

Figure 4.2.7 Fifteen Trial Average MSE for the FTF/TLMS Algorithm colored noise input, gain change at iteration 2,500

In conclusion, the switching structure of the FTF/TLMS algorithm is more promising than the FTF algorithm. This comes from considering that in many applications we need fast convergence and that the system changes from time to time which are situations that can not be handled properly by the FTF algorithm. Furthermore, the FTF/TLMS algorithm as well as the FTF algorithm exceed the CCITT Rec. G.165 requirements regarding convergence rate and steady state cancellation.

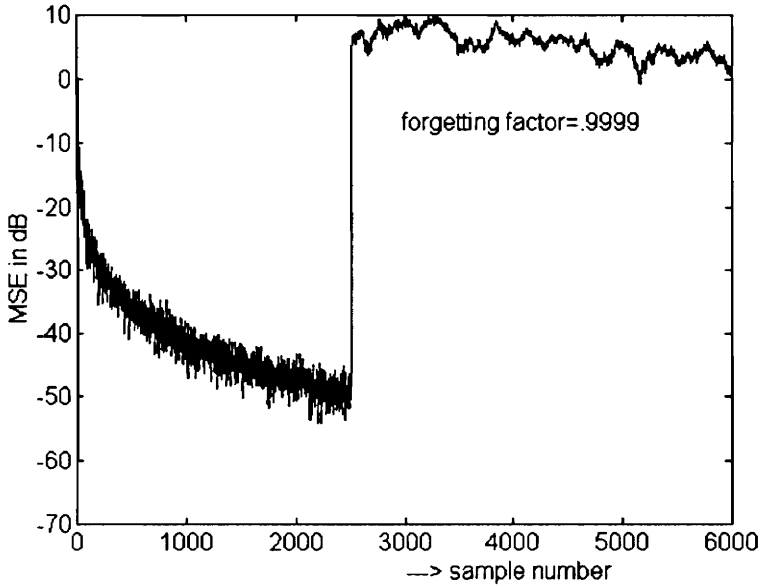

Figure 4.2.8 Fifteen Trial Average MSE for the FTF Algorithm colored noise input, gain change at iteration 2,500

4.3 Switching Using the GAL-JP/TLMS Algorithm

4.3.1 Results Using the White Gaussian Noise Input

For this case, Data Set 1 is used as the input. The GAL-JP algorithm is used in the GAL-JP/TLMS as the "fast convergence structure" and it is used with a *forgetting factor* α equal to .9999 in order to get fast convergence, while the TLMS algorithm is used as the "low MSE structure" with a small step-size μ of .0003 in order to give low MSE. For the GAL-JP algorithm the forgetting factor, α, is .99995 since for any α less than .99995, the GAL-JP algorithm diverges after several thousand iterations; note that this will not occur in the GAL-JP/TLMS algorithm because the GAL-JP algorithm will be replaced after it has converged by the TLMS algorithm. The order of the filter is 64.

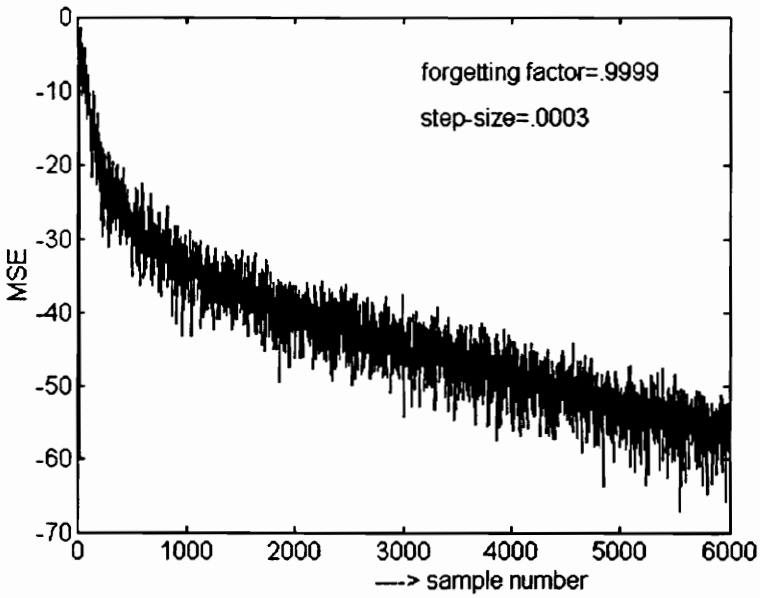

Figure 4.3.1 Eight Trial Average MSE for the GAL-JP/TLMS Algorithm

It turns out that the GAL-JP/TLMS algorithm converges faster than the GAL-JP algorithm as shown in Figures 4.3.1 and 4.3.2 respectively, because the forgetting factor for the GAL-JP/TLMS algorithm is less than the forgetting factor for the GAL-JP algorithm. Moreover, the MSE for the GAL-JP/TLMS algorithm of 6.88×10^{-6} or -51.6 dB is smaller than the MSE for the GAL-JP algorithm of 2.05×10^{-5} or -46.9 dB.

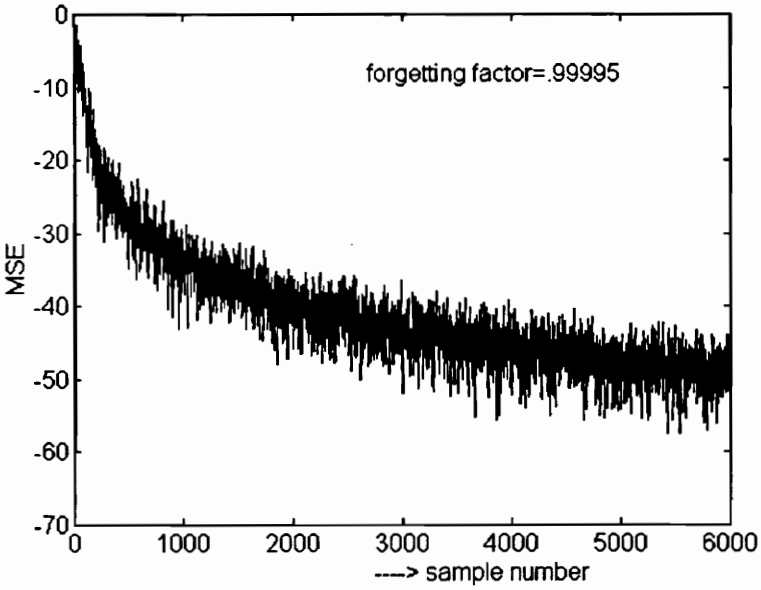

Figure 4.3.2 Eight Trial Average MSE for the GAL-JP Algorithm

In fact, with a forgetting factor of .99995, the GAL-JP algorithm will eventually have a smaller MSE than the TLMS algorithm with step-size .0003 which is used as the "low MSE structure". However, the GAL-JP algorithm with a forgetting factor very close to one will converge very slowly, and also it will adapt very slowly if there is any change in the system to be identified as will be shown in the next simulation.

Now, we change the unknown system at iteration 4,000. From Figure 4.3.3 and Figure 4.3.4, we see that the GAL-JP/TLMS algorithm in Figure 4.3.3

converges faster than the GAL-JP algorithm in Figure 4.3.4. This is, again, because of the effect of the initial conditions as mentioned in the previous simulations and the value of the forgetting factor.

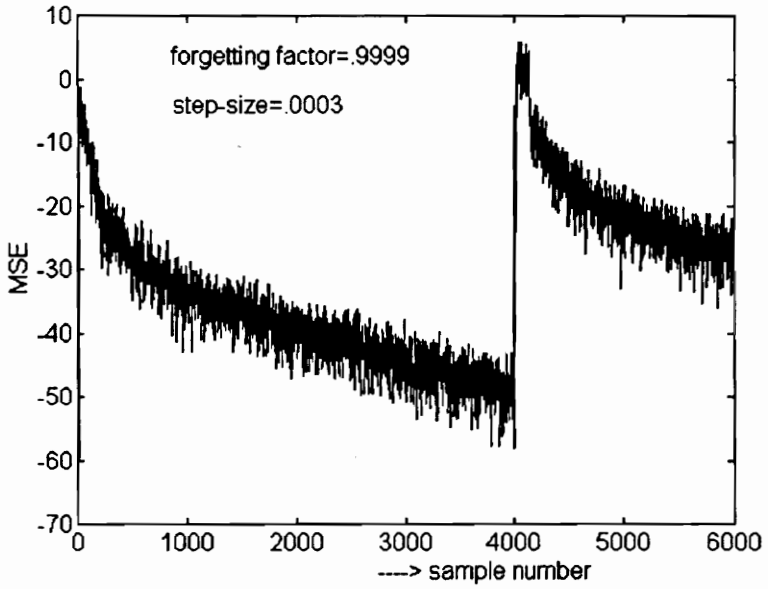

Figure 4.3.3 Eight Trial Average MSE for the GAL-JP/TLMS Algorithm gain change at iteration 4,000

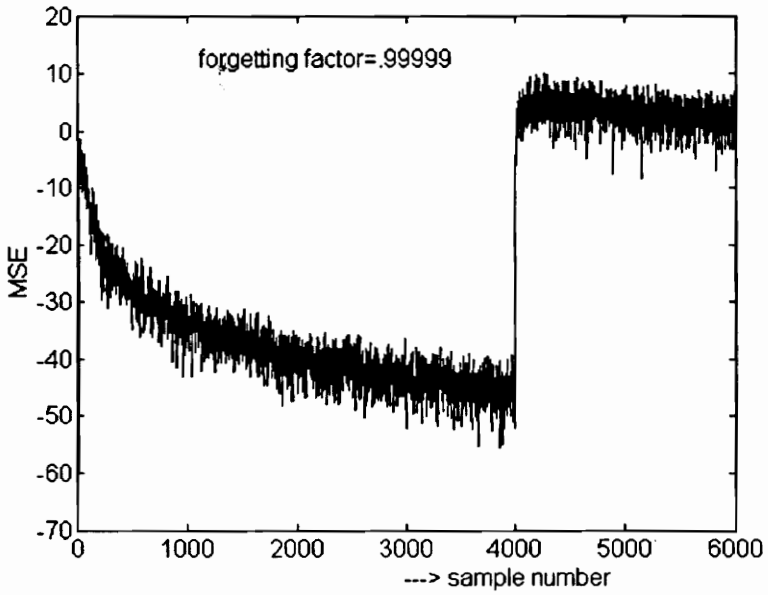

Figure 4.3.4 Eight Trial Average MSE for the GAL-JP Algorithm gain change at iteration 4,000

4.3.2 Results Using the Colored Noise Input

For this case, Data Set 2 is used as the input. The GAL-JP algorithm in the GAL-JP/TLMS algorithm, which is used as the "fast convergence structure", is used with a *forgetting factor* α equal to .9999 in order to get fast convergence, while the TLMS algorithm as the "low MSE structure" is used with a small step-size μ of .0003 in order to give low MSE. For the GAL-JP algorithm, the forgetting factor is 0.99999. The MSE for the GAL-JP/TLMS algorithm and the GAL-JP algorithm are shown in Figures 4.3.5 and 4.3.6 respectively.

