ANALYSES AND APPLICATIONS OF GENERALIZED TRANSFORMED DOMAIN LEAST-MEAN SQUARE (LMS) ADAPTIVE FILTERS

BY

Shiunn-Jang Chern, M. ENG.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

(Engineering)

McMaster University

 September 1986

ISBN 0-315-33444-4

Transformed Domain Adaptive Filters

To My Parents

DOCTOR OF PHILOSOPHY (1986) McMASTER UNIVERSITY
(Engineering) Hamilton, Ontario

TITLE : Analyses and Applications of Generalized Transformed Domain Least-Mean Square (LMS) Adaptive Filters

AUTHOR: Shiunn-Jang Chern, B.E. (Tamkang University, Taiwan)
M.E. (Southeastern Massachusetts University, North Dartmouth, USA)

SUPERVISOR: Professor K. M. Wong

NUMBER OF PAGES: xx , 250

ABSTRACT

In this thesis, a comprehensive analysis of the transformed domain adaptive filtering algorithms is presented. The idea behind this study is to expand the time domain filter impulse response as a set of weighted orthogonal functions. The least-mean square (LMS) adaptation algorithm is applied to the weights. In brief, we systematize and generalize the existing transformed domain algorithms and study their convergence properties.

Based on the concept described above, we develop two types of transformed domain adaptive algorithms, viz, the transformed domain LMS (TDLMS) adaptive algorithm and the block transformed domain LMS (BTDLMS) adaptive algorithm. For the purpose of comparison, time domain expressions of various transformed domain adaptive algorithms are also obtained. We show that the TDLMS algorithm with the Karhunen-Loève (K-L) transform is equivalent to the time domain LMS adaptive algorithm. Whereas, the time domain expression of the TDLMS algorithm with the normalized K-L transform is equivalent to the generalized Newton-Raphson method.

To study the effect of the normalized K-L TDLMS adaptive algorithm, we have derived an analytic expression of mean-square error (MSE) in terms of weight vector, weight covariance matrix, input autocorrelation matrix and the

minimum MSE in the application of adaptive line-enhancer (ALE).An analytic expression of MSE using the time domain LMS algorithm in the ALE will also be obtained for comparison. It will be proved that the convergence rate, in terms of MSE, of the time domain LMS algorithm not only depends on the eigenvalue spread of the input autocorrelation matrix, but also depends highly on the power ratio between the sinusoidal signals in the ALE. However, the convergence rate of the normalized K-L TDLMS algorithm is independent of the eigenvalue spread and power ratio but depends only on the stepsize.

The problem of employing different orthogonal transforms is also studied. The discrete cosine transform (DCT) is used to investigate the effect on convergence rate due to incomplete diagonalization of the autocorrelation matrix. To reduce the computational effort, a recursive equation for evaluating the DCT output is obtained. Furthermore, in order to apply the transmultiplexer transform (TMT) to the transformed domain adaptive algorithms, the simplified form of TMT is also obtained.

Finally, for limited supply of input data, an algorithm of re-circulating the input data in the BTDLMS adaptive algorithm with the discrete Fourier transform (DFT) is proposed. An analysis of the performance, in terms of MSE, of such an algorithm indicates that superior performance can be achieved compared to the algorithm without re-circulation.

ACKNOWLEDGMENTS

I wish to express my most sincere gratitude to my supervisor Professor K. Max Wong for his patience, guidance and tremendous help during the work and preparation of this thesis.

I would like to thank the members of my supervisory committee: Drs : C. R. Carter, N. K. Sinha and P. Yip for their helpful comments and suggestions.

I am also deeply grateful for the financial support provided by the McMaster University and the Department of Electrical and Computer Engineering.

I would also like to acknowledge the inspiring discussion with my colleagues and friends : Drs: J. R. Raol, M. Shafi and Mr. M. V. Dubey.

My special thanks are due to my wife Yuh-Yuann Chern, for her love,patience,support, understanding and innumberable sacrifices. But for her help this research could not have reached this stage. The smiles of my sons Yahn-Bor and Jeffy have helped me to overcome many frustrating moments during the development of this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PRINCIPAL SYMBOLS

a_n	Stepsize parameter in the gradient method
$\underline{A}$	Symmetric matrix defined in (5-19a) and (5-43)
$\underline{b}(n)$	Transformed domain tap-weight vector
$\underline{b}_{opt}$	Optimum value of $\underline{b}(n)$
$\underline{B}$	Column vector defined in (5-22) and (5-46)
$\underline{c}(n)$	Column vector
Cov[.]	Tap-weight covariance matrix
$\underline{C}_s$	Discrete cosine transform matrix
$d(n)$	Desired response signal
$\underline{d}(m)$	Desired response vector at the mth block instant consisting of $d(mN),\ldots,d((m+1)N-1)$ as elements
$\hat{\underline{d}}(m)$	Transformed domain desired response vector
e	Base of natural logarithm
$e(n)$	Error signal at time n
exp	Exponential
E[.]	Expectation operator
$F(\underline{w})$	A scalar functional of $\underline{w}$
$\underline{F}$	Discrete Fourier transform matrix
$\underline{g}$	Negative gradient vector with respect to $\underline{w}$
$\underline{G}(n)$	Square matrix defined as the metric in the gradient method

$\underline{H}$	Square matrix defined in (5-13) and (5-38)
$\underline{I}$	Identity matrix
Im[.]	Imaginary part of the bracket quantity
j	Square root of -1
j	Integer index
$k_o(n)$	Scalar quantity defined in (5-41)
L'	Scalar quantity defined in (2-17a)
L	Number of sinusoidal signals in the received signal
$\underline{M}_{w(n)}$	Mean tap-weight vector at time n
N	Data length
n_{m1}	Noise component in the transformed desired response
n_{m2}	Noise component in the transfomed input signal
p	Number of re-circulations
p_n	Scalar parameter defined in (F-2)
$\underline{p}$	Cross-correlation vector between tap-input vector $\underline{x}(n)$ and desired response signal d(n)
$\underline{P}(m)$	Diagonal matrix at mth block instant consisting of $s_m(0),\ldots,s_m(N-1)$ as elements
$\underline{q}_k$	kth eigenvector of $\underline{R}$
$\underline{Q}$	Karhunen-Loeve transformation matrix
$\underline{Q}$	Tranformation matrix
$\underline{r}(n)$	Equivalent vector of $Cov[\underline{w}(n)\underline{w}^{\ddagger}(n)]$
$\underline{R}$	Autocorrelation matrix of tap-input vector $\underline{x}(n)$

$\mathrm{Re}[.]$	Real part of the bracket quantity
$\underline{R}_\lambda$	Diagonal matrix consisting of the eigenvalues of $\underline{R}$
$\underline{S}$	Similarity transformation matrix of $\underline{A}$
$\underline{s}(n)$	Transformed domain tap-input vector at time n
$\hat{\underline{s}}(m)$	Transformed domain tap-input vector at mth block instant
$S(w)$	Power spectral density
T	Transpose operator
T_s	Sampling time interval
$\underline{T}$	Similarity transformation matrix of $\underline{H}$
$\mathrm{Tr}\{.\}$	Trace of the bracket matrix
t_i	Time constant of ith natural mode of the time domain LMS algorithm
$\underline{v}(m)$	Transformed domain error vector at the mth block instant
$\underline{w}$	A set of independent variable
$\underline{w}(n)$	Time domain tap-weight vector at time n
$w_k(n)$	kth element of $\underline{w}(n)$
$\underline{w}_{opt}$	Optimum value of $\underline{w}(n)$
$\underline{w}^e(n)$	Error weight vector
$x(n)$	Sample value of tap-input in transversal filter at time n
$\underline{x}(n)$	Tap-input vector consisting of $x(n),\ldots,x(n-N+1)$ as elements
$\underline{X}(m)$	Block matrix at the mth block instant

$y(n)$	Output signal of time domain adaptive filter
$\underline{y}(m)$	Output vector at the mth block instant
z^{-1}	Unit-sample delay in defining the Z-transform of a sequence
δ	Kronecker delta
$\underline{\nabla}$	Gradient vector with respect to $\underline{w}$
ξ_n	Index of performance defined as the mean-squared error
ξ_{min}	Minimum value of ξ_n
ξ_∞	Steady-state value of ξ_n
α	Step-size parameter in the gradient method
α_k	kth eigenvalue of $\underline{A}$
$\underline{\alpha}$	Diagonal matrix consisting of $\alpha_1, \ldots$ as elements
μ	Step-size parameter in the time domain LMS algorithm
μ_B	Step-size parameter in the block LMS algorithm
$\tilde{\mu}_k$	Normalized step-size in the frequency domain algorithm
λ_k	kth eigenvalue of $\underline{R}$
λ_L	Largest eigenvalue of $\underline{R}$
λ_S	Smallest eigenvalue of $\underline{R}$
$\underline{\lambda}$	Vector consisting of the eigenvalues of $\underline{R}$
$\underline{\lambda}^{-1}$	vector consisting of the inverse eigenvalues of $\underline{R}$
σ_s^2	Signal power
σ_n^2	Noise power

$\underline{\rho}$	Cross-correlation vector between $\underline{X}(m)$ and $\underline{d}(m)$
$\underline{R}$	Auto-correlation matrix of $\underline{X}(m)$
$*$	Complex conjugate operator
$\ddagger$	Complex conjugate transpose operator
$\circledast$	Convolution operator

CHAPTER 1

INTRODUCTION

1.1 Adaptive Filters

In signal processing two important classes of digital filters nonrecursive and recursive, are extensively used in practice for their frequency-selective properties, smoothing capability and as estimators of signal in the presence of noise.

There are various techniques [1-6] for estimating random signals in the presence of noise. The earlier studies were made independently by Kolmogorov [5] and Wiener [6] on the problem of optimum linear filtering which is known as the Wiener filter. The criterion on which their work was based is known as the least mean-squared (LMS) error criterion. The LMS error criterion requires that the ensemble expectation of the squared difference between the true and the estimate be minimized. To obtain the unknown filter parameters of Wiener filter , we have to then solve sets of algebraic equations simultaneously. This requires inverting a matrix whose order is equal to the number of simultaneous equations involved. Many computational efficient techniques exist for carrying out the necessary matrix inversion. The problem with these techniques is that they rely on a specified number of data samples to be processed. Should new data become available,

the matrix inversion has to be done again , and each successive matrix to be inverted has more rows and columns. One would like a technique in which previously determined estimates are simply updated as new data become available, rather than solving the problem all over again. The recursive estimation is exactly such a scheme which include the estimation of time-varying random signals or random processes . Kalman filtering [1,3] is a very important technique in recursive estimation which requires a priori knowledge about the statistics of signal and / or noise to be processed.

An adaptive filter is another important technique in recursive-estimation which is found useful in a great variety of engineering applications. Such applications include array processing, noise and echo cancellation, line enhancement, time-delay estimation, system identification, and channel equalization in data communication systems [7-13]. The advantage of the adaptive filters lies in the fact that they are extremely robust. That is, they perform remarkably well over a wide range of input signal parameters and statistics with no a priori information about the precise nature of these parameters. The design of an adaptive filter is accomplished in a single process by a recursive algorithm that automatically updates the filter parameters with the arrival of the new data samples.

1.2 Fundamental Structure of an Adaptive Filter

The fundamental study of adaptive filtering is to explore the statistical behaviour of its convergence rate. In order for us to carry out the study of convergence of the adaptive filtering algorithm, it will be useful to review its fundamental structure and some commonly used techniques.

Basically, an adaptive filter consists of two distinct parts: a filter, whose structure is designed to perform a desired processing function, and an adaptation algorithm for adjusting the parameters or coefficients of that filter. Different combinations of filters and adaptation algorithms will lead to a variety of adaptive filters. Since both recursive and nonrecursive filters are the most commonly used in practice, therefore, they are extensively employed in adaptive filtering. On the other hand, the recursive least square (RLS) algorithm and the least mean square (LMS) algorithm are two of the most popular principal adaptation algorithms which have been used in adaptive filtering. Both the LMS algorithm and the RLS algorithm are named direct from the names of error criteria which are used for designing the adaptive filters.

Traditionally, an adaptive filter is implemented as a tapped delay line (TDL) transversal filter or nonrecursive linear finite impulse response (FIR) filter such that the output is the sum of the weighted input samples. The values of the tap weights are adjusted by a recursive algorithm. The most commonly used algorithm being the least mean square (LMS) algorithm because of its simplicity, economy in

computation, and ease in implementation.

The LMS adaptive filter suggested by Widrow-Hoff [14] is based on a simple gradient search algorithm which in turn is based on the standard steepest descent method [15]. The configuration of the LMS adaptive filter is depicted in Figure 1.1. An adaptive filter of the type described above is called a time - domain adaptive filter because the adjustment of its tap weights is based on the sample values in the time domain.

On the other hand, the conventional time-domain LMS adaptive filtering can be carried out by processing the data in a blockwise manner such that the tap-weights are up-dated only after every block of data. This results in computational savings and is referred to as a block LMS (BLMS) adaptive filter. The diagram of BLMS adaptive filter is shown in Figure 1.2. The main computational part of the BLMS adaptive filtering technique is a convolution operation. Efficient implementation of this operation can be realized by using various discrete transforms [16-19]. However, the tap-weight adjustment for minimizing the MSE is performed in the time - domain.

Alternatively, adaptive filtering can also adopt a transformed domain approach, that is, the adjustment of the transformed domain filter weights is based on the transformed sample values [19-21,23].

1.3 Current Trends in Transformed Domain Adaptive Filtering

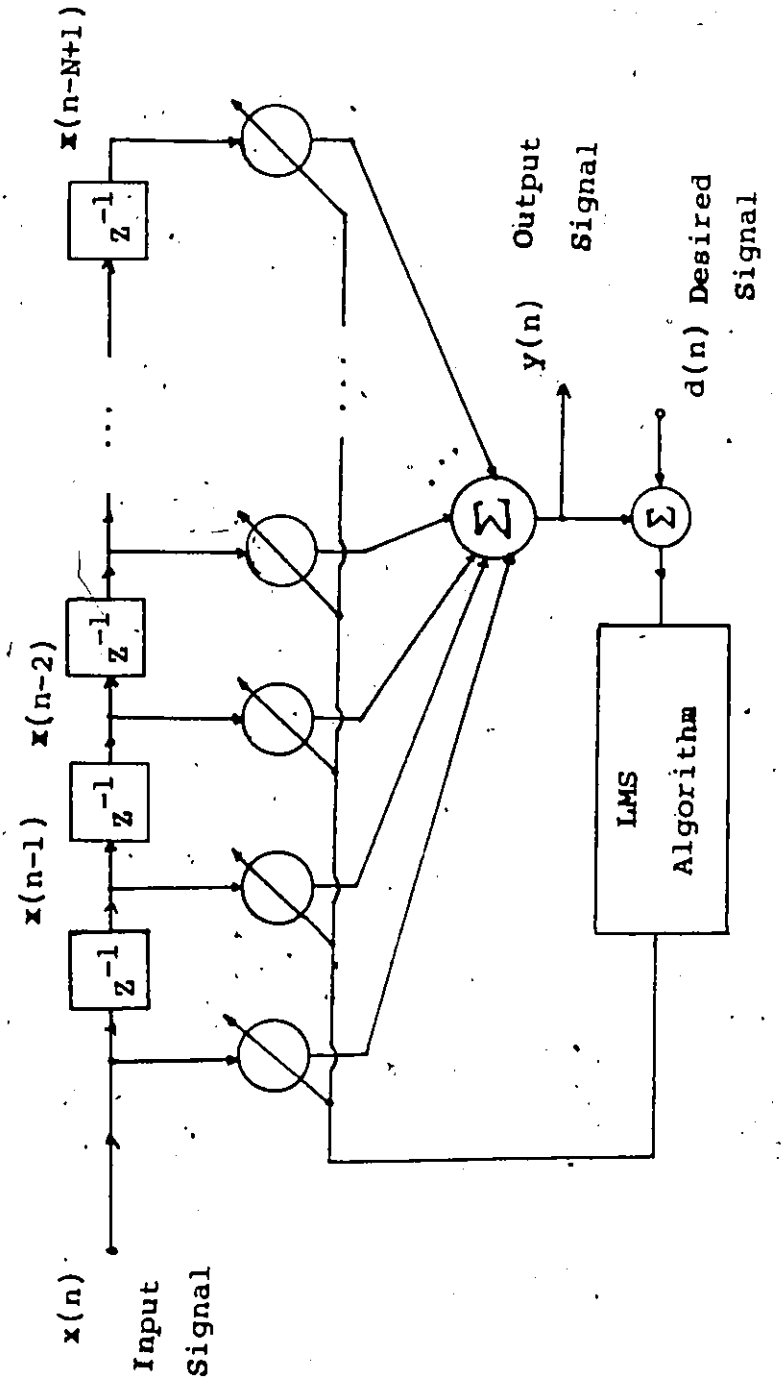


Figure 1.1 The Least Mean-Squared (LMS) Adaptive Filter.

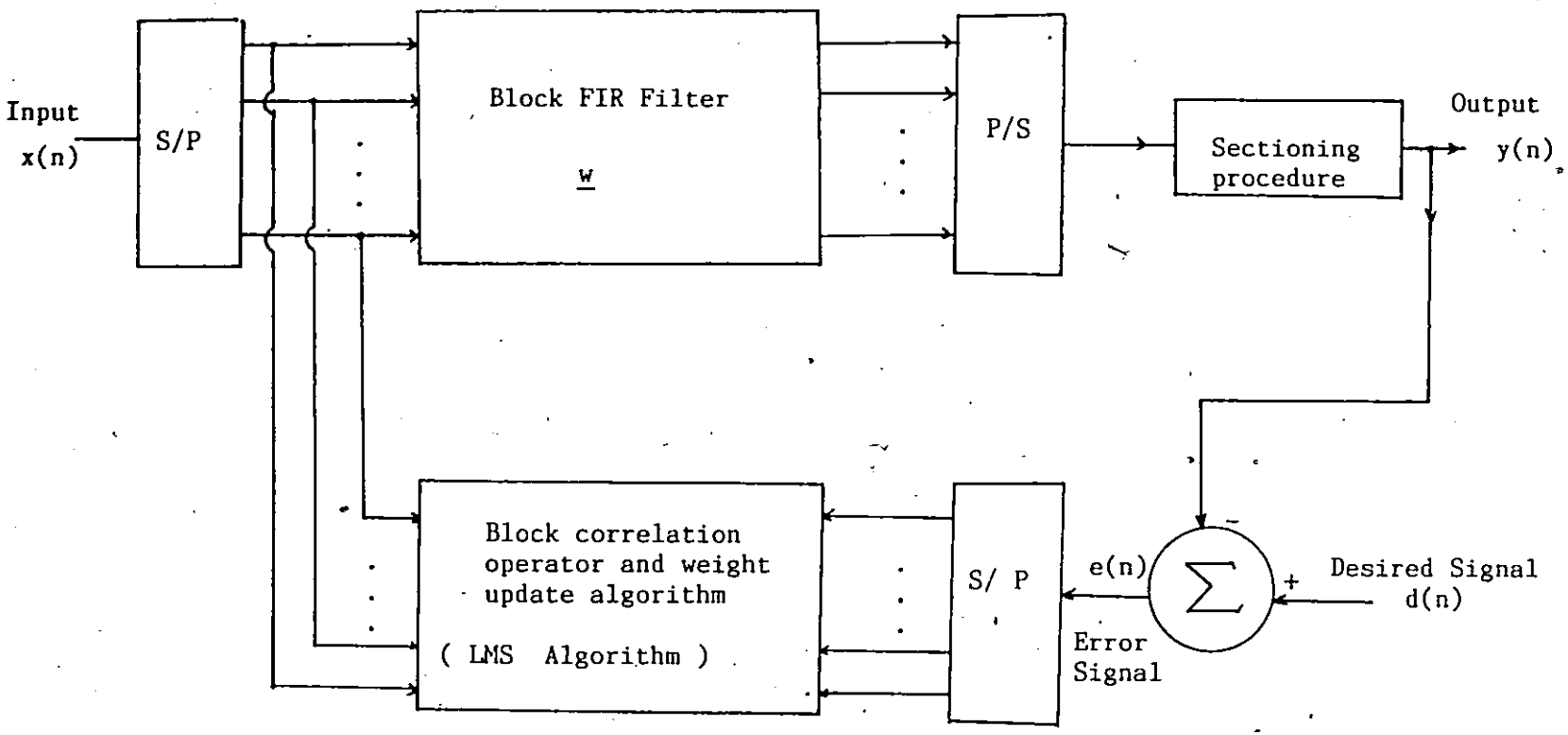


Figure 1.2 The Block Least Mean-Squared (BLMS) Adaptive Filter.

S/P : Serial-to-parallel. P/S : Parallel-to-serial.

Many realizations of the adaptive filter have been proposed in the literature [24-26]. Most of these algorithms addressed the problem of improving the convergence rate and to reduce computational complexity. In general, fast convergence rate is achieved at the expense of increased computation. Consequently, a compromise between these two is necessary. The compromise is highly dependent on the application.

To achieve the goals described above, various transformed domain LMS adaptive filters have been recently proposed [19-23]. Basically, these adaptive filtering algorithms have two major features:

(1) They employ a fast orthogonal transformation algorithm to realize the time domain BLMS adaptation algorithm. This results in reducing the computation complexity.

(2) They employ a fast orthogonal transformation algorithm to represent the time-domain filter weights in a transformed domain. Simultaneously, they utilize the normalized step - size to improve the convergence rate in the case of large disparity in eigenvalues of the input autocorrelation matrix.

In general, the theoretical analysis of the convergence rate of the time domain adaptive filtering algorithms is not tractable . The transformed domain adaptive filtering algorithm, does not suffer from this problem and enables us to gain an insight to other more complicated adaptation processes [28].

Theoretical analysis on adaptive filtering algorithms

can usually reveal the convergence behaviour of the algorithm. Frequently, this may introduce a new algorithm which is a compromise between existing algorithms. However, although, various transformed domain algorithms have been empirically addressed via computer simulations, theoretical analysis on such algorithm have scarcely been carried out. The major difficulty in the theoretical analysis is due to the fact that the algorithm is nonlinear in the observed data.

The objective of this thesis is mainly to incorporate these major features, and to study the statistical behaviour of the transformed domain LMS algorithm and the convergence rate analytically and empirically.

1.4 Generalized Transformed Domain Least-Mean Square (LMS) Adaptive Filtering

In this thesis, a new comprehensive study of the transformed domain LMS adaptive filters is considered. The basic idea behind this study is to expand the time domain filter impulse response as a set of weighted orthogonal functions. Then the LMS adaptation algorithm is applied to the weights. Through this study we can systematize and generalize the existing transformed domain algorithms and then study their convergence properties.

Corresponding to the time domain LMS adaptive filtering algorithms, the transformed domain LMS adaptive filtering

algorithms can be classified as : 1) the transformed domain LMS (TDLMS) algorithms [23,27], and 2) the block transformed domain LMS (BTDLMS) algorithms [20,22]. The difference between the two types of transformed domain LMS adaptive filter is analogous to the difference between the corresponding time domain LMS adaptive filters, i.e., the first type continuously up-dates the tap-weights as the data stream comes in, whereas the second type up-dates the tap - weights only after every block of data.

Basically, the TDLMS adaptive filter is related to the conventional time-domain LMS adaptive filter in the sense that both filters employ the same cost function, i.e., the mean square error (MSE), for the filter design. However, the BTDLMS adaptive filter is similar to the time domain BLMS adaptive filter only in that block processing is employed in both cases. Although MSE is used as the criterion for optimizing the tap-weights, the cost functions are different in the two cases.

To study the transformed domain LMS adaptive filtering algorithms, the Karhunen - Loève (K-L) transform [30] is essential. This is because it can be so chosen that the transformed input autocorrelation matrix is completely diagonalized. Therefore, theoretically, the K-L transform can be used to derive the transformed domain adaptive filter as well as to study the effect of the transformed domain implementation analytically. Furthermore, we introduce the normalized K-L transform to the design of transformed domain

LMS adaptive filtering algorithms. Normalized K-L transform means that the transformed input autocorrelation matrix under such transformation will be an identity matrix. Here, the normalized K-L transform is the K-L transform multiplied by the inverse of the square root diagonal matrix whose entry elements are the eigenvalues of the input autocorrelation matrix.

Since the equivalent time domain expressions for the K-L TDLMS and the normalized K-L TDLMS algorithms can be explicitly obtained, therefore, the analytic study of the convergence properties of these algorithms can be performed. Indeed, the K-L TDLMS algorithm can be shown to be equivalent to the conventional time domain LMS algorithm, while the normalized K-L TDLMS algorithms is equivalent to the stochastic analog of the generalized Newton-Raphson method [35].

However, in real time implementation, the K-L transform is difficult to obtain , therefore , other suboptimal orthogonal transformations are used. In general, the suboptimal orthogonal transformations may not completely diagonalize the input autocorrelation matrix. The effect of incomplete diagonalization of the input autocorrelation matrix is that the convergence rate is slower.

1.5 Scope of Thesis

This thesis is divided into eight chapters. Chapter 2 is mainly tutorial in nature, whereas the main contributions of this thesis appear in chapters 3, 4, 5, 6, and 7.

In chapter 2, the fundamental characteristics of the gradient methods and the time domain LMS algorithms are reviewed. The interrelation between these algorithms are emphasized.

The mathematical formulation of the generalized orthogonal transformed domain adaptive filter is considered in chapter 3. Also, various equivalent time domain expressions of the K-L transformed domain and the normalized K-L transformed domain for both the TDLMS and the BTDLMS adaptive filtering algorithms are derived. These results are used for further study in the subsequent chapters.

In chapter 4, the statistical analysis of the normalized K-L transform algorithm in time-domain is performed. The recursive equations for the mean weight vector and the weight covariance matrix are derived for both real and complex data. These are used to explicitly express the steady - state and/ or transient mean square errors.

In chapter 5, the configuration of the adaptive-line enhancer (ALE) is also introduced. The comparison for both time domain LMS ALE and the normalized K-L transform algorithm in time domain suggests that a hybrid algorithm for combining both algorithms is necessary to obtain superior convergence property.

In chapter 6, the problem of employing various

suboptimal orthogonal transformations to the generalized transformed domain LMS adaptive algorithm is addressed. The effect of the convergence rate due to incomplete diagonalization is investigated via the discrete cosine transform (DCT) for stationary, first-order Markov process. To circumvent the increase in computational complexity in the TDLMS algorithm compared to the time domain LMS algorithm, a recursive equation for the DCT of signal is derived.

In chapter 7, the re-circulating data frequency domain adaptive algorithm is addressed and analyzed. A closed form expression, for the single complex weight in the re-circulation algorithm is derived,which allows the statistical analysis to be performed.

Finally, in chapter 8, suggestions for future investigation in the problem of transformed domain implementation of the adaptive algorithms are presented and conclusions are given.

CHAPTER 2

CONVENTIONAL GRADIENT METHODS AND LMS ADAPTIVE ALGORITHMS

2.1 Introduction

The purpose of the present chapter is to review various minimization algorithms. This is very helpful for developing the generalized orthogonal transformed domain LMS adaptive filters and to further investigate their statistical behavior in terms of the convergence rate and the mean-square error (MSE). In the subsequent sections, the conventional gradient methods, the time domain LMS and the block LMS (BLMS) adaptive algorithms are thoroughly discussed. The inter-relation among these algorithms are then emphasized.

One of the most frequently used ideas in numerical computation is iteration or successive approximation. In general, iteration means the repetition of a pattern of processes. In the repeated application of a numerical process the use of iteration is to successively improve previous results.

Gradient methods are techniques used mostly in optimization problems to approximate the optimal solution by iteration. All iterative gradient methods for locating the minima of functions consist of calculating $\nabla F(\underline{w})$, the gradient of $F(\underline{w})$, for various $\underline{w}$ in an effort to locate those

values of $\underline{w}$ for which $\underline{\nabla} F(\underline{w})=0$. The second derivatives of $F(\underline{w})$, the Hessian matrix, should be positive definite. Here, $\underline{\nabla}$ is a vector and is defined as,

$$\underline{\nabla} = \frac{\partial}{\partial \underline{w}} ,$$

$F(\underline{w})$ is a scalar functional of $\underline{w}$, and $\underline{w}$ is a set of independent variables.

In general, all of the gradient descent methods can be described by a basic iterative equation. Different approaches of the gradient descent methods can be chosen depending on the selection of the parameters of the basic iterative equation.

The method of steepest descent is one of the oldest and most commonly used methods for obtaining a minimum point of the cost function. There are two classes of steepest descent minimization viz, the standard steepest descent and the generalized Newton-Raphson methods. Both of these descent minimization algorithms will be discussed in the subsequent sections. It is widely known [34,35], that the method of standard steepest descent is very useful for a large class of well-conditioned problems (i.e., the eigenvalue spread is close to unity). The eigenvalue spread is the ratio of the largest and the smallest eigenvalues of the second derivatives of the cost function. It is also reported [34,35], that the convergence rate of this method can be extremely slow. However, it will surely converge to a minimum. On the other hand, the generalized Newton-Raphson method is ideal from the standpoint of speed of convergence.

However, the generalized Newton-Raphson method is applicable only to the final portion of the descent iteration, where the function is positive definite [35].

The conventional time domain LMS adaptive algorithm was suggested by Widrow and Hoff [14]. The algorithm is a stochastic analog of the standard steepest descent minimization procedure. In the LMS adaptation algorithm, estimate gradient is used instead of the true gradient which is in the standard steepest descent method. The estimated gradient is sequentially estimated as a function of the current estimated error. The LMS algorithm is described by a stochastic vector recursive equation which is linear in the weight and nonlinear in the observed data. The weight vector sequence generated by this equation is a stochastic process. The convergence rate of the LMS adaptive algorithm can be expected to perform similar to the standard steepest descent method. This is due to the fact that the gradient estimate is an unbiased estimate of the true gradient [14]. In addition, the implementation of time domain LMS adaptive algorithm involves only the input and desired signals, and hence it becomes very attractive in real time applications.

The block LMS adaptive filtering algorithm is the realization of the time domain LMS adaptive filtering algorithm by blockwise processing of the data. This results in computational advantage over the time domain LMS algorithm. In order to use the efficient block procedures, an adaptive algorithm must allow a whole block of outputs to

be calculated without modifying the filter parameters. It was shown in [16], that the time domain LMS adaptive filter, which adjusts parameters once each sample, is a special case of the block adaptive filter with a block length of one. Therefore, under some circumstances the convergence rate in terms of the MSE is very close to the conventional time domain LMS algorithm but requires less computational effort [16].

2.2 Performance Measure and the Wiener Solution

As mentioned in Chapter 1, an adaptive filter consists of two distinct parts, a filter and an adaptation algorithm. The adaptive filter can be viewed as a system (a filter itself) which provides a suitable choice (an adaptation algorithm) of the weight vector $\underline{w}(n)$ based upon a " training sample " of previous observations of input vectors, $\underline{x}(k)$, and desired signals, $d(k)$, for $0 \leq k < n$. The basic component of most adaptive filtering and signal processing systems is the adaptive linear combiner [14] which is depicted in Figure 2.1. Here the output is the weighted sum of the input samples.

The nature of the performance surface over which the adaptive processor must operate is determined by the selected performance measure and the parameters to be adjusted. Furthermore, the performance surface is determined by the nature of $F(\underline{w})$, where $F(\cdot)$ is the performance measure and $\underline{w}$

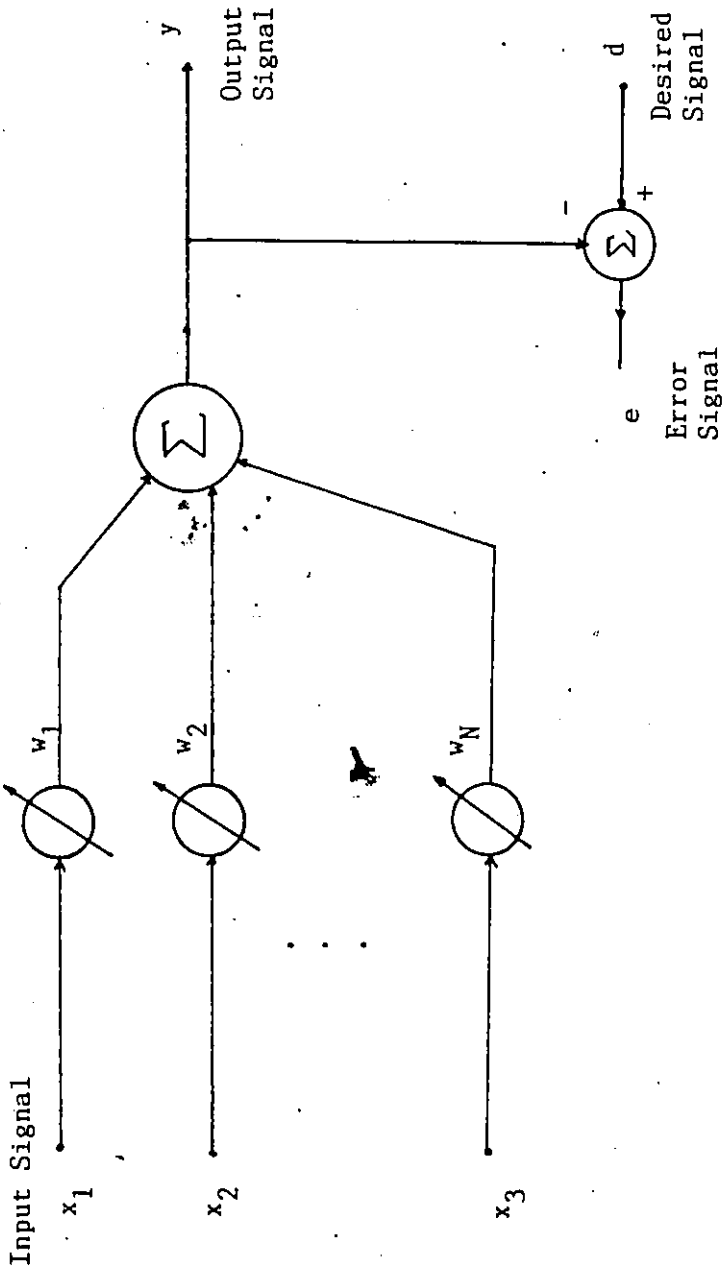


Figure 2.1 The Adaptive Linear Combiner.

is the weight vector (real or complex) whose components are desired to be adjusted.

The performance measure used in this thesis is the mean-square error (MSE) $E[|e(n)|^2]$, where e(n) is the estimation error, given by

$$e(n) = d(n) - \underline{w}^T(n)\,\underline{x}(n) \qquad (2\text{-}1)$$

and E{.} means statistical expectation operator. The superscript T stands for the transpose of a vector/matrix. If $\underline{x}(n)$ and d(n) are assumed to be real jointly stationary, then the cost function, $F(\underline{w})$, or the MSE can be written as

$$F(\underline{w}) = E\,[|e(n)|^2]$$
$$= \underline{w}^T\underline{R}\,\underline{w} - 2\,\underline{p}^T\underline{w} + c \qquad (2\text{-}2)$$

where $\underline{R}$ is the input autocorrelation matrix, $\underline{p}$ is the crosscorrelation vector of d(n) and $\underline{x}(n)$, and c is the autocorrelation value of the desired signal and are defined as :

$$c = E\,[\,d^2(n)\,],$$
$$\underline{p} = E\,[\,d(n)\,\underline{x}(n)\,],$$

and

$$\underline{R} = E\,[\underline{x}(n)\,\underline{x}^T(n)\,] \qquad (2\text{-}3)$$

Note that, in (2-2) the performance measure, $F(\underline{w})$ is a quadratic function of $\underline{w}$. Consequently, the level surfaces of the quadratic function in (2-2) are concentric ellipsoids in N dimensions. The N axes of the ellipsoids are each collinear with one of the eigenvectors of the matrix $\underline{R}$. The lengths of the axes of each ellipsoid are inversely proportional to the square roots of the corresponding eigenvalues. In the case

N=2, we have the two-dimensional ellipsoid which is depicted in Figure 2.2.

Assume that the inverse of $\underline{R}$, $\underline{R}^{-1}$, exists. Then the minimum of $F(\underline{w})$ is found as a solution of the linear equation [14,34,35],i.e.,

$$\underline{\nabla} F(\underline{w}) = 2 (\underline{R}\,\underline{w} - \underline{p}) = \underline{o} \qquad (2\text{-}4a)$$

or

$$\underline{w}_{opt} = \underline{R}^{-1}\,\underline{p} \qquad (2\text{-}4b)$$

Equation (2-4b) is the well-known Wiener solution or the optimal solution. Here $\underline{\nabla}$ stands for the gradient operator. Substituting (2-4b) into (2-2), we have the minimum MSE (MMSE), ξ_{min}, given by

$$\begin{aligned} \xi_{min} &= F(\underline{w}_{opt}) \\ &= E[\, d^2(n) \,] - \underline{w}^T_{opt}\,\underline{p} \end{aligned} \qquad (2\text{-}5)$$

2.3 Conventional Gradient Methods

Since in (2-2), the performance measure, $F(\underline{w})$, is a quadratic function of $\underline{w}$, then the performance measure can be visualized as a bowl-shaped surface and the task of the adaptive processor is to seek the " bottom of the bowl" iteratively. There are many ways to achieve the minimization of the performance measure. The techniques in which we are interested in the present section are called the descent methods and are the representative of various gradient methods. By " descent " method we mean that the value of $F(\underline{w})$ at n+1 st iteration, $F(\underline{w}(n+1))$, is less than the value

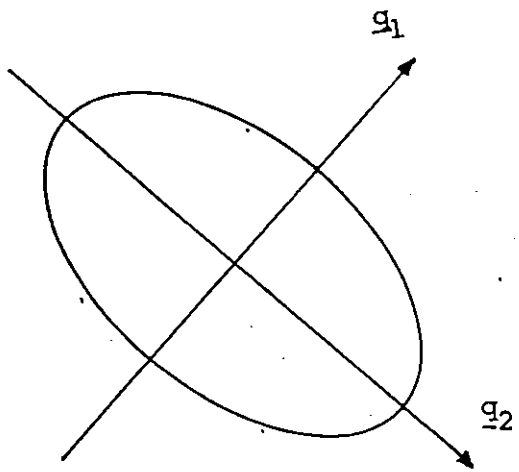


Figure 2.2 Two-dimensional Ellipsoid (N=2) Level Surface.

at n th iteration, $F(\underline{w}(n))$. Given an initial set of the weights of the filter, the gradient methods can be used to iteratively approach the MMSE.

As mentioned in the previous section, the gradient or descent methods use the basic iterative equation:

$$\underline{w}(n+1) = \underline{w}(n) + \alpha \ \underline{G}(n) \underline{\nabla} F(\underline{w}(n)) \qquad (2\text{-}6a)$$

or

$$\underline{w}(n+1) = \underline{w}(n) + \alpha \ \underline{r}(n) \qquad (2\text{-}6b)$$

where

$$\underline{r}(n) = \underline{G}(n) \underline{\nabla} F(\underline{w}(n)) \qquad (2\text{-}7)$$

is the direction vector. The matrix $\underline{G}(n)$, which is called the metric, establishes the dependence of the direction vector, $\underline{r}(n)$, on $\underline{\nabla} F(\underline{w}(n))$. The various techniques depend on the choice of the metric, $\underline{G}(n)$.

If at the nth iteration, a correction $\underline{w}(n+1)$ is required for $\underline{w}(n)$, such that (2-6) is satisfied, the direction of search $\underline{r}(n)$ is chosen to satisfy the following condition:

$$F(\underline{w}(n+1)) < F(\underline{w}(n)) \qquad (2\text{-}8)$$

Note that, we call a sequence $\{\underline{w}(n)\}$, a descending sequence, if for each n (2-8) holds. Further, if we compute the direction vector, $\underline{r}(n)$ at each iteration such that (2-8) is satisfied, it seems likely that the sequence $\{\underline{w}(n)\}$ will converge to the local minimum of $F(\underline{w})$.

The variable stepsize α, can be chosen such that $F(\underline{w}(n) + a_n \underline{r}(n))$ is minimum for $\alpha = a_n$. Since $F(\underline{w})$ is a quadratic function of $\underline{w}$, we have the following equality:

$$F(\underline{w} + \underline{v}) = F(\underline{w}) - \underline{v}^T \underline{g} + \underline{v}^T \underline{R}\, \underline{v} \qquad (2\text{-}9)$$

where $\underline{g}$ is the negative gradient and $\underline{v}$ is the error vector. From (2-4a), the vector $\underline{g}$ is given by

$$\underline{g} = -\underline{\nabla} F(\underline{w})$$
$$= 2 (\underline{p} - \underline{R}\, \underline{w})$$

or

$$\underline{g}(n+1) = 2 (\underline{p} - \underline{R}\, \underline{w}(n+1))$$
$$= \underline{g}(n) - 2\, a_n\, \underline{R}\, \underline{r}(n) \qquad (2\text{-}10)$$

where $\underline{g}(n)$ is the negative gradient of $F(\underline{w})$ at $\underline{w}(n)$. Now using (2-6) and (2-7) in (2-9), we have

$$F(\underline{w}(n+1)) = F(\underline{w}(n) + \alpha\, \underline{r}(n))$$
$$= F(\underline{w}(n)) + a_n \underline{r}(n))$$
$$= F(\underline{w}(n)) - a_n \underline{r}^T(n) \underline{g}(n)$$
$$+ a_n^2\, \underline{r}^T(n)\, \underline{R}\, \underline{r}(n) \qquad (2\text{-}11a)$$

In view of (2-11a) the point $\underline{w}(n+1)$ minimizes $F(\underline{w})$ on the line $\underline{w} = \underline{w}(n) + a_n\, \underline{r}(n)$. It can now be seen that the negative gradient $\underline{g}(n+1)$ of $F(\underline{w})$ at $\underline{w}(n+1)$ is therefore orthogonal to the direction $\underline{r}(n)$ of this line. Hence we have

$$0 = \underline{r}^T(n)\, \underline{g}(n+1)$$
$$= \underline{r}^T(n) [\, \underline{g}(n) - 2\, a_n\, \underline{R}\, \underline{r}(n)]$$
$$= \underline{r}^T(n)\, \underline{g}(n) - 2\, a_n\, \underline{r}^T(n)\, \underline{R}\, \underline{r}(n) \qquad (2\text{-}12a)$$

or

$$a_n = \frac{< \underline{r}(n),\, \underline{g}(n) >}{2 < \underline{r}(n),\, \underline{R}\, \underline{r}(n) >} \qquad (2\text{-}12b)$$

Note that the variable stepsize defined in (2-12b) is chosen to be a positive value. Now, applying this result to (2-11a),

we have

$$F(\underline{w}(n+1)) = F(\underline{w}(n)) - a_n \underline{r}^T(n)\, \underline{g}(n) + a^2(n)\, \underline{r}^T(n)\, \underline{R}\, \underline{r}(n)$$

or

$$F(\underline{w}(n+1)) = F(\underline{w}(n)) - \frac{1}{2} a_n \underline{r}^T(n)\, \underline{g}(n).$$

$$= F(\underline{w}(n)) - \frac{|< \underline{r}(n), \underline{g}(n) >|^2}{4<\underline{r}(n), \underline{R}\, \underline{r}(n)>} \qquad (2\text{-}11b)$$

The second term on the right hand side of (2-11b) is negative provided that the matrix $\underline{R}$ is a positive definite and symmetric, and hence (2-11b) satisfies the condition of (2-8), i.e.,

$$F(\underline{w}(n+1)) < F(\underline{w}(n)) \qquad (2\text{-}8)$$

Note that, in (2-12a) and (2-11b) $< , >$ indicates the inner product, i.e.,

$$< \underline{a} , \underline{b} > = \underline{a}^T \underline{b} \qquad (2\text{-}13)$$

where $\underline{a}$ and $\underline{b}$ are column vectors. Also, in (2-11b), $| . |$ indicates the absolute value .

In the following sections two of most popular principal methods, viz, the standard steepest descent method and the generalized Newton-Raphson method are considered.

2.3.1 The Standard Steepest Descent Methods

In (2-11b) $\underline{r}(n)$ only has to satisfy (2-8), and thus its choice is not unique. Thus we can simply choose $\underline{r}(n)$ to

be the downhill gradient vector or the steepest descent:

$$\underline{r}(n) = \underline{g}(n) \qquad (2\text{-}14)$$

The direction is chosen in the direction of steepest descent (negative gradient) and the method is then called the " standard steepest descent " method. Note that, in this case, the metric, $\underline{G}(n)$, is an identity matrix, $\underline{I}$. Now, begining with the initial guess $\underline{w}(0)$, the iterative equation of this method can be obtained by applying (2-14) into (2-6b) which is given as

$$\underline{w}(n+1) = \underline{w}(n) + a_n \underline{g}(n) \qquad (2\text{-}15)$$

This method was originally proposed by Cauchy in 1847 for the solution of systems of nonlinear equations. The advantage of this method is that the method usually converges successfully even if the initial vector $\underline{w}(0)$ is far from the optimum weight vector.

Two different procedures are available for setting a_n. In the first, a_n is simply held constant for all steps The alternative to this is to select a new value for a_n that is proportional to the magnitude of the gradient vector at each step. This can be obtained by applying (2-14) into (2-12b) which is

$$a_n = \frac{\langle \underline{g}(n), \underline{g}(n) \rangle}{2 \langle \underline{g}(n), \underline{R}\, \underline{g}(n) \rangle} \qquad (2\text{-}16)$$

Note that the determination of the best stepsize described in (2-16) is sometimes called the "one dimensional minimization" or the " line search " method [26,27]. This latter method,

shown in (2-16), tends to take a big step on steep surfaces and a small step on relatively flat surfaces. It is possible to "overshoot" the extremum in such a manner that computation will continue to oscillate about the extremum point [35].

The constant-step method, on the other hand, may therefore be the safer of the two methods, although it may require more function evaluations than the variable-step method to achieve the desired performances. Specifically, if we were to use very small value for the multipliers $\{a_n\}$, the sequence $\{\underline{w}(n)\}$ would closely follow the true path of steepest descent. However, the use of very small stepsize would result in a very slow crawl along the descent path and require an extremely large number of evaluations of $F(\underline{w}(n))$.

The convergence rate of the standard steepest descent method can be measured by comparing the estimated weight vector with the optimal weight vector for the subsequent iterative estimate. Here, the difference for both weight vectors described above is referred to as an error weight vector. Kantorovich [10,35] has shown that the error weight vector in the normed space in the standard steepest descent method with (2-15) and (2-16), obeys

$$|| \underline{w}(n+1) - \underline{w}_{opt} || < \frac{\lambda_L - \lambda_S}{\lambda_S + \lambda_L} || \underline{w}(n) - \underline{w}_{opt} || \qquad (2\text{-}17)$$

for n=1,2, ..., where λ_L and λ_S are the largest and smallest eigenvalues of $\underline{R}$. Here $||.||$ is the Euclidean norm of vector and is defined by

$$|| \underline{w} || = (\underline{w}^T \underline{w})^{1/2}$$

On the other hand, the convergence rate can be measured in terms of the performance measure, i.e., $F(\underline{w})$, for the nth and n+1 st iteration which has the following relation [34]

$$F(\underline{w}(n+1)) - F(\underline{w}_{opt}) \leq L' \left(F(\underline{w}(n)) - F(\underline{w}_{opt}) \right)$$

with

$$L' = \left(\frac{\lambda_L - \lambda_S}{\lambda_L + \lambda_S}\right)^2 \qquad (2\text{-}17a)$$

(2-17) means that the error weight vector converges:

$$\underline{w}(n) \longrightarrow \underline{w}_{opt}$$

linearly with constant value $\sqrt{L'}$. By (2-17a) the performance measure at nth iteration converges to the optimal weight:

$$F(\underline{w}(n)) \longrightarrow F(\underline{w}_{opt})$$

linearly with constant value, L'. Indeed, both measures are related to each other because the performance measure is the quadratic function of the weight vector, $\underline{w}$. In general, from (2-17), we can expect that the rate of convergence for the standard steepest descent method to be slow when the ratio of the largest and smallest eigenvalues is large (slightly ill-condition). It is worth noting that in Chapter 5, we will show that this may not be true all the time. The rate of convergence also depends on other factors, not only on the ratio of the largest and smallest eigenvalues of $\underline{R}$. Furthermore, in pratice, the standard steepest descent method usually improves $F(\underline{w}(n))$ rapidly on the first few iterations and give rise to oscillatory progress and becomes unsatisfactory. The reason was pointed out in [37] and is primarily due to the fact that the search directions generated are not linearly independent. The

functions for which the steepest descent method properly works are those with spherical contours.

2.3.2 Generalized Newton- Raphson Method

Due to the poor convergence of the standard steepest descent method in the case of the level surfaces being long and thin ellipsoids (i.e., if the eigenvalues of $\underline{R}$ differ significantly), other methods are required to improve the convergence rate. One of the methods which can be used to improve the convergence rate is addressed in what follows.

The second descent method, the generalized Newton - Raphson method, addressed in this section, is one of the most successful methods used by authors for unconstrained optimization problems [34,35]. This method belongs to a class of gradient methods called the variable metric method [38]. The name of the method, variable metric method, reflects the fact that $\underline{G}(n)$, the metric, is changed after each iteration which in the steepest descent method is an identity matrix, $\underline{I}_{NxN}$.

In the generalized Newton-Raphson method, the metric is chosen as

$$\underline{G}(n) = \underline{R}^{-1}_{w(n)} \qquad (2\text{-}18)$$

where $\underline{R}_{w(n)}$ is the Hessian matrix of $F(\underline{w}(n))$. If the input data to be processed is a stationary sequence then we have

$$\underline{R}_{w(n)} = \underline{R}$$

The iterative equation is then given by

$$\underline{w}(n+1) = \underline{w}(n) + a_n \underline{R}^{-1}_{w(n)} \underline{g}(n) \qquad (2\text{-}19)$$

where $a_n > 0$ is chosen to minimize $F(\underline{w})$ along $\underline{R}^{-1}_{w(n)} \underline{g}(n)$, starting from $\underline{w}(n)$. With $a_n = 1$, this is called Newton - Raphson method. The primary advantage of the Newton-Raphson method compared with other methods, is its relatively rapid rate of convergence once a good approximate solution $\underline{w}(0)$ has been obtained. In some literatures, the generalized Newton - Raphson method is named as the best-step Newton descent and the best-steepest Newton descent as well [34,35].

The generalized Newton-Raphson descent method provides the fastest convergence of all practical minimization techniques once $\underline{w}(n)$ has approached $\underline{w}_{opt}$, the optimum solution. If $\underline{w}(n)$ is not close to $\underline{w}_{opt}$, the inverse of Hessian may not be positive definite, and the Newton step direction:

$$\underline{r}(n) = \underline{R}^{-1}_{w(n)} \underline{g}(n) \qquad (2\text{-}20)$$

may not even be a descent direction; that is the iteration may not converge [35]. In fact, the amount of computation per iteration can be high for the generalized Newton-Raphson method because we require to compute second derivative of the cost function with respect to the component of $\underline{w}(n)$.

To explore the convergence rate of the generalized Newton-Raphson method and to see the relationship with the standard steepest descent method for a quadratic function, $F(\underline{w})$, we can rescale or represent the weight vector, $\underline{w}$, such that:

$$\underline{w} = \hat{\underline{Q}}\,\underline{b} \qquad (2\text{-}21)$$

Here $\hat{\underline{Q}}$ is an N by N symmetric and positive definite matrix and $\underline{b}$ is a new weight vector or rescaled weight vector. Substituting (2-21) into (2-2) and defining a functional $\hat{F}(\underline{b})$, we have

$$\begin{aligned}\hat{F}(\underline{b}) &= F(\underline{w}) \\ &= (\hat{\underline{Q}}\,\underline{b})^T\,\underline{R}\,(\hat{\underline{Q}}\,\underline{b}) - 2\,\underline{p}^T\,(\hat{\underline{Q}}\,\underline{b}) + c \\ &= \underline{b}^T\,\hat{\underline{Q}}^T\,\underline{R}\,\hat{\underline{Q}}\,\underline{b} - 2\,\underline{p}^T\hat{\underline{Q}}\,\underline{b} + c \qquad (2\text{-}22)\end{aligned}$$

If we define $\underline{R}^{\#}$, as

$$\underline{R}^{\#} = \hat{\underline{Q}}^T\,\underline{R}\,\hat{\underline{Q}} \qquad (2\text{-}23)$$

then (2-22) can be rewritten as

$$\hat{F}(\underline{b}) = \underline{b}^T\,\underline{R}^{\#}\,\underline{b} - 2\,\underline{p}^T\,\hat{\underline{Q}}\,\underline{b} + c \qquad (2\text{-}24)$$

Again, $\hat{F}(\underline{b})$ is a quadratic functional of $\underline{b}$. It can be easily seen that if the matrix $\hat{\underline{Q}}$ is chosen such that the matrix $\underline{R}^{\#}$ is an identity matrix, i.e.,

$$\begin{aligned}\underline{R}^{\#} &= \hat{\underline{Q}}^T\,\underline{R}\,\hat{\underline{Q}} \\ &= \underline{I}_{NxN} \qquad (2\text{-}25a)\end{aligned}$$

or

$$\underline{R} = (\hat{\underline{Q}}^T)^{-1}\,\hat{\underline{Q}}^{-1} \qquad (2\text{-}25b)$$

then applying the standard steepest descent method for the quadratic function, $\hat{F}(\underline{b})$, from (2-17) the rate of convergence will be an one-step convergence [35,36]. This is because the constant value L' in (2-17a) becomes zero. In fact, in this case the smallest and the largest eigenvalues of $\underline{R}^{\#}$, are identical. Indeed, the iterative equation of the rescaled version standard steepest descent method is equivalent to the generalized Newton-Raphson method (Appendix A). Consequently,

the generalized Newton-Raphson method converges in one step theoretically. This suggests that by rescaling the original weight vector the convergence rate can be improved significantly. This is because the level surfaces are now spherical, and the standard steepest method will work properly.

The idea of rescaling the weight vector can be extended to orthogonal transformed domain adaptive filtering problems, and is the main topic of investigation in the ensuing chapters.

2.4 The Conventional Time Domain LMS Adaptive Algorithms

The conventional time domain LMS adaptive filter [14] is shown in Figure 2.3. The filter structure used in the LMS adaptive filters is the transversal tapped-delay line filter and the adaptive algorithm is called the least-mean square (LMS) algorithm. In the time domain LMS adaptive filter, the LMS adaptive algorithm is used to automatically seek an optimal impulse response by adjusting the weights as the data streams comes in.

The desired output signal d(n) can be viewed as the linear combination of the last N samples of the input signal vector, y(n), plus an observation noise e(n), i.e.,

$$d(n) = y(n) + e(n) \qquad (2\text{-}26)$$

The filter output, y(n), is given by

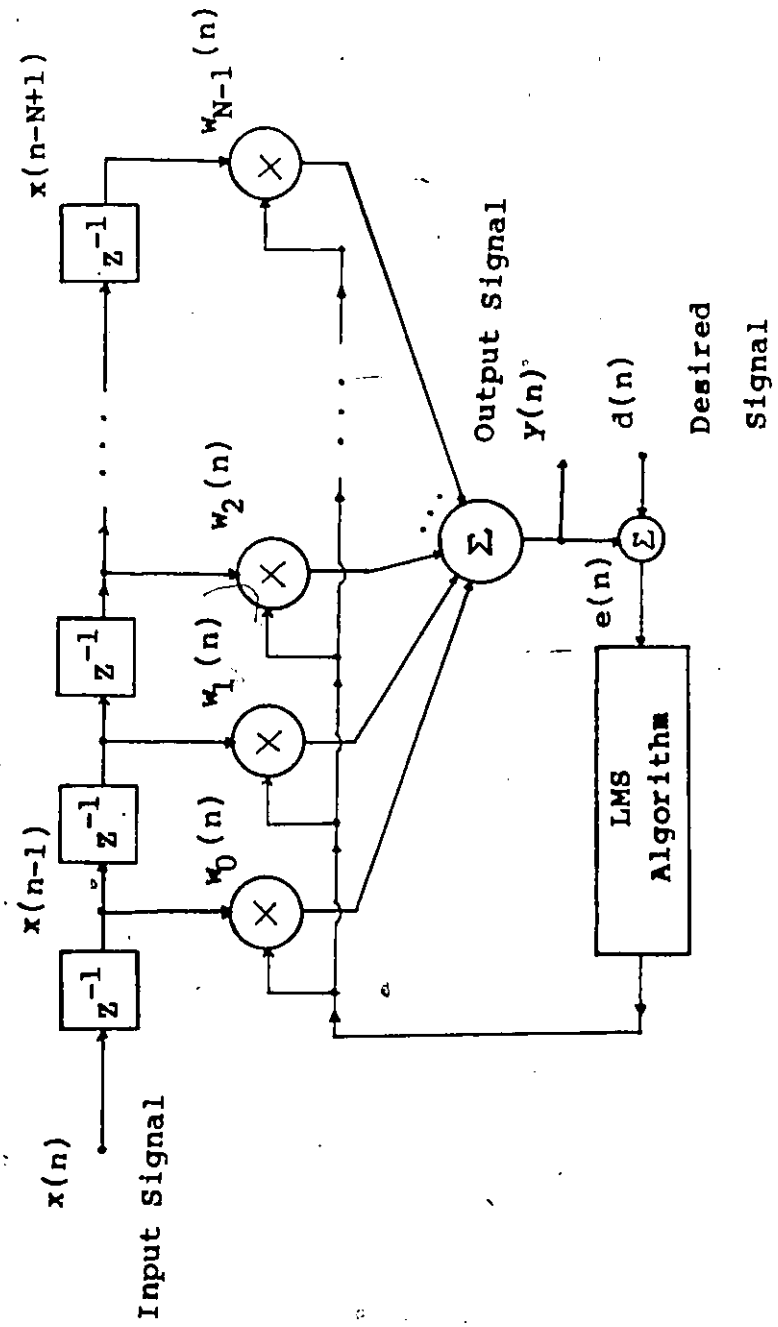


Figure 2.3 The Transversal Tapped-delay Line
LMS Adaptive Filter

$$y(n) = \underline{x}^T(n)\,\underline{w}(n)$$
$$= \underline{w}^T(n)\,\underline{x}(n) \qquad (2\text{-}27)$$

where $\underline{w}(n)$ and $\underline{x}(n)$ are vectors representing, the filter weights and the outputs of the tapped delay line at sampling index n :

$$\underline{w}^T(n) = [w_1(n),\ w_2(n),\ldots,\ w_N(n)] \qquad (2\text{-}28)$$

$$\underline{x}^T(n) = [x(n),\ x(n-1),\ \ldots,\ x(n-N+1)] \qquad (2\text{-}29)$$

From (2-27), the observation error is given as

$$e(n) = d(n) - y(n) \qquad (2\text{-}27a)$$

The LMS adaptive algorithm is then used iteratively to minimize the mean square error of (2-27a).