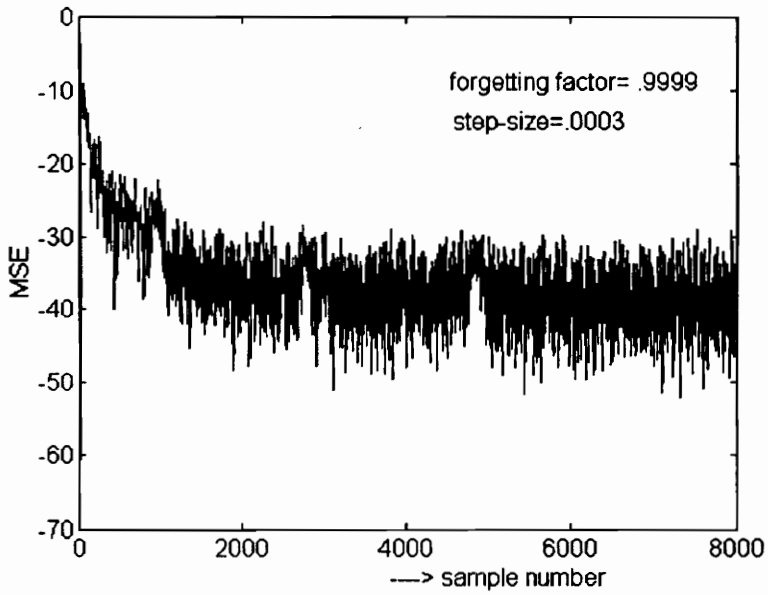

Figure 4.3.5 Eight Trial Average MSE for the GAL-JP/TLMS Algorithm colored noise input

The MSE of the GAL-JP/TLMS algorithm is 1.675×10^{-4} or -37.8 dB, whereas the MSE of the GAL-JP algorithm is 7.675×10^{-5} or -41.1 dB and this result is expected since with a forgetting factor very close to one, the GAL-JP algorithm will have very small MSE. However, even though the MSE of the GAL-JP algorithm is somewhat smaller than the MSE for the GAL-JP/TLMS algorithm,

the MSE curve of the switching structure is smoother, without spikes, as we see from Figure 4.3.6 that the spikes of the GAL-JP algorithm can be more than 10 dB. This behavior is undesirable since it can cause the algorithm to become unstable. In fact, if we let the iterations of the GAL-JP algorithm continue, it diverges. Also with a forgetting factor very close to one, the GAL-JP algorithm will not be able to adapt very easily if there is a change in the system to be identified as will be shown in the next simulation.

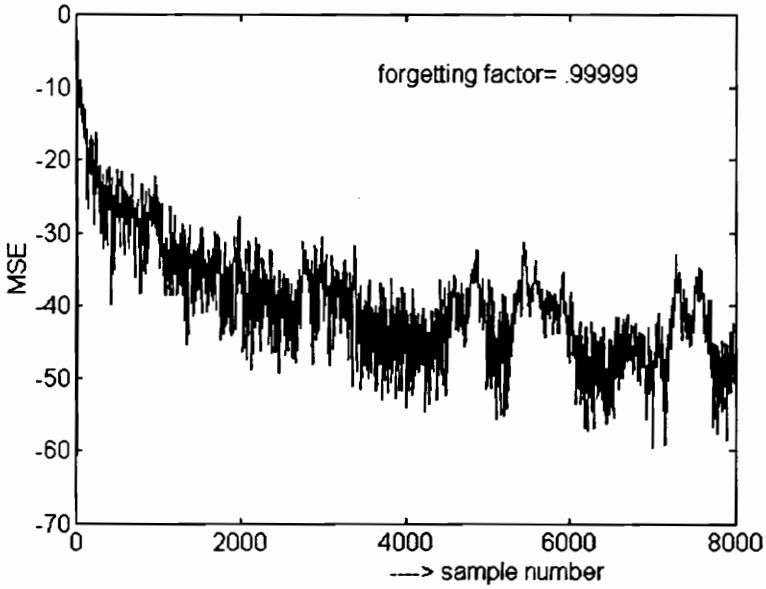

Figure 4.3.6 Eight Trial Average MSE for the GAL-JP Algorithm colored noise input

Next, we look at the performances of the GAL-JP/TLMS algorithm and the GAL-JP algorithm when the system to be identified changes, in this case it is changed at iteration 6,500. The MSE for the GAL-JP/TLMS algorithm converges fast after the change as shown in Figure 4.3.7, while the MSE for the GAL-JP algorithm does not seem to start converging yet at iteration 9,500 as shown in Figure 4.3.8. As in the previous cases, the slow convergence of the GAL-JP

algorithm is caused by incorrect initial conditions, and also because the forgetting factor is very close to one.

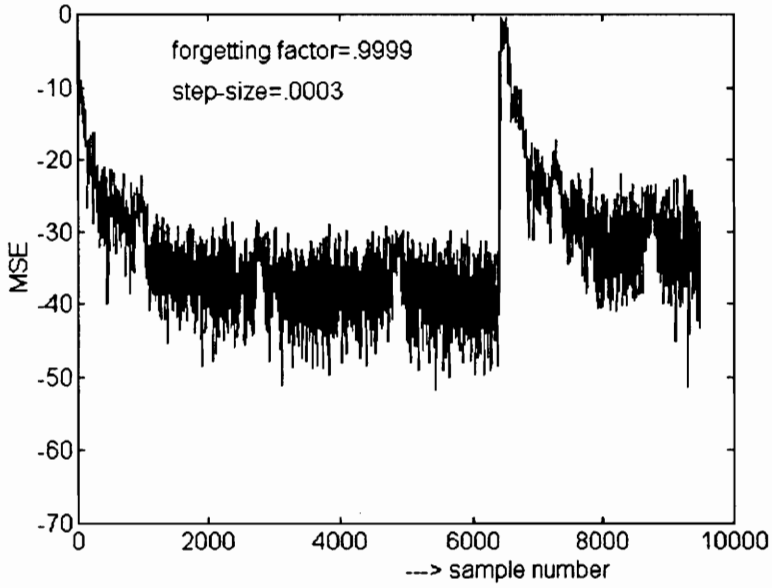

Figure 4.3.7 Eight Trial Average MSE for the GAL-JP/TLMS Algorithm colored noise input, gain change at iteration 6,500

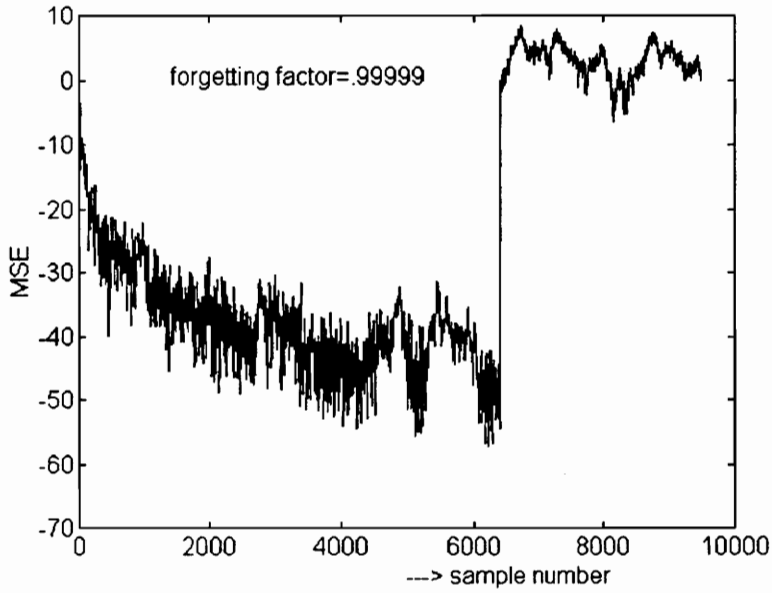

Figure 4.3.8 Eight Trial Average MSE for the GAL-JP Algorithm colored noise input, gain change at iteration 6,500

In conclusion, the GAL-JP/TLMS algorithm, again, shows better performance than the GAL-JP algorithm. Moreover, as for the previous cases, the GAL-JP/TLMS algorithm easily satisfies the CCITT Rec. G.165 requirements regarding the convergence rate and the steady state cancellation, and so does the GAL-JP algorithm.

CHAPTER FIVE

CONCLUSION AND FURTHER RESEARCH

5.1 Conclusion

Simulations for switching adaptive filter structures have been presented. The algorithms that are used for the switching structure are the *recursive-least-squares-lattice joint process* (RLSL-JP) algorithm together with the transversal LMS algorithm, abbreviated as the RLSL-JP/TLMS algorithm, the fast transversal filter (FTF) algorithm together with the transversal LMS algorithm, abbreviated as the FTF/TLMS algorithm, and the *gradient-adaptive-lattice joint process*(GAL-JP) algorithm together with the transversal LMS algorithm, abbreviated as the GAL-JP/TLMS. The simulations are conducted using white Gaussian noise and colored-noise having eigenvalue ratio of 1.387×10^4 if the filter order is 64, and eigenvalue ratio of 4.872×10^4 if the filter order is 256. The results of the switching structure are compared with the results of the non-switching structure, which are the RLSL-JP algorithm, the FTF algorithm, and the GAL-JP algorithm.

The transversal LMS algorithm is chosen as the "low MSE structure" since not only is the LMS algorithm numerically more stable than the other algorithms mentioned above, the LMS algorithm can also have a smaller MSE and a lower computational load. The other algorithms are used as the "fast convergence structure" since they are actually faster in convergence than the LMS algorithm. We do not use the LMS algorithm as the "fast convergence structure" since the LMS algorithm is very slow if the input is colored noise.

The switching between the "fast convergence structure" and the "low MSE structure", and vice versa, is determined by the error. If the average of the squared error is small and almost the same for several iterations, then it is declared that the "fast convergence structure" has converged and the switching structure will switch to use the "low MSE structure". On the other hand, when the average of the squared error is large (exceeds a pre-determined value), the switching structure switches to the "fast convergence structure" and starts converging if this is caused by a change in the unknown system.

In general, the switching structure shows more preferable performance: it can converge faster, it produces a smaller MSE, it can adapt to changes in the unknown system without any problems, and it is more stable.

By ignoring the values of the impulse response which are less than some threshold, we can reduce the computational load of the adaptive filter. In the switching adaptive filter structure, the computationally-reduced order technique is applied to the "low MSE structure". The computationally-reduced order technique seems to yield good results in the sense that the MSE drops even lower than for the switching structure without the computationally-reduced order technique. The possibility to implement the computationally-reduced order technique is another advantage of the switching adaptive filter structure; we can not apply the computationally reduced order technique for the non-switching structure. However, in applications where there are few impulse response values equal to zero, the influence of the computationally reduced order is hardly noticeable. Actually, the computationally-reduced order technique can potentially produce a large error when the threshold is so big that many significant impulse response values are discarded.

Some algorithms like the FTF algorithm and the GAL-JP algorithm turned out to need a forgetting factor very close to one, otherwise they diverged after several iterations. The forgetting factor close to one could yield a small MSE for the price of very slow convergence. Even worse, these algorithms are not able to easily adapt to a change in the system as the experiments show that after the unknown system changed, the FTF algorithm and the GAL-JP algorithm hardly converged again.

By using the switching structure, we can make some compromises. We let the FTF algorithm, and also the GAL-JP algorithm, have a forgetting factor not too close to one, so that we can have faster convergence. Then we switch to the LMS algorithm. This way we can make the FTF algorithm, and also the GAL-JP algorithm, converge faster, yet we prevent the switching structure from diverging, and the final MSE is that associated with the LMS algorithm. Also, if there is any change in the unknown system, the switching structure can converge readily to the new system.

5.2 Further Research

Switching using the average of the squared error as a criterion is simple, however, this is not optimum. The optimum criterion for switching needs investigation. The switching optimization here means that at a certain time we compute the criterion for each algorithm using the parameters for each algorithm at that time. The algorithm that fulfills the criterion better will be used by the switching structure. The variables that are taken into account are the state of an algorithm at that time and the rate of convergence. The optimization function uses these two variables to determine which algorithm should be used from that time on (until the criterion dictates another switch).

The simulation experiments in this thesis are based on the assumption of stationarity for the system to be identified. The next thing to do is to find out the behavior of the switching structure in a non-stationary environment and to make some modifications so that it can perform well. This is especially interesting since several researchers show that even though the least-squares based algorithms demonstrate fast convergence, they are outperformed by the LMS algorithm in a non-stationary environment; in other words, the LMS algorithm tracks better than the least-squares based algorithms. By finding a mathematical explanation for this phenomenon, we can hopefully devise a switching adaptive structure that can converge fast and track fast as well.

APPENDIX A

The Gain-Normalized Fast Transversal Filter Algorithm

The fast-transversal-filter (FTF) algorithm consists of two sections: a *Prediction* section and a *Filtering* section, which can be summarized as follows [16];

Prediction

$$\eta_M(n) = \underline{a}_M^H(n-1)\underline{u}_{M+1}(n) \tag{A.1}$$

$$f_M(n) = \gamma_M(n-1)\eta_M(n) \tag{A.2}$$

$$\mathfrak{I}_M(n) = \lambda\mathfrak{I}_M(n-1) + \eta_M(n-1)f^*_{M+1}(n) \tag{A.3}$$

$$\gamma_{M+1}(n) = \lambda\frac{\mathfrak{I}_M(n-1)}{\mathfrak{I}_M(n)}\gamma_M(n-1) \tag{A.4}$$

$$\underline{\tilde{k}}_{M+1}(n) = \begin{bmatrix} 0 \\ \underline{\tilde{k}}_M(n-1) \end{bmatrix} + \lambda^{-1}\frac{\eta_M(n)}{\mathfrak{I}_M(n-1)}a_M(n-1) \tag{A.5}$$

$$\underline{a}_M(n) = a_M(n-1) - f^*_M(n)\begin{bmatrix} 0 \\ \underline{\tilde{k}}_M(n-1) \end{bmatrix} \tag{A.6}$$

$$\psi_M(n) = \lambda B_M(n-1)\tilde{k}_{M+1,M+1}(n) \tag{A.7}$$

$$\gamma_M(n) = \left[1 - \psi^*_M(n)\gamma_{M+1}(n)\tilde{k}_{M+1,M+1}(n)\right]^{-1}\gamma_{M+1}(n) \tag{A.8}$$

$$\text{Rescue variable} = \left[1 - \psi^*_M(n)\gamma_{M+1}(n)\tilde{k}_{M+1,M+1}(n)\right] \tag{A.9}$$

$$b_M(n) = \gamma_M(n)\psi_M(n) \tag{A.10}$$

$$B_M(n) = \lambda B_M(n-1) + \psi_M(n-1)b^*_{M+1}(n) \tag{A.11}$$

$$\begin{bmatrix} \underline{\tilde{k}}_M(n) \\ 0 \end{bmatrix} = \underline{\tilde{k}}_{M+1}(n) - \tilde{k}_{M+1,M+1}(n)\underline{c}_M(n-1) \tag{A.12}$$

$$\underline{c}_M(n) = \underline{c}_M(n-1) - b^*_M(n)\begin{bmatrix}\tilde{\underline{k}}(n)\\ 0\end{bmatrix} \quad \text{(A.13)}$$

Filtering

$$\alpha_M(n) = d(n) - \hat{\underline{w}}^H_M(n-1)\underline{u}_M(n) \quad \text{(A.14)}$$

$$e(n) = \gamma_M(n-1)\alpha_M(n) \quad \text{(A.15)}$$

$$\hat{\underline{w}}_M(n) = \hat{\underline{w}}_M(n-1) + \tilde{\underline{k}}^H_M(n)e^*_M(n) \quad \text{(A.16)}$$

APPENDIX B

MSE and Convergence Analysis for the GAL-JP Algorithm

The *gradient adaptive lattice joint process* (GAL-JP) algorithm can be summarized as follows [33], it consists of a *prediction* section and a *filtering* section:

Prediction

$$\mu_{m+1}(n+1) = \alpha\ \mu_{m+1}(n) + \left|f_m(n+1)\right|^2 + \left|b_m(n)\right|^2 \qquad \text{(B.1.a)}$$

$$\Gamma_{m+1}(n+1) = \Gamma_{m+1}(n) - \frac{f_m(n)\, b^*_{m+1}(n) + b_m(n-1)\, f^*_{m+1}(n)}{\mu_{m+1}(n+1)} \qquad \text{(B.1.b)}$$

$$f_m(n+1) = f_{m-1}(n+1) + \Gamma^*_m(n)\, b_{m-1}(n) \qquad \text{(B.1.c)}$$

$$b_m(n+1) = b_{m-1}(n) + \Gamma^*_m(n)\, f_{m-1}(n+1) \qquad \text{(B.1.d)}$$

Filtering

$$e_{m+1}(n+1) = e_m(n+1) - c^*_m(n+1)\, b_m(n+1) \qquad \text{(B.1.e)}$$

$$\hat{d}(n+1) = \underline{c}^H_m(n+1)\, \underline{b}_m(n+1) \qquad \text{(B.1.f)}$$

$$c_m(n+1) = c_m(n) + \frac{2 b_m(n) e^*_{m+1}(n)}{\mu_{m+1}(n)} \qquad \text{(B.1.g)}$$

where:

$\mu(n)$	: sum of squares of forward and backward prediction errors
α:	:forgetting factor, $\alpha<1$, $\alpha\approx1$
$f_{m+1}(n+1)$	: forward prediction error
$b_{m+1}(n+1)$	: backward prediction error
$\Gamma_{m+1}(n+1)$	: reflection coefficient

$e_{m+1}(n+1)$: the estimation error

$\hat{d}(n+1)$: the estimate of the desired signal

$c_m(n+1)$: regression coefficient

m : 0 , 1 , ... , M,

M : the order of the filter

From Figure 2.4.1.b, we see that

$$e_0(n+1) = d(n+1) \tag{B.2}$$

where $d(n+1)$ is the desired signal.

Define the regression-coefficient error at stage m, $\varepsilon_m(n+1)$, as:

$$\varepsilon_m(n+1) = c_m(n+1) - c_{m\,opt} \tag{B.3}$$

where $c_{m\,opt}$ is the optimum value of $c_m(n+1)$.