Recall from (2-15), the standard steepest descent iterative equation , i.e.,

$$\underline{w}(n+1) = \underline{w}(n) + a_n\,\underline{g}(n) \qquad (2\text{-}15)$$

with

$$\underline{g}(n) = -\,\underline{\nabla}\,F(\underline{w}(n))$$
$$= -\,2\,(\underline{R}\,\underline{w}\,(n) - \underline{p}\,) \qquad (2\text{-}2)$$

Substituting (2-2) into (2-15) and choosing a_n to be a positive constant value, μ, we have the simple steepest descent iterative equation, i.e.,

$$\underline{w}(n+1) = \underline{w}(n) + 2\,\mu\,[\,\underline{p} - \underline{R}\,\underline{w}(n)\,] \qquad (2\text{-}30)$$

Further, if we use the estimated gradient in place of the true gradient, (2-30) becomes as [14]

$$\underline{w}(n+1) = \underline{w}(n) + 2\mu[d(n)\underline{x}(n) - \underline{x}(n)\underline{x}^T(n)\underline{w}(n)] \qquad (2\text{-}31)$$

where (2-31) is the LMS adaptive filtering algorithm. Again, by substituting (2-26) and (2-29) into (2-31) we have

$$\underline{w}(n+1) = \underline{w}(n) + 2\,\mu\,e(n)\,\underline{x}(n) \qquad (2\text{-}32)$$

Here μ, the constant stepsize, governs the rate of convergence and the stability of the adaptation process. It has been shown in [39] that to ensure the convergence of the time domain LMS adaptive algorithm, the stepsize, μ, obeys

$$0 < \mu \leq \frac{1}{3\ tr\{\underline{R}\}} \qquad (2\text{-}33)$$

where tr{.} stands for the trace of the matrix and tr{ $\underline{R}$ }, by definition, is the total average input signal power which can be estimated from the received signal. Note that, (2-32) is the LMS adaptive algorithm with real input signals. There is also a complex LMS adaptive algorithm in the time domain. In the complex LMS adaptive algorithm we deal with complex input signals. The iterative equation is basicly the same as the real value LMS algorithm except that the input vector, $\underline{x}(n)$, in (2-32) is substituted by the complex conjugate of $\underline{x}(n)$, $\underline{x}^*(n)$, i.e., [40]

$$\underline{w}(n+1) = \underline{w}(n) + 2\ \mu\ e(n)\ \underline{x}^*(n) \qquad (2\text{-}32a)$$

where " * " denotes the complex conjugate.

In order to see the similarity between the simple standard steepest descent method and the time domain LMS adaptive algorithm, some assumptions about the signals have to be made. Let us assume that the time between successive iterations of the time domain LMS algorithm is sufficiently long so that the sample input vectors $\underline{x}(n)$ and $\underline{x}(n+1)$ are uncorrelated. Furthermore, because the weight vector $\underline{w}(n)$,

is a function of the input vectors $\underline{x}(k)$, $0 \leq k < n$, therefore, $\underline{w}(n)$ is independent of $\underline{x}(n)$. Under these conditions, for stationary input processes, the expected value of the weight vector recursive equation can be written as

$$E[\underline{w}(n+1)] = E[\underline{w}(n)] + 2\mu\{ E[d(n)\underline{x}(n)] - E[\underline{x}(n)\ \underline{x}^T(n)]\ E[\underline{w}(n)] \} \quad (2\text{-}34a)$$

or

$$E[\underline{w}(n+1)] = E[\underline{w}(n)] + 2\mu \{ \underline{p} - \underline{R}\ E[\underline{w}(n)]\} \quad (2\text{-}34b)$$

From (2-31), (2-32), and (2-34), we observe that the mean weight vector recursive equation of the time domain LMS algorithm is equivalent to the simple standard steepest descent method. Moreover, the estimated gradient used in the time domain LMS algorithm is an unbiased estimate, i.e.,

$$E[\ e(n)\ \underline{x}(n)\ ] = \underline{g}(n) \quad (2\text{-}2a)$$

The theoretical analysis of the convergence properties of the time domain LMS algorithm has been done by many authors [14, 39-42]. In general, the analysis of its behavior during adaptation can give insight to other more complicated adaptive processes. For tractability of analysis, certain assumptions are typically necessary to achieve detailed results. Under the assumptions which we used to derive (2-34) and assuming that the input signals and the desired process samples are real jointly Gaussian random variables, the steady-state MSE in terms of the MMSE, eigenvalues and the stepsize, can be explicily obtained, i.e., [42]

$$\xi_{\infty} = \frac{\xi_{min}}{1 - \sum_{i=1}^{N} \{ \frac{2 \mu \lambda_i}{1 - 2\mu\lambda_i} \}} \qquad (2\text{-}35a)$$

where λ_i is the ith eigenvalue of R. Furthermore, in the complex value LMS adaptive filtering algorithm the assumption of circular jointly Gaussian process is used to substitute the real joint Gaussian process which we made for the real value LMS adaptive algorithm. Under these assumptions, the steady-state MSE of the complex LMS adaptive algorithm can be written as [42]

$$\xi_{\infty} = \frac{\xi_{min}}{1 - \sum_{i=1}^{N} \{ \frac{\mu \lambda_i}{1 - \mu \lambda_i} \}} \qquad (2\text{-}35b)$$

The results shown in (2-35) will be used in chapter 4, for the purpose of comparison.

2.5 The Block LMS Adaptive Algorithm

The realization of adaptive digital filters using the block LMS (BLMS) algorithms has been discussed by many authors [16-19]. Block implementation of adaptive digital filter is based on block estimated gradient algorithms, in which the filter weights are updated once in every block

rather than every incoming sampling instant. These block adaptation algorithms use the block mean-square error (BMSE) as a performance measure.

Using the estimated gradient version of simple standard steepest descent method in updating the weights leads to the BLMS algorithms [16,17]. The technique involves calculation of a finite set of output values from a finite set of input values. This implementation allows efficient use of parallel processor, which results in improved speed [16,17]. Also the efficient block algorithms such as the fast Fourier transform (FFT) can be used to advantage when implementing filters on serial processors in time domain [16]. As pointed out in [16] that the most efficient choice of block length (L) would be N, the order of adaptive filter. In this section only the time domain implementation of the BLMS algorithm is considered.

As mentioned previously, the block LMS adaptation algorithm adjust the weights once per block data. The filtered output vector at the mth block can be expressed as [16-18]

$$\underline{y}(m) = \underline{X}^T(m)\, \underline{w}(m) \qquad (2\text{-}36)$$

where $\underline{X}(m)$ is an NxN square matrix and is given by

$$\underline{X}(m) = [\underline{x}(mN) \;\; \underline{x}(mN+1) \;\ldots\; \underline{x}((m+1)N-1)] \qquad (2\text{-}37)$$

and the output vector, $\underline{y}(m)$, is designated by

$$\underline{y}^T(m) = [y(mN) \;\; y(mN+1) \;\ldots y((m+1)N-1)] \qquad (2\text{-}38)$$

Assume that all inputs are stationary and let

$$\underline{d}^T(m) = [d(mN) \;\; d(mN+1) \;\ldots d((m+1)N-1)] \qquad (2\text{-}39)$$

be the Nxl column vector of the desired responses for block m. Then the error vector, $\underline{e}(m)$, can be defined as

$$\underline{e}(m) = \underline{d}(m) - \underline{y}(m) \qquad (2\text{-}40a)$$

and is designated as

$$\underline{e}^T(m) = [e(mN)\ e(mN+1)\ \ldots e((m+1)N-1)] \qquad (2\text{-}40b)$$

The optimal solution of the block digital filter is then obtained by minimizing the average sum squared error, ASSE, i.e.,

$$ASSE = \frac{1}{N} E[\underline{e}^T(m)\ \underline{e}(m)]$$

$$= E\left[\frac{1}{N} \sum_{k=(m+1)N-1}^{mN} e^2(k)\right] \qquad (2\text{-}41)$$

Note that, in (2-41), the ASSE is the expected value of a smoothed estimate of squared error over one block. Substituting (2-36) and (2-40a) into (2-41) we have

$$ASSE = \frac{1}{N}\{ E[\underline{d}^T(m)\underline{d}(m)] - \underline{\rho}^T\underline{w}(m)$$

$$- \underline{w}^T(m)\underline{\rho} + \underline{w}^T(m)\underline{R}\ \underline{w}(m)\} \qquad (2\text{-}42)$$

where

$$\underline{\rho} = E[\underline{X}(m)\ \underline{d}(m)] \qquad (2\text{-}43a)$$

and

$$\underline{R} = E[\underline{X}(m)\ \underline{X}^T(m)] \qquad (2\text{-}43b)$$

Again, assume that the input signals and the desired process samples are jointly stationary, then the optimal set of weights $\underline{w}_{opt}$ for the block Wiener filter is the same as for the Wiener filter [16], i.e.,

$$\underline{w}_{opt} = \underline{\mathcal{R}}\ \underline{\mathcal{P}}$$
$$= \underline{R}^{-1}\ \underline{p}. \qquad (2\text{-}44)$$

This is because in this case, the matrix $\underline{\mathcal{R}}$ and the vector $\underline{\mathcal{P}}$ will be reduced to

$$\underline{\mathcal{R}} = N\ \underline{R}$$

and

$$\underline{\mathcal{P}} = N\ \underline{p}$$

respectively. Now, analogous to the derivation of the conventional time domain LMS algorithm, the BLMS adaptive algorithm can be expressed as

$$\underline{w}(m+1) = \underline{w}(m) + \frac{2}{N}\mu_B\ \underline{X}(m)\ \underline{e}(m)$$
$$= \underline{w}(m) + \frac{2}{N}\mu_B \sum_{k=mN}^{(m+1)N-1} e(k)\ \underline{x}(k) \qquad (2\text{-}45)$$

where μ_B is the constant stepsize which controls the stability and the speed of adaptation in the BLMS algorithm. It has been shown that to obtain similar error performance (the mean-square error) compared to the conventional time domain LMS algorithm, the relationship between μ and μ_B is [16]

$$\mu_B = N\ \mu \qquad (2\text{-}46)$$

There are many ways to implement the BLMS depending on how the linear convolution of the input block matrix, $\underline{X}(m)$, and the error vector, $\underline{e}(m)$, is performed [16-19]. The frequency domain implementation of BLMS algorithm using the overlapped and save FFT algorithm proposed by Ferrara [19] can be viewed as one of them. The detailed discussion of this can be found in [17,19].

2.6 Conclusions

In this chapter, we reviewed the standard steepest descent method, the generalized Newton-Raphson method, the conventional time domain LMS adaptive algorithm and the time domain BLMS adaptive algorithm. Here the generalized Newton-Raphson method is used to improve the convergence rate when the standard steepest descent suffers the problem of slow convergence. This fast convergence of the generalized Newton-Raphson method is achieved at the expense of the increased computation. That is, apart from the evaluation of the gradient vector which is required in the implementation of the standard steepest descent method, the implementation of the generalized Newton-Raphson method also requires the second derivatives of the cost function with respect to the components of the weight vector. On the other hand, the use of the conventional time domain LMS algorithm is to avoid the direct evaluation of the gradient vector which is required in the standard steepest descent method. This results in simple implementation and less computational effort. Finally, the time domain BLMS algorithm is used to further reduce the computaional complexity compared to the time domain LMS algorithm.

We also showed (Appendix A) that the generalized Newton-Raphson method is equivalent to the rescaled version of standard steepest descent method provided that the transformation matrix used for rescaling the weight vector is

selected properly. Furthermore, because the rescaled version of the standard steepest descent method can be viewed as the transformed domain implementation of the standard steepest descent method, therefore, the generalized Newton-Raphson method can also be implemented via the transformed domain LMS adaptive filtering algorithm. Also, for further reduction of the computational complexity, the block LMS algorithm can be applied to the transformed domain LMS adaptive filtering problem.

CHAPTER 3

MATHEMATICAL FORMULATION OF THE GENERALIZED ORTHOGONAL TRANSFORMED DOMAIN ADAPTIVE FILTERS

3.1 Introduction

In last chapter, we reviewed various gradient-following adaptive algorithms. Moreover, among these algorithms, the Newton-Raphson method can be expected to have the fastest convergence rate, provided that the estimated weight vector is close enough to the true weight vector. The relationship between the standard steepest descent method and the Newton-Raphson method was also developed by means of rescaling the weight vector. In fact, the rescaled version standard steepest descent method can be realized as a transformed domain implementation of the standard steepest descent method. To incorporate the idea of rescaled weight vector in the time domain LMS adaptive algorithm, we develop various transformed domain LMS adaptive filters in this chapter. Furthermore, the adaptive algorithm involved in the transformed domain LMS adaptive filter is referred to as a transformed domain LMS adaptive filtering algorithm.

It is widely known [30] that the Karhunen-Loève (K-L) transform is a very useful orthogonal function because it can be chosen to completely diagonalize the input

autocorrelation matrix, $\underline{R}$. Here we derive the transformed domain LMS adaptive filters based on the K-L transform.

In order for us to develop the transformed domain LMS adaptive filters, we will briefly discuss the properties of the K-L transform in the next section. Furthermore, to compare the performance with the time domain LMS algorithm, the equivalent time domain expressions for the K-L transformed domain LMS adaptive filtering algorithms are obtained.

3.2 The Karhunen-Loève Expansion (KLE)

In various practical problems it is convenient to operate with random vectors having uncorrelated components. Thus, the problem of representing a function in terms of a vector with uncorrelated components arises. The Karhunen-Loève expansion (KLE) is one of the techniques to achieve this objective.

The KLE was originally defined on a finite interval for a random function [36,37]. Usually, the Karhunen - Loève theory is developed for continuous random functions. It can be extended quite naturally to the case of discrete random sequences [21]. In the present section, discrete real random sequences are considered. The case of complex random sequences can be similarly obtained and is not addressed here.

Let $\{ x(n) \}$ be a discrete real random sequence, stationary with respect to the second order properties,

defined on the finite set of integer { 1,2, ... ,N} with N being positive, and having the statistical properties such that

$$E[\, x(n)\,] = 0,$$

$$E[\, x^2(n)\,] = \text{constant}$$

$$E[x(n)\, x(k)] = r_{|n-k|}\ ,\ n,k = 1,2,\ \ldots,N \qquad (3\text{-}1)$$

If the column vector $\underline{x}$ is defined as

$$\underline{x} = [\, x(N), x(N-1)\ \ldots, x(1)]^T \qquad (3\text{-}2)$$

then we may define the covariance matrix, $\underline{R}$, as the NxN matrix given by

$$\underline{R} = E[\, \underline{x}\, \underline{x}^T\,]$$

$$= [r_{|n-k|}]_{NxN} \qquad (3\text{-}3)$$

It is known that $\underline{R}$ is positive definite, symmetric, and of Toeplitz form [21].

The KLE can be shown [21,41] to be dependent on the eigenvalues $\{\lambda_i\}$ and eigenvectors $\{\underline{q}_i\}$ of the matrix such that $\{\lambda_i\}$ and $\{\underline{q}_i\}$ satisfied the equation

$$\underline{R}\ \underline{Q} = \lambda\ \underline{Q} \qquad (3\text{-}4)$$

where $\underline{Q}$ is the square matrix whose columns are the eigenvectors:

$$\underline{Q} = [\, \underline{q}_1\ \underline{q}_2 \cdots \underline{q}_N\,] \qquad (3\text{-}5)$$

Then the following properties which relate to (3-4) are valid:

1) the eigenvalues λ_i are real value

2) the eigenvectors form a unitary system

3) $\underline{R} = \sum_{i=1}^{N} \lambda_i\ \underline{q}_i\ \underline{q}_i^T$, or

$$\underline{R} = \underline{Q}\,\underline{R}_{\lambda}\,\underline{Q}^{T} \quad , \text{ with}$$

$$\underline{R}_{\lambda} = \text{diag}[\,\lambda_1\ \lambda_2 \cdots \lambda_N\,]$$

where $\underline{R}_{\lambda}$ is a diagonal matrix. Consequently, we may consider the series expansion as

$$\underline{x} = \sum_{i=1}^{N} s_i\,\underline{q}_i \qquad (3\text{-}6)$$

with orthogonal random variables $\{s_i\}$ such that

$$E[\,s_i\,] = 0$$

$$E[s_i\,s_k] = \delta_{ik} \qquad (3\text{-}7)$$

where δ_{ik} is known as the Kronecker delta and is defined as

$$\delta_{ik} = \begin{cases} 0\ , & i \neq k \\ 1\ , & i = k \end{cases}$$

It can be shown that the representation described in (3-6) produces the required covariance matrix $\underline{R}$. The matrix notation of (3-6) can be written as

$$\underline{x} = \underline{Q}\,\underline{s} \qquad (3\text{-}8)$$

where the representing vector, $\underline{s}$, is designated by

$$\underline{s} = [\,s_1\ s_2 \cdots s_N\,]^{T} \qquad (3\text{-}9)$$

Also, $\underline{s}$ can be obtained by pre-multiplying the inverse of $\underline{Q}$ on both sides of (3-8), such that

$$\underline{s} = \underline{Q}^{-1}\,\underline{x} \qquad (3\text{-}10)$$

Furthermore, the covariance matrix of vector $\underline{s}$ can be shown to be a diagonal matrix:

$$\begin{aligned} E[\,\underline{s}\,\underline{s}^{T}] &= \underline{Q}^{-1} E[\,\underline{x}\,\underline{x}^{T}](\underline{Q}^{-1})^{T} \\ &= \underline{Q}^{-1}\,\underline{R}\,\underline{Q} \\ &= \underline{R}_{\lambda} \end{aligned} \qquad (3\text{-}11)$$

because $\underline{Q}$ is a real unitary matrix ,i.e.,

$$\underline{Q}^{-1} = \underline{Q}^{T} \qquad (3\text{-}12)$$

Note that, (3-11) means that the K-L transform, $\underline{Q}$, can be used to totally decorrelate the representing coefficients.

Although, many of the theoretical derivations use the KLE, in real time implementation it suffers from three major drawbacks [38]: 1) a large number of sampled data and computational effort are required for estimating the direction cosines of the K-L basis vectors, 2) the task of projecting an N-component incoming signal vector into its principal coordinates requires N^2 computer multiplications, and 3) a new NxN matrix must be calculated for each change in the statistical properties of the environment. Thus, other suboptimal representations which possess fast computation algorithms and employ a fixed set of unitary matrices independent of the signal statistics are desired.

The problem of applying the suboptimal representations to the orthogonal transformed domain LMS adaptive filters are discussed in chapter 6.

3.3 Derivation of the Generalized Orthogonal Transformed Domain LMS Adaptive Filters

Recently, various transformed domain LMS adaptive filters have been proposed by many authors [19-23]. In the transformed domain adaptive filter, the LMS adaptive algorithm is used to update the corresponding domain filter

weights with the transformed input signals. Basically, the transformed domain adaptive filters can be classified as: 1) the transformed domain LMS (TDLMS) adaptive filters, and 2) the block transformed domain LMS (BTDLMS) adaptive filters. Correspondingly, the adaptive algorithms involved in the TDLMS and BTDLMS adaptive filters are referred to as the TDLMS adaptive filtering algorithms and the BTDLMS adaptive filtering algorithms, respectively. In the TDLMS algorithms the corresponding weights are updated for every sampling instant whereas in the BTDLMS algorithms the transformed weights are updated for each block of input signals.

It has been shown empirically that, in general, the LMS algorithms do not perform well when the eigenvalue spread is large [23]. In this case, many normalized algorithms are suggested to improve the convergence rate [17,21-23].

In this section, a new comprehensive study of the generalized transformed domain LMS adaptive algorithm is considered. Through this study a clear picture of the effects due to the use of the normalized algorithms can be obtained. More detailed discussion about the statistical behavior on the convergence rate of the normalized algorithm can be seen in chapter 4 and 5.

3.3.1 The Transformed Domain LMS (TDLMS) Adaptive Algorithms

As mentioned earlier, the rescaled version of the standard steepest descent method can be realized as the

transformed domain implementation of the standard steepest descent method. In which, the standard steepest descent method is used to iteratively minimize the cost function, $\hat{F}(\underline{b})$ which is expressed in terms of the rescaled weight vector:

$$\hat{F}(\underline{b}) = F(\underline{w}) = \underline{b}^T \underline{R}^{\#} \underline{b} - 2 \underline{p}^T \hat{\underline{Q}} \underline{b} + c \qquad (3\text{-}13a)$$

with

$$\underline{R}^{\#} = \hat{\underline{Q}}^T \underline{R} \hat{\underline{Q}} \qquad (3\text{-}13b)$$

and

$$\underline{w} = \hat{\underline{Q}} \underline{b} \quad (\text{or } \underline{b} = \hat{\underline{Q}}^{-1} \underline{w}) \qquad (3\text{-}13c)$$

Here, the quantities $\underline{R}$, $\underline{p}$, and c were defined in (2-3) and are the input autocorrelation matrix, the crosscorrelation vector of d(n) and $\underline{x}(n)$ and the autocorrelation value of the desired signal, respectively. The matrix $\hat{\underline{Q}}$ is a positive definite NxN square matrix which consists of N orthogonal functions:

$$\hat{\underline{Q}} = [\ \hat{\underline{q}}_1\ \hat{\underline{q}}_2 \cdots \hat{\underline{q}}_N\ ] \qquad (3\text{-}14a)$$

The ith orthogonal function $\hat{\underline{q}}_i$ of matrix $\hat{\underline{Q}}$ is designated by

$$\hat{\underline{q}}_i = [\ \hat{q}_i(0)\ \ \hat{q}_i(1) \cdots \hat{q}_i(N\text{-}1)\ ]^T \qquad (3\text{-}14b)$$

In order to develop the TDLMS adaptation algorithm, let us define new parameters as follows:

$$\underline{s}(n) = \hat{\underline{Q}}^T \underline{x}(n) \qquad (3\text{-}15a)$$

and

$$\underline{p}^{\#} = \hat{\underline{Q}}^T \underline{p} \qquad (3\text{-}15b)$$

From (3-13) and (3-15) we recognize that

$$\underline{R}^{\#} = E[\ \underline{s}(n)\underline{s}^{T}(n)\ ] \qquad (3\text{-}16a)$$

and

$$\underline{p}^{\#} = E[\ d(n)\ \underline{s}(n)\ ] \qquad (3\text{-}16b)$$

are the transformed input autocorrelation matrix and the crosscorrelation vector of $d(n)$ and $\underline{s}(n)$, respectively. Substituting (3-15b) into (3-13a) we have

$$\hat{F}(\underline{b}) = \underline{b}^{T}\ \underline{R}^{\#}\ \underline{b} - 2(\underline{p}^{\#})^{T}\underline{b} + c \qquad (3\text{-}17)$$

To search the minima of $\hat{F}(\underline{b})$ the standard steepest descent is then applied. This is given as

$$\underline{b}(n+1) = \underline{b}(n) + a^{\#}_{n}\ \underline{g}^{\#}(n) \qquad (3\text{-}18)$$

where $a^{\#}_{n}$ is the variable stepsize and $\underline{g}^{\#}(n)$ is the negative gradient of $\hat{F}(\underline{b})$. Here, $\underline{g}^{\#}(n)$ is given as

$$\underline{g}^{\#}(n) = -\ \nabla \hat{F}(\underline{b}) \qquad (3\text{-}19)$$

Now, analogous to the approach which we obtained the time domain LMS adapation algorithm, the TDLMS algorithm can be expressed as follows:

$$\begin{aligned}\underline{b}(n+1) &= \underline{b}(n) + 2\ \mu^{\#}\ [\ \underline{s}(n)d(n) - \underline{s}(n)\underline{s}^{T}(n)\underline{b}(n)] \\ &= \underline{b}(n) + 2\ \mu^{\#}\ e(n)\ \underline{s}(n) \qquad (3\text{-}20a)\end{aligned}$$

Here, the stepsize is set to be a constant value $\mu^{\#}$ and does not necessarily have the same value as μ used in the time domain LMS algorithm. For the purpose of comparison we use μ instead of $\mu^{\#}$ in (3-20a). Consequently (3-20a) can be written as

$$\underline{b}(n+1) = \underline{b}(n) + 2\mu\ e(n)\ \underline{s}(n) \qquad (3\text{-}20b)$$

Note that, both $\underline{b}(n)$ and $\underline{s}(n)$ are vectors representing the transformed domain weight vector and the transformed input vector at sampling index n:

$$\underline{b}(n) = [\ b_n(0)\quad b_n(1)\quad \ldots\ b_n(N-1)\ ]^T \tag{3-21a}$$

and

$$\underline{s}(n) = [\ s_n(0)\ s_n(1)\ \ldots\ s_n(N-1)\ ]^T \tag{3-21b}$$

From (3-20), the error signal e(n) at nth sampling instant is given by

$$\begin{aligned} e(n) &= d(n) - \underline{s}^T(n)\,\underline{b}(n) \\ &= d(n) - (\underline{\hat{Q}}^T\underline{x}(n))^T\,\underline{b}(n) \\ &= d(n) - \underline{x}^T(n)\,\underline{w}(n) \end{aligned} \tag{3-22}$$

Based on (3-22), we learn that the expression of error signal is the same as in the time domain LMS algorithm. This is due to the fact that the rescaled weight vector does not change the cost function (3-13a), that is, the cost function is expressed in terms of the new weights rather than the original time domain weights. In fact, the TDLMS algorithm can be viewed as the rescaled version of the time domain LMS algorithm in the sense that we can interpret that the time domain filter weights are represented by a set of orthogonal functions, $\{\ \hat{\underline{q}}_i\ \}_{i=1}^{N}$. This is shown in Figure 3.1; the dash line region.

Similarly, for complex valued signals, the transformed domain weight vector updated equation can be expressed as

$$\underline{b}(k+1) = \underline{b}(k) + 2\,\mu\,e(k)\,\underline{s}^*(k) \tag{3-23}$$

That is, instead of the transformed signal vector the complex conjugate of the transformed signal vector is used.

The effect of the rescaled weight vector in the TDLMS

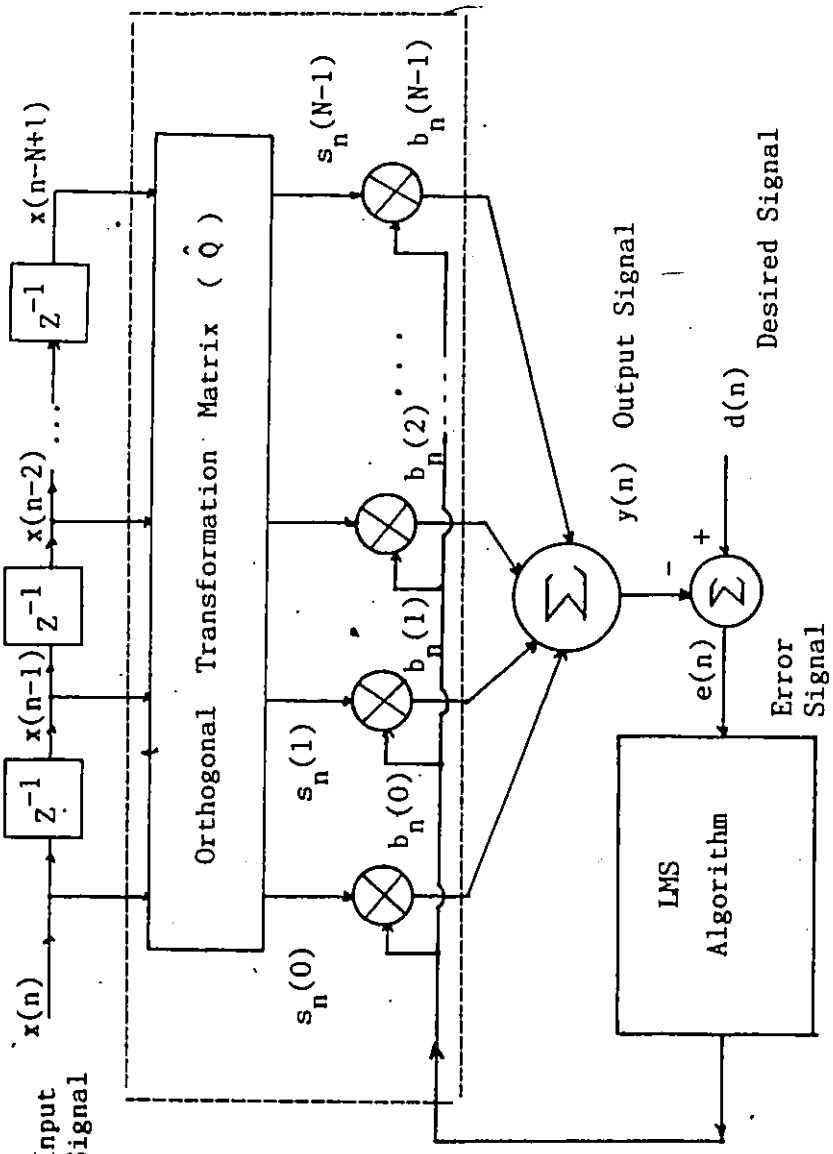


Figure 3.1 The Transformed Domain LMS (TDLMS) Adaptive Filter Structure.

algorithm depends on the transformation matrix $\hat{\underline{Q}}$. This effect has been clearly pointed out in chapter 2, such that an iterative method equivalent to the Newton-Raphson method for searching the mimima of the cost function may result if the matrix $\hat{\underline{Q}}$ is selected properly. In what follows, two specific selections of $\hat{\underline{Q}}$ viz, the K-L transform matrix and the normalized K-L transform matrix are considered. The term " normalized " K-L transform means that the transformed input autocorrelation matrix under such transformation will become an identity matrix.

(A) The TDLMS Adaptive Algorithm with the K-L Transform Matrix

In this section, a real valued TDLMS algorithm is used to study the effect of the transformation matrix. Here the transformation matrix is chosen to be a K-L transform matrix, i.e., $\hat{\underline{Q}} = \underline{Q}$. As we learn from sec. 3.2 that the K-L transform matrix is a real unitary matrix for the real valued signals:

$$\underline{Q}^{-1} = \underline{Q}^{T} \qquad (3\text{-}12)$$

For the purpose of comparison, the equivalent time domain expression of the TDLMS algorithm has to be obtained. To do so, we multiply both sides of (3-20b) by $\underline{Q}$, resulting in the following,

$$\begin{aligned} \underline{w}(k+1) &= \underline{Q}\,\underline{b}(k+1) \\ &= \underline{Q}\,\underline{b}(k) + 2\,\mu\, e(k)\,\underline{Q}\,\underline{s}(k) \\ &= \underline{w}(k) + 2\,\mu\, e(k)\,\underline{x}(k) \end{aligned} \qquad (3\text{-}24)$$

It can be recognized that (3-24) is the same as the time domain LMS adaptive algorithm shown in (2-32). This is because the K-L transform matrix is a unitary matrix. Furthermore, (3-24) suggests that the use of a unitary matrix, in the TDLMS algorithm, does not improve the convergence rate.

(B) The TDLMS Adaptive Algorithm with the Normalized K-L Transform Matrix

In this part, the K-L transform matrix is substituted by the normalized K-L transform matrix, i.e.,

$$\hat{\underline{Q}} = \underline{Q}\ \underline{R}_{\lambda}^{-1/2} \qquad (3\text{-}25)$$

Here, $\underline{R}_{\lambda}$ is a diagonal matrix with its elements being the eigenvalues of the input autocorrelation matrix $\underline{R}$. As we learn from sec. 3.2, the input autocorrelation matrix $\underline{R}$ can be expanded as

$$\underline{R} = \underline{Q}\ \underline{R}_{\lambda}\ \underline{Q}^{T} \qquad (3\text{-}26)$$

Furthermore, in order to see the time domain relation of TDLMS algorithm with the time domain LMS algorithm, we define the inverse square root matrix, $\underline{R}^{-1/2}$ as

$$\begin{aligned}\underline{R}^{-1/2} &= \underline{Q}\ \underline{R}_{\lambda}^{-1/2}\ \underline{Q}^{T} \\ &= \underline{Q}\ \underline{R}_{\lambda}^{-1/2}\ \underline{Q}^{-1}\end{aligned} \qquad (3\text{-}27)$$

It can be shown that the matrix $\underline{R}^{-1/2}$ is equivalent to the normalized K-L transform in (3-25). This is because both matrices can be used to transform the input autocorrelation matrix to an identity matrix. Equivalently, the matrix $\underline{R}^{-1/2}$

can be used to substitute the normalized K-L transform matrix in the TDLMS adaptive algorithm. Now, to obtain the equivalently time domain expression of the TDLMS algorithm, we multiply both sides of (2-20a) by $\underline{R}^{-1/2}$, this results in

$$\begin{aligned}\underline{w}(n+1) &= \underline{R}^{-1/2}\underline{b}(n+1)\\ &= \underline{R}^{-1/2}\underline{b}(n) + 2\mu\, e(n)\, \underline{R}^{-1/2}\underline{s}(n)\\ &= \underline{w}(n) + 2\mu\, e(n)\, \underline{R}^{-1}\underline{x}(n)\end{aligned}$$

or

$$\begin{aligned}\underline{w}(n+1) &= \underline{w}(n) + 2\mu\, e(n)\, \underline{R}^{-1}\, \underline{x}(n)\\ &= \underline{w}(n) + 2\mu\, e(n)\, \underline{Q}\, \underline{R}_{\lambda}^{-1}\, \underline{Q}^{-1}\underline{x}(n) \qquad (3\text{-}28)\end{aligned}$$

where

$$\underline{w}(n) = \underline{R}^{-1/2}\, \underline{b}(n) \qquad (3\text{-}29)$$

and

$$\underline{s}(n) = (\, \underline{R}^{-1/2}\,)^{T}\, \underline{x}(n) \qquad (3\text{-}30)$$

Here, $\underline{R}$ is symmetric, therefore $\underline{R}^{-1/2}$ is also symmetric. Note that, (3-28) is the stochastic analog of the Newton-Raphson method with the stepsize being a constant value and the estimated gradient is used instead of the true gradient (see chapter 2). Further, let's define a new weight vector, $\underline{c}(n)$:

$$\underline{c}(n) = \underline{Q}^{-1}\, \underline{w}(n) \qquad (3\text{-}31)$$

or

$$\underline{w}(n) = \underline{Q}\, \underline{c}(n) \qquad (3\text{-}32)$$

and a new input vector, $\underline{z}(n)$, to be as

$$\begin{aligned}\underline{z}(n) &= \underline{Q}^{T}\underline{x}(n)\\ &= \underline{Q}^{-1}\underline{x}(n) \qquad (3\text{-}33)\end{aligned}$$

Then by substituting (3-31), (3-32) and (3-33) in (3-28) we have

$$\underline{c}(n+1) = \underline{c}(n) + 2\,\mu\,\underline{R}_{\lambda}^{-1}\,e(n)\,\underline{z}(n) \qquad (3\text{-}34)$$

or in the scalar form we have

$$c_{n+1}(i) = c_n(i) + 2\,\mu_i\,e(n)\,z_n(i) \qquad (3\text{-}35)$$

with

$$\mu_i = \frac{\mu}{\lambda_{i+1}}\,, \quad i = 0,1,\ldots,N-1 \qquad (3\text{-}36)$$

where $z_n(i)$ is the ith element of vector $\underline{z}(n)$. Note that, (3-34) is the transformed domain adaptive filtering algorithm proposed by Narayan et. al [23]. Since (3-28) is equivalent to (2-34), we can then view (3-34) as one of the implementations of the normalized K-L transformed domain LMS adaptive algorithm. Furthermore, in practice, the ith eigenvalue can be estimated from the corresponding transformed input signal power. Consequently, if $\underline{Q}$ can be exactly obtained, (3-28) can be implemented via (3-34). Also, as mentioned in sec. 3.2 that in the real time implementation the K-L transform matrix $\underline{Q}$ is difficult to obtain, therefore, the suboptimal orthogonal transformations have to be used. However, from a theoretical point of view, the normalized K-L TDLMS algorithm can be used for analysis and further study of the convergence properties of the TDLMS algorithm. The results can also be compared with the time domain LMS algorithm. These are addressed in the subsequent chapters.

3.3.2 The Block Transformed Domain LMS (BTDLMS) Adaptive Algorithms

In section 2.4, we discussed the BLMS adaptive filtering algorithm which is the realization of the conventional time domain LMS adaptive filtering algorithm by blockwise processing of the data. Also, in the last section, we used a similar approach as we derived the time domain LMS adaptive algorithm to derive the TDLMS adaptive algorithm. To incorporate the ideas of block processing and rescaled weight vector, we develop , in this section, the block transformed domain LMS adaptive filters. Again, the adaptive algorithms involved in the BTDLMS adaptive filters are referred to as the BTDLMS adaptive filtering algorithms.

Basically, the use of the BTDLMS adaptive algorithm is to reduce the computational complexity compared to the TDLMS algorithm. The structure of the BTDLMS adaptive filter is very similar to the basis - restricted (B.R) transformation filter proposed by Pearl [29]. Indeed, the BTDLMS adaptive filter can be viewed as an adaptive implementation of the B.R. transformation filter. It is called the B.R. adaptive filter.

In what follows, the BTDLMS adaptive filters are considered. Again, the transformation matrix is chosen to be the K-L transform matrix or the normalized K-L transform matrix.

The BTDLMS adaptive filter or the B.R. BTDLMS adaptive filter is depicted in Figure 3-2. The two restrictions,i.e.,

(1) the basis set $\{ \underline{q}_i \}$ is to remain fixed all the time

(2) the filter parameters or coefficients operate

separately on each individual component of the transformed input signal vector.

are imposed on the filter structure. Basically, the B.R. transformed domain filter [29] is so designed that produces the best estimate of the transformed domain desired signal vector, $\underline{\hat{d}}(m)$, based on the observation on the transformed input vector, $\underline{\hat{s}}(m)$, at the mth block instant. Here, $\underline{\hat{d}}(m)$ and $\underline{\hat{s}}(m)$ are defined as

$$\underline{\hat{d}}(m) = \hat{\underline{Q}}^{-1} \underline{d}(m) \qquad (3\text{-}37a)$$

and

$$\underline{\hat{s}}(m) = \hat{\underline{Q}}^{-1} \underline{\hat{x}}(m) \qquad (3\text{-}37b)$$

respectively, and are designated by

$$\underline{\hat{d}}(m) = [\, \hat{d}_m(0)\ \hat{d}_m(1)\ \ldots\ \hat{d}_m(N-1)] \qquad (3\text{-}38a)$$

and

$$\underline{\hat{s}}(m) = [\, \hat{s}_m(0)\ \hat{s}_m(1)\ \ldots\ \hat{s}_m(N-1)] \qquad (3\text{-}38b)$$

The mth block input signal vector and the mth block desired signal vector in the time domain are denoted by

$$\underline{\hat{x}}(m) = [x((m+1)N-1)\ x((m+1)N-2)\ \ldots\ x(mN)]^T \qquad (3\text{-}39a)$$

and

$$\underline{d}(m) = [d((m+1)N-1)\ d((m+1)N-2)\ \ldots\ d(mN)]^T \qquad (3\text{-}39b)$$

The total block number to be processed will be the total sampling number divided by the integer number N, the filter length. From Figure 3.2, the filtered output vector $\hat{\underline{y}}(m)$ is given by

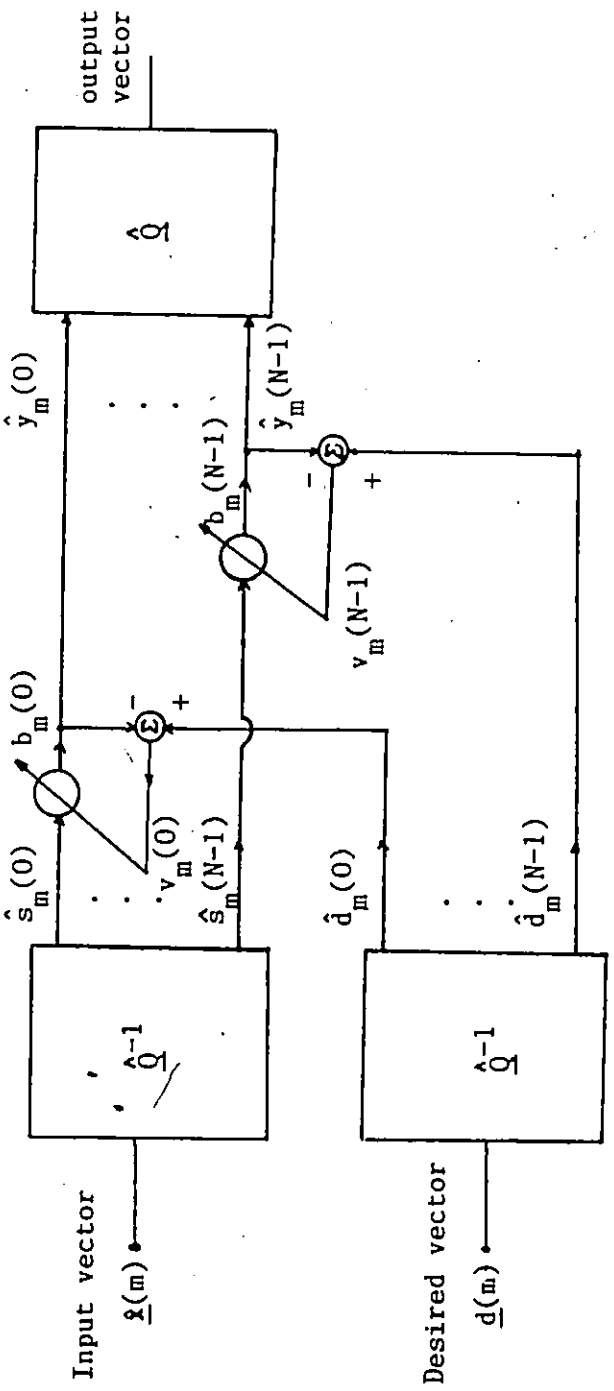


Figure 3.2 The Block Transformed Domain LMS (BTDLMS) Adaptive Filter Structure.

$$\hat{\underline{y}}(m) = \begin{bmatrix} b_m(0)\hat{s}_m(0) \\ b_m(1)\hat{s}_m(1) \\ \vdots \\ b_m(N-1)\hat{s}_m(N-1) \end{bmatrix} = \begin{bmatrix} \hat{s}_m(0) & 0 & \cdots & 0 \\ 0 & \hat{s}_m(1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \hat{s}_m(N-1) \end{bmatrix} \underline{b}(m)$$

$$= \underline{P}(m)\, \underline{b}(m) \qquad (3\text{-}40)$$

where the diagonal matrix , $\underline{P}(m)$, is designated by

$$\underline{P}(m) = \text{diag}\,[\,\hat{s}_m(0)\ \hat{s}_m(1)\ \ldots\ \hat{s}_m(N-1)\,] \qquad (3\text{-}41)$$

The error signal vector, $\underline{v}(m)$, is defined as the difference of the transformed desired signal vector and the transformed output signal vector:

$$\underline{v}(m) = \hat{\underline{d}}(m) - \hat{\underline{y}}(m) \qquad (3\text{-}42)$$

and is designated by

$$\underline{v}(m) = [\,v_m(0)\ \ v_m(1)\ \ldots\ v_m(N-1)\,] \qquad (3\text{-}43)$$

The transformed domain optimal weight vector is obtained by minimizing the average sum squared error, ASSE, i.e.,

$$\text{ASSE} = E\,[\,\underline{v}^T(m)\,\underline{v}(m)\,]$$

$$= E\{\textstyle\sum_{i=0}^{N-1} v_m^2(i)\,\} \qquad (3\text{-}44)$$

This is given by [29]

$$\underline{b}_{opt} = \begin{bmatrix} E[\hat{s}^2_m(0)] & 0 & \cdots & 0 \\ 0 & E[\hat{s}^2_m(1)] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & E[\hat{s}^2_m(N-1)] \end{bmatrix}^{-1} \begin{bmatrix} E[\hat{d}_m(0)\hat{s}_m(0)] \\ E[\hat{d}_m(1)\hat{s}_m(1)] \\ \vdots \\ E[\hat{d}_m(N-1)\hat{s}_m(N-1)] \end{bmatrix}$$

$$= \hat{\underline{R}}^{-1} \hat{\underline{\rho}} \qquad (3\text{-}45)$$

The scalar form of (3-45) can be expressed as

$$b_{opt}(i) = \frac{E[\, \hat{d}_m(i)\, \hat{s}_m(i)\,]}{E[\, \hat{s}_m(i)\, \hat{s}_m(i)\,]} \qquad (3\text{-}46)$$

where $b_{opt}(i)$ is the ith element of $\underline{b}_{opt}$. Since in (3-46) the filter weights are independently operated, we need only the scalar LMS adaptation algorithm to iteratively approximate the optimal weights:

$$b_{m+1}(i) = b_m(i) + 2\, \mu_B\, v_m(i)\, \hat{s}_m(i)$$
$$i = 0, 1, \ldots, N-1 \qquad (3\text{-}47)$$

Here, (3-47) is referred to as the BTDLMS adaptive filtering algorithm.

Similarly, for a complex valued data, the complex valued BTDLMS adaptive algorithm can be expressed as

$$b_{m+1}(i) = b_m(i) + 2\, \mu_B\, v_m(i)\, \hat{s}^*_m(i)$$
$$i = 0, 1, \ldots, N-1 \qquad (3\text{-}48)$$

The advantage of using the BTDLMS adapative filter structure is due to its less computational effort and ease in implementation, especially, when fast algorithm of the transformation are available.

As we mentioned previously, that the LMS adaptation algorithm may have worse performance due to the large eigenvalue spread. To improve the performance we need to reduce the spread. To do so, we can divide the stepsize by the individual transformed signal power to achieve the unity ratio of the eigenvalues. Therefore, the normalized BTDLMS

adaptive algorithm can be given by

$$b_{m+1}(i) = b_m(i) + 2\,\mu_i\, v_m(i)\, \hat{s}_m(i)$$

with

$$\mu_i = \frac{\mu_B}{\lambda_{i+1}} \qquad i=0,1, \ \ldots\ , N-1. \qquad (3\text{-}49)$$

(3-49) can be viewed as the BTDLMS adaptive algorithm with a normalized K-L transform matrix. This is because the form of (3-49) is very similar to (3-35). Further, to see the relation of the BTDLMS algorithm with the time domain BLMS algorithm, the time domain expressions of (3-47) and (3-49) are obtained in the following paragraph.

The time domain expressions for both the BTDLMS and the normalized BTDLMS adaptive algorithms can be obtained by multiplying both sides of (3-47) and (3-49) by the K-L matrix $\underline{Q}$. The matrix form of these are as follows:

$$\begin{aligned}\underline{w}(m+1) &= \underline{Q}\,\underline{b}(m+1)\\ &= \underline{Q}\,\underline{b}(m) + 2\mu_B\,\underline{Q}\,\underline{P}(m)\,\underline{v}(m)\\ &= \underline{w}(m) + 2\,\mu_B\,\underline{X}(m)\,\hat{\underline{e}}(m)\end{aligned} \qquad (3\text{-}50)$$

and

$$\begin{aligned}\underline{w}(m+1) &= \underline{Q}\,\underline{b}(m+1)\\ &= \underline{Q}\,\underline{b}(m) + 2\,\mu_B\underline{Q}\,\underline{R}^{-1}\underline{P}(m)\underline{v}(m)\\ &= \underline{w}(m) + 2\,\mu_B\,\underline{R}_\lambda^{-1}\underline{X}(m)\hat{\underline{e}}(m)\end{aligned} \qquad (3\text{-}51)$$

Note that, (3-50) and (3-51) are the equivalent time domain expressions of (3-47) and (3-49) respectively. Here, the time domain error vector, $\hat{\underline{e}}(m)$, and the block signal matrix, $\underline{X}(m)$, are defined by

$$\hat{\underline{e}}(m) = \underline{Q}\,\underline{v}(m)$$

$$= \underline{d}(m) - \underline{X}(m)\,\underline{w}(m) \qquad (3\text{-}52)$$

and

$$\underline{X}(m) = \underline{Q}\,\underline{P}(m)\,\underline{Q}^{-1}$$

$$= \begin{bmatrix} h_{0,0}(m) & h_{0,1}(m) & \cdots & h_{0,N-1}(m) \\ h_{1,0}(m) & h_{1,1}(m) & \cdots & h_{1,N-1}(m) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ h_{N-1,0}(m) & h_{N-1,1}(m) & \cdots & h_{N-1,N-1}(m) \end{bmatrix} \qquad (3\text{-}53)$$

with

$$h_{n,k}(m) = \sum_{i,j=0}^{N-1} x((m+1)N-j-1)\; q_n(i+1) q_k(i+1) q_j(i+1)$$

$$n,k = 0,1,\ldots,N-1 \qquad (3\text{-}54)$$

Note that, matrix $\underline{R}_\lambda$, (3-51), is due to the fact that $\underline{Q}$ matrix diagonalizes the autocorrelation matrix:

$$E\{\,\hat{\underline{s}}(m)\,\hat{\underline{s}}^T(m)\,\} = \underline{Q}^{-1}\,E\,[\;\underline{x}(m)\,\underline{x}^T(m)\;]\,\underline{Q}$$

$$= \underline{Q}^{-1}\,\underline{R}\,\underline{Q}$$

$$= \underline{R}_\lambda \qquad (3\text{-}55)$$

Here, the assumption that the input signal and desired signal are real and jointly stationary processes, has been used.

3.4 Conclusions

In this chapter, we have developed the mathematical formulations of the TDLMS adaptive filtering algorithm and the BTDLMS adaptive filtering algorithms. Furthermore, for the purpose of comparison, the equivalent time domain

expressions for these algorithms are obtained.

As we have seen, the transformed domain LMS adaptive filtering algorithms are highly dependent on the selection of the filter structure and the transformation matrix which is used to rescale the weight vector. If the same filter structure as in the time domain is selected and the transformation matrix being chosen is a unitary matrix, then the TDLMS adaptive filtering algorithm is equivalent to the time domain LMS adaptation algorithm. Moreover, if the transformation matrix described above is substituted by the normalized K-L transform matrix, the TDLMS adaptive algorithm will be equivalent to the analogous Newton-Raphson method. On the other hand, in the BTDLMS adaptive filter, the filter structure is different from that of the time domain adaptive filter, the algorithms are then different from the time domain algorithms. This is because, in this case, we imposed two restrictions on the filter structure. However, from the transformed domain adaptive algorithms, we observe that the effect of the normalized K-L transform matrix is to re-construct a performance surface, i.e., spherical contours, which is suitable for using the LMS algorithm. This is exactly what we found in the rescaled standard steepest descent method which is, indeed, equivalent to the Newton-Raphson method.

In general, the BTDLMS adaptive algorithms require less computational effort compared to the TDLMS algorithms, especially, when the order of the filter is large. Moreover,

in the transformed domain implementation, it requires a large amount of computation for taking the transformation. This will increase the computation complexity compared to the time domain algorithms. Therefore, a transformation matrix with fast algorithms is required to reduce the computational effort, especially, in the TDLMS adaptive algorithm.

CHAPTER 4

STATISTICAL ANALYSIS OF THE NORMALIZED K-L TDLMS ALGORITHM IN THE TIME DOMAIN

4.1 Introduction

The principal reason for analyzing a particular algorithm is to examine its behaviour during adaptation, which might give insight to other more complicated adaptation processes. Furthermore, the analytical results can be used to compare with other algorithms to see how it performs. Frequently, this may introduce a new algorithm as a compromise between the existing algorithms.

In previous chapters, we discussed various transformed domain LMS adaptive algorithms from the viewpoint of rescaling weight vector. The rescaling weight vector, $\underline{b}$, is implemented in the K-L transformed domain or the normalized K-L transformed domain. We also derived the equivalent time domain expressions for those algorithms. In chapter 3, we showed that the time domain expression of the normalized K-L TDLMS algorithm can be explicitly obtained. In this chapter, this algorithm is chosen to be analyzed.

As we have seen in chapter 3 the normalized K-L TDLMS adaptive algorithm in the time domain has the form analogous to the generalized Newton-Raphson method, which is known to be one of the fastest convergence algorithms among

the existing adaptive algorithms. The normalized K-L TDLMS algorithm is equivalent to the transformed domain adaptive algorithm discussed in [23], it has been shown empirically to be superior in performance to the conventional time domain LMS algorithm when the ratio of the largest and smallest eigenvalues is large. It is interesting to study the effect analytically for the normalized K-L TDLMS algorithm.

It has been mentioned previously that the analysis of the LMS algorithms is not generally tractable. The complication for analyzing the LMS algorithms is due to the fact that the weights are recursively updated by a quadratic function of the input and the desired response signals. Moreover, since the convergence of LMS algorithms is controlled by the constant stepsize, μ, the weight vector can only come close to the neighborhood of the optimum weights. Therefore, certain typical assumptions are necessary to achieve detailed results. Here, we follow the same assumptions as in the real value time domain LMS adaptive algorithm [41] and the complex value time domain LMS adaptive algorithm [42]. This enables us to obtain the mean weights and the weight covariance matrix updating equations for the normalized K-L TDLMS algorithm to explictly express the steady-state MSE as well as the transient MSE.

Before we go further to the analysis of the normalized K-L TDLMS algorithm, we first explore the relation between the normalized K-L TDLMS algorithm in the time domain and the

the generalized Newton-Raphson method.

4.2 A Generalized Newton-Raphson Method

The generalized Newton-Raphson method discussed in chapter 2 can be described as follows :

$$\underline{w}(n+1) = \underline{w}(n) - a_n \underline{G}_n \underline{\nabla} F(\underline{w}) \qquad (4\text{-}1)$$

where

a_n : is the variable stepsize defined in (2-12)

$\underline{G}_n$: is the metric and is chosen to be the inverse of the Hessian matrix, i.e., $\underline{G}_n = \underline{R}^{-1}_{w(n)}$

$\underline{\nabla} F(\underline{w})$: is the gradient of the MSE, $F(\underline{w})$.

$\underline{w}(n)$: is the nth iterative weight vector.

If the process which yields the sequence $\{ \underline{x}(n), d(n) \}$ is wide-sense stationary then the metric, $\underline{G}_n$, and the gradient vector can be expressed as (2-18)

$$\underline{G}_n = \underline{R}^{-1} \qquad (4\text{-}2a)$$

and (2-10)

$$\underline{\nabla} F(\underline{w}(n)) = -2\,\underline{p} + 2\,\underline{R}\,\underline{w}(n) \qquad (4\text{-}2b)$$

respectively. Substituting (4-2a) and (4-2b) in (4-1), results in

$$\underline{w}(n+1) = \underline{w}(n) + 2\,a_n\,\underline{R}^{-1}\,[\underline{p} - \underline{R}\,\underline{w}(n)] \qquad (4\text{-}3)$$

Referring to chapter 3, the normalized K-L TDLMS algorithm in the time domain is given by

$$\begin{aligned}\underline{w}(n+1) &= \underline{w}(n) + 2\mu\,\underline{R}^{-1}\,e(n)\,\underline{x}(n) \\ &= \underline{w}(n) + 2\mu\,\underline{R}^{-1}\{d(n)\underline{x}(n) - \underline{x}(n)\underline{x}^T(n)\underline{w}(n)\}\end{aligned} \qquad (4\text{-}4)$$

where

e(n) : is the error signal at nth sampling instant

$\underline{x}(n)$: is the input vector at nth sampling instant

Now, from (4-3) and (4-4), we recognize that the relationship between the normalized K-L TDLMS algorithm and the generalized Newton- Raphson method is similar to the relationship of the simple standard steepest descent and the time domain LMS adaptive algorithm. That is, the variable stepsize, a_n, is set to be a constant scalar value, μ, and the estimate gradient being used instead of the true gradient. Since the weight vector, $\underline{w}(n)$, in (4-4) behaves similarly to the generalized Newton- Raphson method in the mean weight sense, so that the normalized K-L TDLMS adaptive algorithm can be viewed as the stochastic analog of the generalized Newton-Raphson method.Furthermore,the convergence properties of the normalized K-L TDLMS algorithm in the time domain can be expected to behave similarly to the generalized Newton-Raphson method.

4.3 Statistical Analysis of the Normalized K-L TDLMS Algorithm in the Time-Domain

In the present and subsequent sections, various statistical analysis for the normalized K-L TDLMS algorithm will be performed. It is well-known that, in general, the convergence rate of the adaptive algorithm can be measured in two ways: 1) to determine analytically or empirically how

the weight vector approaches its optimum value with respect to time, 2) to determine analytically or empirically how the MSE decreases with time. Alternatively, the first measure can be viewed as the measure of the error weight vector between the present estimated weight vector and the optimal weight vector. In this section, we will examine the convergence rate analytically by utilizing the first measure. Also, in the next two sections, we will examine the convergence rate by the second measure. It should be noted that, to obtain the MSE we need to derive the mean weight vector as well as the weight covariance matrix recursive equations.

4.3.1 The Convergence Rate of the Normalized K-L TDLMS Algorithm

The convergence rate for $\underline{w}(n)$ to converge to the optimal weight vector, $\underline{w}_{opt}$, can be defined as the error weight vector, $\underline{w}^e(n)$, i.e.,

$$\underline{w}^e(n) = \underline{w}(n) - \underline{w}_{opt} \qquad (4\text{-}5a)$$

It will be more interesting to see the error between the mean weight vector and the optimal weight vector when the input signals are random sequences. In this case, (4-5a) can be written as

$$\underline{w}^e(n) = E[\,\underline{w}(n)\,] - \underline{w}_{opt}$$

$$= \underline{M}_{w(n)} - \underline{w}_{opt} \qquad (4\text{-}5b)$$

Since the mean weight behaviour of (4-5b) is the same as the

generalized Newton-Raphson method, therefore, the convergence rate of the normalized K-L TDLMS adaptive algorithm can be expected to be similar to the generalized Newton-Raphson method in the mean weight sense.

To see the convergence rate in terms of the error weight vector, the difference between the estimated mean weight vector and the optimal weight vector, we substract both sides of (4-4) by $\underline{w}_{opt}$, i.e.,

$$\begin{aligned}\underline{w}^e(n+1) &= E[\ \underline{w}(n+1)\ ] - \underline{w}_{opt} \\ &= E[\ \underline{w}(n)\ ] + 2\mu\, \underline{R}^{-1}\{\underline{p} - \underline{R}\, E[\underline{w}(n)]\} - \underline{w}_{opt} \\ &= (1 - 2\mu)\, \underline{w}^e(n) \qquad (4\text{-}6a)\end{aligned}$$

or

$$\underline{w}^e(n+1) - (1 - 2\mu)\, \underline{w}^e(n) = \underline{0} \qquad (4\text{-}6b)$$

Since the components of $\underline{w}^e(n)$ are uncoupled (4-6), then the convergence rate for each individual component is the same. Therefore, it is only necessary to consider the ith component of the error weight vector to explore the convergence rate, the rate for $\underline{w}(n)$ to approach the optimal weight. We recognize that (4-6b) is the first order difference equation of $\underline{w}^e(n)$, and the impulse response is geometric, having the geometric ratio given by

$$r_i = 1 - 2\mu \qquad (4\text{-}7a)$$

Similar to the case of the time domain LMS algorithm [14], an exponential envelope of time constant t_i can be fitted to the discrete impulse response by considering the unit of time to be one adaptation iterative cycle and making the time constant such that

$$r_i = \exp\{ - 1/ t_i \}$$
$$= 1 - 1/ t_i + 1/ (2! t^2_i) - .... \quad (4\text{-}7b)$$

For large t_i (slow adaptation), the geomeric ratio become as

$$r_i = (1-2\mu)$$
$$\tilde{=} \ 1 - 1/t_i$$

or

$$t_i \tilde{=} \ 1/ (2\mu) \quad (4\text{-}8a)$$

Note that,(4-8a) gives the time constant of the ith component or mode. It has been shown that [14] the time constant, t_i, of the time domain LMS algorithm is

$$t_i \tilde{=} \ 1/ (2 \mu \lambda_i) \quad (4\text{-}8b)$$

Comparing (4-8a) with (4-8b), we found that the time constant, t_i, of the ith mode in (4-8a) is independent of the ith eigenvalue which is not the case as can be seen in (4-8b) for the LMS algorithm. On the other hand, we can interpret the result as that because of the normalized K-L transform matrix, the eigenvalues of the transformed input autocorrelation matrix became unity so that (4-8b) reduces to (4-8a). Furthermore, to assure that (4-6a) converges, it is necessary to have the constant stepsize obey

$$1/2 > \mu > 0 \quad (4\text{-}9)$$

which means that the value of μ should be chosen to be less than 1/2 to assure the adaptation algorithm to converge. Moreover, from (4-9), the upper bound of the constant stepsize for the normalized K-L TDLMS algorithm can also be predetermined. Again, this is not the case for the time domain

LMS algorithm (see Chapter 2), where the value of μ is determined by the trace of the input autocorrelation matrix or the total signal power.

4.4 Statistical Analysis of the Normalized K-L TDLMS Algorithm with Real Valued Data

In this section the normalized K-L TDLMS algorithm in the time domain with real valued data is analyzed. To obtain the MSE, the mean weight vector and the covariance weight matrix recursive equations are obtained.The normalized K-L TDLMS algorithm is described in (4-4), i.e.,

$$\underline{w}(n+1) = \underline{w}(n) + 2\,\mu\,\underline{R}^{-1}\,e(n)\,\underline{x}(n) \qquad (4\text{-}4)$$

where the quantities, $\underline{w}(n)$, $e(n)$, $\underline{R}$, and $\underline{x}(n)$ were defined in previous sections (see Sec. 4.2). Here, the input signals and the desired signals are assumed to be jointly Gaussian, zero-mean, statistical independent random processes. Also, the successive input vectors $\underline{x}(n)$ and $\underline{x}(n+1)$ are uncorrelated with each other. For convenience of analysis,rewrite (4-4) as

$$\begin{aligned}\underline{w}(n+1) &= \underline{w}(n) + 2\,\mu\,\underline{R}^{-1}\{\,[d(n)-y(n)]\,\underline{x}(n)\} \\ &= \underline{w}(n) + 2\,\mu\,\underline{R}^{-1}\{\,d(n)\underline{x}(n) - \underline{x}(n)\underline{x}^{T}(n)\underline{w}(n)\} \\ &= [\underline{I} - 2\mu\,\underline{R}^{-1}\underline{x}(n)\underline{x}^{T}(n)]\,\underline{w}(n) \\ &\quad + 2\mu\,\underline{R}^{-1}d(n)\,\underline{x}(n) \qquad (4\text{-}10)\end{aligned}$$

Based on (4-10), the mean weight vector as well as weight covariance matrix recursive equations are evaluated in the the following section.

4.4.1 The Mean Weight Vector Recursive Equation

Since, in (4-10), $\underline{w}(n)$ depends only on { $d(k)$, $\underline{x}(k)$; $k = 0, 1, ..,n-1$} and also from the assumptions that the successive input vectors $\underline{x}(n)$ and $\underline{x}(n+1)$ are statistically independent so that the mean weight vector recursive equation can be obtained by taking the expectation for both sides of (4-10), which is

$$\begin{aligned} \underline{M}_{w(n+1)} &= E[\ \underline{w}(n+1)\ ] \\ &= \{\ \underline{I} - 2\mu\ \underline{R}^{-1}\ E[\ \underline{x}(n)\ \underline{x}^T(n)]\ \}\ E[\underline{w}(n)] \\ &\quad + 2\mu\ \underline{R}^{-1}\ E[\ d(n)\ \underline{x}\ (n)] \\ &= (1-2\mu)\ \underline{M}_{w(n)} + 2\mu\ \ \underline{R}^{-1}\ \underline{p} \end{aligned} \qquad (4\text{-}11a)$$

Starting with the initial weight vector, $E[\underline{w}(0)]$, and iteratively substituting in (4-11a) results in

$$\underline{M}_{w(n+1)} = (\ 1-2\mu\ )^{n+1}\ E[\underline{w}(0)] + 2\mu \sum_{i=0}^{n} (1-2\mu)^i \underline{R}^{-1}\underline{p} \qquad (4\text{-}11b)$$

where $\underline{I}$ is an NxN identity matrix. The input autocorrelation matrix, $\underline{R}$, and the crosscorrelation vector, $\underline{p}$, are defined by

$$\underline{R} = E[\ \underline{x}(n)\ \underline{x}^T(n)\ ],$$

and

$$\underline{p} = E[\ d(n)\ \underline{x}(n)\ ]$$

respectively. For further evaluation of the MSE we need to derive the weight covariance matrix recursive equation.

4.4.2 The Weight Covariance Matrix Recursive Equation

Now, consider the weight covariance matrix, Cov[$\underline{w}(n+1)$ $\underline{w}^T(n+1)$], i.e.,

$$\text{Cov}[\,\underline{w}(n+1)\,\underline{w}^T(n+1)\,] = E\{[\underline{w}(n+1)-\underline{M}_{w(n+1)}][\underline{w}(n+1)-\underline{M}_{w(n+1)}]^T\} \qquad (4\text{-}12)$$

Substituting (4-10) and (4-11) into (4-12) and after some manipulation, we have

$$\begin{aligned}
&\text{Cov}[\,\underline{w}(n+1)\,\underline{w}^T(n+1)\,] \\
&= \text{Cov}[\,\underline{w}(n)\,\underline{w}^T(n)\,] - 4\mu\,\text{Cov}[\underline{w}(n)\,\underline{w}^T(n)] \\
&+ 4\mu^2\underline{R}^{-1}E\{\underline{x}(n)\underline{x}^T(n)E[\underline{w}(n)\underline{w}^T(n)]\underline{x}(n)\underline{x}^T(n)\}\underline{R}^{-1} \\
&- 4\mu^2\,\underline{R}^{-1}\,E\{\,\underline{x}(n)\underline{x}^T(n)\,E[\underline{w}(n)]d(n)\underline{x}^T(n)\,\}\,\underline{R}^{-1} \\
&+ 4\mu^2\,\underline{M}_{w(n)}\,\underline{p}^T\,\underline{R}^{-1} - 4\mu^2\,\underline{M}_{w(n)}\,\underline{M}^T{}_{w(n)} \\
&- 4\mu^2\,\underline{R}^{-1}\,E\{\,d(n)\underline{x}(n)E[\underline{w}^T(n)\underline{x}(n)\underline{x}^T(n)]\}\,\underline{R}^{-1} \\
&+ 4\mu^2\,\underline{R}^{-1}\,\underline{p}\,M^T{}_{w(n)} \\
&+ 4\mu^2\,\underline{R}^{-1}\,E\{d(n)\underline{x}(n)d(n)\underline{x}^T(n)\}\,\underline{R}^{-1} \\
&- 4\mu^2\underline{R}^{-1}\,\underline{p}\,\underline{p}^T\,\underline{R}^{-1}
\end{aligned} \qquad (4\text{-}13)$$

Note that, to obtain (4-13) we have used the fact that in (4-12) $\underline{w}(n)$ depends only on $\{d(k),\ \underline{x}(k)\ ;\ k=0\ 1\ \ldots,\ n-1\}$. Moreover, the different index elements of the sequence of data vectors and also the desired signal samples were assumed to be statistically independent. Further evaluations of third, fourth, seventh, and ninth terms on the right hand side of (4-13) require the moment theorem.

Let's denote the Gaussian random variables $x_1, x_2, \ldots, x_N$, with zero-mean, by the column vector $\underline{x}$ where

$$\underline{x} = [\,x_1\ x_2\ \ldots\ x_N\,]^T \qquad (4\text{-}14a)$$

Thus, from [47], in the quadrivariate case, we have

$$E\{ x_1 x_2 x_3 x_4 \} = E\{x_1 x_2\} E\{x_3 x_4\} + E\{x_1 x_3\} E\{x_2 x_4\} + E\{ x_1 x_4 \} E\{ x_2 x_3 \} \tag{4-14b}$$

Applying (4-14b) into the third term of the right hand side of (4-13) we have (Appendix B)

$$E\{ \underline{x}(n)\underline{x}^T(n)\, E[\underline{w}(n)\underline{w}^T(n)]\underline{x}(n)\underline{x}^T(n)\} = \underline{R}E[\underline{w}(n)\underline{w}^T(n)]\underline{R} + \underline{R}E[\underline{w}(n)\underline{w}^T(n)]\underline{R} + \underline{R}\, Tr\{\underline{R}\, E[\underline{w}(n)\underline{w}^T(n)]\} \tag{4-15}$$

Similarly, applying (4-14b) to the fourth term of the right hand side of (4-13) (Appendix B) results in

$$E\{\underline{x}(n)\, \underline{x}^T(n)\, \underline{M}_{w(n)} d(n)\underline{x}^T(n)\} = \underline{R}\, \underline{M}_{w(n)}\underline{p} + \underline{p}\, \underline{M}^T_{w(n)}\, \underline{R} + \underline{R}\, \underline{M}^T_{w(n)}\, \underline{p} \tag{4-16}$$

Since the seventh term of the right hand side of (4-13) is the transpose matrix of the fourth term, thus

$$E\{ \underline{x}(n)d(n)\, \underline{M}^T_{w(n)}\, \underline{x}(n)\underline{x}^T(n) \} = \underline{p}\, \underline{M}^T_{w(n)}\, \underline{R} + \underline{R}\, \underline{M}_{w(n)}\, \underline{p}^T + \underline{p}^T\underline{M}_{w(n)}\, \underline{R} \tag{4-17}$$

Note that, in (4-17) we have

$$\underline{M}^T_{w(n)}\underline{p} = \underline{p}^T\underline{M}_{w(n)} \tag{4-18}$$

which is a scalar value. Finally, from (4-14b), the ninth term of the right hand side of (4-13) becomes as

$$E\{\, d(n)\underline{x}(n)\underline{x}^T(n)d(n)\,\}$$
$$= 2\,\underline{p}\,\underline{p}^T + E[d(n)d(n)]\,\underline{R} \qquad (4\text{-}19)$$

Applying (4-15) - (4-19) to (4-13), we have the weight covariance matrix recursive equaton, i.e.,

$$\begin{aligned}
&Cov\{\underline{w}(n+1)\underline{w}^T(n+1)\} \\
&= (1-4\mu+8\mu^2)\, Cov[\underline{w}(n)\underline{w}^T(n)] \\
&\quad + 4\mu^2\, Tr\{\, \underline{R}\, Cov[\underline{w}(n)\underline{w}^T(n)]\,\}\, \underline{R}^{-1} \\
&\quad + 4\mu^2\underline{R}^{-1}\, \{\, E[d^2(n)] - \underline{M}^T{}_{w(n)}\underline{p} - \underline{p}^T\underline{M}_{w(n)} \\
&\quad + \underline{M}^T{}_{w(n)}\, \underline{R}\, \underline{M}_{w(n)}\,\} \\
&\quad + 4\mu^2\underline{R}^{-1}\{\, \underline{p} - \underline{R}\, \underline{M}_{w(n)}\,\}\{\, \underline{p} - \underline{R}\, \underline{M}_{w(n)}\}^T\, \underline{R}^{-1}
\end{aligned} \qquad (4\text{-}20)$$

Since the K-L transform matrix $\underline{Q}$ diagonalizes the input autocorelation matrix, $\underline{R}$, does not diagonalize the weight covariance matrix, so that further evaluation of the transient weight covariance matrix is not available. However, in the steady-state, further evaluation for the weight covariance matrix can be performed.

4.4.3 The Steady-state Mean-square Error (MSE)

The instantaneous MSE, ξ_n, at nth sampling instant, from chapter 3, can be written as

$$\begin{aligned}
\xi_n &= E[\, e^2(n)\,] \\
&= E[d^2(n)] - \underline{M}^T{}_{w(n)}\, \underline{p}^T\underline{M}_{w(n)} \\
&\quad + Tr\{\underline{R}\, E[\underline{w}(n)\, \underline{w}^T(n)]\}
\end{aligned} \qquad (4\text{-}21)$$

The minimum MSE, ξ_{min}, (2-5) is given by

$$\xi_{min} = E\{ [d(n) - \underline{w}_{opt}\, \underline{x}(n)]^2 \}$$
$$= E[d^2(n)] - \underline{p}^T \underline{w}_{opt} \qquad (4\text{-}22)$$

where the optimal weight vector, $\underline{w}_{opt}$, is defined as

$$\underline{w}_{opt} = \underline{R}^{-1}\, \underline{p} \qquad (4\text{-}23)$$

Substituting (4-22) and (4-23) into (4-21), we have

$$\xi_n = \xi_{min} + [\underline{M}_{w(n)} - \underline{w}_{opt}]^T \underline{R} [\underline{M}_{w(n)} - \underline{w}_{opt}]$$
$$+ Tr\{\underline{R}\ Cov[\underline{w}(n)\ \underline{w}^T(n)]\} \qquad (4\text{-}24)$$

In the steady-state, i.e., when n tends to infinity, the mean weight vector, from (4-11), will be

$$\lim_{n\to\infty} \underline{M}_{w(n)} = \underline{w}_{opt} \qquad (4\text{-}25a)$$

because

$$2\mu \sum_{i=0}^{\infty} (1-2\mu)^i \underline{R}^{-1} \underline{p} = 2\mu \{ \frac{1}{2\mu} \underline{R}^{-1}\ \underline{p} \}$$
$$= \underline{R}^{-1}\ \underline{p}$$
$$= \underline{w}_{opt} \qquad (4\text{-}25b)$$

Also, the weight covariance matrix, has the property, i.e.,

$$Cov[\underline{w}(n+1)\underline{w}^T(n+1)] = Cov[\ \underline{w}(n)\underline{w}^T(n)] \qquad (4\text{-}25c)$$

Since the third term of the right hand side of (4-13) can be written as

$$4\mu^2\ \underline{R}^{-1} \{\ E[d^2(n)] - \underline{M}^T_{w(n)} \underline{p} - \underline{p}^T\ \underline{M}_{w(n)} + \underline{M}^T_{w(n)} \underline{R} \underline{M}_{w(n)} \}$$
$$= 4\mu^2\ \underline{R}^{-1} \{ \xi_{min} + [\underline{M}_{w(n)} - \underline{w}_{opt}]^T \underline{R}\ [\ \underline{M}_{w(n)} - \underline{w}_{opt}] \}$$
(4-26)

therefore, in steady-state, (4-26) is reduced to $4\mu^2\ \underline{R}^{-1}$. Applying this result and using (4-24), (4-25) and (4-26) in (4-20), the steady-state weight covariance matrix recursive equation, can be written as

$$\underline{O} = (-4\mu + 8\mu^2)\ Cov[\underline{w}(n)\underline{w}^T(n)] + 4\ \mu^2 Tr\{\ \underline{R}$$
$$\cdot\ Cov[\underline{w}(n)\underline{w}^T(n)]\} + 4\mu^2\ \xi_{min}\ \underline{R}^{-1} \qquad (4\text{-}27)$$

where $\underline{0}$ is the null matrix. Recall that, the input autocorrelation matrix, $\underline{R}$, can be represented as

$$\underline{R} = \underline{Q}\, \underline{R}_\lambda\, \underline{Q}^{-1}$$

or

$$\begin{aligned}\underline{R}_\lambda^{-1} &= \underline{Q}^{-1}\, \underline{R}^{-1} \underline{Q} \\ &= \underline{Q}^T\, \underline{R}^{-1}\, \underline{Q} \end{aligned} \qquad (4\text{-}28)$$

Pre- and post- multiplying (4-27) by $\underline{Q}^T$ and $\underline{Q}$ yields

$$\begin{aligned}\underline{0} = &- 4\mu(1-2\mu)\underline{Q}^T \mathrm{Cov}[\, \underline{w}(n)\, \underline{w}^T(n)\,]\, \underline{Q} \\ &+ 4\mu^2 \underline{Q}^T \mathrm{Tr}\{\underline{Q}\, \underline{R}_\lambda\, \underline{Q}^T\, \mathrm{Cov}[\underline{w}(n)\underline{w}^T(n)]\}\underline{Q}\, \underline{R}_\lambda^{-1} \\ &+ 4\,\mu^2\, \underline{R}_\lambda^{-1} \xi_{min} \\ = &-4\mu(1-2\mu)\, \underline{A} + 4\mu^2 \mathrm{Tr}\,\{\underline{A}\, \underline{R}_\lambda\}\underline{R}_\lambda^{-1} + 4\mu^2\, \underline{R}_\lambda^{-1} \xi_{min} \end{aligned} \qquad (4\text{-}29)$$

where $\underline{A}$ is an NxN matrix and is defined by

$$\underline{A} = \underline{Q}^T\, \mathrm{Cov}[\, \underline{w}(n)\, \underline{w}^T(n)\,]\, \underline{Q} \qquad (4\text{-}30)$$

Note that, we have used the fact that the diagonal terms of $\underline{R}_\lambda \underline{A}$ equal to the diagonal terms of $\underline{A}\, \underline{R}_\lambda$ for any matrix $\underline{A}$ and diagonal matrix $\underline{R}_\lambda$. Since $\underline{Q}^T\, \underline{Q} = \underline{I}$ and the trace is a scalar, so that the following identity holds, i.e.,

$$\begin{aligned}&\underline{Q}^T\, \mathrm{Tr}\{\underline{Q}\, \underline{R}_\lambda\, \underline{Q}^T\, \mathrm{Cov}[\underline{w}(n)\underline{w}^T(n)]\}\, \underline{Q} \\ &= \mathrm{Tr}\{\, \underline{Q}^T\underline{Q}\, \underline{R}_\lambda\, \underline{Q}^T\, \mathrm{Cov}[\, \underline{w}(n)\underline{w}^T(n)]\, \underline{Q}\} \\ &= \mathrm{Tr}\{\, \underline{R}_\lambda\, \underline{A}\} \end{aligned} \qquad (4\text{-}31)$$

Observe from (4-29) that the second and the third terms on the right hand side are diagonal matrices which implies that the symmetric matrix, $\underline{A}$ is a diagonal matrix, i.e.,

$$\underline{A} = \mathrm{diag}\ \{\, a^1\ a^2\ \ldots\ a^N\,\} \qquad (4\text{-}32)$$

Now, in order to obtain exact expression for $\underline{A}$ and the steady-state MSE, (4-29) can be written in the vector form. We first

define

$$\underline{a} = [a^1 \; a^2 \; \ldots \; a^N]^T, \tag{4-33a}$$

$$\underline{\lambda} = [\lambda_1 \; \lambda_2 \; \ldots \; \lambda_N]^T, \tag{4-33b}$$

and

$$\underline{\lambda}^{-1} = [\lambda_1^{-1} \quad \lambda_2^{-1} \quad \ldots \; \lambda_N^{-1}]^T \tag{4-33c}$$

as vectors corresponding to the diagonal matrices, $\underline{A}$, $\underline{R}_\lambda$, and $\underline{R}_\lambda^{-1}$, respectively. Therefore, equivalently, we can write (4-29) as

$$\underline{0} = -\mu(1-2\mu)\,\underline{a} + \mu^2 \sum_{i=1}^{N} (a^i \lambda_i + \xi_{min})\underline{\lambda}^{-1}$$

or

$$\mu^2 \xi_{min} \underline{\lambda}^{-1} = \underline{H}\,\underline{a} \tag{4-34}$$

where the matrix $\underline{H}$ is defined by

$$\underline{H} = \underline{B} + \underline{U}\,\underline{S}\,\underline{V}^T \tag{4-35a}$$

with

$$\underline{B} = \mu(1-2\mu)\,\underline{I}, \quad \underline{U} = \mu\,\underline{\lambda}^{-1}$$

$$\underline{V} = -\mu\,\underline{\lambda}, \text{ and } \quad \underline{S} = \underline{I} \tag{4-35b}$$

The vector $\underline{a}$ can be obtained by multiplying the inverse of $\underline{H}$ to both sides of (4-34), i.e.,

$$\underline{a} = \underline{H}^{-1} \mu^2 \xi_{min} \underline{\lambda}^{-1} \tag{4-36}$$

Now, to obtain the inverse of $\underline{H}$, the Woodburg's identity is employed, i.e.,

$$\underline{H}^{-1} = (\underline{B} + \underline{U}\,\underline{S}\,\underline{V}^T)^{-1}$$

$$= \underline{B}^{-1} - \underline{B}^{-1}\underline{U}(\underline{S}^{-1} + \underline{V}^T\underline{B}^{-1}\underline{U})^{-1}\underline{V}^T\underline{B}^{-1} \tag{4-37a}$$

or

$$\underline{H}^{-1}\ \underline{\lambda}^{-1} = \frac{1}{1-\sum_{i=1}^{N}\frac{\mu}{1-\mu 2}}\left(\ 1/\{\ \mu(1-2\mu)\}\right)\ (\underline{\lambda}^{-1})^{T} \qquad (4\text{-}37b)$$

Substituting (4-37b) in (4-36), we have

$$\underline{a} = \frac{\mu\ \xi_{min}}{1-\sum_{i=1}^{N}\frac{\mu}{1-\mu 2}}\{1/(1-2\mu)\ \}(\underline{\lambda}^{-1})^{T} \qquad (4\text{-}38)$$

Now, from (4-38), the elements of $\underline{a}$ are the eigenvalues of the steady-state covariance matrix. Therefore, the covariance matrix can be represented by

$$\lim_{n\longrightarrow\infty} \mathrm{Cov}[\underline{w}(n)\ \underline{w}^{T}(n)] = \underline{Q}\ \underline{A}\ \underline{Q}^{-1} \qquad (4\text{-}39)$$

Finally, the steady-state MSE, is obtained if we apply (4-39) and (4-31) into (4-24), i.e.,

$$\lim_{n\to\infty}\xi_{n} = \xi_{min} + \sum_{i=1}^{N} a^{i}\lambda_{i}$$

$$= \frac{\xi_{min}}{1-\sum_{i=1}^{N}\frac{\mu}{1-\mu 2}} \qquad (4\text{-}40)$$

The steady-state MSE of the time domain LMS algorithm is given by (chapter 2).

$$\lim_{n\to\infty}\xi_{n} = \frac{\xi_{min}}{1-\sum_{i=1}^{N}\frac{\mu}{1-\mu 2\lambda_{i}}} \qquad (4\text{-}41)$$

Comparing (4-40) with (4-41), we recognize that (4-40) can be viewed as a special case of (4-41), in which all the eigenvalues are unity. This agree quite well as discussed in chapter 2 with the generalized Newton-Raphson method. That is the rescaling performance surface in the normalized K-L transformed domain is a spherical contour. Furthermore, (4-40) means taht the steady-state MSE of the normalized K-L TDLMS algorithm is independent of the eigenvalues of the input autocorrelation matrix which matches the result in Sec.4.3.1 for the time constant, t_i. However, this does not mean that the steady-state MSE of the normalized K-L TDLMS adaptive algorithm is smaller than the conventional time domain LMS algorithm. It only means that using such algorithm the disparity of the eigenvalues does not affect the value of the steady-state MSE. The steady-state MSE depends only on the value of μ, the constant step-size.

4.5 Statistical Analysis of the Normalized K-L TDLMS Algorithm with Complex Valued Data

In this section the normalized K-L TDLMS algorithm with complex valued data is analyzed. The derivation of the complex normalized K-L TDLMS algorithm is similar to the real normalized K-L TDLMS algorithm. The weight vector updated equation is obtained by minimizing the mean square error, ξ_n,

$$\xi_n = E[e(n)e^*(n)]$$

$$= E[d(n)d^*(n)] - \underline{M}^*_{w(n)}\underline{p} - \underline{p}^*\underline{M}_{w(n)} + \mathrm{Tr}\{\underline{R}\, E[\underline{w}(n)\,\underline{w}^*(n)]\} \qquad (4\text{-}42)$$

Thus, following similar procedures we have the weight vector update equation as :

$$\underline{w}(n+1) = \underline{w}(n) + 2\mu\,\underline{R}^{-1}e(n)\underline{x}^*(n)$$

or

$$\begin{aligned}\underline{w}(n+1) &= \underline{w}(n) + 2\mu\,\underline{R}^{-1}\{[d(n)-y(n)]\,\underline{x}^*(n)\} \\ &= \underline{w}(n) + 2\mu\,\underline{R}^{-1}\{d(n)\underline{x}^*(n)-\underline{x}^*(n)\underline{x}^T(n)\underline{w}(n)\} \\ &= [\underline{I} - 2\mu\,\underline{R}^{-1}\underline{x}^*(n)\underline{x}^T(n)]\,\underline{w}(n) \\ &\quad + 2\mu\,\underline{R}^{-1}d(n)\,\underline{x}^*(n) \qquad (4\text{-}43)\end{aligned}$$

where the quantities, $\underline{w}(n)$, $e(n)$, $\underline{R}$, and $\underline{x}(n)$ are completely analogous to the corresponding quantities of (4-10) but are , in general, complex-valued. Here, the input autocorrelation matrix, $\underline{R}$, and the crosscorrelation vector, $\underline{p}$, are defined by

$$\underline{R} = E[\,\underline{x}^*(n)\,\underline{x}^T(n)\,]$$

and

$$\underline{p} = E[\,d(n)\underline{x}^*(n)\,]$$

respectively. The theoretical analysis of the time domaim complex LMS adaptive algorithm has been done by Horowitz and Senne [41] and Fisher [42]. Using the same assumptions that the input signal and the desired response signal are jointly Gaussian, complex valued, statistically independent processes, we can analyze the complex normalized K-L TDLMS adaptive algorithm in the time domain.

Again, to obtain the MSE, the mean weight vector and weight covariance matrix recursive equations must be derived.

These are derived in the subsequent sections.

4.5.1 The Mean Weight Vector Recursive Equation

The mean weight vector recursive equation for the complex normalized K-L TDLMS algorithm can be obtained by using the assumption that the successive input vectors $\underline{x}(n)$ and $\underline{x}(n+1)$ are statistical independent. Also, from (4-43), $\underline{w}(n)$ depends only on $\{ d(k), \underline{x}(k) ; k=0,1,\ldots,n-1\}$ so that the mean weight vector recursive equation can be written as

$$\begin{aligned} \underline{M}_{w(n+1)} &= E[\underline{w}(n+1)] \\ &= \{\underline{I} - 2\mu \underline{R}^{-1} E[\underline{x}^*(n)\underline{x}^T(n)]\} E[\underline{w}(n)] \\ &\quad + 2\mu \underline{R}^{-1} E[d(n) \underline{x}^*(n)] \\ &= (1-2\mu) \underline{M}_{w(n)} + 2\mu \underline{R}^{-1}\underline{p} \end{aligned} \qquad (4\text{-}44a)$$

Again, starting with $E[\underline{w}(0)]$, the initial mean weight vector, and iteratively substituting in (4-44a) results in

$$\underline{M}_{w(n+1)} = (1-2\mu)^{n+1} E[\underline{w}(0)] + 2\mu \sum_{i=0}^{n} (1-2\mu)^i \underline{R}^{-1}\underline{p} \qquad (4\text{-}44b)$$

Also, in steady-state, i.e., n tends to infinity, from (4-44b) the mean weight vector is given by

$$\begin{aligned} \lim_{n\to\infty} E[\underline{w}(n)] &= \underline{R}^{-1} \underline{p} \\ &= \underline{w}_{opt} \end{aligned} \qquad (4\text{-}45)$$

This is because in the steady-state, the first term of right hand side of (4-44) will vanish, provided that $\mu < 1/2$, and the second term is reduced to the optimal weight vector.

4.5.2 The Weight Covariance Matrix Recursive Equation

The derivation of weight covariance matrix recursive equation of the complex normalized K-L TDLMS algorithm is also very similar to the real normalized K-L TDLMS algorithm The major difference is in the Gaussian Moment Factoring Theorem. This theorem states [31] that for given complex Gaussian random variables, x_1, x_2, x_3, and x_4, the following relationship holds,

$$E[x_1 x^*_2 x_3 x^*_4] = E[x_1 x^*_2]\, E[x_3 x^*_4] + E[x_1 x^*_4]\, E[x^*_2 x_3] \qquad (4\text{-}46)$$

This theorem statement is in contrast with the real variable case which is described in (4-14b). Notice that one less term appears on the right hand side of (4-46) than on the right hand side of (4-14b) which results in superior mean-square convergence [41].

Now, consider the weight covariance matrix, Cov [$\underline{w}(n+1)$ $\underline{w}^*(n+1)$]:

$$\text{Cov}[\,\underline{w}(n+1)\,\underline{w}^*(n+1)\,] = E\{[\underline{w}(n+1)-\underline{M}_{w(n+1)}][\,\underline{w}(n+1)-\underline{M}_{w(n+1)}]^*\} \qquad (4\text{-}47)$$

Substituting (4-43) and (4-45) into (4-47) and after some manipulation, we obtain

$$\begin{aligned}\text{Cov}[\underline{w}(n+1)\,\underline{w}^*(n+1)] &= \text{Cov}[\underline{w}(n)\underline{w}^*(n)] - 4\mu\,\text{Cov}[\underline{w}(n)\,\underline{w}^*(n)] \\ &+ 4\mu^2\underline{R}^{-1}E\{\underline{x}^*(n)\underline{x}^T(n)E[\underline{w}(n)\underline{w}^*(n)]\underline{x}^*(n)\underline{x}^T(n)\}\underline{R}^{-1} \\ &- 4\mu^2\underline{R}^{-1}E\{\underline{x}^*(n)\underline{x}^T(n)\,E[\underline{w}(n)]d^*(n)\underline{x}^T(n)\}\underline{R}^{-1}\end{aligned}$$

$$+ 4\mu^2 \underline{M}_{w(n)} \underline{p}^{\#} \underline{R}^{-1} - 4\mu^2 \underline{M}_{w(n)} \underline{M}^{\#}{}_{w(n)}$$
$$- 4\mu^2 \underline{R}^{-1} E\{ d(n)\underline{x}^*(n) E[\underline{w}^{\#}](n)\underline{x}^*(n)\underline{x}^T(n)\ \underline{R}^{-1}$$
$$+ 4\mu^2 \underline{R}^{-1}\ \underline{p}\ \underline{M}^{\#}{}_{w(n)} \}$$
$$+ 4\mu^2\ \underline{R}^{-1}\ E\{d(n)\underline{x}^*(n)d^*(n)\underline{x}^T(n)\}\ \underline{R}^{-1}$$
$$- 4\mu^2\ \underline{R}^{-1}\ \underline{p}\ \underline{p}^{\#}\ \underline{R}^{-1}$$

(4-48)

Again, in obtaining (4-48) we have used the fact that in (4-43) $\underline{w}(n)$ depends only on $\{ d(k), \underline{x}(k) ; k = 0, 1, \ldots, n-1 \}$. Moreover, the different index elements of the sequence of data vectors and also desired signal samples were assumed to be statistically independent. Further evaluation of third, fourth, seventh, and ninth terms of right hand side of (4-48) require the moment theorem which was described in (4-46).

Now, applying (4-46) to the third term of right hand side of (4-48) we have (Appendix C)

$$E\{\underline{x}^*(n)\underline{x}^T(n)\ E[\underline{w}(n)\underline{w}^{\#}(n)]\underline{x}^*(n)\underline{x}^T(n)\}$$
$$= \underline{R}E[\underline{w}(n)\underline{w}^{\#}(n)]\underline{R} + \underline{R}\ Tr\{\underline{R}\ E[\underline{w}(n)\underline{w}^{\#}(n)\} \quad (4-49)$$

Similarly, applying (4-46) to the fourth term of right hand side of (4-48) (Appendix C), we have

$$E\{\underline{x}^*(n)\underline{x}^T(n)\ \underline{M}_{w(n)} d^*(n)\underline{x}^T(n)\}$$
$$= \underline{R}\ \underline{M}_{w(n)}\ \underline{p}^{\#} + \underline{p}^{\#}\ \underline{M}^T{}_{w(n)}\ \underline{R}\ \underline{p}$$

(4-50)

Again, since the seventh term of right hand side of (4-48) is the transpose matrix of the fourth term, thus

$$E\{\ \underline{x}^*(n)d(n)\ \underline{M}^{\#}{}_{w(n)}\ \underline{x}^*(n)\underline{x}^T(n)\ \}$$
$$= \underline{p}\ \underline{M}^{\#}{}_{w(n)}\ \underline{R} + \underline{R}\ \underline{M}^{\#}{}_{w(n)}\ \underline{p} \quad (4-51)$$

Finally, from (4-46), the ninth term of right hand side of (4-48) will be

$$E\{\, d(n)\underline{x}^*(n)\underline{x}^T(n)d^*(n)\,\} = 2\,\underline{p}\,\underline{p}^\sharp + E[d(n)d^*(n)]\,\underline{R} \qquad (4\text{-}52)$$

Now, apply the results from (4-49) - (4-52) to (4-48) yields the weight covariance matrix recursive equation, i.e.,

$$\begin{aligned} &Cov\{\underline{w}(n+1)\underline{w}^\sharp(n+1)\} \\ &= (1-4\mu+4\mu^2)\, Cov[\underline{w}(n)\underline{w}^\sharp(n)] \\ &\quad + 4\mu^2\, Tr\{\, \underline{R}\cdot E[\underline{w}(n)\underline{w}^\sharp(n)]\,\}\,\underline{R}^{-1} \\ &\quad + 4\mu^2\underline{R}^{-1}\{E[d(n)d^*(n)] - \underline{M}^\sharp_{w(n)}\underline{p} - p^\sharp \underline{M}_{w(n)}\} \end{aligned}$$

or

$$\begin{aligned} &Cov\{\underline{w}(n+1)\underline{w}^\sharp(n+1)\} \\ &= (1-4\mu+4\mu^2)\, Cov[\underline{w}(n)\underline{w}^\sharp(n)] \\ &\quad + 4\mu^2\, Tr\{\, \underline{R}\, Cov[\underline{w}(n)\underline{w}^\sharp(n)]\,\}\,\underline{R}^{-1} \\ &\quad + 4\mu^2\underline{R}^{-1}\{E[d(n)d^*(n)] - \underline{M}^\sharp_{w(n)}\underline{p} - p^\sharp \underline{M}_{w(n)} \\ &\qquad + \underline{M}^\sharp_{w(n)}\,\underline{R}\,\underline{M}_{w(n)}\} \end{aligned} \qquad (4\text{-}53)$$

In obtaining (4-53), we have used the relationship

$$\begin{aligned} &Tr\{\, \underline{R}\, E[\underline{w}(n)w^\sharp(n)]\} \\ &= Tr\{\, \underline{R}\, \{\, Cov[\underline{w}(n)\underline{w}^\sharp(n)] + \underline{M}_{w(n)}\underline{M}^\sharp_{w(n)}\,\}\,\} \end{aligned} \qquad (4\text{-}54)$$

Similarly, from (4-42), the complex valued minimum MSE, ξ_{min} is given by

$$\begin{aligned} \xi_{min} &= E\{[d(n)-\underline{w}^T_{opt}\underline{x}(n)]\,[d(n)-\underline{w}^T_{opt}\underline{x}(n)]^*\} \\ &= E[d^*(n)d(n)] - \underline{w}^\sharp_{opt}\,\underline{p} \end{aligned} \qquad (4\text{-}55)$$

where the optimal weight vector, $\underline{w}_{opt}$, is defined as

$$\underline{w}_{opt} = \underline{R}^{-1}\,\underline{p} \qquad (4\text{-}56)$$

Now, for further evaluation of the weight covariance matrix recursive equation in (4-53), we introduce a scalar variable, $k_0(n)$, at nth sampling instant and is defined as

$$k_0(n) = \underline{M}^{*}_{w(n)} \underline{R} \underline{M}_{w(n)} - \underline{M}^{*}_{w(n)} \underline{P} - \underline{P}^{*} \underline{M}_{w(n)} + E[d(n)d^{*}(n)]$$
$$= \xi_{min} + \{E[\underline{w}(n)] - \underline{w}_{opt}\}^{*} \underline{R} \{E[\underline{w}(n)] - \underline{w}_{opt}\} \quad (4\text{-}57)$$

It has been shown by Fisher [42] that in the complex LMS adaptive algorithm, the $\underline{Q}$ matrix diagonalizes the covariance matrix if $\underline{Q}$ matrix is used to diagonalize the initial weight covariance matrix. It can be shown that this is also true for the complex normalized K-L TDLMS algorithm (Appendix D). Now, in order to further evaluate the weight covariance matrix recursive equation, we let

$$\underline{\Gamma}(n) = \text{diag}[\, r_1(n)\ r_2(n) \ \ldots\ r_N(n)] \quad (4\text{-}58)$$

denote the diagonal matrix having the eigenvalues of $\text{Cov}[\underline{w}(n)\ \underline{w}^{*}(n)]$ as its diagonal elements. Pre and post multiplying (4-53) by $\underline{Q}^{*}$ and $\underline{Q}$, we have

$$\underline{\Gamma}(n+1) = \underline{\Gamma}(n)(1-2\mu)^2) + 4\mu^2 k_0(n)\, \underline{R}_{\lambda}^{-1} + 4\mu^2 \{\underline{\Gamma}(n)\, \underline{R}_{\lambda}\}\, \underline{R}_{\lambda}^{-1} \quad (4\text{-}59)$$

In order to obtain exact expression for $\underline{\Gamma}(n)$ we define

$$\underline{r}(n) = [\, r_1(n)\ r_2(n) \ \ldots\ r_N(n)]^T \quad (4\text{-}60a)$$

$$\underline{\lambda} = [\, \lambda_1 \ \ \lambda_2 \ \ldots \lambda_N\,]^T \quad (4\text{-}60b)$$

$$\underline{\lambda}^{-1} = [\, \lambda_1^{-1} \ \ \lambda_2^{-1} \ \ldots \lambda_N^{-1}]^T \quad (4\text{-}60c)$$

as vectors corresponding to the diagonal matric, $\underline{\Gamma}(n)$, $\underline{R}_\lambda$, and $\underline{R}_\lambda^{-1}$, respectively. Therefore, equivalently, (4-59) can be written as

$$\begin{aligned}\underline{r}(n+1) &= \{(1-2\mu)^2+4\mu^2\underline{\lambda}^{-1}\underline{\lambda}^T\}\underline{r}(n) + 4\mu^2 k_0(n)\underline{\lambda}^{-1} \\ &= \{(1-2\mu)^2\,\underline{I}+4\mu^2\underline{\lambda}^{-1}\underline{\lambda}^T\}\underline{r}(n) \\ &\quad + 4\mu^2 k_0(n)\underline{\lambda}^{-1} \qquad (4\text{-}61)\end{aligned}$$

Now, again, define the matrix $\underline{H}$ as

$$\begin{aligned}\underline{H} &= (4\mu-4\mu^2)\underline{I} - 4\mu^2\underline{\lambda}^{-1}\underline{\lambda}^T \\ &= \underline{B} + \underline{U}\,\underline{S}\,\underline{V}^T \qquad (4\text{-}62a)\end{aligned}$$

with

$$\underline{B} = 4\mu(1-\mu)\,\underline{I}, \quad U = 2\mu\,\underline{\lambda}^{-1}$$

$$\underline{V} = -2\mu\,\underline{\lambda}, \text{ and } \underline{S} = \underline{I} \qquad (4\text{-}62b)$$

then (4-61) has the solution:

$$\begin{aligned}\underline{r}(n+1) &= (\underline{I}-\underline{H})^n\,\underline{r}(n) + 4\mu^2 k_0(n)\underline{\lambda}^{-1} \\ &= (\underline{I}-\underline{H})^n\underline{r}(0) + 4\mu^2\sum_{i=0}^{n} k_0(n-i-1)(\underline{I}-\underline{H})^i\underline{\lambda}^{-1} \\ &\qquad n=0,1,\ldots \qquad (4\text{-}63)\end{aligned}$$

where $\underline{r}(0)$ is the initial eigenvalues of $\mathrm{Cov}[\underline{w}(0)\underline{w}^*(0)]$. Since the spectral representation of $\mathrm{Cov}\,[\underline{w}(n)\underline{w}^*(n)]$ can be expressed as

$$\begin{aligned}&\mathrm{Cov}[\underline{w}(n)\underline{w}^*(n)] \\ &= \underline{Q}\,\underline{\Gamma}(n)\,\underline{Q}^* \\ &= \sum_{i=1}^{N} r_i(n)\,\underline{q}_i\underline{q}^*{}_i \qquad (4\text{-}64)\end{aligned}$$

which explicitly display the dependence on $r_i(n)$ and $\underline{q}_i$, $i=0,1,\ldots,N$.

4.5.3 The Steady-state Mean-square Error (MSE)

Now, we can evaluate the MSE by substituting the results of the mean weight vector and weight covariance matrix into (4-42). Before we do that, let us substitute (4-55) and (4-56) into (4-43) which results in

$$\xi_n = \xi_{min} + [\underline{M}_{\underline{w}}(n) - \underline{w}_{opt}]^{\ddagger} \underline{R} [\underline{M}_{\underline{w}}(n) - \underline{w}_{opt}] + Tr\{ \underline{R} \, Cov[\underline{w}(n) \underline{w}^{\ddagger}(n)]\} \qquad (4\text{-}65)$$

Also, if we substitute (4-57) into (4-65), we can have

$$\xi_n = k_0(n) + Tr\{ \underline{R} \, Cov[\underline{w}(n)\underline{w}^{\ddagger}(n)]\} \qquad (4\text{-}66)$$

From (4-66), it is clear that the MSE is the function of minimum MSE, the error weight vector, the autocorrelation matrix and the weight covariance matrix. As mentioned earlier the two measures of the convergence properties are related. Specifically, in the first term of right hand side of (4-66), if the present estimated mean weight vector converges slowly to the optimal weight vector, then the MSE will be larger than the one which has faster convergence rate. This is because the first term is greater than zero. On the other hand, when the mean weight vector approaches to the Wiener weight vector the second term of right hand side of (4-66) will be dominated which is due to the weight covariance matrix. Now, if $\underline{r}(n)$ converges, then we have

$$\lim_{n \to \infty} \underline{r}(n) = 4\mu^2 k_0(\infty) \underline{H}^{-1} \underline{\lambda}^{-1} \qquad (4\text{-}67)$$

To obtain the inverse of $\underline{H}$ in (4-67) the Woodburg's identity

is employed, i.e.,

$$\underline{H}^{-1} = (\underline{B} + \underline{U}\,\underline{S}\,\underline{V}^T)^{-1}$$
$$= \underline{B}^{-1} - \underline{B}^{-1}\underline{U}\,(\underline{S}^{-1} + \underline{V}^T\underline{B}^{-1}\underline{U})^{-1}\underline{V}^T\underline{B}^{-1}$$

(4-68)

Substituting (4-68) into (6-67) we have steady-state $\underline{r}(n)$, i.e.,

$$\lim_{n\to\infty} \underline{r}(n) = \frac{\mu\ \xi_{min}}{1 - \sum_{i=1}^{N} \frac{\mu}{1-\mu}} \{1/(1-\mu)\}(\underline{\lambda}^{-1})^T$$

(4-69)

Applying this result in (4-64), the steady-state weight covariance matrix recursive equation, can be written as

$$\lim_{n\to\infty} \mathrm{Cov}[\underline{w}(n)\underline{w}^{\ddagger}(n)]$$
$$= \frac{\xi_{min}}{1 - \sum_{i=1}^{N} \frac{\mu}{1-\mu}} \{ \sum_{i=1}^{N} \frac{\mu\ \lambda_i^{-1}}{1-\mu}\, \underline{q}_i \underline{q}^{\ddagger}_i \}$$

(4-70)

Finally, the steady-state MSE can be obtained if we apply (4-58) and (4-70) into (4- 66):

$$\lim_{n\to\infty} \xi_n = \xi_{min} + \lim_{n\to\infty} \mathrm{Tr}\{ \underline{R}\ \mathrm{Cov}[\underline{w}(n)\underline{w}^{\ddagger}(n)]$$
$$= \xi_{min} + \lim_{n\to\infty} \sum_{i=1}^{N} \lambda_i\ r_i(n)$$

or

$$\lim_{n\to\infty} \xi_n = \xi_{min} + \frac{\xi_{min}}{1 - \sum_{i=1}^{N} \frac{\mu}{1-\mu}} \left[\sum_{i=1}^{N} \frac{\mu}{1-\mu} \right] \qquad (4\text{-}71)$$

Again, the steady-state MSE (4-71) for the complex normalized K-L TDLMS algoritm is independent of the eigenvalues of the input autocorrelation matrix. That is, in the complex normalized K-L TDLMS algorithm the disparity of eigenvalues does not affect the value of the steady-state MSE which depends only on the value of μ, the constant step-size and the minimum MSE.

4.6 Conclusions

In this chapter, the statistical behaviour for both real valued data and complex valued data of the normalized K-L TDLMS adaptive algorithm are investigated.

In general, the convergence rate can be measured in terms of error weight vector and MSE. We showed that the convergence rate in terms of error weight vector is independent of the eigenvalues of the input autocorrelation matrix. Consequently, the bound of the stepsize can be predetermined. This is not the case for time domain LMS adaptive algorithm. On the other hand, the steady-state MSE values which we obtained for both real and complex valued data are shown to depend only on the values of step-size and minimum MSE but not the eigenvalues of the autocorrelation

matrix. This is because in the normalized K-L transformed domain the transformed autocorrelation matrix is an identity matrix.

Since in the real valued data case, the $\underline{Q}$ matrix does not diagonalize the weight covariance matrix recursive equation, further investigation of the transient MSE is not available. Alternatively, for the complex valued data algorithm, the $\underline{Q}$ matrix diagonalizes the weight covariance matrix recursive equation if $\underline{Q}$ matrix diagonalizes the initial weight covariance matrix. This enables us to further study the transient MSE in closed form and use it to compare to the time domain LMS algorithm analytically.

CHAPTER 5

COMPLEX NORMALIZED K-L TDLMS ADAPTIVE FILTERING WITH APPLICATION TO THE ADAPTIVE LINE ENHANCER

5.1 Introduction

The adaptive line-enhancer (ALE) is an adaptive filter used for the detection of narrowband signal in the presence of broadband noise. It was first proposed by Widrow et. al., [8] and extensively discussed by many others [11,42, 50-52]. As shown in figure 5.1, the ALE is implemented with a two-channel processor in which a delayed version of the received signal is adaptively filtered and substracted from the instantaneous received signal. It is well-known that the ALE can detect narrowband signals in the presence of broadband noise and that the ALE can be viewed as a signal separator, i.e., it separates the desired component from the undesired component.

Although the ALE was developed in the context of signal separation, it has been shown [8,11] to be an adaptive implementation of a Δ - step Wiener predictor. Furthermore, the ALE can be conceived as a degenerate form of adaptive noise canceller [8],in which the received signals are delayed and then adapted to match the undelayed signals. Here, the delay, Δ, causes the broad-band signals to decorrelate, whereas, the narrowband signals are still correlated, and thus are passed by the filter introducing a simple phase

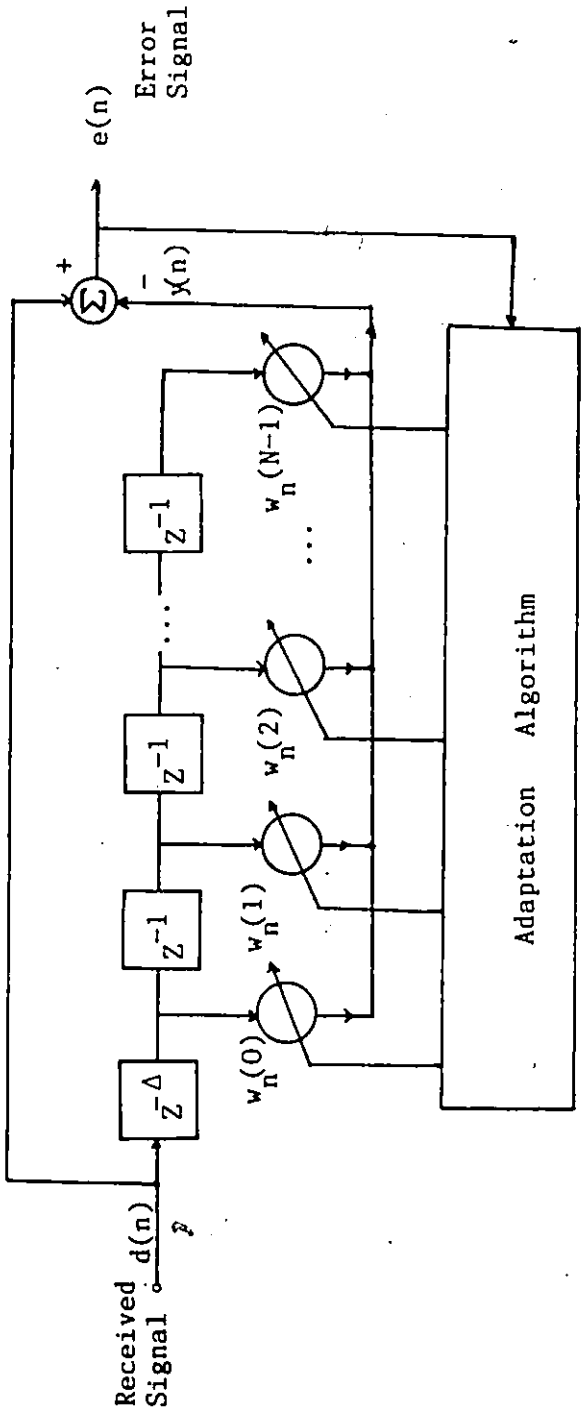


Figure 5.1 Block Digram of Adaptive Line-Enhancer (ALE)

shift between the narrowband signals. Furthermore, in order to improve the spectral resolution , Δ , can be adjusted such that (Δ + N/2) is approximately equal to an integer multiple of the period of the signal [11]. Here, N is the order of adaptive filter.

In ALE, the adaptive filter is seeking to minimize the error power components for the phase shift so that the narrowband components cancel each other at the summing junction. Since it can not compensate for the decorrelation of the noise components, it removes the noise from its own output so that the noise in the primary input appears in the error output.

The theoretical analysis for the ALE has been addressed by many authors [42,50-52]. In order to carry out the theoretical analysis of the ALE, in terms of MSE, certain assumptions for the input signals have to be made. Fisher [42] utilized the complex time domain LMS adaptive filtering algorithm to investigate the behavior of the ALE theoretically. The signal model assumed by Fisher included multiple sinusoids in broadband noise. It is also assumed that each sinusoidal signal is narrowband, statistically independent, orthogonal, Rayleigh fading with equal power. Here, the signal model used by Fisher is extended to the case of multiple sinusoids with equal and un-equal power. The extended signal model is then used to further explore the statistical behaviour of the ALE analytically for the normalized K-L TDLMS adaptive algorithm in the time domain

with complex valued data. It is widely known that, in general, the convergence rate of the LMS algorithm, in terms of MSE is essentially affected by the eigenvalue spread of the input autocorrelation matrix. Here, we will show that the convergence rate of ALE not only depends on eigenvalue spread, but also depends on the ratio between the power of the sinusoids in the received signals when the complex time domain LMS algorithm is employed for the ALE. However, this is not the case for the complex normalized K-L TDLMS ALE in the time domain.

The main objective of this chapter, is to apply the results derived in chapter 4, of the complex normalized K-L TDLMS algorithm in the time domain to the ALE. Under the signal model described earlier we are able to obtain the transient weight covariance matrix of the ALE in closed form. The derived values of the MSE is then used to compare with the MSE of ALE derived by Fisher [42]. For the purpose of comparison, the extended signal model with unequal power is also applied to the complex time domain LMS algorithm for the ALE which is not discussed by Fisher [42].

5.2 Signal Model Description and the Mean Weight Vector

5.2.1 Signal Model of the ALE

Consider an ALE with length N which is used to estimate

L statistically independent, orthogonal, stationary, relatively slow varying Rayleigh fading complex sinusoids in additive circular Gaussian white noise independent of the sinusoids. The i th complex sinusoid can be written in the form, i.e.,

$$s_i(k) = A_i(k) \exp\{ j w_i k T_s \} \qquad k=0,1,2,... \qquad (5\text{-}1a)$$

with

$$j = \sqrt{-1}$$

Here w_i represents the ith carrier frequency and T_s is the sampling interval. Also, $A_i(k)$ denotes a zero-mean circular normal Gaussian [31] complex valued envelope of the i th sinusoid which consists with the general Gaussian assumption described earlier. Further, the complex envelope $A_i(k)$ are assumed slowly varying with the following properties, i.e.,

$$E\{A_i(k)\, A^*_m(n)\} = 0, \quad i \neq m$$

$$E\{A_i(k)\, A^*_i(n)\} = \begin{cases} 0, & \text{if } |k-n| > N \\ \sigma^2_s, & \text{if } |k-n| \leq N \end{cases}$$

$$\text{for } i=1,2,...L \qquad (5\text{-}1b)$$

Now, let

$$d(k) = \sum_{i=1}^{L} s_i(k) + n(k)$$

$$= \sum_{i=1}^{L} A_i(k) \exp\{jw_i kT_s\} + n(k) \quad ;$$

$$k=0,1, ... \qquad (5\text{-}2)$$

denote the received signal of the ALE consisting of L complex sinusoids in additive white noise, uncorrelated with the

sinusoids. Here n(k) is a circularly Gaussian, broadband sequence with the statistical property as follows:

$$E\{ n(k)\, n^*(i) \} = \begin{cases} \sigma^2_n & , \ k = i. \\ 0 & , \ k \neq i. \end{cases} \qquad (5\text{-}3)$$

The assumptions described for above model are such that the ALE operates on narrowband Gaussian sinusoidal inputs with small bandwidth compared to the reciprocal of the filter length, N.

Since the input of the ALE is the delayed version of received signal d(n), we then have

$$E\{ x^*(n)\, x(n-k) \} = \sigma^2_n + \sum_{i=1}^{L} \sigma^2_{s_i} \exp\{ - j\, w_i k T_s \} \quad \text{for } k \leq N \qquad (5\text{-}4a)$$

Consequently, the input autocorrelation matrix, $\underline{R}$, can be expressed as

$$\underline{R} = \sigma^2_n \underline{I} + \sum_{i=1}^{L} \sigma^2_{s_i} \underline{\nu}_i \underline{\nu}^{*}_i \qquad (5\text{-}4b)$$

where $\underline{\nu}_i$; i = 1,2,..., L denote the vectors

$$\underline{\nu}_i = [\, 1 \quad e^{jw_i T_s} \quad \ldots \quad e^{j(N-1)w_i T_s}]^T \qquad i=1,2,\ldots L, \qquad (5\text{-}4c)$$

Similarly, the cross-correlation vector of d(k) and $\underline{x}(k)$ is given by

$$\underline{p} = \sum_{i=1}^{L} \beta_i \ \sigma^2_{s_i} \underline{\nu}_i \qquad (5\text{-}5)$$

where β_i ; i = 1,2,...,L are complex phase factors depending on the lag between x(k) and d(k).

Under the assumption that $\{\underline{\nu}_i\}$ is a set of orthogonal vectors, the eignevalues of $\underline{R}$ are given as

$$\lambda_i = \sigma^2_n + N\sigma^2_{s_i} \qquad i=1,2,\ldots,L \qquad (5\text{-}6a)$$

with associated eigenvectors

$$\underline{q}_i = \underline{\nu}_i / \sqrt{N} \qquad ; \ i=1,2,\ldots,L$$

and

$$\lambda_i = \sigma^2_n \qquad ; \ i = L+1, L+2, \ldots, N \qquad (5\text{-}6b)$$

with associated eigenvetors $\underline{q}_i$; i=L+1, L+2, ... ,N, which form a set of orthonomal vectors and are also orthogonal to $\underline{q}_i$; i=1,2,...,L.