Substituting (B.3) into (B.1.e) yields the estimation error at stage m, $e_{m+1}(n+1)$, as:

$$e_{m+1}(n+1) = e_m(n+1) - \left(\varepsilon_m(n+1) + c_{m\,opt}\right)^* b_m(n+1)$$

$$= e_{m+1\,opt}(n+1) - \varepsilon_m^*(n+1)\, b_m(n+1) \tag{B.4}$$

where

$$e_{m+1\,opt}(n+1) = e_m(n+1) - c_{m\,opt}^*\, b_m(n+1)$$

is the optimum estimation error at stage m. Furthermore, by using (B.4) and (B.1.g), (B.3) can be written in recursive form as:

$$\varepsilon_m(n+1) = c_m(n) + \frac{2b_m(n)\left(e_{m+1\,opt}(n) - \varepsilon_m^*(n)\, b_m(n)\right)^*}{\mu_{m+1}(n)} - c_{m\,opt}$$

$$= \left[1 - \frac{2\, b_m(n)\, b_m^*(n)}{\mu_{m+1}(n)} \right] \varepsilon_m(n) + \frac{2 b_m(n)\, e^*_{m+1\,opt}(n)}{\mu_{m+1}(n)} \qquad \text{(B.5)}$$

Defining the variance of the regression-coefficient-error at stage m, $K_m(n+1)$, as:

$$K_m(n+1) = E\left[\varepsilon_m(n+1)\, \varepsilon_m^*(n+1)\right] \qquad \text{(B.6)}$$

and using (B.5) in (B.6), $K_m(n+1)$ can be written as :

$$K_m(n+1) = E\left[\left(\left\{ 1 - \frac{2\, b_m(n)\, b_m^*(n)}{\mu_{m+1}(n)} \right\} \varepsilon_m(n) + \frac{2 b_m(n)\, e^*_{m+1\,opt}(n)}{\mu_{m+1}(n)} \right) \right.$$

$$\left. \left(\left\{ 1 - \frac{2\, b_m(n)\, b_m^*(n)}{\mu_{m+1}(n)} \right\} \varepsilon_m(n) + \frac{2 b_m(n)\, e^*_{m+1\,opt}(n)}{\mu_{m+1}(n)} \right)^* \right] \qquad \text{(B.7)}$$

By multiplying and rearranging (B.7), and assuming $E\left[\frac{x}{y}\right] \approx \frac{E[x]}{E[y]}$; assuming x and y are statistically independent, and using Taylor series expansion to expand 1/y at its expected value, and also assuming that the variance of y is small, we can approximate E[1/y] with 1/E[y], so that we get:

$$K_m(n+1) = K_m(n) - \frac{E\left[2\, b_m(n)\, b_m^*(n)\, \varepsilon_m(n)\, \varepsilon_m^*(n)\right]}{E\left[\mu_{m+1}(n)\right]}$$

$$- \frac{E\left[2 \varepsilon_m(n)\, \varepsilon_m^*(n)\, b_m(n)\, b_m^*(n)\right]}{E\left[\mu_{m+1}(n)\right]}$$

$$+\frac{E\left[4\, b_m(n)\, b_m^*(n)\, \varepsilon_m(n)\, \varepsilon_m^*(n)\, b_m(n)\, b_m^*(n)\right]}{E\left[\mu_{m+1}(n)\, \mu_{m+1}^*(n)\right]}$$

$$-\frac{E\left[4\, b_m(n)\, b_m^*(n)\, \varepsilon_m(n)\, b_m(n)\, e_{m+1\,opt}^*(n)\right]}{E\left[\mu_{m+1}(n)\, \mu_{m+1}^*(n)\right]}$$

$$-\frac{E\left[4\, e_{m+1\,opt}(n)\, b_m^*(n)\, \varepsilon_m^*(n) b_m(n)\, b_m^*(n)\right]}{E\left[\mu_{m+1}(n)\, \mu_{m+1}^*(n)\right]}$$

$$+\frac{E\left[4\, e_{m+1\,opt}(n)\, b_m^*(n) b_m(n)\, e_{m+1\,opt}^*(n)\right]}{E\left[\mu_{m+1}(n)\, \mu_{m+1}^*(n)\right]} \qquad \text{(B.8)}$$

From random process theory, we know that for a real zero mean Gaussian random variable u we have:

$$E[uuuu] = 3\left\{E\left[u^2\right]\right\}^2 \qquad \text{(B.9)}$$

which is called *the Gaussian fourth moment factoring theorem.* We also know that for two Gaussian random variables u_1 and u_2 we have:

$$E\left[u_1^2 u_2^2\right] = E\left[u_1^2\right]E\left[u_2^2\right] + 2\left(E\left[u_1 u_2\right]\right)^2 \qquad \text{(B.10)}$$

and if u_1 and u_2 are independent and have zero mean, we have

$$E\left[u_1^2 u_2^2\right] = E\left[u_1^2\right]E\left[u_2^2\right] \qquad \text{(B.11)}$$

Equations (B.9) through (B.11) will be used for the subsequent derivations.

Assume that $b_m(n)$, $\varepsilon_m(n)$, and $e_{m+1\,opt}$ are independent and $e_{m+1\,opt}$ has zero mean. Then by applying (B.9) to $E\left[b_m(n)\, b_m^*(n) b_m(n)\, b_m^*(n)\right]$ in (B.8), which

is obtained by separating $b_m(n)$ and $\varepsilon_m(n)$ in the fourth term of (B.8), we can write $K_m(n+1)$ as

$$K_m(n+1) = K_m(n)B_{m+1}(n+1) + P_{m+1}(n+1) \tag{B.12}$$

where:

$$B_{m+1}(n+1) = 1 - \frac{4E\left[b_m^2(n)\right]}{E\left[\mu_{m+1}(n)\right]} + \frac{12\left(E\left[b_m^2(n)\right]\right)^2}{E\left[\mu_{m+1}(n)\mu_{m+1}(n)^*\right]} \tag{B.13}$$

$$P_{m+1}(n+1) = \frac{4E\left[e^*_{m+1\,opt}(n)e_{m+1\,opt}(n)\right]E\left[b_m^2(n)\right]}{E\left[\mu_{m+1}(n)\mu_{m+1}(n)^*\right]} \tag{B.14}$$

By iterating (B.1.a), we obtain:

$$\mu_{m+1}(n) = \alpha^{n-1}\mu_{m+1}(0) + \sum_{i=0}^{n-1}\alpha^i\left|b_m(n-1-i)\right|^2 + \sum_{i=0}^{n-1}\alpha^i\left|f_m(n-i)\right|^2 \tag{B.15}$$

so that the mean of $\mu_{m+1}(n)$ is

$$E\left[\mu_{m+1}(n)\right] = \alpha^{n-1}\mu_{m+1}(0) + \sum_{i=0}^{n-1}\alpha^i E\left[\left|b_m(n-1-i)\right|^2\right] + \sum_{i=0}^{n-1}\alpha^i E\left[\left|f_m(n-i)\right|^2\right] \tag{B.16}$$

and $E\left[\mu_{m+1}(n)^2\right]$ is

$$E\left[\mu_{m+1}(n)\,\mu^*_{m+1}(n)\right] = E\left[\left\{\alpha^{n-1}\mu_{m+1}(0) + \sum_{i=0}^{n-1}\alpha^i\left|b_m(n-1-i)\right|^2 + \sum_{i=0}^{n-1}\alpha^i\left|f_m(n-i)\right|^2\right\}\right.$$

$$\left\{\alpha^{n-1}\,\mu_{m+1}(0)+\sum_{i=0}^{n-1}\alpha^{i}\left|b_m(n-1-i)\right|^2+\sum_{i=0}^{n-1}\alpha^{i}\left|f_m(n-i)\right|^2\right\}^{*}\Bigg] \qquad \text{(B.17)}$$

The MSE at stage m, $J_{m+1}(n+1)$, is defined as:

$$J_{m+1}(n+1) = E\left[e_{m+1}(n+1)e^{*}_{m+1}(n+1)\right] \qquad \text{(B.18)}$$

Replacing the estimation error at stage m, $e_{m+1}(n+1)$, with (B.4), the MSE $J_{m+1}(n+1)$ can be rewritten as:

$$J_{m+1}(n+1)=E\Big[\left(e_{m+1\,opt}(n+1) - \varepsilon_m(n+1)\,b_m(n+1)\right)$$

$$\left(e_{m+1\,opt}(n+1) - \varepsilon_m(n+1)\,b_m(n+1)\right)^{*}\Big] \qquad \text{(B.19)}$$

Assuming $e_{m+1\,opt}$ and $b_m(n+1)$ are orthogonal, and using (B.6), we can rewrite (B.19) as:

$$J_{m+1}(n+1)=E\left[e_{m+1\,opt}(n+1)e^{*}_{m+1\,opt}(n+1)\right]+K_m(n+1)E\left[b_m(n+1)b_m(n+1)^{*}\right]$$

$$= J_{m+1\,opt} + J_{m+1\,exc}(n+1) \qquad \text{(B.20)}$$

where

$$J_{m+1\,opt} = E\left[e_{m+1\,opt}(n+1)e_{m+1\,opt}(n+1)^{*}\right]$$

$$= \sigma^2_{m+1\,opt}$$

$$= \sigma^2_n \qquad \text{(B.21)}$$

σ_n^2 is the variance of the observation noise, and the excess noise at stage m, $J_{m+1\,exc}(n+1)$, is defined as:

$$J_{m+1\,exc}(n+1) = K_m(n+1)E\left[b_m(n+1)b_m(n+1)^*\right] \tag{B.22}$$

where $K_m(n+1)$ is defined as in (B.12).