5.2.2 The Mean Weight Vector of the ALE

Recall from (4-44b) that the transient mean weight vector of the complex valued normalized K-L TDLMS algorithm in time domain is given by

$$\underline{M}_w(n+1) = (1-2\mu)^{n+1}\underline{M}_w(0) + 2\mu \sum_{i=0}^{n} (1-2\mu)^i \underline{R}^{-1}\underline{p} \qquad n= 0, 1, \ldots \qquad (4\text{-}44b)$$

Applying the signal model described in sec. 5.2.1, to the mean weight vector of the complex K-L normalized TDLMS algorithm, results in

$$\underline{M}_w(n) = \sum_{i=1}^{L} \beta_i \frac{\sigma^2_{s_i}}{\sigma^2_n + \sigma^2_{s_i} N} \{ 1-(1-2\mu)^n \} \underline{\nu}_i + \sum_{i=1}^{L} h_i (1-2\mu)^n \underline{\nu}_i + \sum_{i=L+1}^{N} h_i (1-2\mu)^n \underline{q}_i \qquad (5\text{-}7)$$

because

$$\underline{R}^{-1}\underline{p} = \sum_{i=1}^{L} \beta_i \frac{\sigma_{s_i}^2}{\sigma_n^2 + \sigma_{s_i}^2 N} \underline{\nu}_i \qquad (5\text{-}8)$$

In (5-8), the scalar $\{ h_i \}$ denotes the projection of $M_{w(n)}$ in the $\{\underline{\nu}_i\}$ and $\{\underline{q}_i\}$ directions. Note that, the steady-state mean weight vector is convergent to the optimal weight vector which is given in (5-8). Furthermore, the transient mean weight vector obtained in this section will be used to evaluate the transient MSE of the ALE in subsequent sections.

5.3 The Transient and Steady-state Weight Covariance Matrices of ALE

The transient MSE can be expressed in terms of the transient mean weight vector and weight covariance matrix as in (4-65). In this section, we explicitly determine the transient weight covariance matrix which can be used to express the transient MSE in closed form. The transient weight covariance matrix can be represented as (4-64),

$$\mathrm{Cov}[\, \underline{w}(n)\, \underline{w}^{\pm}(n)\,] = \sum_{i=1}^{N} r_i(n)\, \underline{q}_i \underline{q}^{\pm}_i \qquad (4\text{-}64)$$

provided that the initial weight covariance matrix is diagonalized by $\underline{Q}$. Here, $r_i(n)$ is the ith component of $\underline{r}(n)$ which is the equivalent vector form of the weight covariance matrix. To obtain $\underline{r}(n)$, we need to solve (4-61) such that

$$\underline{r}(n+1)= (1-2\mu)^2\underline{r}(n) + 4\mu^2\underline{\lambda}^{-1}\underline{\lambda}^T\underline{r}(n) + 4\mu^2\, k_0(n)\underline{\lambda}^{-1} \qquad (4\text{-}61)$$

In what follows, the weight covariance matrix is derived for the cases of multiple sinusoids with equal power and unequal power.

5.3.1 Multiple Sinusoids with Equal Power

In the following, we assume that : (1) the power of the sinusoids is equal, that is, $\sigma^2_{s_i} = \sigma^2_s$; i = 1, 2, ...,L, and (2) uniform initial condition on $\underline{r}(0)$ apply; i.e., $r_i(0) = r_k(0)$; i,k = 1,2,...,N.

In order to carry out the study, we rewrite (4-61) as

$$\begin{aligned}\underline{r}(n+1)&= (1-2\mu)^2\underline{r}(n) + 4\mu^2\underline{\lambda}^{-1}\underline{\lambda}^T\underline{r}(n) + 4\mu^2\, k_0(n)\underline{\lambda}^{-1}\\ &= (1-2\mu)^2\underline{r}(n) + 4\mu^2\,\underline{\lambda}^{-1}\sum_{i=1}^{N}\lambda_i r_i(n) + 4\mu^2 k_0(n)\,\underline{\lambda}^{-1}\end{aligned} \qquad (5\text{-}9)$$

In the case of equal power, (5-9) can be written with reduced rank, as the pair of scalar difference equations in two unknowns:

$$r_1(n+1) = (1-2\mu)^2 r_1(n) + 4\mu^2\,\lambda_1^{-1}\sum_{i=1}^{N}\lambda_i r_i + 4\mu^2\, k_0(n)\,\lambda_1^{-1}$$

and

$$r_2(n+1) = (1-2\mu)^2 r_2(n) + 4\mu^2\,\lambda_2^{-1}\sum_{i=1}^{N}\lambda_i r_i(n) + 4\mu^2 k_0(n)\lambda_2^{-1} \qquad (5\text{-}10)$$

where $r_1(n)$ and $r_2(n)$ are associated with the eigenvectors $\underline{\nu}_i/\sqrt{N}$, $i = 1,2,\ldots,L$ and $\underline{q}_i$, $i = L+1,\ldots,N$, respectively. Replacing λ_i with λ_1 ; $i = 1,2,\ldots,L$ and λ_2 ; $i = L+1,\ldots,N$ in the sums of the right hand side of (5-10) yields,

$$\sum_{i=1}^{N} \lambda_i \, r_i(n) = \lambda_1 \, L \, r_1(n) + \lambda_2 (N-L) \, r_2(n) \tag{5-11}$$

Substituting (5-11) in (5-10), results in

$$\underline{r}(n+1) = \begin{bmatrix} r_1(n+1) \\ \\ r_2(n+1) \end{bmatrix} = (\underline{I} - \underline{H})\underline{r}(n) + 4\mu^2 k_0(n)\underline{\lambda}^{-1} \tag{5-12}$$

where

$$\underline{H} = \begin{bmatrix} 4\mu\{1-\mu(L+1)\} & -4\mu^2(N-L)\lambda_2/\lambda_1 \\ \\ -4\mu^2 L\lambda_1/\lambda_2 & 4\mu\{1-\mu(N-L+1)\} \end{bmatrix} \tag{5-13}$$

and

$$\underline{\lambda} = [\, \lambda_1 \quad \lambda_2 \,]^T \tag{5-14a}$$

$$\underline{\lambda}^{-1} = [\, \lambda_1^{-1} \quad \lambda_2^{-1} \,]^T \tag{5-14b}$$

Without loss of generality, we assume that the initial weight vector is a null vector, $\underline{0}$. Therefore, we can write the error weight vector as

$$\underline{M}_w(n) - \underline{w}_{opt} = -\frac{\sigma_s^2}{\lambda_1} (1-2\mu)^n \sum_{i=1}^{L} \beta_i \, \underline{\nu}_i \tag{5-15}$$

Now, putting (5-15) and (5-4) in the scalar quantity, $k_0(n)$, i.e.,

$$k_0(n) = \xi_{min} + \{E[\underline{w}(n)]-\underline{w}_{opt}\}^{\dagger}\underline{R}\{E[\underline{w}(n)]-\underline{w}_{opt}\}$$

$$= \xi_{min} + \frac{\sigma_s^2}{\lambda^2}(1-2\mu)^{2n}\sum_{i=1}^{L}\beta_i^*\underline{\nu}_i^{\dagger} \cdot \{\lambda_1\sum_{i=1}^{L}\beta_i\underline{\nu}_i\}$$

$$= \xi_{min} + \frac{(\sigma_s^2)^2}{\lambda_1} N L (1-2\mu)^{2n} \qquad (5\text{-}16)$$

where the orthogonality of the vectors $\{\underline{\nu}_i\}$ has been used to complete the sums on the right hand side of the first quantity.

Note that, the matrix $\underline{H}$ in (5-13) is not symmetric. It is well-known that a simple matrix, i.e., an NxN matrix with N distict eigenvalues, is similar to a diagonal matrix [53]. Since it has been found that in general, $\underline{H}$ has distinct eigenvalues, hence, there exists a non-unitary diagonalizing transformation of $\underline{H}$. In order to easily carry out the study, let us denote $\underline{H}$ as

$$\underline{H} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad (5\text{-}17)$$

where the values of a, b, c and d are corresponding to the elements in (5-13) and b≠c. We can then find a daigonal matrix $\underline{T}$:

$$\underline{T} = \text{diag}(1, \sqrt{b/c}) \qquad (5\text{-}18)$$

the similarity transformation resulting in a symmetric matrix,

if the inverse of $\underline{T}$ exists. That is,

$$\underline{T}\,\underline{H}\,\underline{T}^{-1} = \begin{bmatrix} a & \sqrt{bc} \\ \sqrt{bc} & d \end{bmatrix} = \underline{A} \qquad (5\text{-}19a)$$

or

$$\underline{A} = \begin{bmatrix} 4\mu\{1-(L+1)\mu\} & -4\mu^2\sqrt{(N-L)L} \\ -4\mu^2\sqrt{(N-L)L} & 4\mu\{1-\mu(N-L+1)\} \end{bmatrix} \qquad (5\text{-}19b)$$

For further analysis, we define a new vector, $\underline{c}(n+1)$, as

$$\underline{c}(n+1) = \underline{T}\,\underline{r}(n+1)$$
$$= (\underline{I} - \underline{A})\,\underline{c}(n) + 4\mu^2 k_0(n)\underline{B} \qquad (5\text{-}20)$$

which is designated by

$$\underline{c}(n) = [\, c_1(n) \quad c_2(n) \,]^T \qquad (5\text{-}21a)$$

or

$$\underline{c}(n) = \left[\, r_1(n) \quad (\lambda_2/\lambda_1)\sqrt{\frac{N-L}{L}}\, r_2(n)\right]^T \qquad (5\text{-}21b)$$

The vector $\underline{B}$, in (5-20), is defined as

$$\underline{B} = \underline{T}\,\underline{\lambda}^{-1} = \begin{bmatrix} \lambda_1^{-1} \\ \sqrt{(N-L)/L}\,\lambda_1^{-1} \end{bmatrix} \qquad (5\text{-}22)$$

Now, because $\underline{A}$ is a symmetric matrix, there exists an unitary transformation which diagonalizes $\underline{A}$. If we let α_i ; i = 1,2, denote the eigenvalues of $\underline{A}$ and $\underline{s}_i$; i = 1, 2 denote the associated eigenvectors, i.e.,

$$\underline{s}_i = [\, s_1^{\,i} \quad s_2^{\,i} \,]^T \qquad (5\text{-}23)$$

Therefore, we have the unitary matrix, $\underline{S}$, i.e.,

$$\underline{S} = (\underline{s}_1 \quad \underline{s}_2) \tag{5-24}$$

which transforms matrix $\underline{A}$ into a diagonal form:

$$\underline{\alpha} = \text{diag} (\alpha_1 \quad \alpha_2) \tag{5-25a}$$

that is,

$$\underline{S}^T \underline{A} \underline{S} = \underline{\alpha} \tag{5-25b}$$

Now, starting with the initial vector, $\underline{c}(0)$, and iteratively substituting in (5-20), we have

$$\underline{c}(n) = (\underline{I} - \underline{A})^n \underline{c}(0) + 4\mu^2 \xi_{min} \sum_{i=0}^{n-1} (\underline{I}-\underline{A})^i \underline{B}$$

$$+ \frac{4\mu^2 (\sigma_s^2)^2 NL(1-2\mu)^{2(n-1)}}{\lambda_1} \sum_{i=0}^{n-1} (1-2\mu)^{-2i} (\underline{I}-\underline{A})^i \underline{B} \tag{5-26}$$

Since $\underline{S}$ is a similarity matrix of $\underline{A}$, we have

$$(\underline{I}-\underline{A})^n = \underline{S} \begin{bmatrix} \{1-\alpha_1\}^n & 0 \\ 0 & \{1-\alpha_2\}^n \end{bmatrix} \underline{S}^T \tag{5-27}$$

Now, substituting (5-27) into (5-26), we can express (5-26) in closed form:

$$\underline{c}(n) = \underline{S} \begin{bmatrix} (1-\alpha_1)^n & 0 \\ 0 & (1-\alpha_2)^n \end{bmatrix} \underline{S}^T \underline{c}(0)$$

$$+4\mu^2 \xi_{min} \underline{S} \begin{bmatrix} \frac{1-(1-\alpha_1)^n}{\alpha_1} & 0 \\ 0 & \frac{1-(1-\alpha_2)^n}{\alpha_2} \end{bmatrix} \underline{S}^T \underline{B}$$

$$+ \frac{4\mu^2 NL(\sigma_s^2)^2 \underline{S}}{\lambda_1} \begin{bmatrix} \dfrac{(1-2\mu)^{2n}-(1-\alpha_1)^n}{(1-2\mu)^2-(1-\alpha_1)} & 0 \\ 0 & \dfrac{(1-2\mu)^{2n}-(1-\alpha_2)^n}{(1-2\mu)^2-(1-\alpha_2)} \end{bmatrix} \underline{S}^T \underline{B}$$

$$n = 0, 1, \ldots \qquad (5\text{-}28)$$

Again, since $\underline{r}(n)$ is the equivalent vector form of the weight covariance matrix, from (5-21b) and (4-64) the weight covariance matrix, $\mathrm{Cov}[\underline{w}(n)\,\underline{w}^{\ast}(n)]$ can be expressed in terms of $\underline{c}_i(n)$; i=1,2 :

$$\mathrm{Cov}[\,\underline{w}(n)\underline{w}^{\ast}(n)\,] = \frac{\lambda_1}{\lambda_2}\sqrt{\frac{L}{N-L}}\, c_2(n)\, \underline{I} + \left\{ c_1(n) - \frac{\lambda_1}{\lambda_2}\sqrt{\frac{L}{N-L}}\, c_2(n) \right\} \sum_{i=1}^{L} \underline{\nu}_i \underline{\nu}^{\ast}{}_i / N$$

$$n= 0,1, \ldots \qquad (5\text{-}29)$$

In general, the mean weights converge if the magnitude of the gain scale eigenvalues $\mu\lambda_i$; i =1,2,...,N, of the input autocorrelation matrix $\underline{R}$ are less than unity, that is,

$$|\alpha_i| < 1 \ ; \ i=1,2 \qquad (5\text{-}30)$$

Note that, based on (5-28) and (5-29), the transient weight covariance matrix can be explicitly determined. Now, if μ, the step-size, is selected so that the condition of (5-30) is satisfied, then the steady-state $\underline{c}(n)$ will be (5-28)

$$\lim_{n\to\infty} \underline{c}(n) = 4\mu^2 \xi_{min}\, \underline{S} \begin{bmatrix} \alpha_1^{-1} & 0 \\ 0 & \alpha_2^{-1} \end{bmatrix} \underline{S}^T \underline{B}$$

$$= 4\mu^2 \xi_{min}\, \underline{A}^{-1}\, \underline{B} \qquad (5\text{-}31)$$

or

$$\lim_{n\to\infty} \underline{c}(n) = \frac{\mu\ \xi_{min}}{2\ [1-\mu(N+2)+\mu^2(N+1)]} \begin{bmatrix} (2-2\mu)\lambda_1^{-1} \\ (2-2\mu)\sqrt{(N-L)/L}\ \lambda_1^{-1} \end{bmatrix} \tag{5-32}$$

where

$$\underline{A}^{-1} = \frac{1}{16\mu^2[1-\mu(N+2)+\mu^2(N+1)]} \begin{bmatrix} 4\mu\{1-\mu(N-L+1)\} & 4\mu^2\sqrt{(N-L)L} \\ 4\mu^2\sqrt{(N-L)L} & 4\mu\{1-\mu(L-1) \end{bmatrix} \tag{5-33a}$$

and

$$\underline{A}^{-1}\underline{B} = \frac{1}{8\ \mu[1-\mu(N+2)+\mu^2(N+1)]} \begin{bmatrix} (2-2\mu)\lambda_1^{-1} \\ (2-2\mu)\sqrt{(N-L)/L}\ \lambda_1^{-1} \end{bmatrix} \tag{5-33b}$$

Applying (5-32) to (5-29), the steady-state weight covariance matrix, results in

$$\mathrm{Cov}[\ \underline{w}(n)\underline{w}^{*}(n)\ ]$$

$$= \frac{\mu\ \xi_{min}}{\lambda_1\ \lambda_2[1-\mu(N+1)]}\ \{\ \lambda_1\underline{I} + (\lambda_2-\lambda_1)\sum_{i=1}^{L} \underline{v}_i\underline{v}^{*}_i / N\}$$

$$n = 0, 1, \ldots \tag{5-34}$$

5.3.2 Multiple Sinusoids with Unequal Power (L=2)

The reason for developing the unequal power case is to see how does the power ratio between the sinusoidal signals affect the convergence rate. Here, the case that the received signal with two unequal power sinusoidal signals (L=2) is considered. The basic difference for the analysis of the weight covariance matrix compared to the equal power case is that in the unequal power case the reduced rank (see (5-12)) will be 3x3 rather than 2x2, as in the equal power case.

Assume that the power relationship for both sinusoidal components, i=1 and i=2, is

$$\sigma^2_{s_1} = SNR_2 \, \sigma^2_{s_2} \qquad (5\text{-}35)$$

where SNR_2 is a positive, real number. Now, with L=2 and the power ratio relationship from (5-35), the corresponding equation of (5-11), in this case, will be

$$\sum_{i=1}^{N} \lambda_i \, r_i(n) = \lambda_1 r_1(n) + \lambda_2 r_2(n) + \lambda_3 r_3(n)(N-2) \qquad (5\text{-}36a)$$

Here, the eigenvalues are defined as

$$\lambda_1 = \sigma^2_n + N \sigma^2_{s_1}, \qquad (5\text{-}36b)$$

$$\lambda_2 = \sigma^2_n + N \sigma^2_{s_2}, \qquad (5\text{-}36c)$$

and

$$\lambda_3 = \sigma^2_n \qquad (5\text{-}36d)$$

Now, substituting (5-36) into (5-9), results in

$$\underline{r}(n+1) = \begin{bmatrix} r_1(n+1) \\ r_2(n+1) \\ r_3(n+1) \end{bmatrix}$$

$$= (\underline{I} - \underline{H})\underline{r}(n) + 4\mu^2 k_0(n)\underline{\lambda}^{-1} \qquad (5\text{-}37)$$

where

$$\underline{H} = \begin{bmatrix} 4\mu(1-2\mu) & -4\mu^2\frac{\lambda_2}{\lambda_1} & -4\mu^2(N-2)\frac{\lambda_3}{\lambda_1} \\ -4\mu^2\frac{\lambda_1}{\lambda_2} & 4\mu(1-2u) & -4\mu^2(N-2)\frac{\lambda_3}{\lambda_2} \\ -4\mu^2\frac{\lambda_1}{\lambda_3} & -4\mu^2\frac{\lambda_2}{\lambda_3} & 4\mu\{1-\mu(N-1)\} \end{bmatrix} \qquad (5\text{-}38)$$

and

$$\underline{\lambda} = [\lambda_1 \quad \lambda_2 \quad \lambda_3]^T \qquad (5\text{-}39a)$$

$$\underline{\lambda}^{-1} = [\lambda_1^{-1} \quad \lambda_2^{-1} \quad \lambda_3^{-1}]^T \qquad (5\text{-}39b)$$

Again, without loss of generality, we assume that the initial weight vector is a null vector, $\underline{0}$, thus, we have the error weight vector :

$$\underline{M}_{w(n)} - \underline{w}_{opt} = \sum_{i=1}^{2} \frac{\sigma_{s_i}^2}{\sigma_n^2 + N\,\sigma_{s_i}^2}(1-2\mu)^n \beta_i \underline{\nu}_i \qquad (5\text{-}40)$$

Now, putting (5-40) and (5-4) into the scalar quantity, $k_o(n)$, we obtain

$$k_0(n) = \xi_{min} + \{E[\underline{w}(n)] - \underline{w}_{opt}\}^{*}\underline{R}\{E[\underline{w}(n)] - \underline{w}_{opt}\}$$

$$= \xi_{min} + \frac{N(\sigma_{s1}^2)^2}{\lambda_1}(1-2\mu)^{2n} + \frac{N(\sigma_{s2}^2)^2}{\lambda_2}(1-2\mu)^{2n} \qquad (5\text{-}41)$$

Again, the orthogonality of the vectors $\{ \underline{v}_i \}$ has been used to complete the sums on the right hand side of the first quantity.

As in the equal power case, the matrix $\underline{H}$ in (5-38) is not symmetric. Similarly, we can find a similarity transformation, $\underline{T}$:

$$\underline{T} = \text{diag}(1, \lambda_2/\lambda_1, (\lambda_3/\lambda_1)\sqrt{N-2}) \qquad (5\text{-}42)$$

such that $\underline{H}$ can be transformed to be as

$$\underline{T}\,\underline{H}\,\underline{T}^{-1} = \underline{A}$$

$$= \begin{bmatrix} 4\mu(1-2\mu) & -4\mu^2 & -4\mu^2\sqrt{N-2} \\ -4\mu^2 & 4\mu(1-2\mu) & -4\mu^2\sqrt{N-2} \\ -4\mu^2\sqrt{N-2} & -4\mu^2\sqrt{N-2} & 4\mu\{1-\mu(N-1)\} \end{bmatrix} \qquad (5\text{-}43)$$

We recognize that $\underline{A}$ is a symmetric matrix. Again, we define a new vector, $\underline{c}(n+1)$:

$$\underline{c}(n+1) = \underline{T}\,\underline{r}(n+1)$$
$$= (\underline{I} - \underline{A})\,\underline{c}(n) + 4\mu^2 k_0(n)\underline{B} \qquad (5\text{-}44)$$

which is designated by

$$\underline{c}(n) = [\, c_1(n) \quad c_2(n) \quad c_3(n)\,]^T$$
$$= [\, r_1(n) \quad \frac{\lambda_2}{\lambda_1} r_2(n) \quad \frac{\lambda_3}{\lambda_1} r_3(n)\,]^T \qquad (5\text{-}45)$$

The vector $\underline{B}$, (5-44), is defined as

$$\underline{B} = \underline{T}\,\underline{\lambda}^{-1}$$

$$= \begin{bmatrix} \lambda_1^{-1} & & \\ & \lambda_1^{-1} & \\ & & \sqrt{N-2}\ \lambda_1^{-1} \end{bmatrix} \qquad (5\text{-}46)$$

Now, since $\underline{A}$ is a symmetric matrix, there exists the unitary transformation which diagonalizes $\underline{A}$. Again, we let α_i ; i= 1,2, and 3, denote the eigenvalues of $\underline{A}$ and $\underline{s}_i$; i = 1,2, and 3, denote the associated eigenvectors :

$$\underline{s}_i = [\, s_1^{\,i} \quad s_2^{\,i} \quad s_3^{\,i}]^T \qquad (5\text{-}47)$$

then we have the unitary matrix $\underline{S}$:

$$\underline{S} = (\, \underline{s}_1 \quad \underline{s}_2 \quad \underline{s}_3) \qquad (5\text{-}48)$$

which transforms matrix $\underline{A}$ into a diagonal form, i.e.,

$$\underline{\alpha} = \mathrm{diag}(\alpha_1 \quad \alpha_2 \quad \alpha_3) \qquad (5\text{-}49a)$$

that is,

$$\underline{\alpha} = \underline{S}^T\, \underline{A}\, \underline{S} \qquad (5\text{-}49b)$$

Again, starting with $\underline{c}(0)$, and iteratively substituting in (5-44), we have

$$\underline{c}(n) = (\, \underline{I} - \underline{A})^n\, \underline{c}(0) + 4\mu^2 \xi_{min} \sum_{i=0}^{n-1} (\underline{I}-\underline{A})^i \underline{B}$$

$$+ 4\mu^2 N \left\{ \frac{(\sigma_{s1}^2)^2}{\lambda_1} + \frac{(\sigma_{s2}^2)^2}{\lambda_2} \right\} (1-2\mu)^{2(n-1)} \sum_{i=0}^{n-1} (1-2\mu)^{-2i} (\underline{I}-\underline{A})^i\, \underline{B} \qquad (5\text{-}50)$$

Since $\underline{S}$ is a unitary matrix, we have

$$(\underline{I}-\underline{A})^n = \underline{S}\begin{bmatrix}(1-\alpha_1)^n & 0 & 0\\ 0 & (1-\alpha_2)^n & 0\\ 0 & 0 & (1-\alpha_3)^n\end{bmatrix}\underline{S}^T \qquad (5\text{-}51)$$

We can also express (5-50) in the close form :

$$\underline{c}(n) = \underline{S}\begin{bmatrix}(1-\alpha_1)^n & 0 & 0\\ 0 & (1-\alpha_2)^n & 0\\ 0 & 0 & (1-\alpha_3)^n\end{bmatrix}\underline{S}^T\underline{c}(0)$$

$$+4\mu^2\xi_{min}\underline{S}\begin{bmatrix}\frac{1-(1-\alpha_1)^n}{\alpha_1} & 0 & 0\\ 0 & \frac{1-(1-\alpha_2)^n}{\alpha_2} & 0\\ 0 & 0 & \frac{1-(1-\alpha_3)^n}{\alpha_3}\end{bmatrix}\underline{S}^T\underline{B}$$

$$+4\mu^2 N\left(\frac{\sigma_{s_1}^4}{\lambda_1}+\frac{\sigma_{s_2}^4}{\lambda_2}\right)\underline{S}\begin{bmatrix}\frac{(1-2\mu)^{2n}-(1-\alpha_1)^n}{(1-2\mu)^2-(1-\alpha_1)} & 0 & 0\\ 0 & \frac{(1-2\mu)^{2n}-(1-\alpha_2)^n}{(1-2\mu)^2-(1-\alpha_2)} & 0\\ 0 & 0 & \frac{(1-2\mu)^{2n}-(1-\alpha_3)^n}{(1-2\mu)^2-(1-\alpha_3)}\end{bmatrix}\underline{S}^T\underline{B}$$

$$n = 0,1, \ldots \qquad (5\text{-}52)$$

Similarly, since the weight covariance matrix can be

equivalently expressed by $\underline{r}(n)$ in the vector form, therefore, from (5-21b) and (5-45), the weight covariance matrix, Cov $[\underline{w}(n)\ \underline{w}^{*}(n)]$ can be expressed in terms of $c_i(n)$, i = 1, 2, and 3 :

$$\text{Cov}[\ \underline{w}(n)\underline{w}^{*}(n)\ ]$$

$$= \frac{\lambda_1}{\lambda_3}\ \frac{1}{\sqrt{N-2}}\ c_3(n)\ \underline{I}$$

$$+\left(\frac{\lambda_1}{\lambda_2}\ c_2(n) - \frac{\lambda_1}{\lambda_3}\ \frac{1}{\sqrt{N-2}}\ c_3(n)\right)\underline{v}_2\underline{v}^{*}_2 / N$$

$$+\left(c_1(n) - \frac{\lambda_1}{\lambda_3}\ \frac{1}{\sqrt{N-2}}\ c_3(n)\right)\underline{v}_1\underline{v}^{*}_1 / N \qquad (5\text{-}53)$$

Note that (5-53) is the transient weight covariance matrix, with un-equal power and L=2, which can be used to explicitly express the transient MSE. Similarly, if μ, the step-size, is selected so that the condition :

$$|\alpha_i| < 1\ ;\ i=1,2,\text{ and } 3. \qquad (5\text{-}54)$$

is satisfied, then the steady-state $\underline{c}(n)$ can be given as

$$\lim_{n\to\infty} \underline{c}(n) = 4\mu^2\xi_{min}\ \underline{S} \begin{bmatrix} \alpha_1^{-1} & 0 & 0 \\ 0 & \alpha_2^{-1} & 0 \\ 0 & 0 & \alpha_3^{-1} \end{bmatrix} \underline{S}^T\underline{B}$$

$$= 4\mu^2\ \xi_{min}\ \underline{A}^{-1}\ \underline{B}$$

or

$$\lim_{n\to\infty} \underline{c}(n) = \frac{\mu\ \xi_{min}}{[\ 1 - \mu\ (N+1)\ ]} \begin{bmatrix} \lambda_1^{-1} \\ & \lambda_1^{-1} \\ & & \sqrt{N-2}\ \lambda_1^{-1} \end{bmatrix} \qquad (5\text{-}55)$$

because

$$\underline{A}^{-1} = \frac{1}{8\mu[1-\mu][1-\mu(N+1)]} \begin{bmatrix} 2(1-\mu N) & 2\mu & 2\mu\sqrt{N-2} \\ 2\mu & 2(1-\mu N) & 2\mu\sqrt{N-2} \\ 2\mu\sqrt{N-2} & 2\mu\sqrt{N-2} & 2(1-3\mu) \end{bmatrix} \qquad (5\text{-}56a)$$

and

$$\underline{A}^{-1}\underline{B} = \frac{1}{4\mu\ [\ 1 - \mu(N+1)]} \begin{bmatrix} \lambda_1^{-1} \\ & \lambda_1^{-1} \\ & & \sqrt{N-2}\ \lambda_1^{-1} \end{bmatrix} \qquad (5\text{-}56b)$$

Consequently, from (5-53) the steady-state weight covariance matrix is given as

$$\lim_{n\to\infty} \mathrm{Cov}[\ \underline{w}(n)\underline{w}^{*}(n)\ ]$$

$$= \frac{\mu\ \xi_{min}}{\lambda_1\ \lambda_2\ \lambda_3[1-\mu(N+1)]} \{\ \lambda_2\ \lambda_3 \underline{I}$$

$$+ \lambda_1(\lambda_3-\lambda_2)\, \underline{v}_2\underline{v}^{\pm}_2/N$$

$$+ \lambda_2(\lambda_3-\lambda_1)\, \underline{v}_1\underline{v}^{\pm}_1/N\}$$

(5-57)

5.4 Mean Square Error (MSE) of the ALE

The transient MSE of ALE can be expressed in terms of minimum MSE, mean weight vector, weight covariance matrix, Wiener weight and the input autocorrelation matrix (4-65). The results derived in previous sections can now be used to study the transient behaviour of ALE in terms of MSE. The transient MSE can be written as (4-66)

$$\xi n = k_0(n) + Tr\{\underline{R}\ Cov\{\underline{w}(n)\ \underline{w}^{\pm}\}\}$$

$$= k_0(n) + \sum_{i=1}^{N} r_i(n)\, \lambda_i \qquad (5\text{-}58)$$

Based on (5-58) two cases viz, the mixed signal with the equal and unequal power multiple sinusoids are considered.

5 4.1 Multiple Sinusoids with Equal Power

In case of equal power sinusoids, the transient MSE of ALE can be evaluated by substituting (5-11) and (5-16) in (5-58), this results in

$$\xi n = \xi min + \sigma^2_s\, NL\, (1-2\mu)^{2n} + \sum_{i=1}^{N} r_i(n)\, \lambda_i$$

(5-59)

where the minimum MSE, ξ_{min}, is given as (4-55)

$$\xi_{min} = E[d(n)\, d^*(n)] - \underline{p}^H \underline{w}_{opt}$$
$$= (\sigma_n^2 + L\,\sigma_s^2) - N L (\sigma_s^2)^2/\lambda_1 \qquad (5\text{-}60)$$

In (5-59) the components of $\underline{r}(n)$, i.e., $r_i(n)$; i=1,2, can be obtained as follows:

$$\underline{r}(n) = \begin{bmatrix} r_1(n) \\ \\ r_2(n) \end{bmatrix}$$
$$= \underline{T}^{-1}\, \underline{c}(n) \qquad (5\text{-}61)$$

where $\underline{c}(n)$ is evaluated via (5-28). Note that, in this case, the eigenvalues of the autocorrelation matrix have two values (see (5-6a) and (5-6c)), that is

$$\lambda_1 = \sigma_n^2 + N\,\sigma_s^2 \quad ; \; i=1,2,\ldots,L$$
$$\lambda_2 = \sigma_n^2 \quad ; \; i=L+1, \ldots, N. \qquad (5\text{-}62)$$

Based on (5-60), (5-61) and (5-62), the transient MSE can be explicitly determined. In steady-state, the second term on the right hand side of (5-59) vanishes, therefore, we have

$$\lim_{n\to\infty} \xi_n = \xi_{min} + \lim_{n\to\infty} \sum_{i=1}^{N} r_i(n)\,\lambda_i \qquad (5\text{-}63)$$

Since in steady-state we have

$$\lim_{n\to\infty} \underline{r}(n) = \underline{T}^{-1} \lim_{n\to\infty} \underline{c}(n)$$
$$= \frac{\mu(1-\mu)\,\xi_{min}}{[1-\mu(N+2)+\mu^2(N+1)]} \begin{bmatrix} \lambda_1^{-1} \\ \\ \lambda_2^{-1} \end{bmatrix} \qquad (5\text{-}64)$$

the steady-state MSE can be expressed as

$$\lim_{n\to\infty} \xi_n = \frac{(1-\mu)\,\xi_{min}}{1-\mu(N+1)}$$

$$= \frac{\xi_{min}}{1-\sum_{i=1}^{N} \frac{\mu}{1-\mu}} \qquad (5\text{-}65)$$

Note that, the result showed in (5-65) agree with the result obtained in chapter 4 (see (4-71)).

5.4.2 Multiple Sinusoids with Unequal Power (L=2)

Similar to the case discussed in sec. 5.4.1, the transient MSE of ALE with unequal power sinusoids can also be evaluated by substituting (5-36) and (5-41) in (5-58), this results in

$$\xi_n = k_0(n) + \sum_{i=1}^{N} r_i(n)\ \lambda_i$$

$$= \xi_{min} + N\left(\sigma^2_{s_1}\right)^2 (1-2\mu)^{2n} + N\left(\sigma^2_{s_2}\right)^2 (1-2\mu)^{2n} + \sum_{i=1}^{N} r_i(n)\,\lambda_i \qquad (5\text{-}66)$$

where the values $r_i(n)$; i=1,2,3 can be similarly obtained that is

$$\underline{r}(n) = \begin{bmatrix} r_1(n) \\ r_2(n) \\ r_3(n) \end{bmatrix}$$

$$= T^{-1}\,\underline{c}(n)$$

$$= \left[\ c_1(n)\ \ \frac{\lambda_1}{\lambda_2}c_2(n)\ \ \frac{\lambda_1}{\lambda_3\sqrt{N-2}}c_3(n)\ \right] \qquad (5\text{-}67)$$

Note that, $\underline{c}(n)$ can be evaluated via (5-52) and $\underline{T}$ was defined in (5-42). Furthermore, the minimum MSE, ξ_{min}, is given by

$$\xi_{min} = \sum_{i=1}^{2} \sigma_{s_i}^2 + \sigma_n^2 - \sum_{i=1}^{2} N(\sigma_{s_i}^2)^2 \tag{5-68}$$

Again, in steady state, (5-67) can be expressed as

$$\lim_{n\to\infty} \underline{r}(n) = T^{-1} \lim_{n\to\infty} \underline{c}(n) = \frac{\mu \xi_{min}}{1 - \mu(N+1)} \underline{\lambda}^{-1} \tag{5-69}$$

consequently, the steady state MSE can be written as (5-66)

$$\lim_{n\to\infty} \xi_n = \frac{\xi_{min}}{1 - \sum_{i=1}^{N} \frac{\mu}{1-\mu}} \tag{5-70}$$

From (5-65) and (5-70), we recognize that both equal and unequal power cases have the same expression of the steady-state MSE. Since the minimum MSE values for both cases described in (5-60) and (5-68) are not equal, the steady-state MSE is different in both cases. For the purpose of comparison, it is interesting to use the relative MSE which is known as the misadjustment [8] instead of the MSE. Here, the misadjustment is defined as

$$\text{Misadjustment} = \frac{\xi_n - \xi_{min}}{\xi_{min}} \tag{5-71}$$

Under this definition, we can see that both equal and unequal power cases have the same steady state misadjustment:

$$\text{Misadjustment} = \frac{\sum_{i=1}^{N} \frac{\mu}{1-\mu}}{1 - \sum_{i=1}^{N} \frac{\mu}{1-\mu}} \qquad (5\text{-}72)$$

(5-72) means that the steady-state misadjustment only depends on the step-size and the number of taps used in ALE. Again, this is not the case of the conventional LMS ALE which was discussed by Fisher [42].

5.5 Computer Simulation Results

In order to explore the transient behaviour of the normalized K-L TDLMS algorithm in the application of ALE, we evaluate the transient MSE values using the equations derived in previous sections. Furthermore, for the purpose of comparison, the transient MSE of the conventional LMS ALE with unequal power sinusoidal singnals is evaluated using equations derived in Appendix F.

It is widely known that, in general, the time domain LMS adaptive algorithm converges slowly, if the eigenvalue spread of the autocorrelatin matrix is large [8,23]. It is in this situation that other algorithms with fast convergence are useful. The purpose of this section is to study how does the power ratio between the sinusoidal signals in ALE affect the convergence rate in the situation described above. Although, in practice application to the ALE, the received signal with low signal-to-noise ratio (SNR) is generally

considered [42], however, in this section we will assume the received signal with high SNR. This is because at high SNR, the eigenvalue spread of the input autocorrelation matrix is large, which enables us to study the effect of power ratio on convergence rate.

In brief, the purpose of this section is two fold: (1) to see how does the normalized K-L TDLMS algorithm perform in the high SNR case compared to the time domain LMS algorithm, and (2) to see how the power ratio between the sinusoidal signals affects the convergence rate for both algorithms. To do so, first we consider a signal model with the parameters; N=16, L=2, μ=0.03, SNR=60dB and SNR_2 =30dB. The transient behaviour for both algorithms is evaluated using equations given in section 5.4 and Appendix F. The results are shown in figure 5.2. From figure 5.2, we found that the normalized K-L TDLMS algorithm converges much faster than the time domain LMS algorithm and also has a small steady-state MSE value. This is no surprise because, in this case, the eigenvalue spread is very high and the power ratio between the sinusoidal signals is also large. On the other hand, in the normalized K-L TDLMS algorithm the transformed autocorrelation matrix is an identity matrix and hence, the eigenvalue spread is an unity. Next, to see the effect on the convergence rate due to the power ratio between the sinusoidal signals, the same signal parameters as in figure 5.2 but with different SNR_2 are used. The results are shown in figure 5.3, in which the curves with solid line are the

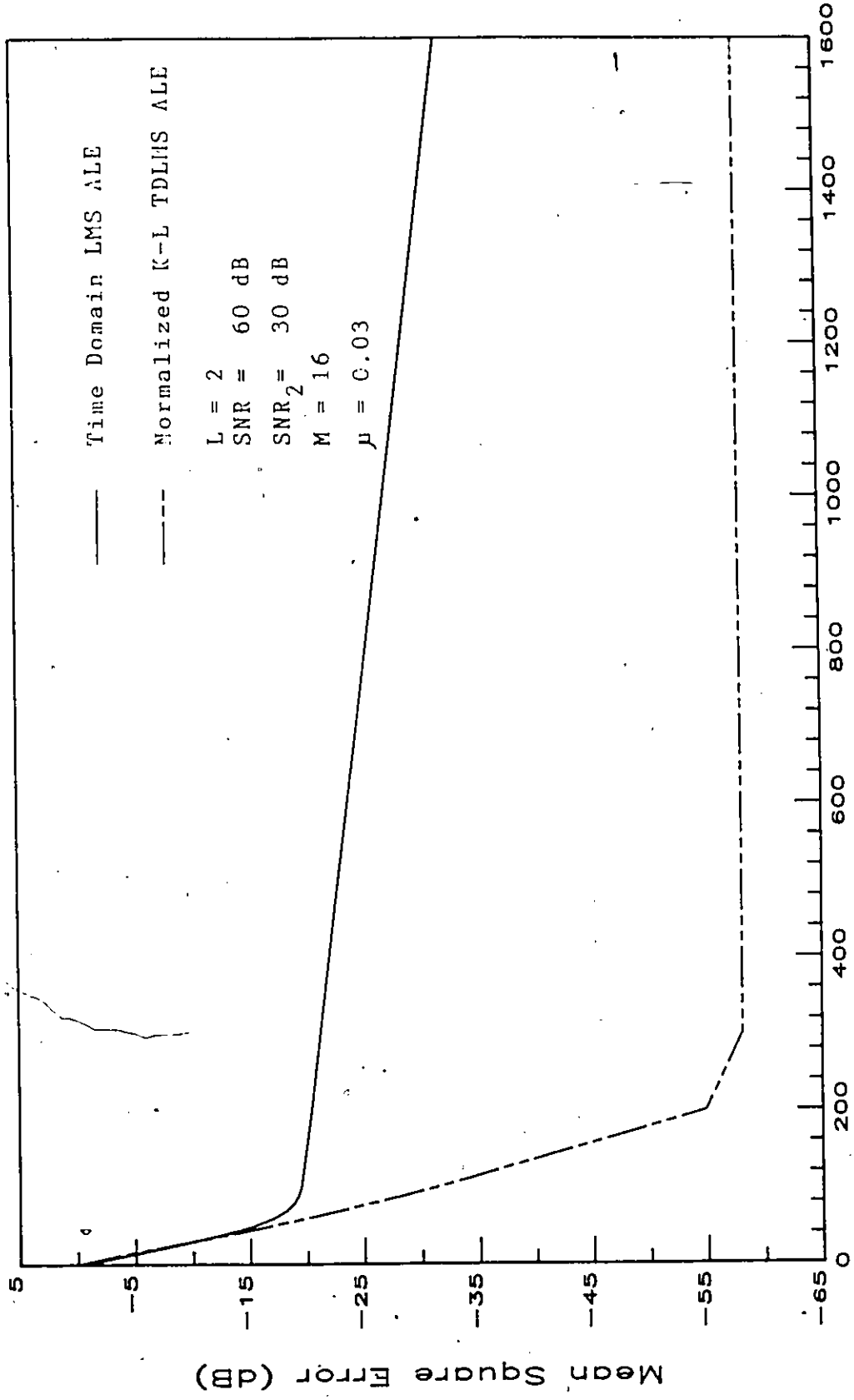


Figure 5.2 Comparison of Performances for the Time Domain LMS and the Normalized K-L TDLMS ALE

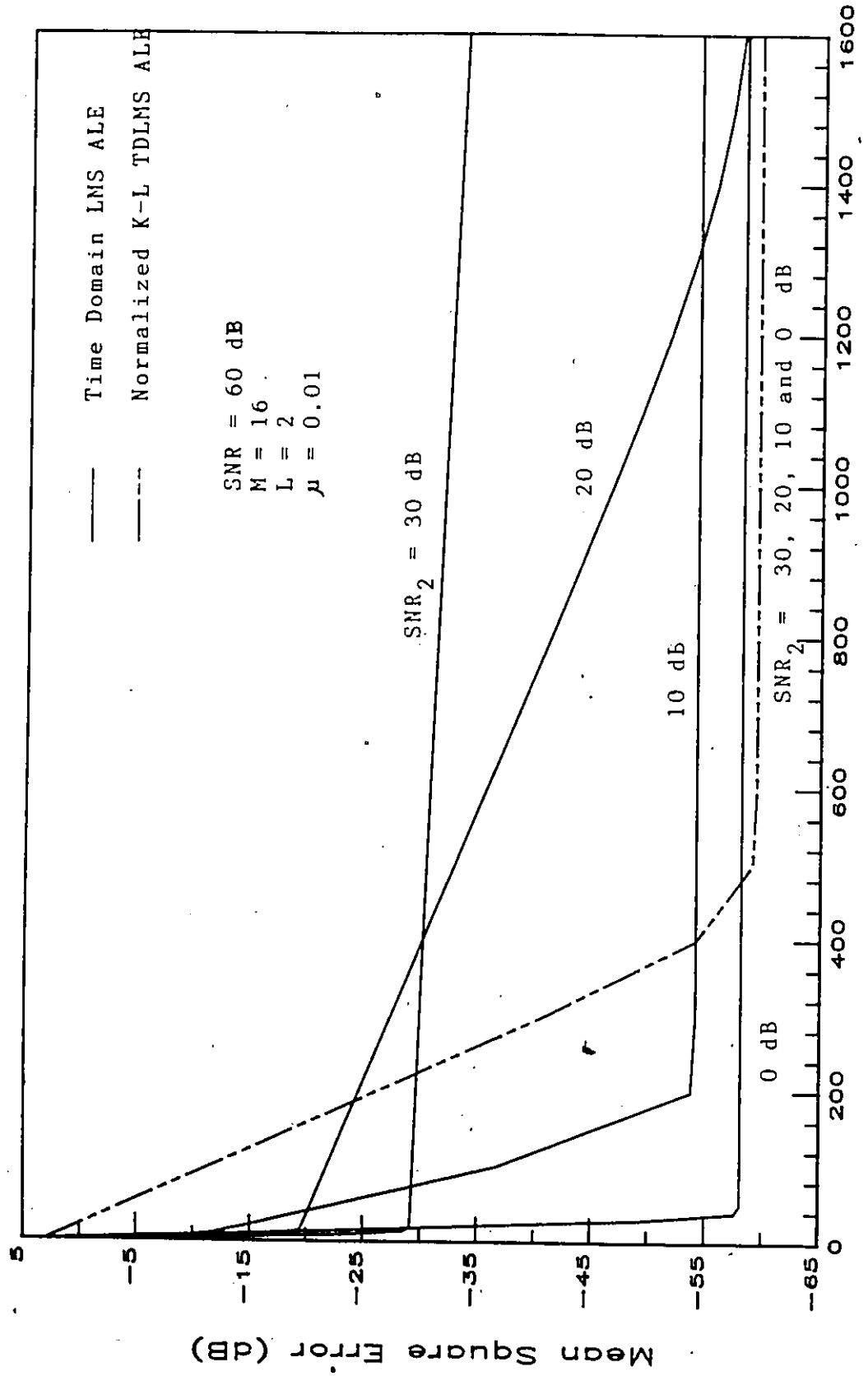


Figure 5.3 Comparison of Performances with Varied SNR_2 for the ALE

transient behaviour of the time domain LMS ALE with the values of SNR_2 being 30 dB, 20 dB, 10 dB and 0 dB respectively. From these curves, we can see that the convergence rate of the time domain LMS ALE depends very much on the power ratio between the sinusoidal signals. That is, the algorithm converges faster as the value of SNR decreases. As we mentioned earlier, in the normalized K-L TDLMS algorithm, the transformed autocorrelation matrix is an identity matrix. Therefore, we can expect that the power ratio does not affect the convergence rate when the same values of SNR_2, which is used in the time domain LMS ALE, are applied to the normalized K-L TDLMS ALE. This is exactly shown in figure 5.3, a curve with solid / two centre dot. Finally, in figure 5.4, we show that the convergence rate of the normalized K-L TDLMS ALE depends only on the values of step-size μ. Here, the parameters are N=16, L=2, SNR=60 dB, SNR_2=30 dB and the various values of μ, 0.01, 0.02 and 0.03.

The results for the normalized K-L TDLMS algorithm are in agreement with the results obtained in chapter 4. That is, the convergence rate depends only on the values of step-size. This can be interpreted as that the eigenvalues of the transformed autocorrelation matrix being all unity. However, it is surprising to note that the convergence rate of the conventional time domain LMS algorithm not only depends on the spread of eigenvalues of the input autocorrelation matrix, but also depends highly on the power

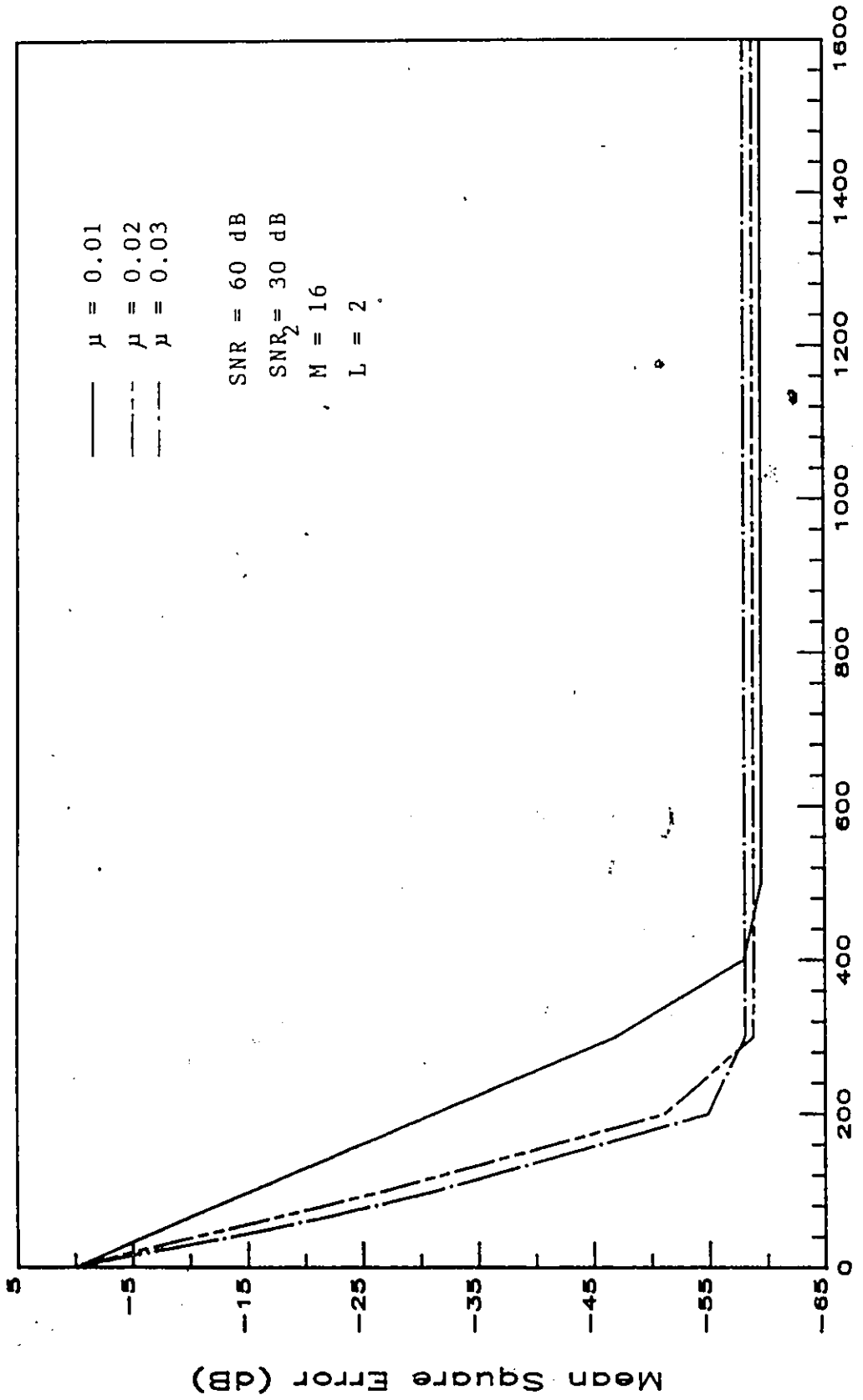


Figure 5.4 Comparison of Performance with Varied stepsize in the Normalized K-L TDLMS ALE

ratio of the sinusoidal signal.

5.6 Conclusions

In this chapter, we have derived the explicit expressions of the transient MSE of the normalized K-L TDLMS ALE with equal and unequal powers sinusoidal signals in the received signal. These analytical expressions of the transient MSE enable us to further investigate the transient behaviour of the normalized K-L TDLMS algorithm in the application of the ALE. Furthermore, for the purpose of comparison, the transient MSE of the time domain LMS ALE with unequal power in the received signal is also derived. This is then used to further examine the effect of the power ratio between the sinusoidal signals on the convergence rate.

In all the computer simulations with high SNR, we showed that the normalized K-L TDLMS algorithm in the application of ALE, the convergence rate depends only on the values of step-size. On the other hand, the convergence rate of the time domain LMS ALE depends not only on the eigenvalue spread, but also depends on the factor of the power ratio between the sinusoidal signals in the received signal. However, we found that the time domain LMS algorithm converges rapidly on the first few iterations and give rise to oscillatory progress and become unsatisfactory. This situation is very similar to the standard steepest descent

method which we described in chapter 2. Because the implementation of the normalized K-L TDLMS algorithm requires a large amount of computations, therefore, we feel that a hybrid algorithm which combines both the time domain LMS algorithm and the normalized K-L TDLMS algorithm is necessary to achieve fast convergence and at the same time reducing the computational efforts. That is, on the first few iterations, we utilize the time domain LMS algorithm and after that the normalized K-L TDLMS algorithm is used.

CHAPTER 6

SUBOPTIMAL ORTHOGONAL TRANSFORMED DOMAIN LMS ADAPTIVE FILTERS

6.1 Introduction

In the preceding chapters, we utilised the K-L transform to study the effect of the transformed domain implementation analytically. However, in real time implementation of the K-L transform is difficult (sec. 3.2), therefore, the suboptimal orthogonal transforms are used. These orthogonal transforms possess fast computation algorithms or good orthogonality for the input signals. Recently, fast orthogonal transforms have played an important role in applied engineering problems [55]. Fast transforms can be used directly to filter signals in the frequency domain and indirectly to design digital filters for time domain processing [55]. They are also used for convolution evaluation and signal decomposition and other applications. In the present chapter, the orthogonal transforms are employed in the transformed domain LMS adaptive algorithms for filtering. In the transformed domain adaptive filtering, the decorrelation property of the orthogonal transforms is important. The decorrelation property of the orthogonal transform for a specified signal can be measured by

evaluating the sum of the absolute values of the off-diagonal elements of the transformed covariance matrix [56].

Basically, the orthogonal transforms can be categorized as 1) unitary transforms and, 2) non-unitary orthogonal transforms. The unitary transforms are information preserving, and no bandwidth reduction [46] would result from the application of the transform to the signal. Its beneficial effect lies in redistributing the variance associated with each signal into almost uncorrelated variables, thus permitting the transform coefficients to be processed (e.g. filtered) individually. The nonunitary transforms also have the same beneficial effect in redistributing the variance as the unitary transforms did, but are not information preserving. That is the transformed signal vector does not have the same Euclidean norm as the original input vector [60].

The most commonly used discrete transforms are the discrete Fourier transform (DFT) , the discrete cosine transform (DCT), discrete sine transform (DST), the discrete Walsh - Hadamard transform (WHT), the transmultiplexing transform (TMT), the lattice filter transform. Since the decorrelation property of the transforms is signal dependent, one should study various transforms for different signals. In particular, it is well-known [46,57-59], that among the discrete transforms the DCT is essential. This is because the DCT is asymptotically equivalent to the K-L transform for a stationary, finite order Markov signals. However, in this chapter, we will consider the most

commonly used transforms like DFT, DCT, TMT and lattice filter transform. The main issue in this chapter is to apply various orthogonal transforms to the TDLMS algorithms and the BTDLMS algorithms. Furthermore, the effect due to the incomplete diagonalization is also investigated via the DCT.

As mentioned previously, the implementation of adaptive filtering using the TDLMS algorithms requires more computations compared to the time domain LMS algorithm. That is because for each incoming signal the transformation has to be taken to obtain the transformed input vector. To circumvent the increase of the computational effort in the TDLMS adaptive algorithm, a recursive equation for obtaining the DCT of signals is also derived in this chapter.

6.2 Discrete Fourier Transformed Domain Adaptive Filter

The DFT is a Fourier representation of a finite-length sequence which is itself a sequence rather than a continuous function, and it corresponds to samples equally spaced. The development of the fast algorithms [55] has resulted in the wide-range applications of DFT. In this case, the orthogonal functions used for the representation are the exponential functions. Consequently, the filter weights can be expressed as

$$w_i(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} b_k(n) \exp\{ -j \frac{2\pi ni}{N} \} \qquad i=0,1, \ldots ,N-1. \quad (6\text{-}1a)$$

with

$$j = \sqrt{-1}$$

The matrix form of (6-1a) can be expressed as

$$\underline{w}(k) = \underline{F}\,\underline{b}(k) \qquad (6\text{-}1b)$$

where the matrix $\underline{F}$ is the DFT matrix and is defined by

$$\underline{F} = \sqrt{\frac{1}{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-j\frac{2\pi}{N}} & \cdots & e^{-j(N-1)\frac{2\pi}{N}} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 1 & e^{-j(N-1)\frac{\pi 2}{N}} & \cdots & e^{-j(N-1)^2\frac{2\pi}{N}} \end{bmatrix} \qquad (6\text{-}2)$$

Premultiplying the inverse DFT matrix, $\underline{F}^{-1}$, in both sides of (6-1b), we have

$$\underline{b}(k) = \underline{F}^{-1}\,\underline{w}(k) \qquad (6\text{-}3)$$

Here the inverse DFT, $\underline{F}^{-1}$, is defined as

$$\underline{F}^{-1} = \sqrt{\frac{1}{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{j\frac{2\pi}{N}} & \cdots & e^{j(N-1)\frac{2\pi}{N}} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 1 & e^{j(N-1)\frac{2\pi}{N}} & \cdots & e^{j(N-1)^2\frac{2\pi}{N}} \end{bmatrix} \qquad (6\text{-}4)$$

and the scalar form of (6-3) is given as

$$b_k(n) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} w_i(k) \exp\{j\,\frac{2\pi\, in}{N}\} \qquad n=0,1,\ldots,N-1 \qquad (6\text{-}5)$$

From (6-2) and (6-4), we recognize that the DFT is an unitary transform, i.e.,

$$\underline{F}^{*} = \underline{F}^{-1}$$

or

$$\underline{F}^{*} = \underline{F}^{-1} \quad (\quad \underline{F} = \underline{F}^{T} \;) \qquad (6\text{-}6)$$

In what follows, the DFT is applied to the transformed domain algorithms.

6.2.1 The TDLMS Adaptive Algorithm with DFT

The use of DFT in the TDLMS adaptive algorithm is similar to the use of K-L transform in TDLMS adaptive algorithm. The weight vector updated equation can be obtained from (3-23) by substituting the matrix $\underline{Q}$ by matrix $\underline{F}$. This is given by

$$\underline{b}(k+1) = \underline{b}(k) + 2\,\mu\, e(k)\, \underline{s}^{*}(k) \qquad (6\text{-}7)$$

where $e(k)$ and $\underline{s}(k)$ are the error signal and the transformed input vector respectively. In this case $\underline{s}(k)$ is defined as

$$\underline{s}(k) = \underline{F}^{T}\, \underline{x}(k) \qquad (6\text{-}8)$$

and $\underline{b}(k)$ was defined in (6-3). We now show that the equivalent time domain expression of the TDLMS algorithm is equivalent to the time domain LMS complex data algorithm. First, premultiplying $\underline{F}$ in both sides of (6-7), this results in

$$\begin{aligned} \underline{w}(k+1) &= \underline{F}\, \underline{b}(k+1) \\ &= \underline{F}\, \underline{b}(k) + 2\,\mu\, e(k)\, \underline{F}\, (\underline{F}^{T}\, \underline{x}(k))^{*} \\ &= \underline{w}(k) + 2\mu\, e(k)\, \underline{x}^{*}(k) \end{aligned}$$

or

$$\underline{w}(k+1) = \underline{w}(k) + 2\mu\, e(k)\, \underline{x}^{*}(k) \qquad (6\text{-}9)$$

Note that, for real valued input data, (6-9) reduces to the real valued time domain LMS algorithm. Furthermore, we recognize that the transformed domain implementation (6-9), in this specified case can not improve the convergence rate, as it is equivalent to the time domain LMS algorithm.

In chapter 3, we showed that the normalized K-L TDLMS algorithm is equivalent to the transformed domain LMS algorithms developed in [23], where the TDLMS adaptive algorithm with the normalized DFT is implemented in following manner:

$$\underline{c}(k+1) = \underline{c}(k) + 2\,\mu\,\tilde{\underline{R}}_{\lambda}^{-1}\, e(k)\,\underline{z}^{*}(k) \qquad (6\text{-}10a)$$

The scalar form of (6-10a) is given by

$$c_{k+1}(i) = c_k(i) + 2\mu_i\, e(k)\, z_k^{*}(i)$$

with

$$\mu_i = \frac{\mu}{E[\,|z_k(i)|^2\,]} \qquad i=0,\ 1,\ \ldots,\ N-1 \qquad (6\text{-}10b)$$

where the estimate power of the individual DFT output signal is used to estimate the corresponding eigenvalues of the input autocorrelation matrix. In (6-10), the quantities $\tilde{\underline{R}}_{\lambda}$, $\underline{z}(k)$ and $\underline{c}(k)$ are defined as

$$\tilde{\underline{R}}_{\lambda}^{-1} = \mathrm{diag}\left(\frac{1}{E[\,|z_k(0)|^2\,]}\ \frac{1}{E[\,|z_k(1)|^2\,]}\ \ldots\ \frac{1}{E[\,|z_k(N-1)|^2\,]}\right),$$

$$\underline{c}(k) = \underline{F}^{-1}\,\underline{w}(k),$$

and

$$\underline{z}(k) = \underline{F}^{T}\,\underline{x}(k), \qquad (6\text{-}11)$$

If the input autocorrelation matrix can be completely

diagonalized by the DFT, then the individual transformed signal power will be equal to the corresponding eigenvalues. Consequently, the TDLMS adaptive algorithm with normalized DFT can be viewed as the DFT domain implementation of the normalized K-L TDLMS algorithm in the time domain.

The advantage of using the fast transform in this adaptive algorithm is the following. In the time domain, the normalized K-L TDLMS algorithm involves the computation of inverse autocorrelation matrix to carry out the adaptive filtering. Although, in practice, the inverse autocorrelation matrix can be recursively estimated, however, it still requires a lot of computations. This can be avoided if the adaptive filtering is carried out in the transformed domain with fast transform. This is because in the transformed domain implementation only the individual transformed signal power has to be obtained, unlike estimating the inverse input autocorrelation matrix in time domain for each iteration. Furthermore, for reducing the computational effort, the recursive equation of DFT signals can be used [23] :

$$z_k(i) = z_{k-1}(i)\exp\{-j\frac{2\pi i}{N}\} + \frac{1}{\sqrt{N}}[x(k) - x(k-N)]$$
$$i=0,1, \ldots ,N-1. \qquad (6\text{-}12)$$

where $z_k(i)$ is the ith component of the transformed input signal at kth sampling instatnt. It can be seen (6-12), that for each iteration with filter order N, N complex multiplication and N+1 complex addition are required to obtain the transformed input signals.

6.2.2 The BTDLMS Adaptive Filter with DFT

In chapter 3, we have shown that, the BTDLMS adaptive filter is the adaptive implementation of the B.R transformation filters discussed in [29]. Here we will show that the DFT can also be similarly applied to the BTDLMS adaptive algorithm. To see this, first, substituting $\underline{Q}^{-1}$ by $\underline{F}^{-1}$ in (3-52), we have

$$\hat{\underline{d}}(m) = \underline{F}^{-1}\ \underline{d}(m) \qquad (6\text{-}13a)$$

and

$$\hat{\underline{s}}(m) = \underline{F}^{-1}\ \hat{\underline{x}}(m) \qquad (6\text{-}13b)$$

From (3-63), the updated equation of the BTDLMS adaptive algorithm with DFT is given as

$$b_{m+1}(i) = b_m(i) + 2\,\mu_B\ v_m(i)\ \hat{s}^*_m(i) \qquad i=0,1,\ldots,N-1 \qquad (6\text{-}14)$$

We recognize that (6-14) is the Dentino-type frequency domain LMS adaptive filter [20]. Similarly, the BTDLMS adaptive algorithm with normalized DFT can be written as

$$b_{m+1}(i) = b_m(i) + 2\,\mu_i\ v_m(i)\ \hat{s}^*_m(i)$$

with

$$\mu_i = \frac{\mu_B}{E[|\hat{s}_m(i)|^2]}, \quad i=0,1,\ldots,N-1 \qquad (6\text{-}15)$$

The equivalent time domain expressions for both (6-14) and (6-15) are given respectively as follows:

$$\underline{w}(m+1) = \underline{F}\ \underline{b}(m+1)$$
$$= \underline{w}(m) + 2\mu_B\ \hat{\underline{X}}^*(m)\ \hat{\underline{e}}(m) \qquad (6\text{-}16)$$

and

$$\underline{w}(m+1) = \underline{F}\ \underline{b}(m+1)$$
$$= \underline{w}(m) + 2\mu_B\ \underline{R}_X^{-1}\ \hat{\underline{X}}^*(m)\ \hat{\underline{e}}(m) \qquad (6\text{-}17)$$

where

$$\hat{\underline{e}}(m) = \underline{F}\ \underline{v}(m) \qquad (6\text{-}18)$$

$$\hat{\underline{X}}(m) = \underline{F}\ \underline{P}^*(m)\ \underline{F}^{-1} \qquad (6\text{-}19)$$

and

$$\hat{\underline{R}}_X = \underline{F}\ \mathrm{diag}(E[|\hat{s}_m(0)|^2] \quad \ldots \quad E[|\hat{s}_m(N-1)|^2])\underline{F}^{-1}$$
$$= E[\ \hat{\underline{X}}(m)\ \hat{\underline{X}}^*(m)\ ] \qquad (6\text{-}20)$$

In this case the matrix $\hat{\underline{X}}(m)$ can be explicitly obtained via (3-54), that is

$$\hat{\underline{X}}(m) = \begin{bmatrix} x(mN+N-1) & x(mN) & \ldots & x(mN+N-2) \\ \vdots & \vdots & & \vdots \\ x(mN) & x(mN+1) & \ldots & x(mN+N-1) \end{bmatrix} \qquad (6\text{-}21)$$

Note that, in this case the square matrix $\hat{\underline{X}}(m)$ is a circulant matrix whose rows are obtained by successive right end-around shifts of the first row. Substituting (6-20) and (6-21) into (6-16) and (6-17), we have

$$\underline{w}(m+1) = \underline{w}(m) + 2\mu_B \sum_{i=0}^{N-1} \hat{e}_i(m)\ \tilde{\underline{x}}_m(i) \qquad (6\text{-}22)$$

and

$$\underline{w}(m+1) = \underline{w}(m) + 2\mu_B\ \underline{R}_X^{-1} \sum_{i=0}^{N-1} \hat{e}_i(m)\tilde{\underline{x}}_m(i) \qquad (6\text{-}23)$$

respectively. In (6-22) and (6-23), $\tilde{\underline{x}}^T{}_m(i)$ is denoted as

the (i+1) st row of $\hat{\underline{X}}(m)$ and $\hat{e}_i(m)$ is the i th element of $\hat{\underline{e}}(m)$, i.e.,

$$\hat{e}_i(m) = d(mN+N-1-i) - \tilde{y}_i(m) \qquad (6\text{-}24)$$

and $\tilde{y}_i(m)$ is defined as

$$\tilde{y}_i(m) = \tilde{\underline{x}}^T{}_m(i)\underline{w}(m) \qquad (6\text{-}25)$$

If $\underline{F}^{-1}$ can not be used to completely diagonalize the input autocorrelation, then we can expect that the BTDLMS algorithm will perform worse than the conventional time domain LMS algorithm [19,28]. This is because in the BTDLMS algorithm we neglect the off-diagonal elements of the transformed input autocorrelation in the implementation of adaptive filtering processes. That is because we have put two restrictions on the B.R. transformation filters (see chapter 3). Furthermore, in the BTDLMS algorithm with normalized DFT the estimated powers might be quite different from the corresponding eigenvalues due to the incomplete diagonalization of the input autocorelation matrix. However, fast convergence can still be achieved in the normalized algorithm compared to the algorithm without normalization [69].

6.3 The Discrete Cosine Transformed Domain Adaptive Filters

The discrete cosine transform (DCT) was developed by Ahmed et al. [58] and has been used in the area of digital processing for the purposes of pattern recognition and the generalized Wiener filtering [61].

The basis functions of the DCT are a class of discrete Chebyshev polynomials [58]. The transformed coefficients are conventionally computed by a 2N-point FFT algorithm [58]. Recently, various fast computational algorithms with relatively less computations have been derived by many authors [36,62,63]. Since the DCT has advantageous property in terms of decorrelating the transformed signal, it is chosen to examine the effect of incomplete diagonalization in the transformed domain adaptive algorithms.

In this case, the scalar variable, $w_i(k)$, can be represented by a set of discrete cosine functions (DCF) :

$$w_i(k) = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} p_n \, b_k(n) \cos \frac{(2i+1)\pi n}{2N} \qquad i = 0,1, \ldots, N-1 \qquad (6\text{-}26a)$$

with

$$p_n = \begin{cases} 1, & \text{if } n=0 \text{ and } n= N \\ \frac{1}{\sqrt{2}}, & \text{if } n\neq 0 \text{ or } n \neq N \end{cases} \qquad (6\text{-}26b)$$

Equivalently, we can express (6-26a) in the vector form :

$$\underline{w}(k) = \underline{C}_s^{-1} \, \underline{b}(k) \qquad (6\text{-}27)$$

where the square matrix, $\underline{C}_s^{-1}$, is the inverse of DCT and is given by

$$\underline{C}^{-1}_s = \sqrt{\frac{2}{N}} \begin{bmatrix} \sqrt{1/2} & \cos\frac{\pi}{2N} & \cdots & \cos\frac{(N-1)\pi}{2N} \\ \sqrt{1/2} & \cos\frac{3\pi}{2N} & \cdots & \cos\frac{3(N-1)\pi}{2N} \\ \vdots & \vdots & & \vdots \\ \sqrt{1/2} & \cos\frac{(2N-1)\pi}{2N} & \cdots & \cos\frac{(N-1)(2N-1)\pi}{2N} \end{bmatrix} \qquad (6\text{-}28)$$

The transformed weight vector, from (6-31) is given as

$$\underline{b}(k) = \underline{C}_s \, \underline{w}(k) \qquad (6\text{-}29)$$

where $\underline{C}_s$ is the DCT matrix and is given as

$$\underline{C}_s = \sqrt{\frac{2}{N}} \begin{bmatrix} \sqrt{1/2} & \sqrt{1/2} & \cdots & \sqrt{1/2} \\ \cos\frac{\pi}{2N} & \cos\frac{3\pi}{2N} & \cdots & \cos\frac{(2N-1)\pi}{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \cos\frac{(N-1)\pi}{2N} & \cos\frac{3(N-1)\pi}{2N} & \cdots & \cos\frac{(N-1)(2N-1)\pi}{2N} \end{bmatrix} \qquad (6\text{-}30)$$

Note that, $\underline{C}_s$ is an unitary matrix:

$$\underline{C}^T{}_s = \underline{C}^{-1}{}_s \qquad (6\text{-}31)$$

6.3.1 The TDLMS Adaptive Filter with DCT

The TDLMS adaptive algorithm with DCT can be derived by proceeding in similar way as was used to derive the TDLMS adaptive algorithm with DFT. We now obtain the TDLMS adaptive algorithm with DCT. Substituting matrix $\underline{Q}$ by matrix $\underline{C}_s^{-1}$ in (3-15a), we have

$$\underline{s}(n) = (\underline{C}^{-1}{}_s)^T \, \underline{x}(n) \qquad (6\text{-}32)$$

Further, from (6-20b) the weight vector updated equation is given by:

$$\underline{b}(n+1) = \underline{b}(n) + 2\,\mu\, e(n)\, \underline{s}(n) \qquad (6\text{-}33)$$

Since the DCT matrix is a unitary matrix (6-31), the equivalent time domain expression of (6-33) can be obtained as follows :

$$\underline{w}(n+1) = \underline{C}^{-1}_{s}\,\underline{b}(n+1)$$
$$= \underline{C}^{-1}_{s}\,\underline{b}(n) + 2u\,e(n)\,\underline{C}^{-1}_{s}\,\underline{s}(n)$$
$$= \underline{w}(n) + 2\,\mu\,e(n)\,\underline{x}(n). \qquad (6\text{-}34)$$

Again, we recognize that (6-34) is the same as the time domain LMS adaptive algorithm.

Similarly, for the TDLMS adaptive algorithm with normalized DCT, the tap-weights update equation can be written as (3-35)

$$c_{n+1}(i) = c_n(i) + 2\mu_i\,e(n)\,z_n(i)$$

with

$$\mu_i = \frac{\mu}{E[|z_n(i)|^2]}\,, \quad i=0,1,\ldots,N-1 \qquad (6\text{-}35)$$

where $c_n(i)$ and $z_n(i)$ are the ith component of $\underline{c}(n)$ and $\underline{z}(n)$ respectively. Here, $\underline{c}(n)$ and $\underline{z}(n)$ are defined as

$$\underline{c}(n) = \underline{C}_s\,\underline{w}(n) \qquad (6\text{-}36)$$

and

$$\underline{z}(n) = \underline{C}_s\,\underline{x}(n) \qquad (6\text{-}37)$$

Again,(6-35) can be viewed as the discrete cosine transformed domain implementation of the normalized K-L TDLMS adaptive algorithm in the time domain. The computational complexity of implementing (6-35) can be further reduced. To do so, a recursive equation for obtaining $z_n(i)$ is obtained (Appendix F) as

$$z_n(i) = \cos\frac{i\pi}{N}\,z_{n-1}(i) + \sin\frac{i\pi}{N}\,z^{Im}_{n-1}(i)$$
$$+\sqrt{\frac{2}{N}}\,p_i\,[\,x(n) - (-1)^i\,x(n-N)]\cos\frac{\pi i}{2N}$$
$$i=0,1,\ldots,N-1 \qquad (6\text{-}38a)$$

and

$$z^{Im}{}_n(i) = \cos\frac{i\pi}{N}\, z^{Im}{}_{n-1}(i) - \sin\frac{i\pi}{N}\, z_{n-1}(i) - \sqrt{\frac{2}{N}}\, p_i[x(n)-(-1)^i\, x(n-N)]\sin\frac{i\pi}{2N}$$

$$i = 0,1,\ldots,N-1 \qquad (6\text{-}38b)$$

where

$$z^{Im}{}_n(i) = p_i\ \mathrm{Im}\{[\exp(-j(\pi i)/2N]\, z_n(i)\} \qquad i=0,1,\ldots,N-1 \qquad (6\text{-}39)$$

Here $z_n(i)$ is defined as

$$z_n(i) = \sqrt{\frac{2}{N}} \sum_{p=0}^{2N-1} \tilde{x}(n-p)\exp(-jp2i\pi/2N) \qquad i=0,1,\ldots,N-1 \qquad (6\text{-}40)$$

with

$$\tilde{x}(n-p) = \begin{cases} x(n-p), & p=0,1,\ldots,N-1 \\ 0, & p=N, N+1, \ldots, 2N \end{cases} \qquad (6\text{-}41)$$

The computational complexity compared to the fast DCT proposed by Yip et al. [36] is tabulated in Table 6.1. From Table 6.1, we can see, in general, for higher order of transform ($N \geq 512$), recursive method requires less computations. Furthermore, the implementation of the recursive equation is simpler than the fast DCT transforms.

6.3.2 The Effect of Convergence Rate Due to the Incomplete Diagonalization

As mentioned before, the DCT is asymtotically equivalent to the K-L transform for a stationary, finite-order Markov process signal sequence. This means that for such signal sequence, the discrete cosine functions (DCF) are very similar to the eigenvectors of the autocorrelation matrix. In

N	Recursive Method		Yip & Rao's Method	
	Addition	Multiplication	Addition	Multiplication
	4N - 3	5N - 9	(1)	: (2)
8	29	31	36	14
16	61	71	97	36
32	125	151	244	88
64	253	311	587	208
128	509	631	1370	480
256	1021	1271	3129	1088
512	2045	2551	7032	2432
1024	4093	5111	15607	5376
2048	8189	10231	33793	11776

(1) Addition : $(3N/2 - 1) \log_2 N + N/4 + 1$.

(2) Multiplication : $(N/2) \log_2 N + N/4$.

Table 6.1 Comparison of Computation Complexity for the Recursive Method and Yip & Rao's Method.

the following, a stationary, first-order Markov process signal sequence is used to examine the effect of incomplete diagonalization for the DCT in the transformed domain LMS adaptive algorithm.

In this case, the autocorrelation matrix of input signal vector is given as

$$\underline{R} = \sigma_x^2 \begin{bmatrix} 1 & \rho & \cdots & \rho^{N-1} \\ \rho & 1 & \cdots & \rho^{N-2} \\ \vdots & \vdots & & \vdots \\ \rho^{N-1} & \rho^{N-2} & \cdots & 1 \end{bmatrix} \tag{6-42}$$

where σ_x^2 is the variance of the input process and ρ is the correlation coefficient, i.e.,

$$\rho = \frac{E[x(n)x(n-1)]}{\sigma_x^2} \tag{6-43}$$

In order to see the similarity between the DCT and the K-L transform, we consider an example with N=4, $\sigma_x^2=1$ and $\rho = 0.95$. In this case the eigenvalues of $\underline{R}$ and the corresponding eigenvectors are given by

$$\lambda_1 = 3.75677 , \quad \underline{q}_1 = [\, 0.5 \quad 0.5 \quad 0.5 \quad 0.5]^T$$
$$\lambda_2 = 0.16265 , \quad \underline{q}_2 = [\, 0.367 \quad 0.151 \quad -0.349 \quad -0.849]^T$$
$$\lambda_3 = 0.05061 , \quad \underline{q}_3 = [\, 0.367 \quad -0.151 \quad -0.349 \quad 0.849]^T$$

and

$$\lambda_4 = 0.02997 , \quad \underline{q}_4 = [\, 0.5 \quad -0.5 \quad 0.5 \quad -0.5\,]^T$$

(6-44)

On the other hand, for N = 4, the DCF from (6-30) are given as

$$[\ 0.5 \quad 0.5 \quad 0.5 \quad 0.5\ ],$$

$$[\ 0.635 \quad 0.271 \quad -0\ 271 \quad 0.653\ ],$$

$$[\ 0.5 \quad -0.5 \quad -0.5 \quad 0.5\ ],$$

$$[\ 0.271 \quad -0.653 \quad 0.653 \quad -0.271\ ] \qquad (6\text{-}45)$$

Furthermore, the transformed autocorrelation matrix with DCT is given by

$$\underline{R}^{\#} = \underline{C}^{-1}_{s}\ \underline{R}\ \underline{C}_{s}$$

$$= \begin{bmatrix} 3.75619 & 0 & -0.04631 & 0 \\ 0 & 0.16265 & 0 & -0.00084 \\ -0.04631 & 0 & -0.05119 & 0 \\ 0 & -0.00084 & 0 & 0.02998 \end{bmatrix} \qquad (6\text{-}46)$$

From (6-46), we can see that the diagonal elements of $\underline{R}^{\#}$ are very close to the eigenvalues of $\underline{R}$, (6-44). However, matrix $\underline{R}^{\#}$ is not a diagonal matrix because the off-diagonal elements are not all zeros. The effect of the incomplete diagonalization will be shown now.

Recall that, the TDLMS adaptive algorithm with the normalized DCT is (6-35):

$$c_{n+1}(i) = c_n(i) + 2\ \mu_i\ e(n)\ z_n(i),$$

with

$$\mu_i = \frac{\mu}{E[|z_n(i)|^2]}, \quad i = 0,1,\ldots, N\text{-}1. \qquad (6\text{-}35)$$

Since the diagonal terms of $\underline{R}^{\#}$ are very close to the eigenvalues of $\underline{R}$ so that we simply assume that

$$\lambda_{i+1} = E[\, z^2_n(i)\,] \quad , \; i=0,1,\ldots,N-1. \qquad (6\text{-}47)$$

Using (6-47), the vector form of (6-35) is given by

$$\underline{c}(n+1) = \underline{c}(n) + 2\,\mu\, \underline{R}_\lambda^{-1}\, e(n)\, \underline{z}(n) \qquad (6\text{-}48)$$

Note that, (6-48) is an equivalent expression of the TDLMS adaptive algorithm with normalized K-L transform. However, the equivalent time domain expression of (6-48) is not equivalent to the time domain expression of the normalized K-L TDLMS adaptive filtering algorithm :

$$\begin{aligned}\underline{w}(n+1) &= \underline{C}^{-1}{}_s\, \underline{c}(n+1) \\ &= \underline{C}^{-1}{}_s\, \underline{c}(n) + 2\,\mu\, \underline{C}^{-1}{}_s \underline{R}_\lambda^{-1} e(n)\, \underline{s}(n) \\ &= \underline{w}(n) + 2\mu\, \tilde{\underline{R}}^{-1}\, e(n)\, \underline{x}(n) \qquad (6\text{-}49)\end{aligned}$$

where

$$\tilde{\underline{R}} = \underline{C}^{-1}{}_s\, \underline{R}_\lambda^{-1}\, \underline{C}_s \qquad (6\text{-}50)$$

Note that matrix $\tilde{\underline{R}}$ is not the same as matrix $\underline{R}$. To further study the effect of convergence rate due to the incomplete diagonalization, we then write the mean value of (6-49) into the error weight vector recursive equation as in (4-5b):

$$\begin{aligned}\underline{w}^e(n+1) &= E[\, \underline{w}(n+1)] - \underline{w}_{opt} \\ &= E[\underline{w}(n)] + 2\,\mu\, \tilde{\underline{R}}^{-1}\, \underline{R}\, \{\underline{w}_{opt} - E[\underline{w}(n)]\} - \underline{w}_{opt} \\ &= [\, \underline{I} - 2\mu\, (\, \underline{R}^{-1}\, \tilde{\underline{R}})^{-1}] \, \underline{w}^e(n) \qquad (6\text{-}51a)\end{aligned}$$

or

$$\underline{w}^e(n+1) = [\, \underline{I} - 2\mu\, \tilde{\underline{R}})^{-1} \,]\, \underline{w}^e(n) \qquad (6\text{-}51b)$$

Since the eigenvalues of $\underline{R}^{-1}\tilde{\underline{R}}$ are not same, its eigenvalue spread as defined in (sec.2.1)is not unity.This implies that, in general, the convergence rate of the TDLMS adaptive filter with normalized DCT is slower than the normalized K-L TDLMS adaptive algorithm. The eigenvalues of $\underline{R}^{-1}\tilde{\underline{R}}$ are as follows:

$$\tilde{\lambda}_1 = 1.1181, \quad \tilde{\lambda}_2 = 1.0122, \quad \tilde{\lambda}_3 = 0.9881,$$

and

$$\tilde{\lambda}_4 = 0.9045 \qquad (6\text{-}52)$$

From computer simulation, we also observed that the eigenvalue spread of $\underline{R}^{-1}\,\underline{\tilde{R}}$ is decreased when the value of ρ is increased. However, the eigenvalue spread of the TDLMS adaptive algorithm with normalized DCT is significantly improved compared to the time domain LMS algorithm. Consequently, the convergence rate with very large eigenvalue spread will, in general, converge faster than the time domain LMS algorithm.

6.3.3 The BTDLMS Adaptive Filter with DCT

The DCT matrix is a real unitary matrix and can be similarly applied to the BTDLMS adapfive algorithm as the DFT matrix. In this case, the transformed desired signal vector and the transformed input vector are defined by

$$\hat{\underline{d}}(m) = \underline{C}_s\,\underline{d}(m) \qquad (6\text{-}53a)$$

and

$$\hat{\underline{s}}(m) = \underline{C}_s\,\hat{\underline{x}}(m) \qquad (6\text{-}53b)$$

respectively. Also, the time domain weight vector at m th iteration is given as

$$\underline{w}(m) = \underline{C}^{-1}{}_s\,\underline{b}(m) \qquad (6\text{-}54)$$

Again, from (3-47), the updated equation of the tap-weights of the BTDLMS adaptive algorithm with DCT is written as

$$b_{m+1}(i) = b_m(i) + 2\mu_B\, v_m(i)\, \hat{s}_m(i)$$

$$i=0,1, \ldots, N-1 \qquad (6-55)$$

Alternatively, the BTDLMS adaptive algorithm with normalized DCT can be expressed as (3-49)

$$b_{m+1}(i) = b_m(i) + 2\mu_i \, v_m(i) \, \hat{s}_m(i)$$

with

$$\mu_i = \frac{\mu_B}{E[\,|\hat{s}_m(i)|^2]}, \qquad i=0,1, \ldots, N-1 \qquad (6-56)$$

Now, proceeding in the similar way as we obtained the BTDLMS algorithms with DFT and normalized DFT, we have

$$\underline{w}(m+1) = \underline{C}^{-1}_s \, \underline{b}(m+1)$$
$$= \underline{w}(m) + 2\mu_B \, \hat{\underline{X}}(m) \, \hat{\underline{e}}(m) \qquad (6-57)$$

and

$$\underline{w}(m+1) = \underline{C}^{-1}_s \, \underline{b}(m+1)$$
$$= \underline{w}(m) + 2\mu_B \, \underline{R}^{-1}_s \, \hat{\underline{X}}(m) \, \hat{\underline{e}}(m) \qquad (6-58)$$

with

$$\hat{\underline{X}}(m) = \underline{C}^{-1}_s \, \underline{P}(m) \, \underline{C}_s \qquad (6-59a)$$

$$\underline{R}_s = \underline{C}^{-1}_s \, \hat{\underline{R}} \, \underline{C}_s$$
$$= E[\,\hat{\underline{X}}(m) \, \hat{\underline{X}}(m)\,] \qquad (6-59b)$$

and

$$\hat{\underline{e}}(m) = \underline{C}^{-1}_s \, \underline{v}(m)$$
$$= \underline{d}(m) - \hat{\underline{X}}(m) \, \underline{w}(m) \qquad (6-59c)$$

Here, $\underline{P}(m)$ and $\hat{\underline{R}}$ were defined in (3-40) and (3-45) respectively. We also note that (6-57) and (6-58) are the time domain expressions of (6-55) and (6-56) respectively.

6.4 The Lattice Transformed Domain Adaptive Filter

It is widely known that the digital lattice filter is essential in various engineering applications [64-66]. This is because the digital lattice filter inherits many of the properties of the analog lattice filter. Furthermore, a digital lattice filter is not sensitive to finite-word length arithmetic errors. In this section, the digital lattice filter or simply lattice filter is used as a discrete transform rather than a filter itself. That is, the lattice filter is applied to the transformed domain LMS adaptive filtering problems. In order to employ the lattice filter to the transformed domaim adaptive algorithm, we need to briefly review the fundamental structure of the lattice filter.

6.4.1 The Lattice Filter Structure

The fundamental equation describing the lattice filter structure is as follows:

$$\begin{aligned} e_f(m+1:k) &= e_f(m:k) - k_m\, e_b(m:k-1) \\ e_b(m+1:k) &= e_b(m:k-1) - k_m\, e_f(m:k) \end{aligned} \qquad (6\text{-}60)$$

which is depicted in figure 6.1. Here $e_f(m:k)$ and $e_b(m:k)$ are the mth stage forward prediction error (FPE) signal and the backward prediction error (BPE) signal, at kth sampling instant, respectively. The scalar values, k_m, at mth stage, are known as the reflection coefficients or

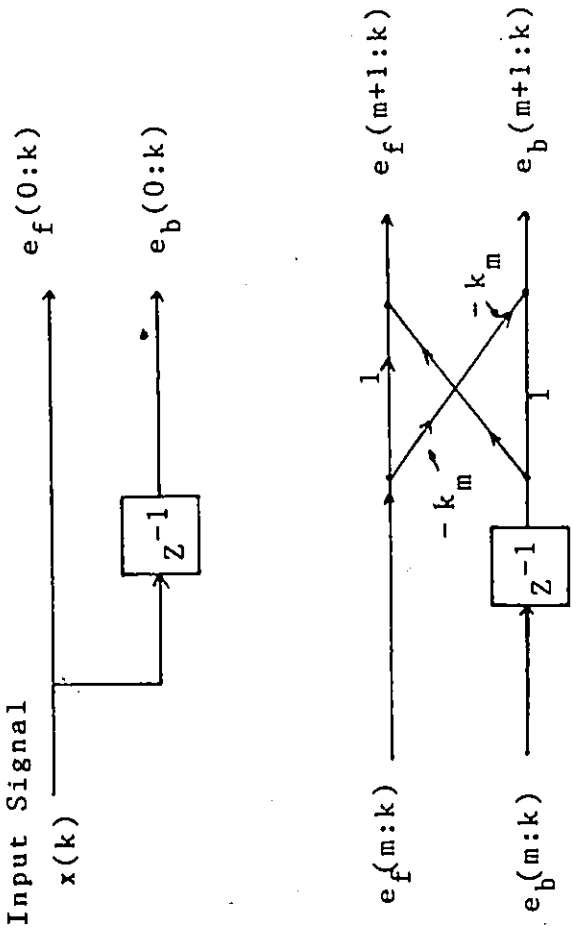


Figure 6.1 Fundamental Structure of Lattice Filter.

the partial correlation (PARCOR) coefficients. Assume that the input data is a wide-sense stationary process with zero-mean, the values of k_m are obtained by minimizing the mean-square values of $e_f(m+1:k)$ and $e_b(m+1:k)$, where the optimum k_m can be defined in many ways [64,67].In practice, the PARCOR coefficients are obtained recursively and depend on how the optimum k_m is defined [64,67].

Since the lattice structure is an alternative FIR filter realization [68,69], which can be considered as a cascade of single tap prediction error filter (PEF). It has been shown [70] that the lattice structure illustrated in figure 6-1 can be realized as a transversal tapped delay-line filter. To see this relationship, let us consider an example for four stages lattice filter realization showed in figure 6-2. Refer from figure 6-2, the BPE signals at kth sampling instant can be written as

$$\begin{aligned} e_b(0:k) &= x(k) \\ e_b(1:k) &= x(k-1) - a_{03}\, x(k) \\ e_b(2:k) &= x(k-2) - [\, a_{12}\, x(k-1) + a_{02} x(k)] \\ e_b(3:k) &= x(k-3) - [\, a_{21} x(k-2) + (a_{11} x(k-1) \\ &\qquad + a_{01}\, x(k)\,)] \end{aligned} \tag{6-61}$$

The matrix form of (6-61) can be expressed as

$$\underline{e}_b(k) = \underline{Q}_L\, \underline{x}(k) \tag{6-62}$$

where the backward prediction error signal vector is denoted as

$$\underline{e}_b(k) = [\, e_b(0:k) \quad e_b(1:k) \ \ldots\ e_b(3:k)\,]^T \tag{6-63}$$

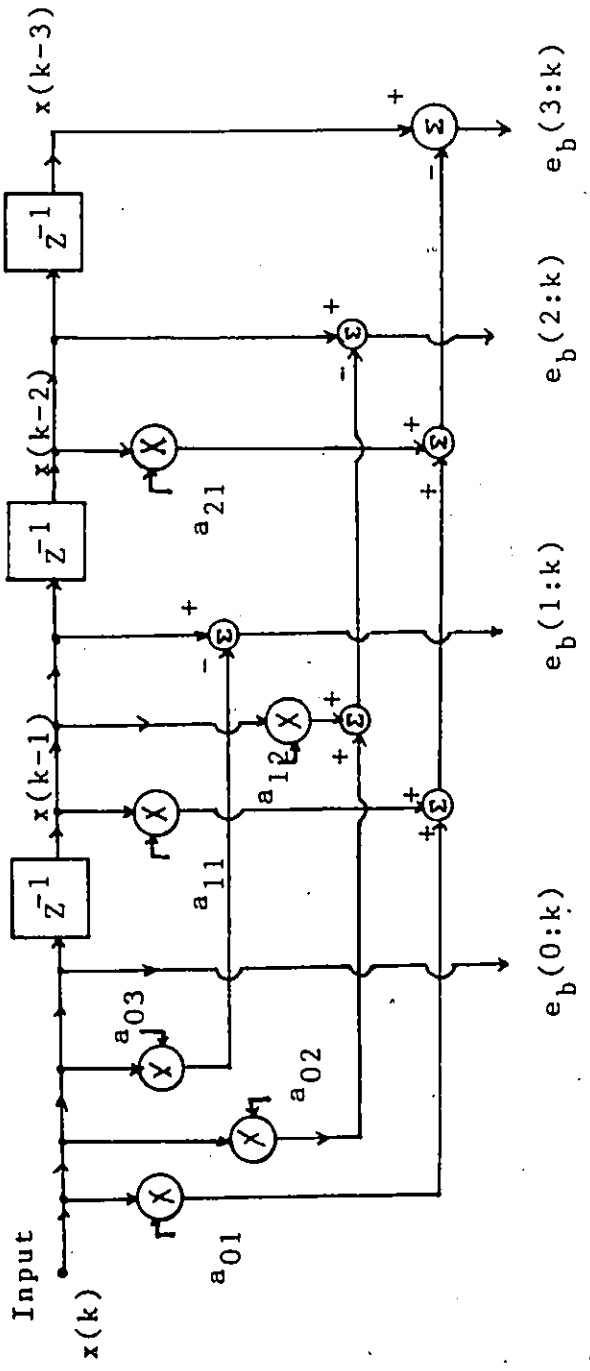


Figure 6.2 Equivalent Realization of the Lattice Filter as a Transversal Filter.

and the matrix $\underline{Q}_L$ is defined by

$$\underline{Q}_L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -a_{03} & 1 & 0 & 0 \\ -a_{02} & -a_{12} & 1 & 0 \\ -a_{01} & -a_{11} & -a_{21} & 1 \end{bmatrix} \tag{6-64}$$

and is a lower triangluar matrix. For a stationary signal it can be shown that [71] the autocorrelation matrix of the backward prediction error signal is a diagonal matrix:

$$\begin{aligned} & E[\, \underline{e}_b{}^T(k)\, \underline{e}_b(k)\,] \\ &= \underline{Q}_L\, \underline{R}\, \underline{Q}^T_L \\ &= \underline{R}_b \end{aligned} \tag{6-65}$$

or

$$\underline{R}^{-1} = \underline{Q}^T_L \underline{R}^{-1}_b \underline{Q}_L \tag{6-66}$$

We can show that (Appendix G), in general, the lower triangular matrix, $\underline{Q}_L$ is not a unitary matrix, i.e.,

$$\underline{Q}_L{}^T \neq \underline{Q}^{-1}_L \tag{6-67}$$

6.4.2 The TDLMS Adaptive Filter with Lattice Transform

The Lattice transform described above can be applied to the TDLMS adaptive filter structure. To do so, we define

$$\underline{w}(k) = \underline{Q}^T_L\, \underline{b}(k) \tag{6-68}$$

and

$$\begin{aligned} \underline{s}(k) &= \underline{e}_b(k) \\ &= \underline{Q}_L\, \underline{x}(k) \end{aligned} \tag{6-69}$$

where $\underline{s}(k)$ is the BPE signal vector. Similarly, the TDLMS adaptive filtering algorithm with Lattice transform and normalized Lattice transform can be expressed as

$$\underline{b}(k+1) = \underline{b}(k) + 2\mu\, e(k)\, \underline{s}(k) \qquad (6\text{-}70)$$

and

$$\underline{b}(k+1) = \underline{b}(k) + 2\mu\, e(k)\, \underline{R}^{-1}_{b}\, \underline{s}(k) \qquad (6\text{-}71)$$

respectively. Again, to obtain the time domain expression of (6-70), we pre-multiply both sides of (6-70) by $\underline{Q}^{T}_{L}$,

$$\begin{aligned}\underline{w}(k+1) &= \underline{Q}^{T}_{L}\, \underline{b}(k+1) \\ &= \underline{Q}^{T}_{L}\, \underline{b}(k) + 2\mu\, e(k)\, \underline{Q}^{T}_{L}\, \underline{s}(k) \\ &= \underline{w}(k) + 2\mu\, e(k)\, \underline{Q}^{T}_{L}\, \underline{Q}_{L}\, \underline{x}(k) \qquad (6\text{-}72)\end{aligned}$$

From (6-72), we can see that the equivalent time domain expression of TDLMS adaptive algorithm with lattice transform is not equivalent to the time domain LMS adaptive algorithm. This is due to the fact that the matrix $\underline{Q}_{L}$ is not a unitary matrix. Similarly, the equivalent time domain expression of the TDLMS adaptive algorithm with normalized lattice transform is given as

$$\begin{aligned}\underline{w}(k+1) &= \underline{w}(k) + 2\mu\, e(k)\, \underline{Q}^{T}_{L}\, \underline{R}^{-1}_{b}\, \underline{Q}_{L}\, \underline{x}(k) \\ &= \underline{w}(k) + 2\mu\, e(k)\, \underline{R}^{-1}\, \underline{x}(k) \qquad (6\text{-}73)\end{aligned}$$

We recognize that (6-73)is the normalized K-L TDLMS algorithm in the time domain. This implies that, ideally, the TDLMS adaptive filter with normalized lattice transform can achieve the same performance as the normalized K-L TDLMS adaptive algorithm. It should be mentioned here, that the matrix $\underline{Q}_{L}$ is signal dependent and varies at each iteration, therefore, it is not suitable for the use in the BTDLMS adaptive filtering problems.

6.5 The Transmultiplexer Transformed Domain Adaptive Filter

The transmultiplexer (TM) is a technique which has been used in telephone system to efficiently translate the signals between frequency division multiplexed (FDM) and time division multiplexed (TDM) formats [72-75]. The FDM is a common means of combining a number of individual analog signals (channels) into a group which can be transmitted efficiently over one wideband communication. Whereas the TDM is a digital technique which interleaves bits or characters, one from each attached channel, and transmit them at higher speed down the high-speed line.

The TM transform (TMT) is a transformation which utilizes the concept of all digital FDM-TDM translation [74-76] and has been used in multirate digital signal processing [76] and adaptive filtering [70,77] problems. The objective of this section is to apply the TMT to the transformed domain adaptive filtering algorithms. To do so, a simplified form of the TMT is derived.

6.5.1 The Complex Modulation Scheme of the FDM-to-TDM Transmultiplexing Transformation

In this part, the TMT in a matrix form is obtained. Consider the power spectrum $S_x(w)$ of the signal depicted in figure 6-3, the spectral interpretation of FDM-to-TDM signal translation can then be illustrated as in figure 6-4, in

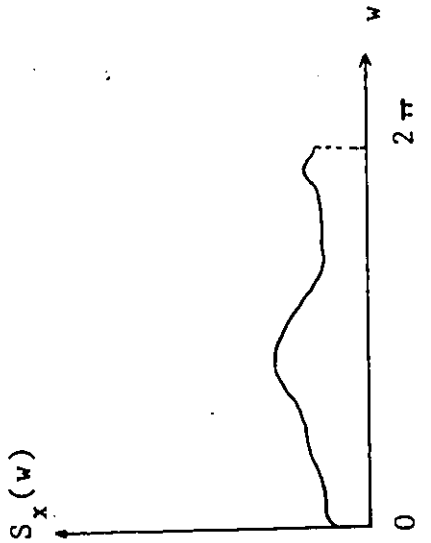


Figure 6.3 The Power Spectrum of Input Signal x(t).

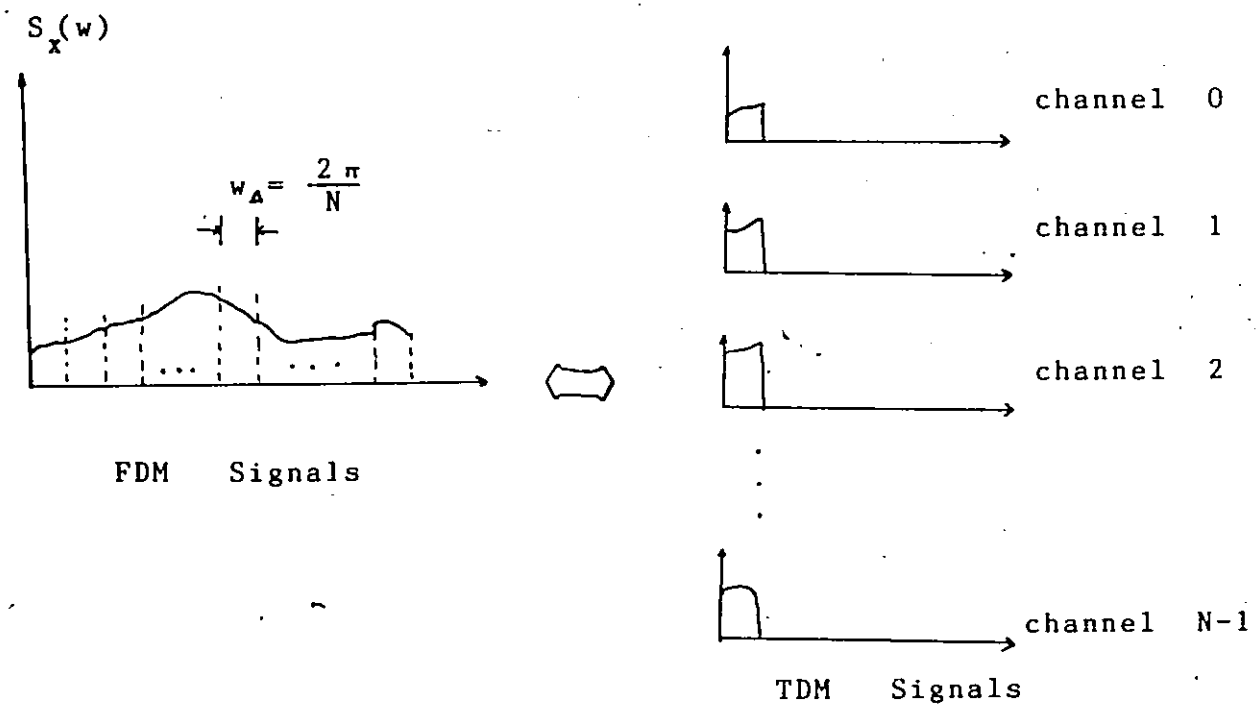


Figure 6.4 Spectral Interpretation of FDM-to-TDM Signal Translation.

which the power spectrum of the signal is partitioned into N segments with each segment corresponding to one of the FDM channel signals. The ith channel of TDM signals are obtained by demodulating the ith segment of the FDM channels signals into a baseband signals followed by an ideal lowpass filter. Since the sampling rate of the FDM signals is higher than the TDM signals, so that in each channel of the FDM-TDM translator the sampling rate is efficiently decreased. The block diagram of the i th channel FDM → to - TDM transmultiplexing via the complex demoduation scheme is shown in Figure 6.5. Assume that the channel bandwidth is w_Δ, then theoretically, the sampling rate of each of the channel signals can be reduced by a factor M [75]:

$$M \leq \frac{2\pi}{w_\Delta} \qquad (6\text{-}74)$$

If the bands in an N-channel complex filter bank are uniformly and contiguously spaced such that

$$w_\Delta = \frac{2\pi}{N} \qquad (6\text{-}75)$$

and are critically sampled, i.e., the equality applies in (6-74), then we have M = N and the filter bank is referred to as a critically sampled filter bank. Now, if the i = 0 channel is centered at $w = \pi / N$ then, the bands are at frequencies

$$w_i = \frac{2\pi i}{N} + \frac{\pi}{N}, \quad i = 0,1,\ldots, N\text{-}1 \qquad (6\text{-}76)$$

It is worth noting that, if the lowpass filter shown in Figure 6.5 is an ideal lowpass filter:

$$H(e^{jw}) = \begin{cases} 1, & |w| \leq \frac{\pi}{M} \\ 0, & \text{otherwise} \end{cases} \qquad (6\text{-}77)$$

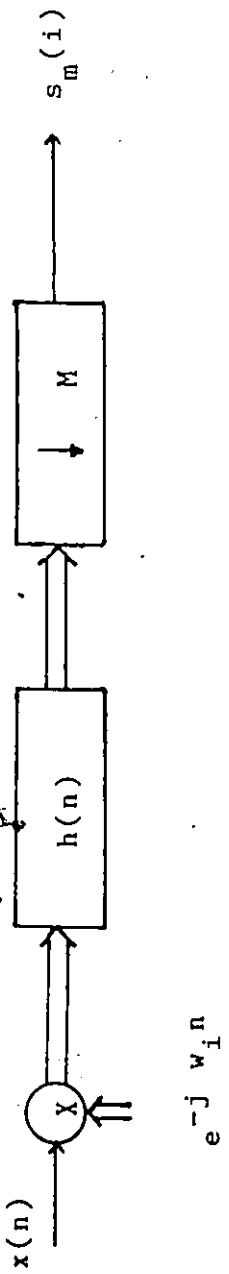


Figure 6.5 The i th Channel FDM-to-TDM Transmultiplexing via the Complex Demodulation Scheme.

then, the output samples obtained in the TDM channels are orthogonal to each other within the neighborhood of N samples. This is because that the cross-power spectrum between any two TDM channels is null.

From figure 6-5, the ith channel output, $s_m(i)$ can be expressed as [75]

$$s_m(i) = \sum_{n=-\infty}^{\infty} h(mM-n) \left[x(n) \exp\{-j\frac{(i+\frac{1}{2})\pi}{N} n\} \right]$$
$$i = 0,1,\ldots,N-1 \qquad (6\text{-}78)$$

Applying the change of variables

$$n = rM + t$$
$$= rN + t \quad (M = N) \qquad (6\text{-}79)$$

and with some manipulations (6-78) can be written as

$$s_m(i) = \sum_{t=0}^{N-1} \{ \sum_{n=-\infty}^{\infty} \{h[(m-r)N-t] \exp\{j \pi(m-r)\} x(rN+t) \exp\{-jm\pi\} \} \exp[-j \frac{\pi(i+1/2)}{N}]$$
$$= \exp\{-jm\pi\} \{\sum_{t=0}^{N-1} \{ [\bar{p}_t(m) \exp\{j\pi m\}] \circledast x_t(m)\} \exp\{-j\frac{2\pi it}{N}\} \exp\{-j \frac{\pi t}{N} \}\}$$
$$, i = 0,1,\ldots, N-1 \qquad (6\text{-}80)$$

where $\circledast$ denotes the convolution for two functions and $\bar{p}_t(m)$ and $x_t(m)$ are defined as

$$\bar{p}_t(m) = h(mN-t)$$

and

$$x_t(m) = x(mN+t), \quad t=0,1,\ldots,N-1 \qquad (6\text{-}81)$$

Now, (6-80) can be realized via the polyphase structure [76] of the complex demodulation scheme of FDM-to-TDM transmulti - plexer and is depicted in figure 6-6(a). Alternatively, the polyphase network can be realized by a time varying FIR filter [78] and this is shown in figure 6-6(b).

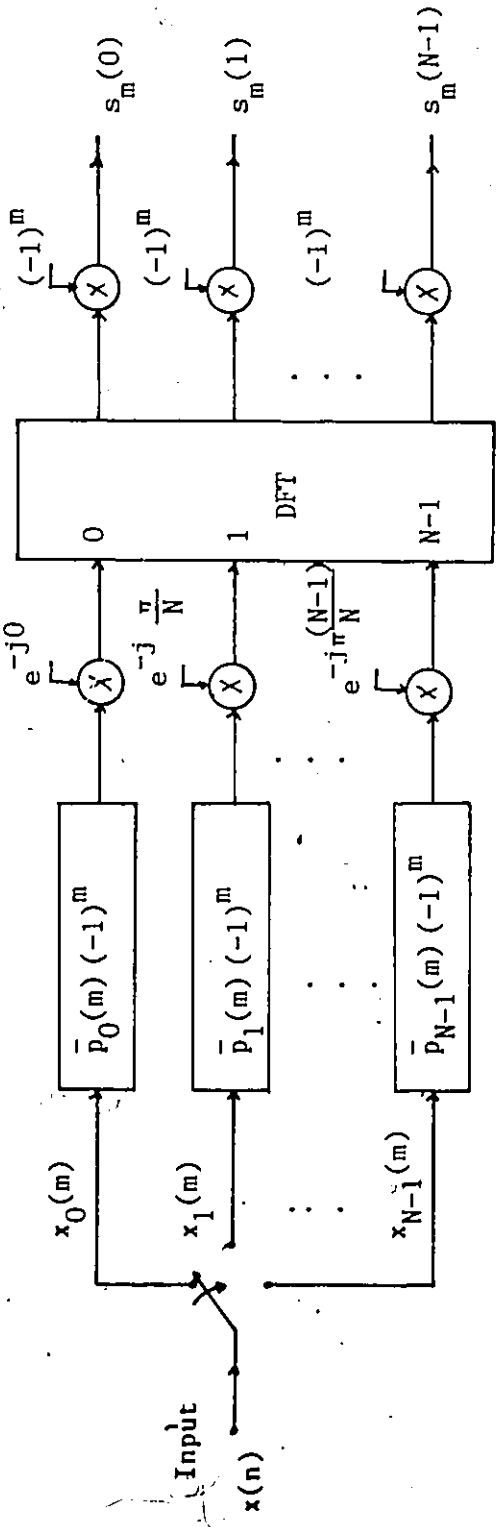
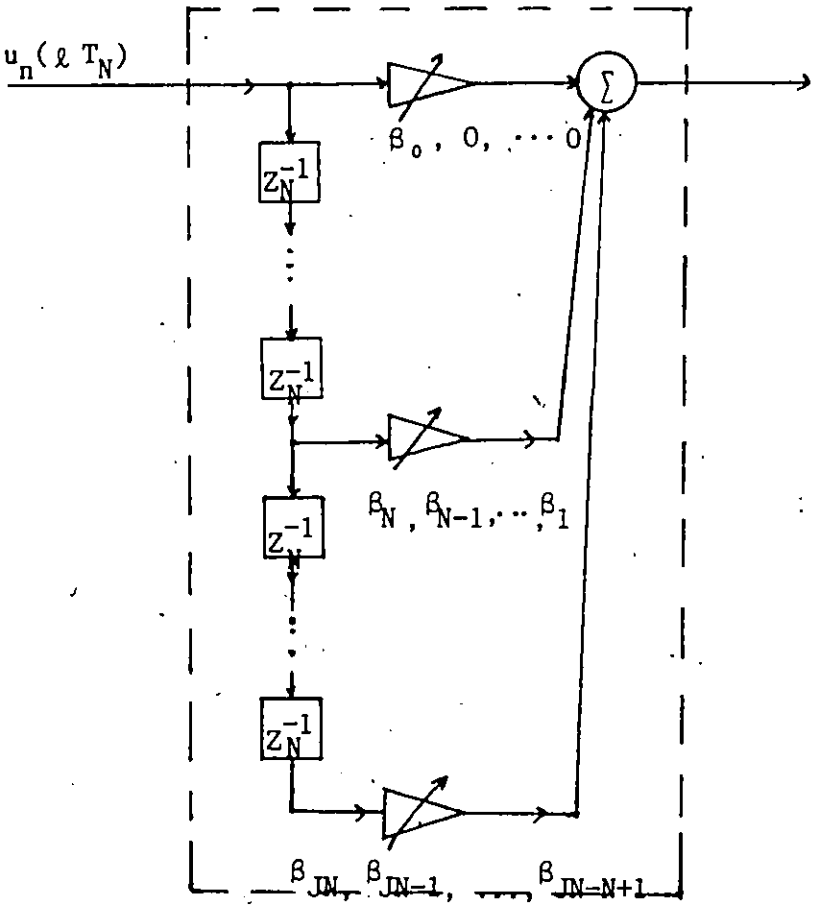


Figure 6.6(a) The Polyphase Structure of the Complex Demodulation Scheme FDM-to-TDM Transmultiplexing.



Periodically Varying FIR Filter

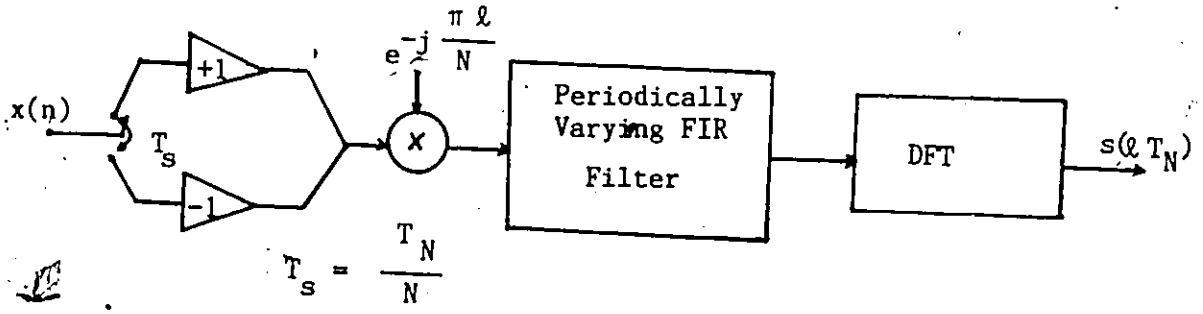


Figure 6.6(b) An Alternative Realization of the Polyphase Network Using a Time-Varying FIR Filter.