For n large enough

$$E\left[b_m(n+1)b_m(n+1)^*\right] = E\left[b_m(n)b_m(n)^*\right] = \sigma_{b,m}^2 \tag{B.23}$$

$$E\left[f_m(n+1)f_m(n+1)^*\right] = E\left[f_m(n)f_m(n)^*\right] = \sigma_{f,m}^2 \tag{B.24}$$

and

$$\sigma_{b,m}^2 \approx \sigma_{f,m}^2 \tag{B.25}$$

so that from (B.16), we obtain

$$E\left[\mu_{m+1}(n)\right] = \alpha^{n-1}\mu_{m+1}(0) + \sigma_{b_m}^2\sum_{i=0}^{n-1}\alpha^i + \sigma_{f_m}^2\sum_{i=0}^{n-1}\alpha^i$$

$$= \alpha^{n-1}\mu_{m+1}(0) + 2\sigma_{b_m}^2\sum_{i=0}^{n-1}\alpha^i$$

$$= \alpha^{n-1}\mu_{m+1}(0) + 2\sigma_{b_m}^2\frac{1-\alpha^n}{1-\alpha} \tag{B.26}$$

and from (B.17), assuming all the variables are real number, we get

$$E\left[\mu_{m+1}(n)\mu^{*}_{m+1}(n)\right]=\left\{\alpha^{n-1}\mu^{2}_{m+1}(0)+\sigma^{2}_{b_m}\sum_{i=0}^{n-1}\alpha^{i}+\sigma^{2}_{f_m}\sum_{i=0}^{n-1}\alpha^{i}\right\}^{2}$$

$$=\left\{\alpha^{n-1}\mu^{2}_{m+1}(0)+2\sigma^{2}_{b_m}\sum_{i=0}^{n-1}\alpha^{i}\right\}^{2}$$

$$=\left\{\alpha^{n-1}\mu^{2}_{m+1}(0)+2\sigma^{2}_{b_m}\frac{1-\alpha^{n}}{1-\alpha}\right\}^{2} \qquad \text{(B.27)}$$

By squaring and re-arranging (B.27), we obtain

$$E\left[\mu_{m+1}(n)\mu^{*}_{m+1}(n)\right]=\alpha^{2(n-1)}\mu^{2}_{m+1}(0)+4\alpha^{n-1}\mu_{m+1}(0)\sigma^{2}_{b_m}\frac{1-\alpha^{n}}{1-\alpha}+4\sigma^{4}_{b_m}\left(\frac{1-\alpha^{n}}{1-\alpha}\right)^{2} \qquad \text{(B.28)}$$

Using the fact that for large n

$$K_m(n+1)=K_m(n) \qquad \text{(B.29)}$$

we can write (B.12) as

$$K_m(n+1)=\frac{P_{m+1}(n+1)}{1-B_{m+1}(n+1)} \qquad \text{(B.30)}$$

As n increases without bound, from (B.26) we get

$$E\left[\mu_{m+1}(n)\right]\rightarrow 2\sigma^{2}_{b_m}\frac{1}{1-\alpha} \qquad \text{(B.31)}$$

and from (B.27) we get

$$E\left[\mu_{m+1}(n)\mu^{*}_{m+1}(n)\right] \rightarrow 4\sigma^{4}_{b_m}\left(\frac{1}{1-\alpha}\right)^{2} \tag{B.32}$$

Applying (B.31) and (B.32) to (B.13), we get:

$$\begin{aligned} B_{m+1}(n+1) &= 1-\frac{4\sigma^{2}_{b_m}}{2\sigma^{2}_{b_m}\frac{1}{1-\alpha}}+\frac{12\left(\sigma^{2}_{b_m}\right)^{2}}{4\sigma^{4}_{b_m}\left(\frac{1}{1-\alpha}\right)^{2}} \\ &= 1-2+2\alpha+3-6\alpha+3\alpha^{2} \\ &= 2-4\alpha+3\alpha^{2} \end{aligned} \tag{B.33}$$

and by applying (B.31) and (B.32) to (B.14), we get:

$$\begin{aligned} P_{m+1}(n+1) &= \frac{4\sigma^{2}_{n}\sigma^{2}_{b_m}}{4\sigma^{4}_{b_m}\left(\frac{1}{1-\alpha}\right)^{2}} \\ &= \frac{\sigma^{2}_{n}(1-\alpha)^{2}}{\sigma^{2}_{b_m}} \end{aligned} \tag{B.34}$$

where

$E\left[e_{m+1\,opt}(n)e^{*}_{m+1\,opt}(n)\right] = \sigma^{2}_{n}$ is the variance of the observation noise.

Using (B.33) and (B.34) in (B.30) the variance of the regression-coefficient-error vector $K_m(n+1)$ as $n \rightarrow \infty$, can be found:

$$K_m(n+1) \rightarrow \frac{\sigma^{2}_{n}(1-\alpha)^{2}}{\sigma^{2}_{b_m}(1-\alpha^{2})}$$

$$\rightarrow \frac{\sigma_n^2(1-\alpha)}{\sigma_{b_m}^2(1+\alpha)} \tag{B.35}$$

Finally the MSE for stage m when $n \rightarrow \infty$ is

$$J_{m+1}(n+1) = J_{m+1\,opt} + J_{m+1\,exc}(n+1)$$

$$= \sigma_n^2 + \frac{\sigma_n^2(1-\alpha)}{(1+\alpha)} \tag{B.36}$$

The Time-constant for Convergence in the Mean

Taking the expected value of the regression-coefficient-error vector $\varepsilon_m(n+1)$ in (B.5), yields:

$$E[\varepsilon_m(n+1)] = E\left[\left[1 - \frac{2\, b_m(n)\, b_m^*(n)}{\mu_{m+1}(n)}\right]\varepsilon_m(n) + \frac{2 b_m(n)\, e_{m+1\,opt}^*(n)}{\mu_{m+1}(n)}\right] \tag{B.37}$$

Assuming that $e_{m+1\,opt}(n)$ and $b_m(n)$ are orthogonal, we obtain

$$E[\varepsilon_m(n+1)] = E\left[1 - \frac{2\, b_m(n)\, b_m^*(n)}{\mu_{m+1}(n)}\right] E[\varepsilon_m(n)] \tag{B.38}$$

By iterating (B.38), we obtain:

$$E[\varepsilon_m(n+1)] = \left[\prod_{i=0}^{n}\left[1 - \frac{2\, E[b_m(i)\, b_m^*(i)]}{E[\mu_{m+1}(i)]}\right]\right] \varepsilon_m(0) \tag{B.39}$$

For large n, by assuming

$$E\left[b_m(n)\, b_m^*(n)\right] = E\left[b_m(n-1)\, b_m^*(n-1)\right]$$

and

$$E\left[\mu_{m+1}(n)\right] = E\left[\mu_{m+1}(n-1)\right]$$

we obtain

$$E\left[\varepsilon_m(n+1)\right] = \left[\, 1 - \frac{2\, E\left[b_m(n)\, b_m^*(n)\right]}{E\left[\mu_{m+1}(n)\right]} \right]^n \varepsilon_m(0) \qquad \text{(B.40)}$$

so that the time constant at stage m, τ_m, is

$$\tau_m = \frac{-1}{\ln\left(1 - \dfrac{2\, E\left[b_m(n)\, b_m^*(n)\right]}{E\left[\mu_{m+1}(n)\right]}\right)} \qquad \text{(B.41)}$$

The MSE for the multi-stage GAL-JP Algorithm

The dependencies amongst the reflection coefficients, the backward prediction errors, the forward prediction errors, and the regression coefficients make the analysis of the multi-stage GAL-JP algorithm difficult. Here we try to analyze the MSE of the multi-stage GAL-JP algorithm using the independence assumption, and then observe the deviation from an experiment.

The output of stage m, $y_{m+1}(n)$, of the GAL-JP algorithm is stated as:

$$y_{m+1}(n+1) = \sum_{i=1}^{m} c_i^*(n+1)b_i(n+1) \tag{B.42}$$

where $c_{m+1}(n+1)$ is the regression coefficient at stage m, and $b_m(n+1)$ is the backward prediction error at stage m. The estimation error $e_{m+1}(n+1)$ is:

$$\begin{aligned} e_{m+1}(n+1) &= d(n+1) - y_{m+1}(n+1) \\ &= d(n+1) - \sum_{i=0}^{m} c_i^*(n+1)b_{i-1}(n+1) \\ &= d(n+1) - \underline{c}^*(n+1)\underline{b}(n+1) \end{aligned} \tag{B.43}$$

where

$\underline{c} = [c_0, c_2, \cdots, c_m]^T$ is the regression coefficient vector

$\underline{b} = [b_0, b_2, \cdots, b_m]^T$ is the backward prediction error vector at order m

and replacing $\underline{c}$ with the regression-coefficient-error vector $\underline{\varepsilon}(n+1)$ as in (B.4), we obtain:

$$e_{m+1}(n+1) = e_{m+1\,opt}(n+1) - \underline{\varepsilon}^*(n+1)\underline{b}\,(n+1) \tag{B.44}$$

where $\underline{\varepsilon}(n+1) = [\varepsilon_0(n+1), \varepsilon_2(n+1), \cdots, \varepsilon_m(n+1)]^T$ is the regression-coefficient-error vector.

The MSE $J_{m+1}(n+1)$ is:

$$J_{m+1}(n+1) = E\left[e_{m+1}(n+1)e_{m+1}^{*}(n+1)\right]$$

$$= E\Big[\left\{e_{m+1\,opt}(n+1) - \underline{\varepsilon}^{T}(n+1)\underline{b}\,(n+1)\right\}$$

$$\left\{e_{m+1\,opt}(n+1) - \underline{\varepsilon}^{T}(n+1)\underline{b}\,(n+1)\right\}^{*}\Big]$$

$$= E\left[e_{m+1\,opt}(n+1)e_{m+1\,opt}^{*}(n+1)\right] - E\left[e_{m+1\,opt}^{*}(n+1)\underline{\varepsilon}^{T}(n+1)\underline{b}\,(n+1)\right]$$

$$- E\left[e_{m+1\,opt}(n+1)\underline{b}_{m}^{*}(n+1)\underline{\varepsilon}\,(n+1)\right]$$

$$+ E\left[\left\{\underline{\varepsilon}^{T}(n+1)\underline{b}\,(n+1)\right\}\left\{\underline{\varepsilon}^{T}(n+1)\underline{b}\,(n+1)\right\}^{*}\right] \qquad \text{(B.45)}$$

By assuming that $e_{m+1\,opt}(n+1)$ and $b_m(n+1)$ are orthogonal, we obtain the MSE $J_{m+1}(n+1)$ as:

$$J_{m+1}(n+1) = \sigma_n^2 + E\left[\underline{\varepsilon}^{T}(n+1)\underline{b}\,(n+1)\underline{b}^{*}(n+1)\underline{\varepsilon}\,(n+1)\right] \qquad \text{(B.46)}$$

Using the assumption that after the prediction section of the GAL-JP algorithm, as in (B.1), has converged, the backward prediction errors $b_i(n+1)$and $b_k(n+1)$ are uncorrelated for $i \neq k$ [16], and also $\varepsilon_i(n+1)$ and $\varepsilon_k(n+1)$ are uncorrelated for $i \neq k$. We can now write the MSE $J_{m+1}(n+1)$ as

$$J_{m+1}(n+1) = \sigma_n^2 + \sum_{i=0}^{m} E\left[\varepsilon_i(n+1)\varepsilon_i^{*}(n+1)\right]E\left[b_i(n+1)b_i^{*}(n+1)\right]$$

$$= \sigma_n^2 + \sum_{i=0}^{m} K_i(n+1)E\left[b_i(n+1)b_i^*(n+1)\right] \qquad \text{(B.47)}$$

where, as in (B.6),

$$K_i(n+1) = E\left[\varepsilon_i(n+1)\varepsilon_i^*(n+1)\right]$$

As $n \to \infty$, by using the final value of $K_i(n+1)$ and $b_i(n+1)$ as shown in (B.35) and (B.23) respectively, we get the final value of the MSE for m stages of the GAL-JP algorithm, $J_{m+1}(n+1)$, as:

$$J_{m+1}(n+1) = \sigma_n^2 + \frac{\sigma_n^2(1-\alpha)}{(1+\alpha)}(m+1) \qquad \text{(B.48)}$$

APPENDIX C.1

The RLSL Algorithm using a Priori Error Feedback

The RLSL-JP algorithm using a priori estimation errors with error feedback consists of two sections: a *Prediction* section and a *Filtering* section, which can be summarized as follows [16];

Prediction:

M is the last stage of the lattice predictor

m=1,..., M

n = number of iterations

$$\eta_m(n) = \eta_{m-1}(n) + \Gamma^*_{f,m}(n-1)\psi_{m-1}(n) \qquad \text{(C.1.1.a)}$$

$$\psi_m(n) = \psi_{m-1}(n-1) + \Gamma^*_{b,m}(n-1)\eta_{m-1}(n) \qquad \text{(C.1.1.b)}$$

$$\mathfrak{I}_{m-1}(n) = \alpha\mathfrak{I}_{m-1}(n-1) + \gamma_{m-1}(n-1)\left|\eta_{m-1}(n)\right|^2 \qquad \text{(C.1.1.c)}$$

$$\mathbf{B}_{m-1}(n) = \alpha\mathbf{B}_{m-1}(n-1) + \gamma_{m-1}(n)\left|\psi_{m-1}(n)\right|^2 \qquad \text{(C.1.1.d)}$$

$$\Gamma_{f,m}(n) = \Gamma_{f,m}(n-1) - \frac{\gamma_{m-1}(n-1)\psi_{m-1}(n-1)\eta^*_m(n)}{\mathbf{B}_{m-1}(n-1)} \qquad \text{(C.1.1.e)}$$

$$\Gamma_{b,m}(n) = \Gamma_{b,m}(n-1) - \frac{\gamma_{m-1}(n-1)\eta_{m-1}(n-1)\psi^*_m(n)}{\mathfrak{I}_{m-1}(n)} \qquad \text{(C.1.1.f)}$$

$$\gamma_m(n) = \gamma_m(n-1) - \frac{\gamma_{m-1}^2(n-1)\left|\psi_{m-1}(n)\right|^2}{B_{m-1}(n)} \qquad \text{(C.1.1.g)}$$

Filtering

m =0, . . ., M

$$e_{m+1}(n) = e_m(n) - c_m^*(n-1)\,\psi_m(n) \qquad \text{(C.1.2.a)}$$

$$B_m(n) = \alpha B_m(n-1) + \gamma_m(n)\left|\psi_m(n)\right|^2 \qquad \text{(C.1.2.b)}$$

$$c_m(n) = c_m(n-1) + \frac{\gamma_m(n)\psi_m(n)e_{m+1}^*(n)}{B_m(n)} \qquad \text{(C.1.2.c)}$$

Initialization

1. For n=0

$B_{m-1}(0) = \delta$, δ small positive number (I use δ = .1)

$\Im_{m-1}(0) = \delta$

$\Gamma_{f,m}(0) = \Gamma_{b,m}(0) = 0$

$\gamma_0(0) = 1$

$c_m(0) = 0$

2. For n ≥ 1

$\eta_0(n) = \psi_0(n) = u(n)$

$$\mathfrak{I}_0(n) = \mathbf{B}_0(n) = \alpha\mathfrak{I}_0(n - 1) + |u(n)|^2$$

$$\alpha_0(n) = d(n)$$

where $\eta_m(n)$ is the *a priori forward prediction* error, $f_m(n)$ is the *a posteriori forward prediction* error, $\psi_m(n)$ is the *a priori backward prediction* error, $b_m(n)$ is the *a posteriori backward prediction* error, $\Gamma_{f,m}(n)$ is the forward reflection coefficient, and $\Gamma_{b,m}(n)$ is the backward reflection coefficient.

APPENDIX C.2

MSE and Convergence Analysis for the RLSL-JP Algorithm

The mean of the conversion factor at stage m, $\gamma_m(n)$, is computed by taking the expected value of (C.1.1.g) in Appendix C1 as:

$$E[\gamma_m(n)] = E\left[\gamma_{m-1}(n) - \frac{\gamma^2_{m-1}(n)\,|\psi_{m-1}(n)|^2}{B_{m-1}(n)}\right] \tag{C.2.1}$$

Assuming that $\gamma_m(n)$ and $\psi_{m-1}(n)$ are independent, and using the approximation $E\left[\frac{x}{y}\right] \approx \frac{E[x]}{E[y]}$, we can write $E[\gamma_m(n)]$ as:

$$E[\gamma_m(n)] = E[\gamma_{m-1}(n)] - \frac{E[\gamma^2_{m-1}(n)]\,E[|\psi_{m-1}(n)|^2]}{E[B_{m-1}(n)]} \tag{C.2.2}$$

The expected value of the sum of weighted backward a posteriori prediction error squares $B_{m-1}(n)$ is obtained by taking the expected value of (C.1.1.d),

$$E[B_{m-1}(n)] = \alpha\, E[B_{m-1}(n-1)] + E[\gamma_{m-1}(n)|\psi_{m-1}(n)|^2] \tag{C.2.3}$$

Assuming that the conversion factor $\gamma_{m-1}(n)$ and the *a priori backward prediction* error $\psi_{m-1}(n)$ are independent, we can write (C.2.3) as:

$$E[B_{m-1}(n)] = \alpha\, E[B_{m-1}(n-1)] + E[\gamma_{m-1}(n)]\,E[|\psi_{m-1}(n)|^2] \tag{C.2.4}$$

For n so large that $E[B_{m-1}(n)] = E[B_{m-1}(n-1)]$, we obtain

$$E[B_{m-1}(n)] = \frac{E[\gamma_{m-1}(n)]\, E[|\psi_{m-1}(n)|^2]}{(1 - \alpha)} \qquad \text{(C.2.5)}$$

or, rewritten in another form,

$$\frac{E[|\psi_{m-1}(n)|^2]}{E[B_{m-1}(n)]} = \frac{(1-\alpha)}{E[\gamma_{m-1}(n)]} \qquad \text{(C.2.6)}$$

By using (C.2.6) to substitute in the second term of (C.2.2) and assuming that the variance of the conversion factor is small after convergence, so that $E[\gamma_m^2(n)] \approx [E[\gamma_m(n)]]^2$, we get

$$E[\gamma_m(n)] = E[\gamma_{m-1}(n)](\alpha) \qquad \text{(C.2.7)}$$

By replacing the conversion factor at stage m-1 in (C.2.7) with the conversion factor at m-2, all the way to the conversion factor at the zeroth stage, $\gamma_0(n)$, we obtain:

$$E[\gamma_m(n)] = \alpha^m \gamma_0(n) \qquad \text{at stage m} \qquad \text{(C.2.8)}$$

and

$$E[\gamma_m(n)\gamma_m^*(n)] \approx [E[\gamma_m(n)]]^2$$

$$= \alpha^{2m} \qquad \text{(C.2.9)}$$

where $\gamma_0(n)$ is set to 1.

By replacing $E[\gamma_m(n)]$ in (C.2.5) with (C.2.8), we obtain:

$$E[B_{m-1}(n)] = \frac{\alpha^m}{(1-\alpha)} E[|\psi_{m-1}(n)|^2] \qquad \text{(C.2.10)}$$

and the variance of $B_{m-1}(n)$, $E[B_{m-1}(n)B_{m-1}^*(n)]$ is obtained from (C.1.1.d):

$$E[B_{m-1}(n)\, B_{m-1}^*(n)] = E\Big[\{\alpha B_{m-1}(n-1) + \gamma_{m-1}(n)|\psi_{m-1}(n)|^2\}$$

$$\{\alpha B_{m-1}(n-1) + \gamma_{m-1}(n)|\psi_{m-1}(n)|^2\}^*\Big] \qquad \text{(C.2.11)}$$

By multiplying (C.2.11) out and assuming $\gamma_{m-1}(n)$ and $\psi_{m-1}(n)$ are independent, we can write (C.2.11) as

$$E[B_{m-1}(n)\, B_{m-1}^*(n)] = \alpha^2 E[B_{m-1}(n-1)B_{m-1}^*(n-1)]$$

$$+2\alpha E[B_{m-1}(n-1)]\, E[\gamma_{m-1}(n)]\, E[|\psi_{m-1}(n)|^2]$$

$$+ E[\gamma_{m-1}(n)\, \gamma_{m-1}^*(n)]\, E[|\psi_{m-1}(n)|^2]\, E[|\psi_{m-1}(n)|^2] \qquad \text{(C.2.12)}$$

By substituting the value (C.2.10) into (C.2.12), assuming that n is so large that

$$E\left[B_{m-1}(n)B^{*}_{m-1}(n)\right] = E\left[B_{m-1}(n-1)B^{*}_{m-1}(n-1)\right]$$

we get

$$E\left[B_{m-1}(n)\, B^{*}_{m-1}(n)\right] = \alpha^{2}\, E\left[B_{m-1}(n-1)B^{*}_{m-1}(n-1)\right]$$

$$+2\,\alpha\,\frac{\alpha^{m}}{1-\alpha}\,\alpha^{m}\,E\left[\left|\psi_{m-1}(n)\right|^{4}\right] + \alpha^{2m}\,E\left[\left|\psi_{m-1}(n)\right|^{4}\right]$$

$$=\frac{\alpha^{2m}}{(1-\alpha)^{2}}\left[E\left[\left|\psi_{m-1}(n)\right|^{4}\right]\right] \qquad \text{(C.2.13)}$$

The a *priori estimation* error at stage m, $e_{m+1}(n)$, is found from (C.1.2.a)

$$e_{m+1}(n) = e_{m}(n) - c^{*}_{m}(n-1)\,\psi_{m}(n)$$

$$= e_{m+1\,opt} - \varepsilon^{*}_{m}(n-1)\,\psi_{m}(n) \qquad \text{(C.2.14)}$$

where

$$\varepsilon_{m}(n) = c_{m}(n) - c_{m\,opt} \qquad \text{(C.2.15.a)}$$

is the *regression-coefficient error* at stage m, and $c_{m+1\,opt}$ is the optimum regression coefficient at stage m, and

$$e_{m+1\,opt}(n) = e_{m+1}(n) - c^{*}_{m\,opt}\ \psi_m(n) \qquad \text{(C.2.15.b)}$$

is the minimum error.