Since in practice, the ideal lowpass filter is unrealizable, so that a FIR filter with order L is designed to approximate the ideal filter characteristic. The impulse response of such Lth order FIR filter is given as

$$h(n) = \begin{cases} 0, & n < 0 \\ \beta_n, & 0 \leq n \leq L \\ 0, & n > N \end{cases} \qquad (6\text{-}82)$$

where β_n is the nth coefficient of the FIR filter. Let L be given by L = J N, J being an integer and applying (6-82) to (6-81), we have from Figure 6.6, the relation between the input signals and the output signals in the matrix form:

$$\underline{s} = \underline{I}'\underline{\phi}\,\underline{\Gamma}\,\underline{P}\,\underline{x} = \underline{Q}_{TM}\,\underline{x} \qquad (6\text{-}83a)$$

where

$$\underline{Q}_{TM} = \underline{I}'\underline{\phi}\,\underline{\Gamma}\,\underline{P} \qquad (6\text{-}83b)$$

Note that $\underline{Q}_{TM}$ is referred to as a TMT. In (6-83a), $\underline{s}$ and $\underline{x}$ are the output and input vectors and are denoted respectively as

$$\underline{s} = [\ \underline{s}_0\ \underline{s}_1\ \cdots\ \underline{s}_i\ \cdots\ ]^T \qquad (6\text{-}84a)$$

and

$$\underline{x} = [\ \underline{r}_0\ \underline{r}_1\ \cdots\ \underline{r}_i\ \cdots\ ]^T \qquad (6\text{-}84b)$$

with

$$\underline{s}_i = [\ s(iN)\ s(iN+1)\ \ldots\ s(iN+N-1)\ ]^T \qquad (6\text{-}85a)$$

and

$$\underline{r}_i = [\ x(iN)\ x(iN+1)\ \ldots\ x(iN+N-1)\ ]^T \qquad (6\text{-}85b)$$

In (6-83a), $\underline{I}'$ and $\underline{\phi}$ are the block daigonal matrices, each block of which is of size NxN, i.e.,

$$\underline{I}' = \begin{bmatrix} \underline{I} & & & \\ & -\underline{I} & & \\ & & \underline{I} & \\ & & & -\underline{I} \\ & & & & \ddots \end{bmatrix} \qquad (6\text{-}86a)$$

and

$$\underline{\Phi} = \begin{bmatrix} \underline{\Phi}_0 & & & \\ & \underline{\Phi}_0 & & \\ & & \underline{\Phi}_0 & \\ & & & \underline{\Phi}_0 \\ & & & & \ddots \end{bmatrix} \qquad (6\text{-}86b)$$

where $\underline{I}$ is an NxN identity matrix and $\underline{\Phi}_0$ is defined as

$$\underline{\Phi}_0 = \begin{bmatrix} 1 & & & \\ & e^{-j\frac{\pi}{N}} & & \\ & & \ddots & \\ & & & e^{-j\frac{(N-1)\pi}{N}} \end{bmatrix} \qquad (6\text{-}87)$$

The filter bank, $\underline{\Gamma}$, is designated as

$$\underline{\Gamma} = \begin{bmatrix} \underline{F}' & & & \\ & \underline{F}' & & \\ & & \underline{F}' & \\ & & & \ddots \end{bmatrix} \qquad (6\text{-}88)$$

with the DFT matrix, $\underline{F}'$ as

$$\underline{F}' = \sqrt{N}\,\underline{F} \qquad (6\text{-}89)$$

where $\underline{F}$ was defined in (6-2) and the matrix $\underline{P}$ is a band diagonal matrix such that

$$\underline{P} = \begin{bmatrix} \underline{P}_0 & 0 & & \\ \underline{P}_1 & \underline{P}_0 & & \\ \vdots & \vdots & & \\ \underline{P}_J & \underline{P}_{J-1} & \cdots & \underline{P}_0 \\ 0 & \underline{P}_J & \cdots & \underline{P}_1 \quad \underline{P}_0 \\ & \ddots & & \ddots \quad \ddots \end{bmatrix} \tag{6-90}$$

with

$$\underline{P}_i = (-1)^i \begin{bmatrix} \beta_{iN} & & \\ & \beta_{iN-1} & \\ & & \beta_{iN-N+1} \end{bmatrix} \tag{6-91}$$

Alternatively, the inverse transform of TMT can be similarly obtained from the TDM-to-FDM translator. Figure 6-7 shows the i th channel TDM-to-FDM transmultiplexer via the complex modulation scheme. The output signal of the i th channel is given as

$$\tilde{x}_i(n) = \sum_{m=-\infty}^{\infty} h(n-mN) \frac{1}{N} s_m(i) \exp\{j\frac{2\pi}{N}(i+\frac{1}{2})n\}, \quad i = 0,1,\ldots,N-1. \tag{6-92}$$

Thus, the sum of the total N-channel, $\tilde{x}(n)$, is given by

$$\tilde{x}(n) = \frac{1}{N} \sum_{i=0}^{N-1} \{ \sum_{m=-\infty}^{\infty} h(n-mN) s_m(i) \exp\{ j\frac{2\pi}{N}(i+\frac{1}{2})n\}\} \tag{6-93}$$

Again, by changing the variables, i.e., $n = rN+t$, and proceeding in the similar way as we did for the FDM-to-TDM translation, (6-93) can be written as

$$\tilde{x}_t(m) = \tilde{x}(mN+t)$$

$$= [\bar{p}'_t(m)\exp\{jm\pi\}] \circledast \{ \frac{1}{N} \exp\{j\frac{\pi}{N}\}$$

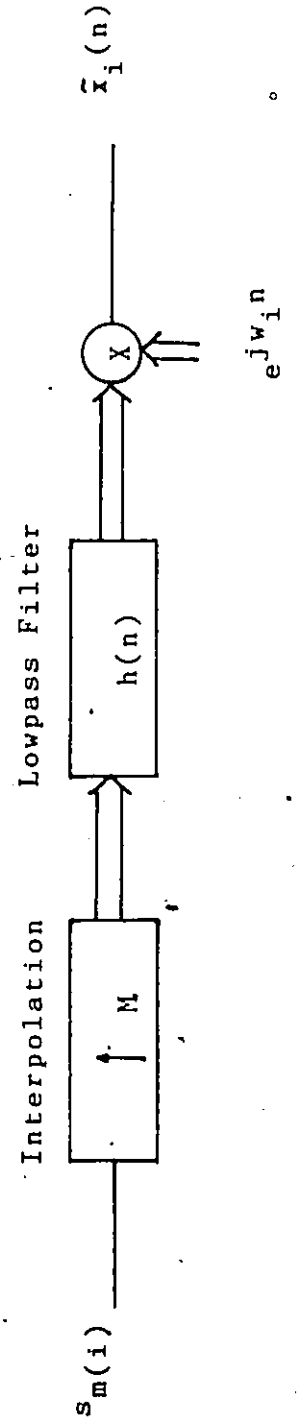


Figure 6.7 The i th Channel TDM-to-FDM Transmultiplexing via the Complex Modulation Scheme.

$$\cdot \sum_{i=0}^{N-1} [s_m(i) \exp\{j\pi m\}] \exp\{j\frac{2\pi it}{N}\} \qquad (6\text{-}94a)$$

where

$$\tilde{p}'_t(m) = h(mN+t) \quad , t=0,1,\ldots,N-1 \qquad (6\text{-}94b)$$

Again, (6-94a) can be realized via the polyphase structure of the complex modulation scheme TDM-to-FDM transmultiplexer and is shown in figure 6-8. The input and output relation of (6-94a) in the matrix form is

$$\underline{\tilde{x}} = \underline{P}' \, \underline{\Phi}' \, \underline{\Gamma}' \, \underline{I}' \underline{s}$$
$$= \underline{Q}'_{TM} \, \underline{s} \qquad (6\text{-}95a)$$

where

$$\underline{Q}'_{TM} = \underline{P}' \, \underline{\Phi}' \underline{\Gamma}' \, \underline{I}' \qquad (6\text{-}95b)$$

Note that, $\underline{Q}'_{TM}$ is referred to as the inverse transform of $\underline{Q}_{TM}$. Also, the vector $\underline{s}$ was defined in (6-84a) and $\underline{\tilde{x}}$ is defined as

$$\underline{\tilde{x}} = [\, \underline{\tilde{r}}_0 \;\; \underline{\tilde{r}}_1 \ldots \underline{\tilde{r}}_i \ldots]^T \qquad (6\text{-}96)$$

with

$$\underline{\tilde{r}}_i = [\, \tilde{x}(iN) \;\; \tilde{x}(iN+1) \ldots \tilde{x}(iN+N-1) \,]^T \qquad (6\text{-}97)$$

Matrices $\underline{\Phi}'$ and $\underline{\Gamma}'$ are the complex conjugate of the matrices $\underline{\Phi}$ and $\underline{\Gamma}$ respectively, i.e.,

$$\underline{\Phi}' = \underline{\Phi}^* \qquad (6\text{-}98)$$

and

$$\underline{\Gamma}' = \underline{\Gamma}^* \qquad (6\text{-}99)$$

Moreover, the band diagonal matrix, $\underline{P}'$ is defined as

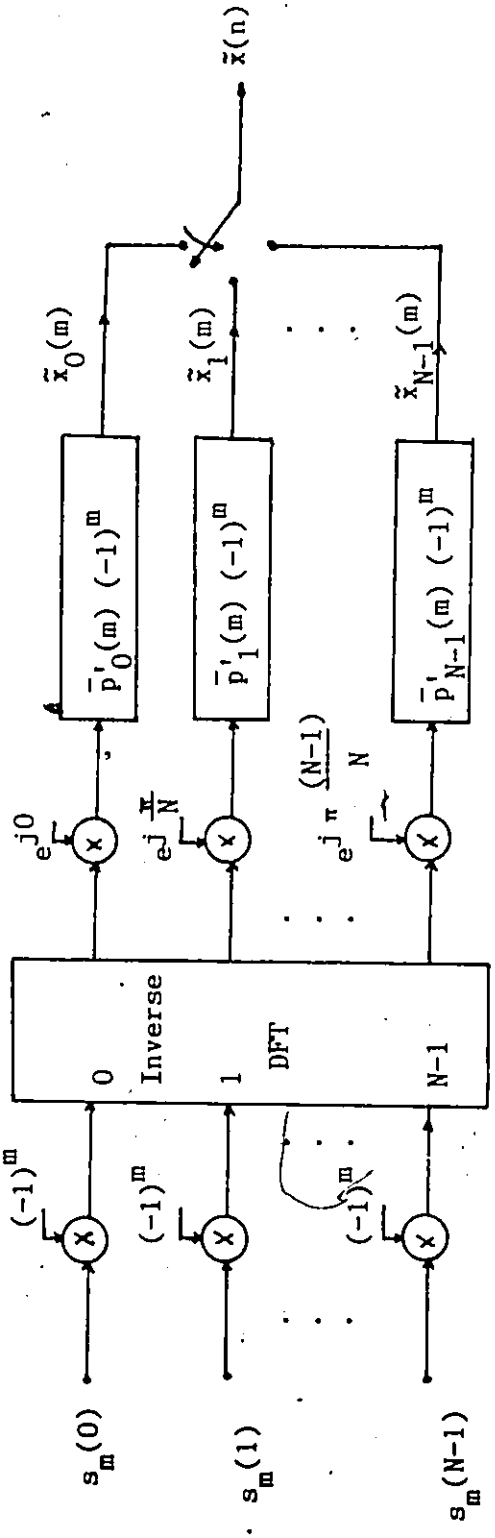


Figure 6.8 The Polyphase Structure of the Complex Modulation Scheme TDM-to-FDM Transmultiplexing.

$$\underline{P}' = \begin{bmatrix} \underline{P}'_0 & 0 & & \\ \underline{P}'_1 & \underline{P}'_0 & & \\ \vdots & \vdots & \ddots & \\ \underline{P}'_J & \underline{P}'_{J-1} & \cdots & \underline{P}'_0 \\ 0 & \underline{P}'_J & \cdots & \underline{P}'_1 \quad \underline{P}'_0 \end{bmatrix} \qquad (6\text{-}100)$$

with

$$\underline{P}'_i = (-1)^i \begin{bmatrix} \beta_{iN} & & \\ & \beta_{iN+1} & \\ & & \beta_{iN+N-1} \end{bmatrix} \qquad (6\text{-}101)$$

6.5.2 The Simplified Form of the Complex Modulation Scheme of the FDM-to-TDM Transmultiplexing Transformtion

As shown in the previous section, that the evaluation of the present transformed signal using the TMT pairs involves all the past signals. This results in a requirement of a large amount of memory to store the original signal and the transformed signals to carry out the transformation in real time implementation. Therefore, the computational effort will be increased accumulatively. Consequently, it is not suitable to be applied directly to the TDLMS and BTDLMS adaptive filtering algorithms. To circumvent this problem,a simplified form of TMT which can be employed in the

adaptive filtering algorithms, is desired. One way to achieve this will be to truncate the lowpass FIR filter coefficients such that it fits the adaptive filter structure. To demonstrate this, we consider a case with N=16 and L=512. Note that, a high order FIR lowpass filter is generally desired to achieve closely to the desired frequency response of the ideal lowpass filter. The frequency response of such FIR lowpass filter is shown in Figure 6.9.

Since the order of the lowpass filter is L (= 512), there is a L/2 (= 256) delay between the input and the output. Furthermore, from computer simulation we found that in order to have less distortion due to the truncation of the lowpass filter at least 2 block coefficients of the lowpass filter (in this case blocks 16 and 17) are required. Therefore, the simplified form of the TMT from (6-83b) will be

$$\underline{Q}_{TM} = \underline{I}'_{2N}\,\underline{\Gamma}_{2N}\,\underline{\Phi}_{2N}\,\underline{P} \qquad (6\text{-}102)$$

where the matrices, $\underline{I}'_{2N}$, $\underline{\Gamma}_{2N}$, $\underline{\Phi}_{2N}$, and $\underline{P}$ are defined as follows:

$$\underline{I}'_{2N} = \begin{bmatrix} \underline{I} & \bigcirc \\ \bigcirc & -\underline{I} \end{bmatrix} \qquad (6\text{-}103a)$$

$$\underline{\Gamma}_{2N} = \begin{bmatrix} \underline{F}' & \bigcirc \\ \bigcirc & \underline{F}' \end{bmatrix} \qquad (6\text{-}103b)$$

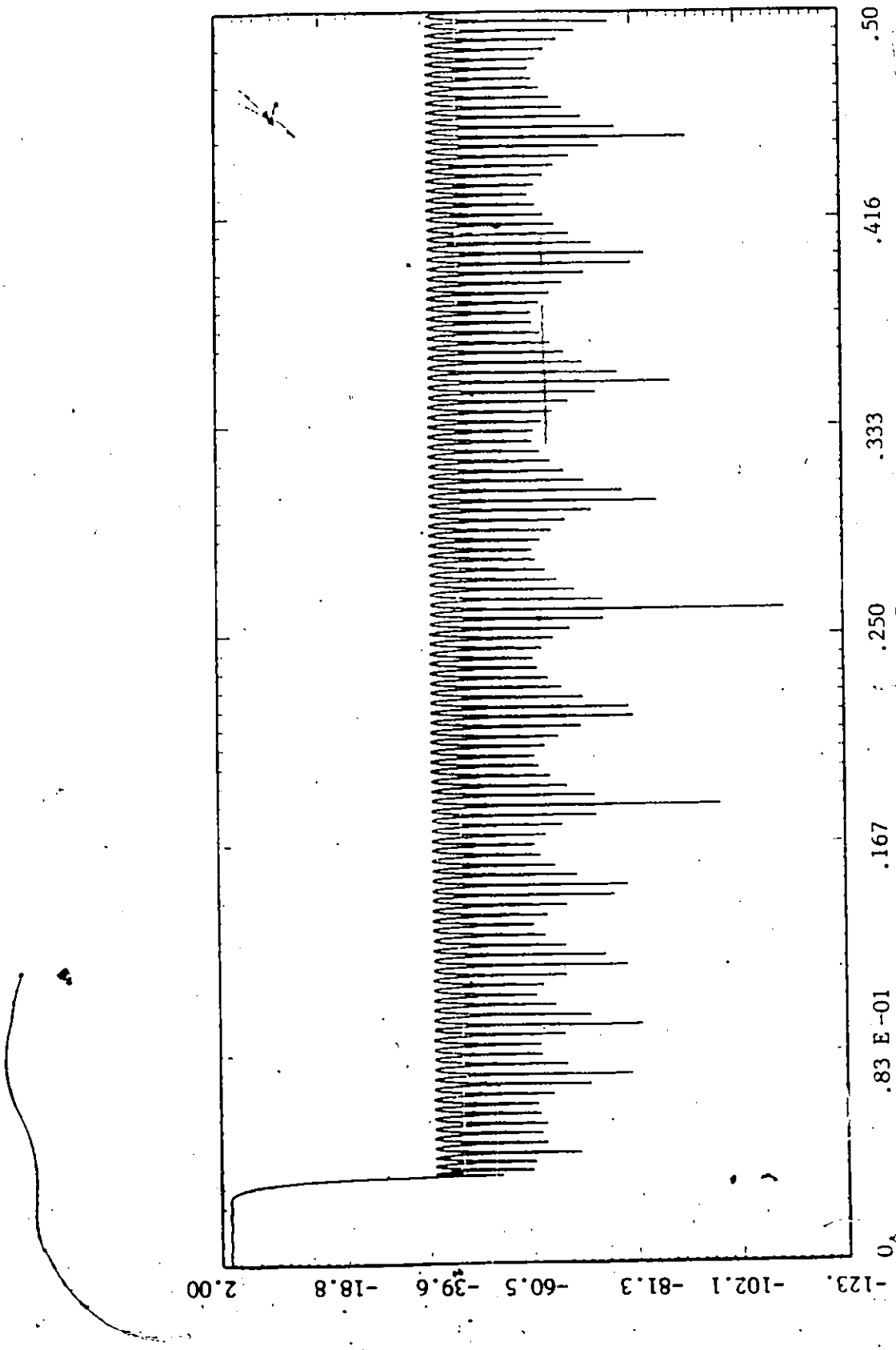


Figure 6.9 The Frequency Response of Equal Ripple FIR Lowpass Filter with Order L = 512.

$$\underline{\Phi}_{2N} = \begin{bmatrix} \underline{\Phi}_0 & \bigcirc \\ \bigcirc & \underline{\Phi}_0 \end{bmatrix} \qquad (6\text{-}103c)$$

and

$$\underline{P} = \begin{bmatrix} \underline{P}_{16} \\ \\ \underline{P}_{17} \end{bmatrix} \qquad (6\text{-}103d)$$

with the NxN square matrices, $\underline{I}$, $\underline{F}'$, $\underline{\Phi}_0$ and $\underline{P}_i$ are defined as before. Similarly, the inverse transform of (6-102) can be simplified as

$$\underline{Q}'_{TM} = \underline{P}' \, \underline{\Phi}'_{2N} \, \underline{\Gamma}'_{2N} \, \underline{I}'_{2N} \qquad (6\text{-}104)$$

where the Nx2N matrix $\underline{P}'$ is defined as

$$\underline{P}' = [\, \underline{P}'_{16} \quad \underline{P}'_{15} \,]_{Nx2N} \qquad (6\text{-}105)$$

and $\underline{P}'$ was defined in (6-101). Similarly, the matrices $\underline{\Phi}'_{2N}$ and $\underline{\Gamma}'_{2N}$ are the complex conjugate of the matrices $\underline{\Phi}_{2N}$ and $\underline{\Gamma}_{2N}$ respectively.

However, the simplified TMT pairs are not square matrices so that the inverse of the matrices has to be defined. From (6-74a), (6-95a), (6-102) and (6-104), if completely transformed pairs are used, ideally we will have

$$\underline{Q}'_{TM} \, \underline{Q}_{TM} = \underline{I} \qquad (6\text{-}106)$$

where $\underline{I}$ is an NxN identity matrix. Consequently, the left inverse matrix of $\underline{Q}_{TM}$ is defined as

$$\underline{Q}^{-1}_{TM} = \underline{Q}'_{TM} \qquad (6\text{-}107)$$

Note that (6-106) is true only when the coefficients of the lowpass filter satisfy the following identity equations:

$$\beta_{16N+i}\,\beta_{16N-i} + \beta_{15N+i}\,\beta_{17N-i} = 1/N$$

$$i = 0, 1, \ldots, N-1$$

Now, based on these notations, we can apply the simplified form of TMT to the transformed domain adaptive filtering algorithms.

6.5.3 The TDLMS Adaptive Filter with TMT

In this case, with N = 16, the time domain weight vector $\underline{w}(k)$ can be represented as

$$\underline{w}(k) = \underline{Q}^T_{TM}\,\underline{b}(k) \qquad (6\text{-}108a)$$

where $\underline{w}(k)$ is a Nx1 column vector and $\underline{b}(k)$ is a 2Nx1 column vector. Equivalently, $\underline{b}(k)$ can be expressed as

$$\underline{b}(k) = (\underline{Q}^T_{TM})^{-1}\,\underline{w}(k) \qquad (6\text{-}108b)$$

Correspondingly, the transformed input vector is defined as

$$\underline{s}(k) = \underline{Q}_{TM}\,\underline{x}(k), \qquad (6\text{-}109)$$

and is a 2Nx1 column vector. The output signal, y(k) at k th sampling instant can be written as

$$\begin{aligned} y(k) &= \underline{w}^T(k)\underline{x}(k) \\ &= \underline{b}^T(k)\,\underline{Q}_{TM}\underline{x}(k) \\ &= \underline{b}^T(k)\,\underline{s}(k) \qquad (6\text{-}110) \end{aligned}$$

Following similar approach as before, we have the weight vector up-dated equation of TDLMS adaptive algorithm with simplified TMT:

$$\underline{b}(k+1) = \underline{b}(k) + 2\,\mu\,e(k)\,\underline{s}^*(k) \qquad (6\text{-}111)$$

We can see that (6-111) is very similar to the TDLMS adaptive filtering algorithm discussed in the previous sections except

that the length of $\underline{b}(k)$ is 2N instead of N. Furthermore, the time domain expression of (6-111) can be obtained by pre-multiplying $\underline{Q}^T_{TM}$ in both sides of (6-111):

$$\begin{aligned}\underline{w}(k+1) &= \underline{Q}^T_{TM}\,\underline{b}(k+1)\\ &= \underline{w}(k) + 2\,\mu\, e(k)\,\underline{Q}^T_{TM}\,\underline{s}^*(k)\\ &= \underline{w}(k) + 2\mu\, e(k)\,\underline{Q}^T_{TM}\,\underline{Q}^*_{TM}\,\underline{x}^*(k)\end{aligned} \qquad (6\text{-}112)$$

We recognize that (6-112) is not equivalent to the time domain LMS algorithm. This is because

$$\underline{Q}^T_{TM}\,\underline{Q}^*_{TM} \neq \underline{I} \qquad (6\text{-}113)$$

Alternatively, the TDLMS adaptive algorithm with normalized simplified TMT can be expressed as

$$\underline{b}(k+1) = \underline{b}(k) + 2\,\mu_i\, e(k)\,\underline{s}^*(k)$$

with

$$\mu_i = \frac{\mu}{E[|s_k(i)|^2]}, \quad i=0,1,\ldots,2N-1. \qquad (6\text{-}114)$$

Similarly, the time domain expression of (6-114) is given as

$$\begin{aligned}\underline{w}(k+1) &= \underline{Q}^T_{TM}\,\underline{b}(k+1)\\ &= \underline{w}(k) + 2\,\mu\,\underline{Q}^T_{TM}\,\underline{R}^{-1}\,\underline{Q}^*_{TM}\,\underline{x}^*(k)\\ &= \underline{w}(k) + 2\mu\,\underline{R}^{-1}_{TM}\,\underline{x}^*(k)\end{aligned} \qquad (6\text{-}115)$$

where

$$\underline{R}^{-1}_{TM} = \underline{Q}^T_{TM}\,\underline{R}^{-1}\,\underline{Q}^*_{TM} \qquad (6\text{-}116)$$

Since the simplfied TMT can not diagonalize the input auto-correlation matrix, (6-115) is not equivalent to the normalized K-L TDLMS adaptive algorithm. This is due to the fact that we used the finite order lowpass filter to

approximate the ideal lowpass filter in the real time application.

6.5.4 The BTDLMS Adaptive Algorithm with TMT

The simplified form of TMT can also be applied to the BTDLMS adaptive algorithm. To do that, first we define the transformed desired vector and the transformed input vector as

$$\underline{\hat{d}}(m) = Q_{TM}\, \underline{d}(m) \qquad (6\text{-}117)$$

and

$$\underline{\hat{s}}(m) = \underline{Q}_{TM}\, \underline{\hat{x}}(m) \qquad (6\text{-}118)$$

respectively. It may be noted that both transformed vectors are with length 2N. Also, in the BTDLMS adaptive algorithm the weight vector is defined as

$$\underline{w}(m) = \underline{Q}^{T}_{TM}\, \underline{b}(m) \qquad (6\text{-}119)$$

Here the integer m is the block index. Consequently, we have the BTDLMS adaptive algorithm with simplified TMT as follows:

$$b_{m+1}(i) = b_m(i) + 2\mu_B\, v_m(i)\, \hat{s}^{*}_{m}(i)$$
$$i = 0,1, \ldots, 2N-1 \qquad (6\text{-}120)$$

Similarly, the weight update equation of the BTDLMS with normalized simplified TMT can be written as

$$b_{m+1}(i) = b_m(i) + 2\mu_i\, v_m(i)\, \hat{s}^{*}_{m}(i)$$

with

$$\mu_i = \frac{\mu_B}{E[|\hat{s}_m(i)|^2]}, \qquad i=0,1,\ldots, 2N-1. \qquad (6\text{-}121)$$

Since the general orthogonal basis function of the TMT can

not be explicitly expressed, we can not explicitly obtain the time domain expressions for both algorithms, (6-120) and (6-121). However, in the real time implementation, both the equivalent time domain weight vector and error signals can be obtained by computer simulation.

6.6 Transformed Domain Adaptive Line Enhancer (ALE)

In this section, the performance, in terms of MSE, of various suboptimal orthogonal transformed domain adaptive algorithms is investigated via computer simulations. These results are then compared to the time domain LMS algorithm in the ALE. The received signal of ALE is given by

$$d(k) = A_1 \cos\left(\frac{\pi k}{5.41}\right) + A_2 \cos\left(\frac{5\pi k}{16}\right) + n(k)$$

where A_1 and A_2 are the amplitudes of the cosine signals and n(k), the noise signal, is the additive white Gaussian, zero-mean process. The SNR is defined as the power ratio between the square value of A_2 and n(k) in dB. Also, for convenience, we assumed A_1 is less than A_2 and both are constant values. The curves of the simulation results are the MSE vs the adaptation number, k.

Again, here we only consider the high SNR case. This is because, in such case we are able to investigate the effect due to the power ratio between A_1 and A_2. To do so, a signal model with the parameters N=16, μ=0.01 and SNR=40

and 60 dB is considered From figures 6-10 ($A_1 = A_2$) and 6-11 ($A_1 = 0.1\ A_2$), we observed that the convergence rate of the LMS algorithm is highly dependent on the ratio between A_1 and A_2. This agrees with the results of chapter 5 for the complex time domain LMS algorithm. However, this is not the case of the TDLMS ALE with normalized Lattice transform which are shown in figures (6-12) and (6-13). Furthermore, figures (6-14), (6-15), (6-16) and (6-17) are the curves with the same parameters as figure 6-11 of various suboptimal orthogonal transformed domain adaptive algorithms. From these curves, we have the same conclusions as we made in chapter 5. That is, the normalized transformed domain adaptive algorithm can be used to improve the convergence rate when the time domain LMS algorithm does not perform well which is the case when the eigenvalue spread as well as the power ratio of A_1 and A_2 are large. Other curves are also plotted which have the same conclusion as observed above.

Finally, since the time domain LMS algorithm, in general, converges rapidly on the first few iterations, therefore, a hybrid algorithm which combines the time domain LMS algorithm and the normalized transformed domain adaptive algorithm can be used. This may result in having less computational effort and at the same time has good convergent property compared to the normalized transformed domain LMS adaptive algorithm.

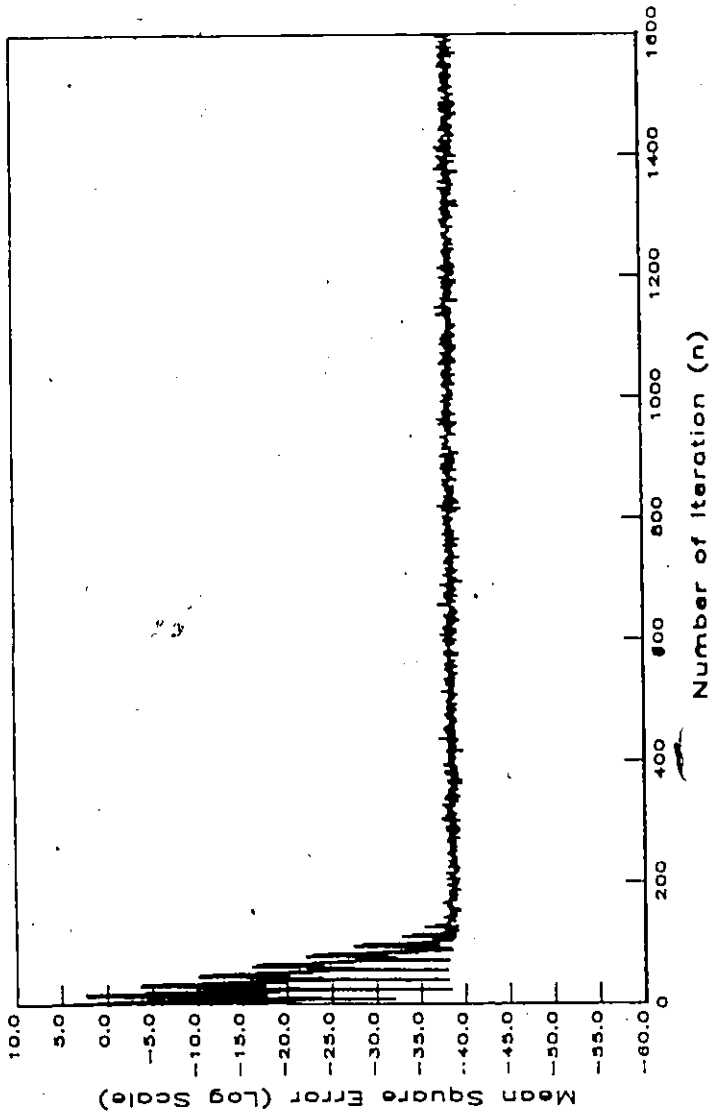


Figure 6. 10 The Performance of Conventional LMS ALE with N=16, SNR = 40 dB, $A_1 = A_2 = 1$, $\mu = 0.01$.

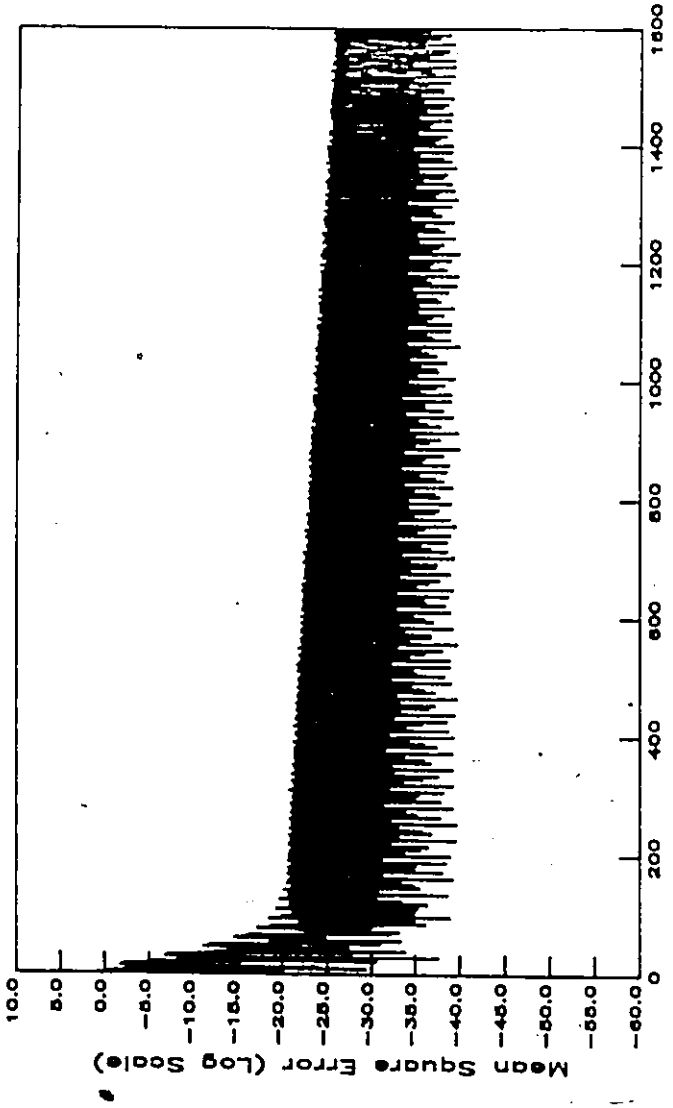


Figure 6.11 The Performance of Conventional LMS ALE with N=16, SMR = 40 dB, A_1 = 0.1, A_2 =1, and μ = 0.01.

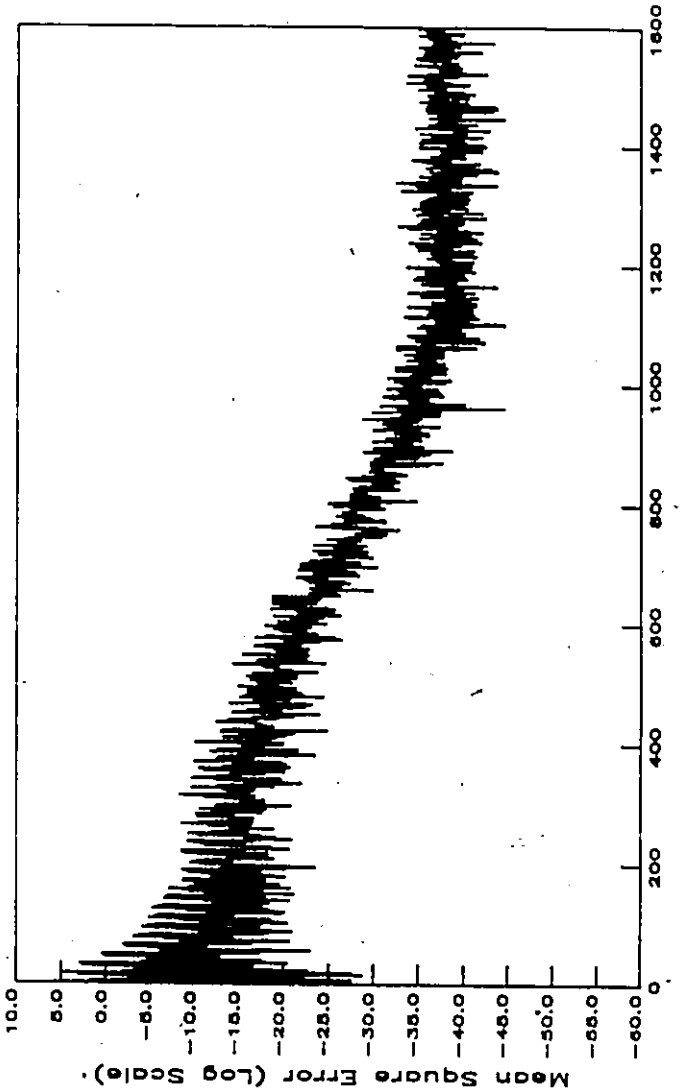


Figure 6.12 The Performance of the TDLMS ALE with Normalized Lattice Transform and N=16, SNR = 40 dB, $A_1 = A_2 = 1$ and $\mu = 0.01$.

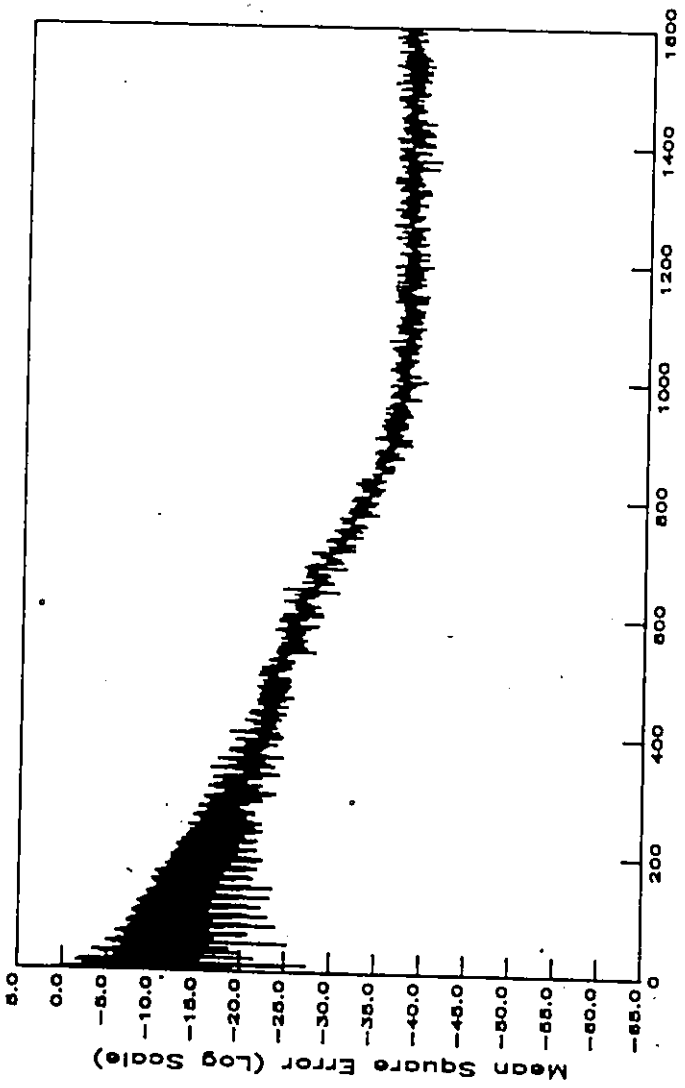


Figure 6. 13 The Performance of the TDLMS ALE with Normalized Lattice Transform and N=16 A_1=0.1, A_2=1, SNR=40 dB and μ =0.01.

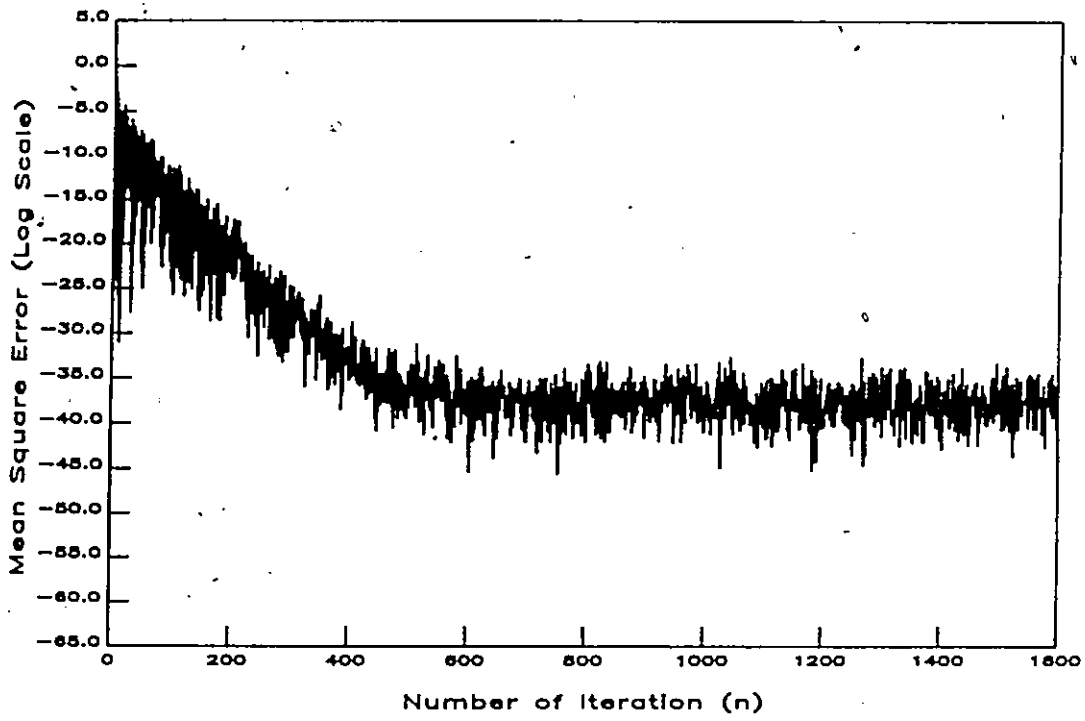


Figure 6. 14 The Performance of TDLMS ALE with Normalized DCT and N=16, A_1=0.1, A_2=1, SNR=40 dB, and μ = 0.01.

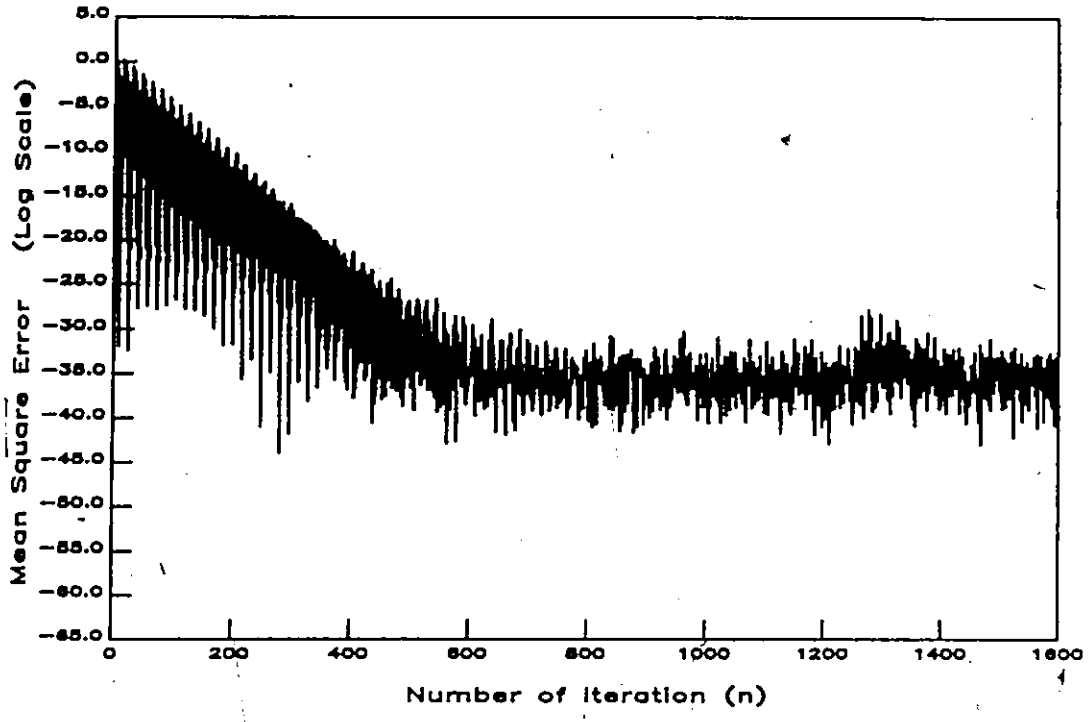


Figure 6.15 The Performance of BTDLMS ALE with Normalized FFT and N=16, SNR=40 dB, A_1=0.1, A_2=1, and μ=0.1.

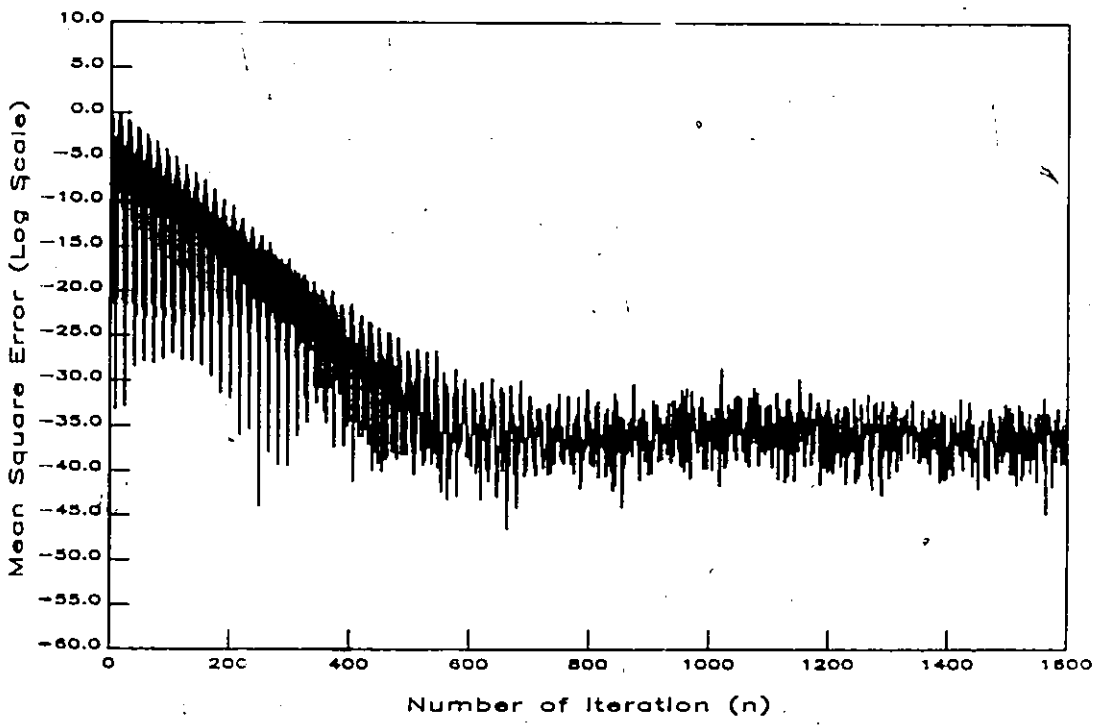


Figure 6. 16 The Performance of BTDLMS ALE with Normalized TMT (Simplified form) and N=16, A_1=0.1, A_2=1, SNR=40dB and μ=0.1.

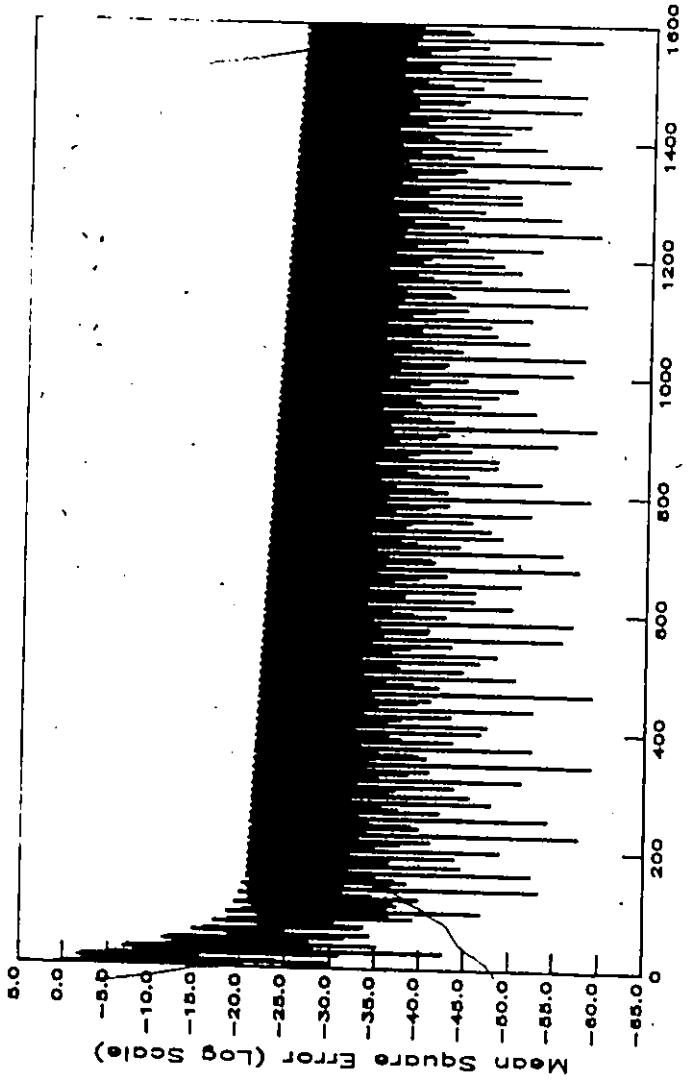


Figure 6.17 The Performance of Conventional LMS ALE with N=16, SNR = 60 dB, A_1=0.1, A_2= 1, and μ = 0.01.

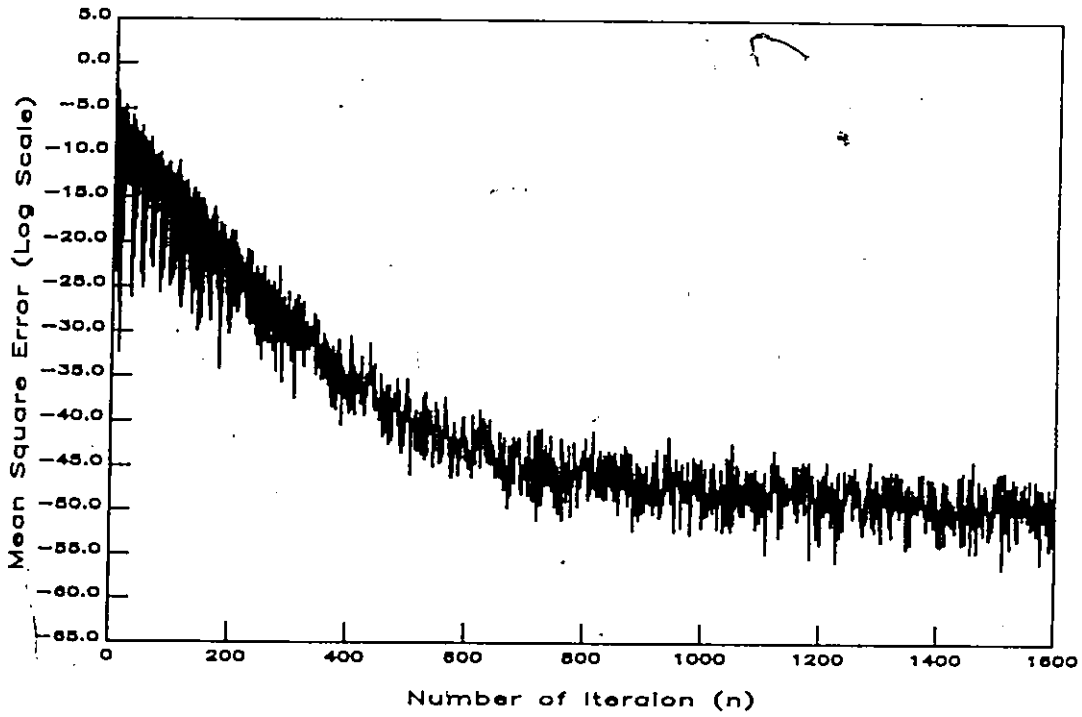


Figure 6. 18 The Performance of TDLMS ALE with Normalized DCT and N=16, A_1=0.1, A_2=1, SNR =60 dB and μ=0.01.

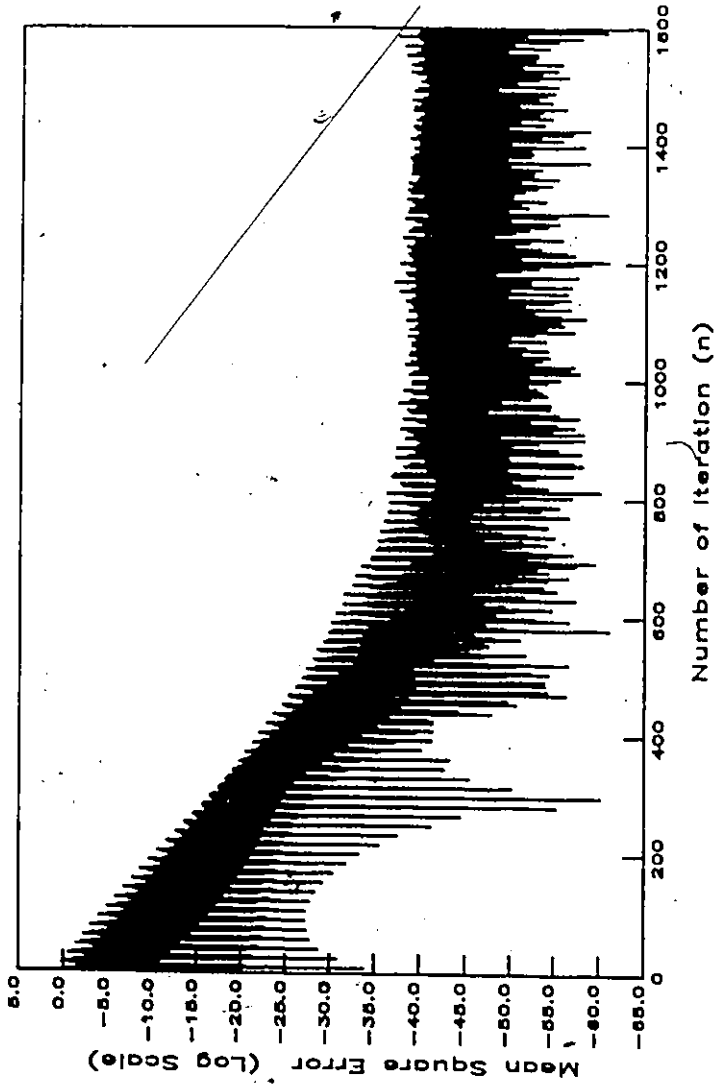


Figure 6. 19 The Performance of BTDLMS ALE with Normalized FFT and N=16, SNR=60dB, A_1=0.1, A_2=1 and μ=0.1.

6.7 Conclusions

We have applied various suboptimal orthogonal transforms to the TDLMS and BTDLMS adaptive algorithms. The effect due to the incomplete diagonalization in the transformed domain adaptive filtering was examined via the DCT. We showed that the convergence rate of the TDLMS adaptive algorithm with the normalized DCT is slower than the TDLMS adaptive algorithm with K-L transform. This is true even that the eigenvalues of the input autocorrelation matrix can be exactly estimated.

To reduce the computational effort in the TDLMS adaptive algorithm with the DCT, we have derived a recursive equation to obtain the output of DCT. This results in simpler implementation and less computation compared to the fast DCT, especially, when the order of the adaptive filter is very large.

Moreover, a simplified form of the TMT was obtained with N=16. This enables us to apply the TMT to the transformed domain adaptive algorithms addressed in this thesis. It can be seen that the size of the memory used for storing the input and transformed signals can be significantly reduced, also, the computations are reduced remarkably.

Finally, some computer simulation results are presented for the application of ALE. In which the signal model used is two real sinusoidal signals buried in white Gaussian noise

with zero-mean. These results agree quite well with the results of chapter 5 for the complex ALE.

CHAPTER 7

RE-CIRCULATION OF INPUT DATA IN FREQUENCY DOMAIN ADAPTIVE FILTERING

7.1 Introduction

In the preceeding chapter we showed that the Dentino-type frequency-domain adaptive filter is a special type of the BTDLMS adaptive filter in which the discrete transform is chosen to be a discrete Fourier transform (DFT). The BTDLMS adaptive filter can be implemented with reduced computational effort by evaluating DFT coefficients using computationally efficient FFT algorithm. Furthermore, since the individual tap weight is updated separately, under certain conditions, both the mean and the variance of the tap weights of the frequency domain adaptive filter can be obtained relatively simply allowing the statistical analysis to be performed and can often be used to predict the performance of time domain adaptive filtering [28].

In this chapter, the frequency-domain adaptive filtering algorithm is analyzed when the data is passed through the adaptive algorithm several times. It is referred here as the re-circulation of the input data in frequency-domain adaptive algorithm.

The Dentino-type adaptive filter can be employed in many cases in which the time domain LMS adaptive filter is

applicable. However, in this chapter, we are concerned with the special case of applying it to sonar signal processing in which the number of input signal samples available is often limited. Under such circumstances,the adaptive filter may not have sufficient supply of signal samples to converge to the value of minimum MSE, and therefore may not operate in its optimum state. In order to circumvent this, an algorithm is proposed in next section such that the limited supply of signal samples is re-circulated to the adaptive filter so that further adjustment of tap weights can be carried out. An analysis of the effects of the signal re-circulation is first presented and is then verified by computer simulation.

7.2 Re-circulation of Input Data in Frequency Domain Adaptive Filtering

The idea of recirculation data algorithm has been studied by many authors [26,32,33,79]. The idea of re-circulating the data algorithm addressed here is similar to the approach discussed in [26,32,33,79],where the on-line algorithm or the adaptive algorithm is applied with off-line or the batch data to the system identification problem. The advantages for using such an adaptive algorithm for batch data are:

(1) If one already has recursive software available, it may be more expensive to develop batch software than use the available software.

(2) The minimum of the cost function can sometimes be

found faster with recursive algorithm, because if the recursive estimates from one pass are close to the minimum, then the initial conditions for the next pass (being close to the minimum), will essentially bring the estimate to the minimum.

(3) The recursive algorithms can be used to detect non-stationaries in the data and the system properties.

To facilitate our analysis, we assume that the frequency domain adaptive filtering is employed in an environment such that the desired signal vector $\underline{d}(m)$ and the input signal vector $\underline{\hat{x}}(m)$ contain a signal vector $\underline{s}_d(m)$ buried in statistically independent white noise with zero-mean. Thus, the kth frequency component of $\underline{d}(m)$ and $\underline{\hat{x}}(m)$ are given by

$$\hat{d}_m(k) = a(k)\, z_m(k) + n_{m,1}(k)$$
$$\hat{s}_m(k) = z_m(k) + n_{m,2}(k) \qquad (7\text{-}1)$$

where $a(k)$ is a complex coefficient. This situation arises very often in signal processing especially in problems concerning noise cancellation and time delay estimation. Furthermore, we assume that the frequency components of both sequences $\underline{d}(m)$ and $\underline{\hat{x}}(m)$ are zero mean, white complex circular Gaussian processes [31]. This means that the signal and noise in the kth frequency components are related by the following equations

$$E[z_m(k)\, z^*_n(k)] = \sigma^2_{sk}\, \delta_{mn} \quad ,k=0,1, \ldots,N-1$$
$$E[z_m(k)\, z_n(k)] = 0, \text{ for all m and n.} \qquad (7\text{-}2)$$
$$E[n_{m,1}(k) n^*_{n,2}(k)] = \sigma^2_{nk}\, \delta_{mn} \quad ,i=1,2$$
$$E[n_{m,1}(k) n_{n,2}(k)] = 0 \qquad (7\text{-}3)$$

where

σ^2_{sk}: is the power in the kth bin of the FFT of the signal

σ^2_{nk}: is the noise power in every frequency component

The assumptions of the signal and noise properties in (7-2) and (7-3) are valid in broadband noise cancellation and in time-delay estimation when the number of time-delay samples is small compared to the order of the FFT employed [28].

The configuration of a frequency-domain adaptive filter having the ability of re-circulation of input data is shown in figure 7.2, and is, in essence similar to a simple frequency-domain adaptive filter. The additional features are the delay elements which carry out the function of storing the signals $\hat{d}_m(k)$ and $\hat{s}_m(k)$. These delays must be at least of length MN samples so as to allow all the input signal samples to be processed first before the next re-circulation begins. With this extra facility, the notations for the output signal vector, the error vector, and the tap weights of the adaptive filter have to be slightly modified. We let

$b_k(p,m)$: be the kth complex tap weight for the mth signal block

$\hat{y}_k(p,m)$: be the kth output sample for the mth N-point signal block

$v_k(p,m)$: be the kth error sample for the mth N-point signal block

where p denotes the number of times the input data have been

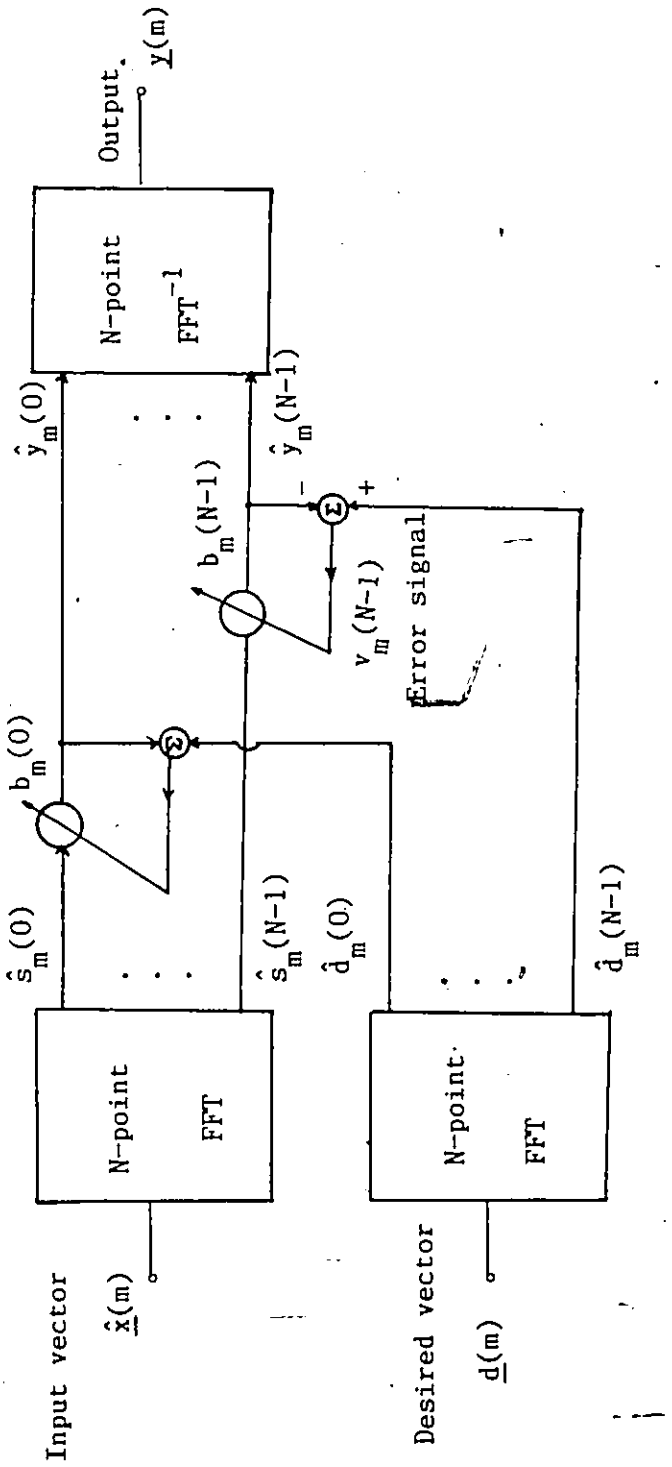


Figure 7.1 An Adaptive Filter in Frequency Domain.

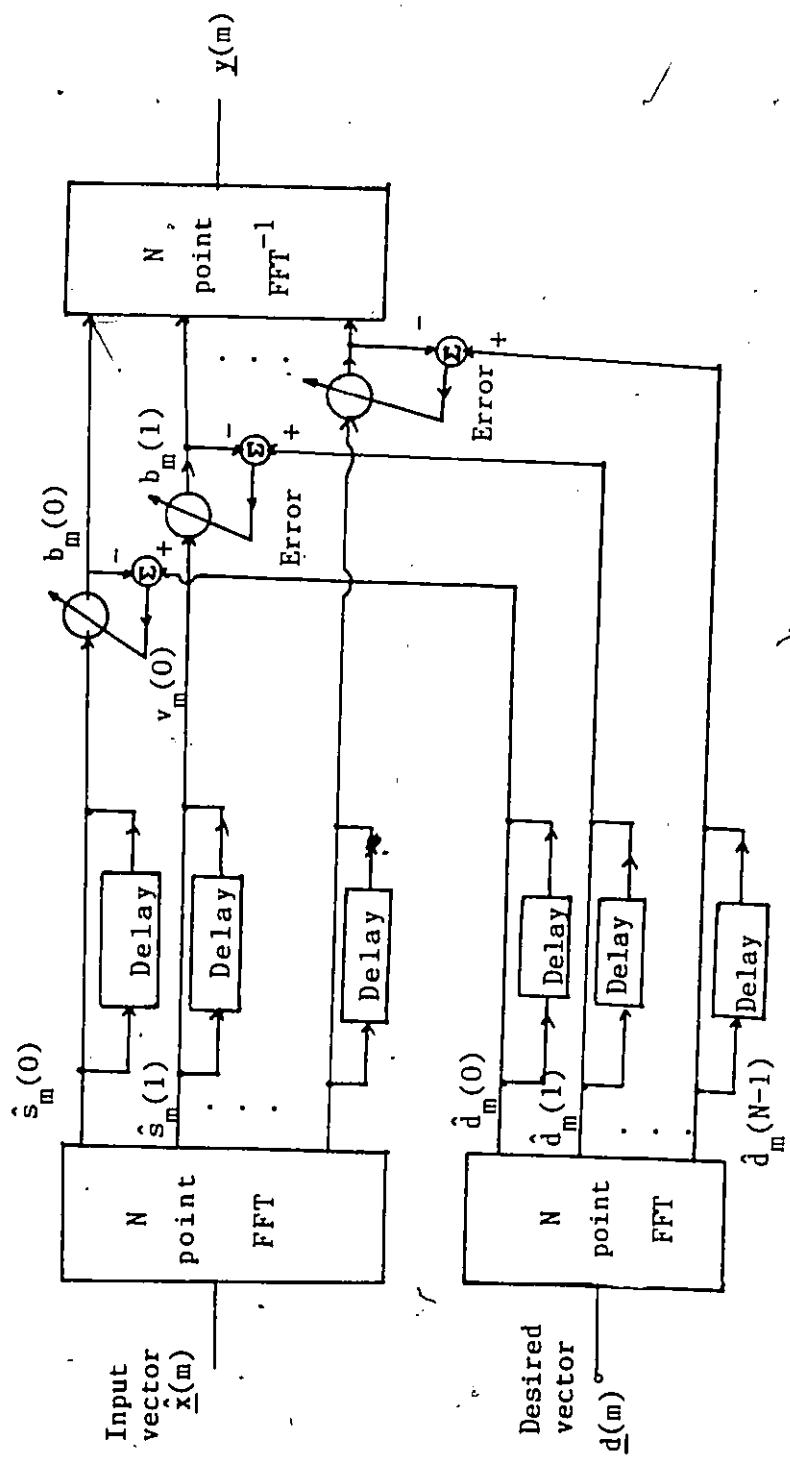


Figure 7.2 A Frequency-domain Adaptive Filter with Input Re-circulation Facility.

re-circulated. Thus, for a simple frequency-domain adaptive filter with no re-circulation of input data, p=0. The procedure of steepest descent is again used for updating the tap weights, i.e.

$$b_k(p,m+1) = b_k(p,m) + \mu_r\, v_k(p,m)\hat{s}^*_m(k) \qquad m=0,1,..M-1 \qquad (7\text{-}4)$$

Note that

$$b_k(p+1,0) = b_k(p,M) \qquad (7\text{-}5)$$

Also, since $\hat{s}_m(k)$ and $\hat{d}_m(k)$ are the re-circulated signals, they are independent of p. Since

$$v_k(p,m) = \hat{d}_m(k) - \hat{y}_k(p,m)$$
$$= \hat{d}_m(k) - b_k(p,m)\hat{s}_m(k) \qquad (7\text{-}6)$$

(7-4) can be rewritten as

$$b_k(p,m+1) = b_k(p,m)[1-\mu_r|\hat{s}_m(k)|^2] + \mu_r \hat{d}_m(k)\hat{s}^*_m(k) \qquad (7\text{-}7)$$

which is a first order difference equation the solution of which can be written as

$$b_k(p,m+1) = b_k(p,0)\prod_{i=0}^{m}[1-\mu_r|\hat{s}_i(k)|^2] + \mu_r\sum_{j=0}^{m}\hat{d}_j(k)\hat{s}_j(k)\prod_{\ell=j+1}^{m}[1-\mu_r|\hat{s}_\ell(k)|^2]$$
$$= b_k(p,0)\,A_k(m) + B_k(m) \qquad (7\text{-}8)$$

where

$$A_k(m) = \prod_{i=0}^{m}[1-\mu_r|\hat{s}_i(k)|^2] \qquad (7\text{-}9a)$$

and

$$B_k(m) = \mu_r\sum_{j=0}^{m}\hat{d}_j(k)\hat{s}^*_j(k)\prod_{\ell=j+1}^{m}[1-\mu_r|\hat{s}_\ell(k)|^2] \qquad (7\text{-}9b)$$

Now, if we start the tap weights at zero value, i.e., $b_k(0,0) = 0$, then at the begining of the first re-circulation we have from (7-5) and (7-8)

$$b_k(1,0) = b_k(0,M)$$
$$= B_k(M-1)$$

and at the end of the first re-circulation, we have

$$b_k(1,M) = B_k(M-1)\, A_k(M-1) + B_k(M-1)$$
$$= B_k(M-1)[\,1 + A_k(M-1)] \qquad (7\text{-}10)$$

Again, using (7-5) and (7-8) and continuting the recursive relationship, we have, at the end of the pth re-circulation of input data,

$$b_k(p,M) = \sum_{n=0}^{p} B_k(M-1)\, A_k^{\,n}(M-1) \qquad (7\text{-}11)$$

Since $\hat{d}_m(k)$ and $\hat{s}_m(k)$ are random variables, $A_k(M-1)$ and $B_k(M-1)$ are also random variables. However, they are correlated. In order to compute for the mean and mean-square values of the tap weights, it is necessary to separate $A_k(M-1)$ and $B_k(M-1)$ into their correlated and uncorrelated parts. From (7-9a) and (7-9b), we get

$$B_k(M-1)\, A^n_k(M-1)$$

$$= \{\mu_r \sum_{j=0}^{M-1} \hat{d}_j(k)\hat{s}_j^{\,*}(k) \prod_{\ell=j+1}^{M-1} [\,1-\mu_r|\hat{s}_\ell(k)|^2]\}\{\prod_{i=0}^{j-1}[1-\mu_r|\hat{s}_i(k)|^2]\}$$

$$= \sum_{j=0}^{M-1} \mu_r \hat{d}_j(k)\hat{s}_j(k)[1-\mu_r|\hat{s}_j(k)|^2]^n \prod_{i=0}^{j-1}[1-\mu_r|\hat{s}_i(k)|^2]^n$$

$$\prod_{\ell=j+1}^{M-1}[1-\mu_r|\hat{s}_\ell(k)|^2]^{n+1}$$

$$= \sum_{j=0}^{M-1} \{\Theta_k(j,n) \prod_{i=0}^{j-1} \Phi_k(i,n) \prod_{\ell=j+1}^{M-1} \Psi_k(\ell,n)\} \qquad (7\text{-}12)$$

where

$$\Theta_k(j,n) = \mu_r \hat{d}_j(k)\, \hat{s}^*_j(k)[1-\mu_r|\hat{s}_j(k)|^2]^n \qquad (7\text{-}13a)$$

$$\Phi_k(i,n) = [\,1-\mu_r|\hat{s}_i(k)|^2]^n \qquad (7\text{-}13b)$$

and

$$\Psi_k(\ell,n) = [\,1-\mu_r|\hat{s}_\ell(k)|^2]^{n+1}] \qquad (7\text{-}13c)$$

Now, $\Theta_k(j,n)$, $\Phi_k(i,n)$ and $\Psi_k(\ell,n)$ are uncorrelated since they contain input signals from different blocks. The mean and mean square-value of the tap weights can now be evaluated using (7-12) and (7-13). These results can be used to further express the mean square error, $E[|v_k(p,m)|^2]$, in closed form.

7.3 Mean Weight Value of the Re-circulation Algorithm

The mean value of tap weights can be evaluated by taking the expectation for both sides of (7-11), that is,

$$E[b_k(p,M)] = \sum_{n=0}^{p} E[\,B_k(M-1)\,A_k^n(M-1)]$$

$$= \sum_{n=0}^{p}\{\sum_{j=0}^{M-1} E[\Theta_k(j,n)] \prod_{i=0}^{j-1} E[\Phi_k(i,n)] \prod_{\ell=j+1}^{M-1} E[\Psi_k(\ell,n)]\} \qquad (7\text{-}14)$$

It can be shown (Appendix H) that

$$E[\Theta_k(j,n)] \simeq \mu_r a(k)\,\sigma^2_{sk}\{\,1-2n\mu_r(\sigma^2_{sk}+\sigma^2_{nk})\} \qquad (7\text{-}15a)$$

$$E[\Phi_k(i,n)] \simeq 1-n\,\mu_r(\sigma^2_{sk}+\sigma^2_{nk}) \qquad (7\text{-}15b)$$

and

$$E[\psi_k(\ell,n)] = 1-(n+1)\mu_r(\sigma^2_{sk}+\sigma^2_{nk}) \qquad (7\text{-}15c)$$

From (7-15), we see that the value of $E[b_k(p,M)]$ is a function of $\mu_r(\sigma^2_{sk} + \sigma^2_{nk})$. To simplify the notations, we define the normalized step size $\tilde{\mu}_k$ as

$$\tilde{\mu}_k = \mu_r(\sigma^2_{sk} + \sigma^2_{nk}) \qquad (7\text{-}16)$$

and is generally chosen such that $\tilde{\mu}_k << 1$. Using this condition and substituting (7-15) into (7-14), the mean of the kth tap weight can be expressed as (Appendix H)

$$E[b_k(p,M)] \simeq \frac{a(k)\sigma^2_{sk}}{(\sigma^2_{sk}+\sigma^2_{nk})}[1-\{1-(p+1)\tilde{\mu}_k\}^{M+2}] \qquad (7\text{-}17)$$

(7-17) gives the mean value of the tap weight. It is clear that if there is a large number of data blocks, i.e. $M \to \infty$, then the final mean value of the kth tap weight is given by

$$b_{k\,\infty} = \lim_{M\to\infty} E[b_k(p,M)]$$

$$= \frac{a(k)\,\sigma^2_{sk}}{(\sigma^2_{sk}+\sigma^2_{nk})} \qquad (7\text{-}18)$$

which is reminiscent of the optimum Wiener solution. Thus, if we have a large number of data blocks, there is no necessity to re-circulate the input data. However, if the number of data blocks is limited, it can be deduced from (7-17) that by re-circulating the input, i.e. $p>0$, the mean value of tap weight will be closer to the final optimum value.

7.4 Weight Covariance Value of Re-circulating Algorithm

For further evaluating the MSE, we need to derive weight covariance, which can be expressed in terms of mean weight values and the mean square value of tap weights. Since the mean weight values has been obtained in (7-17), so that we need only to derive the mean square values of the tap weights. From (7-11) the mean square value of tap weight can be expressed as

$$|\, b_k(p,M)\,|^2 = \Big|\sum_{n=0}^{p} B_k(M-1)\; A^n{}_k(M-1)\Big|^2$$

$$= |B_k(M-1)|^2 \Big|\sum_{n=0}^{p} A^n{}_k(M-1)\Big|^2 \qquad (7\text{-}19)$$

Now, from (7-9b)

$$|B_k(M-1)|^2 = \mu_r^2 \sum_{i,j=0}^{M-1} \sum [\hat{d}_i(k)\hat{d}^*_j(k)\hat{s}_i(k)\hat{s}^*{}_j(k) \prod_{\ell=i+1}^{M-1} \prod_{m=j+1}^{M-1}$$

$$\{1-\mu_r|\hat{s}_\ell(k)|^2\}\{1-\mu_r|\hat{s}_m(k)|^2\}]$$

$$= \Gamma_{1k}(M-1) + \Gamma_{2k}(M-1) + \Gamma_{3k}(M-1) \qquad (7\text{-}20)$$

where

$$\Gamma_{1k}(M-1) = \mu_r^2 \sum_{j=0}^{M-1} \{|\hat{d}_j(k)|^2 |\hat{s}_j(k)|^2 \prod_{\ell=j+1}^{M-1} \{1-\mu_r|\hat{s}_\ell(k)|^2\}^2] \qquad (7\text{-}21)$$

$$\Gamma_{2k}(M-1) = \mu_r^2 \sum_{i=0}^{M-1} \sum_{j=0}^{i-1} [\hat{d}_i(k)\hat{d}^*{}_j(k)\hat{s}_i(k)\hat{s}^*{}_j(k) \prod_{\ell=i+1}^{M-1} \prod_{m=j+1}^{M-1}$$

$$\{1-\mu_r|\hat{s}_\ell(k)|^2\}\{1-\mu_r|\hat{s}_m(k)|^2\}] \qquad (7\text{-}22)$$

and

$$\Gamma_{3k}(M-1) = \mu_r^2 \sum_{i=0}^{M-1}\sum_{j=0}^{i-1} [\hat{d}_i(k)\hat{d}^*_j(k)\hat{s}_i(k)\hat{s}^*_j(k)\prod_{\ell=i+1}^{M-1}\prod_{m=j+1}^{M-1}$$

$$\{1-\mu_r|\hat{s}_\ell(k)|^2\}\{1-\mu_r|\hat{s}_m(k)|^2\}] \qquad (7-23)$$

Substituting (7-20) in (7-19), we have

$$E[|b_k(p,M)|^2] = E[\{\Gamma_{1k}(M-1) + \Gamma_{2k}(M-1) + \Gamma_{3k}(M-1)\} \cdot |\sum_{n=0}^{p} A_k{}^n(M-1)|^2]$$

$$= E[C_{1k}(p,M-1)] + E[C_{2k}(p,M-1)] + E[C_{3k}(p,M-1)] \qquad (7-24)$$

where

$$E[C_{ik}(p,M-1)] = E[\Gamma_{ik}(M-1)\,|\sum_{n=0}^{p} A_k{}^n(M-1)|^2] \quad \text{for } i=1,2 \text{ and } 3. \qquad (7-25)$$

It can be shown (Appendix H) that

$$E[C_{1k}(p,M-1)] \tilde{-} \frac{1}{2}\tilde{\mu}_r\{|b_{k\infty}|^2 + (\frac{|a(k)|^2}{a(k)})b_{k\infty} + 1\}$$

$$[1-\{1-2(p+1)\tilde{\mu}_k\}^{M+2} + 2\sum_{n=1}\{1-n\tilde{\mu}_k\}^{M+2}$$

$$-\{1-(p+n+1)\tilde{\mu}_k\}^{M+2}] \qquad (7-26)$$

and

$$E[C_{2k}(p,M-1)] = E[C_{3k}(p,M-1)]$$

$$\tilde{-} \frac{1}{2}|b_{k\infty}|^2[1-2\{1-(p+1)\tilde{\mu}_k\}^{M+4} + \{1-2(p+1)\tilde{\mu}_k\}^{M+4}] \qquad (7-27)$$

Again, for a large number of data blocks

$$\lim_{M\to\infty} E[|b_k(p,M)|^2] = \frac{1}{2}\tilde{\mu}_k\{|b_{k\infty}|^2$$

$$+ \left(\frac{|a(k)|^2-1}{a(k)}\right) b_{k\infty}+1\} + |b_{k\infty}|^2 \qquad (7\text{-}28)$$

The last term in (7-28) is the square of the mean of the tap weight for a large number of data blocks. The first term is the variance which is propotional to the normalized step size $\tilde{\mu}_k$. The smaller in $\tilde{\mu}_k$ the smaller is the variance.

7.5 Mean Square Error of Re-circulating Algorithm

The mean square error for the frequency-domain filter having the input data re-circulated p times can be simply evaluated as follows:

$$\begin{aligned}\varepsilon_p^2 &= E[|\hat{d}_m(k) - \hat{y}_k(p,m)|^2] \\ &= E[|a(k)z_m(k) - b_k(p,m)z_m(k)|^2] \\ &\quad + E[|n_{1k}(m)|^2 + E[|b_k(p,m)n_{2k}(m)|^2]\end{aligned} \qquad (7\text{-}29)$$

Using (7-2), we can write

$$\begin{aligned}\varepsilon_p^2 &= \sigma_{sk}^2 \{ |a(k) - E[b_k(p,M)]|^2\} \\ &\quad + \sigma_{nk}^2 \{ 1 + E[|b_k(p,M)|^2]\}\end{aligned} \qquad (7\text{-}30)$$

Substituting the values of $E[b_k(p,M)]$ and $E[|b_k(p,M)|^2]$ from (7-17) and (7-24) respectively into (7-30), the value of the mean-square error after re-circulating the input data p times can be evaluated. Note that if p=0, the values of $E[b_k(0,M)]$ and $E[|b_k(0,M)|^2]$ can be more accurately obtained by using (A-10) in Appendix H and (7-24). Again, if there is a large number of data blocks, i.e. for $M \rightarrow \infty$,

the mean square error becomes

$$\varepsilon_{\infty}^2 = \lim_{M\to\infty} \varepsilon_p^2 = \varepsilon^2_{min} + \varepsilon^2 \qquad (7\text{-}31)$$

where

$$\varepsilon^2_{min} = \sigma^2_{nk} \left[\frac{|a(k)|^2 \sigma^2_{sk}}{(\sigma^2_{sk} + \sigma^2_{nk})} + 1\right] \qquad (7\text{-}32)$$

and

$$\varepsilon^2 = \frac{\tilde{\mu}_k}{2} \left[\frac{|a(k)|^2 \sigma^4_{sk}}{(\sigma^2_{sk} + \sigma^2_{nk})^2} + (|a(k)|^2 - 1) \frac{\sigma^2_{sk}}{(\sigma^2_{sk} + \sigma^2_{nk})} + 1\right] \qquad (7\text{-}33)$$

Here $\varepsilon^2 min$ is a constant for given values of a(k), signal and noise power, and represents the lower bound of the mean-square error when the frequency-domain adaptive filter is applied to the input signal and noise charcterised by (7-2). ε^2 on the other hand, is a non-negative quantity dependent on $\tilde{\mu}_k$. In theory, if we have an infinite amount of input data, we can reduce the mean-square error arbitrarily close to the lower bound by choosing $\tilde{\mu}_k$ very small. For a limited supply of data, however, the dependence of $\varepsilon^2 p$ on $E[b_k(p,M)]$ and $E[|b_k(p,M)|^2]$ means that by re-circulating the input signal, the mean-square error will be reduced further. As a comparison of re-circulating algorithms with various values of p, we plot the normalized step-size $\tilde{\mu}_k$ against the relative mean-square error $\varepsilon^2 Rp$ which is defined as

$$\varepsilon^2 Rp = \frac{\varepsilon^2_p - \varepsilon^2_{min}}{\varepsilon^2_{min}}, \quad p = 0,1 \qquad (7\text{-}34)$$

where $\varepsilon^2 min$ is the minimum mean-square error given by (7-32)

and $\varepsilon^2 p$, the mean-square error of the filter having the input data re-circulated p times, is given by (7-30). The results are shown in figure 7-3, where M, the number of blocks, is taken to be 160, and the signal-to-noise ratio is chosen to be -5 dB. The range of values of $\tilde{\mu}_k$ is chosen so that $\tilde{\mu}_k << 1$, an assumption necessary to arrive at (7-30).

The graphs for various of p exhibit common characteristics:

1) For the same value of $\tilde{\mu}_k$, $\varepsilon^2 Rp < \varepsilon^2 Rq$ for $p > q$. This result is obvious from (7-17), (7-24) and (7-30) where it can be observed that $\varepsilon^2 p$ decreses with p.

2) Each of the curves decreases as $\tilde{\mu}_k$ increases until a minimum value of $\varepsilon^2 Rp$ is reached and then the relative mean-square error increases again beyond this minimum. This result is reasonable because a very small normalized step-size $\tilde{\mu}_k$ will take a long time (i.e. a large amount of data) to converge to the minimum error. For a limited supply of input data (M=160 in this case), a large $\tilde{\mu}_k$ will yield smaller error. However, a too large normalized step-size will yield large error again since the adjustment will then be too coarse. The optimum normalized step-size $\tilde{\mu}_{opt}$ for various values of p in this particular case are shown in Table 7-1. The corresponding values of $\varepsilon^2 Rp$ are also tabulated.

3) It is observed that the optimum normalized step-size

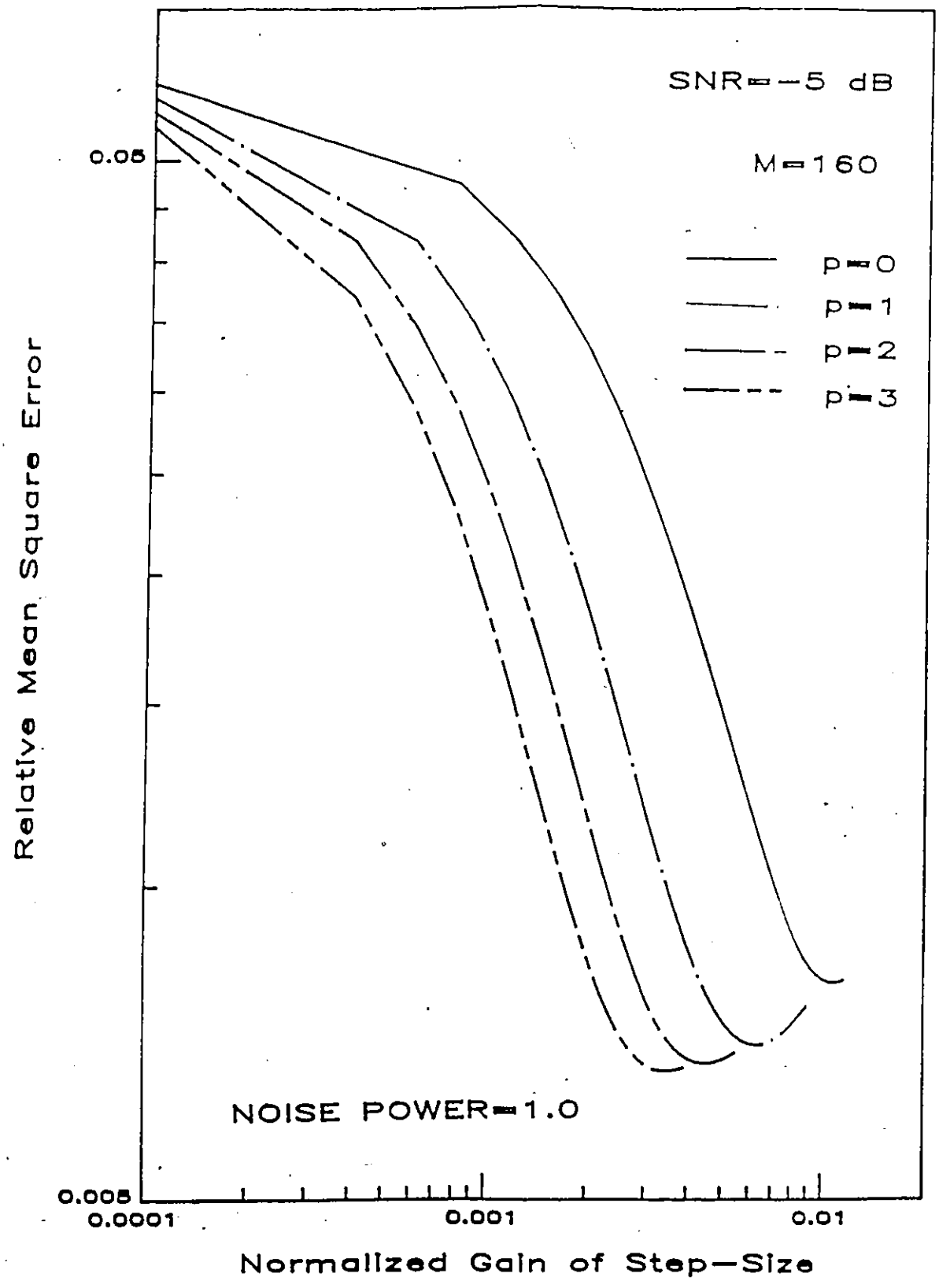


Figure 7.3 Comparison of Performances of Adaptive Filter with Various Number of Input Re-circulationa

p	$\tilde{\mu}_{opt}$	ε^2_{Rp}
0	0.0108	8.044×10^{-3}
1	0.0063	7.023×10^{-3}
2	0.0044	6.756×10^{-3}
3	0.0035	6.647×10^{-3}

Table 7.1 Optimum normalized step-size for different values of p.

$\tilde{\mu}_{opt}$ decreases as p increases. Furthermore, the optimum normalized step-size obeys approximately the relationship that $(p+1)\tilde{\mu}_{opt} = 0.0124$. Intuitively the re-circulating of the input data several time is analogous to having large supply of input data, therefore, for a smaller $\tilde{\mu}_k$ yields smaller error.

7.6 Simulation and Comparison of Performance

To evaluate the performance of the new algorithm of re-circulating input data, and to compare it with the Dentino - type or the normal frequency - domain adaptive filtering, a computer program is written to simulate the two methods.The simulation program can essentially be represented by the block diagram as shown in figure 7.4.

The signal generator produces a discrete circular Gaussian process the Fourier transform of which obeys (7-2). This signal is separated into two branches. The Lower branch carries the signal $\{s_d\}$ which encounters additive circular Gaussian noise $\{n_1\}$ generated by a noise generating subroutine. This mixture of signal and noise represents the desired signal sequence {d}. The upper branch passes the generated signal through a linear channel and then a circular Gaussian noise sequence $\{n_2\}$ is added. This represents the received signal sequence {x}.

The frequency-domain adaptive filters with and without input sample re-circulation are simulated according to figure

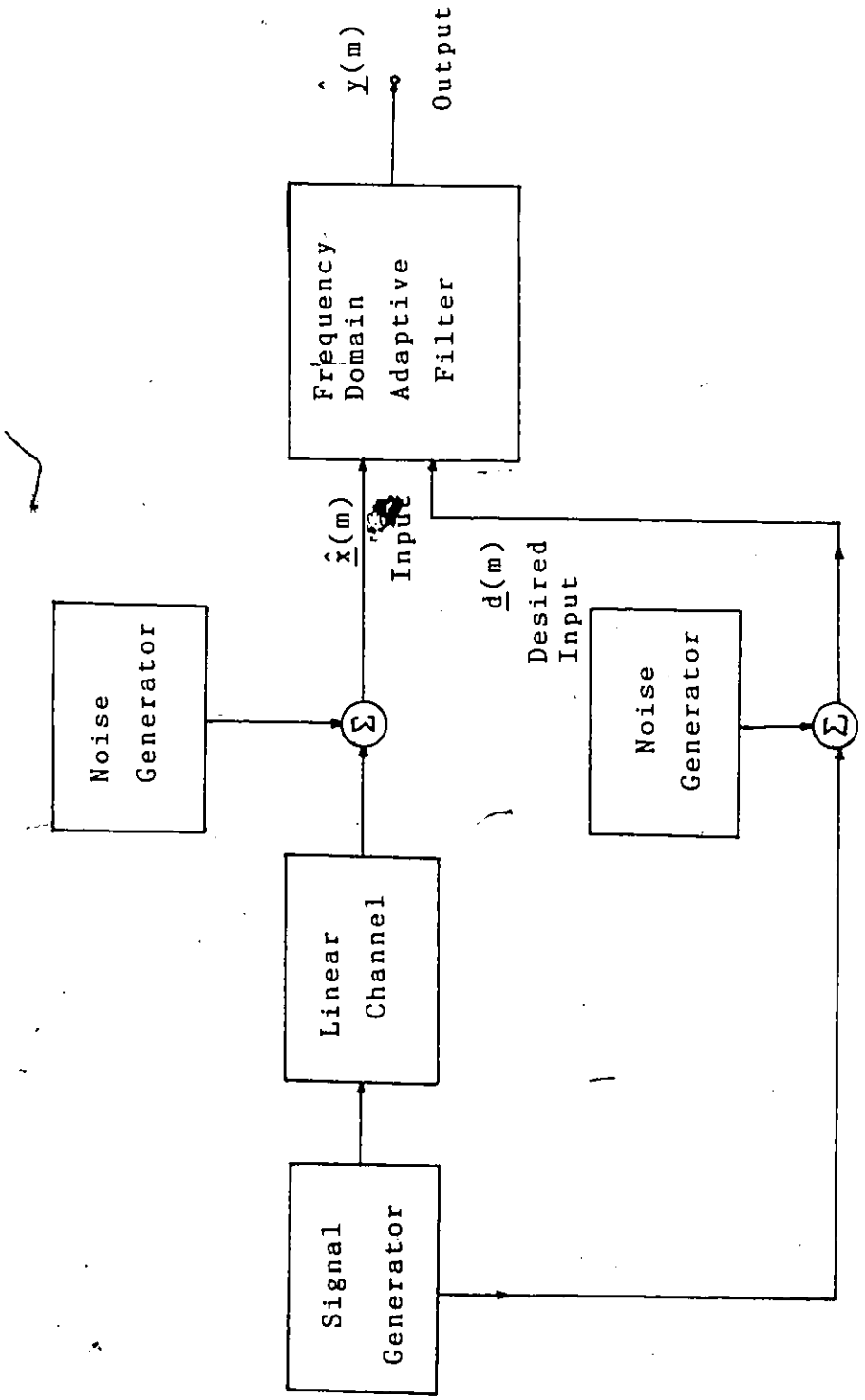


Figure 7.4 Block Digram for Computer Simulation.

7-1 and 7-2 respectively. In our simulation, both the adaptive filters utilise an FFT of 32 points, i.e. N=32, and both use the complex LMS algorithm developed by Widrow et al. [40]. The comparison of performances is carried out between the adaptive filter without re-circulation and the adaptive filter with re-circulates the input samples once only. With m blocks of input data, the filter without re-circulation facility would have m iterations of adjusting the tap-weights The mean-square error, ε^2_0 (m) which is a function of m, is evaluated. The relative mean-square error, ε^2_{R0}(m) defined in (7-34) for p=0 is then calculated. Now, with the same m blocks of input data and no more, the performance of the adaptive filter with the facility of re-circulating the input data once is tested . This is equivalent to having a normal frequency-domain adaptive filter with no re-circulation facility but having the same m blocks of input data twice. Again, the mean-square error ε^2_1 (m) and the relative mean-square error ε^2_{R1}(m) are calculated. The values of ε^2_{R0}(m) and ε^2_{R1}(m) are plotted for various values of m. These theoretical mean-square errors for the two filters are also plotted for comparison.The following examples will illustrate the comparison of the performances of the two filters.

Example 1

In this example we choose the linear channel shown in figure 4 to be a pure delay, i.e. , the desired and received signals are respectively given by

$$d(n) = s_d(n) + n_1(n)$$

$$x(n) = s_d(n-v_0) + n_2(n)$$

where v_0 is a constant. The variance of s(n) is chosen to be unity and signal to noise ratio at the input of the filter is 9.55 dB so that the variances of $n_1(n)$ and $n_2(n)$ are each equal to 0.1109. The total data length used for simulation is 4800 samples and is divided into 150 blocks each having 32 samples. These blocks are processed by the frequency-domain adaptive filter. For every m blocks of data the relative mean-square errors $\varepsilon^2_{RO}(m)$ and $\varepsilon^2_{R1}(m)$ are evaluated and plotted in figure 5. Also shown in Fig.5 are the theoretical values of these relative mean-square errors. It can be seen that the mean-square errors from simulation agree, in general, with their theoretical counterparts.

Other examples were tested, and it was observed that the relative mean-square errors from simulations were in very close agreement with their theoretical values. Thus, in order to depict the comparsion of the performances of the two filters more clearly so that curves of very close values are avoided, only the theoretical values of the relative mean-square errors are plotted. Note that, the relative mean - square error defined in (7-34) is the ratio of the difference of the mean-square error and the minimum MSE. This means that in both the re-circulation and without re-circulation adaptive filtering algorithms we compared the results with the reminiscent frequency-domain

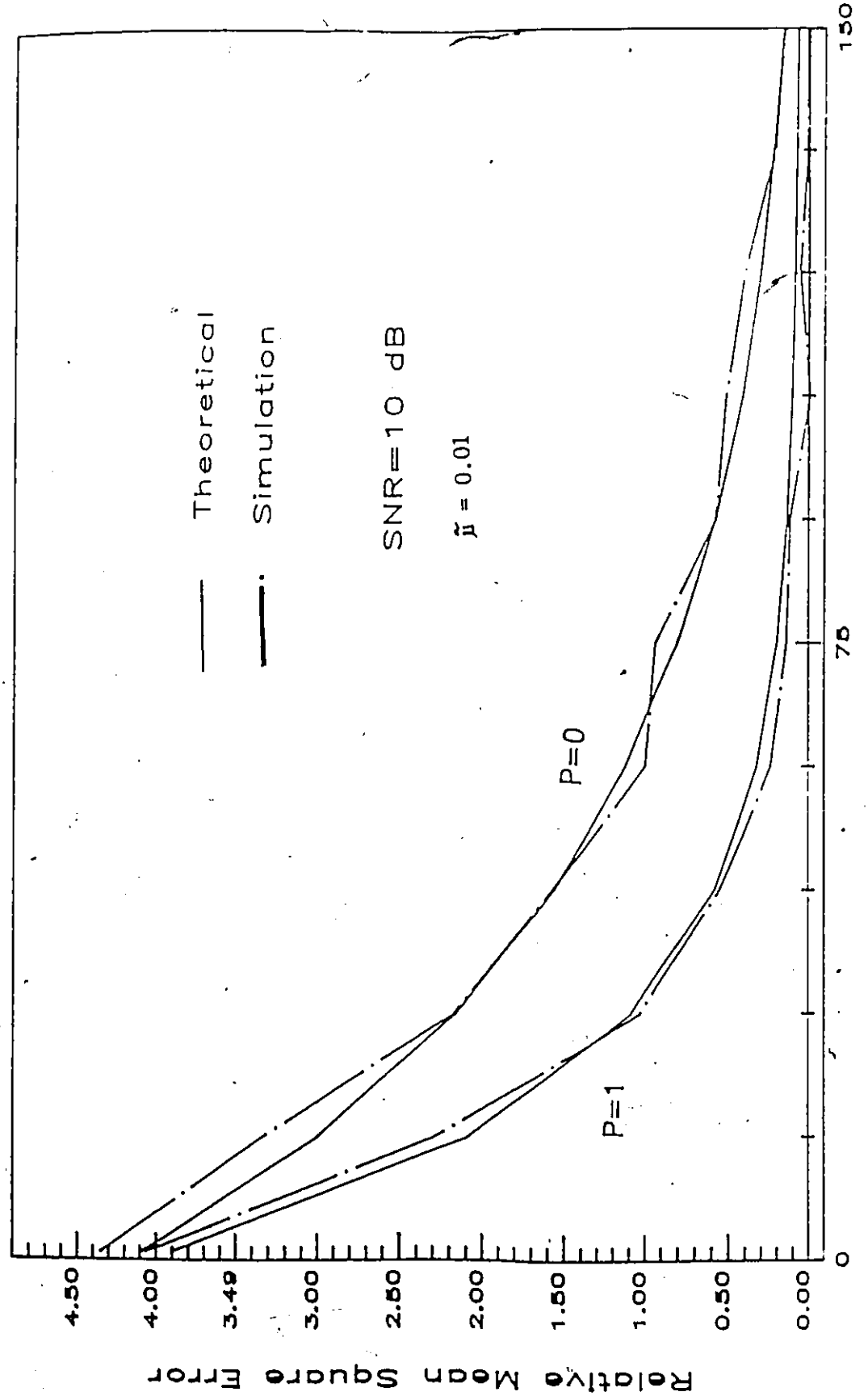


Figure 7.5 Comparison of Performances for the Adaptive Filters in Example 1

optimum Wiener solution. Because the Dentino-type frequency domain adaptive algorithm we considered is not the direct realization of the conventional time domain LMS algorithm so that the reminiscent frequency domain optimum solution is not necessarily equivalent to the time domain Weiner solution. This implies that the minimum mean-square error in the time domain is, in general, not equal to the Dentino - type frequency domain minimum mean - square error. Further, in order to match the assumptions we made in equations (7-2) for the frequency domain algorithm, the signals and noises in time domain were assumed to be white Gaussian random processes which results in similar values of the minimum MSE for both time-domain and frequency-domain algorithms.

Example 2

In this example, the signal model and channel model are chosen to be identical to those in Example 1, and the purpose here is to show the effect of step size on the relative mean-square error. We show here two cases, the first has a high signal-to-noise ratio (SNR) of 10 dB, and the second has a low SNR of -5 dB. In each case we use normalised step-size $\tilde{\mu}_k$ of 0.01, 0.0025 and 0.001; and in each case we compare the adaptive filter without re-circulation of input data to that with input data re-circulated once. The results are shown in figures 7-6 and 7-7 for the high and low SNR respectively. The scale of

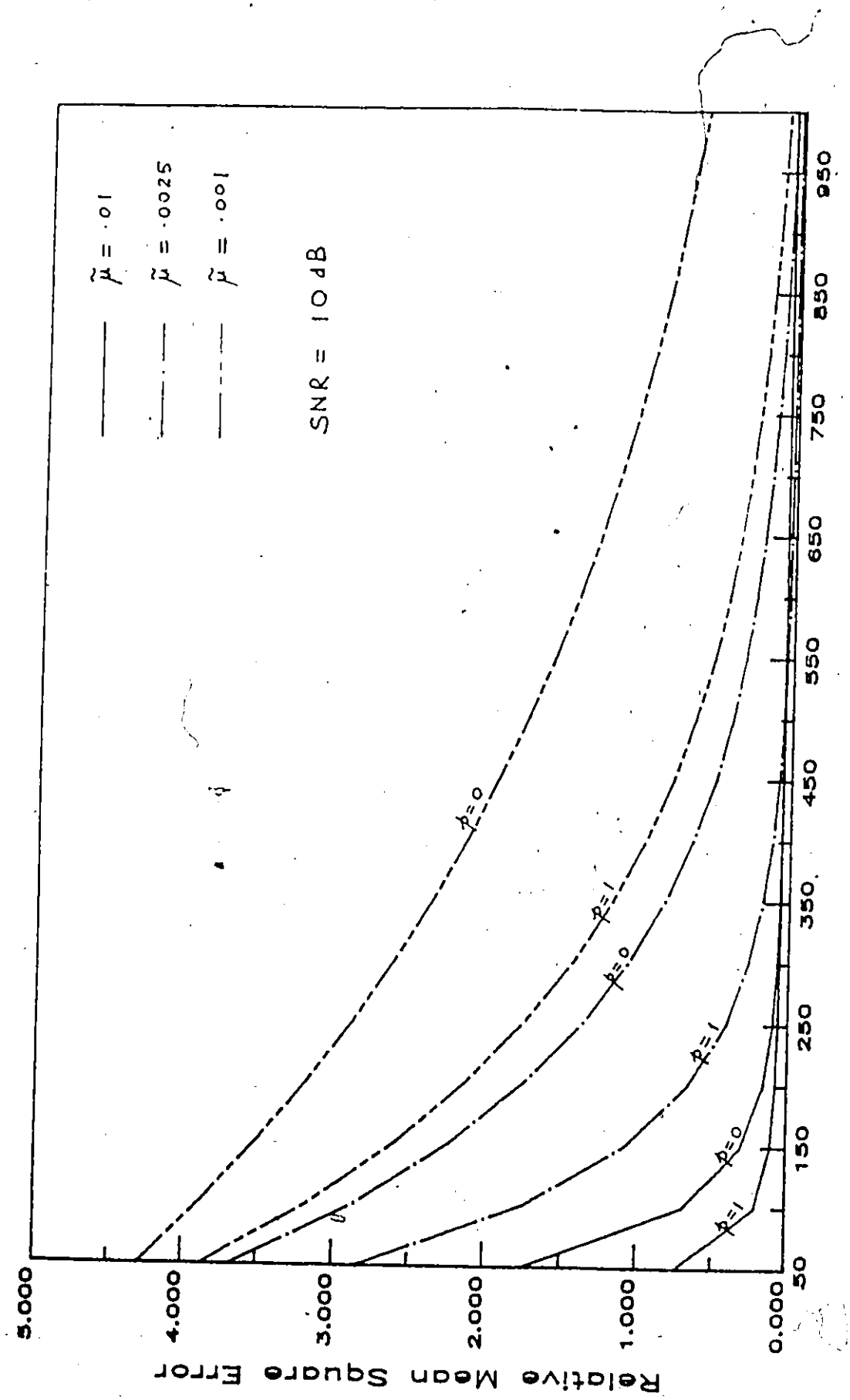


Figure 7.6 Comparison of Performances for the Adaptive Filters in Example 2 with SNR= 10 dB.

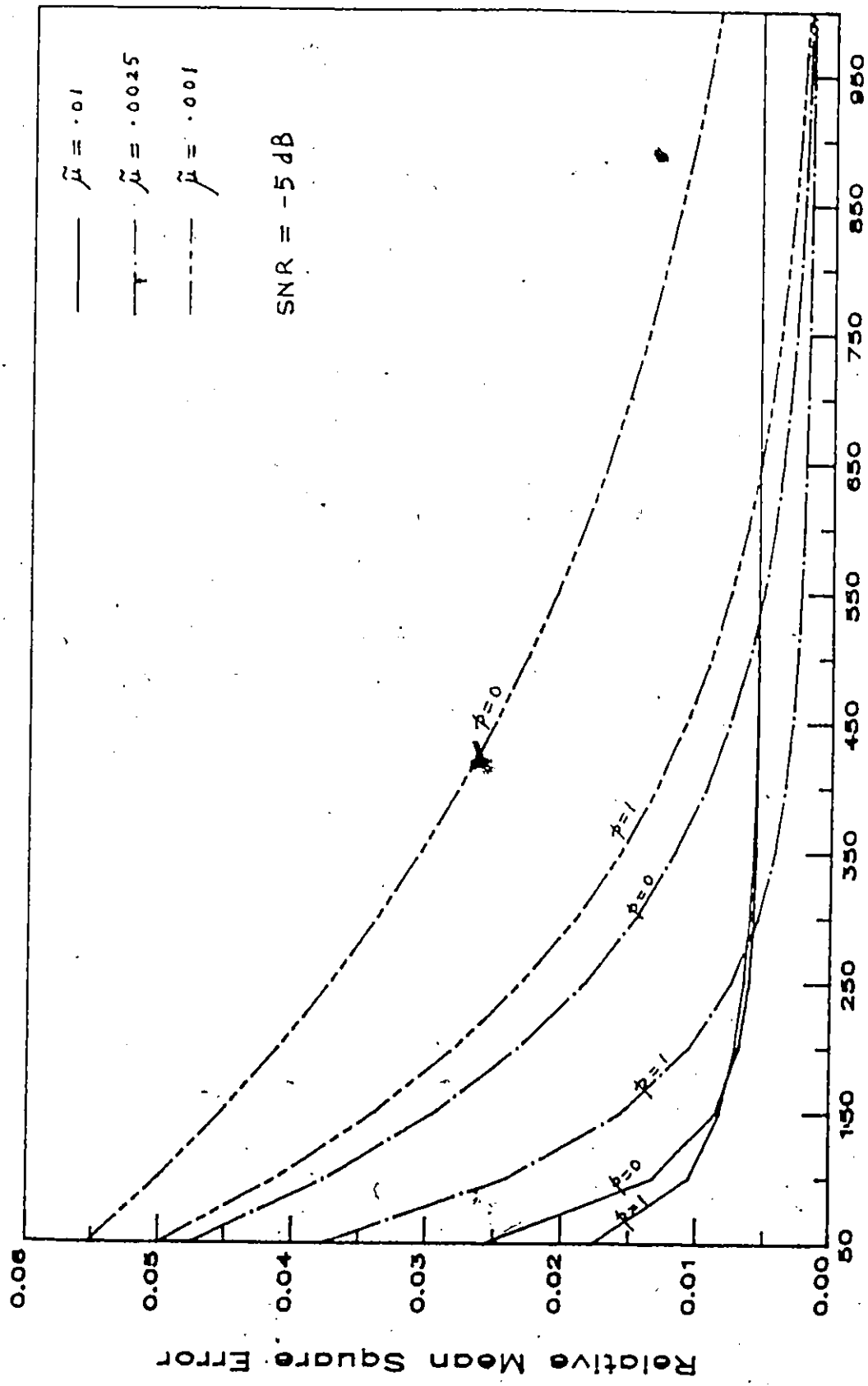


Figure 7.7 Comparison of Performances for the Adaptive Filters in Example 2 with SNR = -5 dB.

relative mean-square error in Fig. 7-6 is of two order higher than that in Fig. 7-7. This is because the noise power is the same in both cases whereas the SNR are vastly different. However, since the desired signal $d_k(m)$ consist of equal power, the minimum mean-square error $\mathcal{E}^2_{min}$ in both cases would be of similar order as evidence in (7-33). Indeed, the minimum mean-square error are 1.91 and 1.24 respectively for the high and low SNR cases. In all cases, as observed from figures 7-6 and 7-7, the improvement in performance by re-circulating the input data is significant.

7.7 Conclusions

In this chapter we have introduced the re-circulating input data algorithm to a frequency domain adaptive filter The algorithms was applied to a situation often encountered in noise cancellation and time-delay estimation, we have also developed an analysis which expresses the mean-square error of the filter as a function of p, the number of data re-circulations, and M, the number of data blocks available, for given values of signal power, noise power, and step-size of tap adjustment. The theoretical results were verified by computer simulation. In all test cases, there is significant improvement of the adaptive filter performance in terms of mean-square error by re-circulating the input data. In the cases where the number of data blocks is relatively small, by re-circulating the input

data once almost invariably reduces the relative mean-square error to half of that yielded by the filter without data re-circulation . Further re-circulation of the data, p>2, however, results in less dramatic improvements. Figure 7.8 shows the relative mean-square errors for different number, p, of data re-circulation. Here, we have chosen that

$$(p+1)\tilde{\mu}_k = 1.24 \times 10^{-2}$$

The reason for this has been stated in observation 3) for figure 7.3 where the optimum normalized step-size obeys (7-35). Once this relationship is satisfied, we are ensured that an optimum step-size is chosen. Furthermore, in the development of (A-9), (7-26), (7-27), and (7-28), we have assumed that the normalized step-size is very much less than unity. (7-35) again ensures this condition to be satisfied. However, we feel, in the real time implementation the step-size , $\tilde{\mu}_k$, could decrease after each pass or the re-circulation of data in order to approach closely to the minimum mean-square error. In practical applications of adaptive filtering to sonar systems, the supply of input data is often limited. It is in such cases that the algorithm of re-circulation of input data becomes attractive..

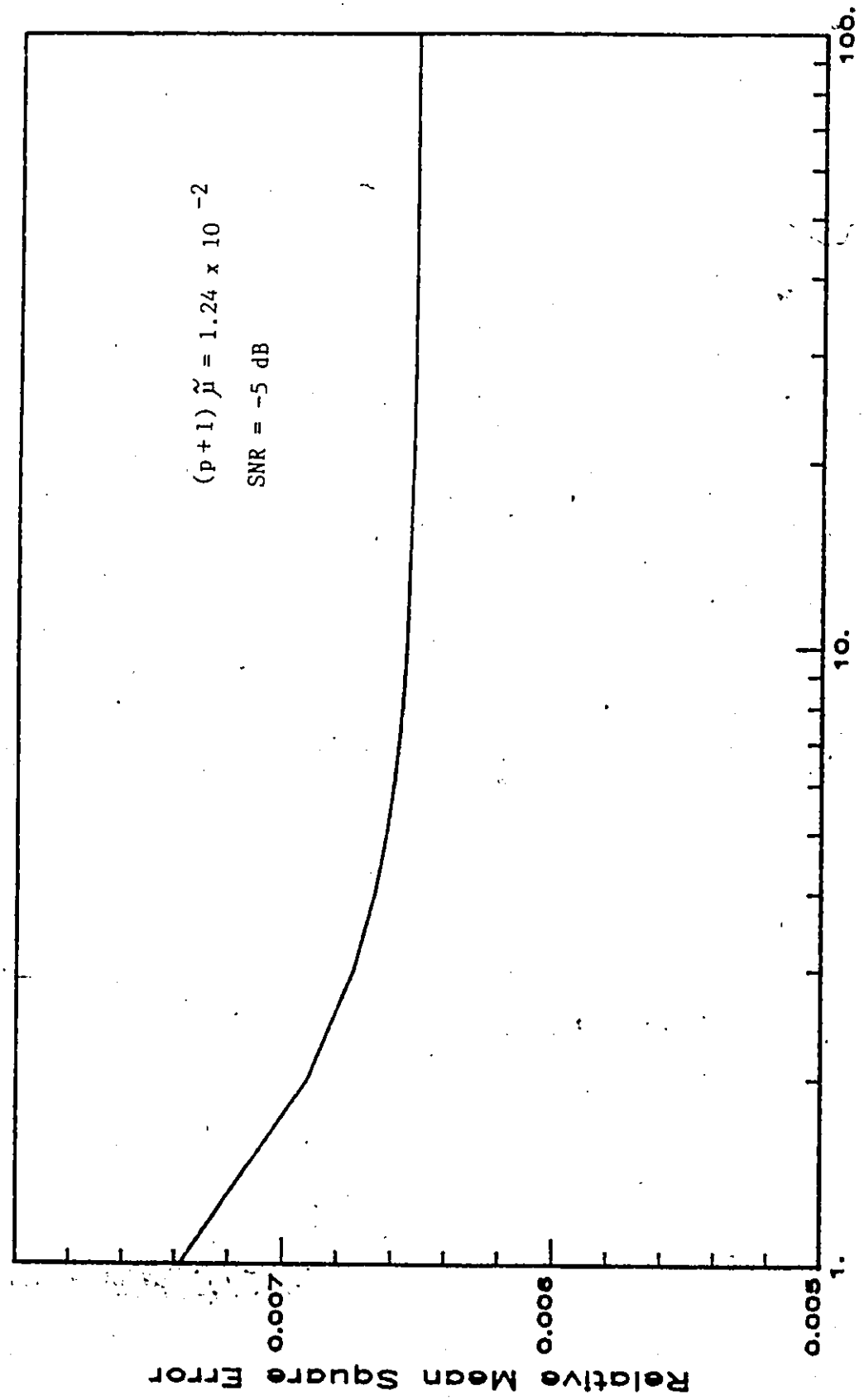


Figure 7.8 Comparison of Performances for the Adaptive Filter with Various Number of Re-circulations (SNR = -5 dB).

CHAPTER 8

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

8.1 Conclusions

A comprehensive study of the transformed domaim adaptive filtering algorithms is addressed in this thesis. The basic ideas behind this study is to expand the time domain filter impulse response as a set of weighted orthogonal functions. Based on this concept, we have developed two types of transformed domain adaptive algorithms viz, the TDLMS adaptive algorithm and the BTDLMS adaptive algorithm.

To study the effect of the normalized K-L TDLMS adaptive algorithm, we derived an analytic expressions of MSE in terms of mean weight vector, weight covariance, input autocorrelation matrix and the minimum MSE. Moreover, for a limited supply of input signal, a re-circulation input data in the BTDLMS adaptive algorithm with the DFT is proposed. Similarly, for analyzing the effect of the re-circulation algorithm, we obtained a closed form expression of the MSE. The performance in terms of convergence rate for these algorithms are verified by computer simulation, based on the analytic expressions of MSE. The main results and the contributions of this thesis are as follows.

1. A new comprehensive treatment of the generalized orthogonal transformed domain LMS adaptive filters is given. Furthermore, for the purpose of comparison, the time domain expressions of various transformed domain adaptive algorithms are also obtained. We showed that the K-L TDLMS adaptive algorithm and the normalized K-L TDLMS algorithm in the time domain are equivalent to the time domain LMS algorithm and the stochastical analog of the generalized Newton-Raphson method, respectively.

2. Theoretical analysis of the normalized K-L TDLMS adaptive algorithm in the time domain for both real and complex valued signals are performed. From analytical expressions for both real and complex signals, of the steady-state MSE, we found that the steady-state MSE value of the normalized K-L TDLMS in the time domain is independent of the eigenvalues of the input autocorrelation matrix. These agree with the results obtained in chapter 4 for the error weight vector. That is the convergence rate of the present weight vector converges to the optimum weight vector is independent of the eigenvalues.

For further investigation of the transient MSE, a complex valued normalized K-L TDLMS adaptive algorithm is used in the application of the ALE. The analytical expression of transient MSE enables us to explore the effect on the convergence rate due to the power ratio between the sinusoidal signals in ALE with high SNR. An analytical expression of transient MSE using the time domain LMS ALE was

also obtained, for the purpose of comparison. It was also found that the convergence rate of the time domain LMS algorithm not only depends on the eigenvalue spread but also depends highly on the power ratio between the sinusoidal signals in the ALE. On the other hand, the transient MSE of the normalized K-L TDLMS ALE is independent of the power ratio between the sinusoidal signals of the received signal but depends only on the stepsize.

3. The problem of employing different orthogonal transforms to the transformed domain adaptive algorithms is also studied.The effect due to the incomplete diagonalization in the transformed domain adaptive algorithm is examined using the DCT.It showed that the covergence rate of the TDLMS adaptive algorithm with normalized DCT is slower than the normalized K-L TDLMS adaptive algorithm. This is true even when the eigenvalues of the input autocorrelation matrix can be exactly estimated. Furthermore, in order to apply the TMT in the transformed domain adaptive algorithms, the simplified form of the TMT is derived. This results in requiring small amount of memory and reduced the computational effort compared to the original TMT.

4. The implementation of TDLMS adaptive algorithm requires more computation compared to the time domain LMS algorithm. To circumvent this problem, a recursive equation for obtaining the DCT output signals is obtained.This results in simpler implementation and less computation compared to the fast DCT, especially, when the order of the adaptive

filter is very large.

5. Finally, for the limited supply of input signal, a re-circulation algorithm in the BTDLMS adaptive algorithm with the DFT is proposed and analyzed, a closed form expression, for the single complex weight in the re-circulation algorithm has been derived, which allows the statistical analysis to be carried out.The theoretical result of the transient MSE agrees well with the simulation result. Also, the performance in terms of the MSE is significantly improved compared to the without re-circulation algorithm.

8.2 Suggestions for Future Work

Although various theoretical analyses of the transformed domain adaptive algorithms have been given at the present study, some observations and suggestions for the future work are investigated as follows.

1. In chapter 5, we observed that the time domain LMS algorithm, in general, converges fast on the first few iterations. Although, the normalized K-L TDLMS algorithm can be used to improve the convergence rate in the case that the eigenvalue spread is large, however, it requires more computation. One worthwhile problem to study would be to implement the adaptive algorithm in a more efficient way, i.e., by using a hybrid algorithm which combines both the time domain LMS algorithm and the normalized K-L TDLMS algorithm. That is, on the first few iterations, we

utilize the time domain LMS algorithm and after that the normalized K-L TDLMS algorithm is used instead. Moreover, it would be interesting if we can decide the best iteration number, theoretically, in which the time domain LMS algorithm should be used in the implementation of hybrid adaptive algorithm.

2. Another useful problem is to study how the computation can be reduced for the TDLMS adaptive algorithm. One way to achieve this is to process the data in a blockwise manner such that the tap-weights in the TDLMS algorithm are up-dated only after block of data. This is very similar to the time domain BLMS algorithm, which is the blockwise processing of the time domain LMS algorithm. Furthermore, it may be also interesting if an efficient algorithm can be derived for implementing such block processing algorithm.

3. In the re-circulation input data algorithm, we used a fixed constant step-size to derive the analytical expression for the transient MSE. However, in real time implementation the step-size could be made to decrease after each pass or re-circulation of data in order to approach closely to the minimum MSE.