Recall (C. 1.2.c) and replace $c_m(n)$ with $\varepsilon_m(n)$, to get

$$\varepsilon_m(n) = \varepsilon_m(n-1) + \frac{\gamma_m(n)\,\psi_m(n)e^{*}_{m+1}(n)}{\mathbf{B}_m(n)} \qquad \text{(C.2.16)}$$

Substituting for $e_{m+1}(n)$ with (C.2.14) and rearranging, we obtain

$$\varepsilon_m(n) = \left[1 - \frac{\gamma_m(n)\,\psi_m(n)\,\psi^{*}_m(n)}{\mathbf{B}_m(n)}\right]\varepsilon_m(n-1)$$

$$+ \frac{\gamma_m(n)\,\psi_m(n)e^{*}_{m+1\,opt}(n)}{\mathbf{B}_m(n)} \qquad \text{(C.2.17)}$$

Taking the expected value of (C.2.17) and assuming $e_{m\,opt}(n)$ has zero mean and is independent from $\psi_m(n)$, yields

$$E\left[\varepsilon_m(n)\right] = \left[1 - \frac{E\left[\gamma_m(n)\right] E\left[\left|\psi_m(n)\right|^2\right]}{E\left[\mathbf{B}_m(n)\right]}\right] E\left[\varepsilon_m(n-1)\right] \qquad \text{(C.2.18)}$$

Substituting (C.2.6) into (C.2.18), we get:

$$E\left[\varepsilon_m(n)\right] = \alpha\, E\left[\varepsilon_m(n-1)\right] \qquad \text{(C.2.19)}$$

By iterating (C.2.19), we then obtain

$$E[\varepsilon_m(n)] = \alpha^n \varepsilon_m(0) \qquad \text{(C.2.20)}$$

The convergence time of the regression coefficient can be derived from (C.2.20), assuming that the previous stages have converged, which yields

$$\tau_m = \frac{-1}{\ln(\alpha)} \qquad 0 < \alpha < 1 \qquad \text{(C.2.21)}$$

The *variance* of the *regression-coefficient error* $K_m(n)$ is defined as:

$$K_m(n) = E[\varepsilon_m(n)\varepsilon_m(n)^*] \qquad \text{(C.2.22)}$$

Substituting for the regression-coefficient error $\varepsilon_m(n)$ with (C.2.17), we get

$$K_m(n) = E\left[\left\{\left[1 - \frac{\gamma_m(n)\psi_m(n)\psi_m^*(n)}{\mathbf{B}_m(n)}\right]\varepsilon_m(n-1) + \frac{\gamma_m(n)\psi_m(n)e_{m+1\,opt}^*(n)}{\mathbf{B}_m(n)}\right\}\right.$$

$$\left.\left\{\left[1 - \frac{\gamma_m(n)\psi_m(n)\psi_m^*(n)}{\mathbf{B}_m(n)}\right]\varepsilon_m(n-1) + \frac{\gamma_m(n)\psi_m(n)e_{m+1\,opt}^*(n)}{\mathbf{B}_m(n)}\right\}^*\right] \qquad \text{(C.2.23)}$$

For large n, assuming $e_{m+1\,opt}(n)$ has zero mean and is independent from $\psi_m(n)$, and substituting (C.2.8) through (C.2.10) into (C.2.23), we get:

$$K_m(n) = \alpha^2 E[\varepsilon_m(n-1)\varepsilon_m(n-1)^*] + \sigma^2_{m\,opt}\frac{(1-\alpha)^2}{E[|\psi_m(n)|^2]} \qquad \text{(C.2.24)}$$

where

$$\sigma^2_{m\,opt} = E\left[e_{m\,opt}(n+1)e^*_{m\,opt}(n+1)\right]$$

By iterating (C.2.24) we find

$$K_m(n) = \alpha^{2n} K_m(0) + \sigma^2_{m\,opt}\left(1 - \alpha\right)^2 \sum_{i=0}^{n-1} \frac{\alpha^{2(n-1-i)}}{E\left[\left|\psi_m(i)\right|^2\right]} \qquad \text{(C.2.25)}$$

Under the optimum condition, with the length of the lattice longer than the autocorrelation depth of the input signal, $\sigma^2_{m\,opt}$ is equal to the *observation noise* σ^2_n.

At n so large that $E\left[\left|\psi_m(n)\right|^2\right] = E\left[\left|\psi_m(n-1)\right|^2\right]$ we get

$$K_m(n) \approx \sigma^2_n \frac{(1 - \alpha)^2\left(1 - \alpha^{2n}\right)}{\left(1 - \alpha^2\right)E\left[\left|\psi_m(n)\right|^2\right]}$$

$$= \sigma^2_n \frac{(1 - \alpha)\left(1 - \alpha^{2n}\right)}{(1 + \alpha)E\left[\left|\psi_m(n)\right|^2\right]} \qquad \text{(C.2.26)}$$

The MSE at stage m, $J_{m+1}(n)$, is

$$J_{m+1}(n) = E\left[e_{m+1}(n)e^*_{m+1}(n)\right]$$

$$= E\left[\left\{e_{m+1\,opt}(n) - \varepsilon^*_m(n-1)\psi_m(n)\right\}\left\{e_{m+1\,opt}(n) - \varepsilon^*_m(n-1)\psi_m(n)\right\}^*\right]$$

$$= \sigma_n^2 + E\left[\left|\psi_m(n)\right|^2\right] K_m(n-1) \qquad \text{(C.2.27)}$$

$$= J_{m+1\,opt} + J_{m+1\,ex}(n) \qquad \text{(C.2.28)}$$

where

$$J_{m+1\,ex}(n) = E\left[\left|\psi_m(n)\right|^2\right] K_m(n-1)$$

and

$$J_{m+1\,opt} = \sigma_n^2$$

At $n \rightarrow \infty$, the final value of $K_m(n-1)$ is equal to $K_m(n)$, that is, from (C.2.26),

$$K_m(n) = \sigma_n^2 \frac{(1 - \alpha)}{(1 + \alpha) E\left[\left|\psi_m(n)\right|^2\right]} \qquad \text{(C.2.29)}$$

By substituting (C.2.29) into (C.2.27) we get the final value of the MSE at stage m, $J_{m+1}(n)$, as:

$$J_{m+1}(n) \approx \sigma_n^2 \left(1 + \frac{1 - \alpha}{1 + \alpha}\right)$$

$$= \sigma_n^2 \left(\frac{2}{1 + \alpha}\right) \qquad \text{(C.2.30)}$$

Equation (C.2.30) indicates that the smallest MSE is achieved when α is close to unity.

C.2.1 The MSE for the Multi-stage RLSL-JP Algorithm

The output of stage m, $y_{m+1}(n)$, of the RLSL-JP algorithm is stated as

$$y_{m+1}(n) = \sum_{i=1}^{m} c_i^*(n-1)\ \psi_i(n) \qquad \text{(C.2.31)}$$

Recall the a priori estimation error at stage m, $e_{m+1}(n)$, as defined in (C.2.14) and rewrite it with respect to the desired response $d(n)$:

$$e_{m+1}(n) = d(n) - y_{m+1}(n)$$

$$= d(n) - \underline{c}^*(n-1)\,\underline{\psi}(n) \qquad \text{(C.2.32)}$$

where

$$\underline{c}(n-1) = \left[c_0(n-1),\ c_1(n-1),\ \cdots,\ c_m(n-1)\right]^H$$

and

$$\underline{\psi}(n) = \left[\psi_0(n),\ \psi_1(n),\ \cdots,\ \psi_m(n)\right]^H$$

and replacing $\underline{c}(n)$ in (C.2.32) with the regression-coefficient-error vector $\underline{\varepsilon}(n)$, we obtain

$$e_{m+1}(n) = e_{m+1\,opt}(n) - \underline{\varepsilon}^*(n-1)\,\underline{\psi}(n) \qquad \text{(C.2.33)}$$

The MSE, $J_{m+1}(n)$, is

$$J_{m+1}(n) = E\left[e_{m+1}(n)e^{*}_{m+1}(n)\right] \tag{C.2.34}$$

substituting (C.2.33) into (C.2.34), we obtain

$$J_{m+1}(n) = E\Big[\Big\{e_{m+1\,opt}(n) - \underline{\varepsilon}^{*}(n-1)\underline{\psi}(n)\Big\}$$

$$\Big\{e_{m+1\,opt}(n) - \underline{\varepsilon}^{*}(n-1)\underline{\psi}(n)\Big\}^{*}\Big]$$

$$= E\left[e_{m+1\,opt}(n)e^{*}_{m+1\,opt}(n)\right] - E\left[e^{*}_{m+1\,opt}(n)\underline{\varepsilon}^{*}(n-1)\underline{\psi}(n)\right]$$

$$- E\left[e_{m+1\,opt}(n)\underline{\psi}^{*}_{m}(n)\underline{\varepsilon}(n-1)\right]$$

$$+ E\Big[\Big\{\underline{\varepsilon}^{*}(n-1)\underline{\psi}(n)\Big\}\Big\{\underline{\varepsilon}^{*}(n-1)\underline{\psi}(n)\Big\}^{*}\Big] \tag{C.2.35}$$

By assuming that $e_{m+1\,opt}(n)$ and $\psi_m(n)$ are orthogonal, we obtain the MSE $J_{m+1}(n)$ as

$$J_{m+1}(n) = \sigma_n^2 + E\left[\underline{\varepsilon}^{*}(n-1)\underline{\psi}(n)\underline{\psi}^{*}(n)\underline{\varepsilon}(n-1)\right] \tag{C.2.36}$$

Using the assumption that after the prediction section of the RLSL-JP algorithm, as in (C.1.1), has converged, the backward prediction errors $\psi_i(n)$and $\psi_i(n)$ are uncorrelated for i≠k [16], and also $\varepsilon_i(n-1)$ and $\varepsilon_k(n-1)$ are uncorrelated for i≠k. We can now write the MSE $J_{m+1}(n)$ as

$$\approx \sigma_n^2 + \sum_{i=0}^{m} K_i(n-1) E\left[\left|\psi_i(n)\right|^2\right] \qquad \text{(C.2.37)}$$

where $K_i(n-1)$ is defined in (C.2.22).