4. Finally, it may also be useful to obtain a recursive equation for obtaining the variable step-size such that the algorithm can be implemented efficiently in the non-stationary enviroments, especially when the transformed domain adaptive filtering algorithms are used.

APPENDIX A

In this appendix, we develope the relationship between the generalized Newton-Raphson method and the standard steepest descent method.

Consider a quadratic function, $F(\underline{w})$, as described in (2-2), i.e.,

$$F(\underline{w}) = \underline{w}^T \underline{R} \underline{w} - 2 \underline{p}^T \underline{w} + c \qquad (2\text{-}2)$$

where the quantities, $\underline{R}$, $\underline{p}$, and c were defined in (2-3). In order for us to see the relation between the standard steepest descent method and the generalized Newton-Raphson method, first, let us rescale the weight vector, $\underline{w}$, as

$$\underline{b} = \hat{\underline{Q}}^{-1} \underline{w} \qquad (A\text{-}1)$$

or equivalently the weight vector is represented as

$$\underline{w} = \hat{\underline{Q}} \underline{b} \qquad (A\text{-}2)$$

Now, by substituting (A-2) into (2-2), the cost function, $F(\underline{w})$, can be expressed in terms of the rescaled weight vector, $\underline{b}$, as follows

$$\begin{aligned} \hat{F}(\underline{b}) &= F(\underline{w}) \\ &= \underline{b}^T \hat{\underline{Q}}^T \underline{R} \hat{\underline{Q}} \underline{b} - 2 \underline{p}^T \hat{\underline{Q}} \underline{b} + c \\ &= \underline{b}^T \underline{R}^{\#} \underline{b} - 2 \underline{p}^T \hat{\underline{Q}} \underline{b} + c \end{aligned} \qquad (A\text{-}3)$$

where

$$\underline{R}^{\#} = \hat{\underline{Q}}^T \underline{R} \hat{\underline{Q}} \qquad (A\text{-}4)$$

where $\hat{\underline{Q}}$ is an arbitrary NxN symmetric, square matrix. Again, $\hat{F}(\underline{b})$ is the quadratic function of $\underline{b}$. The case in which we are interested is to choose $\hat{\underline{Q}}$ such that

$$\underline{R}^{\#} = \hat{\underline{Q}}^T \underline{R} \hat{\underline{Q}}$$

$$= \underline{I}_{NxN} \qquad \text{(A-5a)}$$

or equivalently we have

$$\hat{\underline{Q}}\,\hat{\underline{Q}}^T = \underline{R}^{-1} \qquad \text{(A-5b)}$$

Since $\hat{F}(\underline{b})$ is a quadratic function of $\underline{b}$, the standard steepest descent method for searching the minimum of $\hat{F}(\underline{b})$ with respect to $\underline{b}$ can be expressed as

$$\underline{b}(n+1) = \underline{b}(n) + a_n^{\#}\, \underline{r}^{\#}(n) \qquad \text{(A-6)}$$

Here, the direction vector, $\underline{r}^{\#}(n)$, is chosen to be the negative gradient vector of $\hat{F}(\underline{b})$

$$\begin{aligned} \underline{r}^{\#}(n) &= \underline{g}^{\#}(n) \\ &= -\,\underline{\nabla}\hat{F}(\underline{b}(n)) \\ &= -\,2\,[\,\underline{R}^{\#}\underline{b} - \underline{Q}^T\underline{p}] \end{aligned} \qquad \text{(A-7a)}$$

and the variable stepsize, $a^{\#}_n$, is given by

$$a^{\#}_n = \frac{-\,\langle\, \underline{r}^{\#}(n),\, \underline{\nabla}\hat{F}(\underline{b}(n))\,\rangle}{2\,\langle\, \underline{r}^{\#}(n),\, \underline{R}^{\#}\,\underline{r}^{\#}(n)\,\rangle} \qquad \text{(A-7b)}$$

To explore the relationship with the generalized Newton-Raphson method, we premultiply both sides of (A-6) by $\hat{\underline{Q}}$, which results in

$$\begin{aligned} \underline{w}(n+1) &= \hat{\underline{Q}}\,\underline{b}(n+1) \\ &= \hat{\underline{Q}}\,\underline{b}(n) - a_n^{\#}\,\hat{\underline{Q}}\;\;\underline{\nabla}F(\underline{b}(n)) \\ &= \underline{w}(n) - 2\,a_n^{\#}\,\hat{\underline{Q}}\,[\;\underline{R}^{\#}\underline{b} - \hat{\underline{Q}}^T\,\underline{p}\;] \end{aligned} \qquad \text{(A-8)}$$

where

$$\underline{\nabla}\hat{F}(\underline{b}(n)) = 2\,\underline{R}^{\#}\underline{b}(n) - 2\,\hat{\underline{Q}}^T\,\underline{p} \qquad \text{(A-9)}$$

Apply (A-4) and (A-5) to (A-8), results in

$$\underline{w}(n+1) = \underline{w}(n) - 2\,a_n^{\#}[\hat{\underline{Q}}\,\hat{\underline{Q}}^T\underline{R}\,\hat{\underline{Q}}\,\underline{b}(n) - \hat{\underline{Q}}\,\hat{\underline{Q}}^T\underline{p}]$$

$$= \underline{w}(n) - 2\, a_n^{\#}\, \underline{R}^{-1}[\, \underline{R}\, \underline{w}(n) - \underline{p}]$$

$$= \underline{w}(n) - a_n^{\#}\, \underline{R}^{-1}\, \underline{\nabla} F(\underline{w}(n)) \qquad \text{(A-10)}$$

Similarly, it can be shown that

$$a^{\#}_n = \frac{-\langle\, \underline{r}^{\#}(n),\, \underline{\nabla} \hat{F}(\underline{b}(n))\, \rangle}{2 \langle\, \underline{r}^{\#}(n),\, \underline{R}^{\#}\, \underline{r}^{\#}(n)\, \rangle} = a_n \qquad \text{(A-11)}$$

because

$$\langle\, \underline{r}^{\#}(n),\ \underline{\nabla} \hat{F}(\underline{b}(n))\, \rangle = \langle\, \underline{r}(n),\, \underline{\nabla} F(\underline{w}(n))\, \rangle$$

and

$$\langle\, \underline{r}^{\#}(n),\, \underline{R}^{\#} \underline{r}^{\#}(n)\, \rangle = \langle\, \underline{r}(n),\ \underline{R}\, \underline{r}(n)\, \rangle \qquad \text{(A-12)}$$

where

$$\underline{r}(n) = -\, \underline{R}^{-1}\, \underline{\nabla} F(\underline{w}(n))$$

$$= \underline{R}^{-1}\, \underline{g}(n)$$

$$= \underline{G}(n)\, \underline{g}(n) \qquad \text{(A-13)}$$

Here the stationarity for the autocorrelation matrix, $\underline{R}$, has been assumed. Now, from (A-10) and (A-11), we can recognize that the rescaled version of the standard steepest descent method is equivalent to the generalized Newton - Raphson method.

APPENDIX B

Applying (4-14b) to the (n,m) element of the matrix of the third term of Eq.(4-13), yields

$$\{ E[\underline{x}(k)\underline{x}^T(k)\ E[\underline{w}(k)\underline{w}^T(k)]\ \underline{x}(k)\underline{x}^T(k)] \}_{n,m}$$

$$= E\{x_n(k) \sum_{i,j=1}^{N} x_i(k)\ \{E[\underline{w}(k)\underline{w}^T(k)\}_{i,j} x_j(k)x_m(k)\}$$

$$= \sum_{i,j=1}^{N} \{E[\underline{w}(k)\underline{w}^T(k)]\}_{i,j}\ \{E[x_n(k)x_i(k)x_j(k)x_m(k)\}$$

$$= \sum_{i,j=1}^{N} E[x_n(k)x_i(k)]E[\underline{w}(k)\underline{w}^T(k)]\}_{i,j} E[x_j(k)x_m(k)]$$

$$+ \sum_{i,j=1}^{N} E[x_n(k)x_j(k)]\{E[\underline{w}(k)\underline{w}^T(k)]\}_{i,j} E[x_i(k)x_m(k)]$$

$$+ \sum_{i,j=1}^{N} E[x_n(k)x_m(k)]\{E[\underline{w}(k)\underline{w}^T(k)]\}_{i,j} E[x_i(k)x_j(k)] \qquad \text{(B-1)}$$

Thus, it follows that

$$E\{ \underline{x}(k)\underline{x}^T(k)\ E[\underline{w}(k)\underline{w}^T(k)]\ \underline{x}(k)\underline{x}^T(k)\}$$
$$= \underline{R}\ E[\underline{w}(k)\underline{w}^T(k)]\ \underline{R} + \underline{R}\ E[\underline{w}(k)\underline{w}^T(k)]\ \underline{R}$$
$$+ \underline{R}\ tr\{\ \underline{R}\ E[\underline{w}(k)\underline{w}^T(k)]\ \} \qquad \text{(B-2)}$$

Note that, the last term on the right hand side of (B-2) can be written as

$$\underline{R}\ \mathrm{tr}\{\ \underline{R}\ E[\underline{w}(k)\underline{w}^T(k)]\ \}$$
$$= \underline{R}\ \mathrm{tr}\{\ \underline{R}\ \mathrm{Cov}\{\underline{w}(k)\underline{w}^T(k)\} + \underline{R}\ \underline{M}_{w(k)}\ \underline{M}^T_{w(k)}\ \}$$
$$= \underline{R}\ \mathrm{tr}\{\ \underline{R}\ \mathrm{Cov}\{\underline{w}(k)\underline{w}^T(k)\}\} + \underline{R}\ \underline{M}^T_{w(k)}\underline{R}\underline{M}_{w(k)}$$

(B-3)

Similarly, we consider the fourth term for the (n,m) element:

$$\{\ E[\underline{x}(k)\underline{x}^T(k)\ E[\underline{w}(k)]\ d(k)\underline{x}^T(k)\}_{n,m}$$

$$= E\{\sum_{i=1}^{N} x_n(k)x_i(k)\ \{E[\underline{w}(k)]\}_i\ d(k)\ x_m(k)\ \}$$

$$= \sum_{i=1}^{N}\{\ E[\underline{w}(k)]\}_i\ E[\ x_n(k)x_i(k)d(k)x_m(k)]$$

$$= \sum_{i=1}^{N} E[x_n(k)x_i(k)]\{E[\underline{w}(k)]\}_i E[d(k)x_m(k)]$$

$$+ \sum_{i=1}^{N} E[x_n(k)d(k)]\{E[\underline{w}(k)]\}_i\ E[x_i(k)x_m(k)]$$

$$+ \sum_{i=1}^{N} E[x_n(k)x_m(k)]\ \{E[\underline{w}(k)]\}_i\ E[x_i(k)d(k)]$$

(B-4)

Now, from (B-4) we have

$$E\{\underline{x}(k)\underline{x}^T(k)\ E[\underline{w}(k)]\underline{d}(k)\underline{x}^T(k)\}$$
$$= \underline{R}\ \underline{M}_{w(k)}\underline{P}^T + \underline{P}\ \underline{M}^T_{w(k)}\underline{R} + \underline{R}\ \underline{M}^T_{w(k)}\ \underline{P}$$

(B-5)

APPENDIX C

Apply (4-46) to the (n,m) element of the matrix of the third term of (4-48):

$$\{E[\underline{x}^*(k)\underline{x}^T(k)\ E[\underline{w}(k)\underline{w}^*(k)]\ \underline{x}^*(k)\underline{x}^T(k)]\}_{n,m}$$

$$= E\{x_n(k) \sum_{i,j=1}^{N} x_i(k)\ \{E[\underline{w}(k)\underline{w}^*(k)]\}_{i,j} x_j(k)x_m(k)\}$$

$$= \sum_{i,j=1}^{N} \{E[\underline{w}(k)\underline{w}^*(k)]\}_{i,j}\ \{E[x_n(k)x_i(k)x_j(k)x_m(k)\}$$

$$= \sum_{i,j=1}^{N} E[x_n(k)x_i(k)]E[\underline{w}(k)\underline{w}^*(k)]\}_{i,j} E[x_j(k)x_m(k)]$$

$$+ \sum_{i,j=1}^{N} E[x_j(k)x_i(k)]\{E[\underline{w}(k)\underline{w}^*(k)]\}_{i,j} E[x_n(k)x_m(k)] \qquad \text{(C-1)}$$

Thus, it follows that

$$E\{\underline{x}^*(k)\underline{x}^T(k)\ E[\underline{w}(k)\underline{w}^*(k)]\ \underline{x}^*(k)\underline{x}^T(k)\}$$
$$= \underline{R}\ E[\underline{w}(k)\underline{w}^*(k)]\ \underline{R} + \underline{R}\ tr\{\underline{R}\ E[\underline{w}(k)\underline{w}^*(k)]\} \qquad \text{(C-2)}$$

Now, consider the fourth term for the (n,m) element, i.e.,

$$\{E[\underline{x}^*(k)\underline{x}^T(k)\ E[\underline{w}(k)]\ d^*(k)\underline{x}^T(k)\}_{n,m}$$

$$= E\{\sum_{i=1}^{N} x^*_n(k)x_i(k)\ \{E[\underline{w}(k)]\}_i\ d^*(k)\ x_m(k)\}$$

$$= \sum_{i=1}^{N} \{E[\underline{w}(k)]\}_i\ E[x^*_n(k)x_i(k)d^*(k)x_m(k)]$$

$$= \sum_{i=1}^{N} E[x^*_n(k)x_i(k)]\{E[\underline{w}(k)]\}_i E[d^*(k)x_m(k)]$$

$$+ \sum_{i=1}^{N} E[x^*_n(k)x_m(k)]\ \{E[\underline{w}(k)]\}_i\ E[x_i(k)d^*(k)] \qquad (C\text{-}3)$$

Again, from (C-3) we have

$$E\{\underline{x}^*(k)\underline{x}^T(k)\ E[\underline{w}(k)]\underline{d}^*(k)\underline{x}^T(k)\}$$

$$= \underline{R}\ \underline{M}_{w(k)}\underline{p}^* + \underline{p}^*\ \underline{M}_{w(k)}\underline{R} \qquad (C\text{-}4)$$

APPENDIX D

In this appendix, we prove that if $\underline{Q}$ matrix daigonalizes $\mathrm{Cov}\{\underline{w}(0)\underline{w}^{\ddagger}(0)\}$ then it diagonalizes $\mathrm{Cov}\{\underline{w}(n)\underline{w}^{\ddagger}(n)\}$ for all n. To prove this, let us write (4-51) as follow:

$$\mathrm{Cov}\{\underline{w}(n+1)\ \underline{w}^{\ddagger}(n+1)\}$$

$$= (1-4\mu+4\mu^2)\ \mathrm{Cov}\{\underline{w}(n)\underline{w}^{\ddagger}(n)\}$$

$$+ 4\mu^2\ \mathrm{tr}\{\underline{R}\ \mathrm{Cov}\{\underline{w}(n)\underline{w}^{\ddagger}(n)\}\}\ \underline{R}^{-1}$$

$$+ 4\mu^2\ \underline{R}^{-1}\ \{\ E[\underline{d}(n)d^{*}(n)] - \underline{M}^{\ddagger}_{w(n)}\underline{p}$$

$$- \underline{p}^{\ddagger}\underline{M}_{w(n)} + \underline{M}^{\ddagger}_{w(n)}\ \underline{R}\ \underline{M}_{w(n)}\} \qquad (4\text{-}51)$$

Now, because that $\underline{R} = \underline{Q}\ \underline{R}_{\lambda}\ \underline{Q}$, we have

$$\underline{R}^{-1} = \underline{Q}\ \underline{R}^{-1}_{\lambda}\underline{Q}^{\ddagger} \qquad (D\text{-}1)$$

where

$$\underline{Q}^{\ddagger} = \underline{Q}^{-1} \qquad (D\text{-}2)$$

From (D-1) we can see that $\underline{Q}$ diagonalizes $\underline{R}^{-1}$. Furthermore, since $\mathrm{tr}\{\cdot\}$ and the bracket quantites in the last term of (4-51) are scalars, thus, if $\underline{Q}$ diagonalizes $\mathrm{Cov}\{\underline{w}(n)\ \underline{w}^{\ddagger}(n)\}$, $\underline{Q}$ diagonalizes $\mathrm{Cov}\{\ \underline{w}(n+1)\ \underline{w}^{\ddagger}(n+1)\ \}$, and if $\underline{Q}$ also diagonalizes $\mathrm{Cov}\{\underline{w}(0)\underline{w}^{\ddagger}(0)\}$ it diagonalizes $\mathrm{Cov}\{\ \underline{w}(n)\underline{w}^{\ddagger}(n)\}$ for all n.

APPENDIX E

In this appendix, we derive an analytical expression of MSE using the time domain LMS algorithm with non-equal power sinusoidal signals in the ALE. From (2-32a), the complex LMS algorithm is given by

$$\underline{w}(n+1) = \underline{w}(n) + 2\,\mu\, e(n)\, \underline{x}^*(n) \qquad (2\text{-}32a)$$

Proceeding in the similar way as we did for the normalized K-L TDLMS algorithm, the mean weight vector and the weight covariance matrix can be expressed as

$$\underline{M}_{w(n+1)} = [\, I - 2\mu\, \underline{R}\,]\, \underline{M}_{w(n)} + 2\mu\, \underline{p} \qquad (E\text{-}1)$$

and

$$\begin{aligned} &\mathrm{Cov}\{\underline{w}(n+1)\underline{w}^{\ddagger}(n+1)\} \\ &= [I - 2\mu\underline{R}]\, \mathrm{Cov}\{\underline{w}(n)\underline{w}^{\ddagger}(n)\} + 4\mu^2 \underline{R}\, \mathrm{Cov}\{\underline{w}(n)\underline{w}^{\ddagger}(n)\}\, \underline{R} \\ &\quad + 4\,\mu^2\, [\, \mathrm{tr}\{\, \underline{R}\, \mathrm{Cov}\{\, \underline{w}(n)\underline{w}^{\ddagger}(n)\}\} - \underline{p}^{\ddagger}\underline{M}_{w(n)} \\ &\quad - \underline{M}^{\ddagger}_{w(n)}\underline{p} + E[d(n)d^*(n)]\,]\, \underline{R} \end{aligned} \qquad (E\text{-}2)$$

Because matrix $\underline{Q}$ diagonalizes $\mathrm{Cov}\{\, \underline{w}(n)\underline{w}^{\ddagger}(n)\}$ for all n [42], therefore, similar to (4-59), (E-2) can be simplified and expressed equivalently as follows:

$$\begin{aligned} \underline{r}(n+1) &= (I - 4\mu\underline{R} + 4\mu^2\underline{R}^2 + 4\mu^2\underline{\lambda}\, \underline{\lambda}^T)\, \underline{r}(n) \\ &\quad + 4\,\mu^2\, k_0(n)\, \underline{\lambda} \end{aligned} \qquad (E\text{-}3)$$

Again, $k_0(n)$ is defined as

$$\begin{aligned} k_0(n) &= \underline{M}^{*}_{w(n)}\, \underline{R}\, \underline{M}_{w(n)} - \underline{p}^{*}\, \underline{M}_{w(n)} \\ &\quad - \underline{M}^{*}_{w(n)}\, \underline{p} + E[\, d(n)d^*(n)] \end{aligned}$$

$$= \xi_{min} + [\underline{M}_{w(n)} - \underline{w}_{opt}]^{\ddagger} \underline{R} [\underline{M}_{w(n)} - \underline{w}_{opt} \qquad \text{(E-4)}$$

Now, for non-equal power case with L=2, (E-3) can be written with reduced rank as:

$$\underline{r}(n+1) = \begin{bmatrix} r_1(n+1) \\ r_2(n+1) \\ r_3(n+1) \end{bmatrix}$$

$$= (\underline{I} - \underline{H})\underline{r}(n) + 4\mu^2 k_0(n)\, \underline{\lambda} \qquad \text{(E-5)}$$

where

$$\underline{H} = \begin{bmatrix} 4\mu\lambda_1(1-\mu\lambda_1) & -4\mu^2\lambda_1\lambda_2 & 4\mu^2\, N\text{-}2\, \lambda_1\lambda_3 \\ -4\mu^2\lambda_2\lambda_1 & 4\mu\lambda_2(1-\mu\lambda_2) & 4\mu^2\, N\text{-}2\, \lambda_2\lambda_3 \\ -4\mu^2\lambda_3\lambda_1 & -4\mu^2\lambda_3\lambda_2 & 4u\lambda_3[1-\mu(N-1)\lambda_3] \end{bmatrix} \qquad \text{(E-6)}$$

and $\underline{\lambda}$ was defined in (5-39a). Noting that $\underline{H}$ is not a symmetric matrix, similarly, we can find a similarity transformation for $\underline{H}$:

$$\underline{T} = \text{diag}\,(\ 1 \quad 1 \quad \sqrt{N-2}\ ) \qquad \text{(E-7)}$$

such that $\underline{H}$ can be transformed to be as

$$\underline{A} = \underline{T}\,\underline{H}\,\underline{T}^{-1}$$

$$= \begin{bmatrix} 4\mu\lambda_1(1-\mu\lambda_1) & -4\mu^2\lambda_1\lambda_2 & 4\mu^2\sqrt{N-2}\ \lambda_1\lambda_3 \\ -4\mu^2\lambda_2\lambda_1 & 4\mu\lambda_2(1-\mu\lambda_2) & 4\mu^2\sqrt{N-2}\ \lambda_2\lambda_3 \\ -4\mu^2\lambda_3\lambda_1\sqrt{N-2} & -4\mu^2\lambda_3\lambda_2\sqrt{N-2} & 4u\lambda_3[1-\mu(N-1)\lambda_3] \end{bmatrix} \qquad \text{(E-8)}$$

Define a new vector, $\underline{c}(n+1)$:

$$\underline{c}(n+1) = \underline{T}\,\underline{r}(n+1)$$

$$= (I-A)\,\underline{c}(n) + 4\mu^2 k_0(n)\,\underline{B} \qquad \text{(E-9)}$$

where

$$\underline{B} = \underline{T}\,\underline{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_{3/N-2} \end{bmatrix} \qquad \text{(E-10)}$$

Now, since $\underline{A}$ is symmetric, it can be factorized as

$$\underline{A} = \underline{S}^T\,\underline{\alpha}\,\underline{S} \qquad \text{(E-11)}$$

where $\underline{S}$ and $\underline{\alpha}$ were defined in (5-48) and (5-49) respectively. Again, starting with $\underline{c}(0)$, and iteratively substituting in (E-9), we have

$$\underline{c}(n) = (\underline{I}-\underline{A})^n\underline{c}(0) + 4\mu^2\,\xi_{min}\sum_{i=0}^{n-1}(\underline{I}-\underline{A})^i\underline{B}$$

$$+ 4\mu^2\sum_{i=0}^{n-1}\frac{N\,\sigma^4_{s_1}}{\lambda_1}(\underline{I}-\underline{A})^{n-1-i}(1-\mu\lambda_1)^{2i}\,\underline{B}$$

$$+ 4\mu^2\sum_{i=0}^{n-1}\frac{N\,\sigma^4_{s_2}}{\lambda_2}(\underline{I}-\underline{A})^{n-1-i}(1-\mu\lambda_2)^{2i}\,\underline{B} \qquad \text{(E-12)}$$

because

$$\underline{M}_{\underline{w}}(n) - \underline{w}_{opt} = \sum_{i=1}^{2}\frac{\sigma^2_{s_i}}{\lambda_i}(1-2\,\mu\lambda_i)^n\,\beta_i\,\underline{v}_i$$

and

$$k_0(n) = \xi_{min} + \sum_{i=1}^{2}\frac{N\,\sigma^4_{s_i}}{\lambda_i}(1-2\mu\lambda_i)^{2n} \qquad \text{(E-13)}$$

Since $\underline{S}$ is a unitary matrix, we have

$$(\underline{I}-\underline{A})^n = \underline{S}\begin{bmatrix} (1-\alpha_1)^n & 0 & 0 \\ 0 & (1-\alpha_2)^n & 0 \\ 0 & 0 & (1-\alpha_3)^n \end{bmatrix}\underline{S}^T \qquad \text{(E-14)}$$

We can also express (E-14) in closed form :

$$\underline{c}(n) = \underline{S}\begin{bmatrix}(1-\alpha_1)^n & 0 & 0\\ 0 & (1-\alpha_2)^n & 0\\ 0 & 0 & (1-\alpha_3)^n\end{bmatrix}\underline{S}^T\underline{c}(0)$$

$$+4\mu^2\xi_{min}\underline{S}\begin{bmatrix}\frac{1-(1-\alpha_1)^n}{\alpha_1} & 0 & 0\\ 0 & \frac{1-(1-\alpha_2)^n}{\alpha_2} & 0\\ 0 & 0 & \frac{1-(1-\alpha_3)^n}{\alpha_3}\end{bmatrix}\underline{S}^T\underline{B}$$

$$+4\mu^2 N\frac{\sigma_{s_1}^4}{\lambda_1}\underline{S}\begin{bmatrix}\frac{(1-2\mu\lambda_1)^{2n}-(1-\alpha_1)^n}{(1-2\mu\lambda_1)^2-(1-\alpha_1)} & 0 & 0\\ 0 & \frac{(1-2\mu\lambda_1)^{2n}-(1-\alpha_2)^n}{(1-2\mu\lambda_1)^2-(1-\alpha_2)} & 0\\ 0 & 0 & \frac{(1-2\mu\lambda_1)^{2n}-(1-\alpha_3)^n}{(1-2\mu\lambda_1)^2-(1-\alpha_3)}\end{bmatrix}\underline{S}^T\underline{B}$$

$$+4\mu^2 N\frac{\sigma_{s_2}^4}{\lambda_2}\underline{S}\begin{bmatrix}\frac{(1-2\mu\lambda_2)^{2n}-(1-\alpha_1)^n}{(1-2\mu\lambda_2)^2-(1-\alpha_1)} & 0 & 0\\ 0 & \frac{(1-2\mu\lambda_2)^{2n}-(1-\alpha_2)^n}{(1-2\mu\lambda_2)^2-(1-\alpha_2)} & 0\\ 0 & 0 & \frac{(1-2\mu\lambda_2)^{2n}-(1-\alpha_3)^n}{(1-2\mu\lambda_2)^2-(1-\alpha_3)}\end{bmatrix}\underline{S}^T\underline{B}$$

$$n = 0, 1, \ldots \qquad \text{(E-15)}$$

The element of $\underline{r}(n)$ can be obtained from (E-9):

$$\underline{r}(n) = \underline{T}^{-1}\ \underline{c}(n) \qquad \text{(E-16)}$$

Finally, since the MSE can be expressed as

$$\begin{aligned}\xi_n &= E[|e(n)|^2] \\ &= \xi_{min} + \{\underline{M}_w(n) - \underline{w}_{opt}\}^{\dagger} \underline{R} \{\underline{M}_w(n) - \underline{w}_{opt}\} \\ &= k_0(n) + \sum_{i=1}^{N} \lambda_i\ r_i(n)\end{aligned} \qquad \text{(E-17)}$$

therefore, from (5-36), (E-13) and (E-16), we can evaluate the MSE in the closed form.

APPENDIX F

In this appendix, a recursive equation for obtaining the DCT of the signals is derived. The DCT of input signal sequence x(n-p), p=0,1,...,N-1 is defined as [57]

$$z_n(k) = p_k\sqrt{\frac{2}{N}} \sum_{p=0}^{N-1} x(n-p) \cos\left(\frac{(2p+1)k\pi}{2N}\right) \qquad k=0,1,\ldots,N-1 \qquad \text{(F-1)}$$

where

$$p_k = \begin{cases} \sqrt{\frac{1}{2}} & , k=0 \text{ or } N \\ 1 & , k\neq 0 \end{cases} \qquad \text{(F-2)}$$

This implies that

$$z_n(0) = \sqrt{\frac{1}{N}} \sum_{p=0}^{N-1} x(n-p) \qquad \text{(F-3a)}$$

and

$$z_n(k) = \sqrt{\frac{2}{N}} \sum_{p=0}^{N-1} x(n-p) \cos\left(\frac{(2p+1)k\pi}{2N}\right) \qquad k=1,2,\ldots,N-1 \qquad \text{(F-3b)}$$

Since DCT can be implemented by using the 2N-point FFT algorithm [57], we can write (F-3) as follows:

$$z_n(0) = \sqrt{\frac{1}{N}} \sum_{p=0}^{2N-1} \bar{x}(n-p) \qquad \text{(F-4a)}$$

and

$$z_n(k) = \sqrt{\frac{2}{N}}\, \mathrm{Re}\left\{ e^{-j(\pi k/2N)} \sum_{p=0}^{2N-1} \bar{x}(n-p)\, e^{-j\frac{2kp\pi}{2N}} \right\} \qquad k=1,2,\ldots,N-1 \qquad \text{(F-4b)}$$

where Re{.} is denoted as the real part of the bracket quantity and x(n-p) is defined by

$$\bar{x}(n-p) = \begin{cases} x(n-p), & p=0,1,\ldots,N-1 \\ 0, & p=N,N+1,\ldots,2N-1 \end{cases} \qquad \text{(F-5)}$$

Now, if we define $z_n(k)$ as

$$\bar{z}_n(k) = \sqrt{\frac{2}{N}} \sum_{p=0}^{2N-1} x(n-p)\, e^{-j\frac{2kp\,\pi}{2N}} \qquad k=0,1,\ldots,N-1 \qquad \text{(F-6)}$$

then (F-4) can be written as

$$z_n(k) = p_k \, \mathrm{Re}\{ e^{-j(\pi k/2N)} \bar{z}_n(k) \} \qquad k=0,1,\ldots N-1 \qquad \text{(F-7)}$$

To obtain the recursive equation, let us define $z^{Im}{}_n(k)$, as follows:

$$z^{Im}{}_n(k) = p_k \, \mathrm{Im}\{ e^{-j(\pi k/2N)} \bar{z}_n(k) \qquad k=0,1,\ldots,N-1 \qquad \text{(F-8)}$$

where Im{.} denotes as the imaginary part of the bracket quantity. Proceeding in the similar way as we obtained (6-12), the recursive equation of $z_n(k)$ can be shown to be as

$$\bar{z}_n(k) = \bar{z}_{n-1}(k)\, e^{-j(\pi k/2N)} + \sqrt{\frac{2}{N}} [\, x(n) - (-1)^k x(n-N)] \qquad k=0,1,\ldots,N-1 \qquad \text{(F-9)}$$

Furthermore, rewrite (F-6) in the complex form as

$$\bar{z}_n(k) = \mathrm{Re}\{\bar{z}_n(k)\} + j\, \mathrm{Im}\{\bar{z}_n(k)\} \qquad k=0,1,\ldots,N-1 \qquad \text{(F-10)}$$

Consequently, we can express (F-7) and (F-8) as

$$z_n(k) = p_k[\mathrm{Re}\{\bar{z}_n(k)\}\cos(\frac{\pi k}{2N}) + \mathrm{Im}\{\bar{z}_n(k)\}\sin(\frac{\pi k}{2N})] \qquad k=0,1,\ldots,N-1 \qquad \text{(F-11)}$$

and

$$z^{Im}{}_n(k) = p_k \{- \mathrm{Re}\{\bar{z}_n(k)\}\sin(\frac{\pi k}{2N}) + \mathrm{Im}\{\bar{z}_n(k)\} \cdot \cos(\frac{\pi k}{2N}), \quad k=0,1,\ldots,N-1 \qquad \text{(F-12)}$$

respectively. Substituting (F-9) into (F-11) and (F-12), we have

$$z_n(k) = p_k \, \mathrm{Re}\{ e^{-j(\pi k/2N)} \{ \bar{z}_{n-1}(k) \, e^{-j(2\pi k/2N)} + \sqrt{\frac{2}{N}} [x(n) - (-1)^k x(n-N)] \} \}$$

$$= p_k \{ [\mathrm{Re}\{ e^{-j(3\pi k)/2N} \bar{z}_{n-1}(k)\}] + \sqrt{\frac{2}{N}} [x(n) - (-1)^k x(n-N)] \cos(\frac{\pi k}{2N}) \}$$

$$k=0,1,\ldots,N-1 \qquad \text{(F-13)}$$

because that $x(n-p)$, $p=0,1,\ldots,N-1$ are real numbers. The first term of (F-13) can be expanded as

$$\mathrm{Re}\{ \bar{z}_{n-1}(k) \, e^{-j(3\pi k)/2N} \}$$

$$= \mathrm{Re}\{ \bar{z}_{n-1}(k) \} \cos(\frac{3\pi k}{2N}) + \mathrm{Im}\{\bar{z}_{n-1}(k)\} \sin(\frac{3\pi k}{2N})$$

$$= \frac{1}{p_k} \{ \cos(\frac{\pi k}{N}) \, z_{n-1}(k) + \sin(\frac{\pi k}{N}) \, z^{\mathrm{Im}}_{n-1}(k) \}$$

$$k=0,1, \ldots, N-1 \qquad \text{(F-14)}$$

where two identities

$$\cos(a_1 + a_2) = \cos(a_1)\cos(a_2) - \sin(a_1)\sin(a_2)$$

and

$$\sin(a_1+a_2) = \sin(a_1)\cos(a_2 + \cos(a_1)\sin(a_2)$$

have been used. Proceeding in the similar way as we obtained (F-14), we have

$$\mathrm{Im}\{ \bar{z}_{n-1}(k) \, e^{-j(3\pi k)/2N} \}$$

$$= \frac{1}{p_k} \{ \cos(\frac{\pi k}{N}) \, z^{\mathrm{Im}}_{n-1}(k) - \sin(\frac{\pi k}{N}) \, z_{n-1}(k) \}$$

$$k=0,1,\ldots,N-1 \qquad \text{(F-15)}$$

Finally, (F-14) can be expressed as

$$z_n(k) = \cos(\frac{\pi k}{N}) \, z_{n-1}(k) + \sin(\frac{\pi k}{N}) \, z^{\mathrm{Im}}_{n-1}(k) + \sqrt{\frac{2}{N}} \, p_k \{ x(N) - (-1)^k x(n-N) \} \cos(\frac{\pi k}{2N})$$

$$k=0,1,\ldots,N-1 \qquad \text{(F-16a)}$$

Similarly, by using (F-15) and substituting (F-9) into (F-8), (F-8) can be expanded as the following:

$$z^{Im}_{n,k} = p_k \, Im\{ e^{-j(-k)/2N} \bar{z}_n(k)\}$$
$$= \cos(\frac{\pi k}{N}) \, z^{Im}_{n-1}(k) - \sin(\frac{\pi k}{N}) \, z_{n-1}(k)$$
$$-\sqrt{\frac{2}{N}} \, p_k\{x(n) - (-1)^k \, x(n-N)\} \sin(\frac{\pi k}{2N})$$
$$k=0,1,\ldots,N-1 \qquad \text{(F-16b)}$$

Based on (F-7), (F-8) and (F-16), the recursive equation of DCT can be performed.

APPENDIX G

In this appendix, we show that the lower triangular matrix, $\underline{Q}_L$ is not a unitary matrix. From (6-64), we have

$$\underline{Q}_L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -a_{03} & 1 & 0 & 0 \\ -a_{02} & -a_{12} & 1 & 0 \\ -a_{01} & -a_{11} & -a_{21} & 1 \end{bmatrix} \qquad (G\text{-}1)$$

To prove this, we first assume that $\underline{Q}_L$ is a unitary matrix and then show that it is contradictory. By definition, if $\underline{Q}_L$ is an unitary matrix, we have from (G-1)

$$\underline{Q}_L \, \underline{Q}^T_L = \underline{I}_{4x4} \qquad (G\text{-}2)$$

(G-2) implies that all the off-diagonal components are zeros and the diagonal components are unity, therefore, we have the following conditions:

$$a_{01} = a_{02} = a_{03} = a_{11} = a_{12} = a_{21} = 0 \qquad (G\text{-}3)$$

Substituting (G-3) into (G-1), we have

$$\underline{Q}_L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \underline{I}_{4x4} \qquad \text{(G-4)}$$

This is controdictory because $\underline{Q}_L$ is not an identity matrix. Similarly, this approach can be extended to NxN matrix and have the following results:

$$a_{m,n} = 0, \quad m=0,1,\ldots, N-2 \text{ and } n= 1,2,\ldots,N-1-m \qquad \text{(G-5)}$$

That is the off-diagonal elements are all zeros. Again, we can conclude that $\underline{Q}_L$ is not a unitary matrix.

APPENDIX H

In this appendix, we present the evaluations of $\Theta_k(j,n)$, $\Phi_k(i,n)$ and $\Psi_k(\ell,n)$. These expressions are then used to obtain the mean weight value. Using (7-13a), we have,

$$E[\Theta_k(j,n)]$$

$$= E[\mu_r \hat{d}_j(k)\hat{s}^*_j(k)\{1-\mu_r|\hat{s}_j(k)|^2\}^n]$$

$$= \mu_r E[\hat{d}_j(k)\hat{s}^*_j(k)\{\sum_{\ell=0}^{n}\binom{n}{\ell}(-1)^{\ell}\mu_r^{\ell}|\hat{s}_j(k)|^{2\ell}\}]$$

$$= \sum_{\ell=0}^{n}\binom{n}{\ell}(-1)^{\ell}\mu_r^{\ell+1} E[\hat{d}_j(k)\hat{s}^*_j(k)\{\hat{s}_j\hat{s}^*_j(k)\}^{\ell}] \qquad (H\text{-}1)$$

To find the mean value of the terms inside the brackets in (H-1) we use the result of the moment theorem for a complex, zero-mean, circular Gaussian process [31] which states that:

If z_n are samples from a zero-mean, complex circular Gaussian process, then

$$E[z_{m1}z_{m2}\cdots z_{mi}z^*_{n1}z^*_{n2}\cdots z^*_{ni}]$$
$$= \sum_{\pi} E[z_{m\pi(1)}z^*_{n1}]\,E[z_{m\pi(2)}z^*_{n2}]\cdots E[z_{m\pi(i)}z_{ni}] \qquad (H\text{-}2)$$

where π is a permutation of the set of integers $\{1,2,\ldots,i\}$. If we let

$$z_{m1} = \hat{d}_j(k)$$

$$z_{m2} = z_{m3} = \cdots = z_{m,i+1} = \hat{s}_j(k)$$

and

$$z^*_{n1} = z^*_{n2} = \cdots = z^*_{n,i+1} = \hat{s}^*_j(k)$$

then, using (H-2) and noting that there are (i+1)! permutations among the i+1 random variables, we have

$$E[\hat{d}_j(k)\hat{s}^*_j(k)\{\hat{s}_j(k)\,\hat{s}^*_j(k)\}] = (i+1)!\,E[\hat{d}_j(k)\hat{s}^*_j(k)]\,E[\{\hat{s}_j(k)\hat{s}^*_j(k)\}^i] \tag{H-3}$$

But from (7-1) - (7 3), we have

$$E[\hat{d}_j(k)\,\hat{s}^*_j(k)] = a(k)\,\sigma^2_{sk}$$
$$E[\hat{s}_j(k)\hat{s}^*_j(k)] = \sigma^2_{sk} + \sigma^2_{nk}, \tag{H-4}$$

Therefore, substituting (H-3) and (H-4) in (H-1), we obtain,

$$E[\Theta_k(j,n)] = \sum_{\ell=0}^{n}\binom{n}{\ell}(-1)^\ell \mu_r^{\ell+1}(\ell+1)!\,a(k)\,\sigma^2_{sk}\{\sigma^2_{sk}+\sigma^2_{nk}\}^\ell$$
$$\simeq \mu_r\,a(k)\,\{1-2n\mu_r[\sigma^2_{sk}+\sigma^2_{nk}]\}\,\sigma^2_{sk} \tag{H-5}$$

where we have made the assumption that $\mu_r(\sigma^2_{sk}+\sigma^2_{nk}) \ll 1$. Proceeding in similar way, we can obtain,

$$E[\phi_k(i,n)] = E[\{1-\mu_r|\hat{s}_j(k)|^2\}^n] = \sum_{\ell=0}^{n}\binom{n}{\ell}(-1)^\ell \mu_r^\ell\,E[|\hat{s}_j(k)|^{2\ell}]$$
$$= \sum_{\ell=0}^{n}\binom{n}{\ell}(-\mu_r)^\ell\,\ell!\,(\sigma^2_{sk}+\sigma^2_{nk})$$
$$\simeq 1 - n\mu_r(\sigma^2_{sk}+\sigma^2_{nk}) \tag{H-6}$$

and

$$E[\ \Psi_k(\ell,n)] = E[\ \{\ 1-\mu_r|\hat{s}_\ell(k)|^2\}^{n+1}\ ]$$
$$\simeq\ 1\ -\ (n+1)\ \mu_r(\ \sigma^2_{sk}\ +\ \sigma^2_{nk}\ ) \qquad (H\text{-}7)$$

To evaluate the mean value of the tap-weight at the end of the re-circulation of input data we substitute (H 5)-(H-7) into (7-14), to obtain

$$E[\ b_k(p,M)]$$

$$= \sum_{n=0}^{p}\sum_{j=0}^{M-1}\{\mu_r\ a(k)\ \sigma^2_{sk}[1\ -\ 2n\mu_r(\ \sigma^2_{sk}\ +\ \sigma^2_{nk})]$$

$$\prod_{i=0}^{j-1}\{\ 1-n\mu_r\ (\ \sigma^2_{sk}\ +\ \sigma^2_{nk})\}\prod_{\ell=j+1}^{M-1}[\ 1\ -\ (n+1)\ \mu_r\ (\ \sigma^2_{sk}\ +\ \sigma^2_{nk})\ \}\ \}$$

$$= \sum_{n=0}^{p}\mu_r\ a(k)\ \sigma^2_{sk}\ [\ 1-2n\mu_r\ (\sigma^2_{sk}\ +\ \sigma^2_{nk})\ ]\ \sum_{j=0}^{M-1}$$

$$\{\ 1\ -\ n\mu_r(\ \sigma^2_{sk}\ +\ \sigma^2_{nk})\ \}^j\ \{\ 1\ -\ (n+1)\mu_r\ (\ \sigma^2_{sk}\ +\sigma^2_{nk}\ )\ \}^{M-1-j}$$

$$= \sum_{n=0}^{p}\mu_r\ a(k)\ \sigma^2_{sk}\ [\ 1\ -\ 2n\mu_r(\ \sigma^2_{sk}\ +\ \sigma^2_{nk})\ ]$$

$$\cdot\{\ 1\ -\ n\mu_r(\ \sigma^2_{sk}\ +\ \sigma^2_{nk})\ \}^{M-1}\sum_{j=0}^{M-1}\left[\frac{1-(n+1)\mu_r\ (\sigma^2_{sk}+\ \sigma^2_{nk})}{1\ -\ n\ \mu_r(\ \sigma^2_{sk}+\ \sigma^2_{nk})}\right]^{M-1-j}$$

$$= \sum_{n=0}^{p}\mu_r\ a(k)\ \sigma^2_{sk}\ [\ 1\ -\ 2n\mu_r(\ \sigma^2_{sk}\ +\ \sigma^2_{nk})\ ]$$

$$\cdot \left[1 - n\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)\right]^{M-1} \left\{ \frac{1 - \left[\frac{1-(n+1)\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)}{}\right]^M}{1 - n\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)} \Big/ \frac{1 - \left[\frac{1-(n+1)\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)}{}\right]}{1 - n\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)} \right\}$$

i.e.,

$$E[b_k(p,M)] = \sum_{n=0}^{p} \frac{a(k)\sigma^2_{sk}}{\left(\sigma^2_{sk} + \sigma^2_{nk}\right)} \left[1 - 2n\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)\right]$$

$$\cdot \left\{\left[1 - n\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)\right]^M - \left[1 - (n+1)\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)\right]^M\right\} \qquad \text{(H-8)}$$

Now, since

$$n\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right) << 1$$

Therefore,

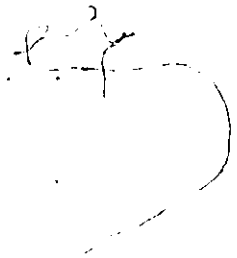
$$E[b_k(p,M)] \simeq \frac{a(k)\sigma^2_{sk}}{\left(\sigma^2_{sk} + \sigma^2_{nk}\right)} \left[1 - n\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)\right]^2 \cdot \left\{\left[1 - n\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)\right]^M - \left[1 - (n+1)\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)\right]^M\right\}$$

$$\simeq \frac{a(k)\sigma^2_{sk}}{\left(\sigma^2_{sk} + \sigma^2_{nk}\right)} \left\{\left[1 - n\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)\right]^{M+2} - \left[1 - (n+1)\mu_r\left(\sigma^2_{sk} + \sigma^2_{nk}\right)\right]^{M+2}\right\}$$

$$= \frac{a(k)\sigma^2_{sk}}{\left(\sigma^2_{sk} + \sigma^2_{nk}\right)} \left\{1 - \left[1 - (p+1)\tilde{\mu}_k\right]^{M+2}\right\} \qquad \text{(H-9)}$$

Note that if $p = 0$, $E[b_k(p,M)]$ can be evaluated from (H-8) and the last step of approximation in (H-9) can be bypassed yielding

$$E[b_k(p,M)] \simeq \frac{a(k)\,\sigma_{sk}^2}{(\sigma_{sk}^2 + \sigma_{nk}^2)}[\,1 - (\,1 - \tilde{\mu}_k\,)^M] \qquad \text{(H-10)}$$



REFERENCES

1. Applied Optimal Estimation, written by the Technical staff, the analytic Sciences corporation, edited by Arthur Gelb. The M.I.T. Press, Cambridge, M.A. 1974.

2. Van Trees, H. L., Detection, Estimation, and Modulation Theory,part I., Detection Estimation and Linear Modulation Theory, John Wiely and Sons, Inc., 1968.

3. Brian D.O. Anderson and John B. Moore, Optimal Filtering, Prentice-Hall, 1979.

4. Gerald J. Bierman, Factorization Methods for Discrete Sequential Estimation, Academic Press, New York 1977.

5. A. N. Kolmogorov, " Interpolation and Extrapolation of Stationary Random Sequences ", (First published 1941 in the USSR Science Academic Bulletin). Translated by W. Doyle and J. Selin, RM-3090-PR, 1962, Rand Corp., Santa Monica Calif.

6. N. Wiener, " The Extrapolation, Interpolation, and Smoothing of Stationary Time Series", OSRD 370, Report to the services 19, Research Project DIC-6037, M.I.T., Feb. 1942.

7. R.A. Monzingo and T.W. Miller, Introduction to Adaptive Arrays, Wiely, New York, 1980.

8. B. Widrow et al., " Adaptive Noise Cancelling: Principles and Applications", Proc. IEEE, vol.63, no.12, Dec. 1975.

9. P.L. Feintuch, N.J. Bershad, and F.A. Reed, " Time Delay Estimation Using the LMS Adaptive Filter - Dynamic Behavior", IEEE Trans. on Acoustic, Speech, Signal Processing, vol. ASSP- 29, no. 3, June, 1981

10. M.M. Sondhi and A.J. Presti, " A Self-adaptive Echo Canceller", BSTJ, vol.46, no.3, March, 1967.

11. J.R. Zeidler et al., " Adaptive Enhancement of Multiple Sinusoids in Uncorrelated Noise", IEEE Trans. on Acoust., Speech, Signal Processing, vol. ASSP-26, June, 1978.

12. N.K. Sinha and B. Kuszta, Modelling and Identification of Dynamic Systems, Van Nostrand Reinhold, 1983.

13. R.W. Lucky, " Automatic Equalization for Digital Communication", BSTJ, vol.44, April,1965.

14. B. Widrow, " Adaptive Filters," in Aspects of Network and System Theory, R. Kalman and N. DeClaris, Eds., Holt.

15. Kantorovich, L.V., Functional Analysis in Normed Spaces, Translated from Russian by D.E. Brown, M.A. edited by Dr. A. P. Robertson, Pergamon press LTD. London, 1964.

16. G. A. Clark, S. K. Mitra, and S. R. Parker, " Block Implementation of Adaptive Digital Filters," IEEE Trans. on Acoust., Speech, Signal Processing, vol. ASSP-29, pp. 744-752, June 1981.

17. G.A. Clark, S.R. Parker, and S.K. Mitra, " A Unified Approach to Time- and Frequency-domain Realization of FIR Adaptive Digital Filters," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-31, pp. 1073-1083, Oct., 1983.

18. J.C. Lee, B.K. Min, and M. Suk, " Realization of Adaptive Digital Filetrs Using the Fermat Number Transform," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-33 ,No. 4, August 1985.

19. E.R. Ferrara, " Fast Implementation of LMS Adaptive Filters," IEEE Trans. on Acoust., Speech, Signal Processing, vol. ASSP-28, pp. 474-475, August, 1980.

20. M. Dentino, J. McCool and B. Widrow, " Adaptive Filtering in the Frequency Domain," Proc. IEEE, pp. 1658-1659, Dec. 1979.

21. D. Mansour, and A.H. Gray, " Unconstrainted Frequency Domain Adaptive Filter, " IEEE Trans. on Acoust., Speech, Signal Processing, vol. ASSP-30, No. 5, Oct. 1982.

22. J. C. Ogue, T. Saito, and Y. Hashiko, " A Fast Convergence Frequency Domain Adaptive Filter,"IEEE Trans. on Acoust., Speech, and Signal Processing, vol. ASSP-31,

No. 5, Oct., 1983.

23. S. S. Narayan, A. M. Peterson, and M.J. Narasimha, " Transform Domain LMS Algorithm," IEEE Trans. on Acoust., Speech, Signal Processing, vol. ASSP-31, no.3, June, 1983

24. Adaptive Filter, Edited by C.F.N. Cowan and P.M. Grant, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

25. G.C. Goodwin and K.S. Sin, Adaptive Filtering, Prediction and Control, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.

26. L. Ljung and T. Soderstorm, Theory and Practice of Recursive Identification, MIT press, Cambridge, 1983.

27. K.M. Wong and Y.G. Jan, " Adaptive Walsh Equalizer for Data Transmission," IEE Proc., vol. 130, Pt. F, No. 2, March 1983.

28. F. A. Reed and P. L. Feintuch, " A Comparison of LMS Adaptive Cancellers Implemented in the Frequency Domain and Time Domain," IEEE Trans. on Acoust., Speech, Signal Processing, vol. ASSP-29,pp. 770-775, June, 1981.

29. J. Peral, " Basis - Restricted Transformations and Performance Measures for Spectral Representations," IEEE Trans. on Informaton Theory, Nov., 1971.

30. W.D. Ray and R.M. Driver, " Further Decomposition of the Karhunen-Loève Series Representation of a Stationary Random Process," IEEE Trans. on Information Theory, Nov., 1970.

31. I.S. Reed, " On a Moment Theory for Complex Gaussian Processes," IRE Trans. on Information Theory, pp.194-196 August 1962.

32. L. Ljung, " Recursive Identification Methods for Off-line Identification Problems ," 6th IFAC Symposium on Identification and System Parameter Estimation, 1982.

33. P.C. Young, " Some Observations on Instrumental Variable Methods of Time Series Analysis," International Journal of Control, vol. 23, 1976, pp.593-612.

34. Magnus Hestenes, Conjugate Direction Methods in Optimization, Springer-Verlag, New York, 1980.

35. C. Nelson Dorny, A Vector Space Approach to Models and Optimization, John Wiely & Sons, Inc., 1975.

36. P. Yip and K.R. Rao, " Fast Decimation-in-Time Algorithms for a Family of Discrete Sine and Cosine Transforms," Circuits Systems Signal Process vol. 3, No.4, 1984.

37. Fletcher, R., " A Review of Methods for Unconstrained Optimization in Optimization , " (R. Fletcher, ed.), pp. 1-12, Academic press , New York,1969.

38. Davidon, W.C., " Variable Metric Method for Minimization ," A.E.C. Res. Dev. Rep. ANL-5990 (rev.), 1959.

39. A. Feuer and E. Weinstein, " Convergence Analysis of LMS Filters with Uncorrelated Gaussian Data," IEEE Trans. on Acoust., Speech, Signal Processing, vol. ASSP-33, No.1, Feb., 1985.

40. B. Widrow, J. McCool and M. Ball, " The Complex LMS Algorithm," Proc. IEEE, vol. 63, pp. 719-720, April, 1975

41. L.L. Horowitz and K.D. Senne, " Performance Advantage of Complex LMS for Controlling Narrowband Adaptive Arrays," IEEE Trans. on Circuits and Systems, vol. CAS-28, No.6, June, 1981.

42. B. Fisher and N.J. Bershad, " The Complex LMS Adaptive Algorithm - Transient Weight Mean and Covariance with Applications to the ALE," IEEE Trans. on Acoust., Speech, Signal Processing, vol. ASSP-31, no.1, Feb., 1983.

43. A.S. Deif, Advanced Matrix Theory for Scientist," Nov. 1981.

44. K. Karhunen, " On Linear Methods in Probability Theory, " I. Selin, transl. , RAND Corporation T-131, August, 1960

45. M. Loève, " Probability Theory ," 2nd ed. Princeton, N.J. Van Nostrand, 1960, pp.478.

46. Y. Yemini and J. Pearl, " Asymptotic Properties of Discrete Unitary Transforms," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no.4,

Oct.,1979.

47. J.B. Thomas, An Introduction to Statistical Communication Theory, John Wiely & Sons, Inc. New York, 1969.

48. G. Zielke, " Inversion of Modified Symmetric Matrices," J. Ass. Comput. July, vol. 15, pp. 402-408, 1968.

49. A. Papoulis, Probability, Random Variables and Stochastic Processes, McGraw-Hill Inc. New York 1965.

50. J.R. Richard, J.R. Zeidler, M.J. Dentino and M. Shensa, " A Performance Analysis of Adaptive Line Enhancer - Augmented Spectral Detectors," IEEE Trans. vol. ASSP-29, No. 3, June 1981

51. J.R. Richard and J.R. Zeidler, " Second-Order Output Statistics of the Adaptive Line Enhancer," IEEE Trans. vol. ASSP-27, No. 1, Feb. 1979.

52. J.R. Treichler, " Transient and Convergent Behavior of the Adaptive Line Enhancer," IEEE Trans. vol. ASSP-27, No.1, Feb. 1979.

53. M.C. Pease, III., Method of Matrix Algebra, Academic Press, New York 1965.

54. D.F. Elliott and K.R. Rao, Fast Transforms Algorithms, Analyses, Applications, Academic Press, New York 1982.

55. H.C. Andrews, Picture Processing and Digital Filtering, Chapter 2, Edited by T. S. Huang, Springer- Verlag, New York, 1975.

56. A. K. Jain, " A Sinusoidal Family of Unitary Transforms," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, No. 4, Oct., 1979.

57. N. Ahmed, T. Natarajan and K. R. Rao, " Discrete Cosine Transform," IEEE Trans. on Comput., pp. 90-93, Jan. 1974.

58. A. K. Jain, " A Fast Karhunen-Loève Transform for a Class of Random Processes," IEEE Trans. on Communications, Sept. 1976.

1009, 1979.

62. Z. Wong, " Fast Algorithms for the Discrete W Transform and for the Discrete Fourier Transform, IEEE Trans. vol. ASSP-32, No. 4, August 1984.

63. J. I. Makhoul and L. Cosell, " Adaptive Lattice Analysis of Speech, " IEEE Trans. vol. ASSP-29, No.3, June 1981.

64. C. W. K. Gritton and D.W. Lin, " Echo Cancellation Algorithms," IEEE Trans. vol. ASSP Magazine April 1984.

65. B. Friedlander, " Lattice Methods for Spectral Estimation, " Proceeding IEEE, vol. 70, No. 9, pp. 990 - 1017, September 1982.

66. C. J. Gibson and S. Haykin, " Learning Characteristics of Adaptive Lattice Filtering Algorithms, IEEE Trans. vol. ASSP-28 No. 6, December 1980.

67. J. Makhoul, " Stable and Efficient Lattice Methods for Linear Prediction," IEEE Trans. vol. ASSP-25, No. 5, pp. 423-428, May 1977.

68. B. Friedlander and M. Morf, " Least-Squares Algorithms for Adaptive Linear-pahse Filtering," IEEE Trans. vol. ASSP-30, No. 3, pp. 381-390, June 1982.

69. C.F.N. Cowan and P. M. Grant, Adaptive Filters, Prentice-Hall, Inc., New York 1985.

70. S. Haykin, Adaptive Filter Theory, Prentice-Hall, Cliff, New Jersey 1986.

71. S.L. Freeny, R.B. Kieburtz, K.V. Mina and S.K. Tewksbury, " Design of Digital Filters for an All Digital Frequency Division Multiplex - Time Division Multiplex Translator," IEEE Trans. Circuit Theory, vol. CT-18, No. 6, pp. 702-711, November 1971.

72. Special Issue on TDM-FDM Conversion, IEEE Trans. on Communications, vol. COM-26, No. 5, May 1978.

73. A.C.M. Claasen and W.F.G. Mechlenbrauker, " A Generalized Scheme for an All Digital Time Division Multiplex to

Frequency Division Multiplex Translator, " IEEE Trans. on Circuits and Systems, vol. CAS-25, No.25, May 1978.

74. M. Tomlinson and K. M. Wong, " Techniques for the Digital Interfacing of TDM-FDM Systems," Proc. IEE, vol. 123, No. 12, December 1976.

75. K. M. Wong, V. K. Aatre and N. Ramamurthy, " Properties and Applications of Some Periodically Varying Digital Signal Processors, " IEE Proc. vol. 128, Pt. F. no. 2, April 1981.

76. E. Crochiere and L. R. Rabiner, Multirate Digital Signal Processing, Prentice-Hall, Inc., New Jersey 1983.

77. E.R Ferrara, " Frequency-Domain Implementations of Periodically Time-varying Filters," IEEE Trans. vol. ASSP-33, No. 4 August 1985.

78. K. M. Wong, " An Orthogonal Expansion for Bandlimited Stochastic Processes with Applications to Adaptive Filtering, " Research Memo, McMaster University March 1985.

79. S. J. Chern and K. M. Wong, " Re-circulation of Input Data in Frequency-domain Adaptive Filtering, " Submitted to the Canadian Electrical Engineering Journal.