At $n \rightarrow \infty$, by using the final value of $K_i(n)$ as in (C.2.29), we obtain the final value of MSE for the m stages of the RLSL-JP algorithm, $J_{m+1}(n)$, as:

$$J_{m+1}(n) = \sigma_n^2 + \sigma_n^2 \frac{1-\alpha}{1+\alpha}(m+1) \qquad \text{(C.2.38)}$$

It turns out that the final MSE value for the RLSL-JP algorithm is equivalent to the final MSE value for the GAL-JP algorithm as derived in Appendix B.

APPENDIX D

DECIBEL UNITS

Basically, the decibel unit is a logarithm to the base 10 of a power ratio.

dBm : power level relative to 1 mW

$$\text{power(dBm)} = 10 \log \frac{\text{power(mW)}}{1\ \text{mW}}$$

dBW : power level relative to 1 W

$$\text{power(dBm)} = 10 \log \frac{\text{power(W)}}{1\ \text{W}}$$

0 dBW = 30 dBm

A test level point (TLP) is a location in a circuit at which a certain level of test tone is expected during testing. A 0 TLP is a point in a circuit where the level of the test tone is expected to be 0 dBm [11]. Other points in the circuit are then related to the 0 TLP using the unit dBr (dB reference). The unit dBm0 is an absolute unit of power in dBm that applies at the 0 TLP [11, pp.17]. The power level at any test point now follows from:

dBm = dBm0 + dBr

for example, a test point at -22 dBr, with -10 dBm0, means that we have a signal with power -32 dBm at the test point.

REFERENCES

[1] S. Thomas Alexander, *Adaptive Signal Processing Theory and Applications*. New York: Springer-Verlag New York Inc., 1986.

[2] A. A. (Louis) Beex, *Efficient Generation of ARMA Cross Covariance Sequence*. Proceedings ICASSP 1985, pp. 327-330, 1985.

[3] CCITT Recommendation (1980). *Echo-Canceller*. CCITT Recommendation Fascicle III.1-Rec G.165, Geneva.

[4] Hyokang Chang, and B. P. Agrawal, *A DSP-Based Echo- Canceller with Two Adaptive Filters*. Proceedings Globecom, pp. 1674-1678, 1986.

[5] Rong-Yih Chen and Chin-liang Wang, *On the Optimum Step-size for the Adaptive Sign and LMS Algorithms*. IEEE Transactions on Circuits and Systems, vol. 37, no. 6, p. 836-840, June 1990.

[6] Gordon W. Davidson and David D. Falconer, *Reduced Complexity Echo Cancellation Using Orthonormal Function*. IEEE Transactions on Circuits and Systems, vol. 38, no. 1, pp. 20-28, January 1991.

[7] D.L. Duttweiler, *A Single-Chip VLSI Echo Canceller*. The Bell System Technical Journal, vol. 59, no. 2, pp. 149-159, February 1980.

[8] Hong Fan and Majid Nayeri, *Asymptotic Stability of SMM and RGM for Insufficient Order IIR Adaptive Filters*. Proceedings ICASSP 1990, pp. 1265-1268, 1990.

[9] Hong Fan and Majid Nayeri, *On Error Surfaces of Sufficient Order Adaptive IIR FIlters: Proofs and Counterexamples to a Unimodality Conjecture*. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol 37, no. 9, pp. 1436-1442, September 1989.

[10] Hong Fan and W. Kenneth Jenkins, *An Investigation of an Adaptive IIR Echo Canceller: Advantages and Problems*. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 36, no. 12, pp. 1819-1834, December 1988.

[11] Roger L. Freeman, *Telecommunication Transmission Handbook*. New York: John Wiley & Sons, 1981.

[12] Fonollosa, Josê A. Rodriguez, and Enrique Masgrau, *Improved Convergence of Gradient Algorithms for Adaptive IIR Filters*. Proceedings ISCAS 1989, pp. 1732-1735, 1989.

[13] Benjamin Friedlander, *Lattice Filters for Adaptive Processing.* Proceedings of the IEEE. vol. 70, no. 8, pp.829-865, August 1982.

[14] Lloyd J. Griffiths, *A Continuously- Adaptive Filter Implemented As a Lattice Structure.* Proceedings ICASSP1977, pp. 683-686, 1977.

[15] Lloyd J. Griffiths, *An Adaptive Lattice Structure for Noise Cancelling Applications.* Proceedings ICASSP1978, pp. 87-90, 1978.

[16] Simon S. Haykin, *Adaptive Filter Theory.* Englewood Cliffs: Prentice Hall Inc., 1991.

[17] Michael L. Honig and David G. Messerschmitt, *Adaptive Filters: Structure, Algorithms, and Applications.* Kluwer Academic Publishers, 1984.

[18] Larry L. Horowitz and Kenneth D. Senne, *Performance Advantage of Complex LMS for Controlling Narrow-Band Adaptive Arrays.* IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-29, no. 3, pp.722-736, June 1981.

[19] Leland B. Jackson, *Digital Filters and Signal Processing.* 2nd ed. Norwell: Kluwer Academic Publishers, 1989.

[20] C. Richard Johnson Jr. , *Lectures On Adaptive Parameter Estimation.* Englewood Cliffs: Prentice-Hall, 1988.

[21] C. Richard Johnson Jr., *Adaptive IIR FIltering: Current Results and Open Issues.* IEEE Transactions on Information Theory, vol. IT-30, no.2, pp. 237-249, 1984.

[22] Steven M. Kay, *Modern Spectral Estimation.* Englewood Cliffs: Prentice-Hall Inc., 1988.

[23] Roman Kuc, *Introduction to Digital Signal Processing.* Singapore: McGraw-Hill International Editions, 1988.

[24] Michael G. Larimore, John R. Treichler, and C. Richard Johnson Jr., *SHARF:An Algorithm for Adapting IIR Digital Filters.* IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-28, no. 4, pp. 428-440, August 1980.

[25] Jae Chon Lee and Chong Kwan Un, *Performance Analysis of Frequency-Domain Block LMS Adaptive Digital Filter.* IEEE Transactions on Circuits and Systems, vol. 36, no. 2, February 1989.

[26] Fuyun Ling, Dmitris Manolakis, and John G. Proakis, *Numerically Robust Least-Squares Lattice-Ladder Algorithm with Direct Updating of the Reflection Coefficients.* IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-34, no.4, pp.837-845, August 1986.

[27] John Makhoul, *A Class of All-Zero Lattice Digital Filters: Properties and Applications.* IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-26, no. 4, pp. 304-314, August 1978.

[28] V. John Mathews and Zhenhua Xie, *Fixed-Point Analysis of Stochastic Gradient Adaptive Lattice Filters.* IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 38, no. 3, p. 70-80, January 1990.

[29] Nirode Mohanty, *Random Signals Estimation and Identification.* New York: Van Nostrand Reinhold Company Inc., 1986.

[30] Bernard Mulgrew, and Collin F.N. Cowan, *Adaptive Filters and Equalizers.* Norwell: Kluwer Academic Publishers, 1988.

[31] Richard C. North et al, *A Floating-Point Arithmetic Error Analysis of Direct and Indirect Coefficient Updating Techniques for Adaptive Lattice Filters.* IEEE Transactions on Signal Processing, vol. 41, No.5, pp. 1809-1823, May 1993.

[32] Alan V. Oppenheim and Ronald W.Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs: Prentice-Hall Inc., 1989.

[33] John G. Proakis et al, *Advanced Digital Signal Processing.* New York: Macmillan Publishing Company, 1992.

[34] K. Sam Shanmugan and Arthur M. Breipohl, *Random Signals: Detection, Estimation, and Data Analysis.* New York: John Wiley & Sons, Inc., 1988.

[35] John J. Shynk, *Adaptive IIR Filtering.* IEEE ASSP Magazine, pp. 4-21, April1989.

[36] Dirk T. M. Slock and Thomas Kailath, *Numerically Stable Fast Transversal Filters for Recursive Least Squares Adaptive Filtering.* IEEE Transactions on Signal Processing, vol. 39, no.1, pp. 92-114, January 1991.

[37] M.M. Sondhi, *An Adaptive Echo Canceller.* The Bell System Technical Journal, vol. XLVI, no. 3, pp. 407-511, March 1967.

[38] Chin-Liang Wang and Rong-Yin Chen, *Optimum Design of the Hard-Switching Dual Sign Algorithm for Adaptive Echo Canceller.* IEEE Transactions on Signal Processing, vol. 41, no. 5, May 1993.

[39] Bernard Widrow and Samuel D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs: Prentice-Hall Inc., 1985.

[40] James R. Zeidler, *Performances Analysis of LMS Adaptive Predictions Filters*. Proceedings of the IEEE, vol. 78, no. 12, pp. 1781-1806, 1990

VITA

Gaguk Zakaria was born on August 23rd, 1960 in Gresik, Indonesia. He earned his first university degree in electrical engineering from Institut Teknologi Bandung, in Bandung Indonesia, in 1985.

He then worked in the Product Development department of P.T. Elektrindo Nusantara, a telecommunication company in Indonesia, from 1985 to 1991. At this time he developed some earth station equipment for Indonesia's communication satellite PALAPA. In 1991 he received a scholarship from USAID to continue his education at Virginia Tech, Blacksburg VA, USA, where he is completing his M.S. in Electrical Engineering, and then continuing for a Ph.D.

He is a member of the IEEE Signal Processing, Communications, and Circuits and Systems Societies.